

Spring 2012

Desktop Sharing Portal

Ming-Chen Tsai
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects



Part of the [Computer Sciences Commons](#)

Recommended Citation

Tsai, Ming-Chen, "Desktop Sharing Portal" (2012). *Master's Projects*. 230.

DOI: <https://doi.org/10.31979/etd.zpxq-4e9b>

https://scholarworks.sjsu.edu/etd_projects/230

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Desktop Sharing Portal

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Ming-Chen Tsai

May 2012

Copyright © 2012
Ming-Chen Tsai
All Rights Reserved

Acknowledgement

I would like to thank Dr. Soon Tee Teoh for his excellent guidance throughout this project work and my committee members, Dr. Mark Stamp and Raj Chandran for their time and effort in reviewing the research content. Lastly, I thank my families for their love and support that enabled me to complete this research.

Abstract

Desktop sharing technologies have existed since the late 80s. It is often used in scenarios where collaborative computing is beneficial to participants in the shared environment by the control of the more knowledgeable party. But the steps required in establishing a session is often cumbersome to many. Selection of a sharing method, obtaining sharing target's network address, sharing tool's desired ports, and firewall issues are major hurdles for a typical non-IT user. In this project, I have constructed a web-portal that helps collaborators to easily locate each other and initialize sharing sessions. The portal that I developed enables collaborated sessions to start as easily as browsing to a URL of the sharing service provider, with no need to download or follow installation instructions on either party's end. In addition, I have added video conferencing and audio streaming capability to bring better collaborative and multimedia experience.

Table of Contents

1. Introduction	7
1.1 Related Work	7
1.2 Project Description	10
2. Design and Implementation.....	12
2.1 Protocols Considered.....	12
2.1.1 RDP	12
2.1.2 NX Technology	13
2.1.3 TrueRemote	13
2.1.4 SPICE.....	13
2.1.5 VNC/RFB	14
2.2 Frontend Components.....	14
2.2.1 Sharing Data Transmission	15
2.2.2 Conferencing	18
2.2.3 Audio Transmission	18
2.3 Backend Modules	22
2.3.1 Image Access Detail File	22
2.3.1 Maintainer	22
2.3.2 Backend Starter.....	23
2.3.3 Database.....	24
2.3.4 Resource Units.....	25
2.3.5 Service Deployments	29
3. Portal Workflow.....	31
3.1 Enrollment	31
3.2 Creating a new room	33
3.3 Collaboration Room	35
3.4 Video conference.....	36
3.5 Audio streaming	38
4. Evaluations and Usability	39
4.1 Sharing Environment	39
4.2 Video Conference Popup.....	39
4.3 Audio Stream Delay.....	40
5. Conclusion.....	41
5.1 Future Research.....	41
References	42

List of Figures

- RemoteFX Architecture	8
- YouOS Web Operating System	9
- Google+ Hangouts	10
- Portal Modules	12
- Remote Desktop Data Path	17
- Wsproxy Startup Message	17
- Source Code: HTML5 audio element	19
- Audio Stream Data Path	20
- DarkICE Configuration Parameters	21
- IceCast Configuration Parameters	21
- Source Code: Database Schema	24
- Portal on EC2s	26
- Amazon AWS - Resources Panel	27
- Amazon AWS Datacenter Locations	29
- Amazon AWS - Management Console	30
- Enrollment Workflow	31
- Enrollment Screenshot	32
- Create New Room	33
- Create New Room Screenshot	34
- Main Lobby Room Index Screenshot	34
- Collaboration Room	35
- Video Conference Workflow	36
- Room with Video Chat	37
- Audio Streaming Workflow	38
- Video Device Access Request Popup	40

1. Introduction

Technical difficulties for establishing desktop sharing sessions are common issue in many IT scenarios. As mentioned in [1], situations such as how to escalate user's browser or operating system in order to allow sharing access privilege, adjusting firewall configuration to enable network traffic, or locating the machine's network IP address and sharing port for the other party - all has proven to be technically advanced for most Internet users. The Remote Desktop solution is a narrow area in the computer software field. Its history of origin dates back to 1987 by Netop [11]. Up until the early 2000s, almost all solutions required software installation and administrator approval on the client side. Only the more recent Java based browser-plug-in solutions take advantages of cached approval in browser to silently initiate sharing sessions upon user acknowledgement. With the growing trend of emerging web standards, and focus on cross-browser compatibility, remote desktop solutions can be re-introduced in new forms.

1.1 Related Work

Options for providing a computing environment over the Internet have been explored in the past. More recently, companies providing virtualization solutions have developed similar technologies, and their differences against remote desktops are also noted in this section.

VDI – Virtual Desktop Infrastructure, a term introduced by VMWare, services virtualized desktop operating systems through centralized servers. Some of the VDI's goals include lessening time needed to provision new client system, centralized location for all client data which allows easier data-backup and management since all client data are stored in a single data storage area. It also allows reduced cost in terms of creating new client systems. Such that when a new client is needed, an older system can be plugged into the infrastructure and be used as a thin, dumb terminal and allowing the server to provide a virtualized desktop which is more powerful. Technologies from other companies

that provide similar technologies include XenDesktop from Citrix [17], and RemoteFX from Microsoft Windows Server 2008 [18].

Different from Remote Desktop Systems, a virtualized desktop client may not always need to have constant network connections. Such solutions are hybrid-VDIs, in that not all clients are thin or dumb ones. There are also client-hosted VDI solutions that transfer a VDI image data to local client for its own operating need without requesting further resource from central server. In this case, the local client-hosted solution would require its own storage space. High-performance GPU/CPU is also desired in some cases, such as RemoteFX where the technique is to forward the graphics device commands directly to client's end.

