Development of an OTA (Over the Air) Mobile Learning Telepresence Platform

BY
Kyle Galvin
HBSc Computer Science, Lakehead University, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MSC COMPUTER SCIENCE
in the department of Computer Science

Supervisory Committee

Development of an OTA (Over the Air) Mobile Learning Telepresence Platform

by

Kyle Galvin
HBSc Computer Science, Lakehead University, 2013

Supervisory Committee

Jinan Fiaidhi and Sabah Mohammed
Supervisor
Co-Supervisor or Departmental Member
Departmental Member
Outside Member

Contents

Chapter 1

Lightweight Telepresence Technologies

Microprocessors have shaped the world over the last century. Reducing in size over time exponentially, we are now able to achieve things that would have been unimaginable in the past. We can squeeze more bits per volume, transport more information and crunch more data each second than ever before. With this explosion of portability and connectivity comes a renaissance of technological growth that is unfolding before our eyes.

Density of information and computation as well as the speed of communication are at the core of modern digital technology, yet focusing on these features displays a very hands-on 'white box' approach. There is also much to be learned with respect to the interaction between digital components and their environments, which could be considered more of an external 'black box' style description. The interface and sensors a device supplies for others to interact with is just as important as the computational and communicative abilities the device has internally to process the environment around it.

By extrapolating on current computational growth trends, we can easily imagine the capabilities we will soon wield while developing applications to improve our everyday lives. By studying these applications (both mundane and whimsical alike) we are likely to find many exciting ideas which are attainable much more immediately than they first appeared as well as many which are just around the bend.

Arthur C. Clarke once wrote "Any sufficiently advanced technology is indistinguishable from magic", and I am inclined to agree. Indeed many amazing discoveries can find roots in sci-fi and futuristic predictions which push the boundaries of our collective knowledge and explore the potential and logical conclusions of current technological progress. One of the most provocative ideas to emerge from these advancements is immersive telepres-

ence.

[CFB+ar]

> "Telepresence systems provide a human operator with the feeling of actual presence in a remote environment, the target environment. The feeling of presence is achieved by visual and acoustic sensory information recorded from the target environment and presented to the user on an immersive display."

[AH11]

1.1 Emerging Mobile Technologies

As microchip density increases, so does the mobility of computational and processing devices. While PDA and hand held gaming devices have been around for decades, the advance of cellular networks which allow for on-the-go personal telecommunications and widely dispersed access to internet services has really driven the shape and design of the current generation of mobile devices.

Smart phones and telecommunications aren't the only technology in this arena, but they are certainly the largest and most influential. Other devices to consider when discussing telepresence devices are lightweight microprocessors and system on a chip designs. These devices can allow industry and hobbiests alike to create a wide array of telepresence hardware that is capable of interacting with the environment around it on another's behalf. In this case, we are now less bound by strict computational limits and are now merely bound by the sensors, motors, and analog/digital conversions available to read from (and interact with) the environment around us.

When we combine our new-found freedom to invent any sort of sensory device with our fully connected and always online 'internet of things', we can begin to explore and create all sorts of ideas that were inaccessible to the real-world and thus bound to the realm of fiction, futurism, and sci-fi.

1.2 Telepresence & Real-Time Communications

The need to provide high fidelity information in real-time is the defining challenge of telepresence systems. Current telepresence infrastructure comes with restrictive limitations due to network latency and available throughput.

These challenges have been met with several techniques such as lowering the fidelity rate transmitted, compressing the data in transit, and prioritizing which data is most relevant to the situation.

1.2.1 Audio/Video Compression

Within the context of mobile devices we need to consider not only the computational complexity and bandwidth consumption of video streaming, but the power consumption of the encoder as well. [AKK09]

1.2.2 Cellular Network Bandwidth Flow & Optimization

Network traffic can be prioritized using QoS (Quality of Service) classification. By prioritizing traffic types into real-time and non-real-time categories we can decrease the average latency on timing-critical services. By extending these concepts and including cooperative game theory strategies (by exploring Nash Bargaining Systems). A game theory strategy can be broken down into three components: Players, Strategies, and Interactions (or game utilities). If we can determine a metric for success in the context of each player, than we can create strategies which each player can use to interact with the system in a way which optimizes the collective success of all the players. Applied to bandwidth optimization, each network device is a player which utilizes a strategy for sharing the limited resources of the network.

