Towards Context-aware Mobile Web 2.0 Augmented Reality

by

Rahim P. Khajei

MSc., Azad Qazvin University, 2011

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

September 2017

© Rahim P. Khajei, 2017
Abstract

Augmented reality (AR) is a Context-aware services service which allows users to have an enhanced perception of the real world through a composition of virtual and actual objects. In recent years, AR has received tremendous attention from both academic and industry sectors. However, developers and end users are still suffering from lack of standard formats and protocols. We believe the obstacles stopping AR from flourishing are partially inherited from context-aware services and partially stem from the architecture of the current AR applications. Here, we aimed to develop a new model that can support AR framework for sharing Content between AR applications and communication between AR users. By incorporating Web 2.0 standards in Client-server architecture, we designed a new architecture for AR named Client Federated Servers (CFS). We implemented an AR application named Scratcher as a proof of concept. Scratcher allows users to search and share Targets as well as communicate with each other.
# TABLE OF CONTENTS

Abstract ii

Table of Contents iii

List of Figures vii

List of Tables ix

Acknowledgements xi

Glossary xii

## 1 Introduction

1.1 Overview ........................................ 1

1.2 Motivation ........................................ 5

1.3 Research Problem ................................ 6

1.4 Purpose of Study ................................... 6

1.5 Objectives of Study ............................... 7

1.6 Research Questions ............................... 7

1.7 Contributions .................................... 8

1.8 Thesis Structure ................................ 9

## 2 Background and Literature Review

2.1 Augmented Reality .............................. 10
2.1.1 Brief History of AR ............................................ 12
2.1.2 How AR Works ............................................. 13
2.1.3 Applications ............................................... 16
  2.1.3.1 Annotation ............................................. 16
  2.1.3.2 Medical ................................................ 17
  2.1.3.3 Entertainment ......................................... 17
  2.1.3.4 Military ............................................... 17
2.1.4 AR Research Areas ......................................... 18
2.1.5 AR Enabling Technologies ................................. 19
  2.1.5.1 Tracking ............................................... 19
  2.1.5.2 Interaction and Interface ............................. 21
  2.1.5.3 Display Methods ...................................... 22
2.1.6 Challenges in AR ........................................... 23
2.2 Client-Server Architecture .................................. 26
  2.2.1 AR Architecture ......................................... 27
2.3 Web 2.0 and Social Services ................................. 32
  2.3.1 Web 2.0 .................................................. 32
  2.3.2 Social Networking ....................................... 33
  2.3.3 Social Media ............................................. 34
2.4 Related Work ................................................... 35
  2.4.1 Target and Content Related Issues .................... 39
  2.4.2 Web 2.0 in Augmented Reality ......................... 43
2.5 Chapter Summary ............................................... 49

3 Proposed Framework ............................................. 50
  3.1 Basic Idea .................................................... 50
  3.2 Limitations with Current Structure ....................... 52
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 Client Federated Servers</td>
<td>54</td>
</tr>
<tr>
<td>3.3.1 Description of Components</td>
<td>55</td>
</tr>
<tr>
<td>3.3.2 Benefits of Client Federated Servers Framework</td>
<td>58</td>
</tr>
<tr>
<td>3.3.3 Practical Scenarios</td>
<td>60</td>
</tr>
<tr>
<td>3.4 Data Flow and Connection between Components</td>
<td>62</td>
</tr>
<tr>
<td>3.5 Main Processes</td>
<td>66</td>
</tr>
<tr>
<td>3.5.1 Registering in Target Hub</td>
<td>67</td>
</tr>
<tr>
<td>3.5.2 Sharing Targets and Contents and Subscription</td>
<td>68</td>
</tr>
<tr>
<td>3.5.3 Searching and Loading Targets</td>
<td>69</td>
</tr>
<tr>
<td>3.5.4 Chat Rooms and Communication Handling</td>
<td>70</td>
</tr>
<tr>
<td>3.5.4.1 Subscription and Notification Handling</td>
<td>72</td>
</tr>
<tr>
<td>3.6 Specifications</td>
<td>73</td>
</tr>
<tr>
<td>3.7 Chapter Summary</td>
<td>75</td>
</tr>
<tr>
<td>4 Scratcher - Proof of Concept</td>
<td>77</td>
</tr>
<tr>
<td>4.1 Mobile Application Implementation</td>
<td>78</td>
</tr>
<tr>
<td>4.1.1 How Does It Work?</td>
<td>79</td>
</tr>
<tr>
<td>4.1.2 Chat System</td>
<td>84</td>
</tr>
<tr>
<td>4.1.3 Storing and Retrieving</td>
<td>84</td>
</tr>
<tr>
<td>4.1.4 Sending and Receiving a Message</td>
<td>86</td>
</tr>
<tr>
<td>4.2 Server Side Implementation</td>
<td>88</td>
</tr>
<tr>
<td>4.3 Target Hub Implementation</td>
<td>93</td>
</tr>
<tr>
<td>4.4 Web APIs</td>
<td>98</td>
</tr>
<tr>
<td>4.5 Expiration and Activation Tags</td>
<td>100</td>
</tr>
<tr>
<td>4.5.1 Test Case</td>
<td>102</td>
</tr>
<tr>
<td>4.6 Chapter Summary</td>
<td>102</td>
</tr>
<tr>
<td>5 Conclusion and Future Directions</td>
<td>105</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>Simple AR application</td>
</tr>
<tr>
<td>2.2</td>
<td>Reality-Virtuality (RV) Continuum [59]</td>
</tr>
<tr>
<td>2.3</td>
<td>A simple AR system [72]</td>
</tr>
<tr>
<td>2.4</td>
<td>Components in AR architecture [34]</td>
</tr>
<tr>
<td>2.5</td>
<td>AR in military application [30]</td>
</tr>
<tr>
<td>2.6</td>
<td>A modern HMD - Microsoft’s HoloLens [9]</td>
</tr>
<tr>
<td>2.7</td>
<td>Client Server processing environment</td>
</tr>
<tr>
<td>2.8</td>
<td>AR using Client-Server Architecture</td>
</tr>
<tr>
<td>2.9</td>
<td>Flow of information in client-server framework [71]</td>
</tr>
<tr>
<td>2.10</td>
<td>Context provisioning ecosystem [77]</td>
</tr>
<tr>
<td>2.11</td>
<td>Infrastructure of Cyber-Physical Web [36]</td>
</tr>
<tr>
<td>2.12</td>
<td>System architecture for AR application [28]</td>
</tr>
<tr>
<td>2.13</td>
<td>System architecture for mobile AR application [75]</td>
</tr>
<tr>
<td>2.14</td>
<td>Visualization of the trail marks in AR [39]</td>
</tr>
<tr>
<td>2.15</td>
<td>Visualization in Link2U [25]</td>
</tr>
<tr>
<td>2.16</td>
<td>Vision sharing feature for SOAR proof of concept [63]</td>
</tr>
<tr>
<td>2.17</td>
<td>Bridging augmented reality and augmented virtuality [41]</td>
</tr>
<tr>
<td>3.1</td>
<td>client federated servers architecture for AR applications</td>
</tr>
<tr>
<td>3.2</td>
<td>Searching for target and list of available contents for a target</td>
</tr>
<tr>
<td>3.3</td>
<td>Data flow diagram of the system</td>
</tr>
</tbody>
</table>
4.21 Forwarding a chat message to the subscribers .......... 98
4.22 Only sphere is in the scene .................................. 103
4.23 Both of sphere and cube are in the scene ............... 103
4.24 Sphere expired and disappeared ......................... 104
<table>
<thead>
<tr>
<th></th>
<th>Document of searching for targets in the target hub</th>
<th>98</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Document of downloading a target from the target hub</td>
<td>99</td>
</tr>
<tr>
<td>4.3</td>
<td>Document of uploading a target to the target hub</td>
<td>99</td>
</tr>
<tr>
<td>4.4</td>
<td>Document of message passing in target hub</td>
<td>100</td>
</tr>
<tr>
<td>4.5</td>
<td>Document of registration to target hub</td>
<td>100</td>
</tr>
<tr>
<td>4.6</td>
<td>Document of showing the servers of the target hub</td>
<td>101</td>
</tr>
</tbody>
</table>
Acknowledgements

“My favourite things in life don't cost any money. It's really clear that the most precious resource we all have is time.”
— Steve Jobs

Firstly, I would like to express my sincere gratitude to my wife Mona Aminorroayaee for her continuous support during my studies.

I would like to thank my supervisor Dr. Alex Aravind for his input, guidance, and feedback. The door to Prof. Aravind’s office was always open when I had questions.

Dear friends, your support surely helped me to finish my thesis and better its quality. My sincere gratitude to my friends and lab mates, namely, Behrooz Dalvandi for helping me in the application development, Mani Samani for helping me in the documentation and his feedback, Conan Veitch for thesis proofreading and his feedback, Nahid Taheri, Shanthini Rajendran, Darshik Shirodaria, Raja Gunasekaran, Arthi Babu, Gurpreet Lakha, and Braemen Stoltz for their passionate participations in my presentations and their feedback.

I would also like to thank my committee members Dr. Luke Harris and Dr. Samuel Walters for their very valuable comments on this thesis and their guidance. I also thank my external examiner Dr. Balbinder Deo for his feedback and his comments and Dr. Ian Hartley for chairing my thesis defence.
Glossary

Activity diagram  UML activity diagram. 66

Adoptability  The likelihood of a product to be accepted and used by other developers. 77

All-in-one systems  A system that has all of its components in one place. 31

API  Application Programming Interface (API) specifies how to interact with software components. 3

AR resource  Context and Content information of an augmented reality system. 67

Augmented reality browser  Type of browsers that use camera to detect and display information related to location around the user. 59

Client Federated Servers  The proposed framework of this thesis which is an enhancement over Client-server. ii

Client-server  Refers to a computational model that service providers called servers and service requesters called clients are scattered over a network. ii, xii

Context Provider  Services that collect and share context information. In augmented reality context is target or point of interest. 99

Co-routine  Is type of a function that can pause the execution and return the controller to caller and resume the execution from where it was stopped if control ever returns to the function. 80
**Content**  The information that is going to be delivered to the user in a Context-aware services service. ii

**Content providers**  Services that collect and share virtual objects. 51

**Context**  Any information that can be used to characterize the situation of an Entity. 1

**Context-aware services**  Services that are trying to provide relevant information to the user by recognizing a situation and adapting according to changes. ii, xii

**Cross-platform**  The ability of a software to run on multiple computer platforms. xiv, 4

**Data flow diagram**  UML data flow diagram. 63

**End user**  The ultimate intended user of a product. 63

**Entity**  An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. xiii, 1

**HTTP requests**  A communication protocol that allows a computer to send a request and a server to respond. 33

**Hub**  A center for data exchange and routing. 43

**KHARMA**  Is an open architecture for augmented reality that allows user contribution through HTML and JavaScript. 40

**Open systems**  Systems that have a combination of flexibility, interoperability, and portability. 26

**Participatory AR**  Refers to type of augmented reality with multiple users interacting in a shared space. 106

**Points of interest**  A location that a Virtual content is going to be registered for it. This is the equivalent of Targets for location based AR applications. xiv, 13
**Proprietary standards**  Protocols and specifications for a software or hardware that is controlled by a company rather than a standards organization. 31

**Prototype**  Incomplete version of a software that serves a special purpose in software development. 8

**Push notification service**  A message that is sent by servers and pops up on a mobile device. It does not need users to be in the application or use the application. 71

**Registration**  Aligning Virtual content on Targets. 11

**Scalability**  Capability of a system to handle its growth in work or number of clients. 30

**Scratcher**  Is an AR prototype application that have been developed as a proof of concept. ii

**SDK**  Software Development Kit is a set of application development tools that facilitates developing an application. xiv, 4

**Target Hub**  The structure that collects and shares Targets with clients. 54

**Targets**  Visual patterns that are to be recognized by an AR application. ii, xiii, xiv

**Token**  An ID issued by server to verify the client for future requests. 66

**Unity**  Is a a Cross-platform game engine developed by Unity Technologies. 4

**Use case diagram**  UML use case diagram. 64

**User profile**  Representation of user model or user identity. 33

**Virtual content**  Computer generated objects such as texts, images, videos and audios that are going to be delivered user upon detection of the Targets and/or Points of interests. xiii, xiv

**Vuforia**  Is a Software Development Kit (SDK) to enable augmented reality functions. 4
**WebClient Class**  A .NET class that Provides functions for sending and receiving data from a URI. 89

**WWW class**  Is a small utility module for simple access to web pages. 86

**XML**  Is a hypertext system that operates over the Internet. 40
Chapter 1

Introduction

1.1 Overview

The Web has affected our lives in many respects from communication and information sharing to business models. However, we have never stopped in asking for more. As such, a new era of Web called Web 2.0 emerged with particular emphasis on two main aspects: user contribution and considering the Web as a platform [61]. For user contribution, it is expected to enable users to create and share contents through participation. This pattern of two-way traffic from developers to users and from users to developers have been encouraged by Web 2.0 and has been adopted in social network and social media. Web 2.0 also expects a multifunctional mash-up of services which is created by combining services offered by different service providers. This allows us to consider Web as a platform for application developing instead of detached (islands of) scattered services.

On the other hand, context-aware services are trying to provide relevant information to the user by recognizing a situation and adapting according to changes. In this regard Dey et al. define Context as “any information that can be used to characterize the situation of an Entity. An entity is a person, place, or object that is considered
relevant to the interaction between a user and an application, including the user and applications themselves” [11]. It is believed that the availability and the quality of contextual information will be crucial factors of future context-aware services [52, 77]. One important issue in context-aware services is a shortage of standards for contexts. Especially the problem gets magnified when it comes to fusing several types of contexts to offer a particular service.

There have been efforts and research studies toward integration of Web 2.0 with context-aware services. Many of the suggested platforms have been adopted over the past few years. For instance, a mobile middleware component has been proposed as a platform to collect user context and authentication information in [49]. This platform allows Web services to subscribe to user context and utilize the services offered by other subscribers. It would be beneficial to bring Web 2.0 and context awareness together. However, in different technologies, context, content, and user contribution may have different meaning and requirements. Therefore, a platform that is desired to deliver both Web 2.0 standards and context-awareness should be tailored and customized according to those requirements.

With the advancement of computing and communication technologies, the virtual worlds have been created. A virtual world is using digital objects such as sound, image, video, and graphic, and it has become part of our everyday life. Virtual reality technology is an attempt to create a fully virtual world that gives users a real life or “make believe” experience. Augmented Reality (AR) takes this idea even further by “augmenting” digital objects into the physical environment for enriching the users’ perception of the real world. That is, AR technology enhances the physical environment by augmenting digital objects and enables users to interact with this enhanced new world. Augmented reality is a cutting edge, under development technology in the realm of context-aware services. It allows computer generated contents in the form of image, text, video, audio and 3D objects to be superimposed on physical objects. AR technology enable users to experience new possibilities that are not feasible either in a
real world environment alone or a virtual world alone.

In theory, AR technology forms a spectrum of possibilities – touching at one end with an entirely real (physical) world environment and the other end of an entirely virtual (digital) world. The goal of augmented reality is to enhance users’ perception of the reality by providing information related to the context. A significant number of AR applications have been developed over the past few years. Nonetheless, only a few of them has proven to be successful among them the most famous is Pokémon Go [10].

In practice, there are some issues and obstacles in augmented reality which stop it from achieving its goals to the fullest. These problems are partially from context-aware services and partially because of the architecture of the current AR applications. For instance, proprietary interfaces and formats have been a huge obstacle for widespread adoption of this technology [36]. On the other hand, there is no standard way of users’ contribution whether to create or share a content or communicating and sharing ideas. This not only makes it difficult to support users’ contribution inside AR technology, but it also becomes a bigger problem when it is desired to share users’ contribution as a service to other Web services. The existing implementations of the AR applications use one of the popular models of distributed systems called client-server model. We believe that this traditional implementation of AR using client-server model has some significant limitations. Especially in compliance with Web 2.0 and context awareness, the need for a platform that can provide Web 2.0 features such as user contribution and a rich high-quality context-aware experience is highly perceivable.