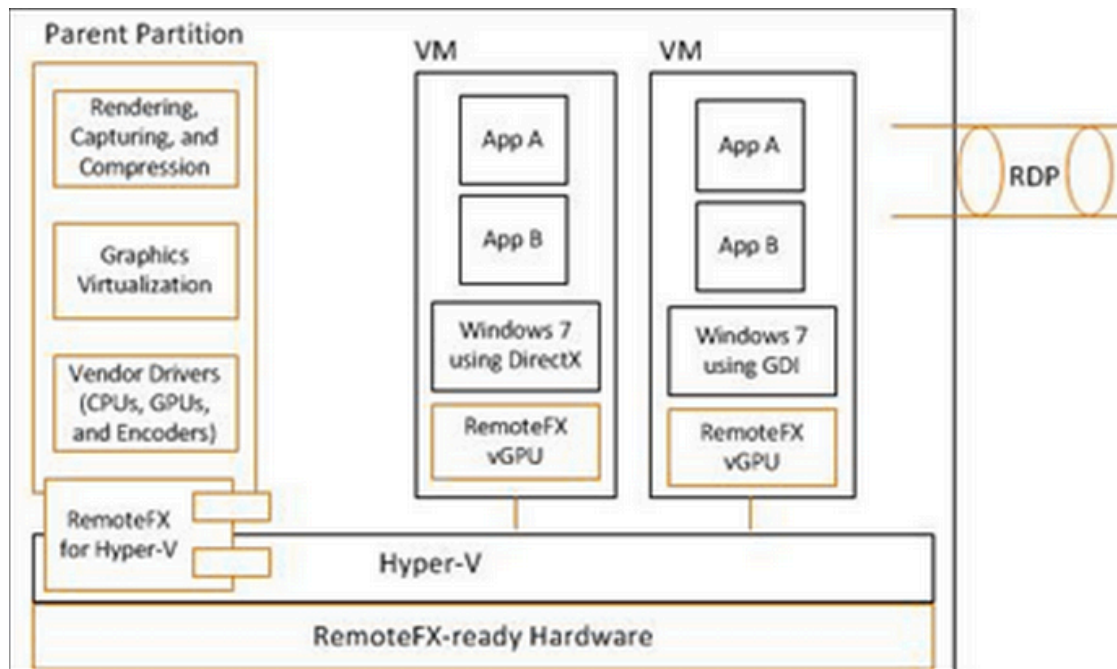


Figure 1 - RemoteFX Architecture

Web Operating Systems – The ideology behind such solutions is to provide an environment where online browsing activity can be shared. A web operating system is neither a remote desktop, nor a virtualized system. It is a client owned operating environment that resides within a web browser, such that

the internals of the operating system has no resemblance of a typical one such as Windows or Linux. When a user opens a web browser to access a resource inside a web operating system, he/she is directly accessing the resource from his/her browser. Data that are stored “locally” on the web operating system is transferred to server through web transfer protocols – such as HTTP, where the hosting entity stores the user profile data and its data storages. Some of the more notable solutions in this area include YouOS [19] and Synaptop [20]. Both of which also provides API interface for external developers to write applications that can be used inside such operating systems.

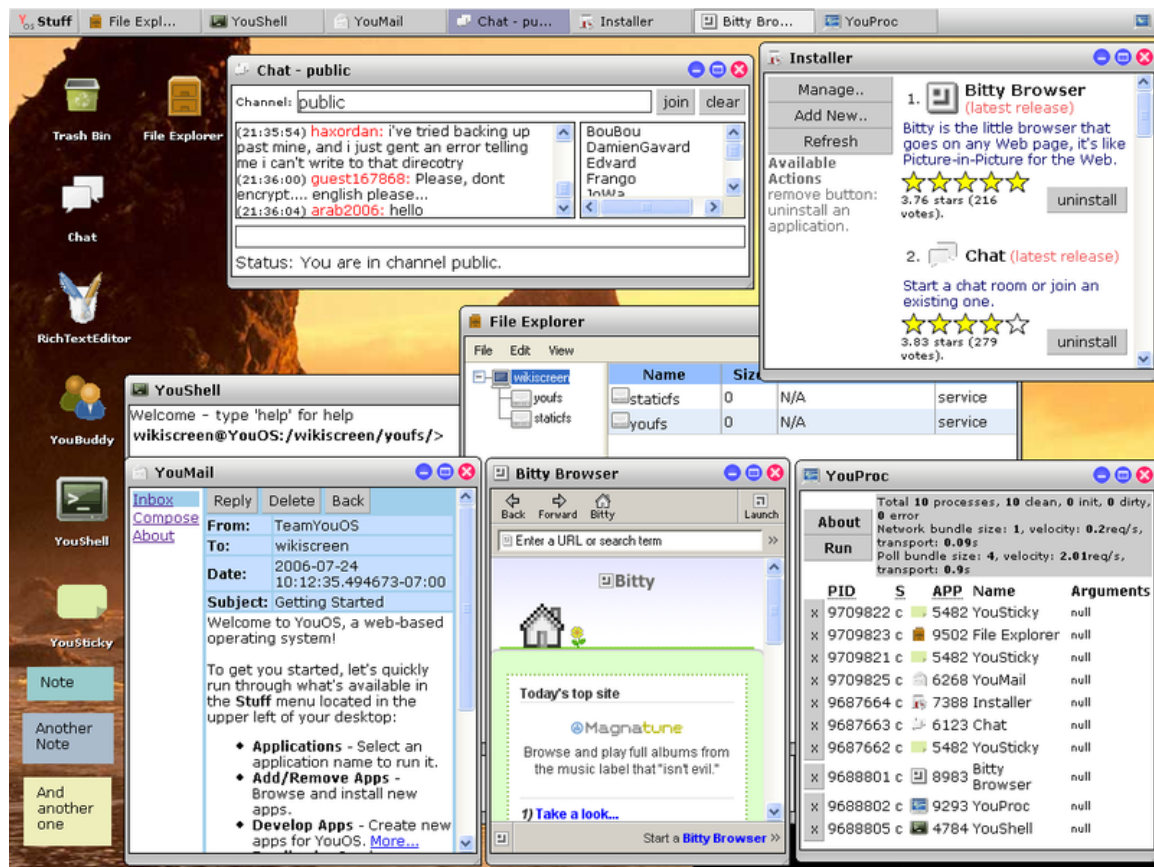


Figure 2 - YouOS Web Operating System

Video Conferencing Systems – It is worth mentioning that the video conferencing capability provided by the portal is similar to current offerings such as Google+ Hangouts [22], VoxWire [23], or FlashMeeting [24]. The portal,

through TokBox, takes the approach of providing such capability in a basic way and not superior to the above offerings.



Figure 3 - Google+ Hangouts

1.2 Project Description

With the growth of the Internet, virtual gaps between individuals are diminishing. Uploaded contents are growing at phenomenal rate, and the equivalent growth rate is true for websites and apps that are focused towards consuming of the uploaded contents. This brings increasing opportunities for collaborated desktop sharing environments where participants are brought together to collaboratively browse and share experiences through the Internet.

In this project, using HTML5 standards and open source technology incorporate after careful analysis. I have constructed a desktop sharing portal that hides away the need for technical knowledge required in order to conduct

collaborative browsing. In contrast to what a VDI would offer, the portal gives emphasis on usability-friendliness. Consider the cases where VDI services require not-so-thin clients, and oblivious to the client's network topology, I have developed the portal with the goal of serving current Internet users who do not have the optimal network setting and neither a powerful GPU client hardware. And unlike a Web Operating System, the portal offers more freedom in terms of application connectivity – since almost any desktop applications can easily be installed and used in the shared portal environment. To enable this, I implemented 4 types of portal components of which their individual tasks are service request fulfiller, image warehouse maintainer, bookkeeping data store, and image units, where the image units can be swapped between different operating systems such as Windows or Linux. I have also developed the frontend module such that it allows videoconferencing capabilities, and audio streaming feature so that it encourages tighter participant interactions during sharing sessions.

2. Design and Implementation

I have implemented the portal to be composed of 4 types of modules, and developed the frontend module in PHP, HTML, JavaScript and CSS on Apache httpd2, backend module in Java SE6, virtual images using Ubuntu and Windows using shell scripts, and data storage using MySQL 5.