The trend among wireless networks is an increased number of cells over smaller and smaller areas serving users. By reducing this cell size, we introduce an increased number of hand offs when the user is mobile. When a user passes from the range of one operator to the next, this hand-off should not interrupt the user's communication. Because of this, our QoS strategies should include a percentage of bandwidth reserved for hand-off services.

With many different use-cases and bandwidth categories our goal is to create a strategy which optimizes the usability of all devices on the network. By ordering our bandwidth categories by highest to lowest priority we can allocate each users traffic into their respective categories while also dynamically adjusting the maximum flow of each category to reflect (as well as prioritize) the immediate demand in real-time [Kim11, KV04, LYC04]

1.2.3 Security

As telepresence technology matures, we will find ourselves relying on it in many new ways. The amount of information each device will be capable

of collecting from our every day lives underscores the importance of security policies and data management. We may, for instance, be comfortable sharing a live video stream with a friend, yet that same video stream may reveal information about our location or activities which we would not want to share with others. The need for confidentiality is important to both individuals as well as businesses and institutions.

To address these circumstances, access policies must be put into place. The use-cases for these access policies can be quite diverse, and any proposed system must properly address them all in order to be widely accepted. For instance, if I have a telepresence system at home it would be reasonable to allow my own cellular device to view my house at any time. However, another user trying to view my home environment should be denied access unless I have explicitly granted it to them. Once their session has ended, their access should again be revoked.

In access control systems, the concept of groups is not new. However, in telepresence systems groups apply to more than just users, they apply to the devices as well. If I have two or more telepresence devices in my home or an organization, I should be able to facilitate another user to view from either environment (and even switch between seamlessly) for the duration of their session. In this way, the remote user is less constrained by the software and can have the flexibility of interacting wherever the hardware allows.

So far we have used the term 'session' in two different scenarios, but have left the details a little vague. A session may not be limited to a series of sequential actions with one agent as in the traditional sense of the word. A session can move beyond the hardware it was created on and travel across several telepresence devices in a group.

The access control should be aware of other variables, such as time and geo-location. This allows for much more dynamic and easily defined roles.

Now that we've provided a background on our improved access control model, we can come up with some use-cases to describe how this control system should work.

1. Trigger Duration 6:00PM to 8:00PM - Allow All in User Group: Family Access to Device Group: MyHousehold

2. Trigger Location Office And Trigger Duration 9:00AM to 5:00PM - Allow All in User Group: Peers Access to Device MyPhone

3. Trigger Duration 5:00PM to 10:00PM - Allow All in User Group: Friends Request Device Group: MyHousehold

The first item states that anyone in the 'family' group can access my household telepresence devices freely for a short duration after dinner. The

second states that my work peers can instantly appear on my phone at their own discretion when I am at the office during working hours. The third states that friends can call me between 5 and 10 PM, but I must answer for them to connect.

1.2.4 Privacy

Encrypting live video streams is computationally expensive. Because of this, there is a trade off between bit rate and security that must be addressed. On one hand, we want to be reasonably confident that our communications are not being intercepted. On the other hand, we want the highest quality video and the longest use of our mobile device batteries. There is much to be gained by exploring this continuum, as well as the possibility of selective encryption to hide only the most sensitive data. [Feh13]

With the advent of Location Based Services (LBS) it is becoming increasingly difficult to control the extent in which a users locational information is used. Mix networks use short-term psudo-anonymous names to mask the identities of participants. With this mechanism, it becomes much more difficult for an adversary to correlate which actions in the system were performed by which users of the system. Since mix networks leverage multiple proxy servers to achieve anonymity, this is going to cause a tremendous spike in latency and bandwidth utilization. [LL12, FRF+07, PHE02]

The ubiquity of cameras and CCTV devices along with the proliferation of digital signal processors and facial tracking techniques are making the collection and centralization of user activities increasingly simple. Proposed countermeasures include using DSP techniques to selectively scramble identifying information such as license plates and faces to preserve user privacy, however these techniques rely on the cooperation of the surveillance administrators as well as the addition of costly components to the surveillance system. Currently the only countermeasure users have to prevent privacy intrusion is to opt-out of an otherwise useful service. In many cases such as public CCTV opting out is difficult if not entirely impossible. [Cav07, HR13]

Efforts to create a system which allows for users and even objects (in the case of license plates and sensitive documents) to register a preference for privacy and be 'scrubbed' from any published video have been explored. This does, however, require the cooperation of video producers and publishers by running their video through a central 'privacy scrubbing' service in a process that is comparable to a telemarketing 'do-not-call' list. While there is no technical reason preventing people from ignoring the video scrubbing process, publishers could be subjected to social and legal pressure to respect the privacy preference of others.

