This document is downloaded from CityU Institutional Repository, Run Run Shaw Library, City University of Hong Kong.

Title	Mobile live broadcasting application
Author(s)	Mok, Ho Leung (莫浩良)
Citation	Mok, H. L. (2012). Mobile live broadcasting application (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.
Issue Date	2012
URL	http://hdl.handle.net/2031/6763
Rights	This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.



City University of Hong Kong
Department of Computer Science

BSCCS Final Year Project Report 2011-2012

11CS017

Mobile live broadcasting application

(Volume ____ of ____)

Student Name : Mok Ho Leung
Student No. :
Programme Code : Bsc CS

Supervisor : CHOW, Chi Yin Ted
1st Reader : CHAN, Mang Tang
2nd Reader : JIA, Xiaohua

For Official Use Only

Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

_____Mobile live broadcasting application_____

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: _____ Signature: _____

Student ID: _____ Date: _____

ABSTRACT

When smart phone are being so popular in this modern era, every user inspires the huge demand of real-time information. Watching instant news, amateur video on smartphone become the mainstream of how medias flowing and become part of our daily life.

Some user with a heart of adventure love to share what they experiences with the eye in the smartphone, but it's a pity there are huge percentage of naïve user in the world, sharing a video to internet is a "mission impossible" to them. No wonder how much interesting contents lie hidden in the memory in people's smartphone.

Think of how many curious users want to pry about what is happening in the city they living? How many users want to see the scene of a match in gymnasium? How much audiences want have an instant view in mountain bike racer and knowing where are they use pass through?

On the market, there is lot of live broadcasting platform, some provide video on demand (VOD) services, some provide Global Position System (GPS) tracking on initial recording spot, some provide live broadcasting, and some provide mobile live function. However this is very rare have a solution that integrates all of these features.

Summarized all of the proposed reasons, It brings out the necessary to this project. This project aims to study the existing video-sharing application with related

technologies, find out the possibility in develop a better solution then existing platforms. Also this project will propose unprecedented approaches in overcome some technical problems in live streaming on mobile devices and algorithm to settle the contents monopolization. At the end, this project develops an integrated iPhone solution with live broadcasting, video sharing and geotagging and tracking plus application server modules. An evaluation to the proposed solution will be given.

The project has multiple achievements aims to achieve. First, the application should simplify the steps on sharing the video to Internet, replace traditional sharing way from "upload video" by "broadcasting video", and make contents widely available to normal user. Second, proposed solution target to be ideal platform in discover more potential amateur video publisher, supporting amateur content spreading on Internet by proposed content recommendation algorithm. Third, real-time geotagging available in live video and recorded video in emphasize map and locations importance apart from video contents itself. Last but not least, contribute to public event for providing instant interior view without special video recording equipment such as mountain bike race.

Acknowledgement

The project is hardly complete without any supportive alliances who provide me a guidance. I would like to take the opportunity to give the honorable salutation to them.

First of all, I am truly thanks to my supervisor, Dr Chi-Yin Chow, Ted. I am appreciating for the quick response when I would like to arrange for a meeting in discuss the FYP matter. The opinions, encouragement and your understanding offer a great support to make this project come true.

I would also like to credit to 3 people whose provide massive support during the development. First, Ho-Wang Wong, who give me the guidance of socket programming in Linux. I am truly thanks for your guidance, otherwise project may still stuck on segmentation fault. Second, Alan po, the ex-supervisor of my final year projects when before I graduated in IVE-TY. Also without the guidance of programming the GPS functionality in iOS, I may not have a smooth start on the project kickoff. Third, Mr. Steve McFarlin, who's the author of Livu app, without your professional opinion in video stream, I may still blindly struggle with RTMP/RTSP transmission.

No doubt I have to thanks to my family for in tolerant the messy life during this year, the understanding and give me a comfortable environment working the project is the best encouragement. I would also thanks to my classmate that we fight side-by-side together within this year.

Table of Contents

Chapter 1: Introduction 1
1.1 Background 2
1.2 Motivations 3
1.3 Solution 5
1.4 Challenges 7
1.5 Project Objectives and Aims 10
Chapter 2: Literature Reviews 12
2.1 GPS/A-GPS 12
2.1.1 GPS 12
2.1.2 A-GPS 13
2.1.3 Existing Location based application 14
2.1.3.1 Google Latitude 14
2.1.3.2 Facebook 15
2.1.3.3 WhoIsNear 17
2.1.3.4 Integration of different characteristics 18
2.2 Video compression technique 18
2.2.1 How video compression works 19
2.2.2 Video compression method 20
2.2.3 Common compression algorithm 21
2.2.4 Video compression standards 22
2.2.4.1 MPEG-4 Part 2 23
2.2.4.2 H.263 (3GP) 24
2.2.4.3 H.264/MPEG4 AVC 25
2.2.4.4 Conclusion 26
2.3 Media streaming and live streaming 26
2.3.1 Streaming bandwidth and storage 27
2.3.2 Unicasting and Multicasting 27
2.3.3 Streaming protocol 27
2.3.3.1 RTSP 28
2.3.3.2 RTMP 29
2.3.3.3 HLS 29
2.3.3.4 Conclusion 30

2.3.4 Existing video sharing application 30
2.3.4.1 Qik 30
2.3.4.2 Stickam 32
2.3.4.3 UStream 33
2.3.4.4 Comparison of Existing Application and Conclusion 34
Chapter 3: Design analysis 36
3.1 System requirement 36
3.2 Application Server 37
3.2.1 PHP Polling and long polling 37
3.2.2 Socket 40
3.2.2.1 Socket server 40
3.2.2.2 select() & kqueue() & epoll() 41
3.3 Media Streaming 42
3.3.1 Streaming protocol 43
Chapter 4: Project equipment 46
4.1 IPhone development 46
4.2 IPhone 49
4.3 IPhone SDK 5.0 50
4.4 Xcode 4.2 on MAC OSX 51
4.5 Ubuntu 11.10 52
4.6 Wowza Media Server 53
4.7 LAMPP 54
Chapter 5: Methodologies 55
5.1 System Overview 55
5.2 Message broadcasting 56
5.3 Video Streaming 59
5.3.1 Video encoding setting 61
5.4 JSON data interchange 62
5.5 Map view and Position tracking 65
5.6 Recommendation 69
5.6.1 Slope one Algorithm 69
5.6.2 User favorite 72
5.6.3 Limitation, Performance and complexity 73
5.6.4 Collecting data for algorithm 74
5.6.5 Recommendation presentation 76

5.6.6 Special Cases 77
Chapter 6: System Design 78
6.1 Provided functions 78
6.2 Use case Diagram 79
6.3 Class Diagram 82
6.4 Sequence Diagram 84
6.5 Database Architecture Design 92
Chapter 7: Detailed System Implementation 93
7.1 Wowza Media Server 93
7.1.1 Start/Stop Media Server 93
7.1.2 Stream content to live session 94
7.2 Socket application Server 97
7.2.1 Start/Close the server 97
7.2.2 Handle client connection/disconnection 98
7.2.3 Handle clients command 101
7.2.3.1 Decoding message 101
7.2.3.2 Handling JSON request 103
7.2.4 Manage a live sessions 104
7.2.4.1 Create live session 104
7.2.4.2 Room logic 105
7.2.4.3 Live broadcasting lifecycle 106
7.2.5 Broadcasting live status 109
7.3 IPhone application 111
7.3.1 Network Connections 111
7.3.2 Positioning and map 115
7.3.3 Live streaming 120
7.3.4 HLS live streaming 126
Chapter 8: Evaluation 127
8.1 User interface 127
8.2 Performances 130
Chapter 9: Limitation 136
9.1 Processing power in iDevice 136
9.2 3G network connectivity 136
9.3 Media Server 137
Chapter 10: Future enhancements 138

10.1 Dynamic video quality 138
10.2 Load balancing 138
10.3 Privacy settings 139
10.4 Social Network features 139
Chapter 11: Conclusion 140
References 142
Appendix 146
Monthly log 146

List of Figures

Figure 1 Deliverable application in this project - iLivecast 6
Figure 2 Standalone GPS require at least 3 satellites 13
Figure 3 A-GPS operation diagram 14
Figure 4 Google latitude is accessible on mobile web page. [8] 15
Figure 5 Facebook check-in feature 16
Figure 6 User can recommend the content to other users. 17
Figure 7 Bit-rate comparison with same image quality among 4 video standards 19
Figure 8 MPEG -1 Motion Compensation 20
Figure 9 Video data reduction between 3 frames 20
Figure 10 Block diagram of video compression [9] 21
Figure 11 Timeline of Compression standard development [10] 23
Figure 12 Qik reducing file size by high video compression, it takes 1MB for shoting 20s video footage 31
Figure 13 Qik maintain live delay in acceptable range. 31
Figure 14 Stickam takes over half minute video delay in broadcasting. 33
Figure 15 UStream broadcasting provide clear UI and poll function for live streaming. 34
Figure 16 Table of comparison between existing application and proposed solution .35
Figure 17 Single connections by PHP Polling invoke server request in an interval time. 38
Figure 18 Single connection by long-polling, new connection made right after previous connection received data or timeout. Connections lengths are irregular. ...39
Figure 19 Benchmark of Epoll(), Kqueue(), poll() and select() with 100 connecitons [24] 42
Figure 20 RTMP port usually blocked by firewall 43

Figure 21 Statement of Requirements for App in Apple iOS Developer Library [25] .44
Figure 22Overview of HTTP Live Streaming [26] 45
Figure 23 Comparison between 3 iPhone Enrollment Programs [27] 47
Figure 24 Member center in apple iOS dev center.....48
Figure 25 Technical Specification of iPhone 4 [28].....49
Figure 26 XCode 4.2 Welcome startup screen.....51
Figure 27 UBuntu 11.10 x64 User interface 52
Figure 28 Wowza media Server architecture.....53
Figure 29 High-level System Architecture of proposed solution.....55
Figure 30 Overview live broadcasting process flow diagram 59
Figure 31 Live streaming architecture diagram.....60
Figure 32 Table of defined Video Compression setting 61
Figure 33 Data Interchange diagram between server, database and iOS App by JSON 62
Figure 34 JSON Protocol Structure Diagram. Defined protocol must implement on both side interface. Server does not understand undefined JSON.....63
Figure 35 Packet structure and length info usage in prepare char[] data 64
Figure 36 System flow of update user position 65
Figure 37 Example of tracking position on map 66
Figure 38 Example of location update occur when out of a circle range 67
Figure 39 Source code of getting 4 corner coordinate by map View 67
Figure 40 Example of using Annotations object indicate location on map.....68
Figure 41 Slope one calculation result (left), predict computation on Mysql (Right) 72
Figure 42 Rated score will use to calculate video average rating and its tags average score with respect to individual user 74
Figure 43 Display existing video content without recommendation 76
Figure 44 Recommendation proposed video that user might interested.....76
Figure 45 List of intractable functions provided for user in mobile application.....78
Figure 46 Use case diagram of iLivecast.....80
Figure 47 Class Diagram of iLiveCast Application 83
Figure 48 Delegate structure example.....84
Figure 49 3-Tier Sequences diagram of create live streaming event (create event)....86
Figure 50 3-Tier Sequence diagram of create live streaming event (start streaming session) 87
Figure 51 3-Tier Sequence diagram of create live streaming event (stop streaming

session) 88
Figure 52 3-Tier Sequence diagram of watch live streaming event 89
Figure 53 3-Tier Sequence diagram of watch recorded streaming event by map view 90
Figure 54 3-Tier sequence diagram of watch recorded streaming event by search list 91
Figure 55 Entity Relationship Diagram of live broadcasting system 92
Figure 56 Startup socket application in background process 93
Figure 57 Startup socket application server in parent process 94
Figure 58 Workflow diagrams of custom application in Media Server 94
Figure 59 Work flow of appending video sources 95
Figure 60 Startup socket application in background process 97
Figure 61 Startup socket application server in parent process 98
Figure 62 Kill the socket server in background process 98
Figure 63 Register connection_accept method for handling network connection establish 98
Figure 64 Event listener handling connection establish 99
Figure 65 Accept socket connection when client connect to server 99
Figure 66 Socket receive notification when connection is lost 100
Figure 67 Workflow diagram of socket server handle connections 100
Figure 68 Code segment: handling receiving data 101
Figure 69 Code segment: Conneciton object structure (left), READ_STATE enumeration (right) 101
Figure 70 Activity diagram of handling new incoming packet in Connection_Read() 102
Figure 71 Code Segment: Method dispatch the JSON body 103
Figure 72 Code segment: switch body handle JSON message type 103
Figure 73 Table of JSON message protocol description 103
Figure 74 Code segment: create new room object and register evtimer upon client JSON request 104
Figure 75 Server message shows user create a new live session 104
Figure 76 State Diagram of room object 105
Figure 77 State Diagram shows action that cause ROOM_STATE change 105
Figure 78 Work flow diagram of live streaming in handle video upload from mobile in STATE_REC 106
Figure 79 Work flow diagram of live streaming in finish streaming session in

STATE_RED_END 107
Figure 80 Work flow diagram in clean up and update room info in STATE_REC_CLOSING 108
Figure 81 Server message shows room state change due to publisher disconnected and begin clear up process for room 109
Figure 82 Message from connection object deliver to particular room object for broadcasting 109
Figure 83 Broadcasting join room JSON to participant 110
Figure 84 Network Connection between server and client 111
Figure 85 Code segment: SocketConnectionManager header structure 112
Figure 86 Code segment: Connect to socket server 113
Figure 87 Code Segment: Append message size header to JSON message before send out to socket 113
Figure 88 Code segment: Receive data from socket 114
Figure 89 Code segment: delegate method in GPSTransmitter handle received JSON from socket 114
Figure 90 Code Segment: delegate function of location update in main map 115
Figure 91 Code segment: centralize the map to the specific coordinate with zoom level 116
Figure 92 List of Pins used to point out video location in map 117
Figure 93 Code segment: add annotation on map 117
Figure 94 Code segment: viewForAnnotation configure differently in user mark and SFAnnotation 118
Figure 95 Difference response when touch on the annotation between SFAnnotation(Left) and User Location(Right) 118
Figure 96 Two different mode switcher 118
Figure 97 Creating streaming session(Left), Waiting user start recording (Right) 120
Figure 98 Class relation workflow in high-level video capture [28] 121
Figure 99 Customize the video encoding property to AVAssetWriter 121
Figure 100 Sample Data capture from device to AVCaptureSession Delegate [28] 122
Figure 101 Code segement: AVCaptureSession delegate appending buffer to AVAssetWriter 123
Figure 102 Video Capturing and broadcasting to server 124
Figure 103 Code segment: Swap new AVAssetWriter 124
Figure 104 Streaming architecture flowing diagram 125

Figure 105 JSON object contents transform to annotation in map 126
Figure 106 Functions are clearly divided on the left and right as sub-functions..... 127
Figure 107 More function in Dropbox app in iOS (left), proximity of function grouped on a side..... 128
Figure 108 Camera view remains visible (left), while user browse video details pages 129
Figure 109 Testing during a bus ride in somewhere in Mongkok – using the new iPad and iPhone 4 testing the location function and video streaming function 134
Figure. 110 Testing in apartment – using the new iPad and iPhone simulator connected to City University Project lab server, testing streaming video function..... 134
Figure. 111 Testing in apartment – using the new iPad and iPhone simulator connected to City University Project lab server, testing position of live streaming in main map 135
Figure. 112 Testing in apartment – using the new iPad and iPhone simulator connected to City University Project lab server, testing position of live streaming in main map 135

Chapter 1: Introduction

In this section, the background of how mobile device and telecommunication network cause the impact to live streaming is introduced. Afterward the challenges of live streaming in mobile device and problems of existing application will be discuss. Then propose a new solution to solve those problems.

1.1 Background

Review past decade, the rapid transformation of telecommunication technologies change communication pattern between people, it change the behavior of how users use the network [1]. Network becomes part of our life. We would read the latest news on the Internet, read friends new thoughts on social network or explore what was happened on another side of the world in video broadcasting platform.

YouTube, one of the most successful exits of the Web 2.0 era, it's single biggest contribution is that it brought into the mainstream the concept of sharing videos online, offer an opportunity to watch what was happened around the world [2]. Most of the users can shot the video with their smartphone and upload it to the broadcasting platform from their PC. Also some amateur content like short original films is so popularity nowadays, such as situation of demonstrators protesting outside the building of legislative council. Moreover for some advanced user who likes to share their thoughts instantly, they can attach a webcam to their computers and have a live broadcasting. The desire of the news and curiosity makes this website become so successful.

Thanks to the popularity of smartphone, especially credit to the achievement of iPhone, with this tiny device implanted with high definitions camera, large touch display, cellar network and GPS function, we can keep in touch with news, also let us contribute what we watched or thoughts to be news. iPhone probably the most influential consumer electronics product ever made [3] , it breaks the technical barrier of sharing, it turns everyone to be a correspondent of this electronic age, we are no longer only be the audiences.

1.2Motivations

Hardware, Resource limitation

Thanks to the nano technology, our standard PC, notebook becomes even strong and capable to complete many tasks we want to accomplish. Meanwhile another new platform is being focus on the development that is the mobile device. Although mobile device can take much more responsibility than ever, unfortunately lot of challenges had note overcome yet. Take video broadcasting as example; PC can live streaming the footage in high quality with HD camera through the high-speed 100M broadband network. On contrary, mobile phone only provide 7.2Mbps 3G network, how to ensure the video quality in limited bandwidth without having more than 30 seconds delay? How to handle the connection signal drop during broadcast? For this circumstance, because of limitation of hardware resources, compression technique and selection of transfer protocol becomes important in this case and makes application develop for mobile platform challenging.

Problem of existing application

Numbers of popular video sharing websites provide services for users upload the movie clips even enable users' mobile device with this functionality. But when people would like to further understand where the movie was taken, usually only simple geotagging details provided. Although Google map [5] is quite common to many applications especially in social network category, surprisingly no broadcasting application would make full use of it. Also no existing video sharing platform able to provide constantly geotagging information during recorded video playback.

In addition, not much platform able to provide live upload streaming because iPhone have no RTMP supported. Without RTMP streaming is there any alternative solution that enable iPhone perform an live streaming? Most of the live streaming enabled platform only provide simulate experiences in watching YouTube, users hard to experience any “live”.

Furthermore, video quality and stability are awful despite that it’s been launched for a while long. But none of the application would address this problem and propose any workaround or alternative options.

Last but not least, most of the video contents uploaded to existing video platform go down the drain because most platform are list view based. In an endless list it is neither possible that user could discover your content nor the system recommend your content to others. But most platforms turn a blind eye of this situation makes, video sharing like law of jungles, a simply sharing should not be like this.

What if we can expand the usage of GPS functions makes the live streaming like a map base social network? Integrate with instant update positioning and nearby broadcast notification, this could solve the problem of attract more users watch the live footage. A smaller scale content recommendation makes video advertised more reasonable to fair to most users.

1.3Solution

To integrate both live broadcasting and geography location services, this project choose a mobile device with enough processing capability, also should have a standard of hardware specification to calibrate best setting for video encoding/streaming. Upon October 2011, over 17.1 million iPhone 4 has been sold in Q4 2011 and over 1 million pre-ordered iPhone 4S are ready to supply to the market [4]. With this popularity and highly match with this project needs, such as A-GPS enabled, 3G or faster network, makes iPhone 4 be the ideal platform for this project.

In this project, an iPhone mobile application will build, which allow users to broadcasting the live video from their iPhone camera to Internet. The application able to stream the video only by HTTP protocol rather than RTMP protocol. In compare to traditional list view based video sharing platform, Location-based application provides more instant experiences and better video promotion. Besides, with GPS function enables, broadcaster's GPS location will update instantly to ensure video followers know the latest geotaggings. Last, a recommendation features is provided assisting advertised user's contents.

The iPhone application named as "ILiveCast". It is developed under iOS Software Development Kit (SDK) 5.0. This project is developing by XCode in a MacBook in operation system Mac OSX 10.7. Program is tested in iPhone4 devices with iOS version 5.0.1. Static library such as JSONKit, GCDAsyncSocket, ASIHTTPRequest, numerous external UI library and apple cocoa frameworks are employed for the purpose of

enhance development efficiency.

In addition, a socket server and media-streaming server will be constructed as HTTP live streaming handler in replaces traditional RTMP upload streaming. This combination not only enables the video broadcasting functionality, but also instantly handles user access, geotagging details, message broadcasting and content filtering.

A Socket server developed under uBuntu (Debian Linux Distribution) platform. Static library “Libevent” is employed for handling socket connection development. Wowza Media Server(WMS) and XAMPP also host on Linux platform. WMS is responsible for broadcasting video content in limited connecting time while XAMPP is responsible for PHP web hosting and provide MYSQL database for system testing.



Figure 1 Deliverable application in this project - iLivecast

1.4Challenges

Question, Study and Technical challenges

The project mixed with multiple technologies in multiple platforms that required lot of study, research and design in building the solution. One of the challenges in this project is find out the answer regarding to following proposed question and uncertainty.

As a server, which media server is suitable for designed approach? Can RTMP/RTSP provide services to mobile devices? How does the HTTP live streaming works? Instead of RTMP incoming stream, any other approaches can be used? What type of video encoding should be used? How to record the live stream and make it available for later preview? How to intelligently recommend some video contents to user?

If an application server is needed to handle real-time message flowing, a web-based socket server or application socket server provide better performance? How to transfer package between socket servers with other application? Can we constantly transfer a media stream to socket server? How to handle multiple connections and deliver message to correct client? How to handle the geotags related to video and store to server?

For the iPhone application, how to access the video feed from camera? How to transfer these feeds as a live stream? How to minimize the video size? How to customize and what are the best video encoding settings suitable for mobile? How to maintain a

socket connection and which approaches should use? How to communicate between socket and application? How to perform a multithreading in iPhone? How to play a live-streaming video in iPhone? How to manipulate the map interface? How to manage annotation update on map in real-time? How to display addiction info like position radar on top of media player? How to broadcasting message to other mobile?

Required technical knowledge

Another challenges of this project are technology used. The project combines with multiple development platform, hardware and software that makes project very challenging. The project combines with 3 different servers and iPhone application, each of them require different programming language, skill sets and certain knowledge to particular techniques. Following skill sets are required in build the solution in this project, without any of them the project is hard to accomplish.

Used Software	
OS:	MAC OSX, linux uBuntu
IDE:	Eclipse, Wowza IDE, Xcode
Software:	Wowza Media Server, iPhone simulator, VMWare Fusion
Hardware:	iPhone, iPad, macbook

Media Server	
Language:	Unix shell script, unix command, XML, java
Skill set:	J2SE, Wowza Media API, IO library, Thread programming
Knowledge:	RTMP streaming, HTTP Live streaming, Wowza Media Server workflow
APIs:	J2SE, wowza Media API

Webserver and Database	
Language:	PHP, JavaScript, JSON, MYSQL, store-procedure
Skill set:	MYSQL DB, HTTP request, PHP IO, JSON encode/decode
Knowledge:	Slope-one algorithm, OO programming
APIs:	Open slope-one algorithm

Socket Server	
Language:	pure C, unix command, Unix shell script
Skill set:	Libevent library, thread programming, cJSON Library, socket programming, OO programming
Knowledge:	C Socket, non-block IO, event-based loop, linux child process, unix system call
APIs:	Libevent, cJSON

iPhone application	
Language:	Objective-C, JSON
Skill set:	GCDAsyncSocket, ASIHTTPReqest, Cocoa Framework, Coca-touch Frameowk, Map-kit, HTTP live streaming, OO programming, Thread programming, JSON message
Knowledge:	GCD async process, Dispatch queue async process, MVC, NS-Notification theory, Class delegation, Async HTTP process, HTTP Live streaming, Google map, iPhone UI programming, Media Player framework
APIs:	DLStarRating, ScrollableTabbar, iToast, BlockAlert, TDBadgedTableCell, DACircularProgress, PullableView, UAPanel, ViewDeck, JSONKit, ASIHTTPRequest, GCDAsyncSocket, DDLogging

1.5Project Objectives and Aims

This project covered objectives in 3 aspects – Study, Design and Development.

Study

1. Study Global Position System (GPS) technology and how GPS effectively used in various related existing location-based social networking application.
2. Study the video compression technique and evaluates various standards have best suitability of using in mobile device.
3. Study workflow of media streaming and existing streaming protocol finding the appropriate one using in mobile device. Also evaluate various web-based video sharing platforms and identify characteristics and shortage individually

Design

1. Design a streaming approach to utilize HTTP only for video broadcasting and streaming on mobile devices.
2. Design an efficient architecture to enable location-awareness video broadcasting.
3. Design a real-time system to enable broadcasting and message flowing in real time.
4. Design a video playback approach to provide geotags information regarding playback time.
5. Design an algorithm to intelligently recommend video contents to users according users deviation within geolocation range.

Development

1. Implement the application and server from the designed approaches.
2. Implement the applications with unique UI design in compare to traditional mobile application.
3. Evaluate the application efficiency and propose improvements.

This project aims to provide a location-based platform for broadcaster broadcast the live footage on their handheld devices, encourage users' broadcast unexpected incident and increase number of audience to support live video watch. Also the project also aims to benefits to following type of users.

- Reporters, provides easy-to-use live camera in report incident on real-time. Saved geotag along with playback helps them provide accurate location info in writing news.
- Audiences, Passages in bus can find live content and use recommendation functions to spot potential interested, nearby contents.
- Friends, sharing funny or stunning moment by live feed with multiple friends. Friends who separate in different part of countries can still sharing live footage together.
- Event observer, such as police, organization workers even normal users, during a big event such as protests, the instant cam with geotag helps monitor the real-time situation in different angle of the area.
- Racer, participant who participate in rally event such as mountain bike, they can easily attach the smartphone in front of their bike as instant first view provide real time situation of what they experiencing.

Chapter 2: Literature Reviews

In this chapter, we would analyze different technique in use for proposed solution and evaluating existing application, address the weakness, extract positive characteristic and summarize what approach should be chosen in the various technical options.

2.1GPS/A-GPS

2.1.1 GPS

The Global Positioning System (GPS) is a global navigation satellite system, which utilizes the satellites between space and earth to provide user location details. There are almost 3000 satellites on the earth and 24 of them are functioning as GPS satellites and 3 more will be launch for the services [6]. To enable the GPS functions, a GPS receiver is needed to receive more than 4 satellites signal and time information to computes the distances between each of them. With these details the GPS receiver can then compute the actual geography locations. GPS has great accuracy to locate the device location with approximate 3 to 4 miters error outdoor. Contrarily, the error will be huge or GPS undiscoverable if the GPS using indoor. Additionally, some GPS with extra variable can help faster determine the receiver position with only 3 satellites.

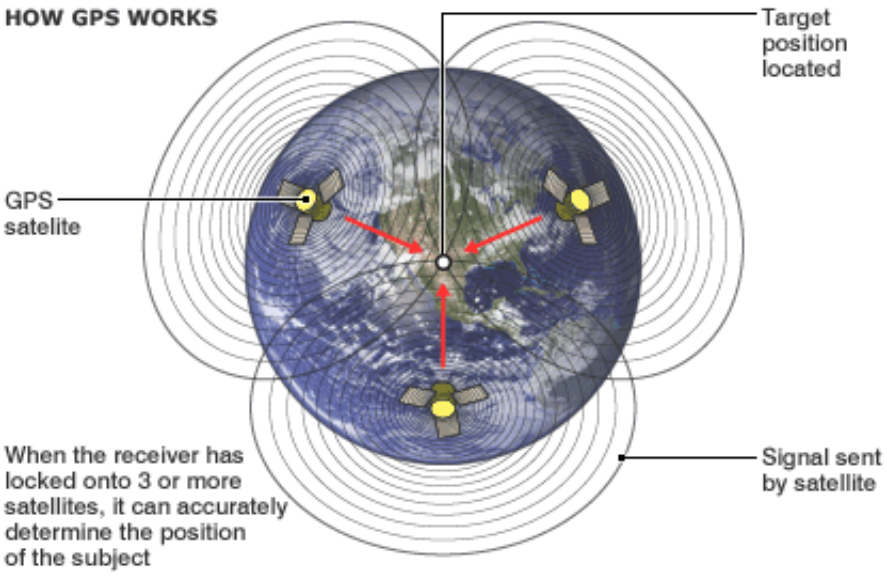


Figure 2 Standalone GPS require at least 3 satellites

2.1.2 A-GPS

Assisted Global Positioning System (AGPS) is an extended GPS system that can improve the startup performance in compare with "Autonomous" GPS. A-GPS is implanted into iPhone 4 and iPhone 4S [7]. The AGPS technique has been built in cellular phones with GPS capability extensively. The traditional GPS system need approximately 30-40 seconds to determine users position by calculate satellites timing, While A-GPS address these problem, it mix in with cellular network and WIFI network to connect to an assistance server for AGPS information.

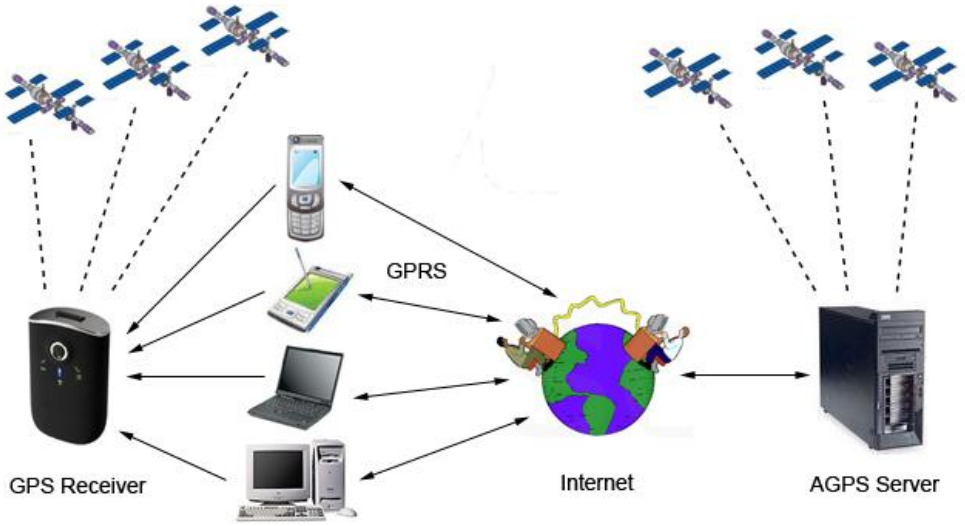


Figure 3 A-GPS operation diagram

The assistance server contains information such as orbital data of satellites, it help GPS receiver acquire to the satellites more quickly. A typical A-GPS receiver require Internet connection through cellular network or wifi network, it is also discoverable if user connected to AGPS assistance server by wifi network instead of cellular network.

2.1.3 Existing Location based application

2.1.3.1 Google Latitude

Google latitude

Google latitude [8] is a location base mobile app develop by Google and available for web application and mobile application. It was designed for people sharing their current geography location or view friends locations, Google account is required. Google latitude provide instant update of friends position, also we can limit the searching area so that friends pins will not be crowded. Google latitude API is free to use for any application, also they provide location history for each users, new

application do not need to implement database to store locations history of each users. Regarding to the privacy, Google latitude allow users customize location publication setting, which able to avoid illegal retrieve personal track. However to subscribe Google latitude services, it means users must register a Google account before use the application, also every privacy setting must configure in Google account control panel. It might defeat the users control of own system.

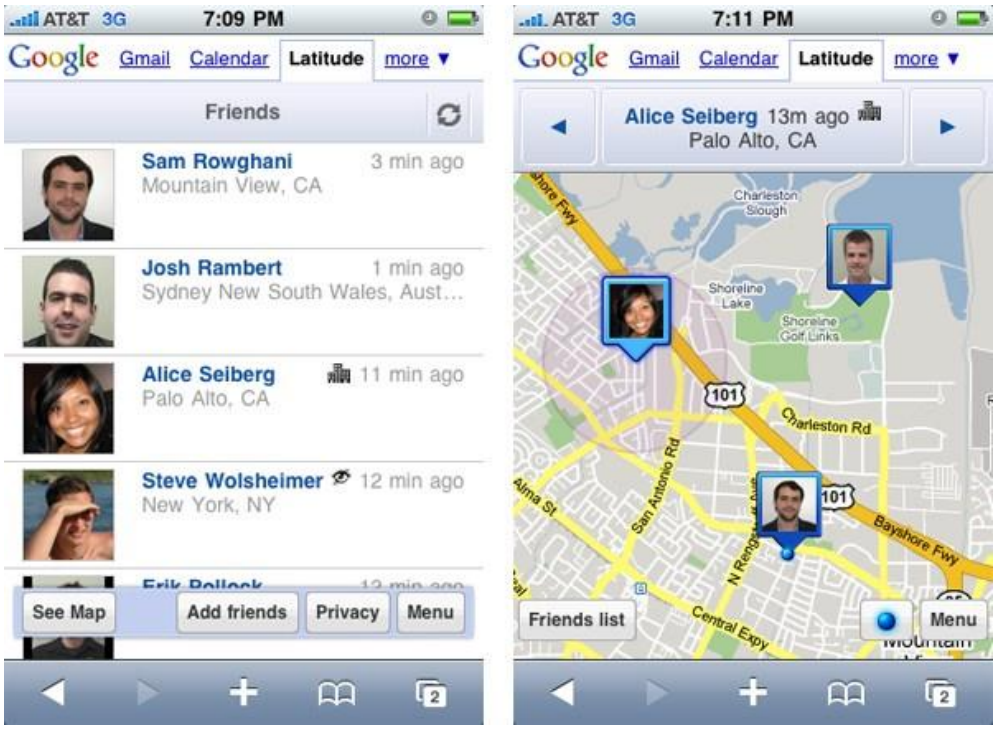


Figure 4 Google latitude is accessible on mobile web page. [8]

2.1.3.2 Facebook

Facebook is a social networking service was found in February 2004. Until July 2011, Facebook has more than 800 million users around the world. Facebook first

launch the website since 2004, and introduce mobile version later. One of the unique features in mobile version is check-in function. It use cell phone GPS functions to identify the nearby place, so users can "check in" and share the current location to friends. "Check in" functions provide list of nearby place for selection assist users to find specific location on map, in addition it shows unknown users who also checked in at same place on the list. Facebook check-in features doesn't rely on Google latitude services, still it offer satisfaction experiences of checking friends' status. It would be better if Facebook could expand with more features such as user location tracking.



Figure 5 Facebook check-in feature

2.1.3.3 WhoIsNear

WhoIsNear is a social networking application developed by Snap Interactive in 2011. Its mobile application is available to iPhone device and mobile website. WhoisNear like other location based application utilize Google map services to provide nearby friends location details. Apart from friends status update, WhoIsNear able to recommend interesting things around user based on rate/vote system, more positive rates will earn recommended content from system.