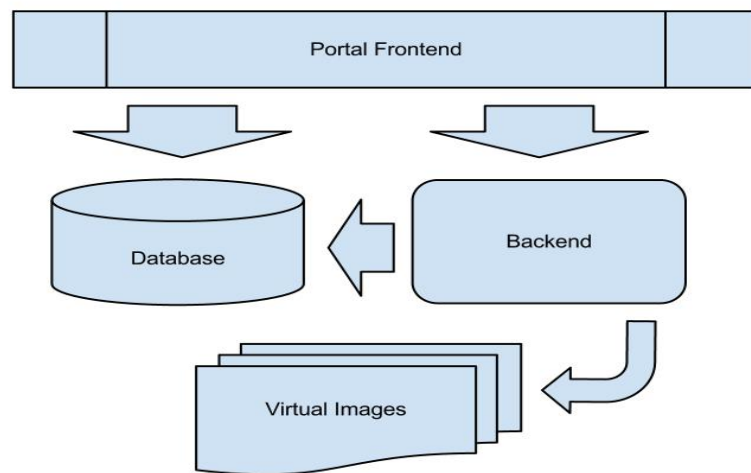


Figure 4 - Portal Modules

2.1 Protocols Considered

A key component for this portal is the protocol used for delivering remote desktop. Below, I present my analysis on the protocol options in the context of the portal usage and outline each protocol's strength and weaknesses.

2.1.1 RDP

Microsoft's Remote Desktop Protocol [7] was first introduced during the release of Windows NT 4.0 in July 1996. The main strength for RDP is that it is deeply embedded in Microsoft operating systems, meaning the majority of the Internet users today. The RDP today offers a strong suite of features – audio redirection, file transfer redirection, and printer redirection. Some of its

weaknesses includes, due to popularity, is that it is often the target of malicious attacks. Another downside is that it is not supported on hosts that are non-Windows operating system.

2.1.2 NX Technology

Developed by the Italian company NoMachine [15], NX technology allows X Windows System based system. It is based on the Differential X Protocol Compressor project, and capable of delivering improved X display performance even through slow dial-up link. However, there is currently not a HTML5 based NX client, and the open source effort has stalled due to company business direction. Therefore, it is removed from the list of choices to be considered for our portal usage.

2.1.3 TrueRemote

TrueRemote, developed IchiGeki in Japan [16], is based on a proprietary video codec called “GaeBolgVideoCodec” to provide high speed video and audio remote desktop experience. It currently works on Microsoft Windows platform, and provides the performance improvement by optimization at GDI and Direct3D levels. But due to its limited offering on only Windows operating system, and IP protection, it is not considered for the portal usage.

2.1.4 SPICE

Simple Protocol for Independent Computing Environments [8] is an open source project led by RedHat. SPICE was initially designed by Qumranet Israel in 2008 to focus on richer remote multimedia experience of which older protocols such as VNC or RDP was not initially designed for. SPICE requires the guest operating system to be a virtualized one. It transmits virtualization level commands directly to the client for rendering tasks on both video and audio data. But the weakness of SPICE is that the client side is required to have a high performance-rendering chip in order to guarantee smooth processing of commands received. Also, the current state of SPICE client requires library driver installation that could increase technical complexity on end user’s end in our scenario.

2.1.5 VNC/RFB

Virtual Network Computing [4] is a highly popular open source remote desktop protocol that has been ported to most operating systems. The protocol behind VNC is RFB – Remote Frame Buffer protocol [5]. The original RFB protocol is simply a graphics display transmission protocol, meaning that it lacks what RDP or SPICE offers. However, many variants of RFB have been developed to incorporate those features. Such as UltraVNC for adding file transfer capability, multi-monitor support, or TigerVNC for transfer rate adjustments on different display needs. The weakness of the RFB protocol lies in its image frame based transmission protocol. In comparison to other protocols that sends higher-level system commands, RFB data is graphics based hence taking up more data bandwidth need. Though at the same time, this transmission method has shown to have good flexibility such that any client system can easily process the buffer display commands that are received.

Our choice of protocol is VNC due to its ease of portability. As seen with projects such as [2] and [3], bringing RFB data directly to user browsers is now achievable with HTML5. Previously, the client side permission would be required for installation need, which would introduce room for complexity and higher chances of incompatibilities.

2.2 Frontend Components

I have developed the frontend components that consist of the following. Apache2 webserver is used with mod_php enabled. PHP server enabled with MySQL database connectivity. User profile registration handling and profile data insertion to database is handled by the registration handler I wrote using PHP, and SQL queries. Its job is to receive profile data and processing requests for new user profile data against member profile table. If an account already exists, registration request will be rejected.

Users do not have to provide a password during the sign up stage. A random password is generated and sent in an email to the user's email account

in order to ensure authenticity. User passwords are hashed using MD5 before being stored into database table. No password is stored in plaintext form. I developed the portal landing, lobby, and room pages in HTML, CSS and uses JavaScript and jQuery in order to provide a web UI immediate display capability. This removes the need for page refresh before seeing new content updates. I have written the create-room or remove-room handler such that when the request is received, a forked “getJSON” jQuery call is made out to separate removeImage or createImage PHP handlers, where the actual room ownership grant or removal actions are completed against database.

Since HTTP is a stateless protocol, I used PHP’s session management to keep track of the user’s states between the portal pages in order to serve the user correct page contents. During the login phase, when the user enters password characters into the password textbox, it is masked to provide privacy. This is developed using the standard HTML password textbox feature. The entered username and password field data fields are string-escaped first before being used for MySQL query in order protect against Sql-Injection attack. In order to authenticate login request, I wrote a query process in PHP to MD5 hash the incoming plaintext password first before using SQL query to check against member profile table and to determine whether to allow or disallow a user login. I have added page analytic JavaScripts to all the pages in order track page usage statistics.

In order to provide real-time room current condition in the main lobby page, I employed JavaScript and jQuery’s “.attr()” method to cause the thumbnail image to be constantly queried by the browser with time interval of 1 second between each query. I used Chrome’s embedded Developer Tools’s query history to determine whether the query implementation would be too aggressive or not. 1 second interval implementation in this situation seems suitable in our scenario.

2.2.1 Sharing Data Transmission

Before WebSocket and Comet (another web transmission protocol that allows efficient 2-way data transmission) were standardized by W3C, most

browsers would use Java Applets with TCP/IP traffic to achieve desired streaming data transmission. In the portal's scenario and with the decision of using VNC/RFB protocol, I have developed the following in order to provide sharing room that can be used by multiple participants.

In the main image page, I wrote sharing image queries to check whether this image is indeed owned by the user through PHP, and MySQL queries. After which, a data row is inserted into `ds_session` table to denote the time of which this sharing usage session started, and also the `member_id` and `image_id`.

noVNC [2] embeds both the RFB protocol JavaScript library, and the Websock library to provide the WebSocket based connection between browser and the proxy server. Since HTTP does not provide streaming based bi-directional data transmission. I installed and configured `vncserver` to wait for connection on port 5901 with a required password set on the Linux image that is to be used for sharing room. In the backend module's startup script that I have written, it initiates a websocket proxy server. The websocket proxy server's job is to route traffic between the websocket, on browser's end, and the TCP socket, on Linux image's end. In our case, this proxy pipes sharing image traffic between the web browser and the VNC server.