Ironically, the need for users to identify their privacy preference as well as their timestamped location into the central database dictates that they must give up their locational privacy in order to preserve their video privacy. [Bra05]

I propose that with additional countermeasures, the location of a user wishing to remain anonymous can be effectively verified while their identity remains a secret, provided several non-anonymous users whom the central privacy authority trusts each independently verifies the locational claim of the anonymous user via radio signal triangulation. In this way, the anonymous user can verify their location at a particular time without revealing any digital fingerprint or certificate to other nodes in the network including the central database.

1.2.5 Cloud Based Assisted Technologies

With the advance of internet connectivity, it is rare that modern mobile cellular devices are off line. If we are constantly in communication by means of a global (universal) IP address, It follows that we can achieve two things which we previously could not.

First, we can synchronize our local data with the data of others in real-time. This lends itself to instant news aggregation, social media, e-mail, instant messaging, and even VoIP technologies. This is not in itself extremely surprising as these features have existed in the scope of the desktop application since the dawn of always on high-speed broadband connections, but the ability to bring this to a widely distributed array of mobile devices brings the connectivity of our society (and the speed of information travel as a result) to an all new level.

Second, we can now outsource services which are not desired or capable of running on the mobile device to another computer. While this is typically (as of yet) a cloud service provider's machine, it is reasonable to consider that over time software will develop which allows users to host their own content from a simple always-on home computer which serves as a personal hub for content including but not limited to public social media, geo-secure proxy access, private home surveillance, and data storage. By using strict private/public tags, and 'group' authentication on a server's data as well as RSS-style content aggregators, it should be possible to design a decentralized 'home cloud' service which can serve many useful purposes to a mobile user in the field.

1.3 Identifying the Major Elements for a Lightweight Mobile Telepresence System

Lightweight mobile telepresence systems is a rapidly-evolving concept. Traditionally we have been bound by heavy and cumbersome desktop hardware, high-latency, and low network throughput. As these barriers have been reduced and removed, we have begun to redefine what it means to be connected.

Always-Available network communication
Real-Time video streaming
Geolocational services
Screen sharing (also useful for remote presentation)
File sharing (p2p for reducing infrastructure and bottle necks)
User/Group management and authentication

1.4 Summary

In conclusion, we are capable of much more than what is currently offered in terms of increasing the fidelity of our telepresence systems. On top of the increase in raw information storage capabilities, improvements in sensors as well as interactive peripherals have reshaped the way we use technology. If current trends continue we will soon find ourselves in a high density and highly distributed network of miniature devices, both as stand alone technologies (such as currently emerging smart phones) as well as embedded into every day consumer objects (as is the case with RFID tagging, QR coded items, and micro-controller enabled electronics). With this emerging paradigm it becomes much easier for computers to identify and process the objects around them, leading the way to many new modelling and digitizing techniques.