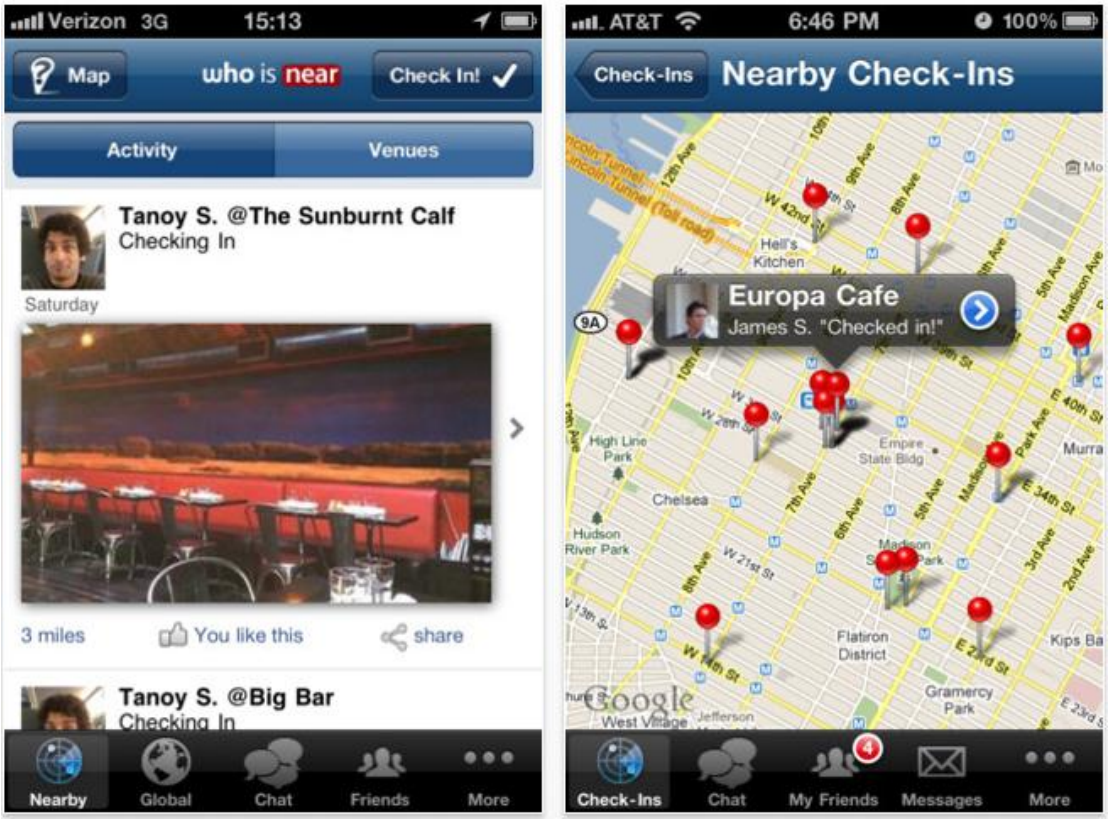


Figure 6 User can recommend the content to other users.

2.1.3.4 Integration of different characteristics

Review 3 popular location-based applications, different location services suit different kind of systems. It is impossible to adopt single type of service since they do not suit for the purpose or achievement for this project, it is necessary to integrate good characteristics to provide better users experiences.

In the study, subscribe the Google latitude services is not an option consider the services stability might corrupted. The proposed solution will combine some unique features on above application. First, location database should maintain in own system like Facebook, it guarantee the flexibility of implementation. Second, limit range of searching area and privacy settings features in Google latitude to be adopted in reducing visual complexity and guarantee users privacy. Third, location suggestion list in Facebook will be used to correct the position error of GPS, ensure users can provide understandable geography locations. Last, rate and vote system added to suggest high rating content to nearby users.

2.2 Video compression technique

An uncompressed video footage takes up about 17MB/sec in camcorder quality, but in order to transfer the file through the Internet with acceptable speed, video compression must applied to reduce size of video file without highly defeats the quality of the video. For most of the video compression are lossy, it major goal is compress very unnecessary data only to achieve good perceptual quality.

2.2.1 How video compression works

Video compression is about how reducing the unnecessary frames or redundant video data so that the digital video file size can minimize. This process involves an algorithm for compress a source video, and require an inverse algorithm to decompress the file and produce as a video. Since different video compression standard are utilize different pair of algorithm (also known as video codec (encoder/decoder)), therefore they are not compatible to each other. Result of using different standard or encoder cannot directly compare with each other because various encoders serve different purpose. For example for mastering optical media should able to delivery better encoded video by using non-real-time encoder while a real-time encoder for video call on hand-held device doesn't.

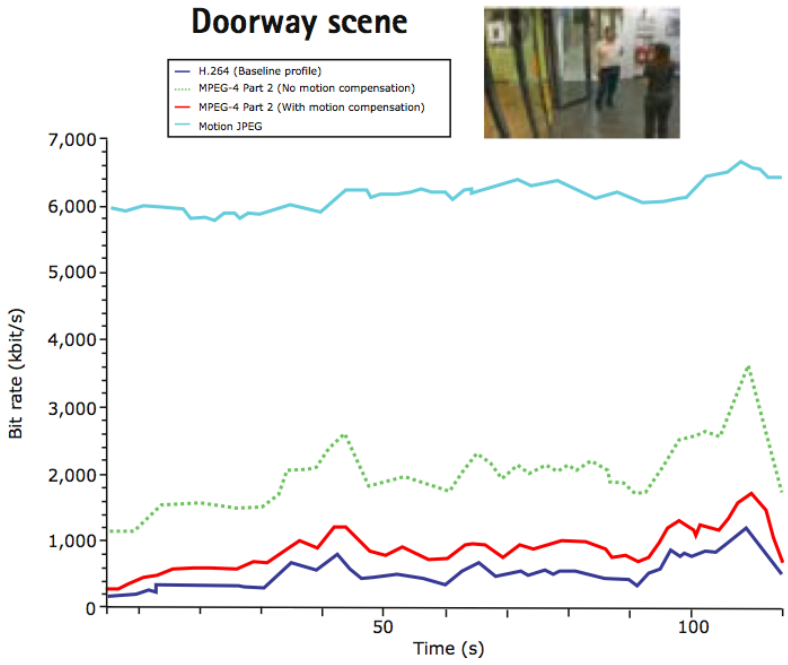


Figure 7 Bit-rate comparison with same image quality among 4 video standards

2.2.2 Video compression method

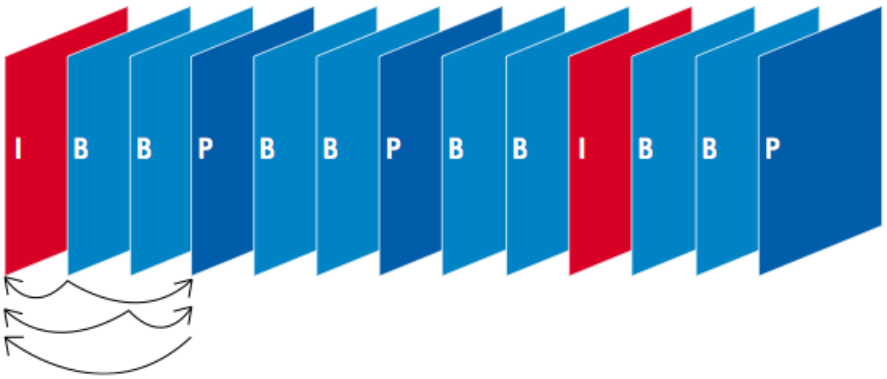


Figure 8 MPEG -1 Motion Compensation

In general most of the encoder profile use different type of frames including I-frames, P-frames and B-frames. An I-frame is a self-contained frame that does not need any reference frame. I-frame always is the first image in a video as a starting point or resynchronization points if bit stream is damaged. P-frame that is predictive inter frame and B-frame is a bi-predictive inter frame. Both frames make references to the previous reference frame or future frame.

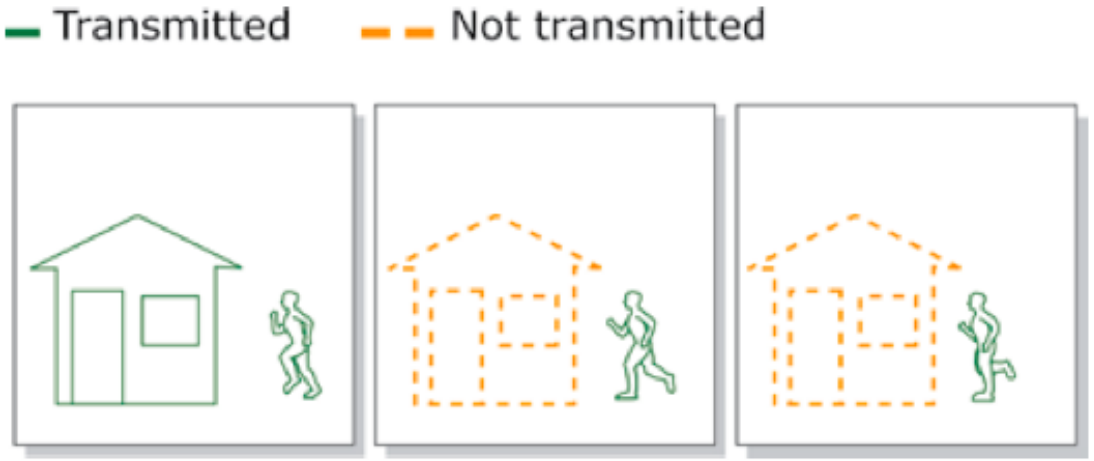


Figure 9 Video data reduction between 3 frames.

Only the first frame (I-frame) is coded entirely. The following two P-frames are references to first frame for the static objects; only motion object data are stored in P-frames. Therefore reducing data can be done in sequences of frames.

There are variety algorithms or methodology can to reduce the video data. Basically, the reducing data occur between I, P, B-frames by remove still object from future frames.

2.2.3 Common compression algorithm

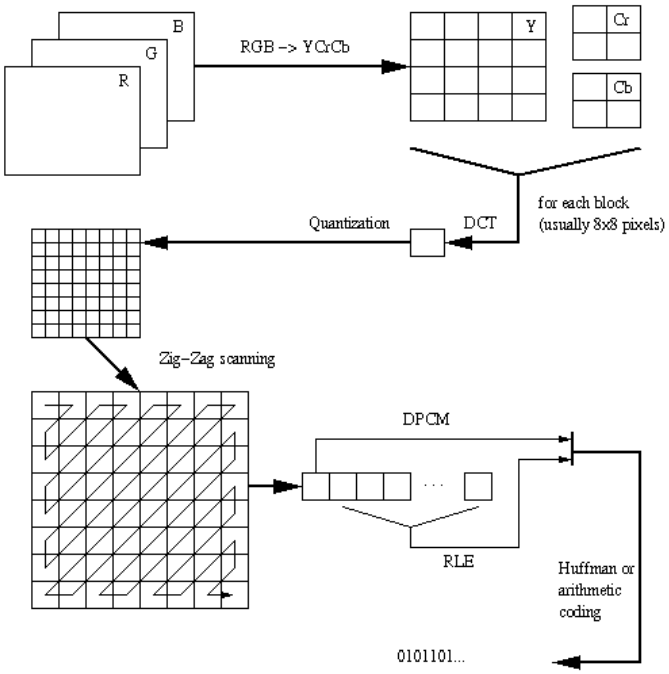


Figure 10 Block diagram of video compression [9]

In order to increase the compression ratio, alternative methodology or additional processes are often added to an new compression standard. Essentially, the compression theory is always likely the same. Anttila (1997) suggested, "The human eye is more sensitive than the location of the brightness of the color change. Therefore, the first image data into a luminance and two chrominance components and chrominance components of the second sample relative to the luminance component. After this step,

the usual lossy compression method used in digital video compression based on discrete cosine transform (DCT) and quantization based. This technique can reduce the space from the picture of high-frequency components, because the human observer is sensitive to low frequency components of the reconstruction error. The purpose of quantization steps is needed to achieve the required accuracy of the DCT coefficients of image quality, is representative. Step in the high-frequency sawtooth-shaped partition coefficient, flow over, because most of them are zero after quantization, Run Length Encoding (RLE) is further compressed. Representative of the upper left corner block encrypted using the difference between the average coefficients on the block (DPCM). The final step in the compression is to reduce the entropy of Huffman or arithmetic coding. Encoded image is often the I-frame (frame), because the coding of other frames do not use the information." [9]

2.2.4 Video compression standards

In the history of video compression standards, over 7 standards were introduced by ITU-T (ITU Telecommunication Standardization Sector), ISO (International Organization for Standardization), IEC (International Electrotechnical Commission) and BBC (British Broadcasting Corporation) since 1984. 3 standards are still in use commonly, they are H.263, MPEG-4 Part 2 and H.264/MPEG-4 AVC.

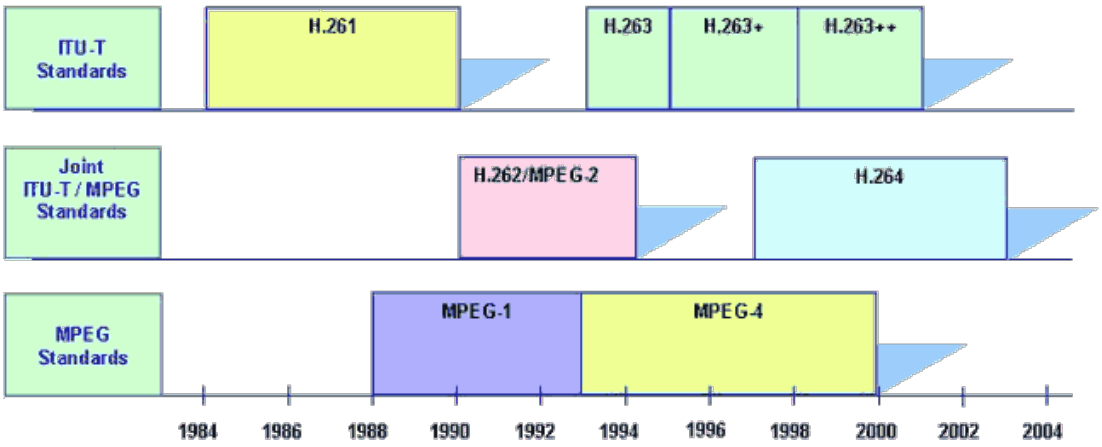


Figure 11 Timeline of Compression standard development [10]

2.2.4.1 MPEG-4 Part 2

MPEG-4 Part2 (MPEG4) provides better compression efficiency than MPEG-2 video, more flexible playback and flexibility of changing the sampling rate [10]. It contains the vast majority of features of MPEG-1 and MPEG-2 and strengths of other video formats. Additionally, it supports simple profiles (SP) and advanced simple profiles (ASP). ASP actually improved featured functions in SP and provided additional streaming methods.

Nonetheless, while ASP covered major roles on in operation, some technical features used in ASP were revealed that harm the compressed product in either speed or quality. The drawback do not indicate MPEG-4 Part2 is not suitable for video streaming, but the fact is it shows the incapability of using it on any low-end processing devices. MPEG-4 Part 2 obviously cannot provide huge bit-rate improvement, although it capable of maintaining acceptable video quality. An unsatisfied user experiences likely occur because of long processing time and slow transmission, not to mention faster draining on battery.

The huge processing power demanding on compression shows that it is unsuitable

to use in mobile device for real-time video streaming purpose.

2.2.4.2 H.263 (3GP)

H.263 video compression standard was designed as a low-bitrate compression on lower processing equipment. It supports for real-time videoconference, Internet applications and mobile phone video playback [11]. The major advantage in H.263 is the motion estimation. It utilize temporary redundancy in video further enhances the encoding performance [12]. It evaluates the previous frame to generate additional B-frames in encoding algorithm and filtering spatial redundancy to reduce large amount of unnecessary frames in encoded video. Chung (2000) suggested packet loss handling" is indispensable for any real-time transmitting purpose encoding algorithm [13]. To address this case, H.263 enhanced the robustness in handling less data loss occurs during video transmission.

Nonetheless, in consider the complexity of implementation, H.263 has not much broadcasting servers that could support. Also H.263 requires codec in 3GPP specification for the legacy services MMS and IMS [14]. System expendability must sacrifice for H.263. Last but not least, the supporting format in H.263 is unplayable in iOS system; real-time recorded playback must seek for other file format for upload. The bad consequences of inconsistence in/out-bound video file format are unavoidable, additional encoders and decoders are required but likely to cause performance issue.

The inadaptability to the client system shows unsuitable side of this compression standard.

2.2.4.3 H.264/MPEG4 AVC



H.264/Mpeg-4 is latest compression standard in recent year that have been fully implemented into major electronic products and praise large amount of recommendation. It is an ideal compression standard that suits for portable device playback. Since H.264 development was based on H.263 compression standard, H.264 offer advantages in H.263 and able to overcome drawbacks in H.263 and H.261.

Sullivan (2004) had performed an experiment on performance and efficiency between different standards. The result of H.264 in bit-rate compression ratio, encoding time and transmission speed is better in any cases. [15]. It is capable providing high reliability in Internet video streaming with over 50% bit-rate reduction but still given a fixed video quality compare with other standard. H.264 has wide range of compatibility across multiple platforms such as adobe media server and iOS Systems. An encoded video file is not necessarily performing different profile encoding again.

In general, H.264 compression is the best option available for multimedia streaming.

2.2.4.4 Conclusion

In video compression study, MPEG-4 Part2 and H.263 compression standard could not offer high ratio compression that H.264 could accomplish. H.264 could encode video in significant low bit-rate that benefits a lot on transmission and limit the latency. Furthermore consider the extendibility of system, MPEG4 AVC is commonly use as High-definition video formatting recent years, implementing such compression to the proposed solution probably makes the implementation flexible for support Higher resolution media stream in the future.

2.3 Media streaming and live streaming

Media streaming refer to deliver digital video content constantly through internet to end-user without by a streaming provider. The streaming means user does not need to completely download the video file in order to watch it, instead user can view the video content, and meanwhile the video content that has not play is downloading on background. Live streaming means the media is live over the network by media publisher, the media content is under recording but the recorded content is available to the end-users.

2.3.1 Streaming bandwidth and storage

Unlike simple file upload/download, to minimize the delay and times to deliver the media content on streaming, it is recommended that publisher should have at least 2.5 Mbits upload bandwidth for streaming movies. For the storage size it can be calculated by following formula.

$$storage\ size = length\ (in\ secs) \times bit\ rate\ (in\ bit/s) / (8 \times 1024 \times 1024)$$

2.3.2 Unicasting and Multicasting

In real world situation, assume a video that requires 100MB storage is ready for on-demanding stream for 1000 people at the same time; this is equivalent to upload 135GB media content per hour. This transfer method consider as Unicast protocol because each end-user demand 1 connection to the media server. While Multicast protocol only require the sever sends out only single stream to serve all viewers.

Lot of internet protocol service different purposes, the most popular such as http, https, ftp, etc. For media transfer there are various network protocols can select, Real-Time Streaming Protocol (RTSP) is a common suggested network protocol designed for entertainment and communicates with media server. More options such as RTMP, RTMPS, RTP also could establish connect to media server but they shows different casting characteristic.

2.3.3 Streaming protocol

To stream the video to target viewer, traditional http file download cannot

accomplish the purpose and consume huge bandwidth to serve multiple users. Consequently, real time streaming protocol takes place as video distributor to establishing media sessions between end users. In recent years, two streaming protocol are in use, they are Real time streaming protocol (RTSP), Real time messaging protocol (RTMP) and HTTP live streaming (HLS).

2.3.3.1 RTSP

Real Time Streaming Protocol (RTSP) is a multimedia streaming protocol used to control the sound or video content. It allows multiple streams on demand control simultaneously [16]. It intelligently divides server load of data demanding to multiple media servers. The Internet traffic can be narrow down in decent range. But RTSP do not emphasis on time synchronization, delay and packet loss in UDP is commonly happened.

RTSP is a charm protocol for streaming. Unfortunately, the effectiveness in cell phone is not that fascinated. Firstly, RTSP demand huge memory on maintaining the streaming session. Secondly, the complexity of implementation stopped the development of RTSP mobile library. Lastly, RTSP have rich support in flash only but it is not for any mobile platform. Steve (2010) proposed the reliability, security and performance problems in flash and concluded iOS will not support flash in the future [17]. In other words, RTSP is not appropriate for streaming in mobile platform.

2.3.3.2 RTMP

Real Time Messaging Protocol (RTMP) similar to RTSP is a multimedia streaming protocol developed by Adobe. It is a TCP-based protocol and delivery stream smoothly by low-latency communication. During the preparation of sending packet, besides the encoded packet, RTMP can assign channel ID, timestamp or size of packet to each packet header and determine delivering in which channel, in typical RTMP session, several channels are available at the same time, this can ensure the completeness of the actual footage and make the error correction.

For the software support, in theory RTMP can be adopted to the mobile platform without issue. However, most of the developer reported that RTMP library is not provided to iOS platform and using RTMP or third party library to achieve it is prohibited.

2.3.3.3 HLS

HTTP Live Streaming (HLS) is an HTTP-based media streaming communication protocol developed by Apple Inc. HLS was designed to deliver multimedia content over HTTP from web server [18]. HLS able to traversing most proxy server or firewall while in UDP-based protocol such as RTP cannot. RTP packets are usually blocked by firewall makes unreliable stream [19]. Unlike Adobe family real time protocol, HLS breaks the stream into a list of http-based file for download. Any browsers or devices do not require protocol implementation before viewing the streaming content. With the popularity of HLS, most media servers on market are updated to support dynamic

protocol switching from RTSP to HLS, for handling content demanded from cell phones.

However, HLS can only be used in Server – Client communication. The client side cannot send the stream to the server by HLS. Still this protocol enhances the reliability and smoothness in distributing content from server to mobile devices.

2.3.3.4 Conclusion

In the study of streaming protocol, RTMP offer promising error control and capable to streaming media in low latency, however in consider the capability to i-Device, RTMP suitable in send/receive streaming content. Consequently, to overcome with upload stream problem from mobile device to server, this study suggest that RTSP should take place the upload parts, meanwhile HLS will be used for view stream part consider it has good compatible for i-device.

2.3.4 Existing video sharing application

2.3.4.1 Qik



Qik is a live video streaming platform owned by Microsoft Skype Division [20], it allow users stream live footage from their cell phone to Internet. Also it supports user make video conferencing between two mobile devices.

Qik launch the iPhone application in mid of Augest 2009, the application not only allow users live stream the video from camera, but also allow user sharing pre-recorded video. The geography location information also updated at the

beginning of live. Qik successfully control the delay between 5-8 seconds approximately in 3G networks.

Qik store the video file on local file system during the stream, to store the video to server after broadcasting, video file must upload entirely from local file system, it can consume huge data usage depends on the file size.

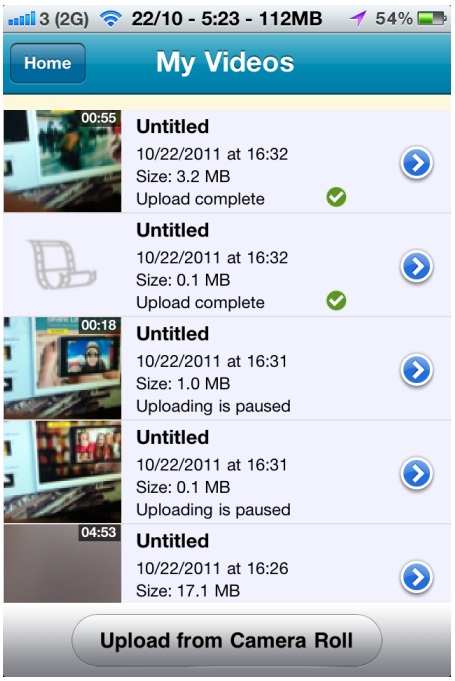


Figure 12 Qik reducing file size by high video compression, it takes 1MB for shoting 20s video footage.



Figure 13 Qik maintain live delay in acceptable range.

2.3.4.2 Stickam



Stickam was found in 2005 developed by Advanced Video Communication, Inc., which is a web live-streaming video services for user-generated content or community [21]. The name of Stickam is refer to the ability of user can "stick" the webcam feed with html codes to embed its flash player to other websites, this unique features become favorite to produce music live shows on website such as MTV, CBS and CES etc because of unnecessary redirect to Stickam site. Alternatively, Stickam let users upload pictures for slide show, songs for music box or video so that users can embed stickam flash player on their personal blog such as blogspot or xanga.

Stickam announced the iPhone application in April of 2011, which allow users to broadcast the live video from iPhone. It ensure satisfy video quality and provide instant chat features between author and audiences. Unfortunately Stickam could not overcome with huge live delay and video lag issue yet.

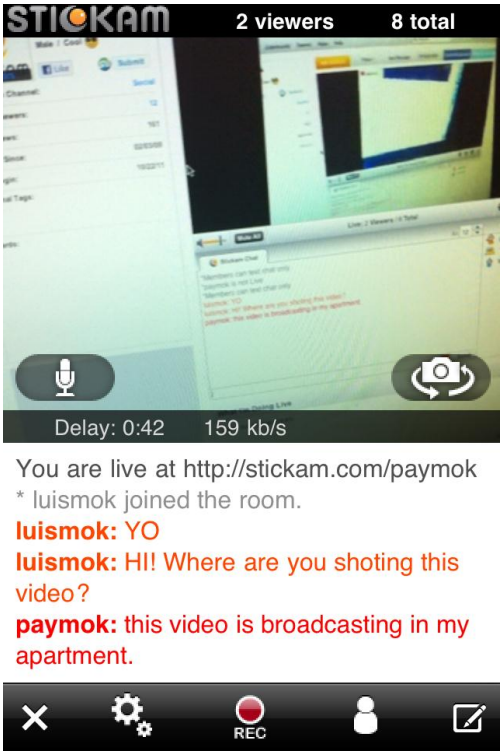


Figure 14 Stickam takes over half minute video delay in broadcasting.

2.3.4.3 UStream

UStream was found in March of 2007 that is a lifecasting video streaming website [22]. UStream also release exclusive software – Ustream Producer, for user to live broadcasting video as different channels and addition screen effect from PC with camera. User can watch the live streaming video from mobile device, besides user can download the mobile Ustream application, which support iPhone, android, and Windows 7 to broadcasting the video or watch the broadcast channel. Like other mobile broadcasting application, it allows users to rate chat or vote in polls for the video. Unfortunately geography locations are not provided. UStream broadcasting application offer stable live streaming experience, also maintain the delayeu not more

than 10 seconds with 3G network. It bitrate can be change dynamically to handle signal interruption maintain video content availability on live by sacrificing video quality. UStream do not require new upload session in order to store the video in server, instead Ustream store the video by received stream during live streaming.

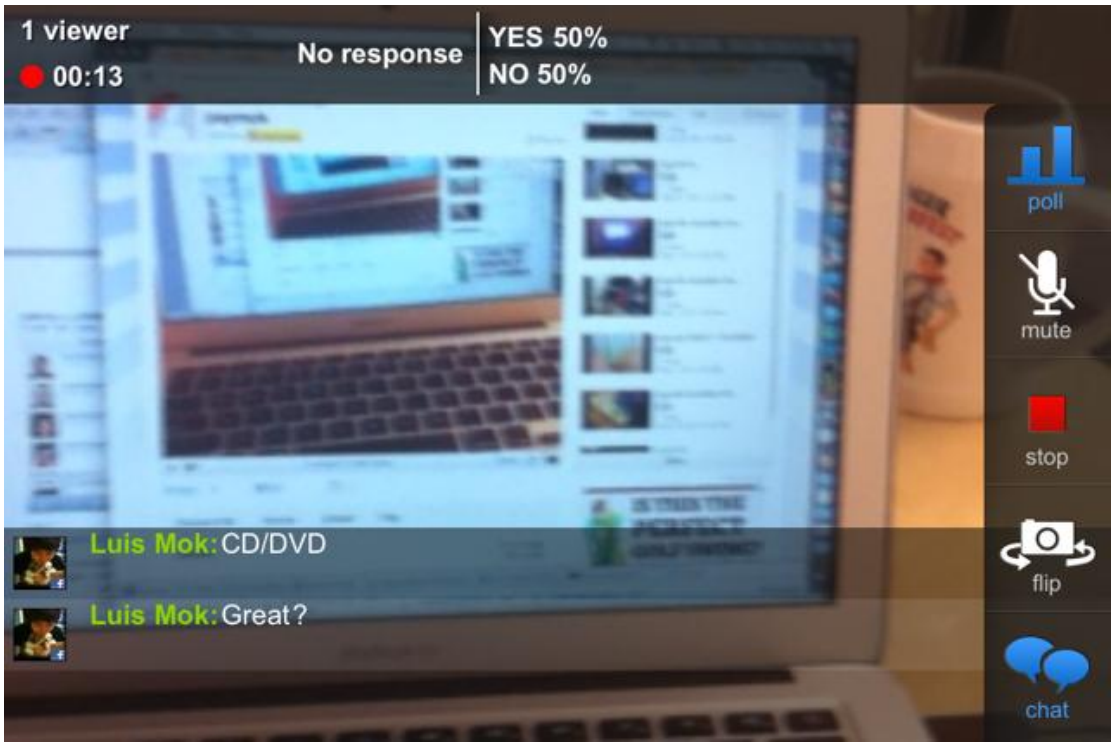


Figure 15 UStream broadcasting provide clear UI and poll function for live streaming.

2.3.4.4 Comparison of Existing Application and Conclusion

After study 3 of the popular stream platform, to overcome transmission delay problem, minimize the resolution of video can control latency in relative small. In addition, archive live footages from received video stream can reduce duplicated cost of network usage. Some features can be adopted to be proposed solution. Such as instant chat, polling, transparent UI design.

The following table provided illustration of the functions comparison between existing application and proposed solution.

	Google latitude	Who is near?	facebook	Qik	Stickam	UStream	Proposed solution
Location update	✓	✓	✓	✗	✗	✗	✓
Instant location update	✓	✓	✗	✗	✗	✗	✓
Video sharing	✗	✗	✓	✓	✓	✓	✓
Video comment	✗	✗	✗	✓	✓	✓	✓
Video search	✗	✗	✗	✓	✓	✓	✓
Instant video sharing	✗	✗	✗	✓	✓	✓	✓
Mobile instant video sharing	✗	✗	✗	✓	✓	✓	✓
Map base view	✓	✓	✗	✗	✗	✗	✓
Mobile application	✓	✓	✓	✓	✓	✓	✓

Figure 16 Table of comparison between existing application and proposed solution

Chapter 3: Design analysis

In this chapter, the various approaches of design analysis are provided. Several possible techniques or platform in this project like network protocol, libraries, operating system or server were evaluated. The conclusion in this chapter provides evaluating result such as performances, apprachof selecting which candidates are the best suitable in this project.

3.1 System requirement

From the problem discussed on previous section, a list of general requirements and characteristics of proposed solution was identified:

- Client – Server architecture
- Server tolerant multiple connection
- Instant update to information to clients
- Account based application
- GPS enabled
- Location tracking in video
- Map based
- Live video broadcasting
- Video stream from server to client
- Video tagging

-

3.2Application Server

In the proposed solution, the system architecture is designed in 2-Tier – Server client. Apart from the real time live media streaming, the system is designed to handle instant geo location update and instant messaging. In order to provide real-time updating functionality, two possible ways has been evaluated for implementing the application server, polling and socket.

3.2.1 PHP Polling and long polling

PHP Polling is a web base method to achieve instant update. Polling is very common and simple to act as constantly broadcasting server, also it has good compatibility because only JavaScript and AJAX is required on client side. It use setInterval() in JavaScript to send a request to server in an interval time, usually json with padding (JSONP) and AJAX is the common way in refresh the latest result. For example, the PHP query to server once to get the latest data in every 5 seconds. But polling also has a lot of disadvantages.

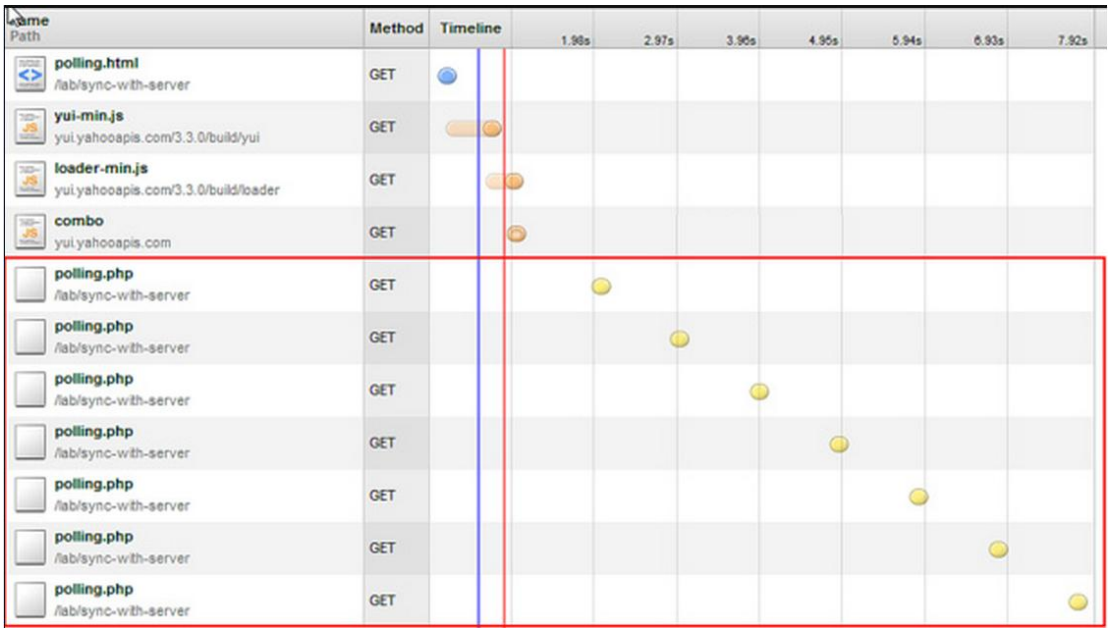


Figure 17 Single connections by PHP Polling invoke server request in an interval time.

Consider a case with 1000 connections connected to the server but nothing updated in 1 minute and polling call for every second.

If inspect the server load there are lot of unnecessary usage request by polling because no updated information needed to inform users. Also consider the polling interval set to 30 seconds, if there are 10 updated committed in a variable in 30 seconds, the polling is failed to reflect the actual transition of that particular variable to the clients.