The main contribution of this thesis is to propose an alternate model of AR implementation to address the limitation of the current AR implementations and offer added benefits to AR users. We refer to the proposed alternate model for future AR applications as “client federated servers.” The key objective of this new model is to enhance user experience and also increase the applications of AR technology. This model combines elements of context awareness and Web 2.0 standards tailored to meet AR needs. It allows suitable collaboration between servers of different AR apps to share
their resources regarding targets and virtual objects. Target hub is the server side and backend infrastructure of the proposed model. The target hub gathers user context and AR content information such as communicated messages and shared contents and makes them available through offered API to its subscribed servers. In client federated servers model, users of an AR app can communicate not only with each other but also with the users of other AR applications. Joining databases of the targets and virtual objects, and allowing communication between users, can make AR environment more intuitive and consistent. This also will enrich the experience of AR users.

In this thesis, we have implemented a proof of concept that we refer to as “Scratcher”. The name stems from the analogy between scratchcard and AR applications. To reveal the information behind the opaque covering, we have to scratch the cover. The same way, Scratcher reveals the information (content) covered behind the physical object (target). Scratcher is an application that has been implemented in a Cross-platform environment using Unity, a powerful game engine, and Vuforia, an SDK for developing AR applications. Users can log into the application and start browsing. They can search for a target by its name or by the assigned tags. A target is any object or image of an object that is recognizable by the app. Targets are stored either locally on the device or in a cloud. Using the target hub, it is possible to share targets with other applications. It is also possible to search for a target using the assigned tags and download it to a device. Upon recognition of a target, users can click on it and enter to its chat room where they can start to share their ideas. The platform allows users to communicate with not just the Scratcher’s users, but everyone subscribed on the target hub. This project enables AR functionalities as well as allowing user’s contribution in multiple levels including sharing content and exchange of ideas and popularity vote.
1.2 Motivation

My first experience with augmented reality started with an NSERC Engage grant project to develop an AR mobile framework for teaching First Nations language and culture. During the project, I learned that AR holds high potential to bring us together and help us share our stories. As Alida Gersie has mentioned in [33], stories are the source of inspiration; through them, we can understand how to value and devalue our planet. The project finally finished, but my work had just begun. I wanted to know with all the potentials that AR holds and all the exciting experience that it can offer: why AR is not popular, where the problem is, and what is lacking?

Pursuing these questions led me to be more familiarized with augmented reality. Then, I noticed the key of success to AR is partially lacking in all aspects of right data, right time, and the right place. I think we all have heard the famous quote that “the key to success is to be in the right place at the right time.” Well, I think augmented reality is supposed to be all about right data in the right place at the right time, except that it is not. In augmented reality, the right data would be high-quality contents. The right place would be targets or points of interests upon which the content is going to be overlaid, and the right time is the time of need that is going to be inferred from context information. In general, I noticed that it is not just the lack of data but also a lack of cooperation among providers and consumers that affects AR. Typically, AR resources are stored in the proprietary databases with proprietary formats. One major motivation of this work is to bring all these resources under a common framework and make it accessible to all AR users.

My other key motivator for this work is the belief in the wisdom of crowds that the many are smarter than few. We need to bring powers of crowds into augmented reality. One way of that would be enabling users’ contribution by supporting bi-directional interactions between users and developers. Users of augmented reality should be able to contribute to the collective experience by sharing their side of the story. This
contribution needs the users to be able to create AR contents and targets and share them with others. Also, users of AR should be able to communicate and share their ideas with the provided medium. I intend to deal with these issues.

1.3 Research Problem

Suppose two persons A and B are walking in a park. User A is using his AR application and browsing a tree. User B is using her AR application and browsing flowers. When User A turns his smartphone to the flowers, he could not browse the flowers. Also, when user B points her phone to the tree, she could not see any superimposed information over the statue. Although both users are browsing in the similar environment at the same time, they lack sharing capability. How can these users share their virtual contents with each other?

Another example, suppose users A and B are browsing a famous statue in the park. User A wants to share his ideas about it with other interested persons (e.g. user B). User B wants to chat about the history of the statue with other concerned users. How can these users communicate?

The research problem is to investigate the causes for the lack of communication and sharing and their consequences on mobile AR applications.

1.4 Purpose of Study

Since virtual content is retrieved from a server, the server side of the application should support any necessary structure for sharing and communication. This leads to the following questions.

From a developer point of view, is there a way to support such a sharing and
communication capability? What are the requirements?

The purpose of the study is to understand the causes for the lack of communication and sharing in AR with the aim to develop a new method that can support AR framework for sharing contents between AR applications and communication between AR users.

1.5 Objectives of Study

Considering the two users A and B mentioned in the research problem; we are investigating the possibility for these two users to see more than they browse now. To be exact, if user A can browse the targets $T_{A1}, T_{A2}, ..., T_{An}$ and user B can browse the targets $T_{B1}, T_{B2}, ..., T_{Bn}$.

- We want to enable user A to browse any subset of $\{T_{B1}, T_{B2}, ..., T_{Bn}\}$ and vice versa for user B to browse any subset of $\{T_{A1}, T_{A2}, ..., T_{An}\}$.

- We want to enable user A to send and receive messages from user B and vice versa for user B to send and receive messages from user A.

1.6 Research Questions

- What kind of software models and protocols would enable user A to browse any subset of $\{T_{B1}, T_{B2}, ..., T_{Bn}\}$ and vice versa for user B how is it possible to browse any subset of $\{T_{A1}, T_{A2}, ..., T_{An}\}$?

- What kind of software models and protocols would make it possible for user A to send and receive messages from user B and vice versa for user B to send and receive messages from user A?
• If we propose a new AR system to solve the above two research questions, how to justify its feasibility in practice?

1.7 Contributions

Our attempt to answer the research questions resulted in the following contributions.

• The main contribution of this thesis is to propose an alternate model of AR implementation to address the limitations of the current AR implementations and offer added benefits to AR users.

• Using the proposed framework, we can connect users of different AR applications. Chatrooms for targets connects the users of various applications around a common topic of interest.

• Our third contribution is expanding the scope of AR applications by providing the ability to share targets among applications of different platforms. Users are also able to add targets and contribute to authoring AR targets and contents. This will lead to a jump in the number of targets. Also, contents can be shared using this framework. However, in the project implementation, we have implemented only the target sharing.

• Last but not least, we implemented the proposed model as a Prototype AR mobile application. Our implementation is not the only way to implement the client federated servers offered in this thesis. However, it encompasses all the necessary modules, and it functions well enough to be considered as a proof of concept for our proposed model.
1.8 Thesis Structure

The structure of the thesis is as follows:

Chapter 2 offers the necessary background to the concepts used in this thesis such as social network, social media, client-server, and augmented reality and its applications. Then, the literature review section covers current implementation of the AR and problems and limitations that it imposes on the current AR applications. Also, in chapter 2 we get to know the previous attempts in dealing with social services in AR as well as the advantages and limitations of one over another.

Chapter 3 gives a detailed technical insight of what we are offering as a framework, the components of the proposed framework and their tasks in the system. Also, the information flow from an AR user to the servers and finally to the target hub has been discussed in Chapter 3.

Chapter 4 evaluates the feasibility and practicality of the proposed framework through a prototype application named Scratcher. The blueprint for the implementation of the AR application, the server and the target hub has been presented in detail.

Chapter 5 discusses the limitations of the work and concludes the thesis by offering guidelines for the interested researchers who want to continue this vision.
Chapter 2

Background and Literature Review

In this section, we will first give a background to augmented reality, its evolution, applications, and technological aspects such as challenges and limitation. Then we discuss the client-server architecture adopted for augmented reality, its advantageous and disadvantageous. Then we provide a brief background to Web 2.0 concept, its features and how it is helping augmented reality. We finalize this chapter by reviewing the previous researches on employing Web 2.0 features in augmented reality. We are particularly interested in the research studies that are proposing AR resource sharing and user contribution in augmented reality.

2.1 Augmented Reality

Informally, AR technology allows computer-generated contents in the form of image, text, video, audio and 3D objects to be superimposed on the physical environment. Consider the two simple example applications of AR given in Figure 2.1a and Figure 2.1b.

In Figure 2.1b, the physical environment has a speed board and a speed detector equipment displaying the speed of the moving vehicles on the road. Here, based on the
speed detected by the equipment, the digital image (speed in number) is created and
displayed on the display board. This simple AR application gives the impression to the
ordinary users that she is driving the car in a smart environment that can provide (alert)
with useful information about her car speed. In the example shown in Figure 2.1b, the
camera from the cell phone detects a body portion of a human and displays the digital
image of the internal parts (organs) of that portion of the body. In this application,
by focusing the camera on the different body parts of a human, one can visualize the
internal structure of that body part. This can be an excellent application of AR system
for educational purposes.

Formally, as defined by Azuma in [16], AR is an interactive space created through
computer-generated images capable of 3D Registration and rendering with the display
of a combination of real and virtual objects. This definition has three basic elements: 1)
AR mixes real and virtual objects; 2) AR is interactive in real time, and 3) AR registers
virtual contents with physical objects in the real world.

Augmented reality is the middle ground between virtual reality as an entirely artifi-
cial world and telepresence as a whole real world [59]. Figure 2.2, shows a continuum
from the real environment in one end and virtual environment on the other end encompassing augmented reality and augmented virtuality. The difference between virtual reality and augmented reality lies in the environment in which the user is positioned. While in virtual reality the user is immersed entirely in a different world, i.e. a virtual world, in augmented reality virtual images and information are delivered to the user in its real physical world.

2.1.1 Brief History of AR

Ivan Sutherland started to work on virtual reality system named “the ultimate display,” and he succeeded to make the first augmented reality and virtual reality head mounted display system about half a century ago in 1968 [81, 82]. Since then, AR had been considered as a subset of mixed reality or as a variation of virtual reality rather than a separate field for a long period. The focus of that era is on visual sensing and display which both are inherited from virtual reality and mixed reality. Later in 1998, with virtual reality fading into the background and emerging the new concepts such as ubiquitous computing (UC) Weiser [85], the focus of AR started to change toward user experience rather than visual display. This shift is understandable considering what UC is proposing. Instead of taking computation into a virtual world, UC brings computation power into the real world. In the same way, AR started a new trend toward smart and networked objects. For instance, Mackay in [56, 57] introduced a wider definition of AR which considers interactive virtual objects and also smart networked objects. This
trend of AR tries to enhance everyday objects with memory, computation power, and a sense of awareness, which would lead to augment the user experience. By then AR had turned to be a field of study and the first AR conference, International Workshop on Augmented Reality (IWAR), was held in San Francisco, October 1998.

After two decay of active research on augmented reality, today’s AR is rather different from what we had expected to experience. There can be many reasons for that as Evan Barba et al. [17] speculated one reason could be lack of technology for smart objects. Major AR technology today boils down to smartphones and, of course, Google glasses or other companies’ head-worn devices. This would never satisfy tangible or smart objects. Meeting users’ needs is the other important factor in forming of today’s AR. By looking back to the proposed AR, it is simply understandable that they either were not beneficial enough or were not easy to have them in everyday life. The important outcome of this is identifying key elements in today’s AR. From a technological point of view, smartphones and head worn glasses are key elements in developing current AR applications. Also, the main source of computational power is the cloud and applications should be tailored to the user.

2.1.2 How AR Works

Typically, AR systems follow a general system flowchart which is initiated with an input sensory device such as a camera capturing the predefined scenes of the real world. AR application reads this input and matches it to a database of patterns that are to be detected. These patterns are formatted images or locations, and often are called targets or Points of interest (POIs). When there is a match in the database of targets, the location and orientation of the camera are calculated, and a virtual content is aligned with the target. This aligning action is called registration. The virtual contents are often labels, images, or 3D models, which can be stored locally on the device or a server. Real scenes captured by a camera and the virtual content should be rendered (combined)
to a new displayable image. Finally, the augmented image will be displayed on the user’s device. Figure 2.3 shows the diagram of these actions in the described AR system. Another important process of any AR application is tracking. Once a target is detected, its location will be tracked by the relative changes of locations and angles calculated from previous scenes.

![Figure 2.3: A simple AR system](image)

In general, AR systems are composed of the following components.

**Input:** Target or POI from the physical environment and its position. Target detection is one of the most difficult tasks computationally. Therefore, to alleviate the computational complexity of this task, targets from the environment are assumed to have easily identifiable features. These identifiable features are referred as *fiducial markers*. From the location of fiducial markers, the positions where the virtual objects must be augmented are computed.

Fiducial marker is a unique pattern that essentially tells what virtual image must be displayed and in what point of view. Fiducial markers are physical entities with unique features, and they could be from the actual environment, landmarks, or objects artificially attached for the purpose of identification. Commonly used artificial fiducial markers in AR applications are unique drawings, images, and “quick response” (QR) codes.

Typically, sensors and cameras are used to identify fiducial markers. Locations are calculated based on the readings from the sensor devices such as a compass, GPS,
gyroscope, accelerometer, etc. Buttons, touchscreen, keyboard, mouse in the user devices can also be considered as sensors that capture user inputs.

**Registration:** Refers to spatial alignment of virtual objects with the physical objects. Registration requires knowing where the physical things are in space in real time so that the virtual objects could be aligned with the real objects accurately.

**Tracking:** Keeping track of the objects’ current locations and positions that are necessary for the accurate registration process. This operation becomes tedious if the device or the fiducial markers are mobile.

**Target and virtual objects repository:** This is a database of targets and virtual objects. Target objects normally are formatted source images of the targets or their signatures. Virtual objects are a predefined set of computer made graphics, images, or texts made ready to be rendered along with physical objects upon the detection of the targets.

**Graphics/Rendering:** When a target is detected, the corresponding virtual objects are first fetched, then aligned with the real scene, and finally compiled into one “image” that can be displayed.

**Display:** The display technology on which the rendered scene (the final image) is displayed.

**Communication:** The messages that are relayed between the AR components mentioned above when they reside on different systems.

From the implementation perspective, these components can physically reside in one system or multiple systems based on the application requirements and performance metrics. Jens Grubert and Raphael Grasset [34] have considered AR applications in three layers: (a) the application layer; (b) the AR layer; and (c) the OS third party layer, as shown in Figure 2.4. The application layer implements the logic of the application. For example, in an AR game, the application layer handles the characters and their
behaviors (such as movement). The AR layer includes main components needed for any AR application namely, display, registration, and interaction between the other AR components. The OS layer provides required essential services such as processing sensory input such as camera, GPS information, etc.

![Components in AR architecture](image)

2.1.3 Applications

Over the past decades, several areas have been discovered and introduced as potential application areas. In some of those areas, AR has flourished much more than others. For instance, the military is using AR in pilots’ helmets as an example, but in education, we have not seen much yet. This section briefly introduces those areas.

2.1.3.1 Annotation

Annotation is almost most typical and preliminary type of AR in practice. Assuming to have a very large database of objects and information, a user can point his smartphone toward different objects. As soon as the objects detected, the related information is overlaid on the objects in real time. This can be helpful in navigation, or any guidance
system [29, 68]. One example of this application has been implemented in [69], called augmented library which assists the user to find a book or answers questions about the books in the library.

2.1.3.2 Medical

It is possible to collect a 3D dataset of a patient through several types of sensors and then combine and render these images to make a compelling virtual content. Doctors can have an “X-ray” vision on the patient in the real-time. Other potential applications can be in the surgery room by providing a vision of the needle inside the patient’s body [31, 78]. This, of course, would need a very precise registration and tracking.

2.1.3.3 Entertainment

Many AR games are already in use. Also, developers all over the world have started to use AR in their games. Unique perspective, interacting with game objects directly and in 3D, and mixing the real environment with the game environment are part of the features that AR has brought into this wonderland. “ARQuake,” created by Piekarski and Thomas [64], is one example of such environment that player is playing in the real environment superimposed by virtual enemies.