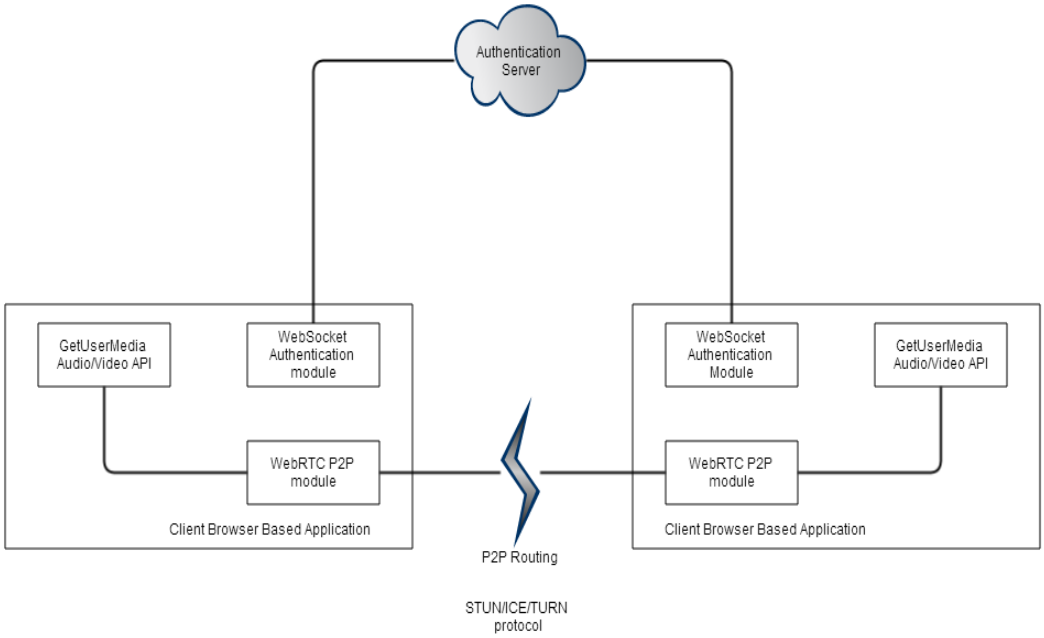
Chapter 2

Designing OTA Mobile Telepresence Services

2.1 Overview

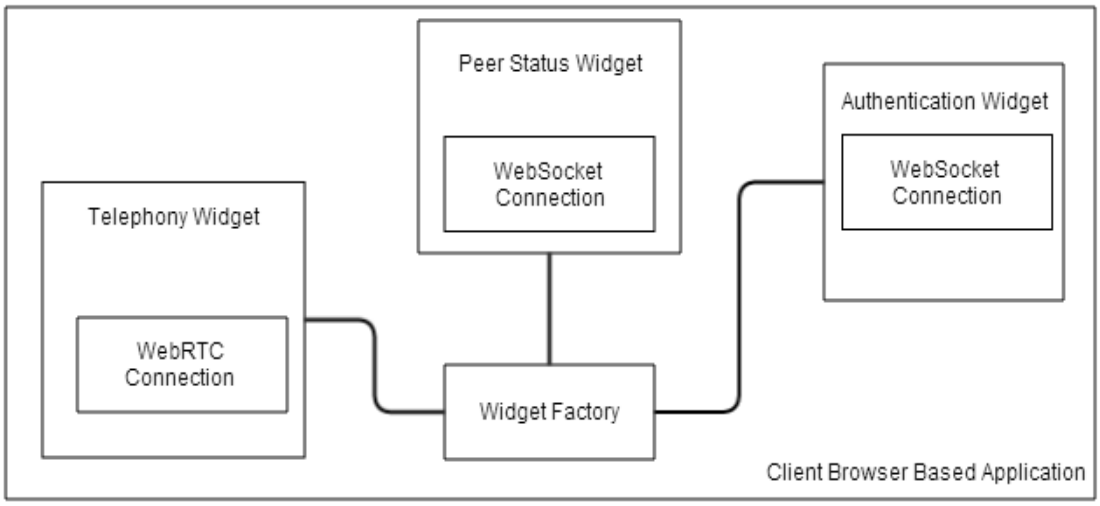
This telepresence system is built using both centralized and P2P routing techniques. While individual calls are handled using a P2P connection, a centralized authentication server is used to allow users to view who is currently online. The central server also plays a role in bootstrapping the P2P route when a call is first initialized.

When the client application is first retrieved from the server, the first task it must perform is creating and initializing the application widgets. Several of these widgets operate over the network using various communication channels that are implemented as widget dependencies. Opening a websocket connection with the authentication server and passing it into the widget constructor as a dependency allows network-enabled widgets to function correctly.