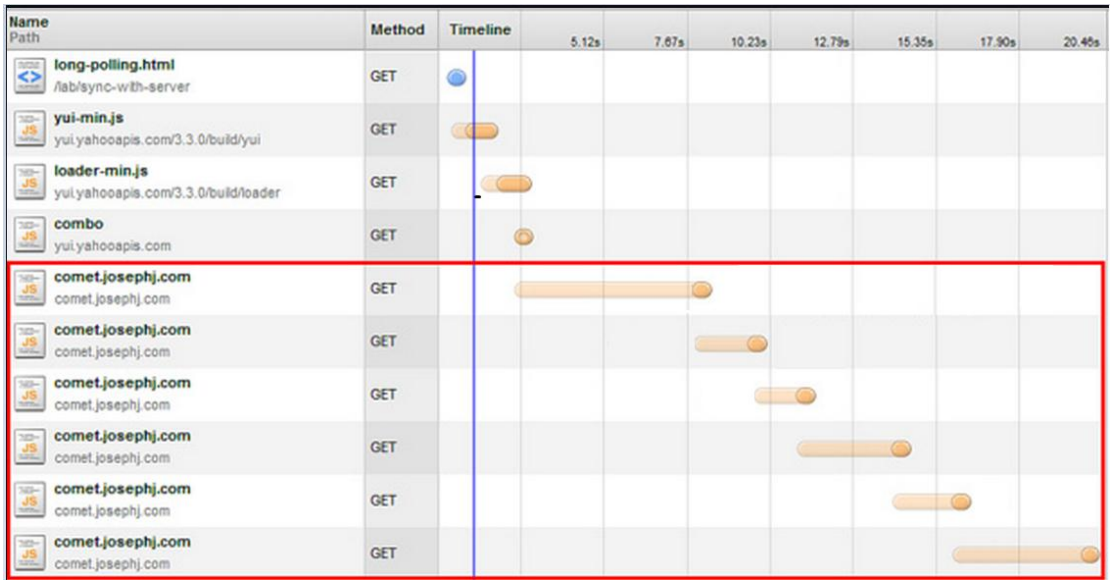


Figure 18 Single connection by long-polling, new connection made right after previous connection received data or timeout. Connections lengths are irregular.

Addressed the issue of polling, another method called long polling is introduced. Long Polling remove the concept of interval call, it establish a connection to web server without disconnected until server response some data to client or timeout reached. It can save more network traffic to server.

But still long-polling cannot solve the huge IO resources usage problem. Once again consider 1000 client long-polling the server at the same time, estimated 4 connections has to be made for each client. 4 * 1000 will cause 4000 requests to webserver in a time unit. Web server is likely to overrun by IO buffer overflow unless the webserver was built with expensive hardware.

Because of the above analysis, web polling is not capable to handle the traffic in proposed solution.

3.2.2 Socket

Nowadays most of the application with instant messaging functionality is utilizes the advance of socket programming. Sockets simplify the concept of the network connection to data stream concept. This data flow existing in each client and server-side port and a stream is like in the interface cable to connect the two passed.
Compare with traditional TCP/IP, Socket do not deal with the underlying operation of the packet header, format, and other details of the handshake. Also unlike standard HTML require generating whole web page code upon request. Socket is much lighter weighted, it only deal with the how message send as a stream to the end point.

When application wants to communicate with another socket application, a network connection has to be established and keep alive between both applications. Each of the socket connection is unique. Each unique socket embedded an IP address and port number and sends to the destination host. With these information socket server can identify individual connected device. Also host disconnection can be instantly detected.

3.2.2.1 Socket server

Consider this solution is requiring handling more than one connection at the same time. If implement with the simplest programming way, a blocking IO calls program, when the call is made, the server operation can be retard because of slow network traffic. The entire program stun if operation is not completed because the IO calls is synchronous. If a program have the needs of serving more than two connections at

runtime, multithreading might be one of the simplest way deal with two connections simultaneously. But consider creation of new thread is expensive, also if thousands or million thread is created handling multiple connection, CPUs are likely to overload. Therefore the socket written in a non-blocking is necessary.

3.2.2.2 select() & kqueue() & epoll()

In non-blocking socket Nick Mathewson (2012) indicated the oldest solution select () method give awful with number of socket is too high. To overcome this problem, different operating systems provide several approaches to replace select (), and provide give O(1) time complexity for socket adding, removing and notice socket ready to IO. [23]

Unfortunately, no interface can ubiquitous. BSD used kqueue(), Linux give epoll() and Solaris offer dev/poll as replacement. For windows development, IOCP (IO Completion Ports) is provided but not as efficiency as others.

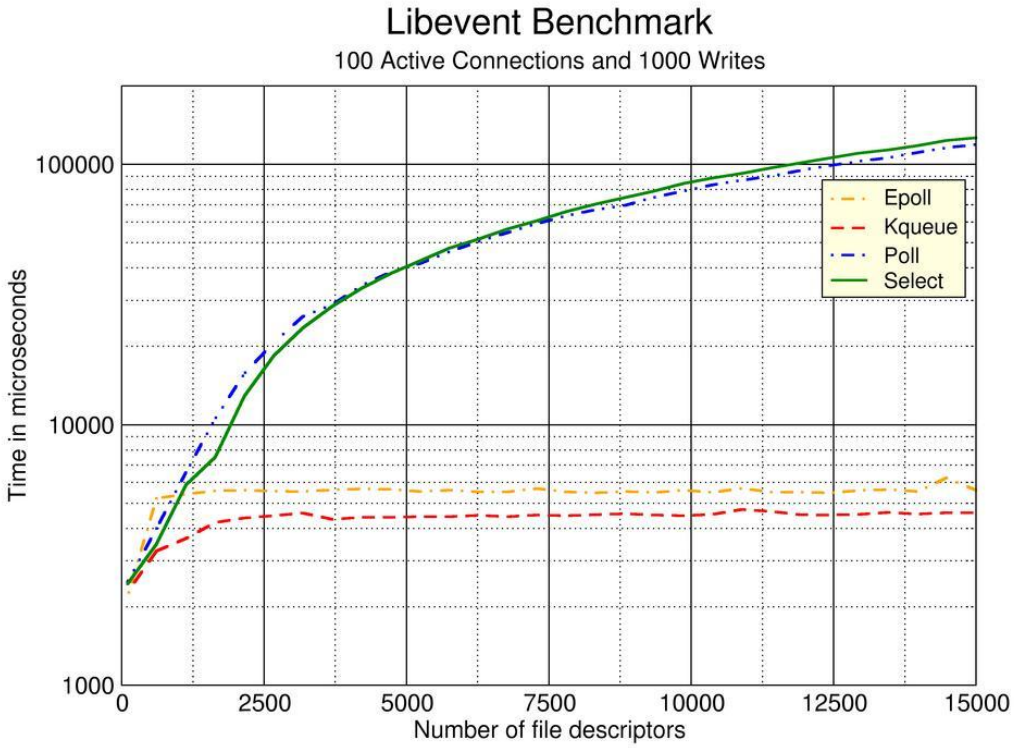


Figure 19 Benchmark of Epoll(), Kqueue(), poll() and select() with 100 connecitons [24]

Conclusion

Because these sack of reasons, and consider the development convenience, Linux with native C has been chosen to be the socket server host platform and programming language.

3.3Media Streaming

Multimedia streaming is the core function in this project. When implement a media stream server, unlike a web base server, there are various compatibility problems need to be take care. For example protocol to be use, which brand of the media stream

server even which type of media data. A wrong combination of above may result hard time on implementation.

3.3.1 Streaming protocol

In tradition, RTMP would be the first consideration streaming protocol in implementing the video streaming because RTMP offer many benefits like stream data info, and satiability of connection. Unfortunately, the follow justification shows RTMP is not suitable in this project.

When the RTMP comes to the mobile, RTMP appear multiple constraints, which make RTMP hard to use on mobile devices.

First, RTMP usually blocked by firewall. As a mobile application, connection base change is frequently occurring by cellular network. RTMP use 1935 and 443 port by default, many firewall and router are blocked these two ports because of traffic control, which makes the application functioning become unstable.

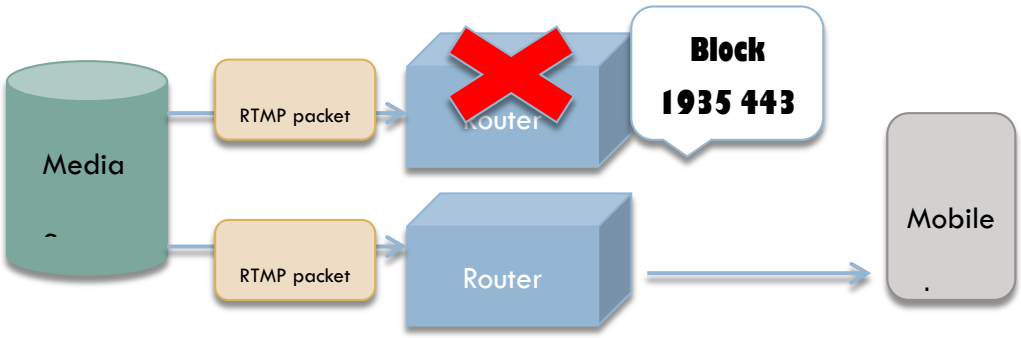


Figure 20 RTMP port usually blocked by firewall

Second, RTMP library in IOS have lot of compatibility problems. While android have an open sources RTMP/RTSP library, iOS does not official announce one. Instead

some third party library like dumpRTMP claimed that enabled RTMP could be use. But due to the insufficient information, not many developers are able to compile and deploy the application with these libraries successfully. Also for each version of iPhone/iPod touch, the device architecture is changed accordingly. Such as iPhone 3G and 3Gs are armv6 architecture based, iPhone 4/iPhone4s are armv7 architecture. When a newer version of devices or firmware released, RTMP library likely to re-compile and newer issue coming up. Without any on-going support to these static libraries, as a result, it makes the development very difficult.

Last, Apple blocks RTMP application. According to Apple requirements for Apps, all submitted application that delivers a video exceeds 10minutes duration or 5MB data in five minutes over cellular networks, the application must use HTTP Live streaming. With this clear statement by apple, Only HLS is accepted as media protocol in server-client stream.

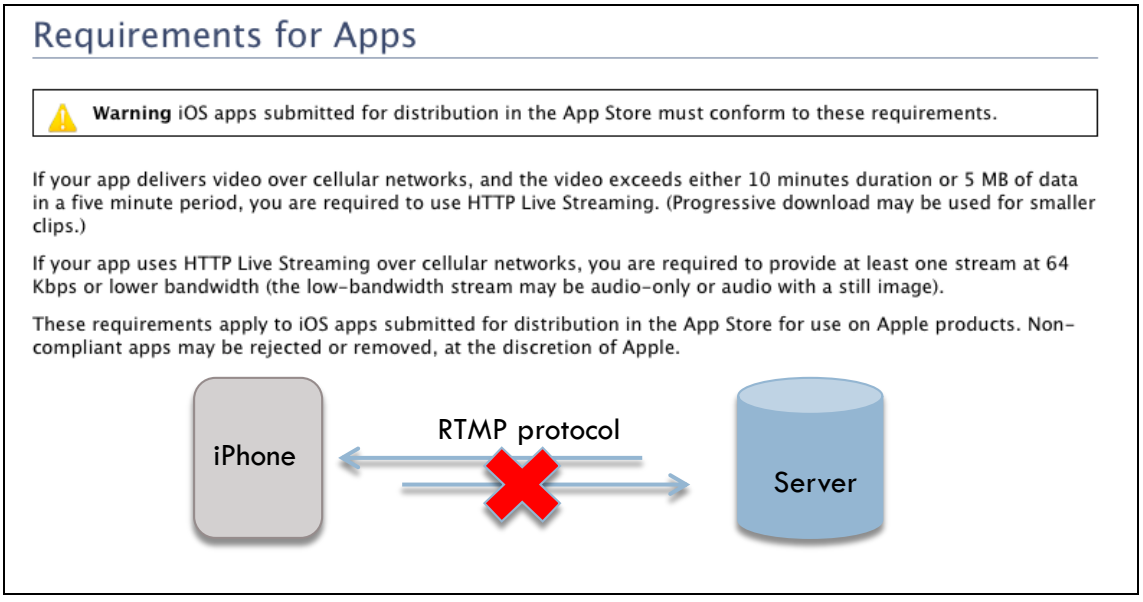


Figure 21 Statement of Requirements for App in Apple iOS Developer Library [25]

In order to keep the value of this project, and without breaking apple's rule, this project abandons the usage of RTMP support.

Apple highly suggested that perform HTTP Live Streaming (HLS) for video streaming instead of other protocol. In cooperate with Apple's rules; this project will use HLS for distributing video. For the client to server video streaming, the alternative solution will introduce in chapter 5.

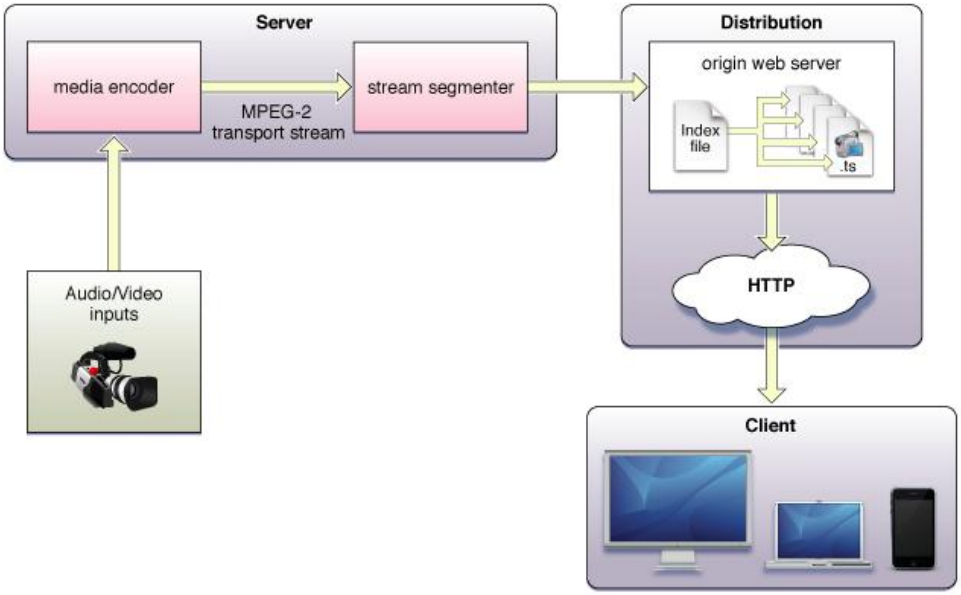


Figure 22Overview of HTTP Live Streaming [26]

Chapter 4: Project equipment

Because of the iPhone development is different from other platform like windows mobile and android, the development tools and developer program are presented and briefly introduced. In this section, the resources that needed to use or development are provided.

4.1 IPhone development

To be an iPhone developer, Apple Inc. offer free IDE Xcode for iPhone application development. Developer can deploy the application to the provided iPhone simulator for testing purpose. If an application has needs to deploy on a real iOS device, developer must enroll as a member of *iOS developer program*[27].

Apple has provided three type of programs for developers. They are *Developer*, *Enterprise* and *University*.

- **Developer Program:** This program for developer who want put the application to the App Store[] for sell or free. This program also allow develop put advertisement by iAd[27] on their apps for profit. [27]
- **Enterprise Program**: This program for companies and organizations to create application for in-house distribution or update only, app store and advertisement is not supported in this program. Also the enrollment is exclusive to the cooperation with A Dun & Bradstreet Number (DUNS). [27]
- **University Program**: This program is for the education institutions to introducing iOS

development. Qualified higher education institution can apply and distribute the developer profile to student team. This is the only program that free of charges in 3 of the enrollment program but no app distribution support on App Store or in-house.

Compare Programs

	Developer Program For developers creating free and paid iOS apps for distribution on the App Store. Learn more	**Enterprise Program** For companies and organizations creating proprietary iOS apps for internal distribution. Learn more	**University Program** For higher education institutions introducing iOS app development into their curriculum. Learn more
Pre-release Versions of iOS SDK	✓	✓	N/A
GM Versions of iOS SDK	✓	✓	✓
Test on iPad, iPhone, and iPod touch	✓	✓	✓
Code-level Technical Support	✓	✓	N/A
Ad Hoc Distribution	✓	✓	N/A
App Store Distribution	✓	N/A	N/A
Custom B2B App Distribution	✓	N/A	N/A
iAd Network and iAd Producer	✓	N/A	N/A
In-house Distribution	N/A	✓	N/A
Cost	$99 year	$299 year	Free
Requirements	A valid credit card for purchase.	A Dun & Bradstreet Number (DUNS) is required to enroll.	The University Program is only available to qualified, degree granting, higher education institutions.

Figure 23 Comparison between 3 iPhone Enrollment Programs [27]

From the figure showing above, noted that ad hoc distribution is allow developer deploy the application by USB connector, email, website or server to a registered device up to 100 users.

In this project the developer has enrolled the “Developer Program” for develop this project. The other two programs, which are not suitable for individual or small group of developer, enroll. “Enterprise” program require legal document registered as a company, also the in house distribution is not necessary to this project. “University” Program is for institution only; individual student is not qualified to enroll. Essentially only “test and iPad, iPhone and iPod touch” feature is suit the purpose of this

project. The application in this project has no intention to distribute to app store.

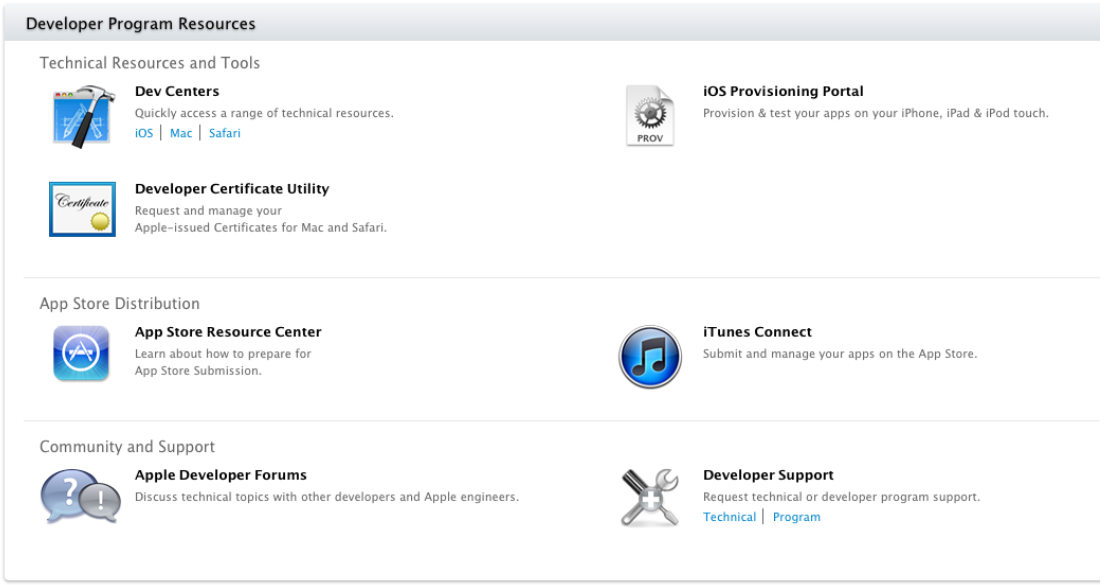


Figure 24 Member center in apple iOS dev center

As an registered member in iPhone developer program, rich resources are provided to developer. Such as dev centers provide latest example of using iOS cocoa API, iOS provisioning portal manging developer's grant ad-hoc devices and apple developer forums provide a discuss platform about mac, iOS or safari development with engineering of apple and developer around the world.

4.2IPhone



Figure 25 Technical Specification of iPhone 4 [28]

IPhone 4 is used as testing device in this project. The iPhone 4 is running in iPhone SDK 5.0.1. The A-GPS, fast CPU clock and the number of video formats supported are the reason of choosing iPhone as the development platform. The embedded camera and microphone offer the enable the live streaming functionality of this project.

In theory this project should able to support iPhone4 or above model, iPod touch 3rd generation or above model and iPad2 or above model.

4.3IPhone SDK 5.0



On the time of project kickoff, the latest available iOS SDK release is version 5.0. In this project the require library is not included in iOS 5.0. Although the based SDK in project is on 5.0, in theory project should compliable in iOS 4.3 or above without any issue on deployment. The following key frameworks are important and used in the project.

- **MapKit**: Provide map base view
- **CoreLocation**: Help user determine current location related to the iPhone.
- **Core Services**: provide high level framework for perform network tasks.
- **AVFoundation**: available on iOS4 provide high-level API in processing media.
-

4.4Xcode 4.2 on MAC OSX

Xcode is the IDE for iPhone application development, it delivery with the iOS SDK.



Figure 26 XCode 4.2 Welcome startup screen

4.5Ubuntu 11.10

uBuntu is an operating system that based on Debian Linux Distribution and a free open source software. Because of the epoll() nonblocking IO feature needed in the socket server, this platform has been chosen to be the server platform in hosting the socket server, web server and media server.

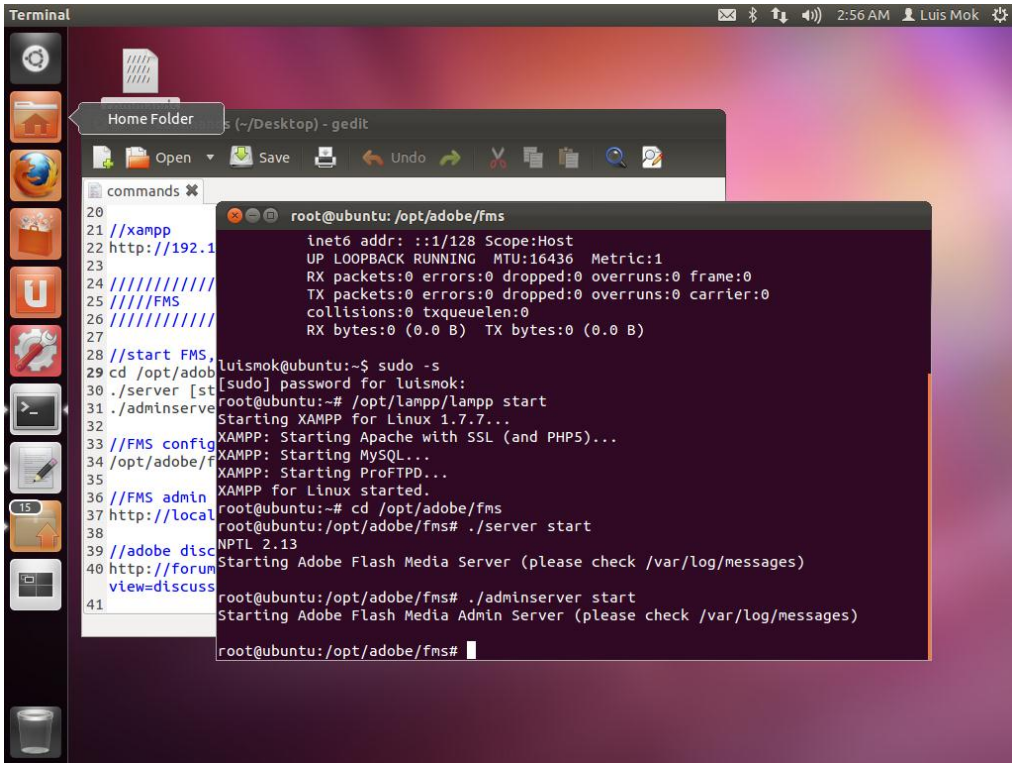


Figure 27 UBuntu 11.10 x64 User interface

4.6Wowza Media Server

Wowza media server 3 (WMS3) is a media streaming solution developed by Wowza Media System Inc. that able to distribute video contents connects in RTMP or HLS. Live streaming or Video on Demand (VOD) is both available to this server.

An evaluation version of Wowza media server 3 is used in this project. The evaluating version WMS3 delivers the same functionality as standard WMS3 only limited 10-connections in/out with 30days evaluating times. HTTP Dynamic Streaming and HLS streaming to iOS also provided in evaluating version. This version of Media Server is suitable for deploy a small scale of solution.

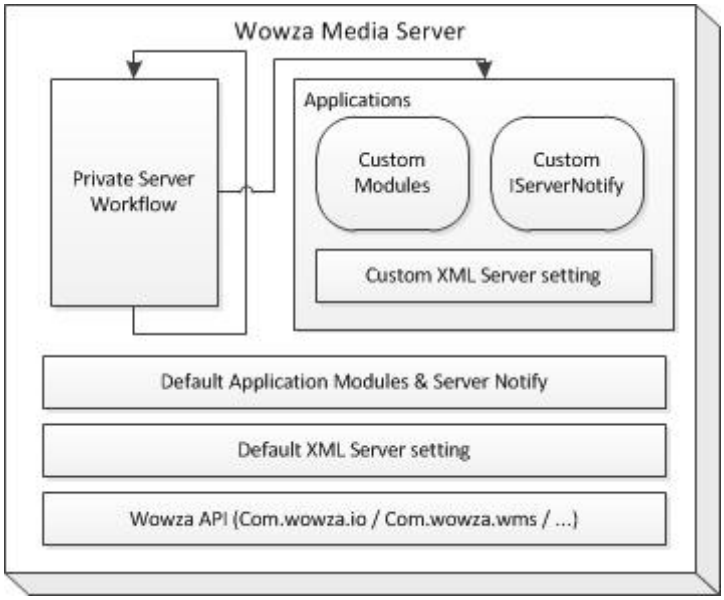


Figure 28 Wowza media Server architecture

Wowza Media Server consists of many applications along with XML files as settings. When new features going to append to server, a new application is added to

the list of applications. There are two types of applications, Module and IServerNotify. A Module usually startup triggered by connected user, module may not be run if no one accessing while IServerNotify are triggered on server init, pre/post shutdown. Applications are encrypted as a *.jar file and loaded to the server on startup.



Wowza Media sever's modules are all java based. Therefore Wowza Media System Inc. provided an IDE for developer develop modules to the server. The IDE is a plugin to the eclipse IDE that makes eclipse can debug the server on runtime.

4.7LAMPP

XAMPP is an open-sources cross-platfrom web server solution package with Apache HTTP Server and MySQL database. LAMPP is the linux version of XAMPP. It can be used to host PHP or Perl application.

Chapter 5: Methodologies

In this chapter, the high level methodologies in implementing various functions are provided. In this part the working theory of major function in the proposed system and overview of the system is being introducing. For a detailed implementation methodology, it will be discussed in later chapter (Chapter 7).

5.1 System Overview

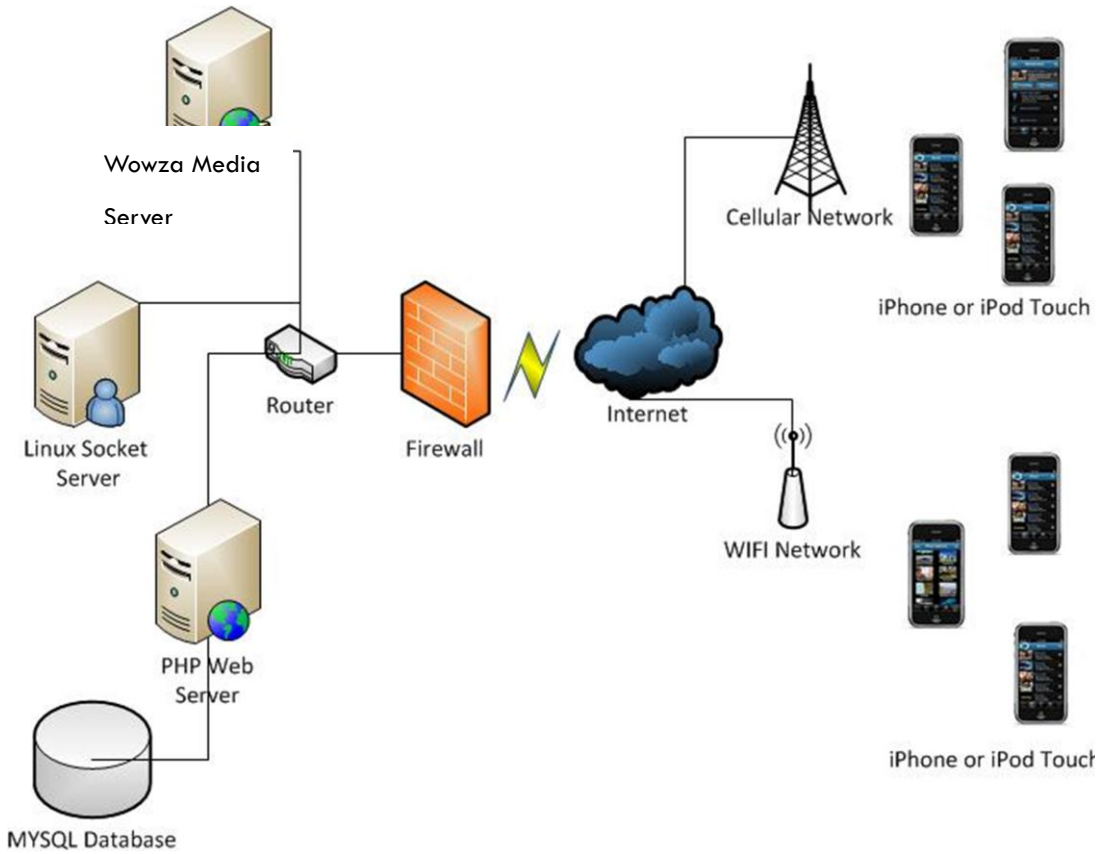


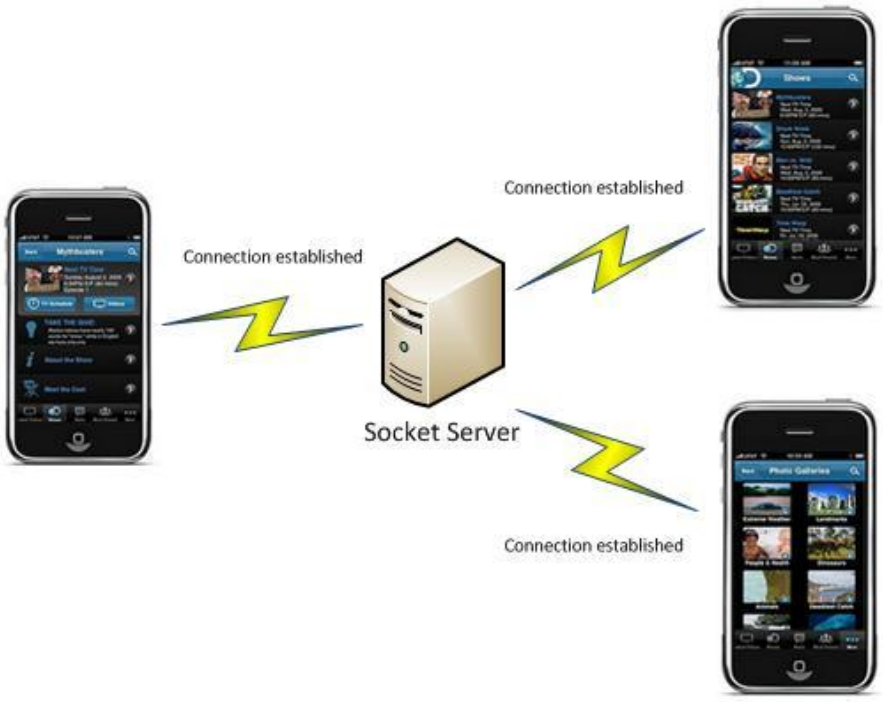
Figure 29 High-level System Architecture of proposed solution

5.2Message broadcasting

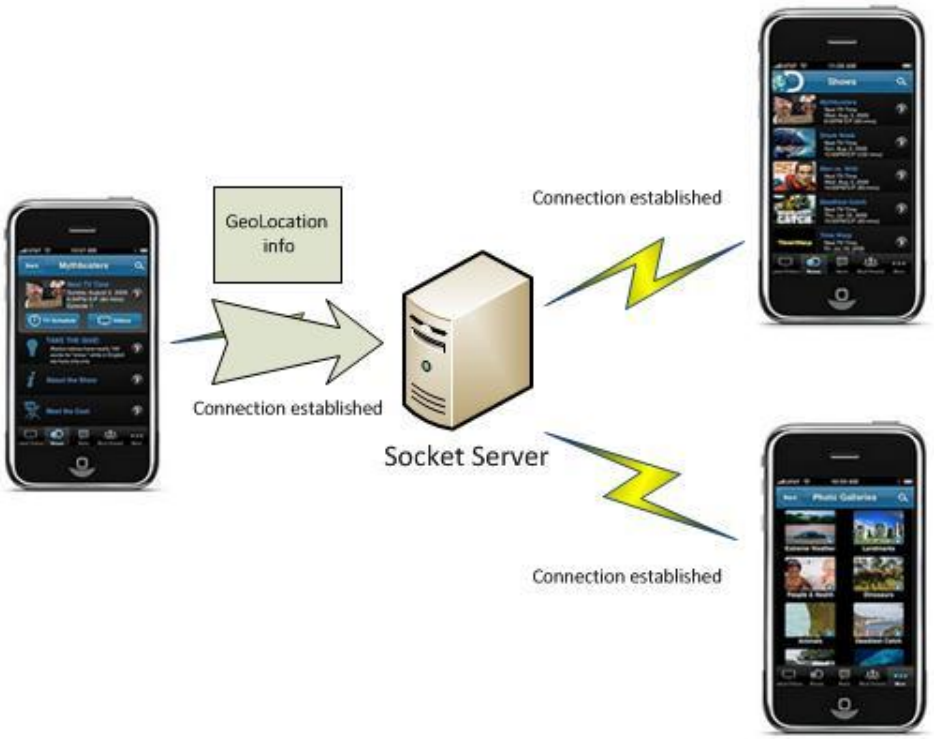
In this proposed solution, the application is designed as 2-Tier architecture. Each user has to make connection to the server while their mobile phone is connected to WiFi or cellular network. The server can connect up to hundred connections without dropping any performances.

To provide instant geolocation update, or instant message broadcasting with very less delay, all connections made from iPhone will connect to a C base socket server. All message sent by a user will broadcast to other connected users. Server will do the action of broadcasting; messages will broadcast to particular users depends on type of message.

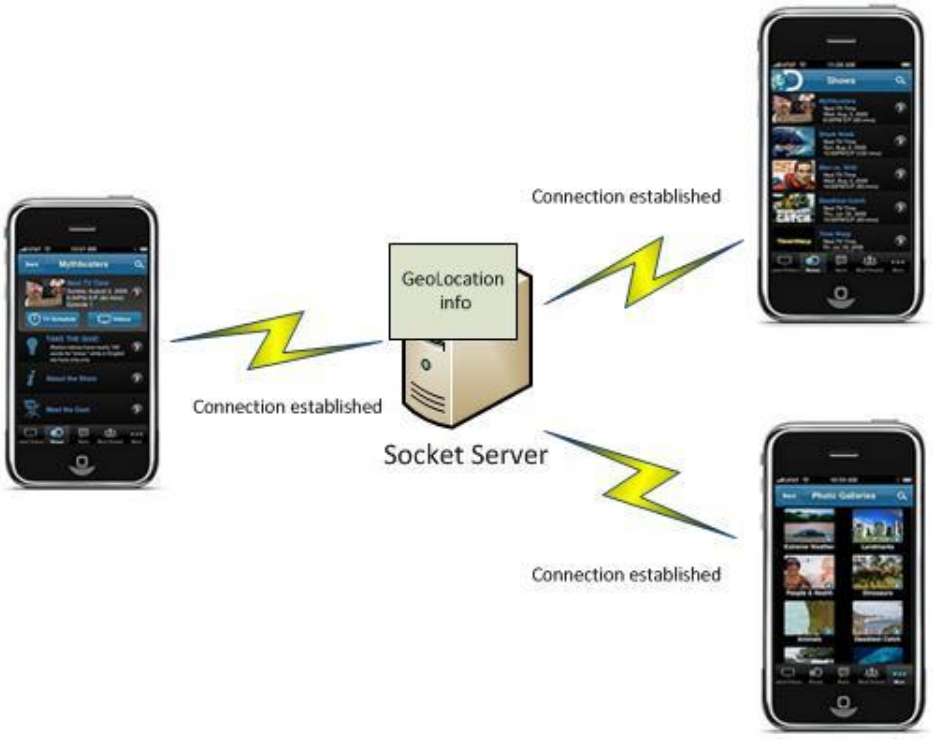
The following figures give the example how's the geo location to be sent from client A to client B.