2.1.3.4 Military

One very famous example of this would be pilots’ helmets. For example, the F-35’s helmet-mounted display system superimposes the necessary information such as air-speed, heading, altitude, targeting information and warnings (see Figure 2.5) [30].

There are many other application areas which we only name them here as we do not intend to survey AR in this section:
2.1.4 AR Research Areas

By studying technology surveys in [15, 16, 90] it is inferable that there are a number of areas to be considered for a successful AR application. Zhou et al. in [90] have listed major areas in AR as follows:

**Graphical hardware and software:** Complex 3D virtual content creation and rendering as well as overlaying those contents on video streams would need suitable graphical hardware and software.
Tracking and registration: Virtual contents should be related to at least one aspect of the real world; this is called registration. In a further attempt, real world objects and location should be tracked so that virtual contents would properly be adjusted to the changes.

Display hardware: The results of tracking and overlaying need to be reflected which is why display hardware is required. It can be a monitor, projector, cell phone, etc.

Processing unit: A computational unit to run AR application code, which might be distributed.

Interface: Any human-computer interaction would need an interface to convey commands and demands between users and the application. This gives the capability of manipulating contents.

Although AR researchers usually have focused on one or number of above topics, it is important to know a typical AR application includes all of these areas.

2.1.5 AR Enabling Technologies

2.1.5.1 Tracking

Tracking is the most popular research topic in AR context. Based on what tool and technique are used for tracking the targets (real world objects), tracking methods can be one of the followings:

A) Sensor based tracking

The main idea is to use sensors such as magnetic, acoustic, optical, and other types of sensors, to detect and track the targets. There are only a few research studies in this context, mainly because of major disadvantages of this method. For example, distortion is a common problem with magnetic sensors. There have been attempts
to combine different type of sensors to reach more accurate tracking. For instance, Klinker et al. tried to combine a local monitoring system installed on the human body with fixed global tracking [48].

B) **Vision based tracking**

Instead of sensors, computer vision along with image processing is used to detect and track the targets as well as dynamically calculate pose and orientation of the camera and objects. This is the main research area in tracking techniques. Vision based tracking divides into two groups: marker-based and feature-based (or markerless) [65].

The marker-based approach uses fiducial markers to calculate the camera pose. One of the dominant tracking techniques was square markers. Stricker et al. investigated a method for finding coordinates of the four corners of a marker [79]. Famous approaches in this area have been thoroughly reviewed in [89]. The feature-based technique which was introduced by Park is trying to find targets using natural information extracted from the edges and lines in the image of the target [62].

C) **Hybrid tracking**

There are advantages and disadvantages of the two mentioned methods. Mainly vision based tracking has low jitter and no drafting, but it is slow to swift and fast motions which might lead to tracking failure and it is time-consuming to resume once the target is lost. Sensor based tracking, on the other hand, is vulnerable to distortion and draft and errors can be accumulated leading to inaccuracy. A combination of both methods seems to be a better method. For example, vision-based tracking along with GPS localization and acceleration sensors for calculating rotation and camera pose.
2.1.5.2 Interaction and Interface

Interactivity and interface are the important aspects of AR which have gained increasing attention. Intractable virtual contents and user-friendly interfaces are major milestones for AR technology on its path of evolution.

It is a delicate way of interaction in which users can interact with the virtual contents and through that with virtual world without dealing with traditional computer interfaces and only by real world objects. The basic idea in AR interaction is to bridge virtual and the real world through manipulation of features in physical objects.

Toward this goal, tangible augmented reality has introduced in [45]. Considering that every action in the real world can be interpreted as an interface command to the virtual world, it is understandable why tangible augmented reality is popular. Hand gestures or finger hints are one of the popular ways of interacting [58]. One challenging issue for developers was how to instruct users to make the right motion to activate the desired command for which a nice solution is proposed in [86]. They were augmenting visual hint on the real object to guide the user to proper action.

Another interesting aspect of AR interaction, is collaborative interaction which mostly happens between multiple users in a shared space. The beauty of the work lies in the intuitiveness of the interactions which is based on already established social protocols. In [37], Henrysson showed how AR can support collaborative interaction by virtually playing a tennis game. In this work, the phone acts as a tennis pad that vibrates when the virtual ball hits the pad (the phone). In another attempt, Stafford et al. introduced an interesting way of interaction between indoor and outdoor users [76]. In this study, an indoor user pointed a location on a map which triggered an augmentation on the same location for the outdoor user as a finger that came from the sky (called God like interaction). This can facilitate interaction between indoor and outdoor users.
2.1.5.3 Display Methods

Any AR experience should be visually reflected for its users at the end. Unless it is the audio augmented reality which is beyond the scope of this study. To this end, there are several methods or in other word several instruments each of which has its advantages and disadvantages. Three main categories of displays have been recognized in [47]:

1. Mobile handheld displays

2. Video spatial displays and spatial augmented reality

3. Wearable displays

Handheld devices Most popular AR displays are handheld displays due to several reasons such as their small size, ease of mobility, lower prices compared to other types, minimally intrusive, and accessibility since they are already present in the social life. Cellphones and Tablets are the most popular devices that are used in this technique. Considering recent advancement in cell phone technology including embodied Cameras, GPS, different sensors, and high-resolution screens, there are high promises for this type of display. This goes that far that many researchers tend to study AR only in the smart phone’s ecosystem (handheld devices generally) [17]. Although there has been a significant technological advancement in handheld devices in recent years, still slow processor and low memory can be their drawbacks in many AR applications. The main limitation for them might be tracking which is mainly based on image processing tools such as ARToolkit, Vuforia, Metaio and similar other instruments.

Video spatial displays and spatial augmented reality Spatial augmented reality refers to a type of device that displays the virtual contents directly on the physical objects. These devices include projector based displays and holographic optical devices or half silvered mirrors. Distinguishing feature in this technology is the natural view and feelings that it serves to its users. However, the need to the extra and
usually expensive device is the disadvantage of this technique. Projector based displays are most suitable for applications with several users demanding to share their AR experiences such as a teaching class or a surgery room. Projection light should be registered with a physical object and illuminated on the object for which a projector illumination method has been proposed in [22].

**Head mounted devices (HMDs)** Wearable devices are goggles such as head mounted displays or glasses which augment the virtual contents in a more natural way. This type of displays is composed of the real-time video stream on which virtual contents are overlaid. Owing to plenty of image processing techniques, handling occlusions and color contrast and other types of lighting difficulties are much easier comparing to optical displays. Modern head mounted devices allow six-degree freedom of movement and monitoring (Figure 2.6). Although HMDs seem to be partially successful, wearing HMDs for too long can make users uncomfortable.

![Figure 2.6: A modern HMD - Microsoft’s HoloLens [9]](image)

2.1.6 **Challenges in AR**

Surrounding user in a mixed world of real and virtual objects leading to a more desirable world is the core idea of augmented reality. To achieve this goal following challenges and limitations should be answered and overcome.
So far data have been categorized by applications. The extreme view of such classification is in smartphones that each application has its own data space (sandbox). Obviously, AR tends to group data based on location and environment. Generally speaking, the current cell phone’s ecosystem could be much better for an AR application than it is now. So this ecosystem can build a state in which there are a huge amount of virtual contents for the desired object but distributed in several applications. In this situation, we might be able to experience all of these contents separately. However, we probably will not get the killing application and AR’s ultimate potentiality unless all those contents are experienced together. Having two services at one time has an added value. Obviously, these contents together are more than the sum of all of them in separate. Hence, experiencing these contents together is a richer experience than experiencing the contents individually.

Another challenge is removing a significant implicit assumption. Strangely it is expected from users to take out his cell phone once in a while and point it in a random direction hoping that there are going to be a virtual content, or otherwise it should be assumed that users already know where they should look for virtual contents. This assumption not only is absurd but in many cases is against the spontaneous nature of augmented reality. For users to enjoy the spontaneous nature of AR, new forms of display technology are needed. Such technologies should be able to contribute to our social life rather than being intrusive. Google glass could be an example of this technology, even though it has its problems.

The two challenges mentioned above are mainly related to IT technologies on a larger scale and not necessarily AR. There are a few number of AR specific challenges which addressing them would pave the way for reaching to AR goals. The most significant AR challenges are:

**Accurate registration:** When it comes to outdoor applications especially in open area, registration becomes a major problem [48]. Integrating virtual contents in the real world depends vastly on the accurate calculation of the camera pose and the
physical object’s position and orientation.

Another problem in this context is switching between different registration techniques such as GPS and fiducial markers or other techniques.

**Virtual content quality and quantity:** AR applications rely on virtual contents, and virtual contents depend on the density of the point of interest (POI). While the density of POIs is high in some locations such as downtowns of the cities, it gets scattered when it gets to rural areas. This brings an unpleasant experience for AR users. The other factor is the quality of the contents especially when it comes to comparison with the Web [79].

Technology itself is imposing limits on AR from different perspectives. These limits are partly because of hardware limits imposed due to frame rate. There are also algorithmic limits imposed due to computational complexity. Scene complexity and calculating state of virtual contents are two examples of such limits.

**Scene complexity:** Tracking moving objects depends highly on some factors including frame rate, the motion of the target object, and prediction algorithms. The speed of the target might cause tracking to fail. Sensor based tracking can be helpful here. However sensors need maintenance, and besides, they have short ranges and can only be tracked when they are in the scene. This makes them unsuitable for outdoor applications.

Another approach to overcoming this problem is to use natural features of the target objects, “edges” for instance. To do so, different prediction algorithms must be exploited, and predictions usually are error prone. Hence, these types of algorithms are computationally heavy and not always applicable. Kalman filters [42], for example, are used to handle the uncertainties of predictions, but these filters only apply to mostly linear systems that can be described by unimodal distributions which often is not the case for outdoor AR applications.
Calculating state of virtual contents: For interaction purposes, AR needs to have an accurate calculation of virtual contents. In many AR applications, users are interacting using the physical objects and virtual information attached to them. Tangible surface as the interface has two main constraints. First, it is difficult to recognize the state of the virtual contents about the physical object. The result of a study done by Grubert et al. in 2011 indicates that the content and registration issues are one of the causes for discontinuing the use of augmented reality browsers [35]. Secondly, dimension calculation in tangible settings depends on the surface of the tangible object. Although it is possible to exploit markers to mitigate the issue, hand occlusion can easily fail marker based solutions.

2.2 Client-Server Architecture

Though an entire AR application can be implemented in a user device, today’s AR applications are typically implemented on a distributed system consisting of client devices, a communication network, and a server. Hence, AR applications are generally distributed applications. Figure 2.7 shows the client-server model that is used to implement distributed systems.

In the client-server architecture, a distributed application is structured with having two main components, namely, clients and a server. The clients reside in the user devices, and the server with associated databases resides elsewhere in the network. A communication network such as the Internet connects the server and the clients. The server provides a set of services to the clients. Often, the clients initiate service requests through the network, and the server responds to them by offering appropriate services.

According to Alex Berson [20], the client-server approach offers many advantageous such as leveraging desktop computing technology and, recently mobile computing technology, reducing the network traffic by residing the processing close to the source
of data, facilitating the use of graphical user interface (GUI), and above all encouraging Open systems. However, it is aptly emphasized that a client-server architecture must be founded on standards-based architectures to fulfil interoperability and application portability requirements.

2.2.1 AR Architecture

Although several frameworks have been proposed and implemented for AR applications, currently, AR applications are typically modeled as client-server architecture [32, 71, 74, 84]. In this model of AR, when a user is close to the target (i.e., a client detects a target) the corresponding client software requests that the server provide the predefined virtual contents. Then the client, with the help of the server when required, registers those virtual contents with the real world of the user. The virtual contents are then rendered and displayed along with real world objects by the client. Some implementations use separate databases for target objects, virtual contents, and subscriber information.

Gassmann et al. split AR into two main tasks [32]: object recognition, which is
handled on the server, and tracking the object, which is on the client side. They have efficiently implemented their platform on the Android system. The proposed platform provided AR compatible target detection service for 300000 object clusters having a response time less than 2 seconds. Another implementation of AR using client-server architecture is shown in Figure 2.8. The tasks of detection and tracking have been implemented on both client and server sides, and their final location depends on the application requirements. The choice of implementation is based on the application’s requirements and available resources such as computing and storage capacity of the client devices and the network delay and bandwidth. Similarly, virtual contents that are to be superimposed on real scenes are stored in the client devices.

Since the focus of our research is on the server side, we are more interested in the role of the server in the client-server framework. Sobota and Janoso [74] have listed the main modules of an AR server application as follows:
A) Data management

This module is responsible for providing AR data including 2D and 3D models, texts, graphs, photos, and target data including markers and targets. This module manages to store and to retrieve requested data from the database.

B) User management

User management includes user authentication, access controls, user data sharing, user profile management, and etc.

C) Calculation of position (content management)

Location and direction of the user are needed for marker identification, camera position calculation and gathering all marker related data. In general, this module is for managing content information.

D) Network connection

All client-server interactions are using the network connection, and AR data, target model, marker data are transferred over the network. Establishing network connection and maintaining it is a crucial factor for any client-server based AR application.

E) User interface

The main role of the server interface is allowing users to modify and edit all types of AR related data. This includes markers, models, targets, labels, texts, clients’ information, and etc.

Shen et al. have proposed a client-server architecture and mechanisms to support product design in a collaborative AR environment [71]. Figure 2.9, shows the flow of information and interaction between clients and the server.

There are several benefits of dividing AR applications into client-server system instead of an all-in-one system [74]. Those benefits are, briefly:
Figure 2.9: Flow of information in client-server framework [71]

A) Mobility

An important aspect of an AR application would be mobility. Since many of AR applications are in the fields, it would be desirable to have a light and easy to move device. Splitting AR into client and server allows developers to use server’s computational power and memory storage. This makes the client-side device more affordable regarding weight, scale, and price.

B) Centralized data

Since client-server architecture provides a centralized management of data, it eventually increases performance comparing to all-in-one system. Updating data or modifying it is centralized and simplified.

C) Scalability

Scalability can be understood in different ways, here we are considering it as the ability of the system to work efficiently with a large number clients and a significant amount of data. In that sense, a client-server approach would outperform an all-in-one system due to having bigger resources and easier way of increasing
these resources. For example, it is difficult for users to increase their smartphone’s memory beyond 128 GB. Therefore, an AR database of 1 TB would be impossible for a big portion of users.

D) Cost

Computational power and memory storage are two costly elements of computer systems. By outsourcing part of these two elements, the client-server approach is more beneficiary in comparison to All-in-one systems.

Although there are many benefits of a client-server architecture for AR applications, however, with the current implementation of this model, each application mostly has its own server and proprietary databases [36, 50]. The database maintains the target and virtual objects and the number of subscribed users. In this regard, Alex Berson in [20], has summarized the disadvantageous of vast proliferation of the Proprietary standards for a client-server architecture as summarized below:

• The very high cost of switching for customers that are locked into one vendor’s system.

• In case of migrating to another environment, the cost of documentation and user training.

• Software developers tend to develop for vendors with larger systems and thus larger systems will always have a competition advantage even when they are not meeting users’ need any more.

Therefore, in the next chapter, we are going to discuss in more detail the restrictions that a typical client-server approach imposes on AR applications, and then we propose a new framework for AR applications.
2.3 Web 2.0 and Social Services

2.3.1 Web 2.0

Web 2.0 refers to standards that enhance the freedom of sharing and reusing of Web contents by using open communication standards and decentralization of authority [18]. Web 2.0 is not about technical specifications, but more about enhancing the way users are utilizing the Web. The focus of the Web 2.0 is creativity, communications, secure information sharing, collaboration and functionality of the Web [53].

Web 2.0 supports the wisdom of crowds [80] and the idea that large groups of people are smarter than an elite few, no matter how gifted those few may be. This way of thinking has led developers to come up with solutions that shift the paradigm of considering users merely as consumers of information to a more interactive and cooperative paradigm in which users are also producers of information. Web 2.0 is distinguished from Web 1.0 with users’ ability to create Web content, whereas in Web 1.0 only developers and authors were making Web content and users were accessing them without the capacity to modify them. This unidirectional way of accessing Web contents in Web 1.0 has been changed to bi-directional communication, allowing users not only access the information but also to create and modify them.