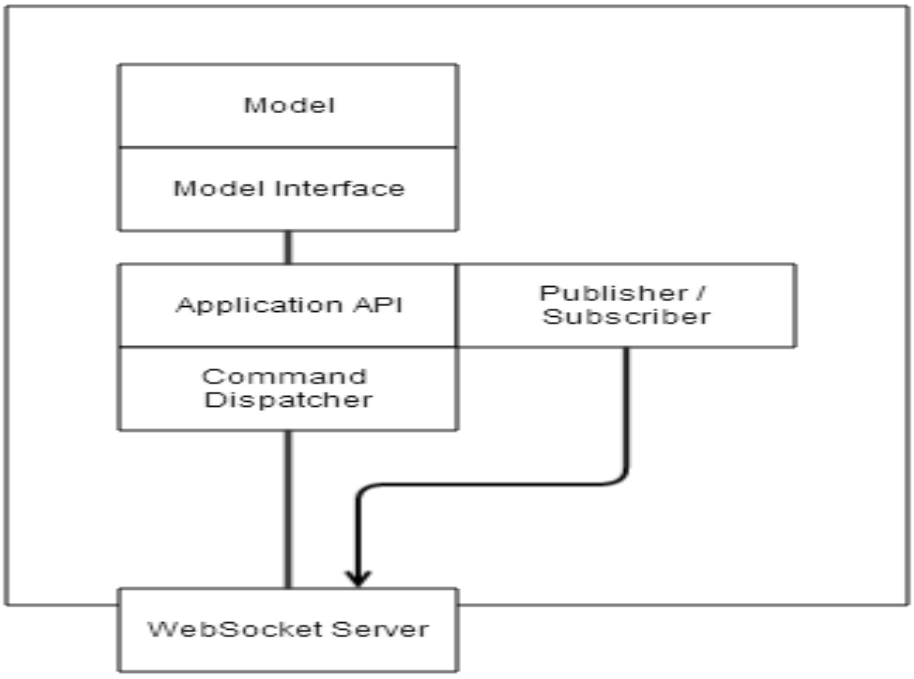


The client application is built using javascript within a browser environment. The main application library consists of a widget factory which is in charge of creating the major application components and injecting their UI into the DOM. Several of these widgets depend on network connectivity, so we provide these upon widget initialization using dependency injection. Because of these network components, it is important that we initialize our network objects before building most of our widget components.

The network dependencies that the widgets rely on primarily exist outside of the widget, and are passed in as references. In this way, several widgets can be utilizing the same socket connection. The socket connections handle addressing and message delivery seamlessly behind the scenes with some help from the widget factory, which registers each widget with a unique address identifier that is bound to each network message.



The authentication server contains an abstract model coupled with a controller allowing for basic read/write operations. The model is managed by an authentication API, which is the interface that client applications directly interact with over a webSocket channel. This authentication API allows for applications to perform various authentication tasks such as creating a user, logging in, and checking which other users are currently online. The authentication API is coupled with a publisher/subscriber module which allows a client to be notified if a particular part of the model changes. This is useful for being alerted when another client changes status from online to offline.



2.2 Authentication Server

The authentication server is in charge of identifying and connecting users. When a user connects to the website, a websocket connection is made between the authentication server and the client. Along with this connection a session ID is provided which we can use to identify the user over multiple transactions. When a user logs in, this session ID is coupled to their user name and marked as an active session. This allows other users to identify the user as online, as the session ID can be decoupled from the user name the moment the socket connection is closed.

2.3 Server Side Command Dispatcher

The authentication server receives data from the user in a 3-tuple form: ¡sender, command, args¿. The sender includes detailed information about the request origin. Not only are the user credentials included, but also which of the client application widgets made the request. This is to allow us to create a return route for the request. The command field is the name of the function we wish to invoke, and the args field is an array of arguments

to pass to the function. The commands for authentication have been kept intentionally simple, consisting of operations to sign up, log in, and request contacts.

2.4 Server Side RESTful Model

The authentication server works as a simple RESTful model. RESTful operations are primarily invoked from the server side command dispatcher which houses the business logic of the application. Our RESTful API only consists of four commands: 'create', 'read', 'update', and 'delete'. The user name of the client invoking the request is also included, allowing us to create permissive rules to limit who can access or modify groups of data.

2.5 Client Side Command Dispatcher

The client side command dispatcher is broken down into multiple parts. Commands may be launched from external sources such as the authentication server (eg. Describing bad log in credentials, or supplying a list of online users) or a P2P connection (such as a remote user initiating a call). Commands may also be launched from the clients web GUI.

Each client widget has it's own command API, and the application has naming and routing mechanisms to relay commands from these various sources to the appropriate widget. The widget then handles the command however it pleases.

2.6 Client Side Widget Factory

As the list of client side widgets grow, the widget factory serves as a way to create and register new widgets. The widget factory is in charge of creating unique names for each widget for message routing. It also acts as a factory/constructor for all client widgets. Because of this, the client side widget factory is a powerful high-level singleton that controls the initialization and address lookup for most of the application's contents

2.7 Client Side Widget Structure

Each widget should be contain a function called 'widget' which will be invoked when the client creates a new widget instance. As input, we pass in

any external dependencies the widget requires. This function is in charge of initializing the widget, building the widgets DOM and layout, as well as defining the external API the widget provides.