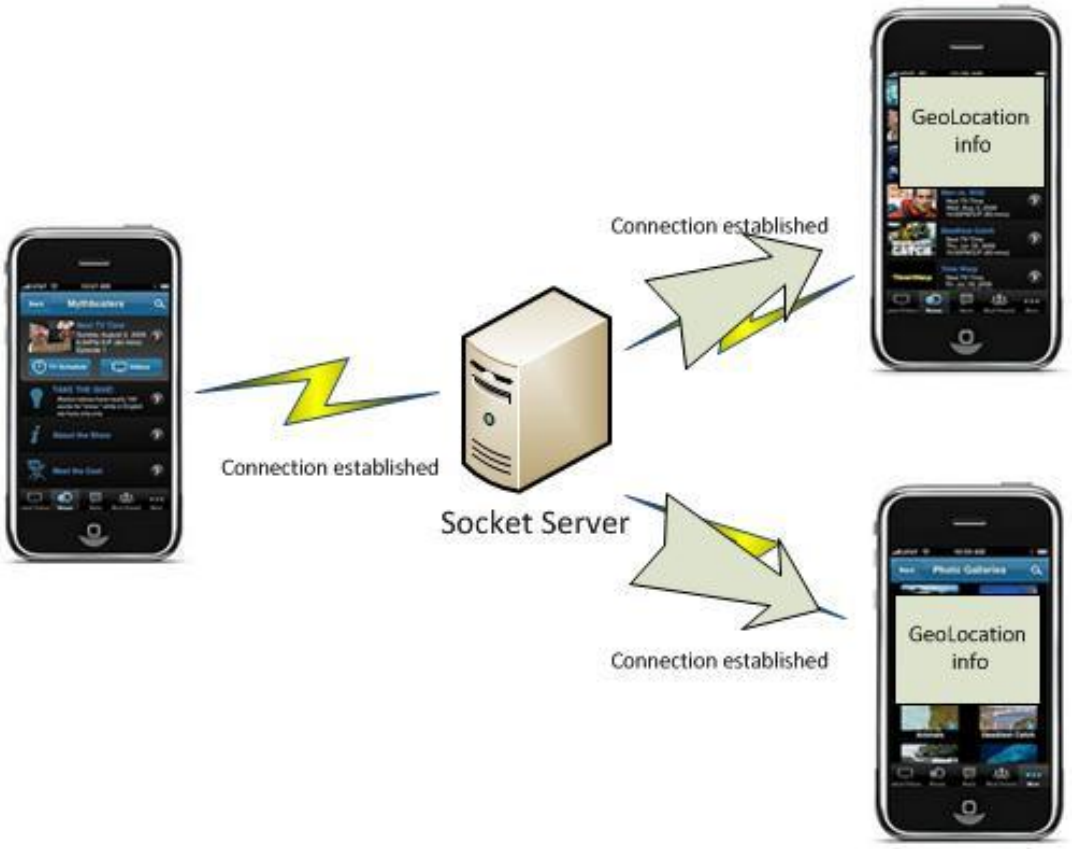
1. The connection between client device and socket must established first.



2. The sender sends the message to server by socket



3. Server receive the message, determine which client needed to be send



4. The message has processed in server and forwarded to other client without send back to original sender.

As discussed on system analysis, the socket server is hosting on linux platform for the performance purpose. Libevent library – one of the revolutionized with high performances asynchronous IO is employed in developing the socket server. Libevent is an async library that capable to writing a high speed portable non-blocking IO. The library is written in C language able to across all platforms and compatible with ten thousands active sockets.

5.3Video Streaming

Refer to the previous chapter (Chapter 3.3), Due to the RTMP limitation; the streaming from iPhone to Server will not transfer through RTMP protocol. Instead the video transfer will occur through socket server.

To simulate the live video stream, the video files constantly produced in local file system from an interval and send it to the socket server. The procedure of the entire streaming is listed below:

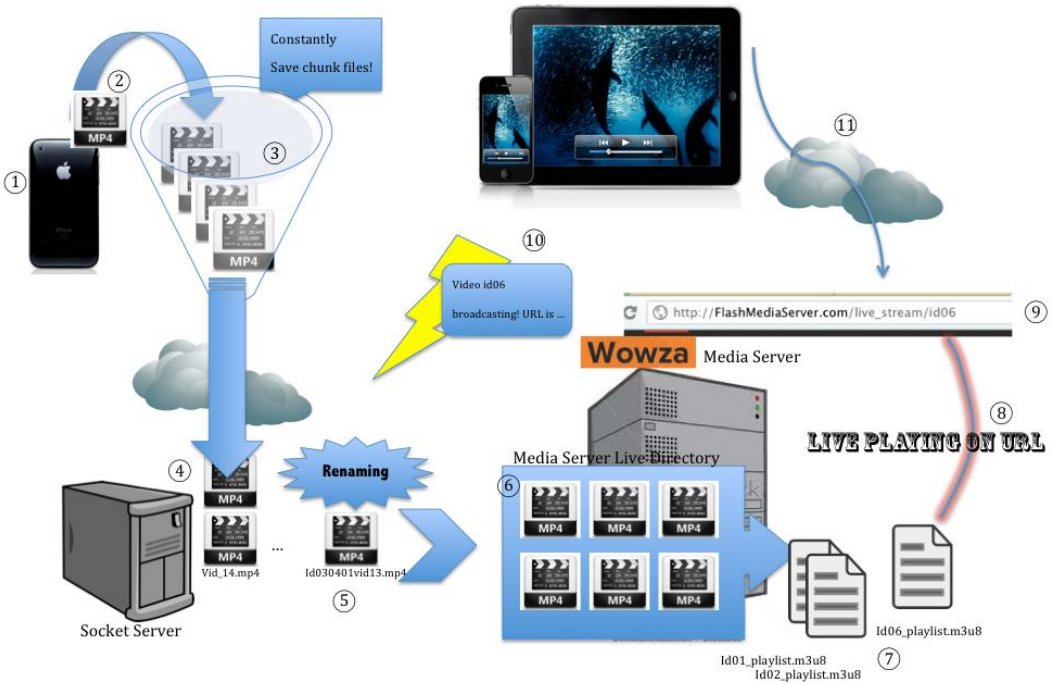


Figure 30 Overview live broadcasting process flow diagram

Steps Description:

1. Video capture from camera
2. Record 8 seconds video
3. Saved the 8 seconds video to local file system (Step 2 and 3 process iteratively)
4. Program send the saved video file to socket server

5. Socket server received the data, rename the file name by receive sequences.
6. Socket server save the file to the Media Server live streaming directory
7. WMS update the live streaming playlist by checking directory presented files
8. WMS play the video files in live stream
9. WMS makes the live stream URL available
10. Socket server broadcast streaming session is available to other client's application.
11. Other clients receive the live stream URL and play the video live streaming.

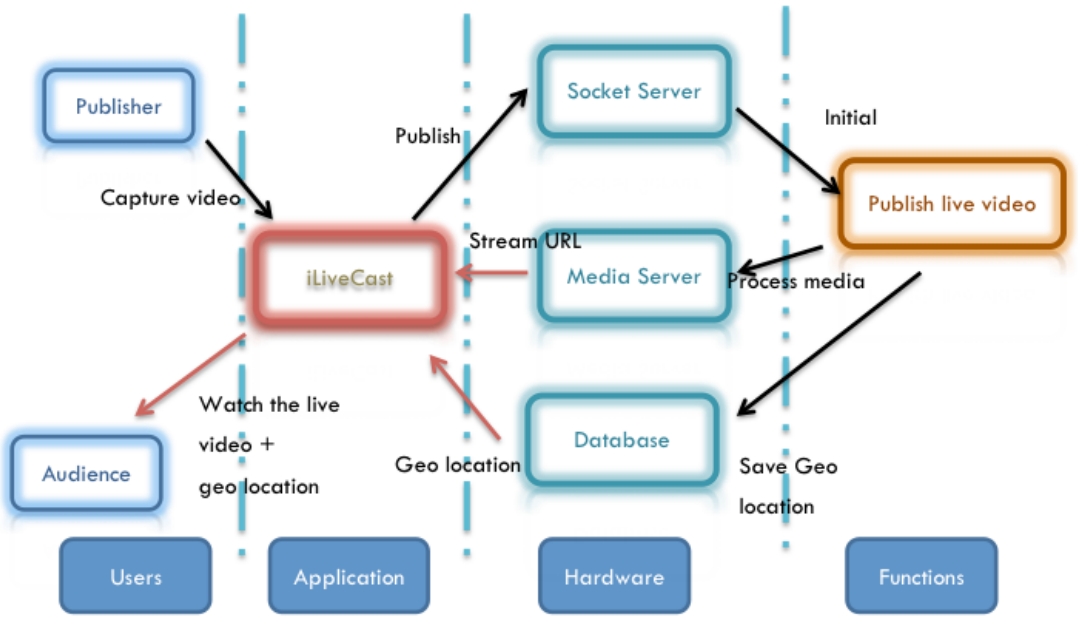


Figure 31 Live streaming architecture diagram

This approach of streaming video can maintain the stream delay in 4-10seconds. The reliability of data transfer can maintain as the socket use TCP/IP connection. The video distribution will deliver by HTTP Live streaming, which can reduce the server load of streaming in compare with RTMP methods.

5.3.1 Video encoding setting

The video broadcasting in this project is limited to iOS platform; therefore the video compression setting for producing the video file can be pre-defined. The following is the configuration of encoding:

Video Dimension:	480 x 320
Video Codec:	AVVideoCodec H.264
Video Frame Rate:	30
Audio Format:	MPEG4 AAC
Audio Channels:	1
Audio SampleRate:	44100
AudioEncoder BitRate:	64000

Figure 32 Table of defined Video Compression setting

5.4JSON data interchange

In this project, JSON [Javascript Object Notation] format is wildly used as a data interchange between iOS application, socket server and php web page. JSON is a completely human-readable which also independent to any programming language, Hence it can be adopted in any environment.

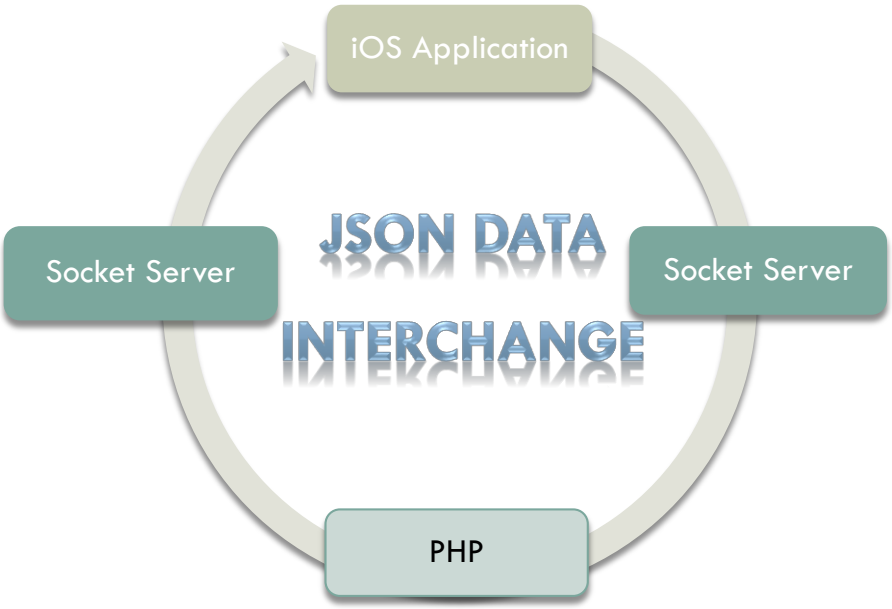


Figure 33 Data Interchange diagram between server, database and iOS App by JSON

JSON data type is simply a collection of name/value pairs. It can be representing in a simple string format like:

{VARIABLE1 : VALUE, VARIABLE2 : VALUE2}

Also JSON can contain a collection of data. The example below shows the how data can store in the JSON and send to server. JSON can embed a data in an object without destruct a OBJECT data structure.

{ VARIABLE 1: VALUE1,

VARIABLE 2: VALUE2,

‘OBJECT1’ : {OBJ_VAR1 : VALUE1, OBJ_VAR2 : VALUE2 },

‘ARRAY1’: [‘ABC’, ‘CDE’, EFG’]

}

On the interface on the server, application has to define same message protocol in order to decode the JSON message. Without a defined protocol, the JSON send from one interface cannot be identifying as a recognized message.

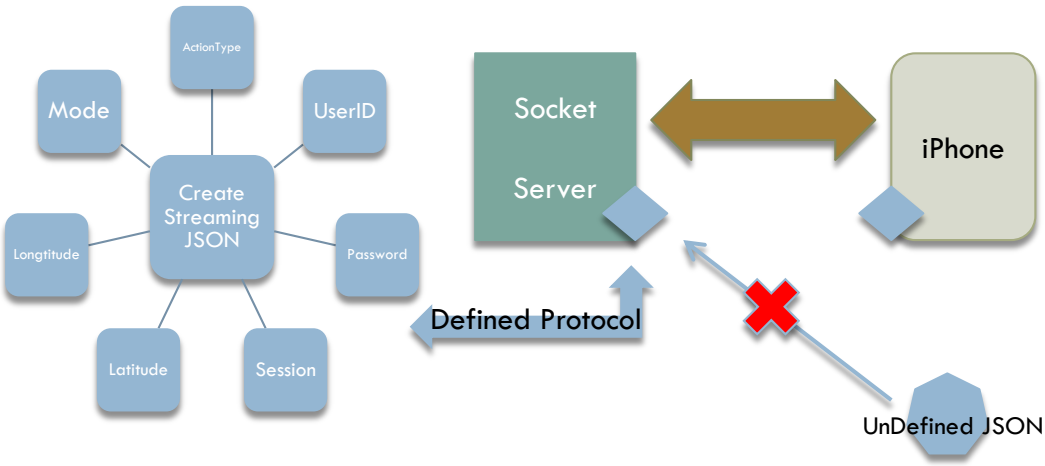


Figure 34 JSON Protocol Structure Diagram. Defined protocol must implement on both side interface. Server does not understand undefined JSON.

Various JSON Protocol is defined on server and application for data interchange. Before data is prepared and ready to send out, a 2-byte data, which is the length of JSON is appending on Data header. Because in socket server, when the buffer first received and transcode data as string, a memory needed to allocated and store the string. When allocating a proper size of the memory for storing the string in order to minimize the memory used in system, the length info can give a server a signal where to put the ‘\0’ or the char array. So that system will not waste memory storing empty block data. Byte length info not only provide advantages of memory management, also provide security purpose in receive data; this practice is not very common in JSON data

interchange. If the data protocols remain secret, a suspicious source tries to send an unknown data to server. The server will determine the data JSON structure and byte value to determine received data is from a trusted source.

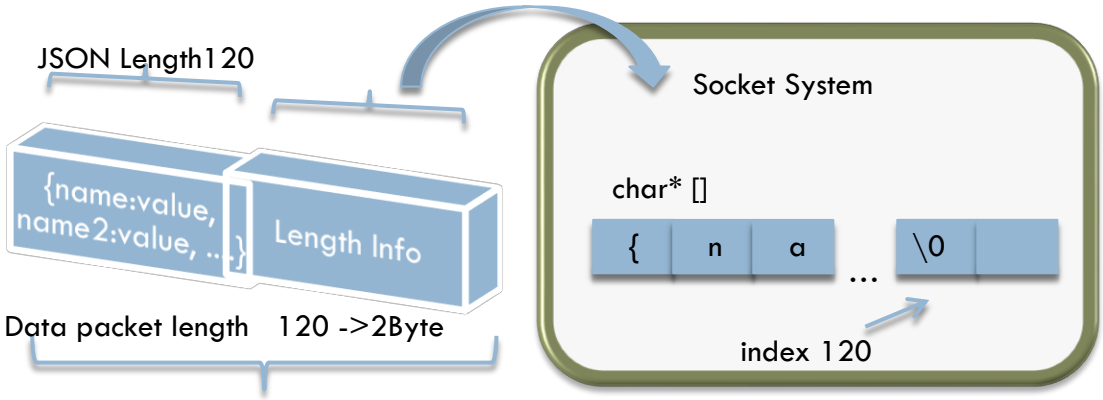


Figure 35 Packet structure and length info usage in prepare char[] data

5.5Map view and Position tracking

Follow Diagram shows the overview of location update:

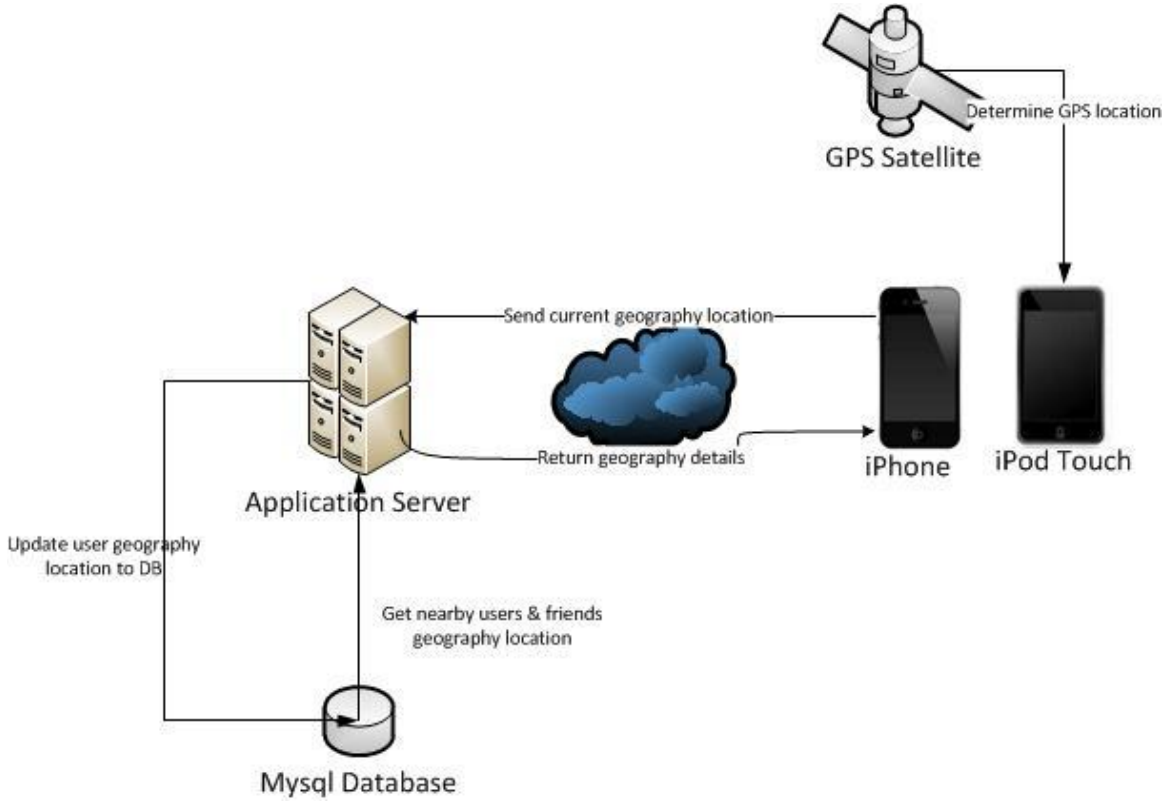


Figure 36 System flow of update user position

The positioning function enabled at the startup of the application. The client side application would determine the location by A-GPS and updated the location details to server side database. Once the location updated, application server retrieve list of nearby publishers. Application server sends the list to the user with coordinates for update pins on the map. On the other hand, the user coordinate and online status will update to others users mobile device accordingly. This operation performs 10 seconds periodically.

Position tracking on map view

IPhone device is deliver build in A-GPS chips that enable the global positioning, the API provided by iOS SDK 5.0 simply return the current position longitude and latitude information. With this information, the mapKit provided in iOS SDK 5.0 could display the position on map.



Figure 37 Example of tracking position on map

Push Location details to server

Memory management in server is significant important, therefore instant location update should not over sending unnecessary latitude in an insignificant distance change. The Core Location framework is capable to detect distances change within a specified meter. A circle range with 30-meter radius calculates the distance. On the following example showing a trace of series location update from bottom to top. The purple pin represent a location update to server is occurred, observed the update is occurred only

the current location is out of previous update circle range.

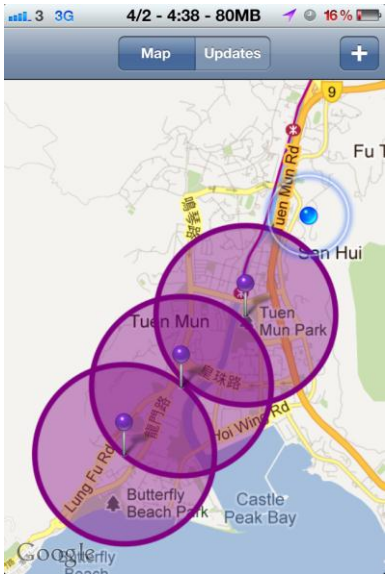


Figure 38 Example of location update occur when out of a circle range

Getting location

In this proposed system, all the longitude and latitude information will send and store to server. Since only publisher location is stored in server memory, publisher location passively enquired by other users. When client holding the screen on map with a certain zoom level, the application can acquire the 4 corner 's latitude and longitude on map screen, these information can be acquire by following code:

```
    double topLeft Latitude=
(mapView.centerCoordinate.latitude)+(mapView.region.span.latitudeDelta)/2;
      double topLeft Longitude=
(mapView.centerCoordinate.latitude)-(mapView.region.span.longitudededeDelta)/2;
      double bottom Right Latitude=
[[[[(mapView.centerCoordinate.latitude)-(mapView.region.span.latitudeDelta)/2;
      double bottom Right Longitude=
(mapView.centerCoordinate.latitude)+(mapView.region.span.longitudededeDelta)/2;
```

Figure 39 Source code of getting 4 corners coordinate by map View

With the 4 corner coordinates, they can send to the server for comparing. Server then can determine which publisher coordinate within the range and return the result set to enquire client.

Video content representation on map

Thanks to the "Annotatior" provided by MapKit framework, a pin or custom MKAnnotations can be display on map indicates users' location. In this project a live video thumbnail picture is expect to use as "Annotatior" for indication of live streaming event. While in the video mode, a blue ball icon represents the position of publisher. User can click on the particular thumbnail to expand event details on map.



Figure 40 Example of using Annotations object indicate location on map.

5.6 Recommendation

Unlike the tradition list view, when millions of video content become available on server, it is not easy to present every video pin on map, also some video contents might not be very interested to some group of users. To handle this situation a recommendation algorithm was implemented to filter the potential video contents to users.

5.6.1 Slope one Algorithm

Slope one is an algorithm for item based-collaborative filtering introduced by Daniel Lemire and Anna Maclachlan. The algorithm is known as the simplest collaborative filtering algorithm which based on the rates of individual item with good prediction accuracy, only users rating is require for calculation rating prediction. The general idea of this algorithm is evaluate the users with similar rating deviation to a particular user, use the deviations to determine what rating for a particular item that user would be given, the prediction usually in numeric form represent as a predicted rating for the item. In this project open-slope one algorithm is implemented to this project. Open-slope one algorithm is an open-source project available in google project developed by Qun, F.C [29]. The algorithm has been modified to fulfill the requirement of this project.

The algorithm consists of two parts; calculate the deviation and compute the predicted score; these two parts of calculation can be denoted as following equation:

Calculate deviation:	To compute an individual item i with respect to item j, and let u be the rated score of the item by a user, X be the entire collection of rating, P be a function to get the users who have rated both item j and I and Total() be the counting function $dev_{j,i} = \sum_{u \in P_{j,i}(X)} \frac{u_j - u_i}{total(P_{j,i}(X))}$
Predict score:	To compute an predicted score, Let $PS(u)_j$ be the predicted score of item j for a particular user u, $T_{i,j}$ be the total number of user who rated both item I and j dev $_{j,i}$ be deviation of item j and i, $\mathrm{PS(u)_j} = \frac{\sum_{i \in S(u)/j} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)/j} \mathrm{c_{i,j}}}$

Example 1

Following example shows the calculation using slope one algorithm to predict a user u rating for item j.

The following table shows the existing video tags rating data rated by user:

	News	**Family**	**Funny**
Diana	2	4	5
Peter	1	2	N/A
Walter	N/A	3.5	5

At first step compute the deviation of each item I with respect to item j using the equation

$$dev_{j,i} = \sum_{u \in P_{j,i}(X)} \frac{u_j - u_i}{total(P_{j,i}(X))}$$

Family tag with respect to news tag deviation = dev = $\frac{2-4}{2} + \frac{1-2}{2} = -1.5$

Funny tag with respect to family tag deviation = dev = $\frac{4-5}{2} + \frac{3.5-5}{2} = -1.25$

Funny tag with respect to News tag deviation = dev = $\frac{2-5}{1} = -3$ …

And so on, compute the rest and filled in the following deviation table:

	News	**Family**	**Funny**
News	0	1.5	1.5
Family	-1.5	0	1.25
Funny	-3	-1.25	0

After getting the deviation data, we can give predict to the particular item, for example what is the predicted score of Funny tag would peter given? Using the above prediction formula:

$$\mathrm{PS(u)_j} = \frac{\sum_{i \in S(u)/j} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)/j} \mathrm{c_{i,j}}} = \frac{\big((1 + (1.5)) * 1\big) + \big((2 + (1.25)) * 2\big)}{3} = \frac{9}{3} = 3$$

So the prediction score of funny tag that peter will rate is 3.

Example 2

what is the predicted score of news tag that walter would give?

$$\mathrm{PS(u)_j} = \frac{\sum_{i \in S(u)/j} (dev_{j,i} + u_i) c_{j,i}}{\sum_{i \in S(u)/j} \mathrm{c_{i,j}}} = \frac{\big((3.5 + (-1.5)) * 2\big) + \big((5 + (-3)) * 2\big)}{4} = \frac{8}{4} = 2$$

So the prediction score of news tag that walter would give is 2.

From the above example, the algorithms always provide a reasonable prediction based on the overall rating deviation. Also the algorithm can base on the individual user rating deviation adjusting the prediction result.

5.6.2 User favorite

id	used_score	user_score	predict_score
79	4.601	4.601	2.017
80	4.507	4.507	2.394
101	4.400	4.400	2.621
90	3.800	3.800	3.712
88	3.800	3.800	3.712
102	3.600	3.600	3.600
99	3.265	NULL	3.265
92	3.265	NULL	3.265
89	3.250	3.250	3.982

```
SELECT r.id,
CASE WHEN (b.score IS NULL)
THEN FORMAT( r.score, 3 )
ELSE FORMAT( b.score, 3 )
END used_score,
FORMAT( b.score, 3 ) AS user_score, FORMAT( r.score, 3 ) AS predict_score
FROM (
SELECT s.item_id2 AS id, SUM( u.rating * s.times - s.rating ) / SUM( s.times ) AS score
FROM oso_slope_one s, oso_user_ratings u
WHERE u.user_id =13
AND s.item_id1 = u.item_id
AND s.item_id2 != u.item_id
GROUP BY s.item_id2
ORDER BY SUM( u.rating * s.times - s.rating ) / SUM( s.times ) DESC
)r LEFT JOIN (
SELECT item_id AS id, rating AS score
FROM oso_user_ratings
WHERE user_id =13
)b ON r.id = b.id
ORDER BY used_score DESC
```

Figure 41 Slope one calculation result (left), predict computation on Mysql (Right)

In the recommendation, system also depends on “user favorites” factor in return the suggestion. Before recommend the result for user, the system did not passively return the result based on the predicted score. Instead the user rated score will be affect the result. From above figure we can see some of the predicted scores are lower then actual rated, this is because the predict score are calculate by deviation, but deviation may not

always reflect what user really wanted. From the result we can see if user had rated on the item, user_score will be used as result. Contrary, if users have not rated, predict_score will be used instead. We can ensure what user like or not like can be reflecting on the result; still the slope one algorithm can provide recommended unfamiliar contents.

5.6.3 Limitation, Performance and complexity

Limitation

The slope one algorithm in this project performs in php web server, due to the live content and VOD content handling in separated server, only VOD content provide recommendation function.

Performance

In this algorithm deviation table is generate by user's rates. Suppose we have million users rating 100,000 tags. There is no need to run the algorithm to the rating dataset for generate deviation dataset. Since we have tracked for how many people rated to the item, updating deviation simply calculated by

((no. of people x currentDeviation) + newDeviation) / Total no. of people.

Still generate deviation is expensive, usually in a large system takes hours to generate deviation during non-peak time. Since the project is consider as non-commercial in this stage, to show the change of result the deviation are generated when a user has rated an item.

Complexity

Suppose we have n tags, with m users and U ratings. For space complexity it requires n (n-1)/2 storage for of deviation of each pair of items. For generate the deviation dataset It takes mn^2 time complexity in calculate all of the deviation. For calculate missing rates for a user who entered j ratings, it take (n-j) j time complexity.

5.6.4 Collecting data for algorithm

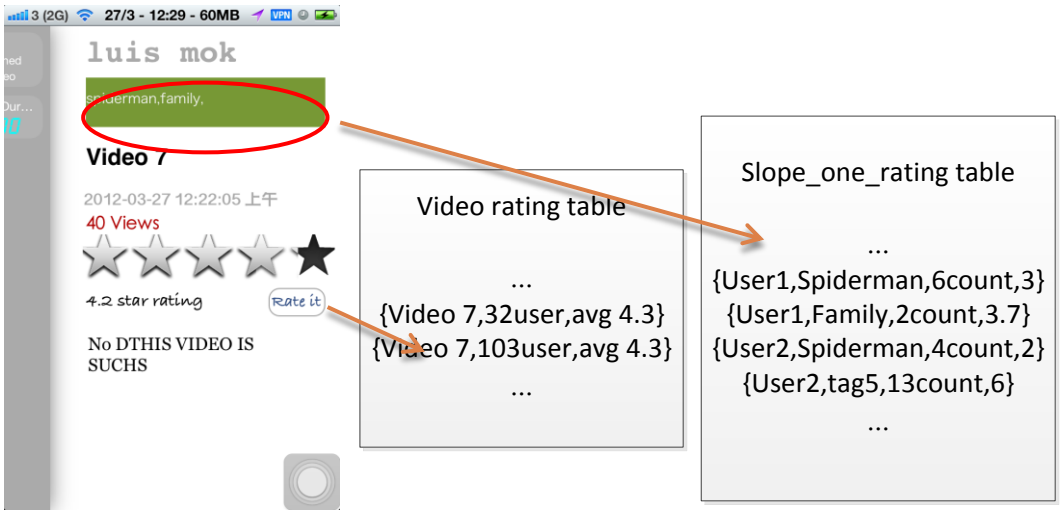


Figure 42 Rated score will use to calculate video average rating and its tags average score with respect to individual user

In the proposed system, while user watching the video, they can always rate a Score for the video. The rated scores will used to calculate the average score for the rated video. This score is only for users references but not used in recommendation algorithm. Two reasons of not using video ratings for recommendation:

- **Highest score videos monopolize the recommendation results**

  Consider when 100 videos have highest rating in Kowloon bay area, when recommendation system can only provide 5 results in one time, the highest rating video

would be always appear on list but the videos with least popularity will have very least chances recommended.

- **Users usually have no rating profiles on new place on map which makes prediction not reliable**

  Video contents are high-density on map but watched video for particular user is low-density in a specific area. For example when user preview the video list on tai po, none of the video is watched hence the recommendation provide result that overall people interested rather than the user interest.

In this system, the algorithm collect the data of the tags rating for video suggestion instead of single video rating, the tag ratings is contributed by the score that user given to the video. For example a Video *VA* was tagged with 3 tags *A, B and* C. When User x rated the video for 4.5 score, the scores will also contribute to calculate the individual average scores for tag A, B and C. The calculated tag ratings are not visible to the users that only for construct the user's deviation profile. The tags rating used to track what kind of video that user like, in other words, the system is expecting to publisher would tag their video with most appropriate tags, the correct use of the tags can enhance the chances of recommend their video to other users.

5.6.5 Recommendation presentation

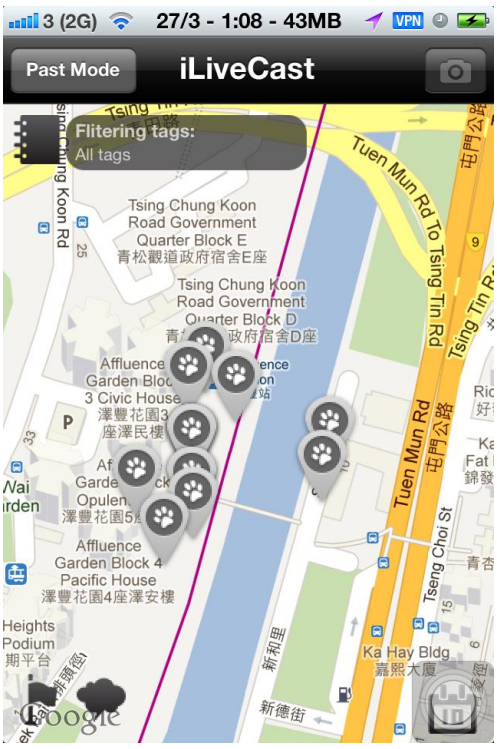


Figure 43 Display existing video content without recommendation



Figure 44 Recommendation proposed video that user might interested

Consider user is searching for video contents within a range in map, the recommendation algorithm should not use the data that beyond map visible range because there are possibilities that the results is beyond watching area makes no result appear on map. To restrict the recommendation that only involving data within the map, following flow diagram shows how the system to perform recommendation:

1. User enable the recommendation function
2. Client side application extract the map visible range and send to server
3. System extract all tags belongs to videos which within the range
4. System perform slopeone algorithm get the recommended tags for user
5. System filter the result by the tags extracted earlier
6. System the videos in database which its coordinates within the range and tagged with the recommended tags
7. System return the recommendation video list to user

5.6.6 Special Cases

There are some special cases when performing the recommendation needed to specially handle:

Rating without login	When a user rated without login, the rating will be contributed to a "overall" account deviation marked as account "0" in database.
Recommendation without login	When user have not logged in to the system, system takes "overall" account as user profile for recommendation. In this way user are likely to retrieve the result that "everyone" would like.
New user without any deviation	A fresh created account usually has no deviation profile related to him. System takes "overall" account to recommend content to new user.
Video have no tags	If a video has not tagged with any video tags, system will not recommended this video. Not tagged video is seen as bad quality content to the system.
Video in map range have no tags recorded user deviation	If there is no deviation can be references (All Video in map have no users rated), the system simply picks the top N result and recommend to the user.