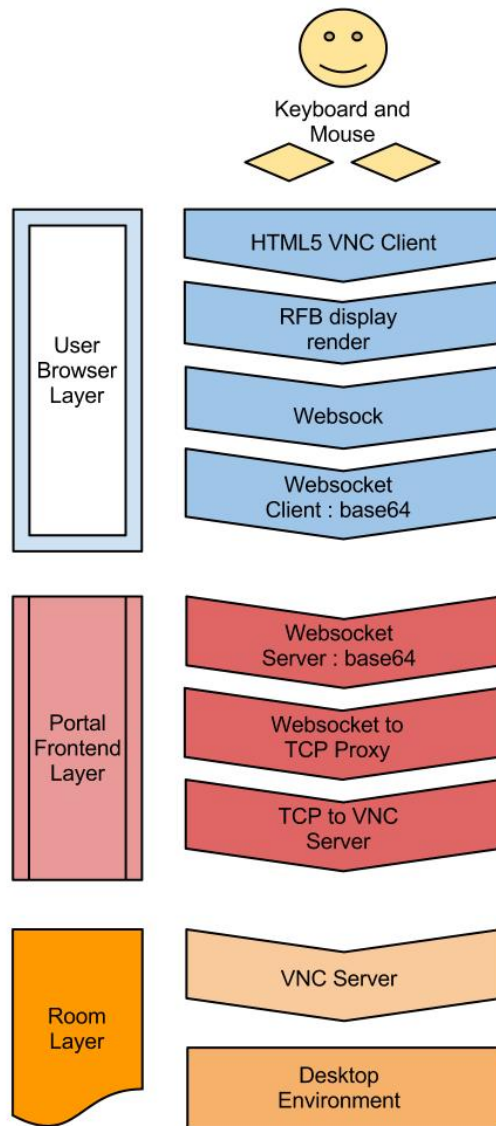


Figure 5 - Remote Desktop Data Path

The above Figure shows the image data flow path between remote desktop and the user's browser through various layers. Different from a typical VNC connection, the data flow from "TCP to VNC Server" onwards goes through base64 transformation and travels using WebSocket protocol in order for the HTML5 based browser to render without any native VNC client need.

```
Starting wsproxy for images...
python wsproxy/wsproxy.py --web /var/www/vnc/ 6080 192.168.56.101:5900 &
root@xn--kpry2pu2i:/dsmain/DSMaintainer# WebSocket server settings:
- Listen on :6080
- Flash security policy server
- Web server. Web root: /var/www/vnc
- No SSL/TLS support (no cert file)
- proxying from :6080 to 192.168.56.101:5900
```

Figure 6 - Wsproxy Startup Message

2.2.2 Conferencing

I have developed the room-sharing page with the following items in order to allow videoconference session creation, access token creation, publish or subscribe to a video session.

I imported TokBox's PHP library in order to be allowed access to their application interface. A one-time API key is used here which was received during a one-time sign-up process. A TokBox session object, hereafter referred to as Tsession, is generated after successful initialization. Its initial state is empty. In order to start receiving notification events, event listeners need to be registered to the Tsession object. Event listener registrations are done by calling a setup function I have written, which adds the listeners to the Tsession object. The setup function is invoked when "Start Video Chat" is clicked.

The 2 listeners registered are named sessionConnectedHandler, and streamCreatedHandler. The sessionConnectedHandler listener is called when the initial connect request has been acknowledged by TokBox. I implemented the listener such that when it is called, a new HTML div element is created on the sharing-room page. A unique div id is set before being added, so that when session publishing starts, I can specify published content to use the new div just added by its id. In order to correctly display the video box, I first created a pointer div element which points to the pre-designated video showing area at the bottom of the page by using document.getElementById(). The newly created div, by session establishment in previous step, is then added as a sibling node to the pointer div by using [new_div_node].parentNode.insertBefore(div, [new_div_node].nextSibling). In streamCreatedHandler's case, I implemented the listener to first ignore the stream notification if the session is the one of which the user is publishing – notification events are received for all new created streams. After which I utilized the same techniques for adding the video box for notified new stream.

2.2.3 Audio Transmission

Using the experience from SPICE protocol analysis, I explored how to achieve the same audio streaming capability even though the choice of protocol,

RFB/VNC, does not provide such feature. I narrowed the selections of open sourced audio capturer down to Ices2 and Darklce. Here are the findings after installation, configuration and experimental streaming runs.

Both Ices2 and Darklce captures audio directly from audio device hardware interface - /dev/sdp. Ices2 captures audio and streams in ogg/wave format, Darklce streams in mp3 format. Due to the yet concluded HTML5 standard on which audio format would be the most suitable for general public use, I have defaulted the portal to use Darklce, which streams mp3. Since Darklce streaming is an OS user-level only stream, I needed to install a broadcaster to publish and serve the stream. I found IceCast, being the most stable of such open source project.

I have configured the audio quality to sub-optimal settings to balance network traffic load and quality of audio stream. This being 22050 sample rate per second, 16 bits per sample, and stereo channel. Which is roughly similar to the standard FM radio channel audio quality, but not matching CD quality. On the sharing room page, I embedded a HTML5 “audio” element that plays the audio stream automatically when the element is loaded by an HTML5 compliant browser. Audio stream authentication between Darklce and IceCast is allowed. I configured this using the default password. Since all 3 data communication channels are separate and the sharing image is inside AWS’s cloud firewall, I emphasis less concern on possibly exploited streaming service but more on the convenience of service setup once server number increases.

```
<audio controls="controls" autobuffer="false" preload="none" autoplay="true">
  <source src="http://192.168.56.101:8000/darklce1" type="audio/mp3" autobuffer="false" preload="none" />
  Your browser does not support the audio element.
</audio>
```