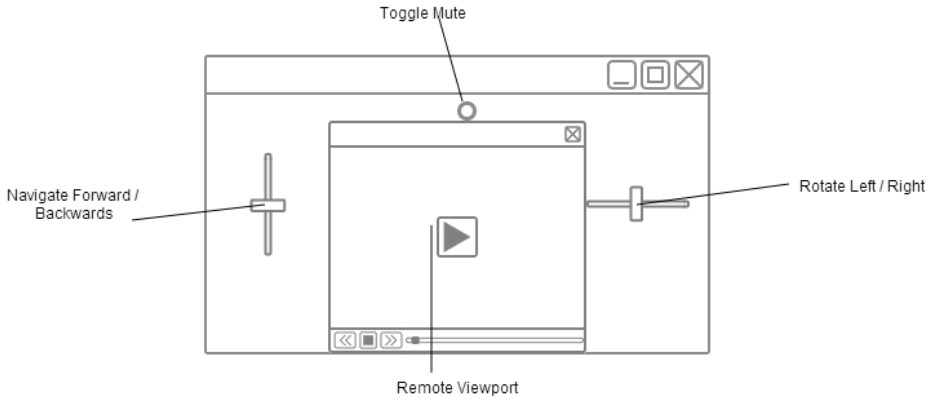
This external API can be accessed either internally from within the widget (via event handlers or timed events), externally from other local client-side widgets, externally from remote widgets on a connected peer machine, or externally from the centralized authentication server. The source of a request accessing the widget's API can be used to determine whether or not the widget performs the requested action.

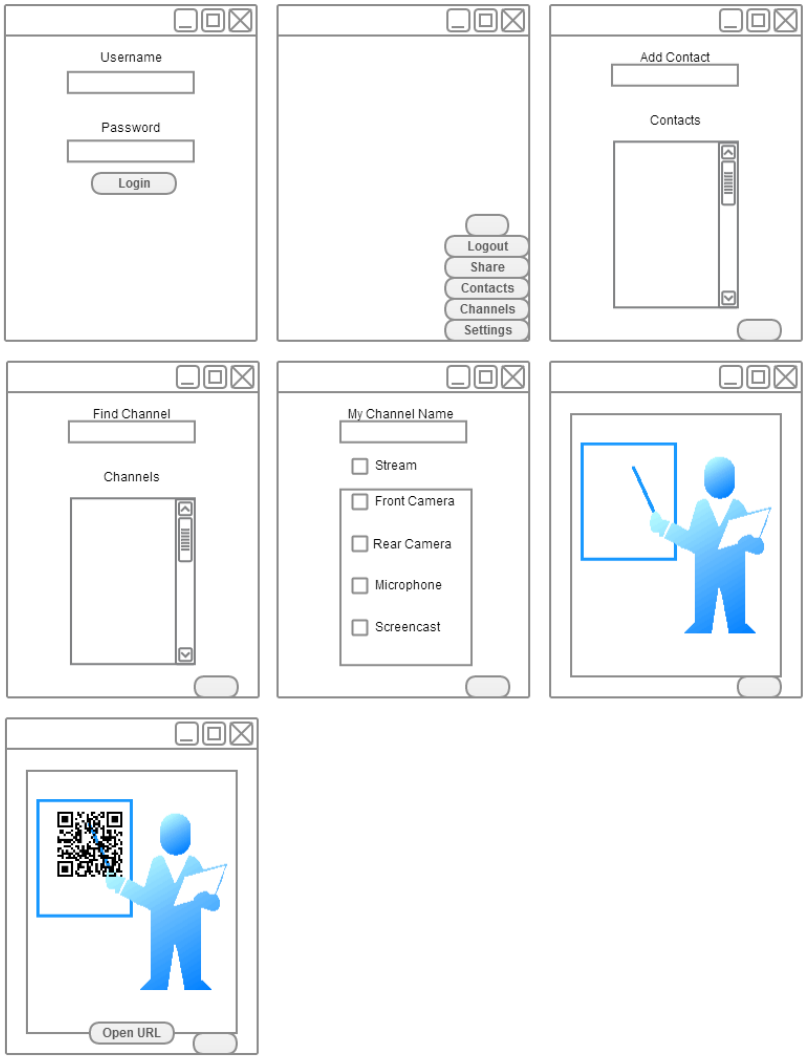
2.8 P2P Connection

A key component of this application will be the P2P widget. The P2P widget will be in charge of facilitating P2P connections with other users. Through this connection, users will be able to share VoIP communications, screencasts, and share files. The P2P connection will be in charge of connectivity, P2P routing, as well as stopping/starting telepresence calls.