Chapter 6: System Design

In this chapter, several diagrams are presented to describe the system design in the proposed system.

6.1 Provided functions

In this project, functions provided to user are mainly focuses on help user creating a live streaming session and browsing the content on map. Throughout the estimation of the size of the project, following list of function will be provided for user on the mobile device:

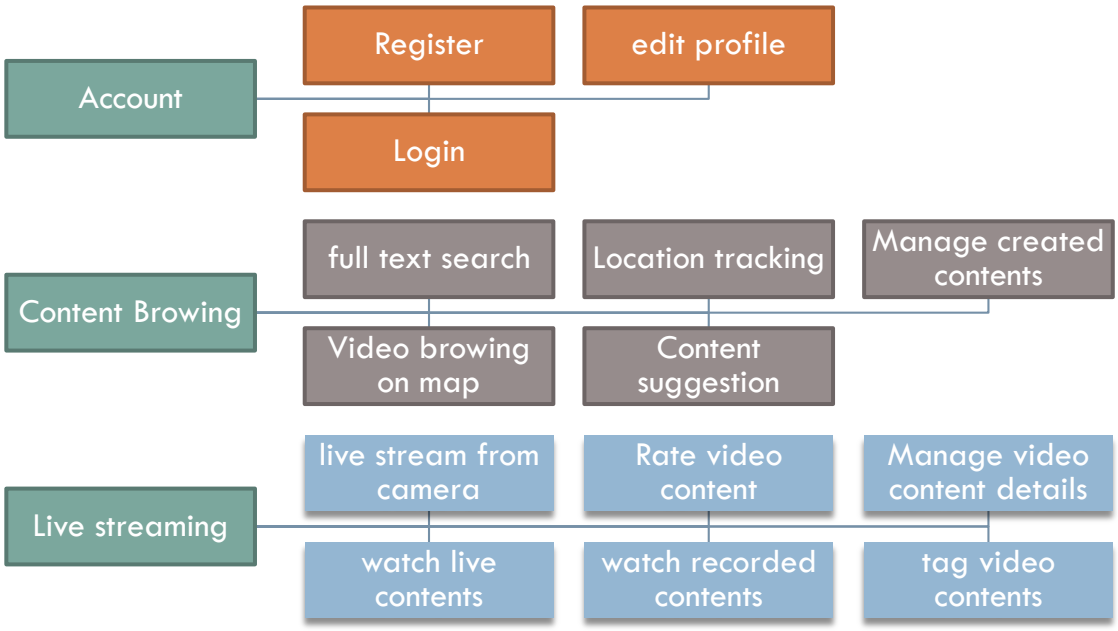


Figure 45 List of intractable functions provided for user in mobile application

6.2Use case Diagram

Use case diagram

On the use case diagram, we can categorize iLivecast application into 3 main categories: Video streaming, Account and Content browsing function. The details of each use case are provided on the next summary table.

There are 3 major actors. User, register user and non-registered-user. In general a non-register user can watch any streaming video only they do not have permission to create a live streaming. A registered user can publish live streaming or manage their own streaming content. A User actor can interact with most of the use cases in the system.

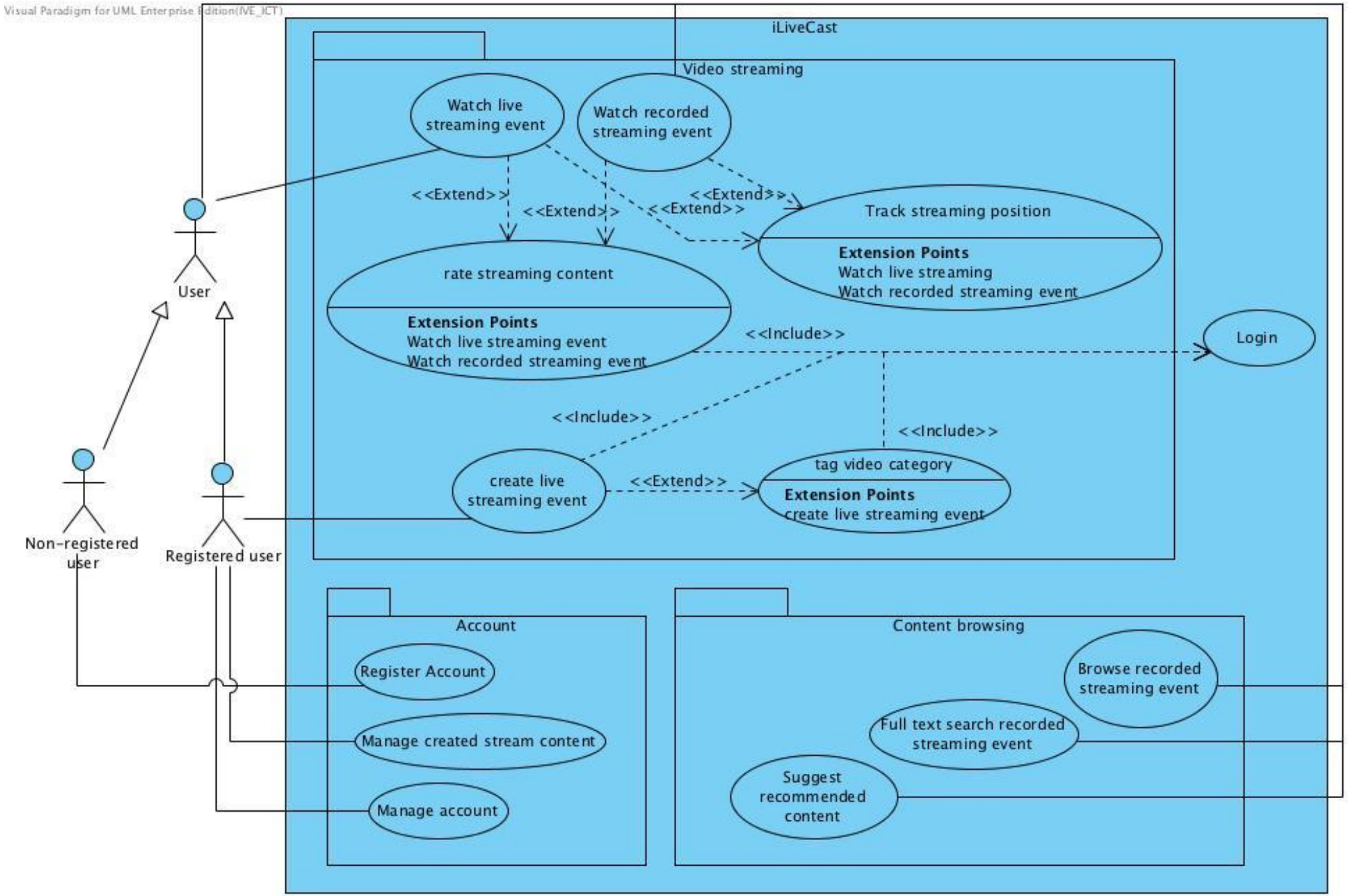


Figure 46 Use case diagram of iLivecast

User case summary table

Design Use Case	Description	Primary Actor
UC1: Register Account	Visitor can register a account on the system.	Non-registered User
UC2: Manage Account	Register user can modify account details of their own user account.	Register User
UC3: Create live streaming event	User can create a event on the map and start the video streaming from their phone camera.	Register User
UC4: Rate streaming content	User can rate a score to the watched video	User
UC5: Track streaming event position	User can track the instant geolocation of the publisher.	User
UC6: tag video category	User can tag the video content to existing or new categories.	Register User
UC7: Watch live streaming event	User can watch the on live event video streaming created by other users.	User
UC8: Watch recorded streaming event	User can search the finished live streaming video footages.	User
UC9: Browse recorded streaming event	User can browse a list of finished live streaming video footages on list.	User
UC10: Full text search recorded streaming event	User can search the video by keyword. tag, description and title of the video will be searched in database.	User
UC11: Suggest recommended content	User can ebable the function let server suggest 5 predicted interesting content to user.	User
UC12: Login	Register user can login to the system with their registered account.	User
UC13: Manage create stream content	User can edit, insert the tag, title or description to the shot video content.	User

6.3Class Diagram

The class diagram is a structure diagram describes the architecture in system's classes. In the systems, classes are categorizing in 3 packages. They are Network, View Controller and Video. The following class diagram only shows the important classes for the operation, list of UI classes are omitted in this diagram.

Network: Classes are responsible to handling socket connection, http connection and JSON encode/decode.

ViewController: Classses are responsible for presentation classes, also coporate with network classes and video classes for video streaming and data interchange.

Video: Classes responsible for video compression from recorded image and audio, cooperate with network classes in sending video stream. Also it handle video playback from receiving video content.

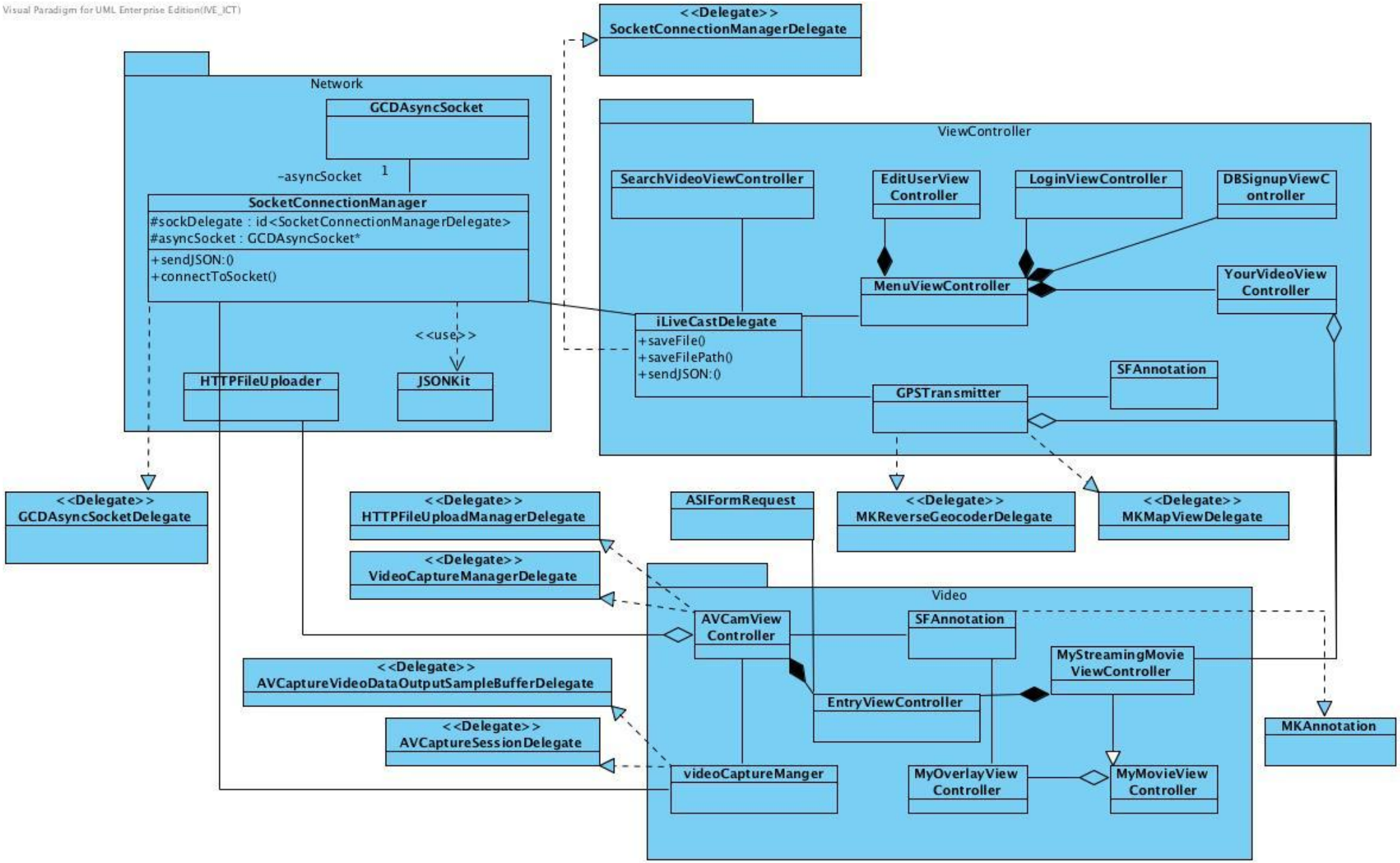


Figure 47 Class Diagram of iLiveCast Application

Some classes are implementing with delegate classes. Delegation is a cocoa framework in Objective-C. A delegation subclass receives notification automatically when delegate class's methods are invoked. Delegation subclass can customize the delegate method.

On the following example, socketDelegate class is part of socket class. Controller 1 and Controller 2 are both implementing SocketDelegate Class, or indirectly inherit some interface in socket class. When Socket methodA is executed, SocketDelegate methodB is invoked. Both controller 1 and 2 will get notification executing their own implementation on methodB(). This is useful that controller can run the customized code before continuing executes the superclass method's body.

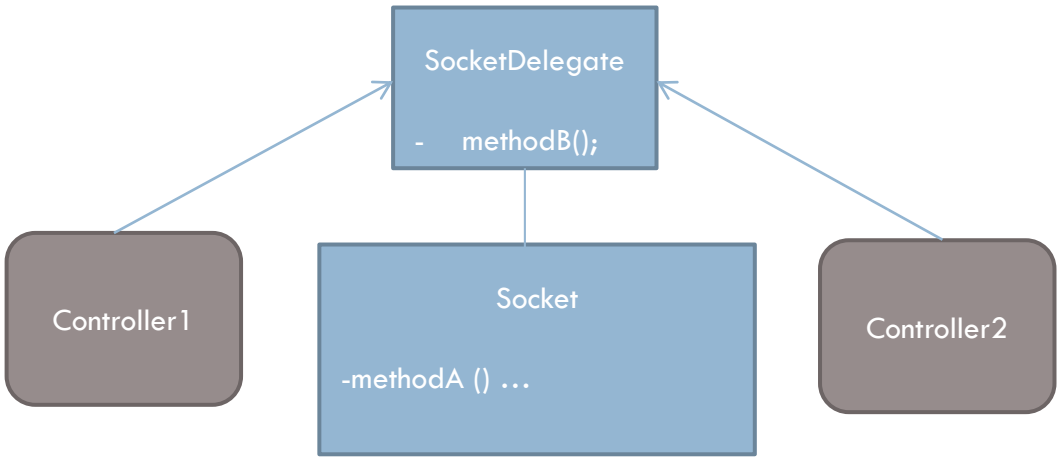


Figure 48 Delegate structure example

6.4Sequence Diagram

Sequence Diagram is one of the Unified Modeling Language (UML) Diagram. It can shows how process flowing in between controller, entities and boundary. On the previous section, the Class Diagram showed classes' relationship and association. In sequence diagram these classes transformed as lifecycle describing how messages

flowing between them in a particular use case.

The following sequence diagram is describing “create live streaming event”, “watch live streaming event”, “browse recorded stream event” and “watch recorded streaming event” use cases. These use cases are extracted from use case diagram which are the important use cases. 3-Tier Sequence diagram can show how classes and actor interaction in completing an event.

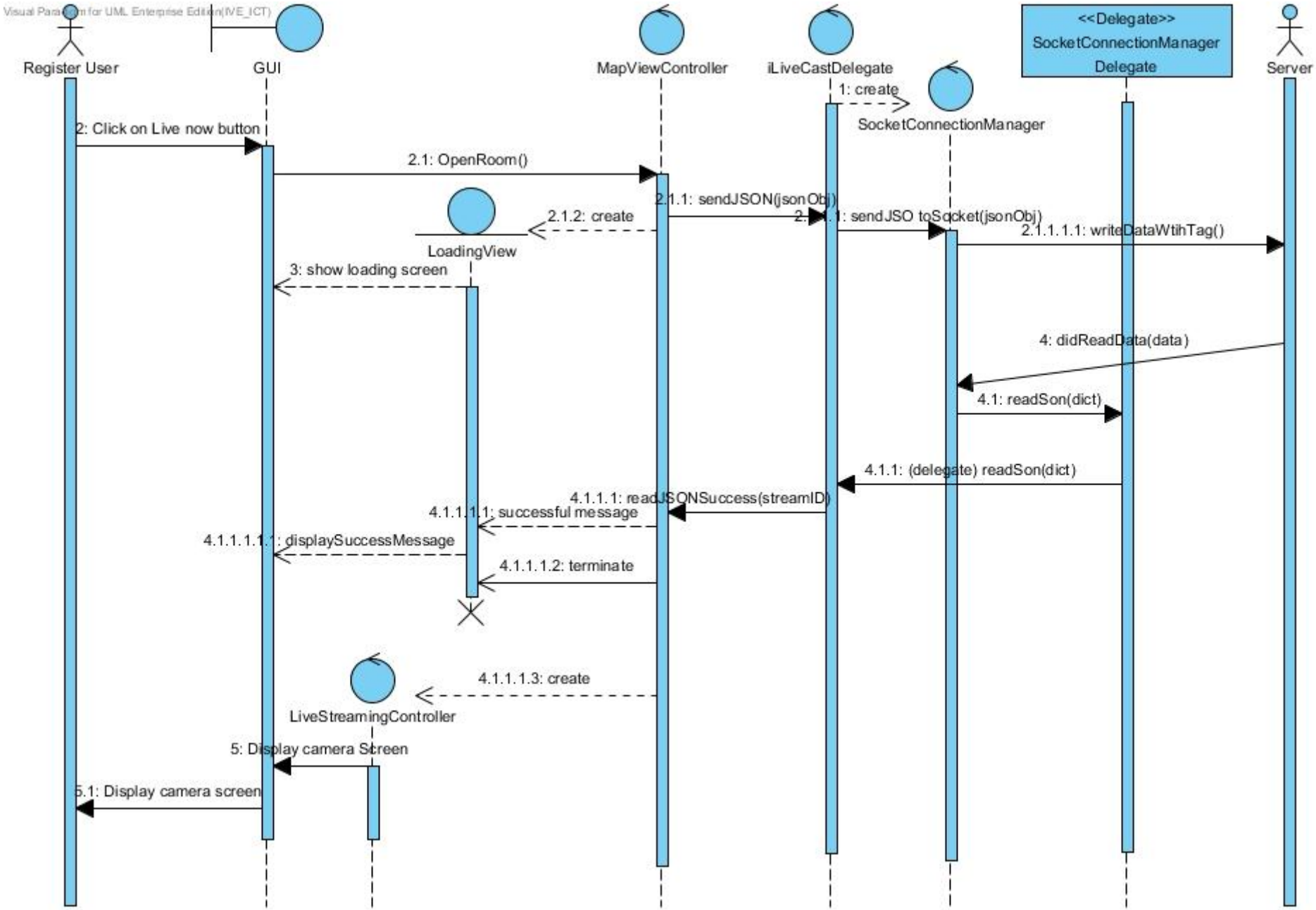


Figure 49 3-Tier Sequences diagram of create live streaming event (create event)

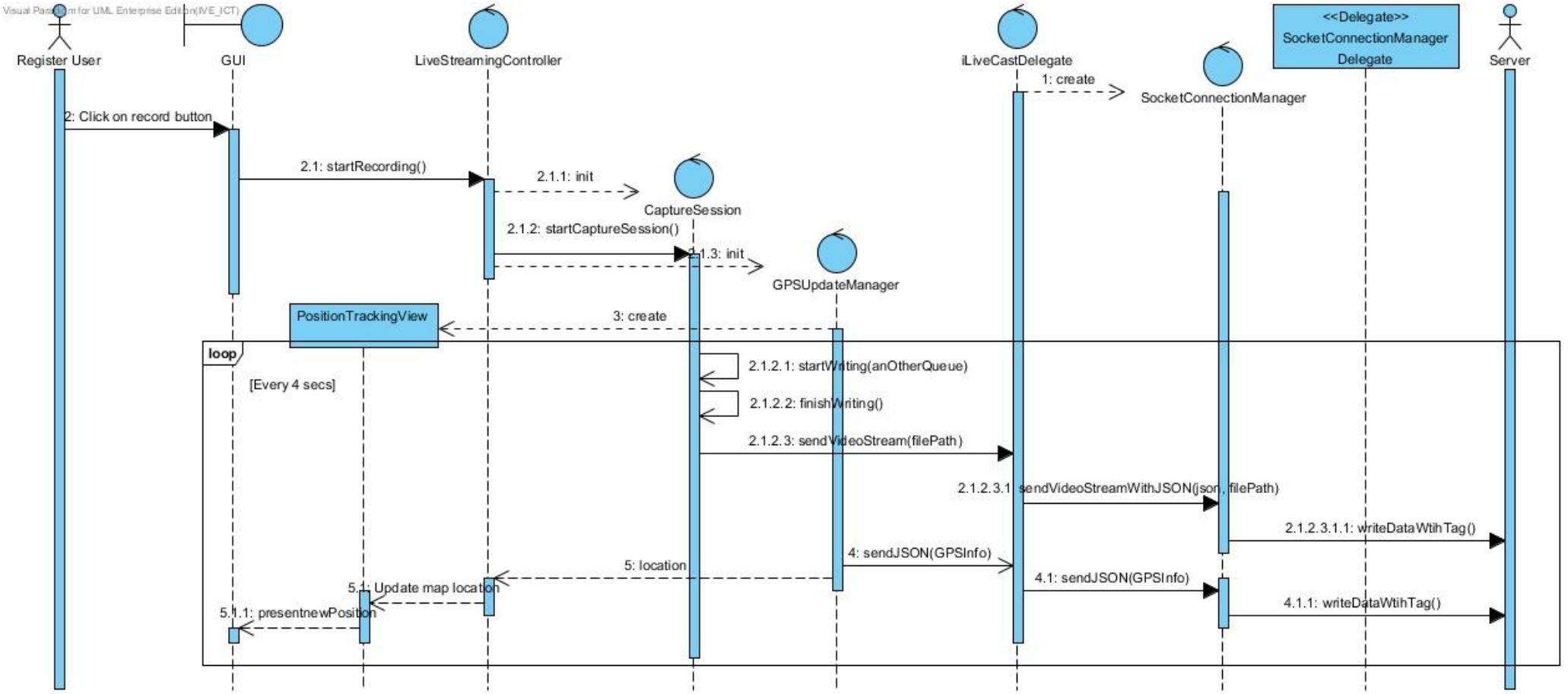


Figure 50 3-Tier Sequence diagram of create live streaming event (start streaming session)

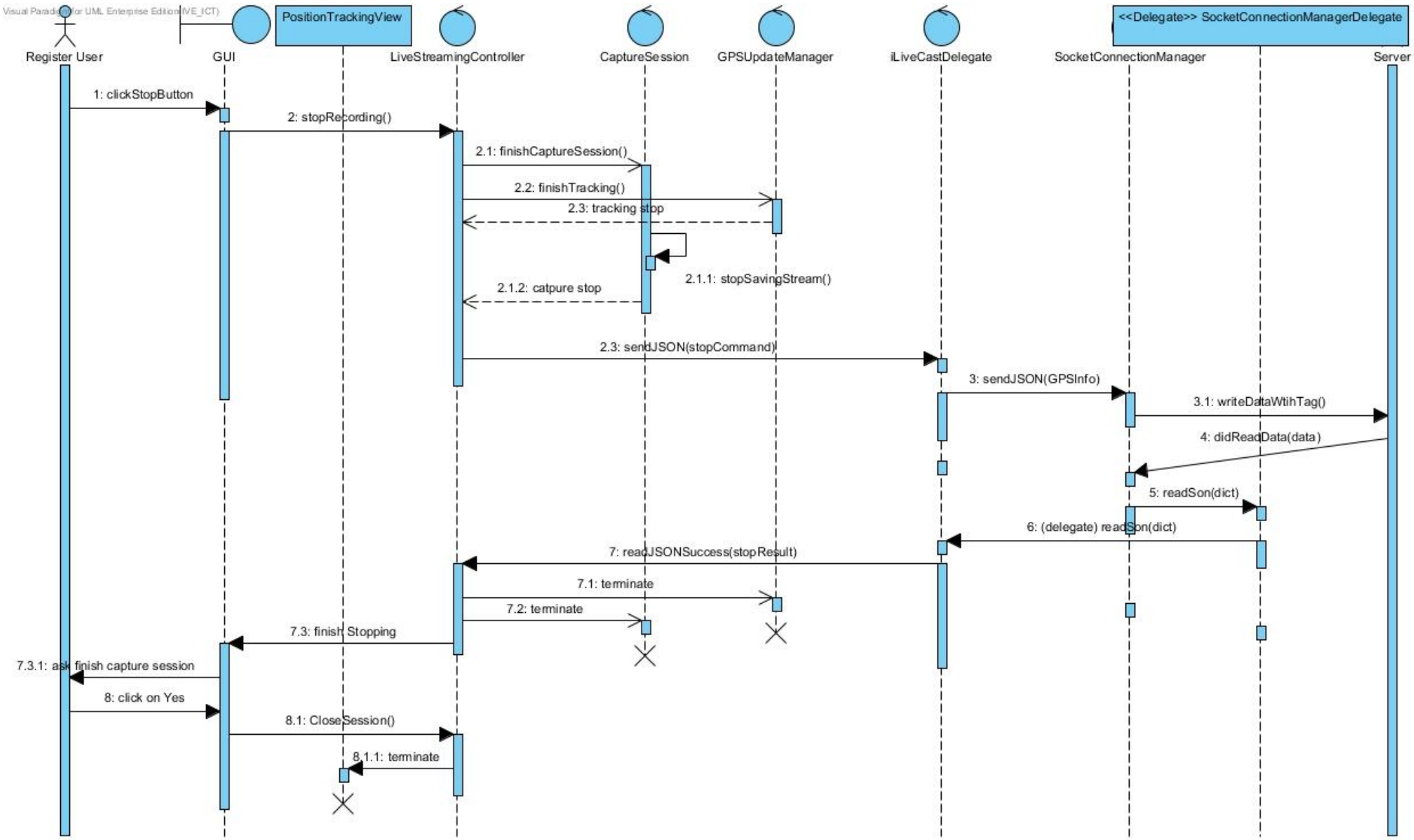


Figure 51 3-Tier Sequence diagram of create live streaming event (stop streaming session)

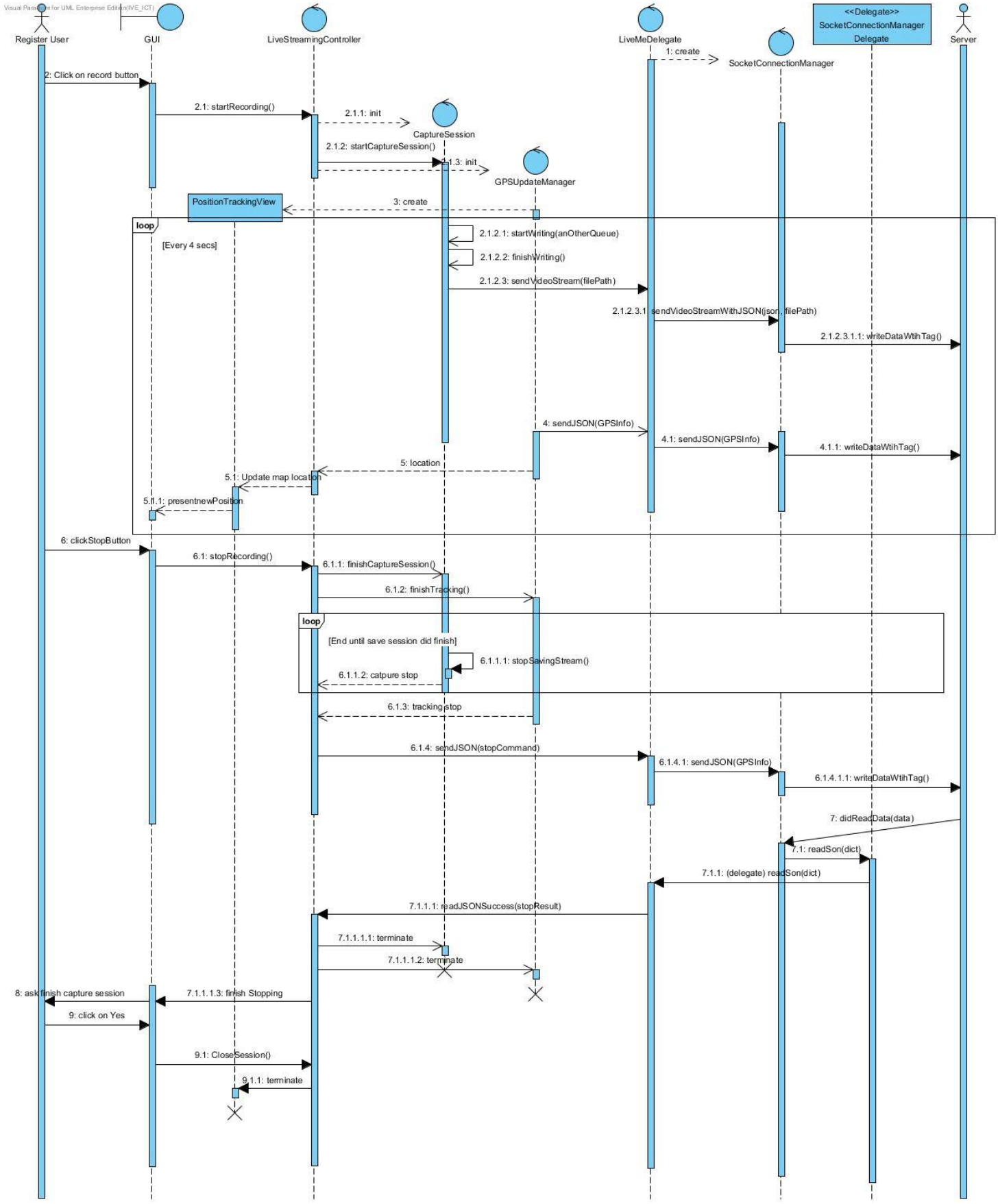


Figure 52 3-Tier Sequence diagram of watch live streaming event

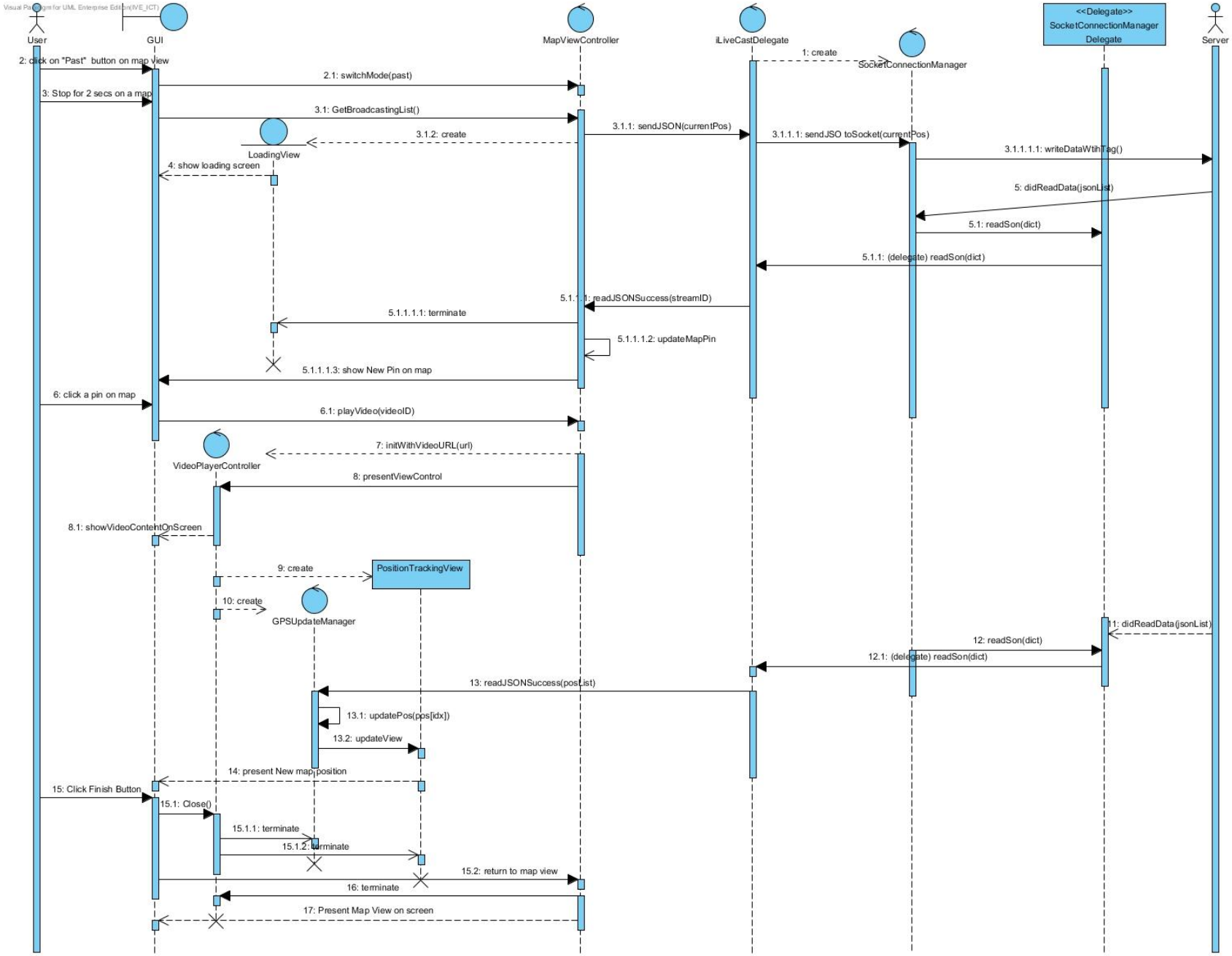


Figure 53 3-Tier Sequence diagram of watch recorded streaming event by map view

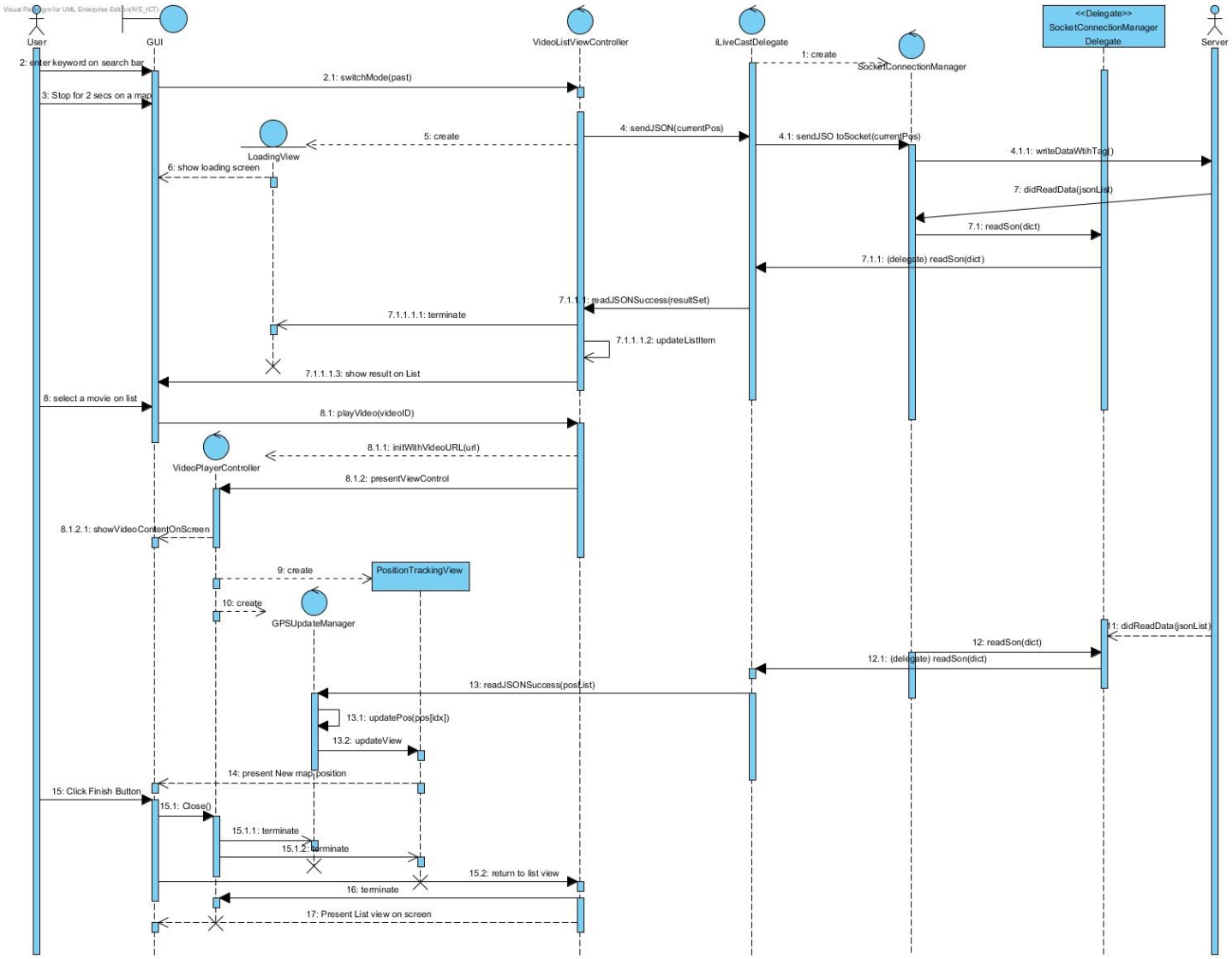


Figure 54 3-Tier sequence diagram of watch recorded streaming event by search list

6.5Database Architecture Design

The following Entity Relationship Diagram reflecting data model in database design. The database architecture is designed to store users' information, video details, coordinate related to video and video categories data.