The other simple yet powerful concept that has been introduced by Web 2.0 is keyword tagging [53, 70]. Tagging is capable of replacing sophisticated semantics of the Web and allows a broad audience to search and access the contents of the Web easily. Tagging also provides an efficient way of organizing and sorting the contents for both developers and users.
2.3.2 Social Networking

Social networking sites work by creating and managing user profiles which are a fundamental concept in all social networking services. These profiles and their data can be shared among members of the social network. The social network itself is formed by linking the profile page of the members. This linking of the pages is a function under the user’s control and part of their contribution. Users link the pages based on their shared interests or shared friends, etc. The social network allows searching for contents and friends’ profile through a vast number of pages [18].

Social networking and crowdsourced contents can be seen as a result of Web 2.0 concept. Especially because of the open standards of Web 2.0 that provide the necessary functions to create, share and search for data in the massive scale of social networks. Through Web 2.0 APIs, information from different sources are combined, and users are experiencing this enriched environment in the social networks. Once again the simplicity of Web 2.0 is in reusing the existing protocols such as HTTP requests, JSON, and AJAX calls that are used to implement the infrastructure of Web 2.0. This infrastructure handles all of the required functionality of Web 2.0 [70].

Social networking is a Web-based service with three key features: a) capability of having a User profile, b) connecting to other users and showing the list of connected users, c) view and traverse the list of connections of other users. The core element of social networking is not just the networking, but the capability of showing and sharing the profile and the network of friends which are members of the system. Several social networking services (SNSs) have been developed, but their main difference is in the structural variations of visibility of profiles and networks and also in access to the contents. There are also functionality differences among SNSs such as capabilities of video and photo sharing, built-in blogging, and instant messaging [23].

The research areas in social networking are mainly about analyzing the behavior of users such as the ways people are communicating, who do people are communi-
cating with, who do people share information with, etc. [24, 27]. DiMicco et al. have summarized the motivation behind sharing information on the social network in three factors of caring, climbing, and campaigning [27]. Caring is about connecting on a social level which is a source of personal satisfaction. Climbing is about career advancement, which doesn’t seem to hold true for all type of social networking sites. The last theme in sharing contents in social networks is campaigning which is about sharing ideas and seeking support for it. Their results show that most shared content belongs to comments that users have written with 20.3% contribution and the next is adding connections with 11.2% and then photos, status messages, about-you’s and list sharing.

2.3.3 Social Media

It seems that social media logic has blended with mass media logic. Therefore, it is imperative to understand mass media’s strategies and tactics first. Mass media considers the world as a continuous flow of events - a stream of things and people out there. Albeit the fact that mass media applies filters and adjustments on the level of exposure for the covered items, they are trying to present themselves as neutral platforms that cover different voices and opinions fair and just. To legitimize its independence, mass media uses ratings, polls, and surveys as evidence of audience demand. Framing reality and media’s neutrality or independence claims have been reported as elements of mass media’s logic by Altheide and Snow [12].

Same as mass media, social media also handles polling and surveys and ratings. The difference is in the capacity of social media platforms to seamlessly integrate those processes in the architecture of social media. According to Dijick and Poell [26], social media logic is about channeling social traffic. More precisely, social media logic refers to “the processes, principles, and practices through which social media platforms process information, news, and communication.”

Having a two-way traffic of data from producers to consumers and also from con-
sumers to producers is the most significant functional difference in the logic of social media and mass media. As mentioned before, the two-way traffic is the key concept and distinguishing factor of Web 2.0. In this regard, Kaplan in [44] defined Social media as “a group of Internet-based applications that build on the ideological and technological foundations of Web 2.0, and that allow the creation and exchange of User Generated Content.”

Contribution of consumers in the process of social media has changed their nature from observers to actuators that can affect and shape the results and the process of social media. This difference alone has hit all of the main principles of social media namely programmability, popularity, connectivity, and datafication. For instance, programmability in mass media is an editorial strategy to glue channels to keep their audiences from one item to the next as a continuous flow by content manipulation [12]. However, when it comes to social media the code and users are taking the place of content and audience, and the one-way traffic is replaced with two-way traffic [83]. In this environment, users can form the stream of information by posting contents and voting to ascend or descend the priority of items.

2.4 Related Work

In dealing with a context-aware service, it is important to understand the nature of the context in the application that the context is going to be used. The better understanding of the context would help the designers to provide better support for the needed behaviors. This discussion of context is relevant to handheld computing since in handheld computing, the freedom of mobility of users is increased. The growth of mobility makes the environment of users much more dynamic, and users’ context such as their locations, and surrounding objects are changing more frequently. Therefore, supporting this dynamic environment would need an adaptive service that can provide necessary information related to the user’s context whenever necessary [11].
According to the definition of context given by Dey and Abowd [11], any information that is useful in the evaluation of an entity’s situation can be considered as context. In this definition, the entity is any object that is relevant to the interaction between a user and an application. Therefore, a context-aware application is using context information to provide relevant service to its users. With this definition in mind, we went back to augmented reality to see what would be the meaning of context in the AR application.

Augmented reality is referred to a technology that overlays virtual contents such as images, texts, graphical 3D models on real objects [16]. As for any context-aware service, the overlaid information are related to the context of the user’s task. Therefore, in any AR application, we have two concepts of context and content. AR applications are using context information to deliver content information to meet user’s need.

Generally, in AR applications the context would be some elements of a physical object, and typically those elements are locations of objects or visual patterns. The visual patterns can be natural patterns of objects or a QR code attached to an object. Also, in some platforms such as Vuforia, the visual pattern of an object is referred as a target. Both of the target and the POI are context information of an object (or a user) in an AR application. For simplicity, we are going to use the target to refer to both of them unless it is needed to distinguish a target from a POI. Contents, on the other hand, are the computer generated graphics/information that is going to be superimposed on top of the physical object.

The separation of content and context does not seem clear in some of the previous works. For instance, Grubert et al. have mentioned the content availability as a source of complaints about users in their report [35]. However, they have used POIs (which is a context information) to refer to content availability. This being said, Slabeva et al. in [77] have classified augmented reality as a context-aware service, and they have given a clear separation between context and content in a context-aware provisioning ecosystem. They have conceptualized a future context provisioning ecosystem shown in Figure 2.10 consisting of three clusters, context provisioning cluster, content provisioning cluster,
and network operator cluster. Network operator’s cluster handles the data connection to the end user. Location information providers, as well as social network site operators and sensor network operators, are in context cluster. This cluster also has a context aggregator which infuses all the context data coming from a variety of providers to provide a comprehensive context information on a specific user which is considered a significant added value. Content provisioning, on the other hand, includes a variety of content providers from broadcasters and newspapers to user-generated content and consumer opinion platforms. The content cluster also includes a content aggregator that bridges the content providers to service providers by delivering the contents from different resources to its consumers in several service providers.

![Figure 2.10: Context provisioning ecosystem](77)

The quality of experience of a context-aware service such as augmented reality
would heavily rely on the quality and quantity of context and content information. The survey report of Grubert et al. shows the content quantity and quality is one of the major sources of user’s complaints and their reason for abandoning the augmented reality application [35]. Hence, it is vital to investigate the reasons behind the lack of context and content information, mechanisms of sharing this information among providers and consumers, and methods of contribution in authoring and provisioning them.

We categorized our study and investigation generally into three sets of problems. The first category is the issue of lack of formats and standards for context and also not having enough context information. Same as context, for contents also there is not any widely-adopted standard format which makes contents platform dependent [36, 40, 50]. This problem affects AR by increasing the cost of application development and by reducing the number of targets for a single AR application. Consequently, AR applications lose AR users due to the shortage of targets.

The second type of problems is related to the architectures that do not provide effective ways of sharing the context and content information. As described in [77], the future ecosystem of context-aware service includes context and content aggregators. AR infrastructure as a context-aware service should include context and content aggregators too. Not being able to share AR resources does not directly affect the overall number of resources in AR. However, it reduces the accessibility of information for a single application, and subsequently, it leads to losing AR users.

Finally, bringing user contribution into AR is a recent research area of AR. In previous sections, we discussed that user contribution is one of the pillars in Web 2.0 and how much it has affected networks and media. In this regard, Schmalstieg et.al [40] have proposed AR 2.0 which is the integration of Web 2.0 and AR. AR 2.0 aims to enable user generated contents, information sharing, and massive-scale deploy of information. In this work, the researchers have named low-cost display platform, mobility for AR, glsbackend infrastructure for distribution of AR content and application, authoring
tools, and real-time AR tracking solutions as five key technologies needed for AR 2.0 to fully emerge. Although the intention of this thesis was not AR 2.0 in the first place, our investigation on the limitations of AR brought us to the same conclusion: current backend infrastructure is not providing an efficient way of distributing and sharing AR resources. We also, agree on the real-time tracking solutions. Therefore, in the next chapter, we are going to propose a new architecture capable of distributing and sharing AR resources and at the same time providing real-time search.

There have been several research studies on each of the mentioned problems of AR, and we could not bring all of them in this work. However, in the following, we are surveying a subset of the previous works based on the attention they have received in the literature by their number of citations or the novelty of the works and also the inspiration that they have had on our work.

2.4.1 Target and Content Related Issues

Gu et al. have emphasized on the proprietary formats, standards, and architectures as the primary source of problems for both communicating and sharing information as well as the wide-spread adoption of AR applications [36]. Hence, they have proposed a new open solution which includes open content format and flexible framework. The solution is based on the physical attributes of the objects and is called Cyber Physical Markup language (CPML). In analogy to the Web which URI is used to identify a resource, here they are using geographic location to identify a physical object. A sophisticated representation of visual features of an object is also being supported. The main features of their proposed work are: a) the ability of grouping multiple objects in one CPML page which allows scalability and flexibility, b) a convenient way of creating and editing CPML pages, and c) adoption of conventional protocols for the Web which ease the navigation system among pages and reduces the development cost for cyber-physical Web. The infrastructure of the CPML is shown in Figure 2.11.
Although CPML is building on top of the existing Web protocols, however, having a proprietary format for visual representation will add another format to the long list of proposed formats and exacerbates the situation. Another aspect of the proposed framework is tightly coupling targets and contents which is not necessarily a good idea. For instance, for a single target each user can get a different content depending on their application or their profile, and this is not addressed in the CPML. Also, their reliance on location-based services such as GPS, which is not always available especially in indoor situations, can be their major drawback. Another problem can be constant updates of the CPML pages due to the dynamic nature of the AR resources. For example, for mobile targets, their location should be updated in their correspondent CPML page constantly. Hence, the structure of the page should be revised as soon as the target leaves one group and joins another group.

According to Applin and Fischer [14], majority of the AR applications is based on the geolocation. Nonetheless, there are plenty of AR SDKs such as Vuforia, Wikitude, ARToolkit, ARmedia, D’Fusion [2, 13] and recent standards such as ARML 2.0 [1, 50] which support natural feature and visual search and detection of targets on a variety of devices from PCs to smartphones and tablets. Therefore, any proposed standard or platform for AR should consider both location-based and visual AR.
Besides the problem of proprietary formats, creating, storing, and sharing of targets and contents are another set of issues that have been addressed in [28]. According to the authors, a major challenge in the current applications is their need for a massive database of targets to recognize objects, and the current infrastructure is not supporting dynamic adding and sharing information (i.e. targets). They have offered a framework for AR in handheld devices which is shown in Figure 2.12. Using their proposed framework users can create, share, and import POIs which are basically in the form of XML files in the format of KHarMA [38]. Users can share the XML file and notify friends on their social network page of the newly created POI.

![Figure 2.12: System architecture for AR application [28]](image)

Fanjiang et al. provided a rapid way of developing AR applications and sharing POIs [28]. However, they have considered only location-based services for AR, and they have not addressed other types of AR. Also, it is not clear how the proposed design would support the already existing AR applications. It is worth to mention that the framework is not supporting communication and messaging between different AR platforms.
The problem of lacking targets and the target database scalability has also been aptly emphasized in [75]. Song et al. have used social network’s service to gather more images as targets. They have implemented a client-server based framework. The server side has three modules: image recognition module, social network service crawling module, and a database of image contents which is used as targets. Their proposed architecture is shown in Figure 2.13. In their proposed architecture, server-side gathers images from social networks which have textual annotations. The image recognition module communicates with the client to respond the client’s image based queries. The client-side app is a tool to query an image in the server side.

Their proposed structure is not supporting social networking in AR. However, it uses social network’s services (SNSs) to help scalability of the target database in AR. The work is aspiring because they have presented a new approach toward integrating SNSs in AR as an open platform. They have also used user participation for contents’ quality and quantity.

Figure 2.13: System architecture for mobile AR application[75]
2.4.2 Web 2.0 in Augmented Reality

In the case of integrating Web 2.0 in AR, many researchers have tried to enable social services such as social navigation, social network, and social media. Some Researchers have enriched the AR experience by bringing more contents and information from social networks into AR.

Kang and Hong defined AR as “technology used to make expressions by combining medium (text, image, sound, video, etc.) linked to the real world.” They argued that such expressions would put the human in the periphery and, because of its medium-center nature, essentially limits the expressions to the relation map around the objects. The authors believed that expressions that put human in the center would be more context-aware. Hence, media and objects would be expressed through contextual reasoning. According to the authors, services based on AR do not have the capacity of supporting dynamic behavioral changes made by users. Therefore, they suggested using social networks capabilities—especially the information stored in the profiles of SNSs—to enrich AR. They have developed a system which is capable of linking to Facebook, LinkedIn, and other SNSs and fetching location information and showing AR objects related to the users’ location information [43]. The similarity to our work is that we also agree the AR alone is defective when it comes to supporting communication and dynamic behavioral changes. However, we want to enable AR through restructuring its framework to be able to overcome those defects. Our approach is embedding social network services into AR rather than borrowing it from SNSs. They have also considered only location-based technology which would be a limitation because not all AR applications are using location based techniques. Another difference between that work and ours is in the fact that they have considered the IT device (handheld cell phone) as a Hub for information sharing and communication. In our framework, we have used a middleware between all app servers and content providers to play the role of an information hub.
AR has been used to connect people either by using some form of social networking or pointing to friend’s location on a 2D/3D map. This way of using AR is interesting for us since we have also offered this feature in our proposed framework.

A platform named SPORANGIUM has been presented in [54], using which it is possible to create ad-hoc networks and support the creation of sporadic social networks. The goal was to get the most from the people and the resources from the surrounding environment. SPORANGIUM provides a broad range of functionalities in different levels, namely, application, knowledge management, mobile cloud computing, and ad-hoc communication. The platform relies on an ad-hoc network on the first level, and therefore it is trying to establish connections proactively. Another aspect is the number of value added services that can be provided and shared even without the internet and just by ad-hoc networks for sporadic social networking. The researcher has used the museum as an example of an application for which users only need to install one application by entering the museum, and they can engage with people that are physically close (in the museum). However, the context is restricted to the location only. For example, if two people in different museums are interested in the same historical location or object, they cannot engage in communication using SPORANGIUM’s design. This is one of the most important points that we are trying to address in our proposed framework. Another problem with SPORANGIUM is that it only connects people who are using their platform (application), whereas in our proposed framework, we are trying to bridge between different platforms and applications.

MeetYou is another example of the applications that are using AR to connect people in close geographical range [73]. The software offers functionalities such as registration, login/logout, friend management, grouping users and assigning different parameters to each group, and notification if a member is close. Users can “check in” the application to let their friends to be notified of their frequently visited places. The novelty in MeetYou was that it notifies its users if a member of a group is nearby.

Hoang et al. used augmented reality to visualize trails of the visited locations by
friends [39]. A blue cone is an icon shows the visited location by a specific person illustrated in figure 2.14. If the user steps into the blue icon, he can call the person who has visited that location before and can start a conversation using VOIP (voice over IP).

