CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Introduction to Project

The main objective of this project is to implement real-time video streaming over multiple wireless hops between peer devices based on Multi-hop Routing. For this to be possible, there has to be discovered a way to allow creation (and termination) of networks, with the Android OS.

The second goal is to implement a simple Android application, to run on these devices, utilizing the main possibilities of the created multi-hop network and streaming video from one phone to other through wireless networks.

1.2 Problem statement

The rapid adoption of smart phones has created a unique opportunity for mobile multimedia services for mobile users. Currently a majority of smart phones are equipped with hardware that supports real-time video processing and wireless communication between peers; this allows real-time video streaming over multiple wireless hops between peer devices.

Currently there are no systems available for wireless multi-hop video streaming. If any of these systems use the TCP/IP protocols for communicate and also for transferring the frames from one system to another. These systems also follow the store and forward procedure for transferring the frames. Normally these systems can transmit and re-transmit the copy of the frames from the source or server.

Qik is one of the best known real-time video streaming services, which has support for a number of mobile devices. It adopts the client server architecture; that is, mobile

phones (with Qik as client) stream live video to centralized processing servers using the available network infrastructure (such as cellular or Wi-Fi networks).

The main problems found are as follows:

- Video sharing is done over the Internet, placing great dependency on the network infrastructure.
- Qik implements a customized codec and development library that poses an issue on portability to other mobile devices for wider deployment.
- Peer-to-peer live video streaming is currently not supported by Qik.

1.3 Brief description of the project

As it is already known majority of smart phones are equipped with hardware that supports real-time video processing and ad-hoc wireless communication between peers, so phones within communication range of each other automatically establish a wireless link creating a client mesh network. Each phone in the client mesh network is able to produce/consume video and also acts as a relay to forward video to its next hop neighbors. Peer-to-peer video streaming from the cameras on smart phones to people nearby allows users to share what they see. Such streaming can be used in a variety of applications, in particular in various social network applications.

This project presents a wireless multi-hop video streaming application for the Android based mobile devices (including Android based tablet PCs). This application allows users to capture live video using camera on mobile devices, and to share this video feed with people nearby using a free-of-charge wireless mesh network i.e., one person will be capturing video using an android phone which can be transmitted to many number of clients such as laptops and other android operating phones. Routing protocols can be installed to facilitate the multi-hop communication to go beyond a single hop. Here the device which is capturing the video is the main source of video transmission, the video is broadcasted to all the registered IP address in the source device later on this video

transmission can be further transmitted to different clients by the receiver by entering the IP address of the clients in clients device (i.e., client to client transmission).

An another feature is included in this application using which an unregistered client can receive the video transmission i.e., if an unknown client wants to receive the live video transmission then the client can send a message to the server with the client IP address (for example: IP 192.168.1.3) then the server automatically reads the message sent by the client and transmits live video to the IP address sent by the client. Also when a client is receiving live video transmission from the server or client it is possible for a client to capture live images of the video transmission.

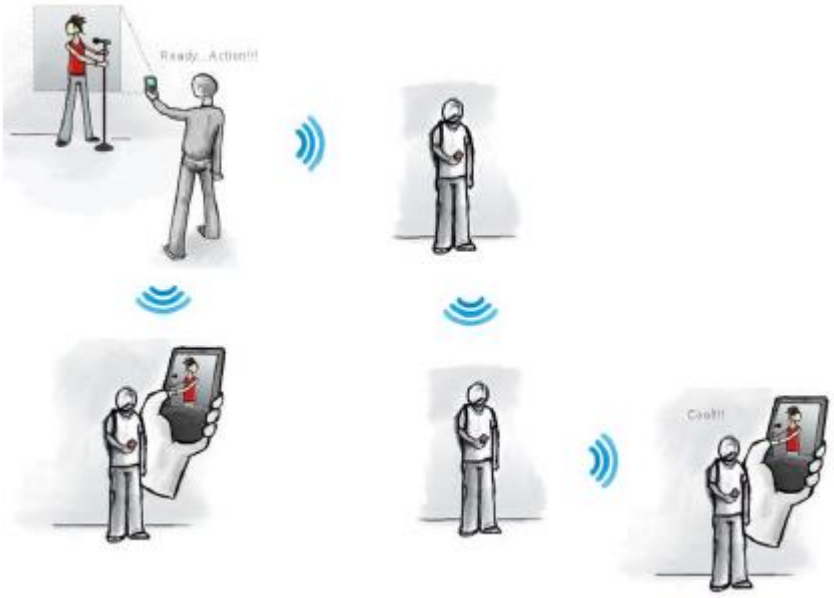


Figure 1.1 Example application scenarios

The Figure 1.1 shows an example application scenario in which a person streams live video to nearby clients over the wireless mesh network of phones and further clients are transmitting the live video to other clients using the same gateway. Here while the video is being captured and transmitting to the clients it uses encoding and decoding technique.

1.4 Scope of the project

In order to develop the new system, first we have to understand the history of the existing system. This history will give the complete information about the existing system and also, this history will be the bright introduction for the new system. This information tells the advantages and the disadvantages of the existing one. So the problems can be identified. This helps for the development for the new system.

Video streaming over multiple wireless hops between peer devices. Phones within communication range of each other automatically establish a wireless link creating a client mesh network (ad-hoc network of devices). Each phone in the client mesh network is able to produce/consume video and also acts as a relay to forward video to its next hop neighbors. Peer-to-peer video streaming from the cameras on smart phones to people nearby allows users to share what they see. Such streaming can be used in a variety of applications, in particular in various social network applications including sharing unforgettable moments with friends that can be multiple wireless hops away, cooperative fieldwork (providing video sharing for teams distributed in a small area, e,g. teams of repairmen, and search and rescue teams in disaster areas), and support for health impaired persons including the elderly.

In this project we present a wireless multi-hop video streaming application for the Android based mobile devices (including Android based tablet PCs). This application allows users capture live video using camera on mobile devices, and to share this video feed with people nearby using a free-of-charge wireless mesh network. Routing protocols can be installed to facilitate the multi-hop communication to go beyond a single hop. an example application scenario in which person streams live video of a concert to friends nearby the wireless mesh network of phones, without being caught by the expensive mobile phone bill. We evaluate the presented peer-to-peer video streaming application in a variety experiments. In these experiments we show feasibility.

1.5 To improve service quality of mobile video streaming

1.5.1 Scalability

Mobile video streaming services should support a wide spectrum of mobile devices; they have different video resolutions, different computing powers, different wireless links (like 3G and LTE) and so on. Also, the available link capacity of a mobile device may vary over tiame and space depending on its signal strength, other users traffic in the same cell, and link condition variation. Storing multiple versions (with different bit rates) of the same video content may incur high overhead in terms of storage and communication. To address this issue, the Scalable Video Coding (SVC) technique of the H.264 AVC video compression standard defines a base layer (BL) with multiple enhance layers (ELs). These substreams can be encoded by exploiting three scalability features:

(i) spatial scalability by layering image resolution (screen pixels)

(ii) temporal scalability by layering the frame rate

(iii) quality scalability by layering the image compression.

By the SVC, a video can be decoded/played at the lowest quality if only the BL is delivered. However, the more ELs can be delivered, the better quality of the video stream is achieved.

1.5.2 Adaptability

Traditional video streaming techniques designed by considering relatively stable traffic links between servers and users, perform poorly in mobile environments. Thus the fluctuating wireless link status should be properly dealt with to provide tolerable‖ video streaming services. To address this issue, we have to adjust the video bit rate adapting to the currently time-varying available link bandwidth of each mobile user. Such adaptive streaming techniques can effectively reduce packet losses and bandwidth waste

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

This chapter consists of the technical literature survey carried out by referring to books, journals, and IEEE papers. The journal papers and IEEE papers used for this project are described in brief. This chapter reflects upon the work going on all over the world in areas of video streaming over Smartphone's. In the end critical conclusion drawn from the carried out literature survey are highlighted. This literature survey also reflects the gaps in the ongoing research work and hence highlighting the need for this project to the fast growing technology of this world.

2.2 Contents and their Abstract

- **Title : ROI-Based Error Resilient Coding of H.264 for Conversational Video Communication**

  Author : Fan Li ; Yixin Gao,2011

  Abstract : Abstract —Transmission of compressed video over packet lossy networks su ff ers from the problem of error propagation which deteriorate the reconstruction quality of a number of successive frames. For conversational video communication, this problem becomes more severe due to the localization and predominance of the attention-attracted areas, where the motion compensated prediction errors have more energy compared to that of the relatively invariable background. In this paper, we propose an error resilient coding scheme for conversational video communication. The proposed scheme combines the region-of-interest (ROI) segmentation and the long-term reference (LTR) frame coding. The scheme can be easily integrated into the H.264 / AVC encoder, and

efficiently improve the video quality of the ROI areas. Simulation results verify the robustness and coding e ffi ciency as well as improved subjective visual quality.

Index Terms —video communications; region-of-interest; H.264.

- **Title : OFDMA Channel Aware Error-Resilient Mobile Streaming Video Service over Mobile WiMAX,2012**

  Author : Jae-Young Pyun, *Member, IEEE*, DongYou Choi, Sun-Kuk Noh, Goo-Rak Kwon, *Member, IEEE*

  Abstract — The growing demands for advanced mobile multimedia let the wireless mobile Internet access be developed rapidly. The Mobile WiMAX (IEEE 802.16e) is capable of providing high data rate and flexible Quality of Service (QoS) mechanisms. However, one of killer applications, mobile streaming video service from mobile station (MS) to fixed internet network can't still avoid a traditional QoS degradation problem caused by unreliable mobile channel environment. The previous researches show that streaming video service based on downward cross-layer design over OFDMA channel has improved robust features to channel errors, but requires modifications of the existing schedulers, packet ordering, and channel allocation strategies in base station (BS). Now, our goal is that we develop a high error resilient and robust method to channel errors on the OFDMA mobile video device without the modification of conventional functions in WiMAX BS. In this paper, we introduce OFDMA channel aware error resilient video coding method using an upward cross-layer design. Through the extensive simulation results, we prove that the proposed channel-aware intra-refresh coding can recover video damages more efficiently than traditional intra-refresh one over OFDMA channel environment. This error resilience feature can be adapted to all types of streaming video server devices .

- **Title : Live Streaming with Receiver-based Peer-division Multiplexing,2011**

  Author: Hyunseok Chang ;Sugih Jamin ;Wenjie Wang

  Abstract—A number of commercial peer-to-peer systems for live streaming have been introduced in recent years. The behavior of these popular systems has been extensively studied in several measurement papers. Due to the proprietary nature of these commercial systems, however, these studies have to rely on a "black-box" approach, where packet traces are collected from a single or a limited number of measurement points, to infer various properties of traffic on the control and data planes. Although such studies are useful to compare different systems from end-user's perspective, it is difficult to intuitively understand the observed properties without fully reverse-engineering the underlying systems. In this paper we describe the network architecture of Zattoo, one of the largest production live streaming providers in Europe at the time of writing, and present a large-scale measurement study of Zattoo using data collected by the provider. To highlight, we found that even when the Zattoo system was heavily loaded with as high as 20,000 concurrent users on a single overlay, the median channel join delay remained less than 2 to 5 seconds, and that, for a majority of users, the streamed signal lags over-the-air broadcast signal by no more than 3 seconds.