The P2P connection will occur over the WebRTC API, which allows for ubiquitous P2P applications to connect within the browser. No additional dependencies, libraries, or plug-ins are required.

2.9 UI




Username
Password
Login
Logout
Share
Contacts
Channels
Settings
Add Contact
Contacts
Find Channel
Channels
My Channel Name
Stream
Front Camera
Rear Camera
Microphone
Screencast
Open URL

Chapter 3

Framework Implementation Details

3.1 Authentication Server

The authentication server begins by initializing a web server which the client can connect to in order to download the client javascript application

```
//create our webserver instance
var express = require('express')
var app = express()
var server = require('http').createServer(app)
var dispatcher = require('./cmdDispatch')
var sessions = {}
//allow the user full read access of our public javascript
   directory
//this allows us to link/embed custom javascript into the client
   application
app.use(express.static(__dirname + '/public'))
console.log("dirname:"+__dirname)
//configure and initialize web server view template engine
app.set( 'views', __dirname + '/../views' )
app.set( 'view engine', 'ejs' )

//serve our index.html page when the user requests the root of our
   webserver
app.get('/',function(req,res){
       res.render('index')
})
//serve our index.html page when the user request the root of our
```

```
    webserver
app.get('/voip',function(req,res){
        res.render('voip')
})
server.listen(80)
```

The authentication server then continues by providing a websocket upgrade to the previously initialized http request. This allows client applications to create a full-duplex real-time connection with the server over top of the existing stateless http protocol.

```
//create websocket server/listener overtop the existing express.js
    web server
var io = require('socket.io').listen(server)
console.log("server stuff")
//when an incoming connection is detected, we save the connection
    in our sessions table.
//when we recieve a message from the client, we send the message
    to our command dispatcher.
//if the dispatcher returns a response, we relay it back to the
    client application via the open socket.
io.sockets.on('connection',function(socket){
        sessions[socket.id] = {}
        sessions[socket.id].created = new Date()
        sessions[socket.id].socket = socket
        socket.on('message',function(data){
                if(data.command != null && data.args != null){
                        var response =
                            dispatcher.call(data.command,data.args)
                        console.log("response?",response)
                        if(response){
                                console.log('sending response')
                                socket.send(response)
                        }
                }
        })
})
```

This websocket connection provides a very simple mechanism. After a client opens a connection with the server, the server registers the socket ID in order to uniquely identify user sessions. The server then provides a messaging API which allows the client to send a JSON object (titled data) to the server. The socket handler passes this on to a command dispatcher

object for processing, which is detailed in the next session. If the dispatcher chooses to return a response to the client (whether this be authentication information, error information, or simply to acknowledge that we received the message) any response generated by the command dispatcher is then returned back to the client through the same socket.

3.2 Server Side Command Dispatcher

The command dispatcher is built within the framework of a requireJS module. This allows the code to be broken up among multiple files and manage their own dependencies using the lookup and code loading mechanisms provided by requireJS. When code is requested via a call to require, an object model containing the required API is returned. Listed below is our dispatcher.call() method that the server side websocket connection uses to interpret client messages.

Client messages are called with two types of data. First, the client supplies the action or command they wish to execute on the server. Secondly, an optional list of arguments or parameters is supplied to that command.

```
var modelHandler = require('./modelHandler')

module.exports = {
        call: function(cmd,args){
                console.log("calling command:",cmd)
                if(_cmds[cmd]){
                        var result = _cmds[cmd](args)
                        if(result){
                                console.log("returning result to
                                    sender",result)
                                return result
                        }
                }else{
                        console.log("command not found!",cmd)
                }
        }
}
```

the command dispatcher largely relies on a function table that supplies the underlying functionality. The structure of this is (unsurprisingly) a JSON table containing named functions that the user is allowed to invoke. We attempt to find the function the user requested, and pass along the user supplied arguments to it if found. If the function returns any result, the

command dispatcher will return the results to the server side web socket, which will then continue to return the results to the client application.