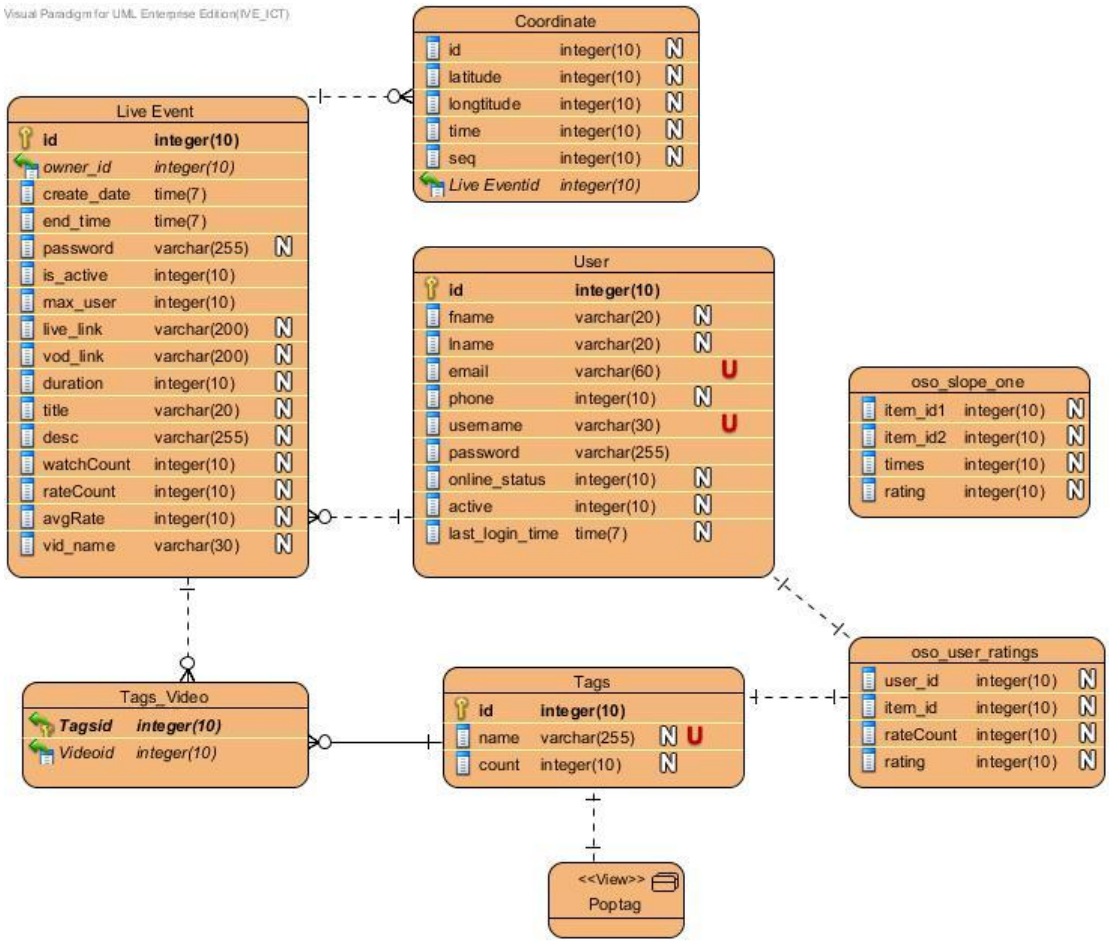


Figure 55 Entity Relationship Diagram of live broadcasting system

Chapter 7: Detailed System Implementation

In chapter 5, the general idea of working theory where introduced. In this chapter, individual part of the detailed implementation is given to answer how it works in the technical aspect.

7.1 Wowza Media Server

Wowza Media Server is used to store and distribute video contents that user publishing or published.

7.1.1 Start/Stop Media Server

In this project, the media server is hosting on Ubuntu VMware instance. Since the media server starts up as services when Ubuntu is started, to manually turn on the server, a startup command on terminal is needed rather than using GUI approaches.

Following steps shows how to start and close the media server:

1. Switch the user to root
2. Navigate the folder to application server folder, Enter "./start.sh", the server will startup automatically.

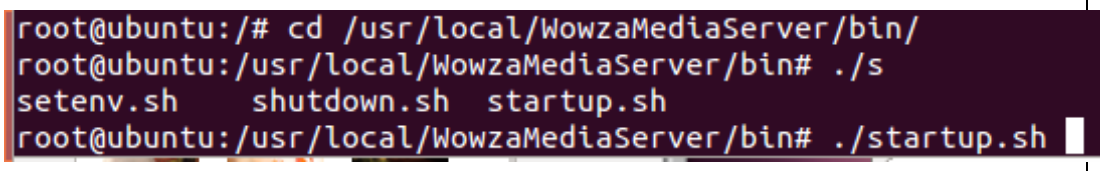


Figure 56 Startup socket application in background process

```
INFO vhost comment _defaultVHost_ Bind attempt ([any]:1935:4)
INFO vhost comment _defaultVHost_ Bind successful ([any]:1935)
INFO vhost comment _defaultVHost_ Bind attempt ([any]:8086:1)
INFO vhost comment _defaultVHost_ Bind successful ([any]:8086)
INFO server comment - Testing - serverinit
INFO server comment - Wowza Media Server is started!
INFO server comment - NewStreamChecker.run: START
```

Figure 57 Startup socket application server in parent process

7.1.2 Stream content to live session

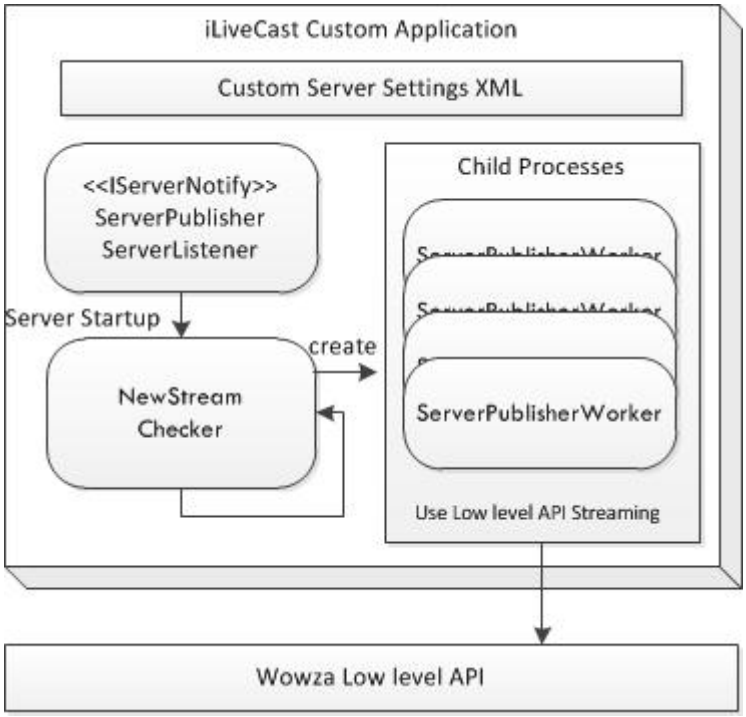


Figure 58 Workflow diagrams of custom application in Media Server

In this project there is no way to retrieve video stream from RTMP, as a result video source are rely on uploaded video files from mobile. While the server is startup, the custom IServerNotify will execute the NewStreamChecker modules. This module checks any new folder existing in live folder. If a new folder appears, a thread process created for a new live stream with the folder name. The folder is actually created by socket application server (Details please refer to section 7.3.3.3), It is like a

manufacture product line that never ask about the sources of raw material.

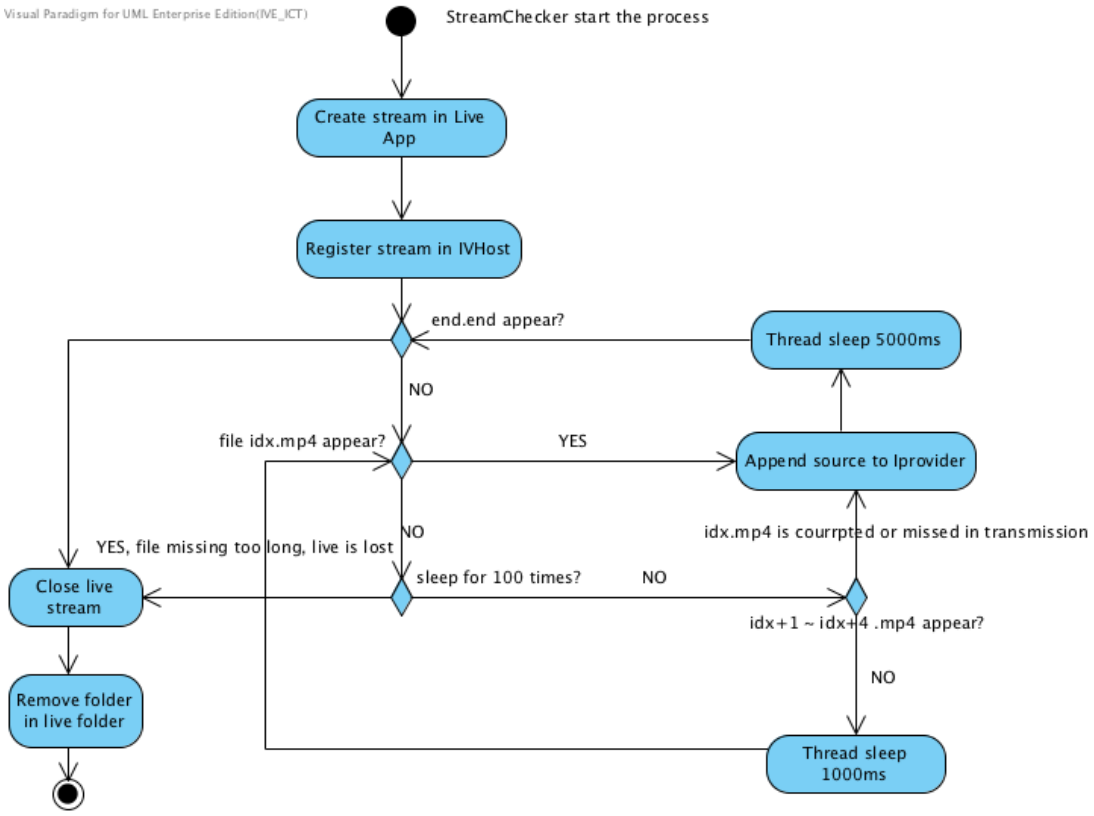


Figure 59 Work flow of appending video sources

Each child process is a class of ServerPublisherWorker that can work independently; each process use the utilize the IVHost, IPublishingProvider and Publisher Class to establish a container and register a corresponding HLS link in Wowza Media Server. The application has config the Cupertino stream method, therefore media server will handle stream and deal with m3u8 chunk playlist when register the content on web server. On the rest of the process, it scans the folder search any new video file is appearing. When new content appear, class PublishingProviderMediaReader append the sources to the stream container. The sources are loaded with sequences by the numeric file index name. When an “end.end” dummy file is appearing on folder (Details please refer to section 7.3.3.3), the ServerPublisherWorker finish the live streaming and

interrupt the process. By the configuration on the application, the played streams are saved as a file automatically to content folder so that user can access the content. Media server itself provided video on demand (VOD) modules that stream contents by HLS to apple devices. Video content can be access by URL without any configuration to individual video contents.

7.2Socket application Server

The socket application is the second major component in this project; it combines with over 3,000 lines of code which handling request from users and manipulate the live video streaming workflow.

7.2.1 Start/Close the server

In this project, the socket application server is hosting on Ubuntu VMware instance. Since the application server starts up as services when Ubuntu is started, to manually turn on the server, a startup command on terminal is needed rather than using GUI approaches.

Following steps shows how to start and close the application server:

1. Switch the user to root
2. Navigate the folder to application server folder
    2.1. Enter "./live_server 43234", the server will running in background.

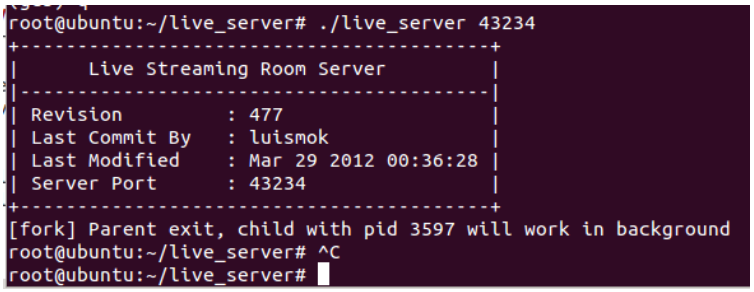


Figure 60 Startup socket application in background process

2.2. In order to see the server response to the client, enter "./live_server_debug 43234" which can keep the application server running in parents process.

```
root@ubuntu:~/live_server# ./live_server_debug 43234
+----------------------------------------+
|      Live Streaming Room Server        |
|----------------------------------------|
| Revision         : 477                 |
| Last Commit By   : luismok             |
| Last Modified    : Mar 29 2012 00:36:28 |
| Server Port      : 43234               |
+----------------------------------------+
[main] Live Streaming Room Server (r477) starting...
chmod: cannot access `*': No such file or directory
rm: cannot remove `*': No such file or directory
[main] libevent 2.0.16-stable initialized with method: epoll
[main] Server is ready on port 43234
```

Figure 61 Startup socket application server in parent process

3. To shutdown sever, if the server is running in background mode, use the grep function to extract process id then kill the process.

```
root@ubuntu:~/live_server# ps aux | grep ./live_server
luismok   3581  0.1  1.7 423188 31824 ?        Sl   00:36   0:01 gedit /home/luismok/live_server/Makefile
root      3660  0.0  0.0  15788   536 pts/0    S    00:51   0:00 ./live_server 43234
root      3672  0.0  0.0  13448   900 pts/0    S+   00:51   0:00 grep --color=auto ./live_server
root@ubuntu:~/live_server# kill 3660
root@ubuntu:~/live_server#
```

Figure 62 Kill the socket server in background process

7.2.2 Handle client connection/disconnection

```
2731    // Prepare socket address structure
2732    struct sockaddr_in servaddr;
2733    servaddr.sin_family = AF_INET;
2734    servaddr.sin_port = htons(port);
2735    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
2736
2737    // Start libevent connection listener
2738    struct evconnlistener *listener = evconnlistener_new_bind(eventBase, connection_accept,
2739    NULL, LEV_OPT_CLOSE_ON_FREE | LEV_OPT_REUSEABLE, 1024, (struct sockaddr *)&servaddr, sizeof(servaddr));
2740
2741    // Check if listener startup correctly
2742    if (listener == NULL)
2743    {
2744        writelog("[main] Error: Cannot initialize server on port %hu\n", port);
2745        fflush(logfd);
2746        fclose(logfd);
2747        exit(1);
2748    }
```

Figure 63 Register connection_accept method for handling network connection establish

On the first startup of the application server, as mentioned on the section 3.2.2, libevent was employed in this project for the server. Libevent is an event loop based notification library, it major advantage can simply invoke event_dispatch() for insert or remove event dynamically without switch the event loop. The libevent help tracking any events invoked and handle it in real time. When the server started up, an event base listener is created for handling any connections.

```
void connection_accept(struct evconnlistener *serv, int sock, struct sockaddr *sa, int sa_len, void *arg)
{
    struct event_base *evbase;
    struct bufferevent *bev;

    evbase = evconnlistener_get_base(serv);

    bev = bufferevent_socket_new(evbase, sock, BEV_OPT_CLOSE_ON_FREE);

    struct sockaddr_in *cliaddr = (struct sockaddr_in *)sa;
    writelog("[conn] Accepted client: %s:%hu\n", (char *)inet_ntoa(cliaddr->sin_addr), ntohs(cliaddr->sin_port));

    room_conn_t *conn = malloc(sizeof(room_conn_t));
    conn->room = NULL;
    conn->this_user = -1;
    conn->bev = bev;
    conn->cliaddr = *cliaddr;
    conn->read_state = READ_LEN;
    conn->read_len = sizeof(unsigned short);
    conn->last_chat = 0;

    // Setup the bufferevent
    bufferevent_enable(bev, EV_READ);
    bufferevent_setcb(bev, connection_read, NULL, connection_event, (void *)conn);
}
```

Create connection object

Method handling event invoked

method handling read data from transmission

Figure 64 Event listener handling connection establish

```
+-----------------------------------------+
|      Live Streaming Room Server         |
|-----------------------------------------|
| Revision          : 477                 |
| Last Commit By    : luismok             |
| Last Modified     : Mar 29 2012 00:36:28 |
| Server Port       : 43234               |
+-----------------------------------------+
[main] Live Streaming Room Server (r477) starting...
chmod: cannot access `*': No such file or directory
rm: cannot remove `*': No such file or directory
[main] libevent 2.0.16-stable initialized with method: epoll
[main] Server is ready on port 43234
[conn] Accepted client: 218.250.60.52:53355
^[
```

Figure 65 Accept socket connection when client connect to server

When a socket connection is connecting, the connecton_accept method is invoked. The server create a connection object retain socket connection saved as a connection object, then create another event which responsible to listen the socket connection.

```
void connection_event(struct bufferevent *bev, short what, void *arg)
{
    if (((what & BEV_EVENT_EOF) > 0) || ((what & BEV_EVENT_ERROR) > 0) || ((what & BEV_EVENT_TIMEOUT) > 0))
    {
        room_conn_t *conn = (room_conn_t *)arg;
        struct sockaddr_in cliaddr = conn->cliaddr;
        writelog("[conn] Client %s:%hu disconnected\n", (char *)inet_ntoa(cliaddr.sin_addr), ntohs(cliaddr.sin_port));

        // Handle disconnect of players
```

```
[main] libevent 2.0.16-stable initialized with method: epoll
[main] Server is ready on port 43234
[conn] Accepted client: 218.250.60.52:53698
[conn] Client 218.250.60.52:53698 disconnected
```

Figure 66 Socket receive notification when connection is lost

When a connection is lost, the registered event is triggered in handling remove connection object retaining in system.

The follow diagram shows the workflow of the event-based server handling connections.

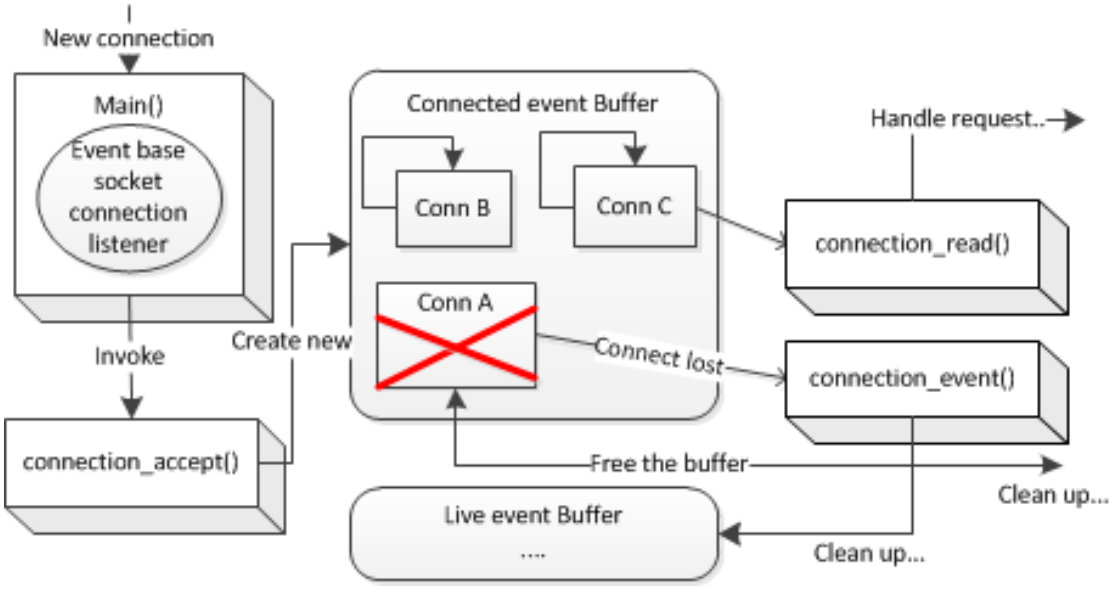


Figure 67 Workflow diagram of socket server handle connections

7.2.3 Handle clients command

7.2.3.1 Decoding message

```
void connection_read(struct bufferevent *bev, void *arg)
{
    struct evbuffer *in = bufferevent_get_input(bev);

    int len = (int)evbuffer_get_length(in);
    unsigned char *str = evbuffer_pullup(in, -1);

    char *client_ip = (char *)inet_ntoa(conn->cliaddr.sin_addr);
```

Figure 68 Code segment: handling receiving data

```
typedef struct room_conn_s {
    room_t *room;
    int this_user;
    struct bufferevent *bev;
    struct sockaddr_in cliaddr;
    READ_STATE read_state;
    unsigned short read_len;
    time_t last_chat;
} room_conn_t;
```

```
typedef enum {
    READ_LEN,
    READ_STRING,
} READ_STATE;
```

Figure 69 Code segment: Conneciton object structure (left), READ_STATE enumeration (right)

In a socket connection data are transfer in serialize, data are deliver byte by byte constantly. A socket connection does not aware the message length, it may cause multiple messages concated or makes socket stunned (incoming byte never stop). In this project, the major message type in network communication between socket server and client is JSON, the json message also embedded a 2 bytes message length identifier on header indicate where is the end of message. When new data packet incoming in socket read stream, socket according to a read state parameter in connection object determining the socket data reading action. If read state is in READ_LEN, convert the packet to number and store the incoming message length. If read state is in READ_STRING, store the incoming data into event buffer until the buffer length equal to stored length. Furthermore, the buffer array is limited in 2bytes size that protect the socket channel would not over occupy by a single message. The following diagram

shows the activities when receive a new packet.

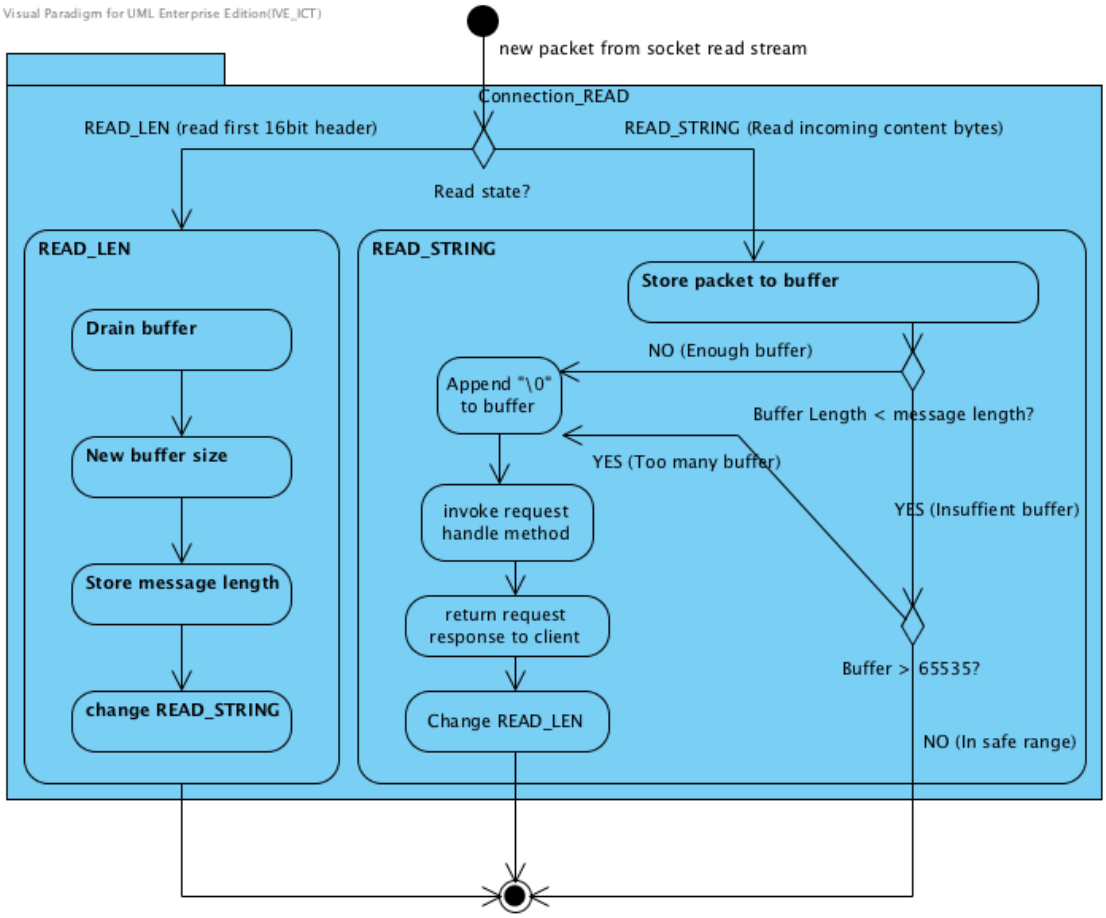


Figure 70 Activity diagram of handling new incoming packet in Connection_Read()

7.2.3.2 Handling JSON request

```
char *handleClientInput(room_conn_t *conn, const char *buff)
{
    cJSON *JSON = cJSON_Parse(buff);
    // Get action
    int action = -1;
    if ((getJSONInt(JSON, "action", &action) != 0) || (action <= 0))
```

Figure 71 Code Segment: Method dispatch the JSON body

```
        case 3: //user send finish video
            //room_t *room = conn->room;

            setNextState(conn->room, STATE_REC_END);
            cJSON_Delete(JSON);

            root =  cJSON_CreateObject();
            cJSON_AddNumberToObject(root, "publisher_status_change_Stop", 0);
            output = cJSON_PrintUnformatted(root);
            cJSON_Delete(root);

            break;

        case 4: //user start the video
```

Figure 72 Code segment: switch body handle JSON message type

Once the message is received, the buffer will pass to corresponding method. The buffer then converts to a cJSON object. In this project, JSON protocol are defined for identify client request, in each JSON message a tag called “action” is used to recognize the type of message. Following table briefly listed the recognized JSON defined in the system.

Action	Description
2	Create Room
3	Publisher finish the video record
4	Publisher start the video record
6	Publisher send location update
8	User request list of room details
9	Room info changed by publisher on live
10	User join room on live
11	User leave room on live
12	Publisher acknowledge file uploaded to server on live

Figure 73 Table of JSON message protocol description

7.2.4 Manage a live sessions

7.2.4.1 Create live session

```
room_t *newRoom(int room_id, int max_users, const char *password, const ch
char *broadcastLink, const char *title, int mode, float latitude, float lo
{
    // Construct the room_t structure
    room_t *room = malloc(sizeof(room_t));
    room->room_id = room_id;
    room->longitude = longitude;
    room->latitude = latitude;
    room->connected_users = malloc(sizeof(char *) * max_users);
    memset(room->connected_users, 0, sizeof(char *) * max_users);
    /*
     ... some codes are omitted ...
    */
    // Add timer event for room logic
    room->timer_event = evtimer_new(eventBase, room_logic, room);
    struct timeval timeout = makeTimeval(1, 0);
    evtimer_add(room->timer_event, &timeout);

    addRoomToList(room);
```

Figure 74 Code segment: create new room object and register evtimer upon client JSON request

```
[createRoom] Going to create the room #1256
[newRoom] +--------------------------------------+
[newRoom] | room_id    = 1256                    |
[newRoom] +--------------------------------------+
[newRoom] | ownername  = Peter pan               |
[newRoom] | max_users  = 99                      |
[newRoom] | password   = C4CA4238A0B923820DCC50... |
[newRoom] | title      = 20120329151752          |
[newRoom] | host       = 172.16.83.130:43234     |
[newRoom] | BCLink     = http://luismok.dyndns.org:1935/live/1256/playlist.m3u8 |
[newRoom] | longitude  = 113.978470              |
[newRoom] | latitude   = 22.402506               |
[newRoom] +--------------------------------------+
[login] Player 1 (peter) Joined room 1256 in pos #0

[createRoom] write file out json: {"type":"create_room","error":"","login":{"pos":0},"room_id":1256,
"room_info":{"roomID":1256,"owner":"Peter pan","players":1,"max_players":99,"link":"http://luismok.d
yndns.org:1935/live/1256/playlist.m3u8","title":"20120329151752","desc":"","tags":"","duration":0,"l
```

Figure 75 Server message shows user create a new live session

The major usage of this socket server is achieving real time streaming session. When a user is going to start a live streaming, a room object is created to maintain the state of a stream session. Also an event base listener is register with an evtimer, the evtimer trigger a room logic method along with room object.

7.2.4.2 Room logic

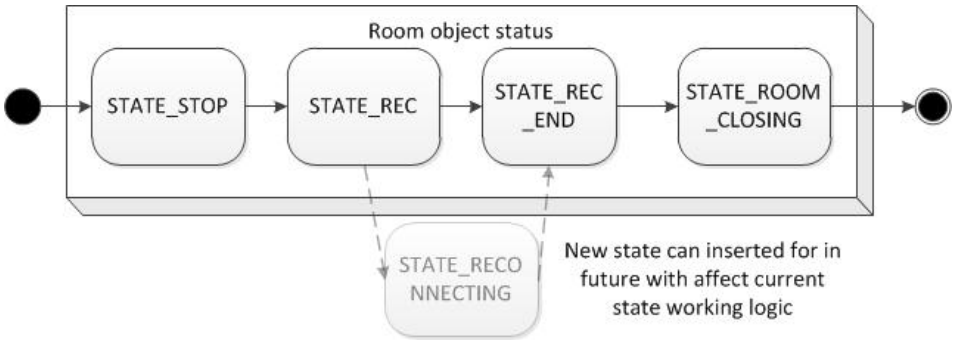


Figure 76 State Diagram of room object

A room object consist of a ROOM_STATE indicate what should or should not be done in room logic methods, how to handle each JSON request also rely on the room status. The ROOM_STATE logic provides the flexibility in insert new state without modifies much logic in existing state.

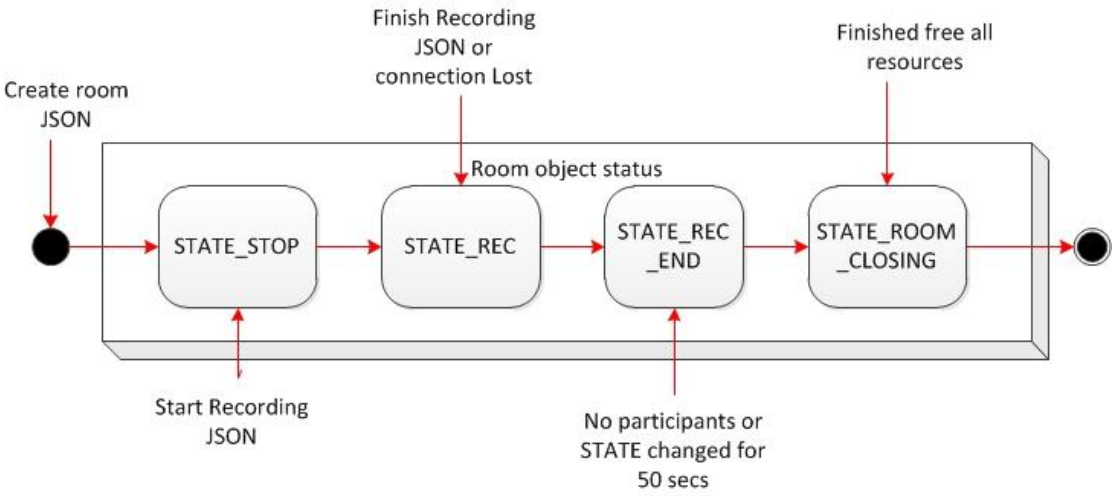


Figure 77 State Diagram shows action that cause ROOM_STATE change

Addition the ROOM_STATE is important in the room logic method triggered by evtimer, it enabled the ability of the room can work automatically and independently according to the room status. ROOM_STATE change according to the user requested but sometimes it can change by system. For example, from STATE_REC to

STATE_REC_CLOSING can trigger by user request on finish live streaming, but it also can change by system if publisher connection lost was detected.