- **Title : Key Frame Extraction from MPEG Video Stream,2009**

  Author : Guozhu Liu ; Junming Zhao

  Abstract—In order to extract valid information from video, process video data efficiently, and reduce the transfer stress of network, more and more attention is being paid to the video processing technology. The amount of data in video processing is significantly reduced by using video segmentation and keyframe extraction. So, these two technologies have gradually become the focus of research. With the features of MPEG compressed video stream, a new method is presented for extracting key frames. Firstly, an improved histogram matching method is used for video segmentation. Secondly, the key frames are extracted utilizing the features of I-frame, Pframe and B-frame for each sub-lens.

Fidelity and compression ratio are used to measure the validity of the method. Experimental results show that the extracted key frames can summarize the salient content of the video and the method is of good feasibility, high efficiency, and high robustness.

- **Seferoglu. H., Keller. L., Cici. B., Le. A. and Markopoulou. A. in "Cooperative Video Streaming on Smartphones"**

  Describes the scenario where in a group of smartphone users, within proximity of each other, are interested in viewing the same video at the same time and is also willing to cooperate with each other. The paper [1] propose a system that maximizes the video quality by appropriately using all available resources, namely the cellular connections to the phones as well as the device-to-device links that can be established via Bluetooth or Wi-Fi.

  Key ingredients in the design are: (i) the cooperation among users, (ii) network coding and (iii) exploiting broadcast in the mobile to mobile links. This approach is grounded on a network utility maximization formulation of the problem. It also includes numerical results that demonstrate the benefit of our approach, and implementation of a prototype on android phones.

- **Justin M. Bailey in "Live Video Streaming from Android-Enabled Devices to Web Browsers"** proposes architecture of a real-time video streaming service from an Android mobile device to a server of the user's choice. The real-time video can then be viewed from a web browser. The project builds on open source code and open protocols to implement a set of software components that successfully stream live video.

  The internet combined with smart phones and its peripherals open the doors to limitless mobile possibilities. One of these possibilities is explored and exploited by capturing video on a mobile device, in real-time, and transferring to a web page, viewable by the entire world.

The project [2] develops an open source solution capable of transferring the live video with little overhead on the phone and/or server. Users will have the ability to broadcast news and events live using only an Android-enabled mobile devices and an internet connection via the cellular network or Wi-Fi.

- **Majumder, M., and Biswas ,D. 2012 in "Real-time Mobile Learning using Mobile Live Video Streaming"** deals with mobile learning also known as m-learning, is a convenient, means of delivering informational content to learners using current mobile technology devices. Nowadays, most of the learners have a smart phone that support video as well as have fast internet connection. They are using native content as well as web content for learning purpose. These learning experiences can be made more efficient by providing a facility to access real time video lectures via mobile computing devices. The paper [3] proposes a mobile learning system that enables live video streaming into users' mobile phone. The prototype includes real time virtual classroom architecture through which learners can access live video of classroom lectures or access recorded video using streaming technology. Present worker did not adopt client-server architecture rather chose agent based architecture as agent can make learning successful even when a user is on roaming and if disconnected from the network can resume their learning from the point of disconnection.

- **Saurabh Goel. 2013 in "Cloud-Based Mobile Video Streaming Techniques"** investigates cloud centered video streaming methods particularly from the mobile viewpoint. The research contains explanations of current video development methods, streaming methods and third celebration cloud centered streaming solutions for different mobile which shows my realistic work relevant to streaming methods with RTMP protocols family and solutions for iPhone, Android, Smart mobile phones, Window and BalackBerry phones etc.

Developing multi-media content for effective indication over reasoning of cloud based centered mobile system with limited data rates, such as the 3G-324M system needs skills and knowledge [4]. It needs an knowing of the fundamentals that have an effect on movie quality, such as codec choice and compression, and the use of specific resources, such as the FFMPEG Development, and Zencoder Cloud centered Development API which can be used to validate that the material of videos clip data file are effectively specified for end customers [4].

- **Ibrahim, N.,Ali, A., Razak, A., and Azhar, M. in "The Performance of Video Streaming Over Wireless-N"** due to the growing popularity of video streaming over the wireless devices especially the smart mobile phone, this paper emphasizes on maintaining the quality of service. The parameters such as Delay, Packet Loss and Jitter play important roles in ensuring the quality of video streaming over WLAN.

  The paper [5], performance of video streaming over wireless-N was studied in two different wireless devices which are the notebook computer and the smart phone. The notebook computer operates at 5GHz on Window operation systems and a smart phone operates at 2.4GHz on Android platform. As expected, the quality of video streaming transmitted at 5GHz outperformed the 2.4GHz transmitted signal. The performance measurements were captured by using Omni peek and PRTG software [5]. The use of WLAN as an extension to the existing wired LAN infrastructure offer the convenience of mobility to users especially in business and enterprise environments.

- **Xing, M., Xiang, S and Cai, L. in "Rate Adaptation Strategy for Video Streaming over Multiple Wireless Access Networks"**

  Presents how to efficiently and cost-effectively utilize multiple links to improve the video streaming quality. In order to maintain high video streaming quality while reduce the wireless service cost, in this paper, the optimal video streaming

process with multiple links is formulated as a Markov Decision Process (MDP). The reward function is designed to consider the quality of experience (QoE) requirements for video traffic, such as the interruption rate, average playback quality, playback smoothness and wireless service cost [6]. Using dynamic programming, the MDP can be solved to obtain the optimal streaming policy.

To evaluate the performance of the proposed multi-link rate adaptation (MLRA) algorithm,a testbed has been implemented using the Android mobile phone and the open-source X264 video codec. Experimental results demonstrate the feasibility and effectiveness of the proposed MLRA algorithm for mobile video streaming applications, which outperforms the existing state-of-the-art one [6].

- **Farkas, L. and Aczel, K. in "Streaming Video from Smart Phones: a Feasibility Study"** targets mobile streaming in which Series 60 smart phones are the stream creators and PC-s are the stream consumers. The aim is to provide experimental data reflecting under which conditions video streaming from smart phones is feasible and whether duplex interactive video between a phone and a PC can be achieved or not [7]. Phones used corporate Wi-Fi and UMTS, PC-s were on the corporate intranet and respectively on an ADSL connection.

  The outcomes of the experiments indicate that using the considered technologies and experimental setups in most scenarios video streaming from smart phones over Wi-Fi and UMTS is feasible today, for about an hour [7]. However, providing the level of interactivity needed for video calls is challenging, requiring additional measures. As for the extension of the streaming duration to several hours a technological breakthrough would be required.

- **Wong, C., Fung, W., Tang, C. and Chan,S. in "TCP Streaming for Low-DelayWireless Video"** describes the need of low-delay video streaming over wireless channel. Traditionally, UDP (User Datagram Protocol) is used for video streaming. However, due to unreliable transmission and fluctuating bandwidth of wireless channel, this requires error concealment and recovery mechanisms which greatly increases the complexity and delay of the system. Furthermore, UDP streams are often more difficult to penetrate firewalls. The paper [8] using TCP (Transmission Control Protocol) for video streaming due to its ease of use, reliability and flexibility in selecting frames to transmit. After discussing the wireless system architecture under consideration, this paper present a multi-worker model as implemented at wireless proxy (or encoder) which handles client requests independently.

  The model makes use of a technique (selective packet drop) which selectively drops those unimportant frames so as to maintain video quality and low delay in the presence of congestion and fluctuating bandwidth. This has a cellular and WLAN-based surveillance system using the model, and conduct real network measurement on its performance. Model is simple and effective for mobile clients of heterogeneous bandwidth and computing power [8]. The results show that using TCP for streaming leads to good video quality in wireless networks.

- **Zhu, X and Girod, B. in "VIDEO STREAMING OVER WIRELESS NETWORKS"** provides an overview of the design challenges for video streaming over wireless networks, and surveys recent research efforts in the field. The paper [9] is organized by wireless streaming problems of increasing complexity, ranging from the simple scenario of delivering a single video stream over a single wireless link, to sharing a wireless multi-access channel among multiple video streams to the general case of multiple streams sharing a mesh network. While most of the issues discussed are general, makes use of high-definition (HD) video streaming over 802.11a home networks as a concrete example when presenting simulation results.

- **Luo, H., Wu, D., Ci, S., Sharif, H. and Tang, H. in "TFRC-based Rate Control for Real-Time Video Streaming over Wireless Multi-hop Mesh Networks"** describes TCP-Friendly Rate Control protocol on the basis of TCP Reno's throughput equation, TFRC is designed to provide optimal service for unicast multimedia delivery over the wired Internet networks. However, when used in wireless environment, it suffers significant performance degradation [10]. Most of the current research on this issue only focuses on the TFRC protocol itself, ignoring tightly-coupled relation between the transport layer and other network layers. In this paper, we propose a new approach to address this problem, integrating TFRC with application layer and physical layer to form a holistic design for real-time video streaming over wireless multi-hop mesh networks. The goal of the proposed approach is to achieve the best user perceived video quality by jointly optimizing system parameters residing in different network layers, including the real-time video coding parameters at the application layer, the packet sending rate at the transport layer, and the modulation and coding scheme at the physical layer.

CHAPTER 3

SYSTEM DEVELOPMENT STARTEGY

CHAPTER 3

SYSTEM DEVELOPMENT STRATEGY

3.1 System Engineering

Systems engineering is an interdisciplinary process that ensures that the customer's needs are satisfied through a system's entire life cycle.

i. State the problem. Stating the problem is the most important systems engineering task. It entails identifying customers, understanding customer needs, establishing the need for change, discovering requirements and defining system functions.

ii. Investigate alternatives. Alternatives are investigated and evaluated based on performance, cost and risk.

iii. Model the system. Running models clarifies requirements, reveals bottlenecks and fragmented activities, reduces cost and exposes duplication of efforts.

iv. Integrate. Integration means designing interfaces and bringing system elements together so they work as a whole. This requires extensive communication and coordination.

v. Launch the system. Launching the system means running the system and producing outputs -- making the system do what it was intended to do.

vi. Assess performance. Performance is assessed using evaluation criteria, technical performance measures and measures -- measurement is the key.

vii. Re-evaluation. Re-evaluation should be a continual and iterative process with many parallel loops.

3.2 System Development Environment

System development environment gives a brief description of the various software components and tools used in this project. There are several number of tools used in achieving the completion of this project which are described below,

3.2.1 Eclipse

In computer programming, Eclipse is a multi-language software development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Fortran, Haskell, JavaScript, Perl, PHP, Pyt hon, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang [12]. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from Visual Age. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its abilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules [12]. Released under the terms of the Eclipse Public License, Eclipse is free and open source software.