Figure 7 - Source Code: HTML5 audio element

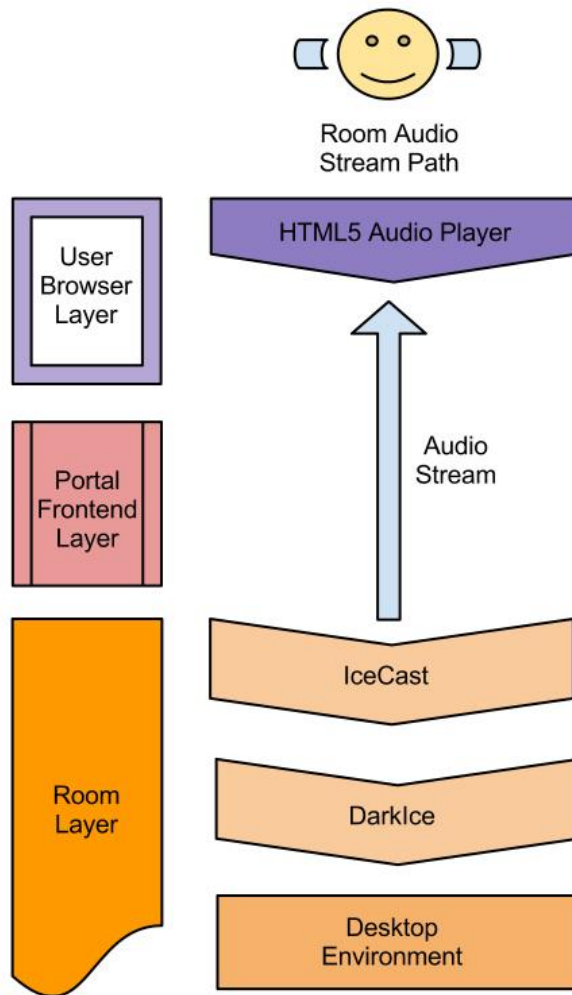


Figure 8 - Audio Stream Data Path

```

root@xn--k... /tmp -- ssh  bash
# sample DarkIce configuration file, edit for your needs before using
# see the darkice.cfg man page for details

# this section describes general aspects of the live streaming session
[general]
duration      = 0      # duration of encoding, in seconds. 0 means forever
bufferSecs    = 1      # size of internal slip buffer, in seconds
reconnect     = yes    # reconnect to the server(s) if disconnected

# this section describes the audio input that will be streamed
[input]
device        = /dev/dsp # OSS DSP soundcard device for the audio input
sampleRate    = 22050    # sample rate in Hz. try 11025, 22050 or 44100
bitsPerSample  = 16      # bits per sample. try 16
channel       = 2        # channels. 1 = mono, 2 = stereo

# this section describes a streaming connection to an IceCast2 server
# there may be up to 8 of these sections, named [icecast2-0] ... [icecast2-7]
# these can be mixed with [icecast-x] and [shoutcast-x] sections
[icecast2-0]
bitrateMode   = abr      # average bit rate
format        = mp3      # format of the stream: ogg vorbis
bitrate       = 96       # bitrate of the stream sent to the server
server        = 127.0.0.1 # host name of the server
port          = 8000     # port of the IceCast2 server, usually 8000
password      = password # source password to the IceCast2 server
mountPoint    = darkice1 # mount point of this stream on the IceCast2 server
name          = DarkIce trial # name of the stream
description   = This is only a trial # description of the stream
genre         = my own   # genre of the stream
public        = no       # advertise this stream?
localDumpFile = dump.ogg  # local dump file

```

Figure 9 - DarkICE Configuration Parameters

The DarkICE configuration file specifies how the audio is captured – such as from which interface, sampling rate, audio format, and buffer settings.

```

<icecast>
  <limits>
    <clients>5</clients>
    <sources>1</sources>
    <threadpool>2</threadpool>
    <!-- <queue-size>524288</queue-size> --> before using
    <queue-size>224288</queue-size>
    <client-timeout>30</client-timeout>
    <header-timeout>15</header-timeout>
    <source-timeout>10</source-timeout>
    <!-- If enabled, this will provide a burst of data when a client
    first connects, thereby significantly reducing the startup
    time for listeners that do substantial buffering. However,
    it also significantly increases latency between the source
    client and listening client. For low-latency setups, you
    might want to disable this. -->
    <burst-on-connect>1</burst-on-connect>
    <!-- same as burst-on-connect, but this allows for being more
    specific on how much to burst. Most people won't need to
    change from the default 64k. Applies to all mountpoints -->
    <!--<burst-size>65535</burst-size>-->
    <burst-size>655350</burst-size>
  </limits>
  <authentication>
    <!-- Sources log in with username 'source' -->
    <source-password>password</source-password>
    <!-- Relays log in username 'relay' -->
    <relay-password>password</relay-password>
    <!-- Admin logs in with the username given below -->
    <admin-user>admin</admin-user>
    <admin-password>password</admin-password>
  </authentication>

```

Figure 10 - IceCast Configuration Parameters

2.3 Backend Modules

2.3.1 Image Access Detail File

I created an initial image access detail file that is to be read by the Maintainer when the Maintainer process starts. Each line listed in the image access file maps to an available sharing image entry. The entry has information regarding the sharing image server's VNC address, VNC port, thumbnail snapshot storage location, and its public access URL. The database connection URL is also stored in the image access detail file.

2.3.1 Maintainer

I developed the Maintainer process and placed it as the backbone of the portal; it manages sessions and images on periodic basis. I implemented the Maintainer using Java SE 6. I designed the Maintainer to be composed of 2 types of job runners – database job runner and snapshot job runner. Both job runners are extended from Java's TimerTask class. I have defined them both to run on 5 seconds intervals. The following describes what I have implemented when a job starts.

Database Job Runner

I have written the Database Job runner to interact with ds_image table in MySQL database. When the Database Job run starts, I loop through the images list that was previously retrieved from the initial image access file reading and check to see for the currently pointed image if there is an entry for it in ds_image table already. If not, I then proceed to a SQL Replace query to insert this available image into the ds_image table.

Snapshot Job Runner

When the Snapshot job runs, I similarly loop through the same list of sharing images and updates each of the thumbnail image files. I start by taking a vncsnapshot of the current condition; a large vnc snapshot jpeg file is produced at the end of this stage. The values used from image detail file in this step include vnc server password, snapshot's storage path, and vncscreen number on the image server. During implementation for this feature, I encountered stale

cache issue on Linux operating system when trying to freshen thumbnail image. Therefore, to ensure that every thumbnail request would always get the latest image, I first use a jpeg converter, from open source project ImageMagick, to convert the large jpeg file to a smaller temporary jpeg file that is separate from what the page would serve. Then in the subsequent stage, I would move the smaller temporary file to overwrite the file of which the page serves to lobby user who is viewing the page with thumbnails.

2.3.2 Backend Starter

I have created a Backend Starter script that acts as the parent process for all the Backend Module activities and also the websocket proxies. When the script is run, I first kill all existing Maintainer or websocket proxy processes to ensure fresh backend state. Then I forward the Image Access Detail file content to Maintainer process. In addition to that, for each entry in the file, a websocket proxy server in the background thread would be started through the AWK function that I have written. This dramatically lessens the work that would be needed to administer startup, refresh, and cleanup of any residual Maintainer or websocket processes that were started during previous test runs.