![Figure 2.14: Visualization of the trail marks in AR [39]](image)

The Researchers have tried to support mobile 3D AR information for WEB 2.0. The assumption was that the locations of the friends were going to be available either by friends or by mining Twitter or Flickr, etc. This work is bringing information mined from social media i.e. the location of users, into AR. Then they were using the mined information to contact the users who were online and reachable through VOIP. Although the proposed method of using SNSs in augmented reality is exciting, however, it only considers location based AR. Also, it was not used for networking or communication with a group of people. The other aspect is about the implementation of their proposed method. It has been implemented on the wearable device, which is not convenient so much, and its technology is not prevalent.

In studying the previous researchers’ works, we saw two approaches regarding to mixing social networks and augmented reality. The first approach is trying to enable user communication and user profiling inside augmented reality. This needs to enhance AR backend infrastructure to be capable of supporting SNSs. This is the approach
we have also adopted. The other approach is using AR features to improve social networking environment and social network experience. For instance, De Chiara et al. have followed the latter [25]. Instead of enabling some form of communication between users in AR, they argued that it is possible to offer new interaction and communication techniques to social networks due to having mobile devices and hence mobile users. They have focused on the mobility aspect of the users with mobile devices. Their work has presented Link2U as an integrated solution which is trying to combine augmented reality with social networking to answer the need of information about the surrounding environment of the users. It offers functionalities such as messaging and calculation of road on a map and identification of other social network’s users and POIs inside AR. In Link2U, users were divided into contact lists. Upon users’ presence and location availability, other members could see the user on a map of the environment (shown in Figure 2.15a) or in the live mode (illustrated in figure 2.15b).

![Link2U map mode visualization](image1)

![Link2U live mode visualization](image2)

*Figure 2.15: Visualization in Link2U [25]*

Functionalities that Link2U offers are visualization of the connected people and route calculation toward a specific POI that can be a user. However, the system is not providing a communication module. Link2U is another example of a client-server based application.

Social service in AR has been used for learning purposes too. Social augmented reality (SoAR) is a framework that has been designed to enhance learning in construction
work. SoAR improves social interaction among peers with the focus on augmenting synchronous communication in response to new contexts [63]. The authors have categorized the found challenges into the emergent context (material shortage and access to context), synchronous communication (synchronous communication with the responsible people at the time of need), bi-directional content authoring (users should be able to generate and publish contents), and social interaction. SoAR enables the communication among the coworkers, and it augments that communication in the form of drawings on the screen. The provided functionalities of SoAR are professional profile building, instant messaging, and vision-sharing. The proof of concept for their framework was a Web-based application that works on mobiles using browsers shown in Figure 2.16.

Figure 2.16: Vision sharing feature for SOAR proof of concept [63]

Mixing real with virtual is a continuum, and there are spaces in between such as augmented reality and augmented virtuality shown in Figure 2. Jang et al. have tried to connect the virtual world with the real world through augmented reality and augmented virtuality [41] (illustrated in Figure 2.17). Their project provides the following functionalities: it maps the real world space and users into a virtual world and also augments the real world with the location of the avatars in the virtual world. Also, the message passing has been enabled between users in reality and the avatars in the virtual world.

Similar works have been done before such as cAR/PE [66] using which users from
different worlds can interact through a video conference. Other studies such as XIM [19] and TwinSpace [67] have tried to build an integrated world of real and virtual world. The problem with all these systems is that they have used a dedicated environment which makes it difficult to be widely adopted.

Researchers in [41] have designed a prototype system called SyncIS (Synchronized Indoor Space). SyncIS has the functionality of supporting location-based social networking to the public users which differentiate this work with works before.

From what reviewed we could see a need for an effective framework that can support user contribution in AR at the same time capable of sharing targets and contents between different AR platforms. Although there are standards proposed for content [38, 50, 55], still the majority of the contents are proprietary formats. Not being able to share the contents contributes to one of the most glaring issues of AR applications which is the paucity of sufficient targets in AR world. The shortage of targets does not entice people to use AR applications in their daily lives.

Many applications such as Life360 [7], LOCiMobile [8], FOURSQUARE [4], and Glympse [5] are offering location-based services that allow the users to share locations and share their path even communicate about places. On the other hand, applications like Layar [6] and Aurasma [3] are offering augmented reality. However, so far we have not seen an application that have combined them both effectively. Although,
communication by itself is not a new concept; forming a communication around a POI in augmented reality space is a novelty.

2.5 Chapter Summary

In the first two sections of this chapter, we overviewed augmented reality by providing its history, the way it functions, and its applications and challenges. We showed the backend infrastructure of a typical AR application and how it is incorporating client-server architecture to implement AR application. The problem of such structure could be the high cost of functioning and maintenance in addition to unfair competition advantageous for certain vendors.

In the rest of the chapter, we covered Web 2.0 and social services. We explained how Web 2.0 and social services are promoting the freedom of sharing and reusing of Web contents. Regarding social services, the effect of having a two-way traffic of data from producers to consumers and also from consumers to producers have been explained in detail. The rest of the chapter is surveying the research studies that have tackled many of the problems related to lack of standards, AR architecture and incorporating Web 2.0 and social services in AR.

What we present in the next chapter is a framework that allows users and developers to share targets from different platforms and provides an efficient way of communication among users of various applications.
Chapter 3

Proposed Framework

“It is not the beauty of a building you should look at; its the construction of the foundation that will stand the test of time.” — David Allan Coe

3.1 Basic Idea

By considering client-server architecture and the reviewed works discussed in the previous section, there are important aspects that are worth to highlight.

1. AR as any context-aware service relies on the availability of contexts (targets in our case). These targets currently are stored either locally on the storage of the device which is running an AR application or on a remote server of the AR application. One major motivation of our work is to make it possible to gather all these targets under a common framework and make it accessible to all AR applications. Obviously, we are not going to propose another proprietary format. Therefore our intention is to offer all of the services provided by the framework through Web APIs.
2. Currently, in most cases of AR applications, context and content developers are setting the stage, and AR users are merely consumers of data. Although it is possible to implement a client-server architecture that is capable of having bi-directional requests and interactions, this is not the case that happens regularly in current architectures. We intend to make it possible for users to contribute in creating and sharing targets and contents.

3. Social services have gained much success and attention recently. However, a reliable way of communication between users of different AR platforms has not been proposed. Users should be able to communicate with each other, review and vote, join and disjoin from AR social groups. User communication is another aspect of our intention behind our proposal.

4. Interestingly, all three points mentioned above are related to a greater concept in the Web, named Web 2.0. It is more than a decade now that Web developers are trying to harness the power of users’ contribution in the form of comments, likes, user profiling, content sharing, etc. Web 2.0 emphasizes user contribution and seeing the Web as a programming platform. Services scattered around the Web are getting integrated which adds value to the information. In this regard, there are efforts to integrate Web 2.0 in context-aware services. Looking from this aspect our proposal becomes an example of integrating Web 2.0 and a context-aware service (i.e. augmented reality).

5. Last but not least, we noticed that there are subtle assumptions or misconceptions of coupling the concepts of target and virtual content in many AR related types of research. Though this is not the case with all of the studies, here we want to emphasize that target and content can be decoupled, and each of them can be found in a separate place of the architecture of an AR application. Another assumption is that the provider of a target and the developer of content for that target are necessarily the same. We want to show by decoupling target and content; content providers would be able to develop their desired contents and deliver it to
their consumers without worrying about creating and managing a target. Hence, for one target there could be various contents and Content providers.

3.2 Limitations with Current Structure

There are limitations imposed on current AR applications due to the aspects mentioned above and present client-server architecture and implementation styles adopted by AR developers. These limitations are as follows:

(A) User interaction and contribution

An example of user interaction would be to imagine a person at a famous scene (like Vancouver’s suspension bridge), and he wants to send a message to users interested in the same target and share his idea or ask them to join him on a special activity. These kinds of interactions are not supported at the moment. According to Applin and Fischer [14], stories are fixed, single narrative and a group-oriented social experience augmented reality is missing. One reason of that as discussed before is because of storing targets on proprietary databases either locally or on cloud servers.

(B) Limited number of targets

AR applications rely on the targets and the virtual contents that are going to be superimposed on the detected targets. Research result conducted by Grubert and et al. shows that users are complaining not just about the shortage of targets around them, but also for not meeting their expectations [35]. Even the existing contents are not up-to-date in many cases. Currently, for example, every browser implements its own client-server architecture for the same application. Each implements its own set of targets and the virtual objects. We explain this problem by analogy from Web browsers. Suppose a person uses Firefox (an Internet browser) to reach to a Web address. The same address returns an error of “Server
not found” when used in Google Chrome. This is what happens with current AR applications. Even though each of AR applications has a database of targets, but in general there is not a standard way of sharing the targets. This is a huge drawback and plays a significant role in the lack of targets in AR world.

(C) Lack of contents and content sharing

Since it is difficult for individuals to create virtual contents in a global scale, there are not an adequate number of virtual contents in AR world in general. This shortage of virtual contents and contents authoring tools is one of the major drawbacks of AR systems [21, 87, 88]. Take the previous example and think after enjoying suspension bridge you want to put a note in the virtual diary of the place or you want to read other visitors’ opinions about the place. Since the diary is on a proprietary server, only a limited number of users would be able to access the diary. One way of coping with this problem would be developing the same content with several formats to support as many AR applications as possible. This approach would increase the cost of development [36]. Furthermore, it is not a scalable approach if there are many AR applications.

(D) Target and content naming convention

Consider a situation that a user has joined an AR experience and wanted to comment about his experience or put some likes under the target. The user would need a method to reference the target. To the best of our knowledge, currently, there is no naming convention for targets and contents in AR that can capture all of the resources in AR. In analogy with the Internet, URL (uniform resource locator) is used to reference and access a resource over the Internet. The same way we need a method to be able to refer to an AR resource. We are aware that reference to a resource and a mechanism to access that resource is not just about naming convention. However, to access a resource, of course, the first step is to have a unique address (or a name) for that resource.

The above limitations motivate us to look for a better model of AR implementation
that could enhance the user experience and expand AR applications. The enhancement in the proposed model is mostly on the server side. The servers of the applications especially, applications with common a platform must have the ability to interact among themselves and provide a comprehensive set of services to the users.

Managing a federated set of servers is a challenging task, but its advantage, we believe, outweighs the challenge. Several implementation frameworks of the proposed architecture are possible. One such implementation framework of the proposed architecture is given in the next chapter.

### 3.3 Client Federated Servers

Here, we propose a framework for AR applications shown in Figure 3.1. The main objective of the proposed framework is to eliminate or alleviate the aforementioned limitations of the current client-server model. It is an improvement and in some senses a generalization of the client-server architecture. We refer to this new improved architecture as **Client Federated Servers (CFS)** architecture.

The proposed framework focuses on the server side, and the client side can be the same as a typical client-server AR application. However, the proposed architecture has been designed to serve a broader range of clients including AR applications, target providers, content developers and any other Web services. This architecture is using Web APIs to receive and respond to requests. AR applications have their own target and content databases. However, using an intermediate target server i.e. “Target Hub” (TH), a server can reach out to other targets and contents that have been shared by other applications and developers. It is important to mention that content providers do not need to develop a whole application to share their AR contents. What they need to do is to subscribe and upload their contents to the target hub to make it globally accessible.
3.3.1 Description of Components

The proposed framework has the following components.

**Web API:** An interface with multiple predefined methods that exposes services and data, and provides a way of communication between the target hub and AR resource providers and consumers. It receives and responds with HTTP protocol using JSON format. Using API allows third parties to access to AR resources easily and plays a major role in realizing the intention of the proposed architecture. Documentation of these APIs is provided in the next chapter.
**Request Manager:** It plays controller role of the architecture. It includes logic, algorithms, and rules of the system. When a request is received in the Web API, the request is translated into a command for request manager. Request manager initiates a chain of method calls, message passing, data request, and data restoring to fulfill the request. Since the provided service is customized to the service requestor, any authorization and billing or customization happens in this layer. Therefore, it is vital to maintaining a profile of clients, and for any service request, client profile manager should be negotiated.

**Client Profile Manager:** This component keeps the profile of the subscribed clients that have registered in the target hub. A registration process starts with an HTTP request which is supported by the Web API. The registration request will finally come to the profile manager to check for name availability and other requirements. Clients can build their profiles by providing basic information including a unique name and a password. Clients also can specify if they allow their contents and targets to be stored in target hub or not (for licensing issue). Keeping a record of client profile would allow the target hub to customize its services in a smarter way. For example, in case there are multiple contents to be forwarded for a server request, target hub is going to decide which content should be sent to the server, based on the provided priority list.

**Target Manager:** It handles the requests for the targets and includes a database of targets and records of the requests for each target. Target providers can share their targets in the target hub with a unique name. A target’s name is composed of two parts. The target provider’s profile name which is a fixed and unique name for all the contents uploaded by that provider and a given name by the target provider. Target manager keeps the record of requested targets, the time and the number of times a target has been requested, the number of servers, etc. This information allows AR developers to have a better understanding of what users are more interested in and what their preferences are. These statistics are shared by target hub as added value service.
**Content Manager:** It is a database of the contents and information about each content. Content developers can upload their contents using the Web API to the target hub. Each content should at least have one target upon which the content is going to be overlaid. The content’s name similar as the target’s name is composed of two parts. The content developer’s profile name which is a fixed and unique name for all the contents uploaded by that developer and a given name by the developer. The content manager, also keeps the record of the requested contents, how many times content has been requested, how many servers have requested for a particular content, etc. This information is reachable by third parties for their analysis. By handling the requests and providing the virtual contents, the target hub is playing the role of a content aggregator which is an essential element in any context-aware service. Content consumers send their request using an HTTP request. The content manager is responsible for retrieving the content and updating the record of the content in the database.

**User Interaction Module:** All forms of communications between users including user’s interactions with contents and messaging between users are handled in this module. User interaction module keeps the history of communication for each target. End users start their communications through their service providers (an AR application for example). In case target hub has been adopted by the service provider, end users have the chance to communicate to other users around the world that are augmenting the same target. The interaction between users can be in voice, image or text form. This module receives the interaction requests from the request manager and processes the requests.

**Synchronizer:** Any update on a client’s profile, target’s record, or on a virtual content’s record has a possibility of creating inconsistency. For instance, for any content, there should be at least one registered target. If there is any content that has not been registered for a target, it is an inconsistency in the target hub. It is the synchronizer’s job to keep the records consistent. The Communication history also needs to be synchronized among all parties of the communication. This is
another responsibility of the synchronizer. The synchronizer also interacts with
the file system to categorize and aggregate contents and targets.

3.3.2 Benefits of Client Federated Servers Framework

In a big picture, the proposed framework is an intermediary that connects AR resource
providers and consumers and also it has the mechanism to support a limited interaction
among AR end users. This framework also plays the role of context and content
aggregator. Meaning that several types of context information of AR applications such
as points of interest and visual patterns in the form of targets are stored and provided
to AR resource consumers. The same is true for the contents; the proposed framework
provides a way that content developers can share their contents independent of the
targets. On the other hand, user generated contents are also stored and provided to any
content consumer. Besides considering the Web as a platform, user contribution has
been emphasized as one of the main objectives of Web 2.0. The new architecture makes
it possible for users of different platforms to communicate with each other and share
ideas about targets and contents. At the same time, users’ contribution in the form of
interaction with targets and contents such as voting and rating can be reached out by
any interested third parties. Especially, popularity rating has been mentioned as a key
element for social media. Enabling this feature in AR increases the chances of using AR
as a platform for social media.

Two main material in AR are targets and contents. As mentioned in previous sections,
lack of targets and contents has been a problem in AR for a long time now. Since app
servers can communicate and register their targets and contents in the target hub, the
most important immediate benefit of the proposed framework would be increasing the
number of targets and contents.