3.2.2 Android

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like

language that utilizes Google-developed Java libraries, but does not support programs developed in native code. The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices.

3.2.2.1 Android architecture

The following diagram shows the major components of the Android operating system. Each section is described in more detail below.

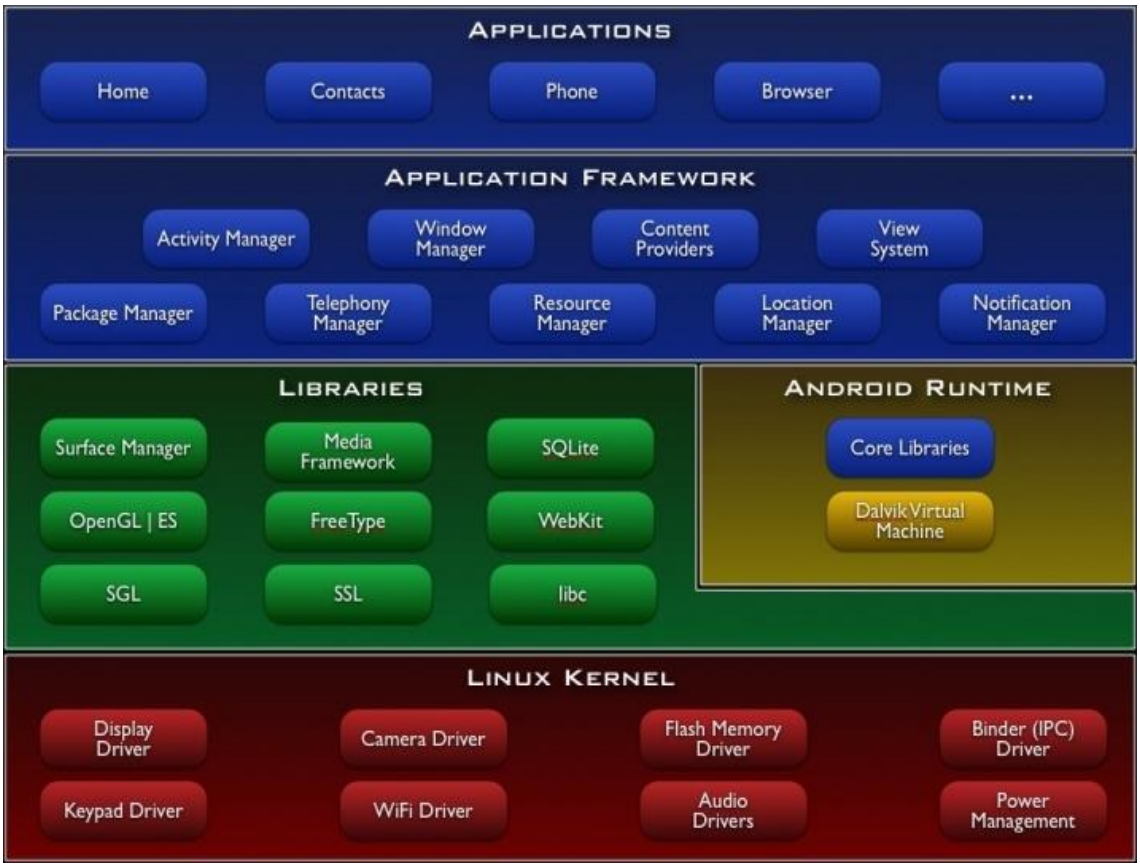


Figure 3.1 Architecture of android

The Figure 3.1 shows android architecture which comprises of several internal components and features.

Some of the features of android that have been utilized are given below,

- **Java Virtual Machine**

  Software written in Java can be compiled into Dalvik bytecodes and executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use, although not technically a standard Java Virtual Machine.

- **Media Support**

  Android will support advanced audio/video/still media formats such as MPEG-4, H.264, MP3, and AAC, AMR, JPEG, PNG, GIF.

- **Data Storage**

  SQLite is used for structured data storage. SQLite is a powerful and lightweight relational database engine available to all applications.

- **Connectivity**

  Android supports a wide variety of connectivity technologies including GSM, CDMA, Bluetooth, EDGE, EVDO, 3G and Wi-Fi.

- **Messaging**

  SMS, MMS, and XMPP are available forms of messaging including threaded text messaging.

- **Android runtime**

  At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main

issues. Android gives an integrated tool “dx”, which converts generated byte code from .jar to .dex file.

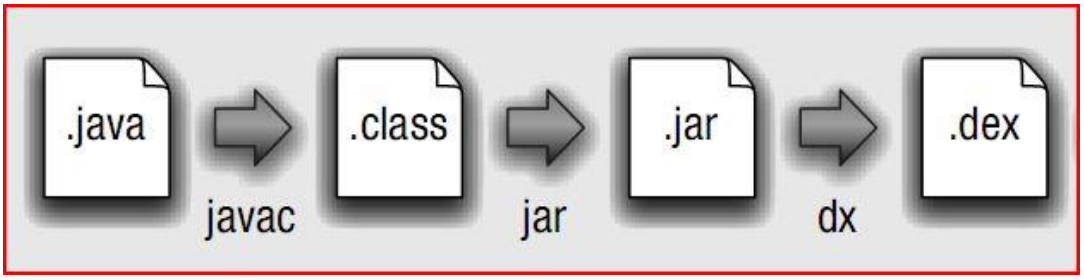


Figure 3.2 Conversion from .java to .dex file

The Figure 3.2 shows the convertion of file from the .java extension to the .dex file extension which is converted a tool called dx tool which is integrated from the andriod and converts byte code from .jar to .dex readable file format.

3.2.3 Cloud Computing

With the rapid development of processing and storage technologies and the success of the Internet, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. This technological trend has enabled the realization of a new computing model called cloud computing, in which resources (e.g., CPU and storage) are provided as general utilities that can be leased and released by users through the Internet in an on-demand fashion.

In a cloud computing environment, the traditional role of service provider is divided into two: the infrastructure providers who manage cloud platforms and lease resources according to a usage-based pricing model, and service providers, who rent resources from one or many infrastructure providers to serve the end users. Cloud computing provides several features shown as below.

No up-front investment: Cloud computing uses a pay as you go pricing model. It simply rents resources from the cloud according to its own needs and pay for the usage.

- Lowering operating cost: Resources in a cloud environment can be rapidly allocated and de-allocated on demand which provides huge savings.
- Highly scalable: Infrastructure providers pool large amount of resources from data centers and make them easily accessible.
- Easy access: Services hosted in the cloud are generally web based.
- Reducing business risks and maintenance expenses: By outsourcing the service infrastructure to the clouds, a service provider shifts its business risks (such as hardware failures) to infrastructure providers.

Cloud services are popular because they can reduce the cost and complexity of owning and operating computers and networks. Since cloud users do not have to invest in information technology infrastructure, purchase hardware, or buy software licenses, the benefits are low up-front costs, rapid return on investment, rapid deployment, customization, flexible use, and solutions that can make use of new innovations

CHAPTER 4

SYSTEM ANALYSIS

CHAPTER 4

SYSTEM ANALYSIS

4.1 Existing system

The study of the existing system helps for a new system to be developed. Currently there are no systems available for wireless multi-hop video streaming & video conference in android smart phones. If any of the available systems use the TCP/IP protocols for communication and also for transfer the frames from one system to another.

These systems also follow the stored and forwarded procedure for transfer the frames. Normally these systems can transmit and re-transmit the copy of the frames from the source or server. It uses the socket side programming for data transfer from server to the requested client. In these systems the IP address can be obtained by TCP/IP protocol. So the data are sent to that particular system. The sockets are used for data communication purpose.

Once the IP addresses of the systems are obtained then the data transmission starts. The receiver must have a network facility to the server and he has to register the IP address of the server. So the client and the server can know each other's. Once the transmission stops then, automatically the process stops. So the receiver cannot view the frames.

4.2 Limitations of the existing system

Since TCP/IP is connection oriented protocol, if there is any congestion during data transfer and data is lost, the data cannot be retransmitted since it is real time data.

4.3 Proposed system

In order to develop the new system, first have to understand the history of the existing system. This history will give the complete information about the existing system and also, this history will be the bright introduction for the new system. This information tells the advantages and the disadvantages of the existing one. So the problems can be identified. This helps for the development for the new system.

Video streaming is over multiple wireless hops between peer devices. Phones within communication range of each other automatically establish a wireless link creating a client mesh network (ad-hoc network of devices). Each phone in the client mesh network is able to produce/consume video and also acts as a relay to forward video to its next hop neighbors. Peer-to-peer video streaming from the cameras on smartphones to people nearby allows users to share what they see. Such streaming can be used in a variety of applications, in particular in various social network applications including sharing unforgettable moments with friends that can be multiple wireless hops away, cooperative field work (providing video sharing for teams distributed in a small area, e,g. teams of repairmen, and search and rescue teams in disaster areas), and support for health impaired persons including the elderly.

This project presents a wireless multi-hop video streaming application for the Android based mobile devices (including Android based tablet PCs). This application allows users capture live video using camera on mobile devices, and to share this video feed with people nearby using a free-of-charge wireless mesh network. Routing protocols can be installed to facilitate the multi-hop communication to go beyond a single hop.

An example application scenario is shown (Figure 1.1), in which person streams live video of a concert to friends nearby over the wireless mesh network of phones.

4.4 Advantages of proposed system

In proposed system we use adaptive bit rate streaming package to transmit live data which uses HTTP protocol. If data is lost in one route, we can receive data from other route. Hence it overcomes drawback of TCP.

- ✓ Smooth and high quality video streaming.

- ✓ Avoid playback interruption and achieve better smoothness and quality.

CHAPTER 5

SYSTEM REQUIREMENTS

CHAPTER 5

SYSTEM REQUIREMENTS

System requirement gives the overall requirement of the project both functionally and non functionally which are given below,

5.1 Functional requirement

5.1.1 User interface

In this module we create a user page using GUI, which will be the media to Connect user with the server and through which client can able to give request to the server and server can send the response to the client, through this module we can establish the communication between client and server. Before client creation we check the user credential by login page, we receive the username and password by the user and we will check in the database is that user have the credential or not to give request to the server. Here also we can add new user through user registration by taking all the important details like user's name, gender, username, password, address, email id, phone no from the user. A GUI is a graphical (rather than purely textual) user interface to a computer. In our project we are using GUI in the form of web pages that is called JSP(Java Server Pages).

5.1.2 Server Module

Mobile acts as server which captures video and broadcast it to the clients over wireless medium using protocols like UDP, RTTP and RTP.

5.1.3 Client Module

Client being a remote PC receives all the data sent by the server and displays it on its monitor. The data sent by the server is broadcasted on multiple clients (PC's).

5.2 Non Functional requirement

The hardware requirements needed for the implementation of the proposed system is listed below.