2.3.3 Database

I have designed the necessary schema for the portal to be contained within 4 tables currently. The portal currently uses one database, and it is implemented on a MySQL server.

```
mysql> desc ds_image;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11) | NO | PRI | NULL | auto_increment |
| screenpath | varchar(256) | YES | | NULL | |
| access_url | varchar(256) | YES | UNI | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc ds_member;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11) | NO | PRI | NULL | auto_increment |
| user   | varchar(32) | NO | | | |
| password | varchar(32) | NO | | | |
| email  | varchar(255) | NO | | | |
| ip     | varchar(15) | NO | | | |
| date   | datetime | YES | | | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> desc ds_member_image;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11) | NO | PRI | NULL | auto_increment |
| member_user | varchar(256) | YES | | NULL | |
| screenpath | varchar(256) | YES | UNI | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> desc ds_session;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int(11) | NO | PRI | NULL | auto_increment |
| member_id | int(11) | YES | UNI | NULL | |
| image_id | int(11) | YES | | NULL | |
| start_time | timestamp | NO | | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Figure 11 - Source Code: Database Schema

The following are the 4 database tables that I have created for use in the current portal design:

- “ds_member” for Storing user profiles. Its primary key is the auto-generated member_id. There is a 1-to-1 relationship between actual user and each row in ds_member table. Access to this table is only from Frontend module.
- “ds_member_image” for tracking member image ownership data.
- “ds_image” for tracking currently available sharing image details.

- “ds_session” to keep track of ongoing session being used in portal.

In the frontend main lobby page, I have written the following query as part of the PHP procedure to determine the user’s image ownership.

- “SELECT di.id, dmi.member_user, dmi.screenpath, di.access_url FROM ds_member_image as dmi join ds_image as di on dmi.screenpath = di.screenpath WHERE member_user = (username)” The user being queried does not currently have an image if the returned result is 0 row.

I have also written the following query that displays to the user the current status of sharing rooms they currently own.

- “SELECT dsm.user FROM ds_member as dsm join ds_session as dss on dsm.id = dss.member_id and dss.image_id = (image_id)” This query effectively retrieves all the users that are currently using the room with “image_id”.

2.3.4 Resource Units

I devised each of the portal’s sharing room to be an EC2 instance on Amazon Web Services Cloud [9] after these considerations. First consideration is the room growth being proportional to the number of registered users. If user volume growth outpaces virtual image hosting capacity, a new user’s image creation would be denied. Hence it is important that the portal be able to scale horizontally if such scenarios occur, and only a cloud-based provider could immediately fulfill this need. Second is Amazon’s “Free Usgae Tier” offering which is unavailable amongst other providers such as GoGrid or RackSpace.

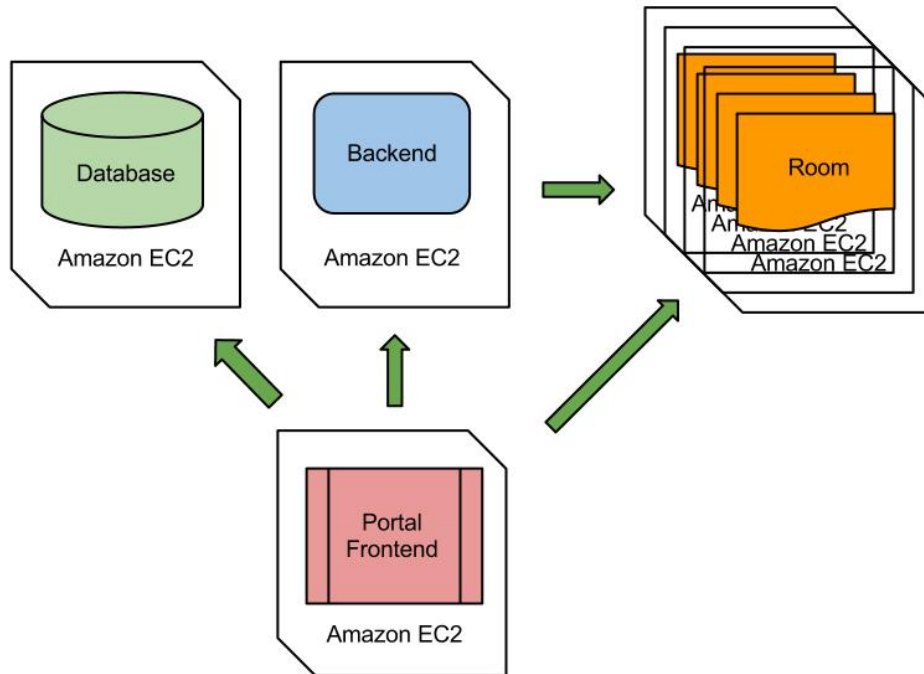


Figure 12 - Portal on EC2s

Note that each of the service modules in above figure is a separate Amazon EC2 instance. I utilized virtual image replication techniques provided by Amazon AWS's administrative service, which are similar to what is found in products such as VMWare Workstation Servers [10].

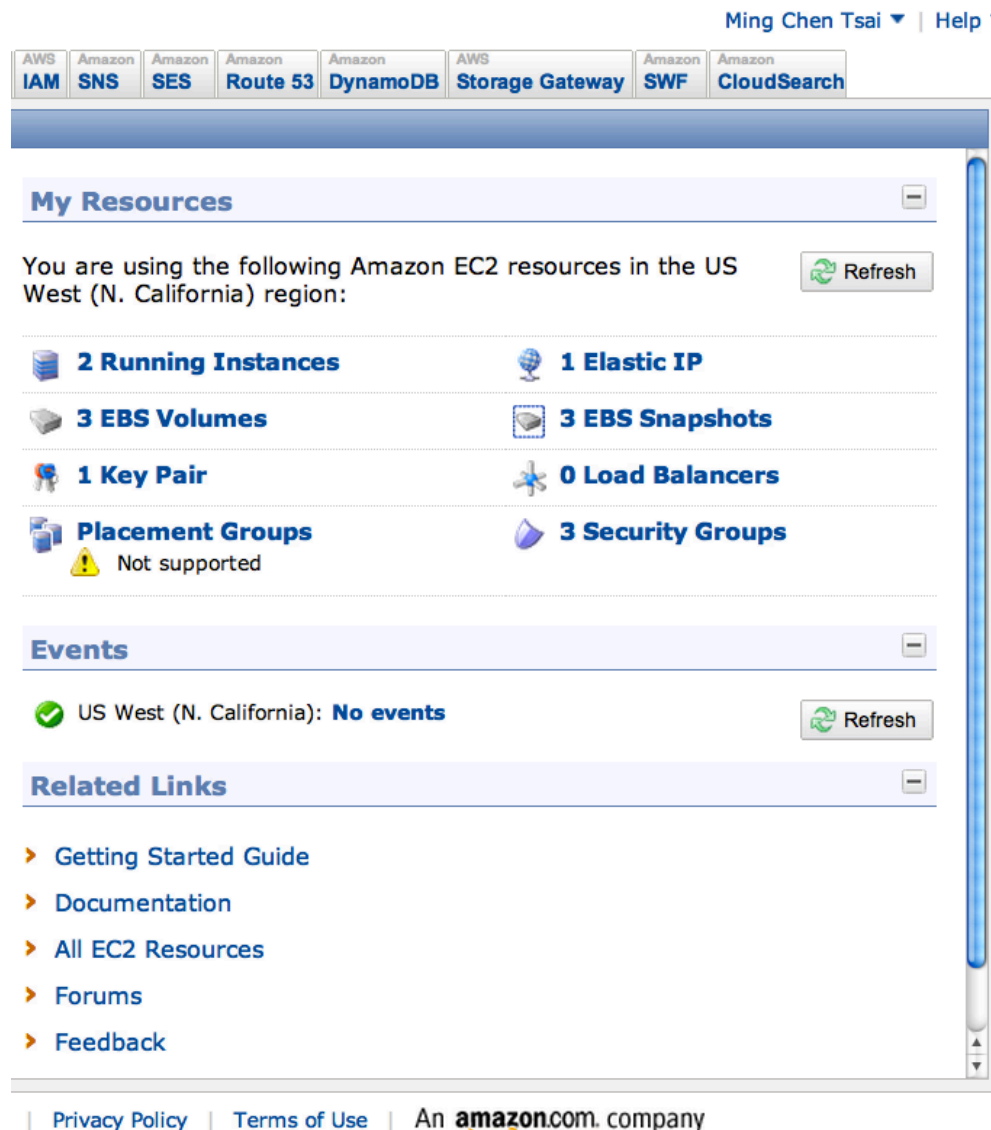


Figure 13 - Amazon AWS - Resources Panel