We believe there is no need to hardly couple a content to a target. We have em-
phasized this conceptual separation by having a separate component for each of them
in the proposed architecture. The proposed architecture allows content developers to deliver their contents to the content consumers without worrying about the complexity of developing a full AR application. Simultaneously, target providers need not to necessarily develop contents for their targets to be reached by end users. A popular place is worth to be added as a target to the target hub which motivates content developers to provide contents for that popular target. On the other hand, a popular content such as Pokémon’s (Pokémon Go game’s monsters) can raise interest to a target which in this case is a location. As we know, people go to these places to hunt some Pokémon’s. This environment would create a positive synergy among end users, target providers, and content developers to encourage each other to create more contents, provide more targets and start to use AR more than before. Having a way to know which targets and contents are more popular than others would potentially encourage developers to adopt the formats of those targets and contents. This, in turn, would open a way toward converging to a limited number of AR formats based on their usability and popularity. The fact that each application has only its own database of targets and contents gives a very restricted view of the world, especially for outdoor applications. AR users would need to switch between applications from place to place. Availability of more targets and contents in one application can reduce the need for switching between AR applications and results in a more intuitive way of using AR technology.

Another outcome of the proposed architecture is its naming convention for AR resources, which provides a unique reference to any AR resource in the target hub. AR is a context-aware service with many resource providers from one hand and many resource consumers on the other hand. Still, no resource referencing method can identify a unique AR resource similar to URL.

Finally, we believe our work has the potential to open up new paths for future of AR. Some of the concepts that are introduced or significantly emphasized by this work are naming convention (or uniform referencing method) for AR resources, decoupling targets and contents, the analogy of Augmented reality browser (ARB) and internet
browsers. In short, the outcomes of the proposed architecture are as follows:

- Sharing targets
- Sharing contents
- Connecting app servers, developers and Web services
- Separating content developing complexity from AR application development
- Decoupling target and content
- Unique naming convention for AR resource referencing
- Context and content aggregator
- Web 2.0 in AR (the Web as a platform and user contribution)
- Communication among users of all platforms
- Recognizing popular formats and platforms and a way to converge toward them
- Potential to open up several new paths for future AR including naming convention for AR resources, decoupling targets and contents, comparing ARB and internet browsers.

3.3.3 Practical Scenarios

Here, we want to show the problems we are concerned with and solutions we propose in the form of practical scenarios that can happen in our real life. Following scenarios are from different prospective.

**Scenario A:** Jules is working with an X-Ray machine with which she is not familiar. She needs help with the instruction of the machine. She is using an AR app that has helped her with other machines before. She runs the AR application and points
her phone to the machine, but apparently, there is no information available for this model of the machine. With the target hub implemented and adopted by the app developer, now she can search a couple of tags such as the machine’s make and model and instruction. What she is supposed to receive is shown in Figure 3.2. For each target that she selects there can be various contents from different content providers. Each of these contents has their features and specifications such as price, popularities (number of stars), and description of the content (Figure 3.2). Jules can install and preview any of them, and if she likes, she can buy and use them. She also can enter to targets’ social room and read or write comments about her experience.

<table>
<thead>
<tr>
<th>Search result for Machine Name, Instruction</th>
<th>List of Contents for each Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Target A</td>
<td>-Content A:</td>
</tr>
<tr>
<td>Description:</td>
<td>Description: Free, ...</td>
</tr>
<tr>
<td>-Target B</td>
<td>-Content B:</td>
</tr>
<tr>
<td>Description:</td>
<td>Description: $5, ...</td>
</tr>
<tr>
<td>-Target C</td>
<td>-Content C:</td>
</tr>
<tr>
<td>Description:</td>
<td>Description: $2, ...</td>
</tr>
</tbody>
</table>

*Figure 3.2: Searching for target and list of available contents for a target*

**Scenario B:** Jane is a computer graphic developer. Recently, she has decided to develop for AR applications. She wants to start her work by developing for popular platforms and targets. She also wants to advertise her design and get feedback from users. Using target hub, she can get a list of most popular contents and targets for each platform. She now knows that there are requests for a newly developed X-Ray machine, but the contents developed for it are not very helpful. She decides to develop a good user-friendly content. The content that Jane is developing here probably will be used by Jules.

**Scenario C:** John is a technician in a medical hardware manufacturing company. His
company has recently introduced a new X-Ray machine with higher capabilities than the previous models. The problem with the new machine is that it has a complex instruction guideline due to having so many features. John is very familiar with the machine, and he can guide the users interactively. He decides to use his AR application to make a target of the machine. Since John’s AR application is using target hub, he now can read comments about the machine and answer questions and help the users.

3.4 Data Flow and Connection between Components

To get into details of the system, we start by analyzing the processes and information flow. The flow of information between main processes of the system is shown in Figure 3.3. Target providers and content providers are sharing their products using HTTP requests. On the other hand, AR applications are using those targets and contents by providing information of their requested targets and the contents. AR applications are also sending and receiving end users’ communicated messages. All of the requests would initiate a request handling process. This process interacts with different modules of the system to handle the request which will be discussed later. If the request is for adding a new content or target to the database or (basically any update to the database), the request needs to be passed to synchronizing process. Requested target or content is provided to AR application directly by the target or content managing module. Communication history between all of the communicators needs to be monitored and synchronized. Therefore, any update on messaging history would go through a synchronizing process. This process checks the message info of the last message and number of communicated messages to keep every part synchronized. Third parties are the ones who desire to get context and content information or reports for their purpose of use. Third parties would need to provide their client info, and after their request is processed, they will get the report. Such reports are generated from databases. Generally,
for report purposes there are replicated databases not to hinder the production database. However, for simplicity, we are not showing replicated databases. Admin of the system can configure the system and generate reports of the system.

For the sake of simplicity we did not put the end user’s interaction in the Data flow diagram shown in Figure 3.3, because in a conceptual view, that type of interaction is handled within AR applications. However, in a less abstracted view, end user’s interaction is shown in Figure 3.4. End user are using their AR applications to browse the world for targets. On the other side, AR applications are providing targets and contents to the applications. In order to have an exciting AR experience, there needs to be enough number of high-quality targets and contents. AR applications are uploading their targets and contents to the target hub. Simultaneously, AR applications are subscribing to the provided services.

When there is no content for a target that has been requested by the user, the AR application server sends a request to the target hub. The request has information about the target that includes tags for the target, platforms specific information and some requirements for the content. One important module in the proposed framework is the module which plays an intermediary role between app servers of different applications. The target manager receives requests for targets from app servers. It keeps a database of the targets. Records of such database include ID, target name, target provider’s info, targets files, target platform, popularity, description, tags, subscribed clients, active connections, and a list of provided virtual contents.

Content manager is keeping the record of information about contents provided by content developers or AR applications. When a new content is uploaded to the content manager, it has a record of information about the content and the targets that it supports. This record includes ID, the name of the content, the supported platform info, list of targets, popularity, description, price, tags, and a list of active connections. One of the services that subscribed clients would receive is the notification of a new content. Each time a new content is uploaded for a target, all the clients that have subscribed for that
target will be notified of the new update. When a user starts to put some comments on a target or content, the message is being sent under the profile of the AR application from which the end user is sending his message. This creates an active connection between the AR application as a client and the target that user is augmenting at that moment. Any new interaction is forwarded to all active connections. This way when a new comment is received in target chat room, all of the users that are augmenting that target would immediately receive the message.
Figure 3.4: Information flow for end user’s interaction
3.5 Main Processes

So far, we introduced the target hub and gave an overview of the system and the main idea behind it. We also presented the entities interacting with the system, the main processes and information flow with a high-level data flow diagram. In the following sections, we get into the details of each main processes of the proposed architecture. We are trying to explain how each part of the architecture is cooperating to deliver a requested resource or service. Main processes of the system are shown in the Use case diagram in Figure 3.5. Communication between target hub and clients are HTTP messages. The details of the HTTP requests and responses are covered in API documentation section of the next chapter.

![Figure 3.5: Use cases of the target hub](image-url)
### 3.5.1 Registering in Target Hub

To get any service from the target hub, the service requester should register in the target hub first. The process starts by sending an HTTP request with specific parameters to the target hub’s address. Typically, a client would use an internet browser or a custom developed function in his server to generate and send the HTTP request. Target hub receives the request and checks the client database to see if there is any client by the provided info already registered. If there is no such client, client profile manager would create a new profile and returns the registration confirmation and a Token ID. The token is going to be used for all the future requests that are coming from that client. Aside from minor technical details, this covers the registration process that is depicted in the Activity diagram in Figure 3.6.

![Figure 3.6: Registration - Activity diagram](image-url)
3.5.2 Sharing Targets and Contents and Subscription

One main objective of the target hub is to create a bridge between AR resource providers and resource consumers. For this to happen, the resources (i.e. targets and contents) need to be uploaded to the target hub and shared. A registered client can start this process by an HTTP post request to which the target or content has been attached. Since the process of uploading and sharing a target and a content are fairly similar, we are only presenting the target sharing process for simplicity. A target can be uploaded without being shared. It would mean that other client can see that the target exist and they can read its description, but they cannot download and use it. However, when a target is not shared, it cannot be subscribed to notification services. If an AR app developer wants to share his targets and get notification whenever it is downloaded, or a new message has arrived for it, he will follow the process shown in Figure 3.7.

![Figure 3.7: Target Sharing - Activity diagram](image)
3.5.3 Searching and Loading Targets

Clients can search targets by their name which is a unique identifier, or by target’s tags which are assigned by its provider. The result of the search for a target would be a list of JSON objects. Each of the JSON objects is a target composed of information about a target’s name, size, platforms, shared status, etc. Clients might have different purposes for such a list of targets. One scenario can be an AR application’s end user who is browsing a target and tries to get some new contents for it. She would start her quest by entering some tags such as the keywords for the target she is augmenting. Those tags would be sent to the application server. When a request for a target comes from a client to its app server, the server looks for the target in its own local target DB. Generally, a target is stored in the form of patterns and a descriptive language such as XML. If there is no such target in the local target DB, the server generates a request for the target by wrapping the requested target info into JSON objects and sends it to the target hub. After authentication and authorization, the target hub returns the list of matched targets information. Now, the user knows about the targets that she can get. Furthermore, she can read some descriptions of the target and its correspondent contents, as well as price tags and reviews. Target hub’s clients can directly request to download a target, whether they will get the target depends on the sharing status of the target and the client’s authorization though. If the end user decides to download any of the provided targets, her request will reach to the AR app server first, in case the target does not exist in the local database, that request is relayed to target hub. Target hub updates the target’s record first. One record for each request should be logged, and then the requester client should be added to the subscribed list of the target. Then, the target’s information and all of the files (if there is any) are sent back to the requester (i.e. the application server). This process is illustrated in detail with an activity diagram in Figure 3.8.
3.5.4 Chat Rooms and Communication Handling

There are two types of interaction messages. The first one is between users, and the other is for chat rooms. Typically, the AR application sends both types of the messages to the target hub. Target hub uses the parameter settings of the message to determine if the message is meant for an end user of an application or a chat room of a target. A communication message is carrying information about the history of a chat such as the number of communicated messages, ID of the last message and some other information. All these information help the synchronizer to keep the history of the communicated messages synchronized between all of the communication parties. If the message is for a target chat room, message synchronizer updates the chat history of the target’s chat room and sends a new notification message to the subscribed clients. If the message is meant for an end user, the synchronizer updates the communication history of the two AR applications. Then, the communication module sends a notification to the destination AR application with the information of the source AR application. The
source is the sender application and the end-user (of the application) from whom the message is coming. The destination is the end-user that the message is meant for her. The process of this message passing has been depicted in Figure 3.9.

![Communication Process Diagram](image-url)

*Figure 3.9: Communication process - Activity diagram*
3.5.4.1 Subscription and Notification Handling

In the previous sections, we have used subscription and notification concepts without going into the details of their meaning. The general idea is to signal the client about the event in which the client is interested. The event can be the availability of a resource or information. Clients can be any of the AR application, target providers, content developers, or third parties. In general, signaling a client can have many forms such as setting a flag, sending a text message, or a Push notification service. The problem with flag setting techniques is the bandwidth and computation power waste. If the target hub sets a flag to signal the availability of a resource, the clients need to poll that flag in every particular period constantly. SMS might be a good idea, but not when the client is not a person and not for fast decision making situations. Pushing notifications seems like a good approach. However, we need an approach that supports all of our clients. So, at the implementation level, it should be decided to take one or a combination of these approaches for signaling system. In any case, the clients need to inform the target hub of their interest to be notified of the desired event that is called being subscribed to that event. For an application subscription, the best way would be providing application server address and to support some callback APIs for notification and message passing.

Clients of the target hub can be subscribed to several types of events. These events and their meaning are as follows:

- Target subscription: By subscribing to a target, the client allows the target hub to notify the subscriber if:
  - A new message arrives for the target’s chat room
  - A new content is available for that target
  - A new target with the same tags is available (less likely)
  - The target is downloaded

- Content subscription: by subscribing to a content, the client will get notifications
if:

- There is any update to the content by its owner such as its price
- A new content for the same target is uploaded
- The content is downloaded

• Messaging subscription: only an application can be subscribed to this service. The application would need to provide a call back address with the parameters specified by the target hub. If any new message comes for an application, it will be authenticated and redirected to the provided address.

3.6 Specifications

Here, we want to summarize the specifications and features of the proposed architecture. The followings are specifications of the framework along with reasons explaining their necessity and benefits.

1. Profile name is unique in the target hub

Profile name should be unique to identify the clients.

2. Target’s name and content’s name are unique

This is to create a way to address and identify AR resources in the target hub; there is a need for a unique identifier and a naming convention. Inspired by URL, Target’s name is composed of two parts, the target provider’s name and a given unique name by the provider. The same applies to the contents, but we are covering only targets for simplicity. Target provider’s name is its profile name, which is a unique name in the target hub. Using profile name for the targets allows all the targets from the same provider to be grouped under the same name. Grouping targets and contents enriches the search method and improves the accessibility of the targets and the contents. The same analogy is all the resources
of a Website has the common domain name. For instance, if there is a case that there are multiple targets and the user is only looking for the targets that are provided by a particular developer, the user can search the developer’s name to find only his products. Another scenario is when many contents are suggested for a user, and she is not familiar with the contents. There she would see the providers name before the content and it can be a useful piece of information.

3. Communications are authenticated by token

After being registered in the target hub, the client receives a token. For all the future interactions that token is going to be sent to the target hub for authentication purpose. The client can reset his token by calling the correspondent API.

4. There can be multiple contents registered for a target

This will give the option to the user to choose which contents she wants to get. From the developer perspective, it gives more opportunity to develop contents for the targets that already have a content for it. From target provider view, it creates a competitive environment among content developers to be able to sell their products that in turn helps a target provider to get the best from the situation.

5. A content is registered for only one target

A content is subjected to constant updates to satisfy its users and is customized to its target. Therefore, it is not a good idea to assign a content to multiple targets. However, a content can be assigned to multiple targets under different names.

6. There is one chat room for every target

For any target in the target hub; there can be a chat room in the target hub under the target’s name. However, the chat room is not created until target hub receives the first message for that target.

7. There is one chat record for any two AR application

Any two AR applications have the option of supporting communication between their end users. Communication history between any two applications is stored
in target hub and is reachable by the applications. The notification system also supports this type of communication. However, the record in the database is not created until the first message is received from an application.

8. All database updates are going through the synchronizer

Synchronizer generates correspondent changes that should be applied to keep the database consistent. All the necessary updates are sent to the update method in the controllers. The method applies the changes to the correspondent model. The reason of aggregating the changes to one method is that either all or none of the updates should be committed on the database. This keeps the database consistent.