5.2.1 HARDWARE

- Android smart phones with Wi-Fi and camera
- Wi-Fi router
- P4 Computer

The software requirements needed for the implementation of the proposed system is listed below.

5.2.2 SOFTWARE

- Android 2.2(minimum)
- Android emulator 2.2 or higher
- Jdk 1.5 or higher
- Eclipse server 6
- Android sqlite database

CHAPTER 6

SYSTEM DESIGN

CHAPTER 6

SYSTEM DESIGN

6.1 High level design

Design is the first step in the development of any system or product. Design can be defined as "the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization". It involves four major steps they are:

i. Understanding how the system is working now

ii. Finding out what the system does now

iii. Understanding what the new system will do

iv. Understanding how the new system will work.

6.1.1 System architecture

The system architecture is the Conceptual model that defines the Structure, Behavior, and more View model of a System. The Figure 6.1.1 shows the system architecture of the system, it consists of several android operating system supportive smart phones and several other laptops. Here one android phone is considered as the server from which the video streaming commences and all the other devices are considered as the client. As shown in Figure 6.1.1 the first android phone is transmitting data to all the other clients.

There is special case in this scenario in which the client can also act like server i.e., the android phone which is receiving the video transmission can further act like a sever and re-transmit the same data to different other clients it is connected to.

This scenario is also shown in Figure 6.1 where the client android phone which is receiving data is also transmitting the data to other phones.

Figure 6.1 System architecture

6.1.2 Data flow diagram

A Data Flow Diagram (DFD) is a logical model of the system. The model does not depend on the hardware, software and data structures of file organization. It tends to be easy for even non-technical users to understand and thus serves as an excellent communication tool.

DFD can be used to suggest automatic boundaries for proposed system at pa very high level; the entire system is shown as a single logical process clearly identifying the sources and destination of data. This is often referred to as zero level DFD.

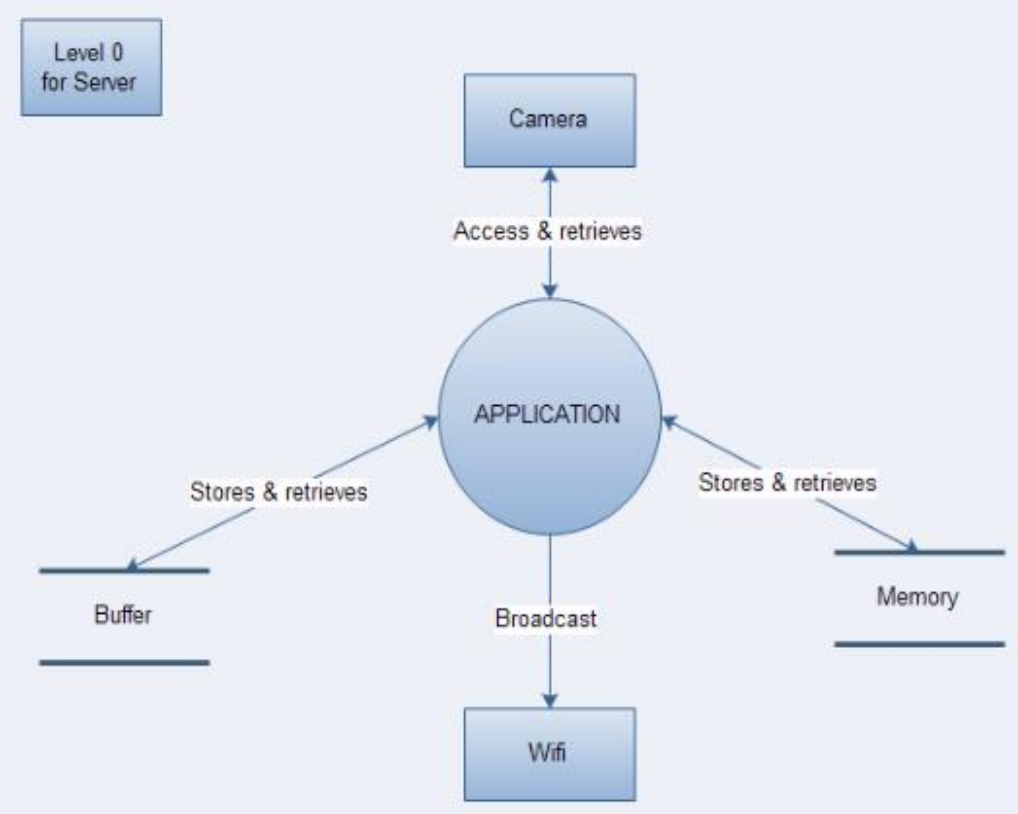


Figure 6.2 Level 0 DFD for server

The Figure 6.2 shows the high level or level 0 data flow diagram for the server which shows that the application in android phone gets data from the camera stores it in memory and then broadcast it to other different clients using the Wi-Fi connection.

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an Information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the Data Visualization of Data Processing (structured design).

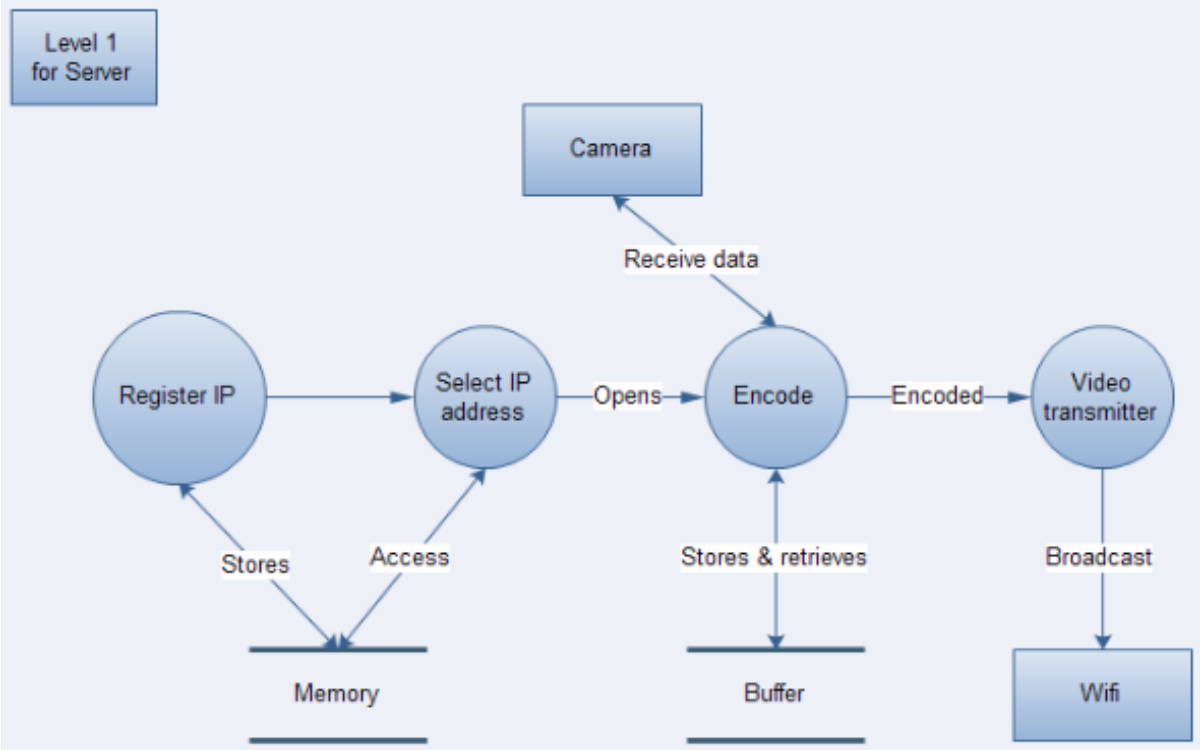


Figure 6.3 Level 1 DFD for server

The Figure 6.3 shows the low level or level 1 DFD for server which indicate all the process that takes place in the server side of the application. At first the server or the sender receives the data from the camera in the form of streams which is stored in buffer for a while for the purpose of transmission later this data is encoded into stream of data and sent to the video transmitter. In video transmitter the encoded data will be broadcasted to all the register clients in the server side.

On the other hand the registration of the clients in the server side takes place where different clients IP address are registered into the server side so that when the video transmission takes place from the server side the video transmission is sent only to the registered clients for security purpose. These registered clients information is stored in the memory of the server.

Also the server should be able to transmit the data to the client who are unregistered earlier i.e., the server can transmit the same stream of data to different other unregistered client in real time. This is achieved, when an unregistered client sends a message to the server with the IP address of the new client in the format (ip 192.168.1.3) then the server automatically reads the message and the message content which is having the IP address of the new client that is to be registered. It automatically registers the clients IP address which is there in the message & starts transmitting real time video to newly added client.

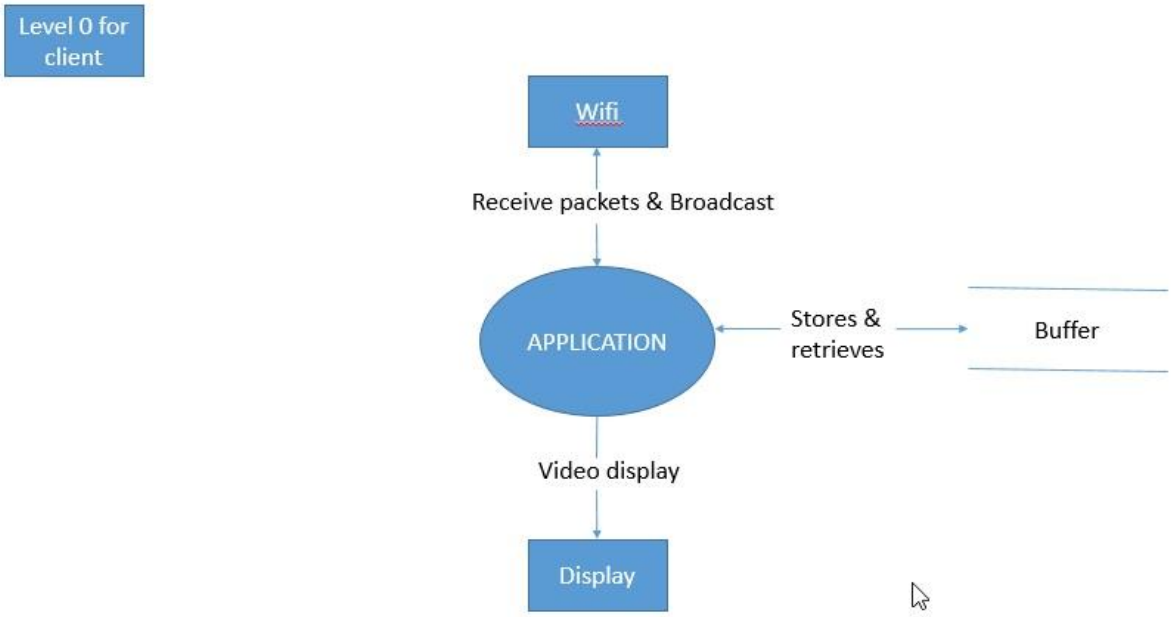


Figure 6.4 Level 0 DFD for clients

The Figure 6.4 shows the level 0 or high level data flow diagram for the clients in which an application in the receiver is responsible for receiving all the data broadcasted from the server. It receives the data and also can perform multi hop transmission i.e., is able to re-transmit the same data to other different clients.