7.2.4.3 Live broadcasting lifecycle

As discussed in the early chapters, due to the problem of RTMP protocol in mobile device, a live streaming is actually done in HTTP. While the media server has no function in retrieve video files but rely on web server, unfortunately the web server cannot manipulate the file system. Consequently, the room logic acts as an adapter in cooperate with both servers.

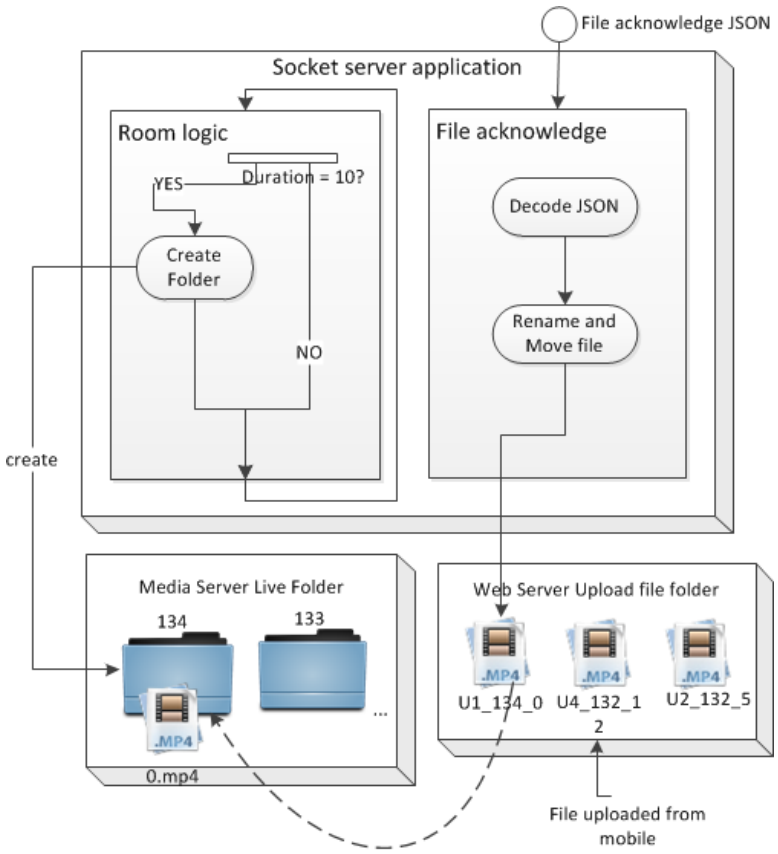


Figure 78 Work flow diagram of live streaming in handle video upload from mobile in STATE_REC

In STATE_REC, According to HLS streaming definition, each video file recorded

from the mobile is 8 seconds long; therefore it is presumed no video file uploaded to webserver on the first 10 seconds. While the STATE_REC changed for 10 seconds, room logic creates a folder named by the room ID in media server live folder. The media server has been programmed create a new live stream when a new folder is appear. When a video file finished uploading to web server, the file name is delivered to socket server through JSON, each file name is unique. According to the JSON message the uploaded video file will be moved and renamed from web server folder to live folder by system call in room logic. In this way, the media server can broadcast the files according to the index. (For the working theory in media server please refer to section 7.1)

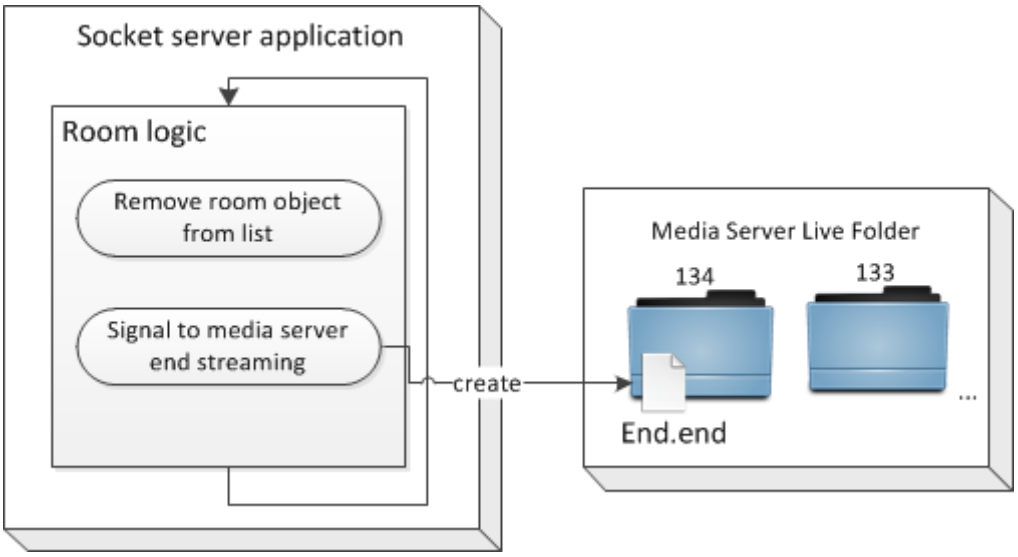


Figure 79 Work flow diagram of live streaming in finish streaming session in STATE_RED_END

While the user is disconnected or request for finish live session, the room logic change the ROOM_STATE to STATE_REC_END. In STATE_REC_END, room logic creates a thumb file called “end.end” in the live folder; it is a signal for media server stops the video streaming. It also removes the room object from list to preventing addition user

joining the room.

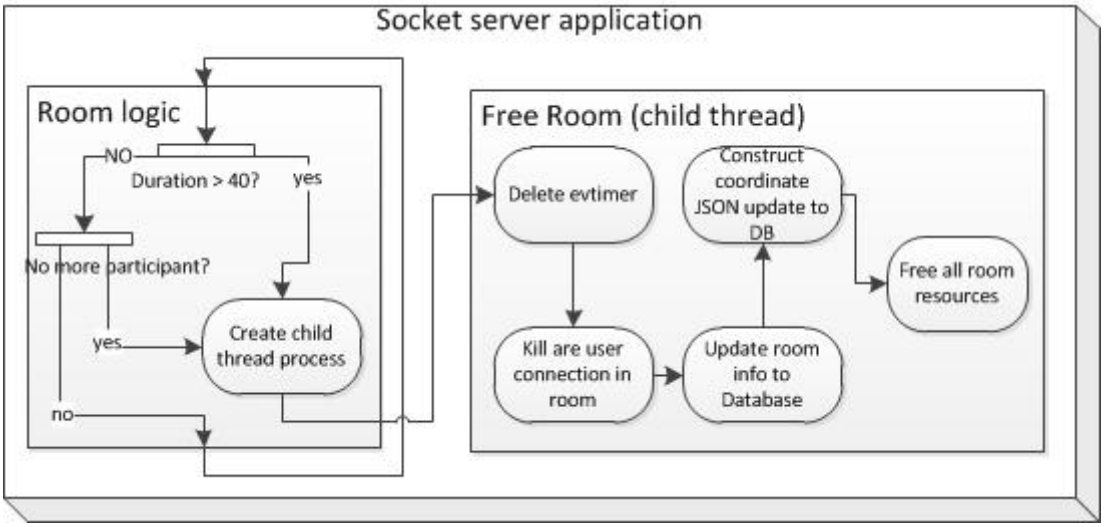


Figure 80 Work flow diagram in clean up and update room info in STATE_REC_CLOSING

The room state change to STATE_REC_CLOSING after STATE_REC_END finished in room logic. In the STATE_REC_CLOSING, consider HLS is a cached streaming in browser; it is hard to predict each users delay in watching the video and odd if force closing the streaming session. In the experiment of HLS streaming delay, the range is generally between 20-40 seconds. With this delay value STATE_REC_CLOSING give 40 seconds as a buffer timing, for those user who still watching the live, it is assumed the playback will be finished with 40 seconds and they should leave the session. Once the 40 seconds reached, the server forces all users quitting the room and begins the clean up process. During the clean up process, memory resources are being free, the video details and coordinate will be updated to database for user accessing later.

```
[conn] Client 218.250.60.52:52398 disconnected
[conn] Player disconnect: 1 (peter)
[room] changeLiveState: room 1256 change from STATE_STOP(0) to STATE_ROOM_CLOSING(8)
[STATE_ROOM_CLOSING] calling freeRoom process 1256
[freeRoom] Start the free room process : 1256
[freeRoom] Cleared out the room  : 1256
[freeRoom] Updated Coordinates info to PHP : 1256
[freeRoom] Updated close room info to PHP : 1256
[freeRoom] free room process completed
```

Figure 81 Server message shows room state change due to publisher disconnected and begin clear up process for room

7.2.5 Broadcasting live status

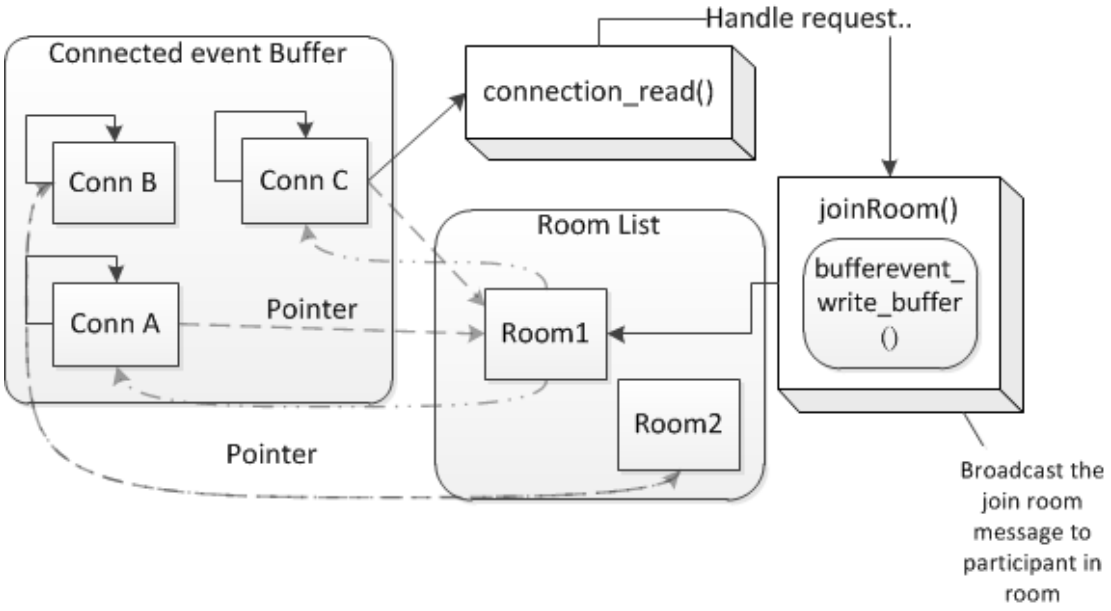


Figure 82 Message from connection object deliver to particular room object for broadcasting

One of the characteristics in socket server is able to provide instant message update to connected users. In the system, some instant messages are provide to participants of the live streaming such as user joined, session closing or session details updated. How can the message deliver to particular user? When a user is connected to a particular room, each connection have a pointer references the connected room object, also the room object have a array pointer references all connected connection object, their user name and ID. On the above figures, when a connection_read() trigger

specific method by “conn C”, such as joinRoom(), the method able to extract the list of connected connection object from pointers in room. From the bufferevent_write_buffer(), the prepared JSON can be write on socket to the participant in room only, without deliver to wrong user.

```
void boardcast_uesr_join_room(room_t *room, int pos) {
    cJSON *root = cJSON_CreateObject();
    cJSON_AddItemToObject(root, "room_join", buildJoinedRoomJSON(room, pos));
    char *output = cJSON_PrintUnformatted(root);
    cJSON_Delete(root);

    int i;
    for (i = 0; i < room->max_users; i++)
    {
        if (room->users_conn[i] != NULL)
        {
            struct evbuffer *out = evbuffer_new();
            evbuffer_add_agt_string(out, output);
            bufferevent_write_buffer(room->users_conn[i]->bev, out);
            evbuffer_free(out);
        }
    }

    free(output);

}
```

Figure 83 Broadcasting join room JSON to participant

7.3IPhone application

7.3.1 Network Connections

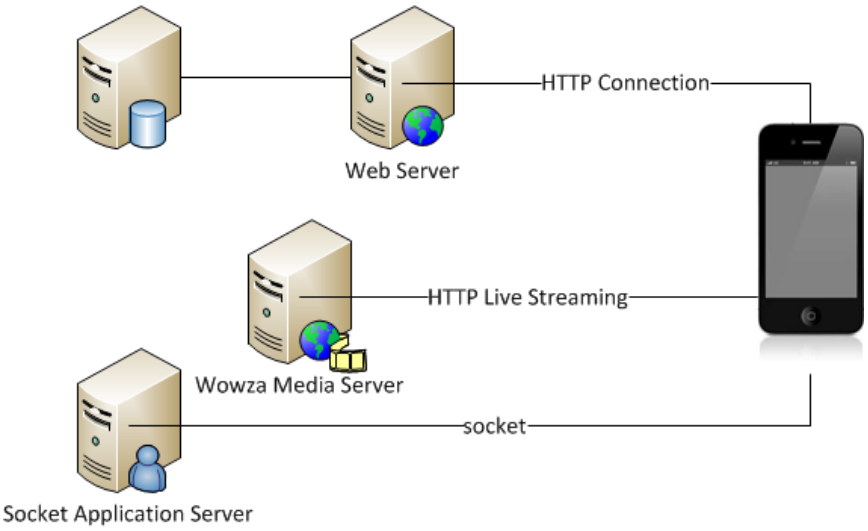


Figure 84 Network Connection between server and client

There are 3 connections that would establish between iPhone to Server. HTTP connection to web server, Socket connection to socket server and HLS connection to Media Server. Each connection serves different purpose that can divide the workload to servers.

There are two modes in the application, live mode and past mode.

In live mode, contents shown on map are currently broadcasting in real time. These content such as video list, coordinates and instant message are query from socket application server.

In Past mode, contents shown on map are broadcasted in the past. Video list, These content such as coordinates and video details are query from web server and database.

There are some differences in handling data like coordinate in this both mode,

these will be discussed in the coming sub-section.

Socket

```
@class GCDAsyncSocket;

@protocol SocketConnectionManagerDelegate <NSObject>
@required
- (int) didreceiveJSON: (NSDictionary *) jsonDictionary;
@end

@interface SocketConnectionManager : NSObject
{
    id <SocketConnectionManagerDelegate> sockDelegate;
    GCDAsyncSocket *asyncSocket;

}

@property (retain) id sockDelegate;

- (void) sendJSON:(NSDictionary *) jsonDictionary ;
- (void) connectToSocket ;

@end
```

Figure 85 Code segment: SocketConnectionManager header structure

In iphone SDK, Apple provides CFNetwork framework handling network connectivity. Unfortunately, CFNetwork for socket connection is in runloop based. It doesn't provide good performances in several aspects. To deal with this problem, this project extended the CocoaAsyncSocket library for socket connection. It is an asynchronous socket libraries and GCD (Grand Central Dispatch) based. The library support either TCP (Transmission Control Protocol) or UDP (User datagram Protocol) sockets. Also it provides SSL (Socket Sockets Layer) and TLS (Transport Layer Security), although they are not necessary to this project. TCP is major use in the project because TCP provide stable and reliable data transmission.

```
- (void) connectToSocket {
    DDLogInfo(@"%@", THIS_METHOD);

    dispatch_queue_t mainQueue = dispatch_get_main_queue();
    asyncSocket = [[GCDAsyncSocket alloc] initWithDelegate:self delegateQueue:mainQueue];
    NSString *host = HOST;
    uint16_t port = PORT;
    NSError *error = nil;
    [asyncSocket connectToHost:host onPort:port error:&error];

    if (error != nil)
    {
        DDLogError(@"Error connecting: %@", error);
    }else {
        DDLogError(@"Connected to \"%@\" on port %hu ... OK!!!!", host, port);
    }
```

Figure 86 Code segment: Connect to socket server

SocketConnectionManager class is responsible all of the operation related to socket connection, such as error handling, data send/receive and connect/reconnect. Connect to socket server is fairly easy than Class CFNetwork provided in Apple Core Foundation Framework. Simply initialize the GCDAsyncSocket object with GCD dispatching queue, and then connect to server by inputted host and port.

```
- (void) sendJSON:(NSDictionary *) jsonDictionary {

    NSMutableData *data = [[NSMutableData alloc] initWithCapacity: 65535];

    NSString *jsonString = [jsonDictionary JSONString];

    unsigned char bytes[2];
    unsigned long n = [jsonString length];

    bytes[0] = (n >> 8) & 0xFF;
    bytes[1] = n & 0xFF;
    DDLogInfo(@"Send out message to \"%@\" on port %hu ... \"%@\"", HOST, PORT, jsonString);

    [data appendBytes: &bytes length:2];
    [data appendData:[jsonString  dataUsingEncoding:NSISOLatin1StringEncoding]];


    [asyncSocket writeData:data withTimeout:-1 tag:0];
    [asyncSocket readDataToData:[@"\n" dataUsingEncoding:NSASCIIStringEncoding] withTimeout:-1 tag:0];
}
```

Figure 87 Code Segment: Append message size header to JSON message before send out to socket

JSON is the only data type that used the socket communication channel between socket server and iPhone. Before a JSON transform to JSON String, the message are actually stored in NSDictionary Class. NSDictionary is a mapping table class providing in iOS library, each data in NSDictionary is paired with a String key. NSDictionary itself does not native support JSON, as a result a static library called JSONKit is used. It is an

extension library for NSDictionary under BSD license. It used to convert NSDictionary to a JSON string or reverse. It is convenient since the structure of JSON is similar to NSDictionary. When a message going to deliver, all the send out socket data will go through the sendJSON method in SocketConnectionManager class for converting to JSON String. Also, refer to the message protocol in socket server, each socket message header should embed 2 byte for recognize message size. In this method, the JSON string length will convert to 16bits value by bitwise operation then divided equally in 8 bits store into two bytes header.

```
- (void)socket:(GCDAsyncSocket *)sock didReadData:(NSData *)data withTag:(long)tag
{
    NSString *socketMessage = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
    NSDictionary *json = [socketMessage objectFromJSONString];
        if (json == NULL) {
        NSLog(@"\nThe JSON convert in SocketConnectionManager is failed!");
        NSLog(@"\nGet json = %@", socketMessage);
    }else {
        [[self sockDelegate] didreceiveJSON:json];
    }
    [socketMessage release];
     [asyncSocket readDataToData:[@"\n" dataUsingEncoding:NSASCIIStringEncoding] withTimeout:-1 tag:0];
}
```

Figure 88 Code segment: Receive data from socket

```
- (void) receivedSocketData: (NSNotification *)notification  {
    NSDictionary *jsonDictionary = [notification userInfo];
    NSString *msgType = [jsonDictionary objectForKey:@"type"] ;
    NSString *error = [jsonDictionary objectForKey:@"error"] ;


    NSLog([NSString stringWithFormat:@"[GPSTransmitter] msgType is %@", msgType]);
    bool equal = NO;
    if ([msgType isEqualToString:@"create_room"]) {
```

Figure 89 Code segment: delegate method in GPSTransmitter handle received JSON from socket

When new data received to the socket, GCDAsyncSocket called he delegated method informing data is ready on socket. Thanks to the NSNotification, socket can notice and deliver the decoded message to corresponding classes. GCDAsyncSocket knows how much available data on the socket in transmission, therefore GCDAsyncSocket can allocate fit size of buffer for incoming data. However, sometimes messages send from socket server are concatenated, which cause JSON cannot convert

successfully. Address this matter, each message send from socket server appended a '\n' on the end of JSON message, in JSON definition '\n' is not allowed, which makes it can be the indicator of the end of JSON message. The readDataToData() tells GCDAsyncSocket listen for socket data, also specified a '\n' so that when the indicator is detected, GCDAsyncSocket will separate the data as two set of message.

7.3.2 Positioning and map

```
- (void)locationManager:(CLLocationManager *)manager didUpdateToLocation:(CLLocation *)newLocation fromLocation:
    (CLLocation *)oldLocation {
    self.nowPosition = newLocation;
    MKCoordinateRegion theRegion;

    //set region center
    CLLocationCoordinate2D theCenter;
    theCenter.latitude = newLocation.coordinate.latitude;
    theCenter.longitude = newLocation.coordinate.longitude;
    myRegion.center=theCenter;
    if (!followme) {
        myRegion.span = mapView.region.span;
        [mapView setRegion:myRegion animated:YES];
    }
}
```

Figure 90 Code Segment: delegate function of location update in main map

CLocationManager is a class provided in CoreLocation framework for tracking device location by GPS. It invokes "didUpdateToLocation" by "CLLocationManagerDelegate" when new position is detected. We can use this method to retrieve the latest location details. When user is on the main map, the CLLocation can keep on focus the region where the device is so that user can explore the video content near him. Also when a user is publishing a video, CLLocation determine any new location is update which help the program determine when should send the location update to server.

```
//set region center
CLLocationCoordinate2D theCenter;
theCenter.latitude = [(Coordinate *)[coordinates objectAtIndex:0] latitude];
theCenter.longitude = [(Coordinate *)[coordinates objectAtIndex:0] longitude];
theRegion.center=theCenter;

//set zoom level
MKCoordinateSpan theSpan;
theSpan.latitudeDelta = 0.0009;
theSpan.longitudeDelta = 0.0009;
theRegion.span = theSpan;

//set scroll and zoom action
```

Figure 91 Code segment: centralize the map to the specific coordinate with zoom level

Map Kit framework is used to provide google map interface. It provides MKMapView class for manipulating the map interface. When we retrieve the user location, we can portal to the specific area along with zoom level just like Google map by using region property in MKMapView. This is frequently used when tracking currently position or publisher location in video.



Live Pin History Pin Recommended Pin Publisher Location Pin

Figure 92 List of Pins used to point out video location in map

Using a pin or annotation is commonly used to point out an accurate position in google map. Map Kit framework provided MKAnnotation for customize a pin for appearances, title or description. In this project, SFAnnotation object used to customize the annotation appear in map, also it can carry video content details retrieve from server.

```
            if (!exist) {
                [self.mapView removeAnnotation:oldA];
                [discardedOldItems addObject:oldA];
            }
        }
    }

    //clean up the items
    [mapAnnotations removeObjectsInArray:discardedOldItems];
    [newAnnos removeObjectsInArray:discardedNewItems];

    //add all new anno to map.
    for (SFAnnotation *newA in newAnnos) {
        [newA setPinmode:(slope ? 2: 1)];
        [self.mapView addAnnotation:newA];
        [mapAnnotations addObject:newA];
    }
```

Figure 93 Code segment: add annotation on map

To show the videos on map, for example, retrieve what live streaming is broadcasting on map, server return a JSON contains list of available content. The delegate method decodes the message and transforms video details into SFAnnotation objects. Each SFAnnotation will be added to map by addAnnotation() in MKMapView. When new annotation added, MKMapViewDelegate called viewForAnnotation delegate function for drawing new annotation on map. The delegation function can

identify the annotation class to configure the appearances and behavior of the annotation.

```
- (MKAnnotationView *)mapView:(MKMapView *)theMapView viewForAnnotation:(id <MKAnno
    if ([annotation isKindOfClass:[MKUserLocation class]])
        return nil;
    if ([annotation isKindOfClass:[SFAnnotation class]]) {
        // try to dequeue an existing pin view first
        NSString* travellerAnnotationIdentifier = [NSString stringWithFormat:@"%@%@
            ], (SFAnnotation *)[annotation pinmode]];
        MKPinAnnotationView* pinView = (MKPinAnnotationView *)
        [mapView dequeueReusableAnnotationViewWithIdentifier:travellerAnnotationIde
```

Figure 94 Code segment: viewForAnnotation configure differently in user mark and SFAnnotation



Figure 95 Difference response when touch on the annotation between SFAnnotation(Left) and User Location(Right)

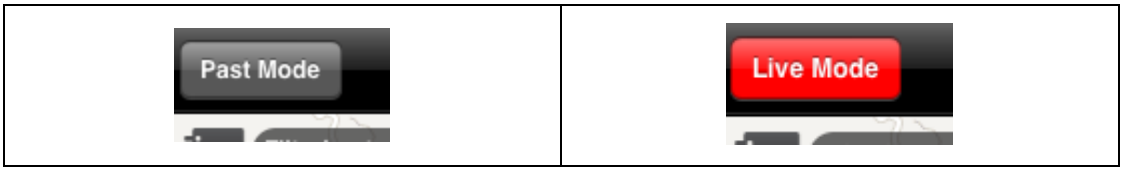


Figure 96 Two different mode switcher

Position tracking behaviors are slightly different in live mode and past mode.

In live mode, coordinates are push to client when there is an update on publisher device.

In past mode, entire set of coordinate regard to the video were located in database and extracted from webserver to the client on video startup. Each coordinate detail related to specific playback time that makes the location change automatically when the video is playing.

7.3.3 Live streaming

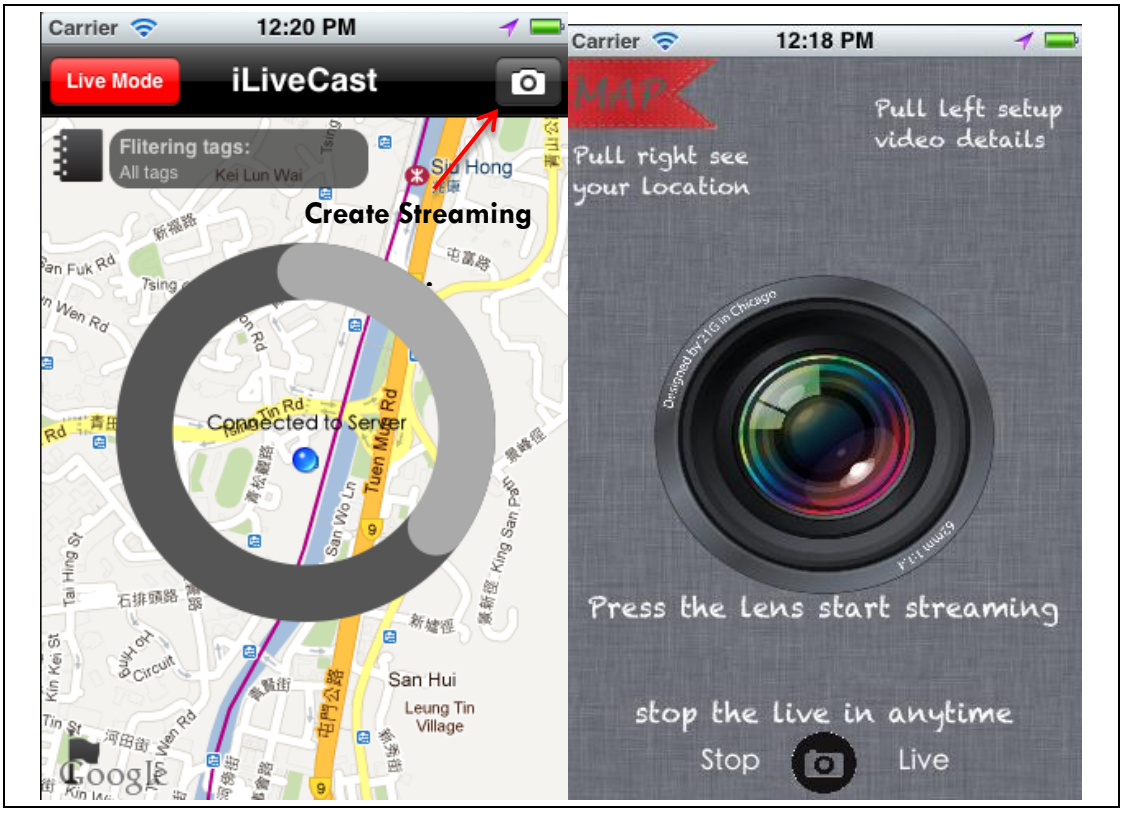


Figure 97 Creating streaming session (Left), Waiting user start recording (Right)

When user going to create a streaming session, a JSON creates room message is sent to socket server. User id, current latitude and longitude etc are embedded in the JSON for register a session in socket server.

Video capture and streaming

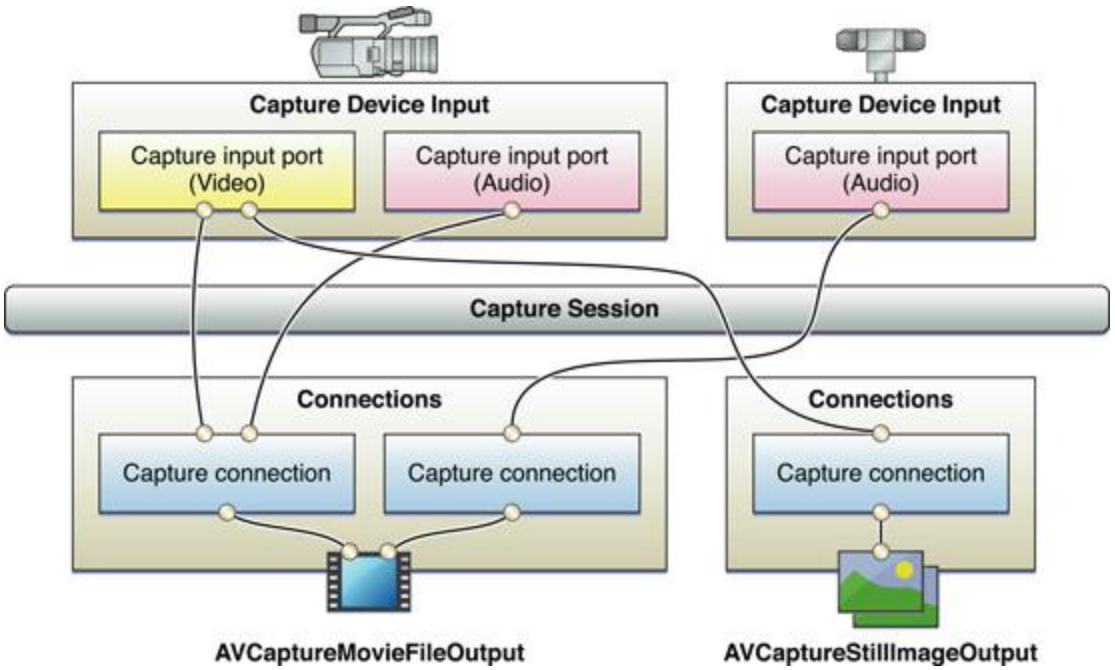


Figure 98 Class relation workflow in high-level video capture [28]

```
// Add video input
NSDictionary *videoCompressionProps = [NSDictionary dictionaryWithObjectsAndKeys:
                                       [NSNumber numberWithDouble:500*1024.0], AVVideoAverageBitRateKey,
                                       nil ];

NSDictionary *videoSettings = [NSDictionary dictionaryWithObjectsAndKeys:
                               AVVideoCodecH264, AVVideoCodecKey,
                               [NSNumber numberWithInt:480], AVVideoWidthKey,
                               [NSNumber numberWithInt:320], AVVideoHeightKey,
                               videoCompressionProps, AVVideoCompressionPropertiesKey,
                               nil];

videoWriterInput = [AVAssetWriterInput assetWriterInputWithMediaType:AVMediaTypeVideo
                                                       outputSettings:videoSettings];
```

Figure 99 Customize the video encoding property to AVAssetWriter

In IOS SDK 5.0, AV foundation Framework provides high-level API for recording video or audio files. Class AVCaptureMovieFileOutput and AVCaptureFileOutRecordingDelegate can capture the video with same quality in camera function in iOS device. Unfortunately, captured video file size is relatively large which only suitable for local playback (~8sec 6MB), also the codec of the video file cannot be configure.

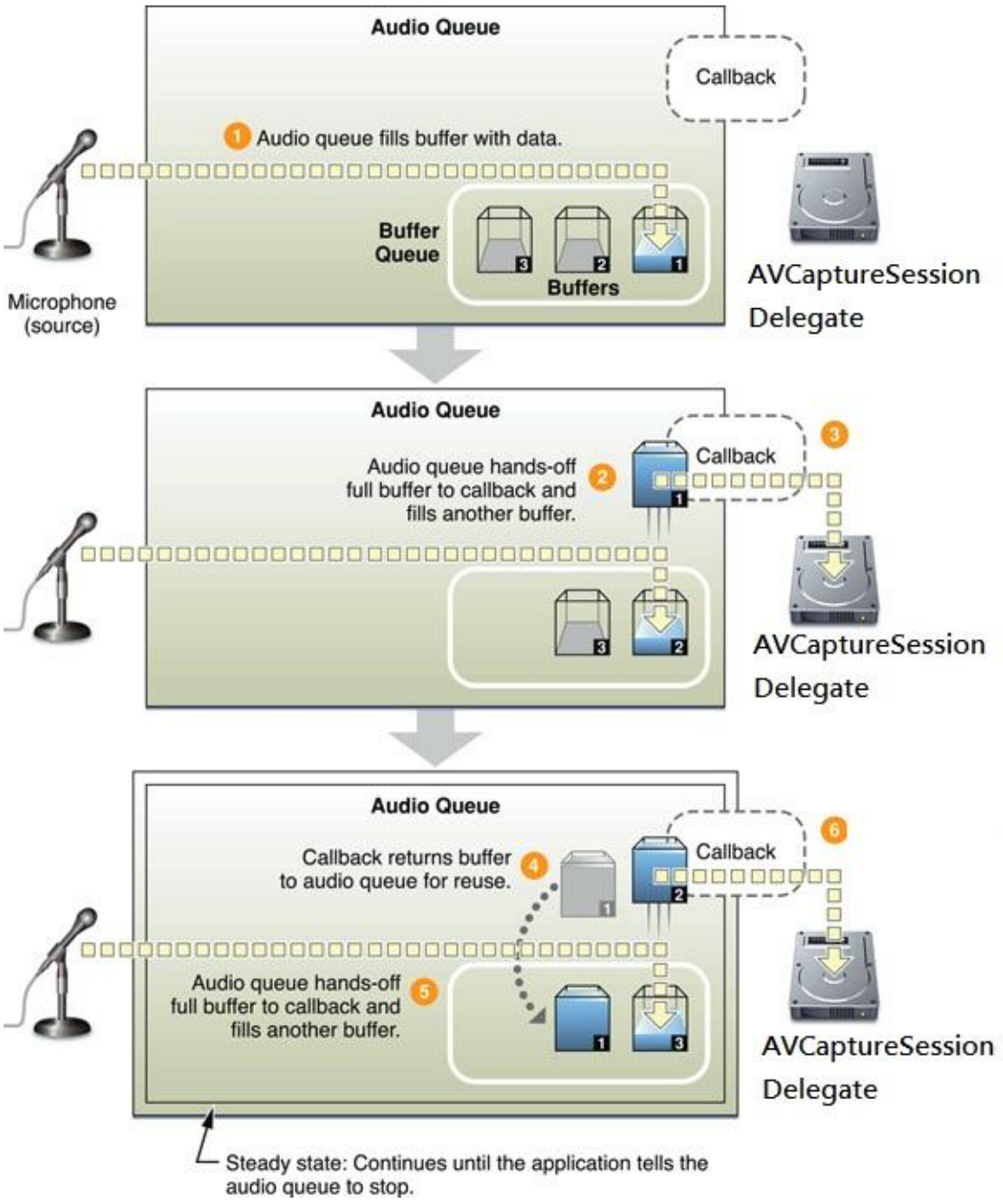


Figure 100 Sample Data capture from device to AVCaptureSession Delegate [28]

To minimize the captured video size which appropriate for streaming, Class AVAssetWriter in AVFoundation Framework provide low level functions for customizing the video encoding process. It is like an encoding writer writing individual frame and audio buffer to the specific time in video file. By this way each video file size can be reduce significantly (~8sec 650KB). Using AVAssetWriter is not as simple as working with AVCaptureMovieFileOutput. When capture session is started, frame and audio are

captured from input devices as a buffer in a separate dispatching queue. Above diagram shows how audio buffer being captured. AVCaptureSession delegate function then append the buffer to AVAssetWriter in the buffer presentation timestamp.

```
case AVAssetWriterStatusWriting:
    //   dispatch_async(movieWritingQueue, ^{

    if( captureOutput == audioOutput) {
        //   NSLog(@"Audio Buffer capped!");
        if( ![audioWriterInput isReadyForMoreMediaData]) {
            return;
        }

        @try {
            if( ![audioWriterInput appendSampleBuffer:sampleBuffer] ) {
                //     NSLog(@"Audio Writing Error");
            } else {
                //  [NSThread sleepForTimeInterval:0.03];
            }
        }
        @catch (NSException *e) {
            NSLog(@"Audio Exception: %@", [e reason]);
        }
    }
    else if( captureOutput == videoOutput ) {
        //  NSLog(@"Video Buffer capped!");

        if( ![videoWriterInput isReadyForMoreMediaData]) {
```

Figure 101 Code segement: AVCaptureSession delegate appending buffer to AVAssetWriter

During the development, AVAssetWriter seems unstable under low memory or high CPU usage; sometimes the captured data buffer appear to be corrupted. Append corrupted buffer might cause a several circumstances; it may cause the frame or audio cannot play out correctly, in worst case these can mislead AVAsseWriter save an invalid duration to video file. Hence, we have to according the AVAssetWriterStatus to determine the buffer should or should not append to AVAssetWriter.