Below is the definition and implementation of our _cmds table, which acts as the business layer logic for the server side application.

The function table is ultimately responsible for manipulating the model that contains our user data. This includes creating new users, verifying the credentials supplied by a client application, and retrieving information about the online/offline status of users.

```
_cmds = {
        make_user : function(args){
                console.log('in make user!',args)
                if(args[1] == args[2] && args[1] !=null){
                        var isExistingUser =
                            modelHandler.query('read',["model","users",stringToModelID(ar
                        console.log("does user exist?",isExistingUser)
                        if(!isExistingUser){
                                var result =
                                    modelHandler.query('create',["model","users",stringToM
                                    ,passwordSignature:stringToModelID(args[1])})
                                if (result){
                                        return {success:true}
                                }else{
                                        return
                                            {success:false,error:"unknown
                                            error with model creation"}

                                }
                        }else{
                                return {success:false,error:"user
                                    already exists"}
                        }
                }else{
                        return {success:false,error:"passwords do not
                            match"}
                }
        },
        login : function(args){
                console.log('in login!',args)
                return {success:false,error:"function not built yet"}
        }
}
```

3.3 Server Side RESTful Model

3.4 Client Side Command Dispatcher

3.5 Client Side Widget Factory

3.6 Client Side Widget Template Structure

3.7 P2P Connection

Chapter 4

Widget Implementation Details

4.1 Login / Create User Widget

4.2 Online Peers Widget

4.3 P2P Call Widget

Bibliography

[AH11] A.P. Arias and U.D. Hanebeck. Motion control of a semi-mobile haptic interface for extended range telepresence. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3053–3059, 2011.

[AKK09] J.J. Ahmad, H.A. Khan, and S.A. Khayam. Energy efficient video compression for wireless sensor networks. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, pages 629–634, 2009.

[Bra05] J. Brassil. Using mobile communications to assert privacy from video surveillance. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 8 pp.–, 2005.

[Cav07] A. Cavallaro. Privacy in video surveillance [in the spotlight]. *Signal Processing Magazine, IEEE*, 24(2):168–166, 2007.

[CFB+ar] Samuel L. Clemens, William C. Faulkner, Elizabeth B. Browning, Judith S. Murray, Louisa M. Alcott, Harriet B. Stowe, and Carl A. Sandburg. Primarytitle. In Ralph W. Emerson, William B. Yeats, and Robert L. Frost, editors, *SecondaryTitle*, volume Volume of *ThirdTitle*, pages StartPg–OtherPg, PlaceofPub, PubDateFreeForm PubYear. AuthorAddress, Publisher. Notes.

[Feh13] G. Feher. The price of secure mobile video streaming. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 126–131, 2013.

[FRF+07] Julien Freudiger, Maxim Raya, Márk Félegyházi, Panos Papadimitratos, et al. Mix-zones for location privacy in vehicular networks. In *Proceedings of the first international workshop on wireless networking for intelligent transportation systems (Win-ITS)*, 2007.

[HR13] M.A. Hossain and S.M.M. Rahman. Towards privacy preserving multimedia surveillance system: A secure privacy vault design. In *Biometrics and Security Technologies (ISBAST), 2013 International Symposium on*, pages 280–285, 2013.

[Kim11] S. Kim. Cellular network bandwidth management scheme by using nash bargaining solution. *Communications, IET*, 5(3):371–380, 2011.

[KV04] Sungwook Kim and P.K. Varshney. An integrated adaptive bandwidth-management framework for QoS-sensitive multimedia cellular networks. *Vehicular Technology, IEEE Transactions on*, 53(3):835–846, 2004.

[LL12] Xinxin Liu and Xiaolin Li. Privacy Preserving Techniques for Location Based Services in Mobile Networks. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2474–2477, 2012.

[LYC04] Kam-Yiu Lam, Joe Yuen, and E. Chan. On using buffered bandwidth to support real-time mobile video playback in cellular networks. In *Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on*, pages 466–473, 2004.

[PHE02] Sang Yun Park, Moon Seog Han, and Young Ik Eom. An efficient authentication protocol supporting privacy in mobile computing environments. In *High Speed Networks and Multimedia Communications 5th IEEE International Conference on*, pages 332–334, 2002.