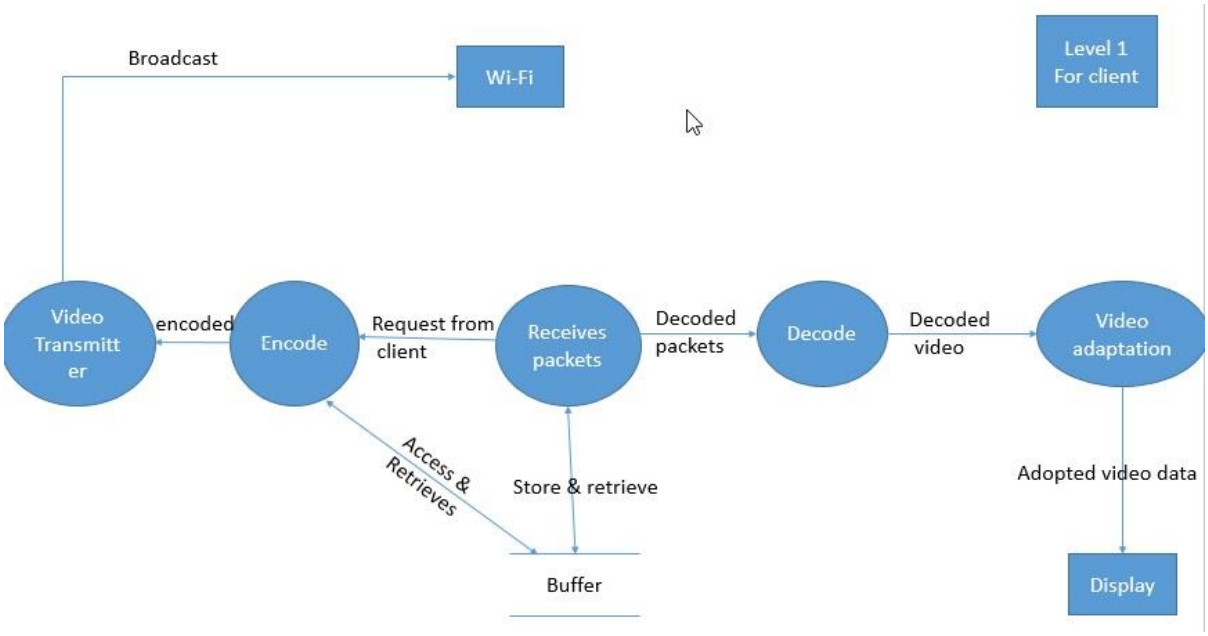


Figure 6.5 Level 1DFD for client

The Figure 6.5 shows the level 1 or low level detailed data flow diagram for the client. The client plays a major role in this application because client is responsible for two main characteristics, they are:

- Client should be able to receive stream of data from the server.
- Client should be able to capture the image of streaming real time video.
-
- Client should be able to stream the same stream of data to different number of other clients which is received from the server.

So as per the major roles that should be performed by the client side are achieved and it is depicted in this low level data flow diagram for clients. First and foremost the client should be able to receive the stream of data sent from the server, which is achieved by connecting to the server through Wi-Fi and receiving the data. This data will be encoded when sent from the server side which is decoded in the decode part of the process and sent to the video adaptation for the display of the data.

Finally the client should be able to re-transmit the data which is received by the client to different other clients i.e., the client can transmit the same stream of data that is been sent from the server to different other client in real time. This is achieved, when an unregistered client sends a message to the client with the IP address of the new client in the format (IP 192.168.1.3) then the client automatically reads the message and the message content which is having the IP address of the new client that is to be registered. It automatically registers the clients IP address that is there in the message and starts transmitting the real time video sent from the server to the newly added client.

6.2 Low level design

Low level design (LLD) is the design process of a system in which it consists of the flowchart, which depicts the entire flow of the system in the pictorial representation also the algorithm that are used for efficient data transmission without data loss in its transmission from one end to the other end.

HLD or high level design of the system consists of the system architecture, data flow diagrams and class diagrams which gives a clear representation of the system whereas the LLD is the basic information of the system starting from the flowchart and algorithms that are used in the efficient performance of the system.

6.2.1 Flow chart

A flowchart is a type of diagram that represents an Algorithm or Process(Science), showing the steps as boxes of various kinds, and their order by connecting them with arrows. This is diagrammatic Knowledge Reputation and Reasoning of solution to a given problem solving. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. Flowcharts are used in designing and documenting complex processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help the viewer to understand a process, and perhaps also find flaws, bottlenecks, and other less-obvious features within it.

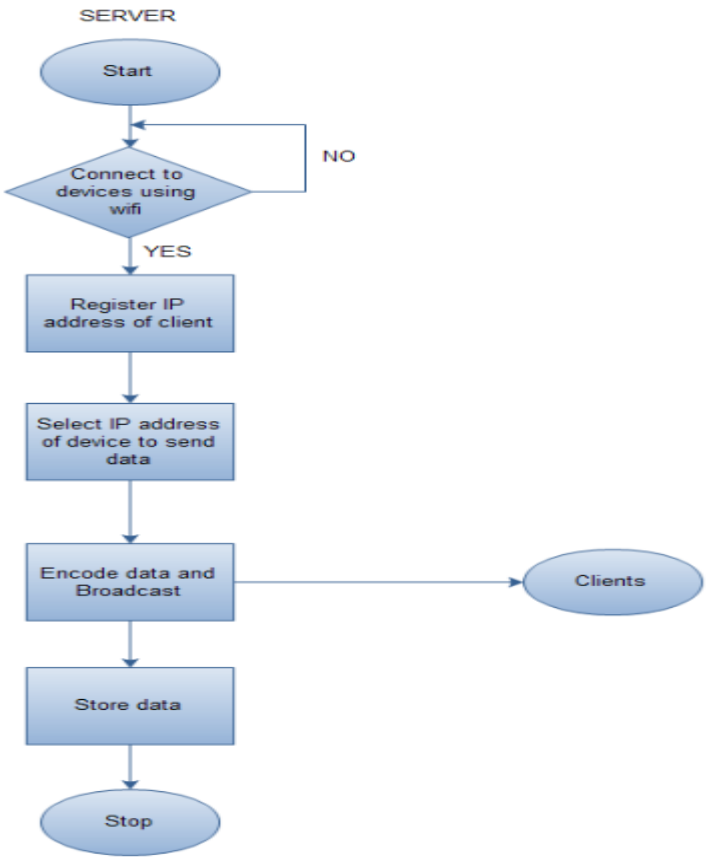


Figure 6.6 Flowchart for the server module

The Figure 6.6 shows the flowchart for the server module where it shows that server is responsible for capturing the video and transmitting the same to the registered clients. The process of transmission takes place by broadcasting the stream of data and also the server stores all the data that is received from the camera.

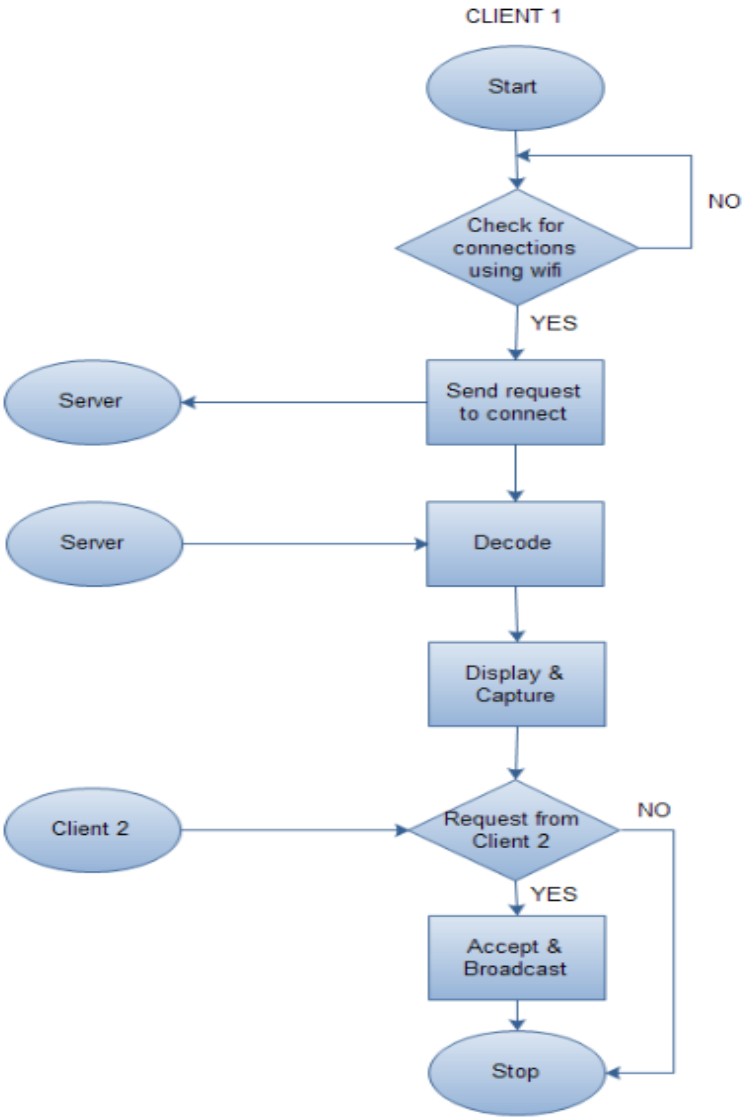


Figure 6.7 Flowchart for the client module

The Figure 6.7 shows the low level detailed flowchart of the client module. It shows that client can perform several other functions along with receiving the stream of data.

6.2.2 Algorithm

Algorithm is a step-by-step procedure for calculations. It is used for Calculation, Data Processing, and Automated Resoning. Video encoding and decoding is an important aspect of any video streaming application. There are many ways by which a video can be encoded or decoded. We briefly describe two widely used video coding techniques which are implemented in our application.

i. Intraframe encoding, is the simplest form of encoding. It treats every frame as an individual image to encode. This method is resilient against lost frames due to each frame having enough information to create an entire image.

ii. Interframe encoding, uses two types of frames, i.e., the key frames and predicted frames, for better compression ratio. The key frame contains complete information to create an image, whereas the predicted frames only contain the differences between frames thus previous frames are required for their successful decoding.

CHAPTER 7

SYSTEM IMPLEMENTATION

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 Routing

To stream a video across multiple hops, a multi-hop path between the sender and receiver has to be found. A routing protocol is therefore required to establish and manage connectivity within the mobile ad hoc network. The peer-to-peer video streaming application relies on Real-Time Transport Protocol (RTP).