Amazon's AWS panel gives the user a one-place dashboard to view resource usage statistics. EC2 is in term what a server instance would be, EBS (Elastic-Block-Storage) is the storage unit that can be used for EC2 instances. I investigated into the various services offered by Amazon AWS, and analyzed the pros and cons of the different options.

The portal Resource Units uses the "Micro instance" which is free under usage hours 750 per month, and charges start to incur after the threshold is reached. Paid options exist which provides more memory and computing power for better instance types. MySQL database instance can be setup either on the

EC2 instance itself or through the offered special database service instance. The benefit of using their specially designed database instance is when performance improvement or scaling is needed; the instance panel can manage these needs easily by click of buttons to replicate instance across the datacenters or query caching. I chose to host the portal's database instance on a EC2 by myself due to the reason of gaining better understanding of how to tune the database configuration and options in order to achieve desired performances, and also to save from specialized database instance costs.

2.3.5 Service Deployments

The portal components are uploaded to Amazon, using EC2 instances and distributed datacenter solutions provided. The benefit of this approach is that low latency can be achieved between user client and sharing environment that is closely located to the user.



Figure 14 - Amazon AWS Datacenter Locations

Referencing the EC2 instances that were mentioned previously in Resource Units section. Consider the scenario where a portal unit is placed in Tokyo and another one is placed in Seattle. When a user in China requests access to the portal, host lookup turns to the portal server in Tokyo, and allows connection with lowest network latency to occur as opposed to server in Seattle. The frontend portal-landing page I have developed is capable of checking the user's current location to determine which datacenter is most suitable for assigning the new room for in order to have quickest image access speed.

AWS Management Console > Amazon EC2

The screenshot displays the Amazon EC2 Console Dashboard. On the left is a navigation pane with a 'Region' dropdown set to 'US West (N. California)'. The navigation menu includes 'EC2 Dashboard', 'Events', 'INSTANCES' (with sub-items: Instances, Spot Requests, Reserved Instances), 'IMAGES' (with sub-items: AMIs, Bundle Tasks), 'ELASTIC BLOCK STORE' (with sub-items: Volumes, Snapshots), and 'NETWORK & SECURITY' (with sub-items: Security Groups, Elastic IPs, Placement Groups, Load Balancers, Key Pairs, Network Interfaces). The main content area is titled 'Amazon EC2 Console Dashboard' and features a 'Getting Started' section with a yellow background. It contains the text: 'To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.' and a prominent 'Launch Instance' button. Below this is a note: 'Note: Your instances will launch in the US West (N. California) region.' The 'Service Health' section shows the 'Service Status' as 'Amazon EC2 (US West - N. California)' with a green checkmark and the message 'Service is operating normally'. It also includes a link to 'View complete service health details'. The 'Availability Zone Status' section shows two zones, 'us-west-1a' and 'us-west-1b', both with green checkmarks and the message 'Availability zone is operating normally'.

Navigation

Region:
US West (N. California)

EC2 Dashboard

Events

INSTANCES

Instances

Spot Requests

Reserved Instances

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Load Balancers

Key Pairs

Network Interfaces