3.7 Chapter Summary

This chapter started by giving the basic idea of what is missing in AR and how we intend to solve it. We covered the goals of the thesis as it follows:

• Bringing AR resources under a common framework and making it accessible to AR applications

• Having bi-directional traffic to enable users in creating and sharing targets and contents

• User communication, content review, polling, tagging, join and disjoin from AR social groups

• Incorporating Web 2.0 in AR

• Clarification of AR basic elements, namely content, and context

In addition to the goals, we listed limitations of the current AR applications under four categories of user interaction and contribution, a limited number of targets, lack of contents and content sharing, and target and content naming convention.
We introduced client federated-server model as the main contribution of this thesis. Practical scenarios of this chapter show the efficiency of the proposed approach using which users would be benefited from AR more than before. We discussed the details of the main components of client federated-server model. Also, the static and dynamic relation of different modules were illustrated with class and activity diagrams. In the end, we listed some specifications of the proposed model.

We showed the feasibility and the functionality of the proposed structure by implementing an application as a proof of concept. In the next chapter, we are introducing projects that include the implementation of our framework, an AR application that works effectively with the target hub and a subproject that shows how expiration tag can be implemented and exploited in the system.
Chapter 4

Scratcher - Proof of Concept

The validity and capability of the proposed framework are demonstrated here through a prototype application that we have named it “Scratcher.” Scratcher can show main functionalities of the proposed architecture including user interactions, target searching, sharing functionalities, chatting, and notification capabilities.

The application has been implemented on the most common platform of the current AR industry, which is smartphones. Although the proposed framework is a general idea that covers other types of platforms, the mobile augmented reality is in the focus of AR industry at the moment. We are also using the Android system, which again is a great portion of the smartphones industry. Scratcher has been designed in Unity using Vuforia plugin, and for the server side of the application, we have used ASP.net and Microsoft SQL Server for its database. Unity allows us to compile the application for different platforms including Android, IOS, and Windows.

We have implemented target hub using Model View Controller (MVC) framework. We have used ASP.NET, C# and Microsoft SQL server in the implementation of the target hub.

In the following, we are presenting the details of the whole system divided into three sections of target hub, app server, and the mobile application. We have tried to keep
each of these entities independent from each other at all levels of the implementation. This separation allows Adoptability and scalability of the proposed architecture.

### 4.1 Mobile Application Implementation

We have implemented the application in Unity which is one of the most powerful and popular game engines. Unity 3D is a cross-platform environment which allows compiling the final product on different platforms such as Android, IOS, and Windows. Figure 4.1, shows the working environment of the Unity with a green box as a 3D model. This model represents a content that can be augmented on a target. In the lower side of Figure 4.1, scripts are red boxed. The scripts are codes that define functions and behaviors. By attaching a script to an object, we can give behavior to an object. A developer can choose any of the JavaScript or C# Script to develop his code. We used C# script because we were more familiar with it.

![Figure 4.1: Working environment of Unity](image-url)
Unity alone does not support necessary functions for augmented reality. There are multiple plugins that we could use to support AR in Unity. We decided to use Vuforia which is a software development Kit (SDK) that supports augmented reality. Vuforia allows positioning a complex 3D object on an image targets.

The client-server architecture and effective implementation of the application enable the application to request for a new target database, and the app server also is capable of responding and returning a new target database. The application also gives a way of searching and selecting the targets using their tags and names. The lowest level of interaction between the application and application server happens when the user starts to send and receive messages in the targets’ chat rooms. The hierarchy of the interactions between the application and the app server is shown in Figure 4.2.

![Interaction hierarchy between client and app server](image)

**Figure 4.2: Interaction hierarchy between client and app server**

4.1.1 How Does It Work?

The application we have developed is using a client-server architecture. It interacts with the end user on one side and with the application server on the other side. As any
AR application, there should be a set of targets and some contents to be augmented over those targets. We have created a default database of targets that are going to be downloaded on the device as soon as a user logs into the application.

After the user runs the application, she is going to be asked for her username and password (shown in Figure 4.3a). The application has been designed to check for internet connectivity and username and password matching with proper prompting messages to guide the user (Figure 4.3b). As soon as the first page loads, the application starts to ping the app server. If the server is not reachable, the user will be notified to check internet connectivity, or the app server is down.

![Username checking](image1.png) ![Password checking](image2.png)

*Figure 4.3: Log in page of the Scratcher*

By clicking on the start button, the next page loads, which is an AR camera. As soon as the page loads, a Co-routine starts to download the default database of the
application. Now user can start to browse the objects and particularly images to see the augmented models upon them. The whole process of activating the AR scene has been illustrated in the activity diagram shown in Figure 4.4.

Vuforia has some very popular benchmark targets namely: stones, chips, and tarmac which we used them in our demo execution. In the scenario that we are presenting here, we are showing that the application can detect the chips target and augment a red box upon it. However, in the same scene, there is another target (the stones target) that the application is not showing any reaction to it, meaning it is not being detected (shown in Figure 4.5).

In the scene shown in Figure 4.5, if the user clicks on the scene, she would enter to the chat room of the chips target which we will discuss about chat room functionality later.

If the user decides to find a target, all she needs to do is to get into the search page. To load the search page, the user needs to click any place on the AR scene when there is no target has been detected. There she can enter the appropriate tags and search for a target. The list of the targets is loaded to the drop-down menu under search button. If there is any target returned, she can select that target and load it. She also can go back to AR scene without loading any target or quit the application from this page. The tags that the user enters work with a logic of “OR,” so if any of the tags hits a target, the target is going to be listed. Figure 4.6 shows that the user has searched “lab” as a tag and the target named “tarmac” has been retrieved from the app server. Also, the user has selected the retrieved target (tarmac), and it is ready to be loaded.

By clicking on “Load Target” button, the application sends an HTTP request to the app server requesting the target. The important point is that the AR application and end user are not aware of the source of the targets, whether it is the app server or the target hub. As soon as the new target is downloaded, the app can detect both of the targets as shown in Figure 4.7. The Tarmac target is on the left side upon which a green
Figure 4.4: Activating the AR scene

sphere is overplayed.
Figure 4.5: Chips has been detected but stones not

Figure 4.6: Target search and load page
4.1.2 Chat System

The main idea behind the chat room for a target is to connect users that are augmenting a common target. For instance, if two persons are looking at a similar target, they should be able to start a communication (shown in Figure 4.8). When the user clicks on a target, the chat room of the target will be loaded. Implementing and managing a chat room can have multiple aspects, including sending and receiving a message, storing and retrieving the messages, synchronizing chat history among the parties of the communication, and notifications. We are covering these aspects in the following sections.

4.1.3 Storing and Retrieving

All chatting history are stored in an XML file, in the local device under the target’s name. A sample chat history for a target with the name of tarmac is shown in Figure 4.9. The number of messages stored in the XML file is three, and the content of the last communicated message is also stored in the XML file. This information is going to be
used for synchronization purpose. We understand that if two chat files get corrupted, and they both have the same number of messages and the same last message, our implementation is not able to detect that corruption and does not fix it. However, we think this is good enough considering that we are implementing only a prototype and the probability of such corruption scenario.

By entering the chat room of a target, the communication history is loaded into a list of messages where each message becomes an object of the message class shown in Figure 4.10. Then the chat room page loads on the screen and the user see the messages and can send a text message. We are loading all of the chat histories in a scrolling view, but it is possible to load only recent “n” number of messages (Figure 4.11A).
4.1.4 Sending and Receiving a Message

When a user sends a message the color of the message is gray, and the font size is smaller than other messages until the message reaches to the server and an acknowledge message is received in the app. Figure 4.11B shows a “test” message that has not been acknowledged yet.

We are using the WWW class of the Unity to send our chat data to the server. This
class can be used to send both GET and POST requests. A chat message has multiple fields including Message type, Target name, sender, receiver, body, the last message of the chat history, number of messages in the history, and other controlling fields.

To receive a message, one way is to poll the server to see if there is any new message has arrived. Polling is sending a message to the server every short period. In response, the server replies an update message if there is any new message or replies an empty message in case there is no new message available for the client. This is a resource-consuming method, and it is not real-time. The other approach is to use push technology in which the server sends the update to the client as soon as there is any update available. The problem with this technology is either they are not supported across all platforms...

Figure 4.11: Chat room scene
or too much complex for our prototype system. We have implemented our notification system using long polling which is in between of the simple polling and server push technology. Long polling is not a push technology. However it emulates the push mechanism, and it has more flexibility in supporting Https and security policies. The sequence diagram in Figure 4.12 shows the difference of polling and long polling. With long polling, we have real-time communication, simplicity, and flexibility at the same time. However, there is a slight cost regarding traffic and reconnection handling.

![Figure 4.12: Polling VS Long polling](image)

### 4.2 Server Side Implementation

The server is a web application that communicates with the AR application from one side and the target hub on the other side. The interactions between the app server and
the target hub are in a three-level hierarchical model that is shown in Figure 4.13. The first level is the interface which is about registration and authentication. The second level is target handling, and the lowest level is the communication that happens over each target.

![Interaction hierarchy between app server and target hub](image)

*Figure 4.13: Interaction hierarchy between app server and target hub*

We have implemented our server using ASP.Net. To show that target hub is truly capable of connecting to different AR application and its users, we are using two different app servers each of which has its own AR application that is interacting with its correspondent server. We have uploaded our servers to the Microsoft Azure platform on the following two addresses:

**Server A:** [http://arapp.azurewebsites.net](http://arapp.azurewebsites.net)

**Server B:** [http://gece-ar.azurewebsites.net](http://gece-ar.azurewebsites.net)

The server is using web methods to interact with the AR application. The first method that an AR application calls after launching is ping method. This method is used to check if the app server is reachable. The next method is about checking for username and password. Figure 4.14 shows the methods of “Ping” and “CheckPass.”
The server is using “WebClient Class” class to implement the interaction methods. The first interaction of any server with the target hub would be registration. The target hub needs to receive three parameters of a server name, a server identifier, and a server address. We are using the server address as a callback address. The target hub is going to use this address to send its requests and notifications. The target hub returns an ID to the server. The returned ID along with the server name is going to be used to authenticate this server in the target hub for all future interactions.

It is possible that the application developer wants to register his server in target hub twice with different configurations. However, if he intends to use the same server name and the same server address, all he needs to do is to register in the target hub again with the same server name and server address but with a different identifier. It will let the target hub to generate another unique ID for this server, and the target hub would keep two separate profiles with similar server name and address but with different configurations. This feature allows the app developer to categorize his users.
and provide profile based service to the users. For instance, the request of free service users is going to be sent to the target hub with an ID different from paid service users. In the target hub, requests authorization is handled based on the profile of the requestors (i.e., ID of the users). Therefore, if there is any service that has costs for the app owner, he will not be charged for his free user’s access. Implementation of the registration function is shown in Figure 4.15.

```java
public static string Register()
{
    Global.Refresh();
    string Response = "";
    int ID = -1;
    using (webClient request = new WebClient())
    {
        string urlAddress = string.Format(Global.TargetHubAddress + "/server/register?server=0&identifier=1&address={2},
        Global.ServerName, Global.Identifier, Global.ServerAddress);
        Response = request.DownloadString(urlAddress);
        if (Response.Contains("ID:"))
        {
            Response = Response.Replace("ID:", "");
            Response = Response.Replace("\", "");
            int.TryParse(Response, out ID);
        }
    }
    if (ID > 0)
    {
        Global.MyServer.Registered = "true";
        Global.MyServer.HubID = ID;
        db.Servers.FirstOrDefault().Registered = "true";
        db.Servers.FirstOrDefault().HubID = ID;
        db.SaveChanges();
        Global.Refresh();
        return "Registration was successful" + "<br />";
    }
    return "Registration failed" + "<br />" + "Hub replied:" + Response + "<br />";
}
```

Figure 4.15: Registration method of the app server

If a user wants a target that does not exist on his app, he is going to search for the target using some keywords as tags for that target. The server of the app is going to search those tags in its target database and also forwards the request to the target hub. The target hub replies the name of the targets that has any of those tags. Then, the server replies all the found targets whether local targets or targets on the target hub to the app. Implementation of the search request from app server to the target hub is shown in
Figure 4.16. If the user chooses to download any of the targets, the target is downloaded first to the server (in case it is on the hub) and then it is forwarded the application.

```csharp
public static List<string> GetTargets(List<string> tags)
{
    using (WebClient client = new WebClient())
    {
        string uriAddress = string.Format(Global.TargetHubAddress + 
            "/Target/GetTargets?Identifier={0}&ID={1}", 
            Global.Identifier,Global.HubID);
        foreach (string s in tags)
            uriAddress += "&Tags[]=" + s;
        var res = JsonConvert.DeserializeObject<List<string>>(Encoding.Default.GetString(client.DownloadData(uriAddress)));
        return res;
    }
}
```

Figure 4.16: Requests target hub for list of targets

The last and important part of the server side implementation is about chatting module. The server side of chatting module has been implemented in three sections. The first part is an interface that interacts with the application. The second part handles updating the communication histories, subscriptions, and notifications that as discussed before has been implemented by long polling technology. The last part is responsible for forwarding messages to the target hub if it is needed. The server is expecting to receive two types of messages from the application namely “last message” and “sent message.”

“Sent message” informs the server that a new message is coming from the app server meaning that one of the users has sent a new text message. The server updates the communication history of that chat room and then sends the update message to all subscribers of that target using the long polling technology. If the target is shared with the target hub, an update message needs to be sent to the target hub. This message refreshes the chat history in the target hub. Figure 4.17 shows how the app server sends the new message to the target hub.

The “Last message” is the long polling’s request message which carries the last
communicated message of in a chat room. This message requests the server for any update. The server is not going to reply to this message unless there is any update for this request. Otherwise, the server is going to keep the request until it expires. The expiration time for an update request is a predefined and equal amount of time in the server and the application. When the application sends an update request, it sets the expiry timer for that request. The same way, when the server receives an update request, it replies the update if there is any new message. Otherwise, it sets the expiry timer. The activity diagram in Figure 4.18, shows the whole chatting process that expands from the user to AR app, app server, and the target hub.

4.3 Target Hub Implementation

Target hub is the heart of our proposed architecture. Surely, there can be many ways to implement an idea. The most important factor in our implementation was to apply the necessary functions of the architecture to show that it is feasible (practical and functional).

We have used MVC framework in the implementation of the target hub which emphasizes separation of the model (M) as data of the application from control (C) as the logic of the application, and these two are separated from view (V) as the interface.
Figure 4.18: Activity diagram of the chatting system

of the application. We do not have a view for our target hub, but methods in controllers are exposing the APIs necessary to interact with the hub. Our prototype target hub is being hosted on Microsoft Azure platform on the following address.

Target hub address: http://arconnect.azurewebsites.net

To manage models of the system without being concerned about underlying database tables and columns, we have adopted the entity framework. Entity framework is part of .Net framework, although it has been separated from .Net after version 6. With the entity framework adopted, our main concern is the logic of the application that how data is processed and manipulated and what are the relations between entities. For instance, when an update happens to an object of a model the entity framework will update the database accordingly.

Main models in the target hub are target, server, tag, subscription, target request,
target request type, server request, and server request type. We have not implemented content because the mechanism that is managing the target works the same way for the content. Therefore we have only implemented the target management and model of the target. The models are represented as classes, and for each class there is a table in the database. The relation of the entities is shown in Figure 4.19. For each target, there can be multiple tags. The tags are used as keywords to search a target. There can be many ways to search a target. However, there are various formats for different targets. Therefore the only uniform searching method for targets would be using tags. A tag is nothing but a label attached to a target. For each target, there can be multiple incoming requests. To generate reports and keep track of the targets, we want to log each request. Also, there are different types of requests.

The server is the other important model of the system. Each server can subscribe to multiple targets, and multiple servers can subscribe to one target. Therefore, we have to normalize the relationship by using a subscription model. The subscription helps when an update happens on a target. For example, if a new message is received for a chat room of a target, all of the subscribed servers need to be notified of the new message. Each server also can have multiple requests with different types. Request types include download, upload, register, unregister, update, message, etc.

The controllers for the models are responsible for the user interaction and working with models. We have implemented the APIs in controllers. For example, to register a server in the target hub, a request should be sent to the controller of the server. Register method in the server controller will check the parameters first and then adds the server to the list of servers. The code for register methods in Figure 4.20 shows how the server controller exposes register API and handles the request.