Real-time Transport Protocol (RTP) defines a standardized packet format for delivering video over IP networks. RTP is used extensively in communication and entertainment systems that involve streaming media, such as telephony, video teleconference applications, television services and web-based push-to-talk features. RTP is used in conjunction with the RTP Control Protocol (RTCP).

RTP is originated and received on even port numbers and the associated RTCP communication uses the next higher odd port number. RTP is one of the technical foundations of Voice over IP and in this context is often used in conjunction with a signaling protocol which assists in setting up connections across the network. The main processes are identified as under:

- Adhoc network /Wi-Fi setting need to be done with IP/Port
- We need to establish the connection between android mobile user and Receiver User
- When application initiate, user has to launch request and start streaming
- It will verify camera is connected and then after process Raw Video
- It Uses Codec Encoder/Decoder for this process
- Once the streaming start, video will process in encoding with the help of MJPEG encoder
- Once Encoding is done and it will go in network with the help of RTP
- Receiver will receive the video and display on their device after the decoding.

7.2 Procedure for encoding and decoding

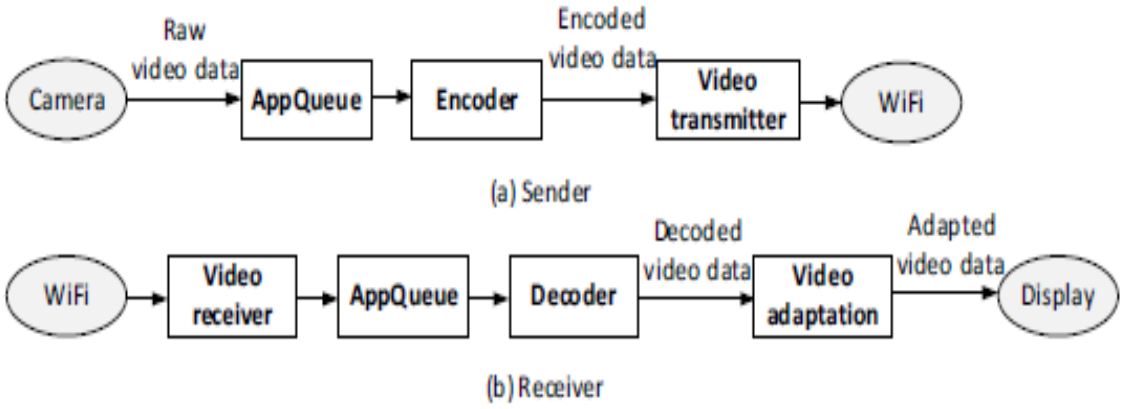


Figure 7.2.1 Procedure for encoding & decoding

The Figure 6.1 shows the general procedure for encoding and decoding technique used for multi hop video streaming, in which Routing is an important aspect of peer-to-peer video streaming as it discovers the path from the sender to the receiver. In the current version of the proposed application, the video content providers use periodic broadcast advertisements over the multi-hop network to advertise to potential receivers whenever they are ready to stream a video. Therefore, video content consumers within the network will receive information about who is offering video contents and the content description. This approach to content provider discovery can be extended in the future to support a publish/subscribe mechanism. Clients could subscribe for video content based on their interest and needs and receive a notification when a video meeting their interest is available.

Video frames need to be encoded before sending on to network. Encoding can significantly reduce the size of individual video frames, therefore minimising the bandwidth required for the streaming application. As shown in Figure 6.1, when starting the streaming application raw video data is retrieved from the camera and stored in the application queue (for buffering purpose). This raw data is then passed to the encoder, which encodes the data using the selected codec and encoding technique.

The encoded video frame is then transmitted over-the-air by the Wi-Fi module. At the receiver, when an encoded video frame arrives it is buffered and sent to the decoder. Before being displayed on the screen, the video frames may require adaptation to the hardware specifications (e.g., screen resolution).

7.3 Pseudo code

Server

```
INITIALIZE Wi-Fi Connectivity

IF IP address is already  registered
    Select the IP addresses
    INITIALIZE Connection with Client
    Capture the video in NV21 format
    Broadcast the video
ELSE IF IP addresses is not registered
    READ IP address from the request message
    Register the IP address
    Select the IP address
    INITIALIZE Connection with Client
    Capture the video in NV21 format
    Broadcast the video
END IF
```

Client

INITIALIZE Wi-Fi Connectivity

SELECT the Broadcasting server

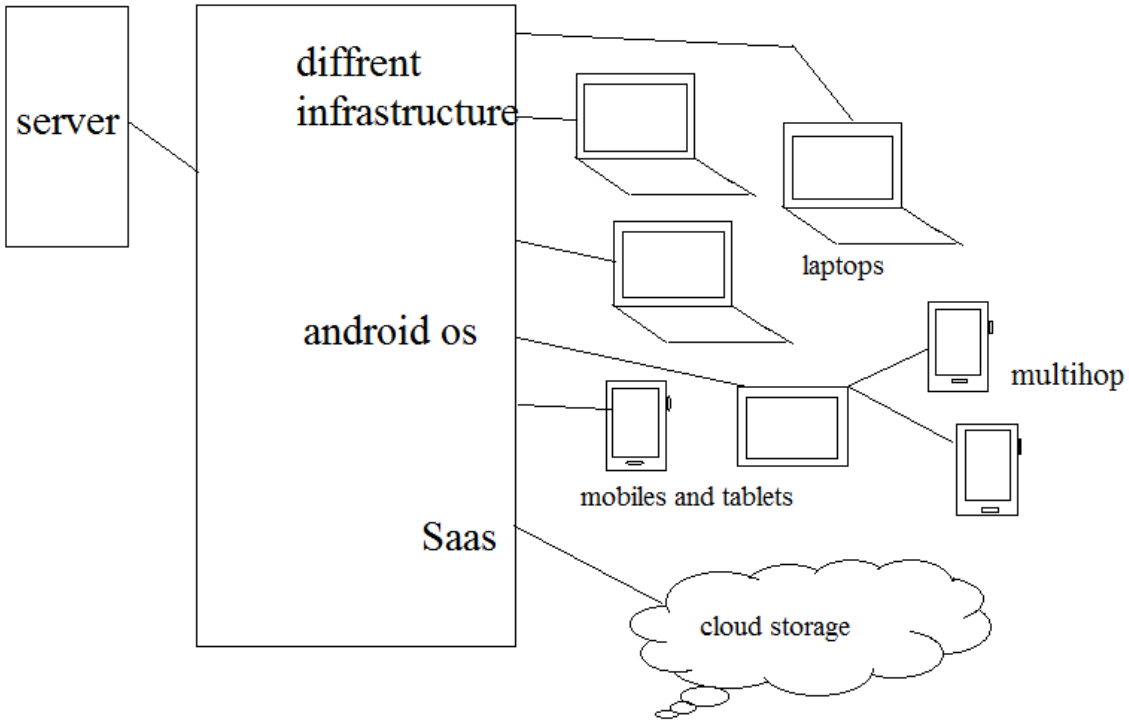
Establish connection with server

Request for Video

SEND Request to connect message containing the IP address of the Client

Wait for response from server.

7.4 MODULES



7.4 Diagram of entire module

7.4.1 Server Module:

In this module many laptops, smart phones are connected in a mesh like network.

The protocols used are :

- RTP - Real Time Protocol
- RTTP - Real Time Transfer Protocol
- UDP - Unified Datagram Protocol

7.4.2 Client Module:

In this module which the major module which consists of three different combination of clients as explained below:

- A client mesh module is created.
- For multi hop streaming it obtains IP address of other connected devices and provides streaming capability within the client mesh.
- Smart phones, tablets, laptops are used as client devices for video sharing.

7.4.3 Bandwidth Sharing Module :

All the devices connected use a bandwidth sharing facility that transfer the frames in a multi hop fashion. Each device connects in a client mesh network through Ethernet connectivity by obtaining the IP address.The advantages of this process is :

- One can transfer video frames through bandwidth sharing with other nearby devices.
- Reduces the congestion.
- The data lost in the transmission process can be retrieved and used for retransmission.
- The buffer space can be reduced.
- Time consumption is less.
- It is Distortion resistant.

7.4.4 Infrastructure Module/ Cloud Module:

The main objective of this module is to provide the following services:

- IaaS(Infrastructure as a service): It provides a virtual interface consisting of virtual machines in which live streaming data can be migrated from one device to another over a wifi connectivity.
- PaaS(Platform as a Service): It provides a platform upon which infrastructure is built and also on which applications can be developed to provide services.
- SaaS(Software as a Service): It provides application services for transferring video streaming data upon cloud from one android device to another.

CHAPTER 8

SYSTEM TESTING

CHAPTER 8

SYSTEM TESTING

8.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

- **Functional test:** Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

- **System Test:** System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **Performance Test:** The Performance test ensures that the output be produced within the time limits and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

8.1.1 Unit test considerations

The module 'interface' is tested to ensure that information properly flows into and out of the program unit under test. The 'local data structures' are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithms execution.

'Boundary Conditions' are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All 'independent paths' through the control structures are exercised to ensure that all statements in a module have been executed at least once. Finally, all 'error-handling paths' are tested.

8.1.2 Unit test procedures

Unit testing is considered an equivalent to the coding step. After the source level code has been developed, reviewed and verified for correct syntax, unit test case design begins since a module is not a stand alone program, 'driver' and/or 'stub' S/W must be developed for each unit test. In most applications, a driver is nothing more than a main program that accepts test case data, passes such data to the module to be tested, and prints the relevant results.

The stubs serve to replace modules that are subordinates called by the modules to be tested. A stub or a dummy stub or a dummy subprogram uses the subordinate modules interface, may do minimal data manipulation, prints verification of entry, and returns. The drivers and scrubs represent overhead i.e., both are S/W that must be written but that is not delivered with the final S/W product. If the drivers and the stub are kept simple, then the overhead is low. The Unit Test is carried out in this project, and is found successful. The data is flowing correctly to all part of the project.

8.2 Integration testing

Integration testing is the phase of software testing in which individual software modules are combined and tested as a group. The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages, In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit-tested modules and build a program structure that has been dictated by design.

8.2.1 Top-down integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level interact without error.

Integration testing for Server Synchronization:

- Testing the IP Address for to communicate with the other Nodes
- Check the Route status in the Cache Table after the status information is received by the Node
- The Messages are displayed throughout the end of the application

8.3 System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified Requirements. System testing falls within the scope of Black box Testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed Integration Testing and also the software system itself integrated with any applicable hardware system. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

The process of performing a variety of test on a system is to explore functionality or to identify problem. System testing is usually required before and after a system is put in place. A series of systematic procedure are referred to while testing is being performed. These procedures tell the tester how the system should perform and where common mistake may be found. Testers usually try to "break the system" by entering data that may cause the system to malfunction or return incorrect information. For example, a tester may put in a city in a search engine designed to only accept state, to see how the system will respond to the incorrect input.