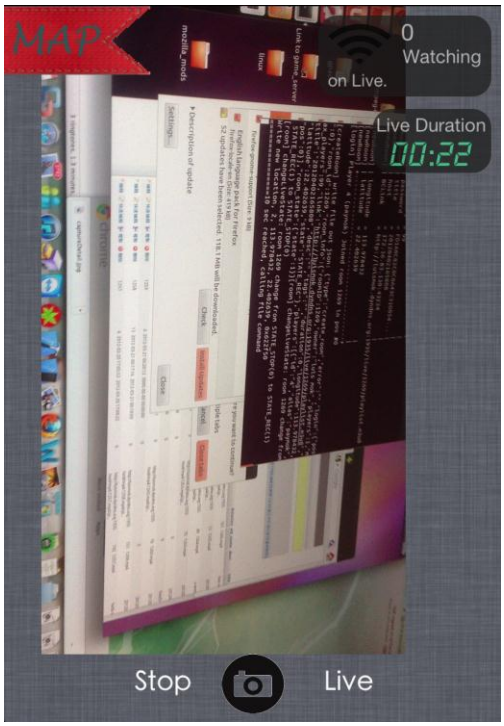


Figure 102 Video Capturing and broadcasting to server

```
- (void) newVideo: (NSTimer*) theTimer
{
    // [videoWriter cancelWriting];
    if(![videoWriter finishWriting]) {
        NSLog(@"finishWriting returned NO") ;
    }

    NSLog(@"Stopped, dict value %@_%@_%d.mov", filename, user, fileCount);
    NSDictionary *dict = nil;

    [[self vcDelegate] finishedVideoCompression:[NSString stringWithFormat:@"%@_%@_%d.mp4", filename, user, fileCount]
        dictionaryWithObjectsAndKeys:

                                                                    @"filename",

                                                                    @"user",

                                                                    numberWithInt:fileCount

    [videoWriter release];
    videoWriter = nil;
    videoWriter = [self getNewAVAssetWriterByURL:[NSString stringWithFormat:@"Documents/%@_%@_%d.mp4", filename, user,
}
```

Figure 103 Code segment: Swap new AVAssetWriter

Playbacks are save to the local file system in every 8 seconds since the recording is started, so that the saved video files can be uploading to server. To make the recording continue in every 8secs, a new AVAssetWriter needed to take place the job. During the development, through multiple experiments, AVAssetWriter invoke finishWriting() function to stop appending further buffer but due to the processing strength, apparently the append buffer process is fall behind the real time which reflected the writer continue appending the buffer in queue after caling finishWriting().

In order words, a new AVAssetWriter needed to replace the old AVAssetWriter once it is finishing writing.

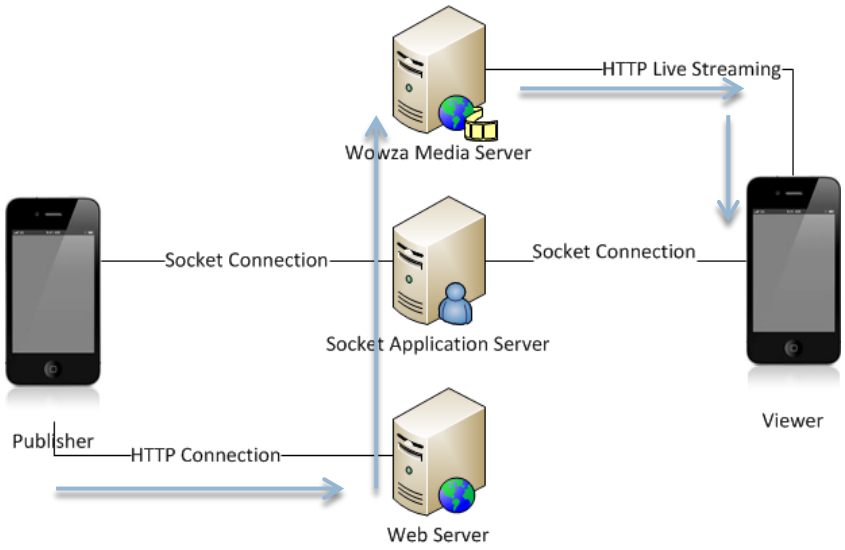
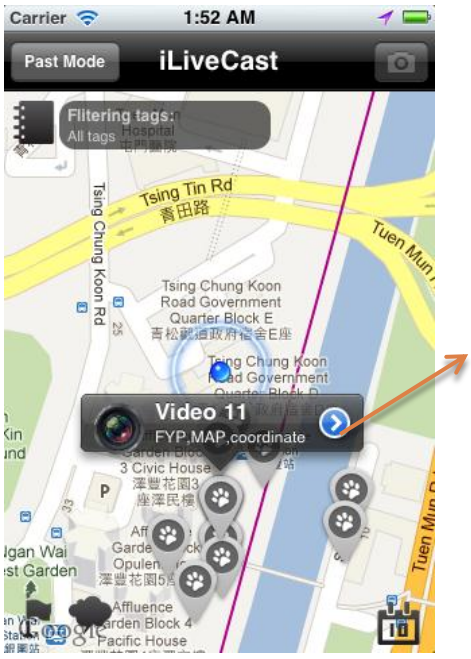


Figure 104 Streaming architecture flowing diagram

Class HTTPFileUploader is an http request handler class, which extended ASIHTTPRequest library and extended to VideoCaptureMangerDelegate. It used to upload video file prepared by capture session to web server. ASIHTTPRequest is a wrapper of CFNetwork API to extend the functionality of communicating with web server. The ASIHTTPRequest provide easy use of submit HTTP request with asynchronous process. It is very important to transfer the file in asynchronous mode, as we cannot afford to block the process of UI, gesture detection or video capturing. When a video file is available in file system, VideoCaptureManager called the delegate functions of "finishVideoCompression" notify resource is ready. The upload process are execute in is a dispatch queue in "DISPATCH_QEUE_SERIAL" mode, upload process is running in FIFO. On each upload finish, a JSON message is delivered to socket server, socket server will handling upload

7.3.4 HLS live streaming



desc = " ";
duration = 42;
latitude = "22.403054";
link = "http://luismok.dyndns.org/vod/mp4:1246.mp4/playlist.m3u8";
longitude = "113.975902";
owner = "luis mok ";
roomID = 1246;
tags = "Video2,UFO";
title = "Video 2 ";
watchCount = 21;

Figure 105 JSON object contents transform to annotation in map

Each annotation object on map contains message as above shown. When users press on the info button, system initializes Class MyMoviePlayerController with the annotation object. The controller extends the MediaPlayer Framework to play URL provided in HTTP Live Streaming (HLS theory please refer to Figure21). MediaPlayer Framework intelligently checks the file type of the link to determine which streaming should use. The m3u8 playlist provided by the media server, which indicates the streaming, is in live or video on demend.

Chapter 8: Evaluation

8.1 User interface



Search Video

User account related

Figure 106 Functions are clearly divided on the left and right as sub-functions.

In tradition iPhone application, tab bar controller and navigation controller are used to switching pages. This design approach extends a few problems. First, it occupies spaces on the bottom. Second, there is less perception for user which page is the most important page for the system. Thrid, this project is aim to provide a map-base live streaming, the map function should have higher contrast compare with others sub-functions.

Therefore the user interface is designed in a slide theme. This approach offers a few advantages in UI interaction and improves the assumption to the UI by user. First, whenever the app first started or user done some action, the map always show on the middle of screen give users a perception which function is major function or

sub-functions in the system. Second, instead of press a button, user swipe left or swipe right to enter specific function provides more interaction between user and system.

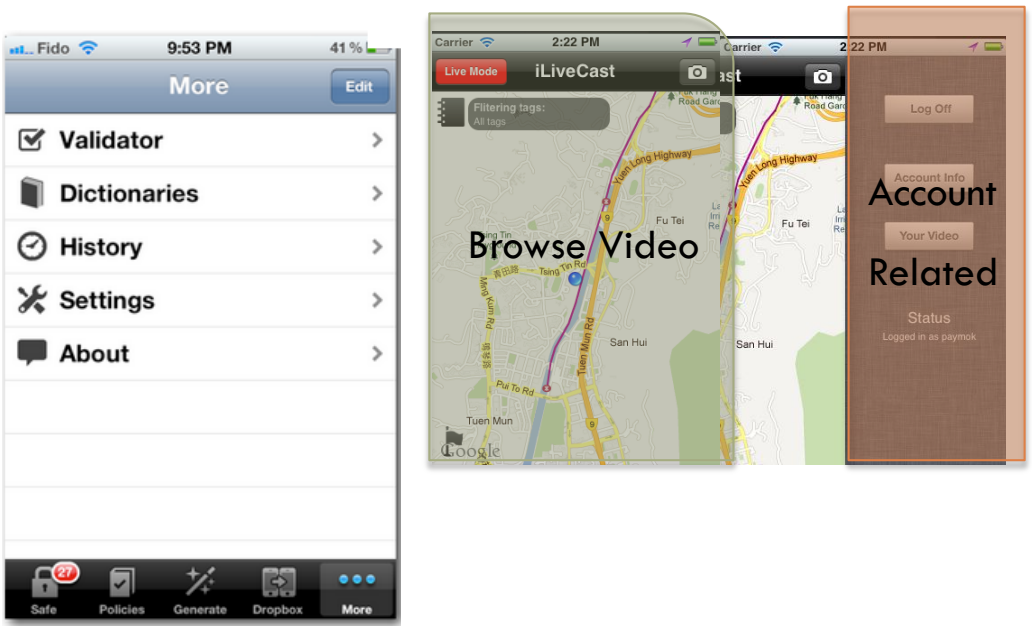


Figure 107 More function in Dropbox app in iOS (left), proximity of function grouped on a side

Third, it grouped the function in proximity, helps user easily to recall which functions are located. For example, “Search” function are in the left view, “Account related” functions are in the right view. “Video radar” on the left view, “Video details related” functions are in the right view. In most of the tab bar based application, core functions are located in tab bar but leave the sub-function or low-priority functions in “More” sections. All kind of options is crowding in the list view, which makes user hard to recognize the function by UI proximity.

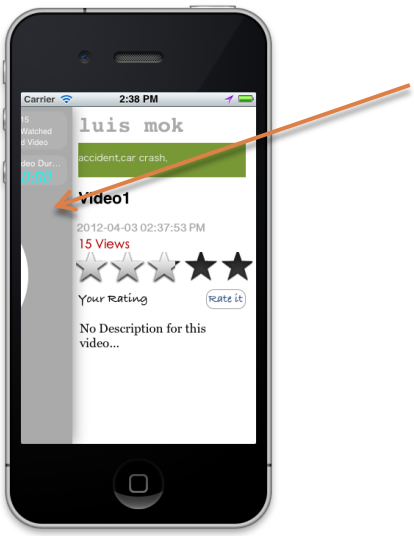


Figure 108 Camera view remains visible (left), while user browse video details pages

Last but not least, the UI structure gives the perception of multitasking. In most of the case, users have a subconscious when a view is completely disappeared on screen, the process should be stopped. By this rule, the UI in the system clearly inform the process is still running by leave part of the view on screen. In other words, user can assume the process is stopped when view is disappeared. It helps user to editing the video details while video captures.

8.2Performances

The application is tested in two areas. (A) Using the WIFI in the apartment, connected to the server hosted in the Project Lab in City University of Hong Kong. (B) A field-testing with cellular network in a bus ride connected to the server in project lab via CS Lab VPN. The testing was covered about live video streaming, location update and content search functions. Following are some testing results of both settings:

Performance on update live content to map on live

Test the refresh response on the map (The update process execute in every 3 seconds iteratively)

Local Wifi network	From the time counter the update occurs every 3.5 - 4 seconds which make around 1 second delay. No update fail due to transmission occurred in 5 minutes testing.
Cellular Network	From the time counter the update occurs every 4 - 7 seconds which make around 3.5 second delay. Sometimes update does not occurred while the connectivity to 3G network has 4/5 signal.

Performance on update history content to map in past mode

Test the refresh response on the map (The update process execute in every 3 seconds iteratively)

Local Wifi network	From the time counter the update occurs every 3.5 - 4 seconds which make around 1 second delay. No update fail due to transmission

	occurred in 5 minutes testing.
Cellular Network	From the time counter the update occurs every 3.5 - 4 seconds which make around 1 second delay. No update fail due to transmission occurred in 5 minutes testing it seems the connection to web server has least delay.

Performance on retrieve recommended result from server

The recommended function is based on the normal query function, when recommendation on a flag enabled to ask server deliver the recommended result rather than normal query result.

Local Wifi network	From the time counter the update occurs every 3.5 - 4 seconds which make around 1 second delay. No update fail due to transmission occurred in 5 minutes testing.
Cellular Network	From the time counter the update occurs every 3.5 - 4 seconds which make around 1 second delay. No update fail due to transmission occurred in 5 minutes testing.

Performance on search function

Test by input a keyword in search function to perform a full-text search in database.

Local Wifi network	Usually the result displayed after 0.5 - 1 second when pressed enter button. No query failed occurred during testing.
Cellular	Result displayed after 1 - 3 second when pressed enter button. No

Network	query failed occurred during testing.

Performance of publish/view a live content

Test the publishing smoothness from iPhone devices by creating a live streaming event and viewed by other devices.

Local Wifi network	Tested to preview content form iPhone simulator. Content streaming quite smooth with a few times paused due to the video buffering with 5 minutes. When additional device joined to streaming session, both audiences device appear auto-skipping playback and more buffering occurred. This is assumed due to the leak of bandwidth with Media Server. Usually the content delay 30-40 seconds compare to the real time.
Cellular Network	Tested to preview content form iPhone 4. Content streaming appear over 20 times paused due to the video buffering with 5 minutes. Playback usually stunned because leak of bandwidth for publisher device uploading video stream to server. Usually the content delay 50-70 seconds compare to the real time.

Performance of instance message

Test by multiple actions, which produce an instant message. Verify the responsiveness from other mobile devices.

Local Wifi network	Tested with info change message from publisher. Message usually appears on participant devices after 1 sec when publisher pressed

	confirm. For leave room and join room message, responds usually in 0.5 second – 1 second. No data from is socket missed.
Cellular Network	Tested with info change, leave room and join room message. Message usually appears on participant devices after 2-8 sec when publisher pressed confirm. Around 2/10 messages were unable to deliver to participant during testing. This may also due to the transmission problem of 3G networks.

Performance of view an video on demand content

Test the content on demand streaming directly from media server.

Local Wifi network	Content streaming quite smooth with a few times paused due to the video buffering with a 3 minutes testing video. When additional device joined to streaming session, both audiences device appear more video buffering but short in most of the time. This is assumed due to the leak of bandwidth with Media Server.
Cellular Network	Content streaming with a few times long paused due to the video buffering with a 3 minutes testing video.

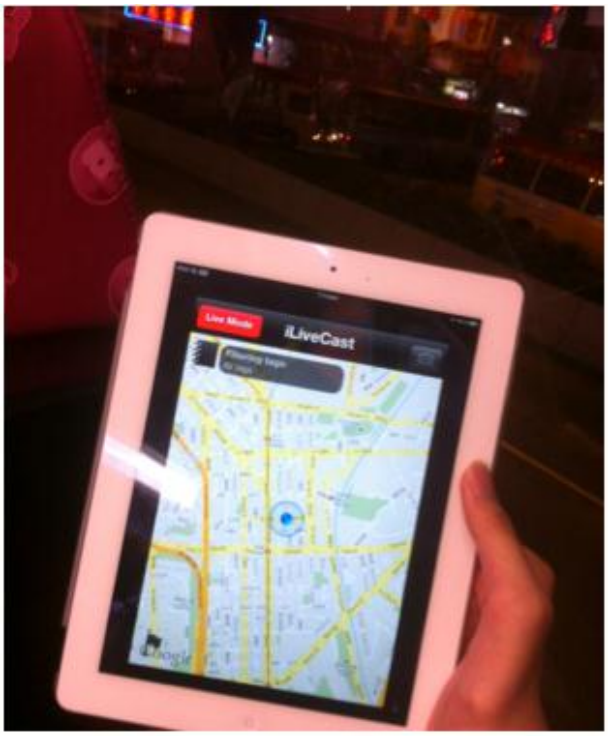


Figure 109 Testing during a bus ride in somewhere in Mongkok – using the new iPad and iPhone 4 testing the location function and video streaming function

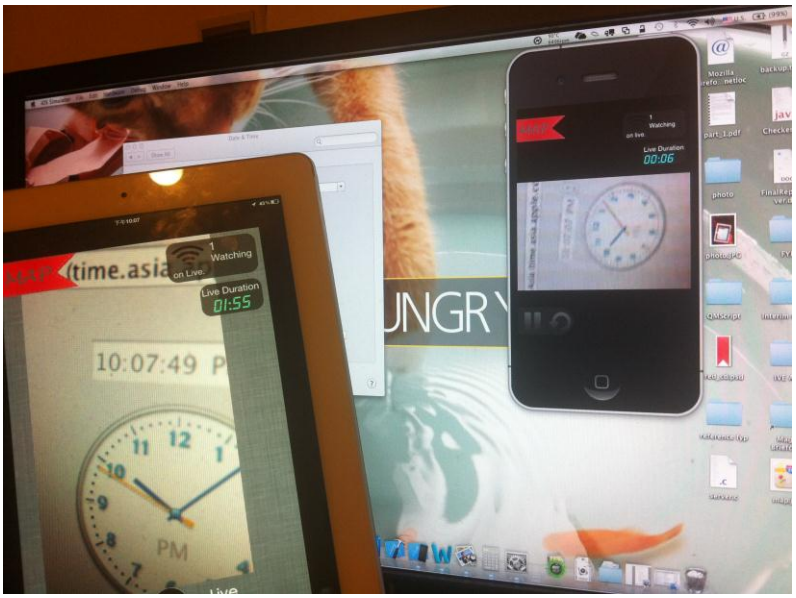


Figure. 110 Testing in apartment – using the new iPad and iPhone simulator connected to City University Project lab server, testing streaming video function

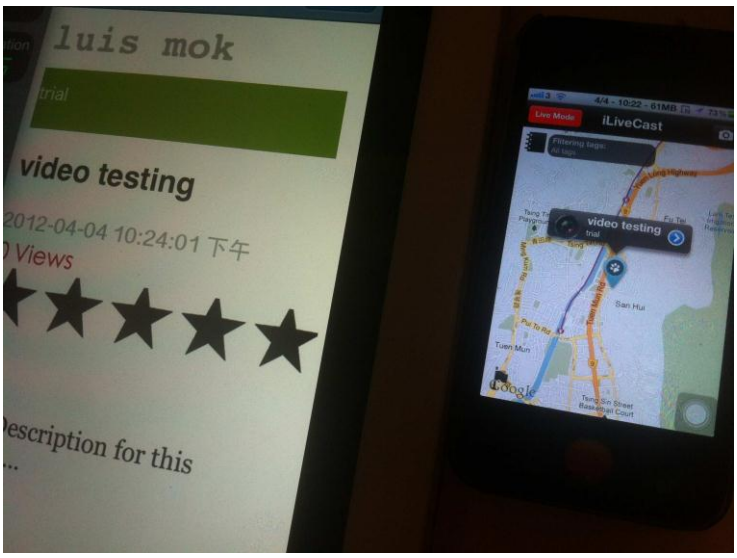


Figure. 111 Testing in apartment – using the new iPad and iPhone simulator connected to City University Project lab server, testing position of live streaming in main map

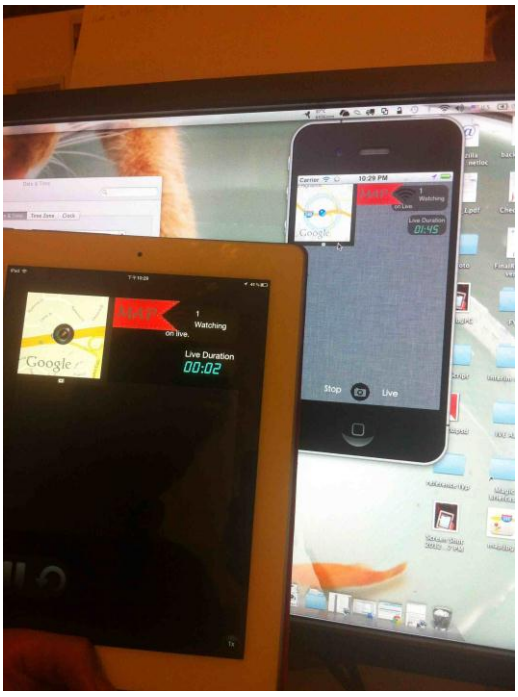


Figure. 112 Testing in apartment – using the new iPad and iPhone simulator connected to City University Project lab server, testing position of live streaming in main map

Chapter 9: Limitation

9.1 Processing power in iDevice

Although hardware specification in real device is quite similar between iPhone 4, iPhone 4S or iPad. The processor makes huge difference in performance. For example, iPhone is a single core A4 chip, while the video is recording, most of the processing power goes to capturing video/audio sample buffer, which makes the UI laggy or stunning when handling socket. Also the high rate of CPU usage may randomly cause the captured buffer corrupted. To reduce this occurrence of this situation, some optimization needed to concentrate very careful such as memory management. In this stage to have a smooth user experience, iPad2, iPhone 4s and iPhone simulator are the ideal testing devices.

9.2 3G network connectivity

It is tragedy of local service provider has a very poor bandwidth especially this project is emphasis on working outdoor with live streaming. Through a multiple testing in outdoors, the 3G networks reflect significantly slow on upload (usually below 20KB/s upload), situation goes worst especially when testing in a bus (0KB/s) in certain area. It makes the streaming become very laggy and unable to maintain the result in least delay which makes it is impossible to test live streaming along with location update in the real situation. To test two of this functions can work properly, the only way is testing in iPhone simulator by simulate a free ride on somewhere in US, but since the simulator cannot use the camera on the computer, a pre-recorded dump video loop constantly to

simulate the recording process.

9.3Media Server

Media Server require huge amount of network bandwidth in transfer video content to client. it might easy to build the media server in a powerful enough computer, but a high-speed WAN network is not easy to afford especially because of the geometry limitation. Also due to the leak of license, maximum connection is limited. It is hard to test the capability of how much current connections can handle. Fortunately, in simulate the high-rate bandwidth, internal network enable the media server with over 2MB upload/download bandwidth. But still the actual performance remain mystery.

Chapter 10: Future enhancements

The project works properly in this stage, as one man has very limited strength also time is limited, we cannot deny that the project is far from perfect. There are still many rooms for improvement to the application in enrich the user experiences.

10.1 Dynamic video quality

It is obvious that the connectivity provided from local cellular network is very stable in certain area, this factor has been proved greatly influence the delay of live streaming even the smoothness. Therefore, in future the system should base on the bandwidth to dynamic switch the quality of recording in order to further minimize the stream size. Also, Media server should implement multicasting streaming in dynamically switch the video quality deliver to viewer.

10.2 Load balancing

The socket server using right now handle up to hundred concurrent connection, but if too much connection or process in the server, a significantly performances drop is expected. In solve the problem the server should designed with load balancing access. In future, the client should not directly connect to the socket application server; instead the client should connect to an entry point (PHP web server). The web server based on the server loads of each socket application server instances to redirect the client connection to particular socket address.

10.3 Privacy settings

All in all, this project is under an experiment stage, to make it usable in real life globally, the privacy settings must be improved from this point. Such as password for limited access in content; limited access of friend only.

10.4 Social Network features

There is very insufficient time in the development for implement social network feature. Following feature definitely should implement in the future for enhance the experiences of easy advertising your own video content. For example, link up account with facebook enable other user can access into iLiveCast through facebook; group or event video enable peoples can easily share the videos with friends; or follow functions let user become a follower to publisher. When new live event is started all follower will receive an instant notification.

Chapter 11: Conclusion

The needs of video sharing application have a very high demanding in every platform. A new location based application live streaming application definitely gives a new type of impact in media sharing area. The objective of the study is find out the possibility in provide a live streaming features along with location information.

In the project, there are studies on mobile device in using GPS location detection, the video compression technique and use streaming protocol. In propose a solution for live streaming, an evaluation of choose of sockets and streaming methodologies has been carry out in select the technical approaches in this project. Proposed an HTTP media streaming approach and Slope-one recommendation algorithm in suggesting content on map are highly valued and respond to the aims of this project.

The overall system is combined with a C socket application server in uBuntu, a web server and Wowza media server. This system architecture able to provide instant response to the developed mobile application. The proposed application developed for iOS devices enable user to stream video from camera to other users. Thanks to the GPS functionality audience can track publisher location instantly while watching the live content. Time reduced in content upload as system able to capture recorded content during streaming. Rather than using list view, the application use google map listing all available contents on map with enable tag or date filter, The system able to suggest contents within visible range in map by users rating deviation. From this view, contents can advertise to other users much more easily. Thus, the project able to confirm the goal and objectives of this project.

However, the project has its limitation. To enable good user experiences network in mobile devices must enforce in certain bandwidth. Also the Media Server must equip with high speed network in distributing video contents, otherwise, the application is like a vase.

The project has a high potential in continue develop as a global platform for video sharing. This enhancement could allow content much easily in broadcasting to the global. And the sharing circulation is more flexible.

References

1. Wheeler, J., Aoyama, Y., & Warf, B. (Eds.). (2000). City space, industrial space, and cyberspace. In *Cities in the telecommunications age: the fracturing of geographies*. New York: Routledge. 3-17.
2. Thomas, D ,& Buch, V (2007, March 18). YouTube Case Study: Widget marketing comes of age. StartupReview. Retrieved October 22, 2011, from http://www.startup-review.com/blog/youtube-case-study-widget-marketing-comes-of-age.php
3. Elgan, M (2011). How iPhone Changed the World. Cult of Mac. Retrieved October 22, 2011, from http://www.cultofmac.com/103229/how-iphone-changed-the-world/
4. Dowling, S (2011). Apple reports third quarter results. Apple. Retrieved October 20, 2011, from http://www.apple.com/pr/library/2011/07/19Apple-Reports-Third-Quarter-Results.html
5. Google Map for mobile [electronic Map]. Google, Retrieved October 3, 2011, from http://www.google.com.hk/mobile/default/maps.html
6. Jagoe, A. (2003) *Mobile Location Services : the Definitive Guide*. New Jersey: Pearson Education Inc.
7. Apple Inc. (2009). *Maps - iPhone and iPod touch: Understanding Location Services*. Retrieved 20, October, 2011 , from http://support.apple.com/kb/HT1975
8. Mat Balez (23 July, 2009). *Google Latitude. Now for iPhone*. Retreived 20 October,

2011, from

http://googlemobile.blogspot.com/2009/07/google-latitude-now-for-iphone.html

9. Anttila, I, & Paakkunainen, M (1997). *Transferring real-time video on the Internet.* Retrieved 20, October, 2011, from

http://www.tml.tkk.fi/Opinnot/Tik-110.551/1997/iwsem.html

10. ISO/IEC. (2004). Coding of audio-visual objects part 2: visual. Retrieved October 18, 2011, from the website of IEC at

http://webstore.iec.ch/corrigenda/iso/isoiec14496-2-cor1%7Bed1.0%7Den.pdf

11. Thinh, P.N. (1999). *Performance analysis of H.263 video encoder for VIRAM.* Berkeley: University of California.

12. Furht, B., Greenberg, J. ,& Westwater ,R. (1996). *Motion estimation algorithms for video compression: Fundamental concepts of motion estimation.* Heidelberg: Springer.

13. Chung-How, J.T.H, Bull, D.R. (2000). *Robust H.263+ video for real-time internet applications.* Vancouver: IEEE.

14. Rijkse, K. (1996, December). *H263: Video coding for low bit rate communication.* IEEE Communications Magazine, p.45.

15. Sullivan G.J, Topiwala P., & Luthra A. (2004). The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions. *Proceedings of SPIE, 5558,* 454.

16. Schulzrinne , H. (1996). *RTP profile for audio and video conferences with minimal control.* United States: RFC.

17. Steve, J. (2010, April). *Thoughts on flash.* Apple, Retrieved November 18, 2011, from the website of Apple at

http://www.apple.com/hotnews/thoughts-on-flash/

18. Apple (2011, April). *HTTP live streaming overview*. Retrieved November 18, 2011, from the website of Apple at http://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/StreamingMediaGuide.pdf

19. McLean, P. (2009). *Apple launches HTTP live streaming standard in iphone 3.0*. Retrieved October 20, 2011, from the website of Apple insider at http://www.appleinsider.com/articles/09/07/08/apple_launches_http_live_streaming_standard_in_iphone_3_0.html

20. Qik (2011). *Mobile video capture and sharing*. Retrieved Feburary 3, 2012, from the website of Qik at http://qik.com/info/overview

21. Stickam(2011). *Get featured on Stickam*. Retrieved Feburary 3, 2012, from the website of Stickam at http://www.stickam.com/help/getFeatured.do

22. UStream(2011), *About Us*. Retrieved Feburary 3, 2012, from the website of UStream at http://www.ustream.tv/about

23. Nick, M. (2011). *A tiny introduction to asynchronous IO*. Retrieved October 20, 2011, from the website of Libevent manual at http://www.wangafu.net/~nickm/libevent-book/01_intro.html

24. Nick, M., Niels, P. (2011). *Libevent – an event notification library*. Retrieved October 20, 2011, from the website of Libevent at

http://libevent.org/

25. Apple (2010). *Using HTTP Live Streaming*. Retrieved November 18, 2011, from the website of Apple iOS Developer Library at https://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html#//apple_ref/doc/uid/TP40008332-CH102-SW5

26. Apple (2010). *HTTP Live Streaming Overview*. Retrieved November 18, 2011, from the website of Apple iOS Developer Library at https://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html

27. Apple (2010). *iOS Developer Program*. Retrieved November 18, 2011, from the website of Apple Developer at https://developer.apple.com/programs/ios/

28. Apple (2010). *iPhone 4 Tcch Specs*. Retrieved November 18, 2011, from the website of Applev at http://www.apple.com/iphone/iphone-4/specs.html

29. Qun, F.C. (2008). Open Slope one. Retrieved March 3, 2012, from website of Foogle code at http://code.google.com/p/openslopeone/

Appendix

Monthly log

March, 2012	- Added instant message for join room, leave room and info change - Fixed video player crashing - Implemented location tracking in video player map view - Implement slope one algorithm for suggesting video Content - Implement filter date when surfing old videos - Implement tags filter on both live/ vod mode - Updated location annotation - Updated implement search function of the video - Implement video rating - Fix location update with animation now - Implement browse own video - Implement video details instant update - Fixed room cannot close after publisher disconnected - Fixed room cannot close if participant still watching the movie - Fixed socket stunning problem - Fixed map view bubble appearances problem - Add video duration to map view - Successful play live video on wowza media server - Fixed user register form problem

February, 2012	- Studied send video files via socket might block the server running also the video might corrupt due to connection unstable - Successful upload the chunk video file via tcp to php web server - Able to preview recorded video via vod - Able to preview reocrded video location track on mini map - Redesigned User interface - Studied the difficult using fms 4.5 concat chunk video files as live, abandon fms 4.5 media server as media server - Successful installed wowza media server - Successful play chunk video files as a playlist on a live stream
January, 2012	- Developing PHP page for handling DB request - Integrated location update with socket server - Implemented database design in MYSQL - Successfully create streaming via HLS by adobe FMS 4.5 - Studying the possibility of constantly live streaming multiple video files - Studying the possibility of RTMP in iPhone - Developing the UI interface - Implemented the account management function - Implemented the instantly location update by socket server
December, 2011	- Successful startup the socket server in C under linux. - iphone able to connect to socket server, - Developing protocol communicate with jSON object between Linux and iphone

November, 2011	- Study socket server in C and start writing a socket server for handling users request in real-time. - Finish the prototype of getting user geo location in iphone
October, 2011	-Week1: Complete a prototype of broadcasting video application, the connection drop constantly in unknown reason. develop google map application prototype on iOS. -Week 2: Research on how to establish a red5 media streaming server, trying to change the media server from wowza to red5 -Week 3: Not good progress on Red5 media server, connection drop problem also have not solve yet. Found Adobe media streaming server also work with iOS device, the media server have capability to switch the streaming protocol to HLS if iOS connection detected, will start the research on next month. -Week 4: Stuck most of the time on streaming issue. Stop the streaming development for a while. Starting research on google map, instant messaging and instant update. Researched in PHP for updating status, not going to use PHP approach, PHP approach cost much server resource in getting update, too much connection will likely to crashed the server. Socket programming is suggested to implement instant chat function and instant update. jSON protocol will be in use to sending the update

	message. start the research on how to working with jSON + socket programming.