The other important controller of the system is target controller. Which exposes APIs such as “GetTargets,” “Download,” “Upload,” and “ForwardMessage.” One of the important modules of the target hub is handling the messages of the chat rooms which we have discussed in the previous sections. Figure 4.21 shows the forward message
Figure 4.19: Entity relationship model of the target hub

method with Identifier and ID to authenticate the request and the target name to find the target, the username of the sender, and the body of the message as input parameters.
Figure 4.20: Register method in server controller

The method checks if the server has been registered or not. Then, it finds the right target in the database, and finally, it forwards the message to each of the target’s subscribed servers. The method replies by a string showing the result of the request. Users of the Scratcher application on server A can communicate with users of a different application on server B because of message handling of the target hub.

So far we discussed the implementation of the all three levels of our implementation that includes the Scratcher which is an AR application, the AR application server, and the target hub. We showed the architecture, activity diagram, entity relationship and source code of the implementation whenever it was necessary.
4.4 Web APIs

As discussed before, app servers and the target hub are interacting using http requests. Here, we are giving the list of APIs exposed by the target hub in our prototype implementation. The target hub APIs are accessed on http://arconnect.azurewebsites.net/api. we are using ‘show targets’ given in Table 4.1 in order to look for targets using tags.

<table>
<thead>
<tr>
<th>Title:</th>
<th>Show targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL:</td>
<td>/Target/GetTargets? Identifier=:identifier&amp;ID=:id</td>
</tr>
<tr>
<td>Method:</td>
<td>GET</td>
</tr>
<tr>
<td>URL Parameters:</td>
<td>Identifier = [integer], ID = [integer]</td>
</tr>
<tr>
<td>Data Parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Success Response:</td>
<td>Code: 200</td>
</tr>
<tr>
<td></td>
<td>Content: {1: “stones”, 2: “tarmac”}</td>
</tr>
<tr>
<td>Error Response:</td>
<td>Null</td>
</tr>
<tr>
<td>Sample Call:</td>
<td>Target/GetTargets?Identifier=7568123&amp;ID=20&amp;tags[]=lab&amp;tags[]=park</td>
</tr>
<tr>
<td>Notes:</td>
<td>Shows name of the targets that has any of the tags specified in the parameters.</td>
</tr>
</tbody>
</table>

When we have the target name we can start downloading the target using ‘Download
Target. Details of the API is given in Table 4.2.

<table>
<thead>
<tr>
<th>Title: Download target</th>
<th>URL: /target/Download? Identifier=:identifier&amp;ID=:id&amp;TargetName=:name,&amp;format=:format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method: GET</td>
<td>URL Parameters: Identifier = [integer], ID = [integer], TargetName,= [string], format = [string]</td>
</tr>
<tr>
<td>Data Parameters: None</td>
<td>Success Response: Code: 200 Content: [file data]</td>
</tr>
<tr>
<td>Error Response: Code: 404 Content: {Message : “This File does,not exist in the Hub!”}</td>
<td>Code: 400 Content: {Message : “Server is not registered!”}</td>
</tr>
<tr>
<td>Sample Call: /target/Download?Identifier=7568123&amp;ID=20&amp;TargetName=tarmac&amp;format=xml</td>
<td>Notes: Downloads a target with specified name,and format</td>
</tr>
</tbody>
</table>

Users, developers, and applications are capable of uploading targets to the target hub using ‘Upload target’ API. Details of the API is explained in Table 4.3.

<table>
<thead>
<tr>
<th>Title: Upload target</th>
<th>URL: /target/Upload? Identifier=:identifier&amp;ID=:id&amp;TargetName=:name,&amp;tags[]=:tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method: POST</td>
<td>URL Parameters: Identifier = [integer], ID = [integer], TargetName,= [string], tags[] = [array string]</td>
</tr>
<tr>
<td>Error Response: Code: 400 Content: {Message : “Server is not,registered!”}</td>
<td>Sample Call: /target/Upload?Identifier=7568123&amp;ID=20&amp;TargetName=bottle&amp;tags[]=glass&amp;tags[]=sport</td>
</tr>
<tr>
<td>Notes: Uploads a target with specified name,tags,and the attached file</td>
<td></td>
</tr>
</tbody>
</table>

Sending and receiving text messages is supported by ‘Message passing’ API. Details of the API is explained in Table 4.4.

All of the services in the target hub is only provided for the registered clients. We are using API ‘Register a server’ in order to register to the target hub. Necessary information for the API is provided in Table 4.5.

Content and Context Provider can use the target hub to collect information for their application purposes. In this regard, we have provided an API called ‘GetServers’ that lists name of the servers that have already registered in the target hub. Details and
### Table 4.4: Document of message passing in target hub

<table>
<thead>
<tr>
<th>Title:</th>
<th>Message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL:</td>
<td>target/ForwardMessage?Identifier=:identifier&amp;ID=:id&amp;TargetName=:name,&amp;UserName=:username&amp;SentMessage=:message</td>
</tr>
<tr>
<td>Method:</td>
<td>GET</td>
</tr>
<tr>
<td>URL Parameters:</td>
<td>Identifier = [integer], ID = [integer], TargetName,= [string], UserName = [string], SentMessage = [string]</td>
</tr>
<tr>
<td>Data Parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Success Response:</td>
<td>Code: 200</td>
</tr>
<tr>
<td></td>
<td>Content: {Message : “Message,forwarded”}</td>
</tr>
<tr>
<td>Error Response:</td>
<td>Code: 400</td>
</tr>
<tr>
<td></td>
<td>Content: {Message : “Server is not registered!”}</td>
</tr>
<tr>
<td>Sample Call:</td>
<td>target/Upload?Identifier=7568123&amp;ID=20&amp;TargetName=bottle&amp;UserName =rahim&amp; SentMessage=Hi</td>
</tr>
<tr>
<td>Notes:</td>
<td>Receives a message from an application,server under a target’s name and forwards it to all servers subscribed for, the specified target.</td>
</tr>
</tbody>
</table>

### Table 4.5: Document of registration to target hub

<table>
<thead>
<tr>
<th>Title:</th>
<th>Register a server</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL:</td>
<td>/server/register?server=:servername&amp;Identifier=:identifier&amp;Address=:address</td>
</tr>
<tr>
<td>Method:</td>
<td>GET</td>
</tr>
<tr>
<td>URL Parameters:</td>
<td>Identifier = [integer], servername = [string], Address = [string]</td>
</tr>
<tr>
<td>Data Parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Success Response:</td>
<td>Code: 200</td>
</tr>
<tr>
<td></td>
<td>Content: {Message : “ID:id”}</td>
</tr>
<tr>
<td>Error Response:</td>
<td>Code: 400</td>
</tr>
<tr>
<td></td>
<td>Content: {Message : “server name or id is null!”}</td>
</tr>
<tr>
<td>Notes:</td>
<td>Registers a server to the target hub, under the specified name and address and returns the id of the server. This, id is going to be used as a token for future requests.</td>
</tr>
</tbody>
</table>

Sample call of the API is provided in Table 4.6.

### 4.5 Expiration and Activation Tags

Previously, we discussed that the burst of the targets and contents should be controlled in the proposed architecture. One way is to keep a target or a content in the hub only until it expires.

Several AR architectures and platforms such as ARML, ARGON, Wikitude, are using tags very similar to XML tags to manage virtual contents. A tag is typically describing an attribute of a content including its type, ID, location, orientation, etc.
Table 4.6: Document of showing the servers of the target hub

<table>
<thead>
<tr>
<th>Title:</th>
<th>Show all registered servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL:</td>
<td>/server/GetServers</td>
</tr>
<tr>
<td>Method:</td>
<td>GET</td>
</tr>
<tr>
<td>URL Parameters:</td>
<td>Identifier = [integer], servername = [string], Address = [string]</td>
</tr>
<tr>
<td>Data Parameters:</td>
<td>None</td>
</tr>
</tbody>
</table>

Success Response:
```
Code: 200
Content: [
{
  "Identifier": "7568123"
  "Name": "ServerB"
  "Requests": [
    {
      "Id": 18
      "Type": "Register"
    }
  ]
}
{
  "Identifier": "123354"
  "Name": "ServerA"
  "Requests": [
    {
      "Id": 19
      "Type": "Register"
    }
  ]
}
```

Error Response: Null

Sample Call: /server/getservers

Notes: Shows the list of all registered servers, with all of the requests they have made.

In this section, we are proposing two essential attributes that many AR contents would need it if it comes to deal with time. We are naming them “Activation Tag” and “Expiration Tag.” Although these concepts can be found in several works, for instance, a time tag on a network packet. However, these concepts have been overlooked in the architectures for AR target’s and content’s format.

- **Activation Tag**

  This tag shows the time after which the content is enabled. Before this time the content will be treated as if it does not exist.

- **Expiration Tag**

  This tag shows the time after which the content is disabled. After this time the
content will be treated as if it does not exist.

We implemented both of the expiration and activation tags to show its feasibility and functionality. We developed a component for the “Activation” and “Expiration” tags. As soon as the component is added to the asset, it is possible to set the times. There is an enable check mark for each tag using which user can disable or enable the tag.

4.5.1 Test Case

We have a cube as an AR content with an Activation tag for a time in the future. The exact time in the future is: 12/29/14 15:010. We also have a sphere with the expiration time tag 12/29/14 15:11. In this scenario, the image targets are stones and chips models that are used in the Vuforia SDK sample examples. Figure 4.22 shows that the target has been found, and the sphere has been augmented, but the box is not in the scene. The reason is the activation tag for the box has been set for 15:10, but the time is 15:09.

There are many ways to implement time triggered events. We have used Unity’s Update method to check the time in every scene update. In Figure 4.23, time is 15:10 and we can see that the box has appeared in the scene.

Update method in Unity is invoked on every frame. Therefore expiration of the contents will be checked and detected with the frame speed. As soon as a content expires, it disappears from the scene. Figure 4.24 shows that at 15:11, the sphere disappears because it has passed its expiration time and the sphere is no longer in the scene.

4.6 Chapter Summary

In this chapter, we showed how we are validating our work by implementing a software and the proposed framework as a proof of concept. The details of the implementation
Figure 4.22: Only sphere is in the scene

Figure 4.23: Both of sphere and cube are in the scene
for each main module was discussed. We also covered the technologies used in the implementation such as MVC framework, long polling and simple polling, Post and Get Http methods.

We implemented the proof of concept application using Web APIs for which the necessary documentation was provided. The next chapter is concluding the thesis by providing a summary of the research objectives and research results, the limitations of our work, and future directions.
Chapter 5

Conclusion and Future Directions

Augmented reality is about superimposing contextually-relevant information onto the real world. The technology has consumed researchers’ and developers’ imaginations for a long time. In recent years, we witnessed rises and falls in different aspects of AR. We saw the problems of head-worn devices such as Google Glass and Microsoft HoloLens, despite the initial excitement, and also the rise of hordes of Pokémon monsters. However, the question of what AR experience people would want or need is even now an open question [51]. Still, Developers and end users cannot properly benefit from AR due to the proprietary formats, lack of standards, and structural problems. Also, the existing AR architectures generally are not designed to enable users’ contribution. This prevents the wide-spread adoption of AR.

User contribution, on the other hand, has been the center of attention for many internet based services including social networks and social media. In fact, user contribution is one the basic pillars of Web 2.0. The ability to involve the users in creating and sharing AR resources to enrich the AR application’s experience forms the premises of investigation of this thesis. The thesis intended to answer these research questions:

- What kind of software models and protocols would enable user A to browse any subset of the targets \{T_{B_1}, T_{B_2}, \ldots, T_{B_n}\} which user B is augmenting?
• What kind of software models and protocols would make it possible for user A to send and receive messages from user B and vice versa?

The attempt to answer these research questions led us to design a new architecture for implementing the AR technology, which we named “Client Federated Servers.” Having user contribution in mind, we designed the client federated server model to be capable of handling AR resource sharing and user communication. Using the proposed model, users can share their targets and communicate with each other. Client federated server architecture is using Web APIs to handle the requests which makes it platform independent.

To demonstrate the validity and feasibility of the proposed architecture, we developed a mobile application called “Scratcher.” Scratcher allows AR users to have communication about a target as a focal point. Users can share their experience in targets’ chat rooms. Also, using Scratcher, the targets of different applications can be shared and augmented among Scratcher users.

5.1 Limitations

Although we have tried to cover drawbacks of the previous works, there are still a number of shortcomings in the proposed architecture. We do not think of these issues as trivial matters. However, we believe that the system is functional enough to be adopted and implemented. We are dividing these limitations to the boundaries of the AR application and the obstacles of functionality. The areas of concern are as follows:

(A) Participatory AR experience

A Participatory AR application is an AR experience with multiple user’s collective interaction environment [60]. Such interactions include the interaction of the user with other users, physical aspects of the target, and the virtual content
superimposed over the target. Target hub supports user-level communication. However, interactions between users and contents is a problem for future work.

(B) Content level interaction

To the best of our knowledge, content interaction has not been scrupulously examined among AR contents with different formats. A standard way of content interaction among both contents with same framework and contents belong to different frameworks is needed. By content interaction, we mean the ability of AR contents to communicate with each other independent of a user’s intervention.

For more clarification, imagine a virtual ball is moving in one AR application, and there is a wall in another application’s environment in the trajectory of the ball of the previous application. The ball should be able to hit the wall and redirect without any outside intervention. With current AR frameworks, the content interaction would be possible only on proprietary contents.

(C) Redeveloping problem

Target hub gives a way to share targets and contents, but in the end, a content will be used on the platform for which it is been developed. Hence, a popular target or content needs to be redeveloped to each desired platform. A standard for all of the targets and a standard for contents yet remains an unresolved issue.

Regarding the obstacles of functionality, there are few potential problems for the proposed framework that should be taken into consideration.

(A) Content and target management

We think the most significant potential problem that the new design can introduce is the burst of increase of targets and contents. Especially, most of these targets and contents are going to have temporal usage for users. This increase will have two impacts on servers. Firstly, it will slow down both servers while looking up for the targets and contents, and also the overall response time of the system.
Secondly, servers will have memory problem. This can damage the scalability of the system, too. One solution for the problem can be implementing the targets and contents with expiration dates. Targets and contents that are expired are going to be deleted from the system. Purging the unnecessary data would effectively decrease the volume of the contents and targets.

(B) Increase of the traffic

With the new design, there is going to be an increase in the network traffic. The traffic induced by target detection request on the target hub as well as the traffic coming from the client due to search request should be effectively managed. One solution for managing traffic on the target hub is to have a distributed structure for the hub. A hierarchical tree structure for the target hub similar to Domain Name System (DNS) can resolve these requests for a target in the lower levels of the tree rather than relaying them to the root servers.

(C) Adaptability

If the proposed framework is not adaptable with current frameworks, it can be rejected from the business sector and also it will be an island of its own. Adaptability is the capability of supporting current frameworks in the sense of software and hardware. The proposed framework should not require radical changes in the hardware. What we are looking for is to build on top of the existing frameworks and increase functionality. Therefore, we are using web APIs to be able to support our clients with minimal adoption effort.

5.2 Future Work

The contributions of this work could be further improved in the following areas:

• Participatory AR has not been supported.
To enable participatory AR, complex behaviors of the targets and contents should be supported in target hub. This needs further investigation.

- Platform independent targets and contents are lacking.

One way of implementing platform independent targets and contents is using a middle-ware such as Java Virtual Machine (JVM) that would reside on the clients’ machine. The middle-ware should be able to render the targets and contents that are shared by all other users and developers.

- Social network in AR has not been implemented.

The main aspect of social network is the ability to traverse the graph of friends. We did not incorporate all of the aspects of social network in the thesis. It would be exciting to see a full integration of social networking in AR.
Bibliography


[17] Evan Barba, Blair MacIntyre, and Elizabeth D. Mynatt, *Here we are! where are we? locating mixed reality in the age of the smartphone*, Proceedings of the IEEE 100 (2012), no. 4, 929–936.