System testing is performed on the entire system in the context of a Functional requirement Specification (FRS) and a Requirement analysis Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software or hardware requirements specification.

8.4 Test cases

Table 8.4.1

Test case id	1
Test case description	start the app
Expected result	app should respond and run
Actual result	responding and running
result	pass

Table 8.4.2

Test case id	2
Test case description	accepts and registers ip
Expected result	registered ip successfully
Actual result	registered
result	pass

Table 8.4.3

Test case id	3
Test case description	video streaming
Expected result	streaming successfully
Actual result	streams video (1 sec delay)
result	pass

Table 8.4.4

Test case id	4
Test case description	video recieving
Expected result	receive and display video
Actual result	successfully display of video
result	pass

Table 8.4.5

Test case id	5
Test case description	running multiple apps
Expected result	receive and display video
Actual result	blank screen (blind error)
result	Fail

Table 8.4.6

Test case id	6
Test case description	running the app for long
Expected result	receive and display video
Actual result	app crashes by displaying an error
result	Fail

CONCLUSION

An attempt to present a wireless multi-hop video streaming application for the Android based mobile devices (including Android based tablet PCs) and laptops have been achieved.

This application allows users to capture live video using camera on mobile devices, and to share this video feed with people nearby using a free-of-charge wireless mesh network. This application can be useful in many real time events and applications for an enhanced user support.

In this project, we presented a wireless multi-hop video streaming application for Android mobile phones.

This application allows users to capture live video feeds using the mobile phone camera and share these feeds with people who might be multiple wireless hops away.

The video feeds are shared using wireless client mesh network (ad hoc network) established between mobile phones.

Thus the video streaming does not rely on a traditional network infrastructure (such as the cellular), therefore it is a free-of-charge communication.

Such a multi-hop video streaming can be used in a variety of application domains including social networking.

FUTURE ENHANCEMENT

The current project allows user to store the image of the real time video in the memory which can further be improved to store the real time video stream itself in the memory by increasing the memory size of the device.

Also the time delay between the packets that is been streamed can also be reduced for further more appropriate data delivery to the clients.

For future work we are planning to extend the evaluation test-bed to study the application performance within a larger network.

We are also considering developing a richer user interface with additional features, such as implementing multicast over multiple hops and allowing users to record video contents on local SD cards while streaming or forwarding.

REFERENCES

[1] Cooperative Video Streaming on Smartphones (Seferoglu. H., Keller. L., Cici. B., Le. A., Markopoulou. A.,) 2010

[2] Live Video Streaming from Android-Enabled Devices to Web Browsers (Justin M. Bailey, 2012)

[3] Real-time Mobile Learning using Mobile Live Video Streaming (Majumder, M., and Biswas ,D. 2012)

[4] Cloud-Based Mobile Video Streaming Techniques (Saurabh Goel. 2013)

[5] The Performance of Video Streaming Over Wireless-N (Ibrahim, N.,Ali, A., Razak, A., and Azhar, M. 2012)

[6] Rate Adaptation Strategy for Video Streaming over Multiple Wireless Access Networks (Xing, M., Xiang, S and Cai, L. 2012)

[7] Streaming Video from Smart Phones: a Feasibility Study(Farkas, L. and Aczel, K. 2008)

[8] VIDEO STREAMING OVER WIRELESS NETWORKS (Zhu, X and Girod, B. 2010)

[9] TCP Streaming for Low-DelayWireless Video (Wong, C., Fung, W., Tang, C. and Chan,S. 2005)

[10] ITU. Statistics on global mobile subscriptions .http://www.itu.int/newsroom/press releases/2010/06.html.

[11] Qik. www.qik.com.

[12] Android development sdk. http://developer.android.com.

APPENDIX A

i. Android SDK

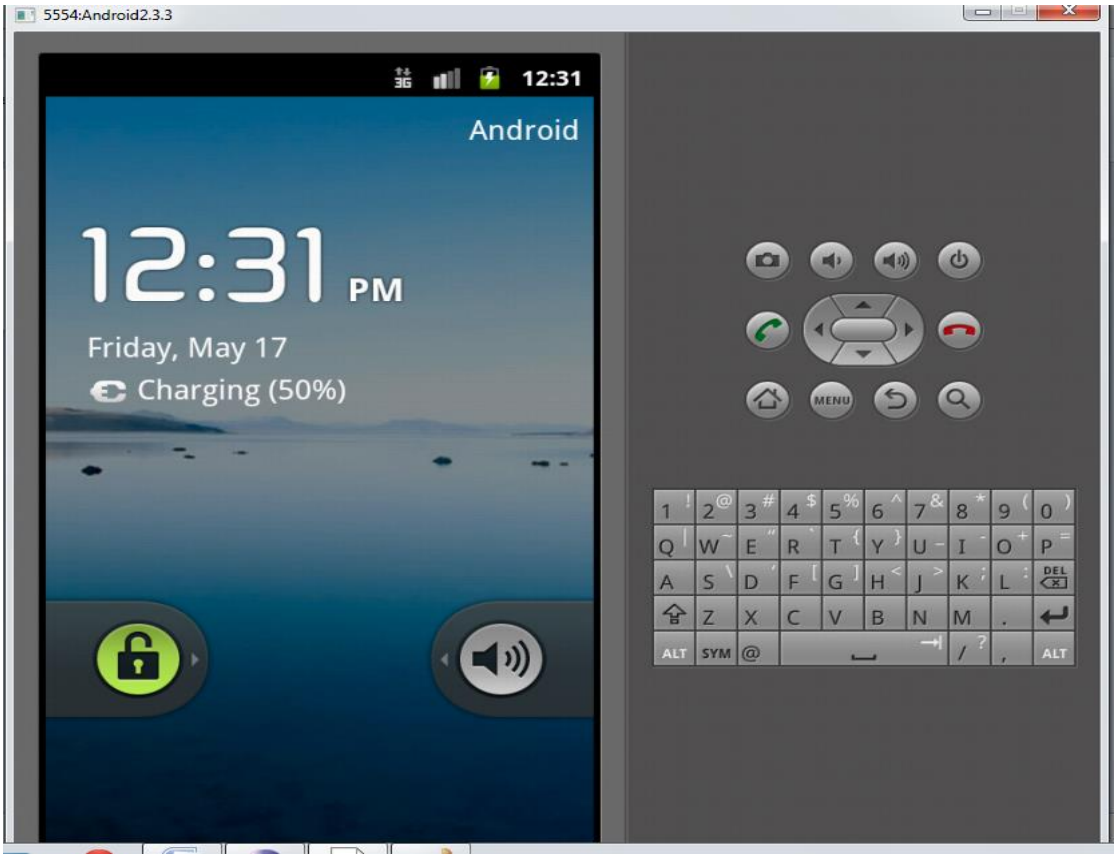


Figure 9.1 Home screen of android SDK

The Figure 9.1 shows the home screen of the android software development kit, in which it shows the virtual function of the android smartphone i.e., here one can work in eclipse as the android smartphone where it allows to work on the platform similar to the working in android smartphones. This helps to test the developed application it functionality of the developed project.

ii. Registering IP address at server side

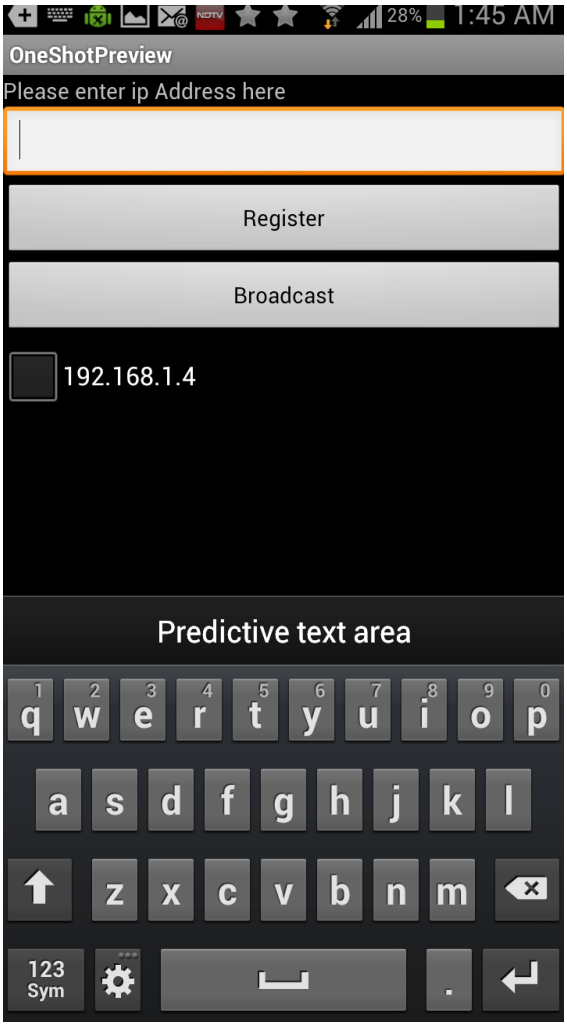


Figure 9.2 Registering IP at the server

The Figure 9.2 shows that the new IP address of a client can be registered in the server side so that when broadcasting is taking place the IP addredd can be selected for broadcast and video streaming can be done. So the new client IP address is registered as shown in the Figure 9.2 as 192.168.1.4 where, the entered IP address is registerd in server side and can be used for further transmission.

iii. Selecting IP Address

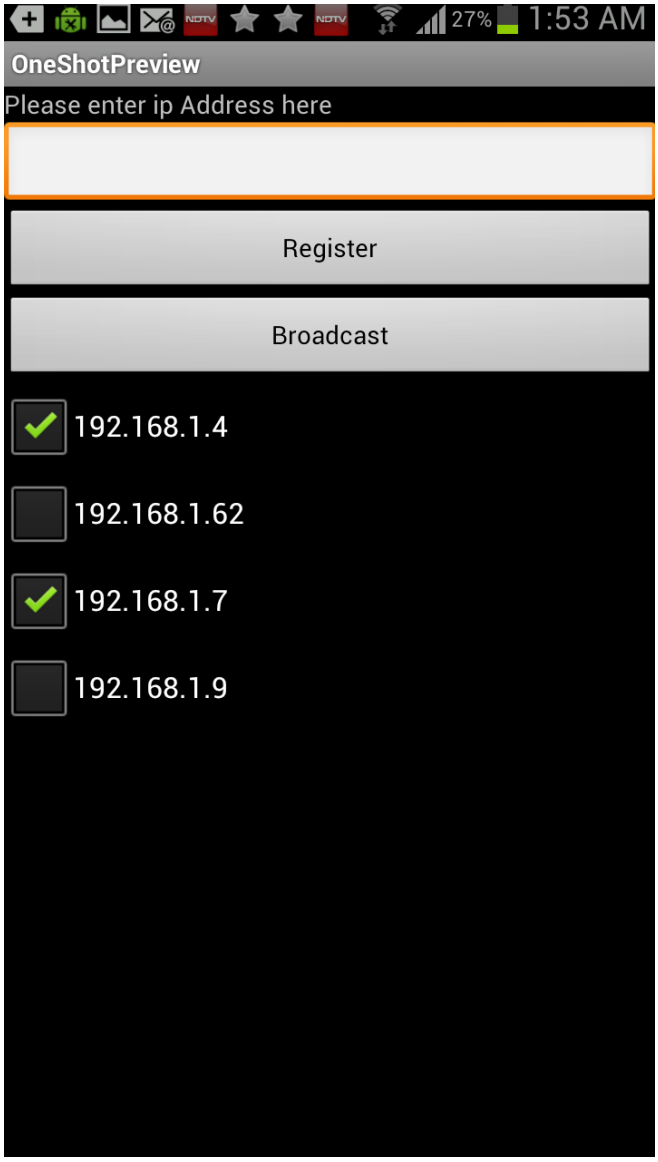


Figure 9.3 Selecting IP at the server

The Figure 9.3 shows that there are several IP addresses registerd in the server side which are viewed and selected for video transmission to occur. Here only wanted IP address for transmission can be selected.

iv. Broadcasting Video to both Mobile and PC

Figure 9.4 Broadcasting video

The Figure 9.4 shows that the server is broadcasting video to both the clients i.e., mobile and laptop. The one from which the snapshot has been taken is the server which is broadcasting the video to the selected clients hence one can see that the same video has been transmitting to both mobile and laptop.

v. Registering IP address at client side

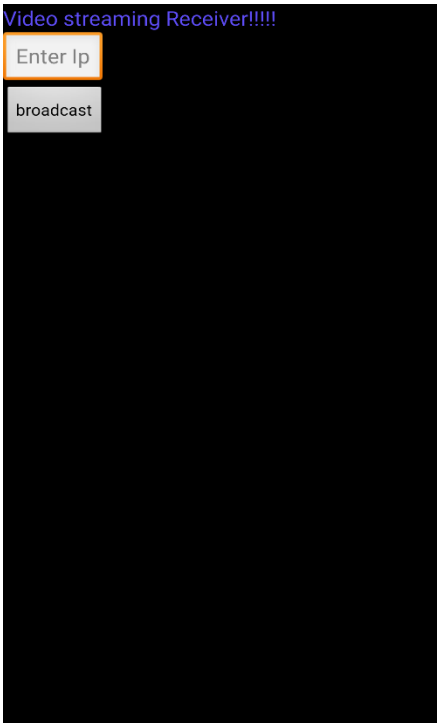


Figure 9.5 Registering at client

The Figure 9.5 shows the new capability of this project i.e., even the client is able to re-transmit the video that is being received, at the same time. At first the client should register the IP address of the other device for which the retransmission should take place, so after registering the client IP address broadcast button is pressed to re-transmit the video that is being received by the client. One more button is provided for the client called capture which is used to capture the image of the live video streaming which is stored in the memory and can be accessed later for further use.

vi. Re-transmission from the client

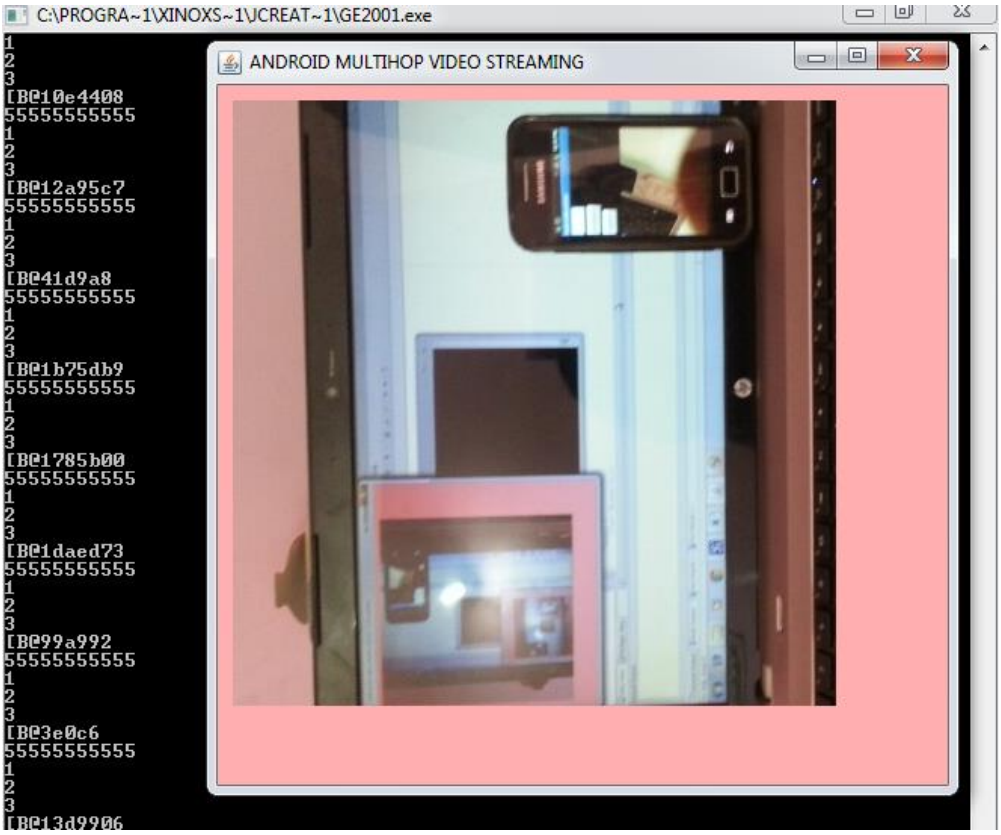


Figure 9.6 Re-transmition from the client

The Figure 9.6 shows the video re-transmission from the client side i.e., the client who is receiving the video transmission from the server is in-turn re-transmitting the same video to different other clients.

APPENDIX B

Configuration

Android is a large and fast-growing segment of the mobile phone market. With potentially android infograpics and multiple Android App Stores popping up (market was recently revamped, and now other markets are coming online, like mobile app). Installing the Android Software Development Kit (Android SDK), installing and configuring the Eclipse IDE for Android development, and choosing and installing Android Virtual Devices (AVDs) to emulate the Android environment right on your local computer. After following these steps, it will be ready to create your first Android application.

i. Download the Android Software Development Kit (SDK)

The very first step is to download the Android Software Development Kit (SDK) that will let to emulate Android on local computer. From the sdk, make sure to choose the version that is correct for the operating system.

ii. Download Eclipse IDE for Java Developers

It's better to choose Eclipse as Integrated Development Environment (IDE). Eclipse can be suitably adapted for Android development since you can get plugins to help with creating your Android project, launching the Android emulator, and preparing Android application for the Android Market. It is not an ideal IDE, but the pros outweigh the cons for Android Development.

From the Eclipse, choose the "Eclipse IDE" for Java Developers. Make sure to get the correct version for your operating system. After your Eclipse IDE download is complete unzip and move to a permanent folder.

iii. Install the Android Development Tools (ADT) plugin for Eclipse

Next, use Eclipse to install the Android Development Tools (ADT) using Eclipse's built-in plug-in system. From within Eclipse:

- Choose "Help" > "Install New Software…."
- Click the "Add…" button and create a new entry:

  Name: "Android ADT"

  Location: "https://dl-ssl.google.com/android/eclipse/"
- Check all the boxes to install all the tools
- Just keep clicking "I agree", "Next", "Yes", etc. until it asks to restart
- Go ahead and restart Eclipse when prompted.

iv. Connect Android SDK with Eclipse IDE

This next step connects the Android SDK from Step #1 to the Eclipse IDE from Step #2. In Step #3. From within Eclipse:

- Click on the "Eclipse" menu and choose "Preferences"
- Click on "Android" heading in the menu-tree to open our Android Eclipse preferences
- Click the "Browse…" button to the right of the "SDK Location" box
- Enter the location of your Android SDK (the $ANDROID path from Step #1)

v. Connect Android SDK with Eclipse IDE

Need to download the Software Development Kits (SDKs) for the Android versions that needs to support. To do this, use the Eclipse IDE + Android ADT that is installed in Step #3. From within Eclipse:

- Click on "Window" then "Android SDK and AVD Manager"
- In "Available packages", select the platforms you want to support. You can either choose all, or pick-and-choose what you want to develop for. For example, 2.1, 2.2, and 2.3.3 are all I care about8, so I am using API 7, 8, and 10. In the "Android Repository" package, I checked the boxes next to:
- Android SDK Platform-tools, revision 3
- SDK Platform Android 2.3.3, API 10, revision 1
- SDK Platform Android 2.2, API 8, revision 1
- SDK Platform Android 2.1, API 7, revision 1
- Samples for SDK API 10, revision 1
- Samples for SDK API 8, revision 1
- Samples for SDK API 7, revision 1
- Android Compatibility package, revision 1
- Choose "Install Selected", then the "Accept All" radio button, then "Install". This may take a while. If it seems like download has paused, check for any confirmation dialogs that need to click "Accept".

vi. Create Your Android Virtual Devices (AVDs)

Last but not least, need to create Android Virtual Devices (AVDs) that will be Android Emulators for running and testing the Android applications on local computer. In the same "Android SDK and AVD Manager" from Step #7, choose "Virtual Devices" on the left and create "New…" ones. Create AVDs to represent different Android versions that need to be tested, as well as different hardware specs and testing that users are likely to be using.

User manual

Creating a New Project

To create a new project:

i. Select **File** > **New** > **Project**.

ii. Select **Android** > **Android Project**, and click **Next**.

iii. Select the contents for the project:

- Enter a *Project Name*. This will be the name of the folder where your project is created.
- Under Contents, select **Create new project in workspace**. Select your project workspace location.
- Under Target, select an Android target to be used as the project's Build Target. The Build Target specifies which Android platform you'd like your application built against.
- Under Properties, fill this all necessary fields.
- Enter an *Application name*. This is the human-readable title for your application- the name that will appear on the Android device.
- Enter a *Package name*. This is the package namespace (following the same rules as for packages in the Java programming language) where all your source code will reside.

- Select *Create Activity* (optional, of course, but common) and enter a name for your main Activity class.

- Enter a *Min SDK Version*. This is an integer that indicates the minimum API Level required to properly run your application.

iv. Click Finish.

Run the application

Before we run our application on the Android Emulator, one **must** create an Android Virtual Device (AVD). An AVD is a configuration that specifies the Android platform to be used on the emulator.

i. Creating an AVD

Here's the basic procedure to create an AVD:

- Open a command-line (eg.,"Command Prompt" application on Windows, or "Terminal" on Mac/Linux) and navigate to our SDK package's tools/ directory.
- First, we need to select a Deployment Target. To view available targets, execute: This will output a list of available Android targets.
- Create a new AVD using your selected Deployment Target. Execute: That's it; our AVD is ready.

ii. Run the Application

- Choose any of the existing android emulators.
- Install the apk file on the system.
- Click and run the application.
- Application runs on the virtual platform.