Amazon EC2 Console Dashboard

Getting Started

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (N. California) region.

Service Health

Service Status

Current Status	Details
✓ Amazon EC2 (US West - N. California)	Service is operating normally

View complete service health details

Availability Zone Status

Current Status	Details
✓ us-west-1a	Availability zone is operating normally
✓ us-west-1b	Availability zone is operating normally

© 2008 - 2012, Amazon Web Services LLC or its affiliates. All rights reserved. | [Feedback](#) | [Support](#)

Figure 15 - Amazon AWS - Management Console

I uploaded the test environment portal to Amazon's AWS after local development and testing has completed. I packaged the component files into individual compressed archive files before transferring them onto EC2 instances. The Management Console snapshot above provides options to easily launch new cloud instances by choosing from Amazon standard system or community-based templates. These templates are in different Windows versions, and Linux flavors.

3. Portal Workflow

In this section, I will outline major workflows through the portal.

3.1 Enrollment

The diagram below outlines the components involved when a user registers or sign-up to the service. User profile data is gathered by the Frontend page and stored to a database behind it.

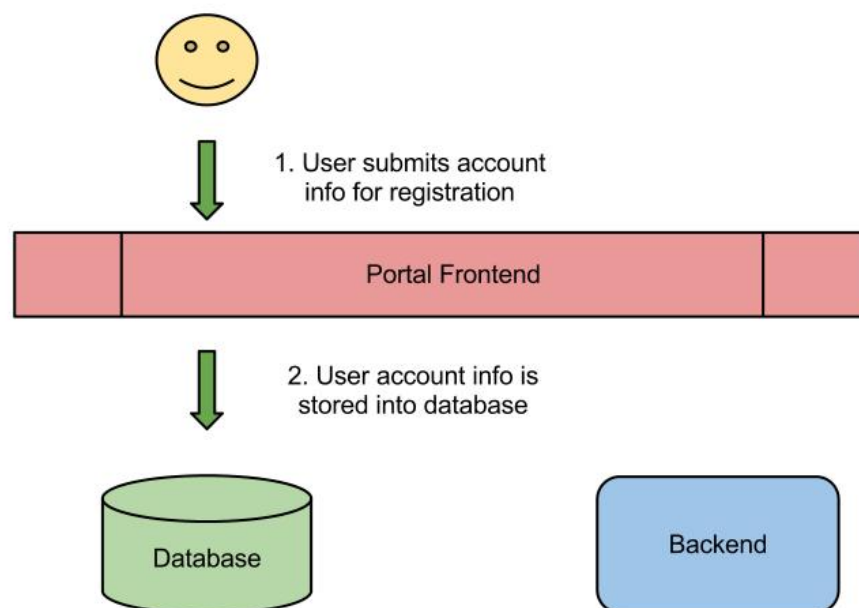
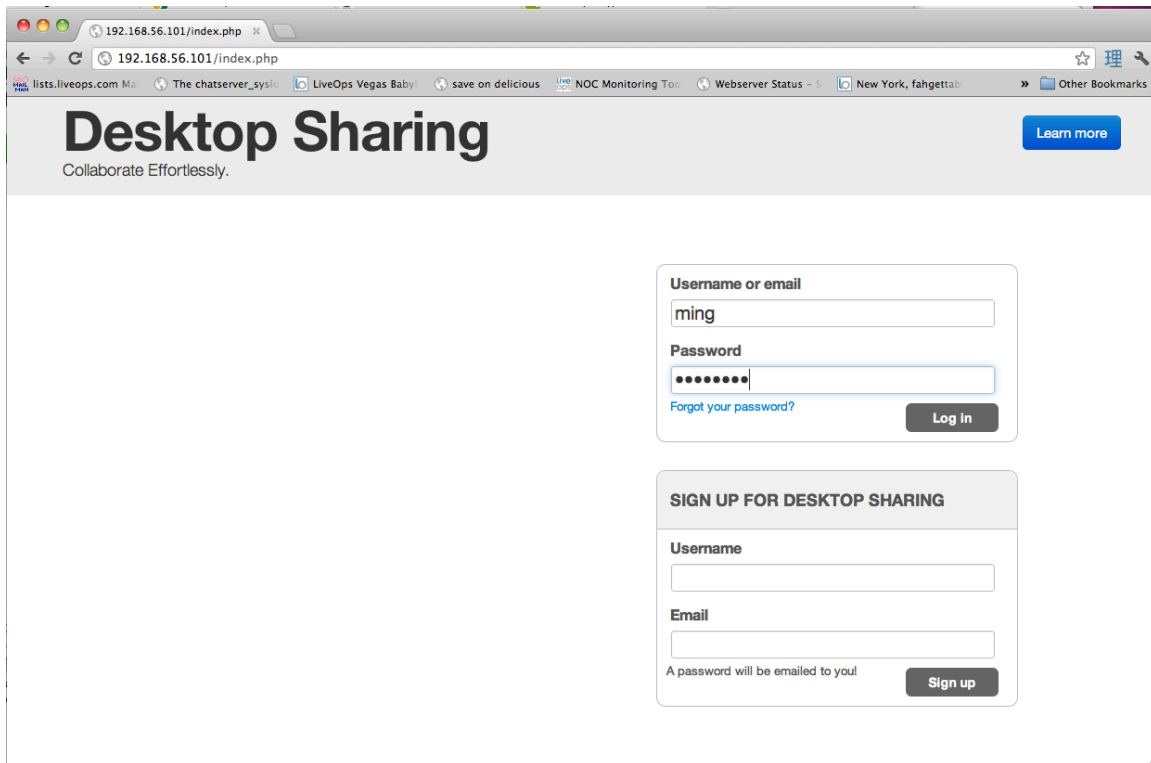


Figure 16 - Enrollment Workflow

List of expected user clients includes current prominent browsers on desktop computers, such as Internet Explorer or Firefox, or browsers on mobile devices such as iPad. The minimum requirement is a HTML5 compliant browser.

The portal-landing page is displayed with 2 main options - login or sign-up to continue portal usage. Depending on the user's current membership status, he/she would proceed to fill out the top portion of the page for login action. If the user does not have an account, the bottom portion allows registration process to occur. After which, an email with login password will be sent to the user. Once the user logs into the service, he/she will be allowed to change the password.



The screenshot shows a web browser window with the address bar displaying "192.168.56.101/index.php". The page title is "Desktop Sharing" with the tagline "Collaborate Effortlessly." and a "Learn more" button. The main content area features two forms. The first form, titled "Username or email", has a text input field containing "ming", a "Password" field with masked characters, a "Forgot your password?" link, and a "Log in" button. The second form, titled "SIGN UP FOR DESKTOP SHARING", has a "Username" field, an "Email" field, and a "Sign up" button. Below the email field, it states "A password will be emailed to you!".

Figure 17 - Enrollment Screenshot

3.2 Creating a new room

The Backend module of the portal maintains the currently available rooms that are ready to be “owned” in the database. When a create room requests arrives, the Frontend checks the database to determine if usage room exists before granting room ownership to user.

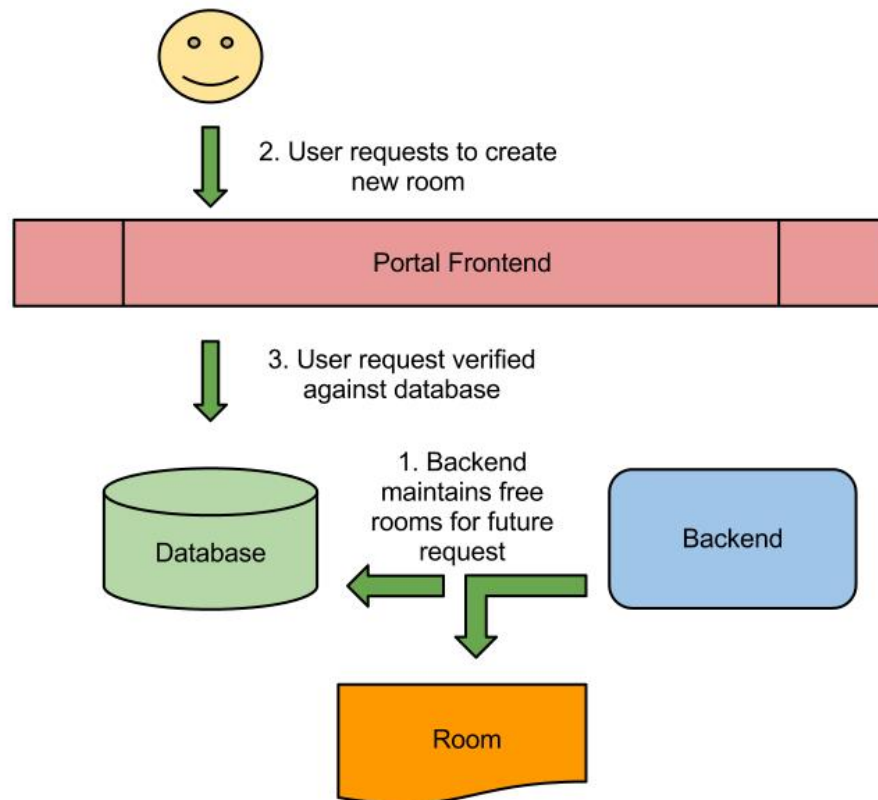


Figure 18 - Create New Room

The concept of a room here translates to a virtual image with remote desktop capability enabled.

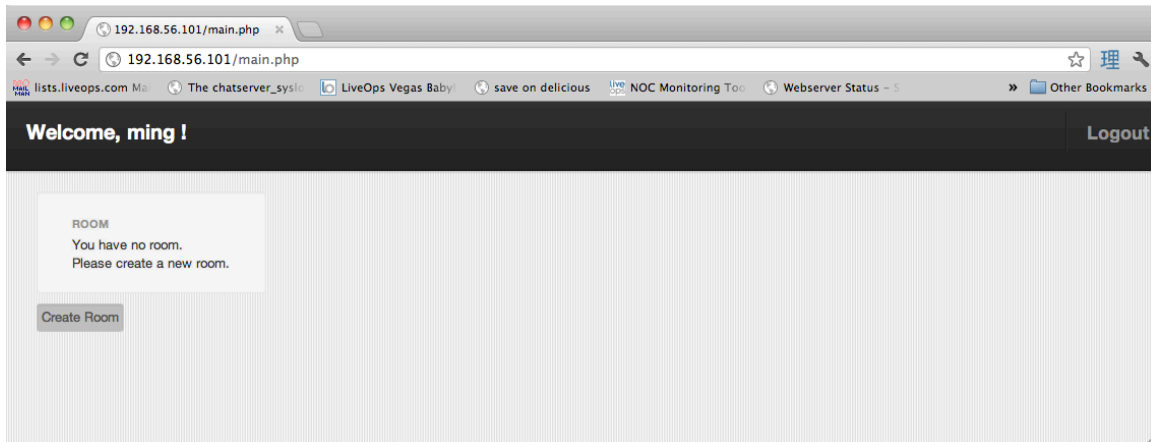


Figure 19 - Create New Room Screenshot

Figure below shows user “ming” owning 2 rooms – Room0 and Room1. Room1’s usage statistics section shows a “Current User” section. It shows that there are currently 2 users using the room – “testuser” and “tester3”. Note that the thumbnail screenshots of the Rooms refreshes every second to give up-to-date showing of what programs are currently opened.

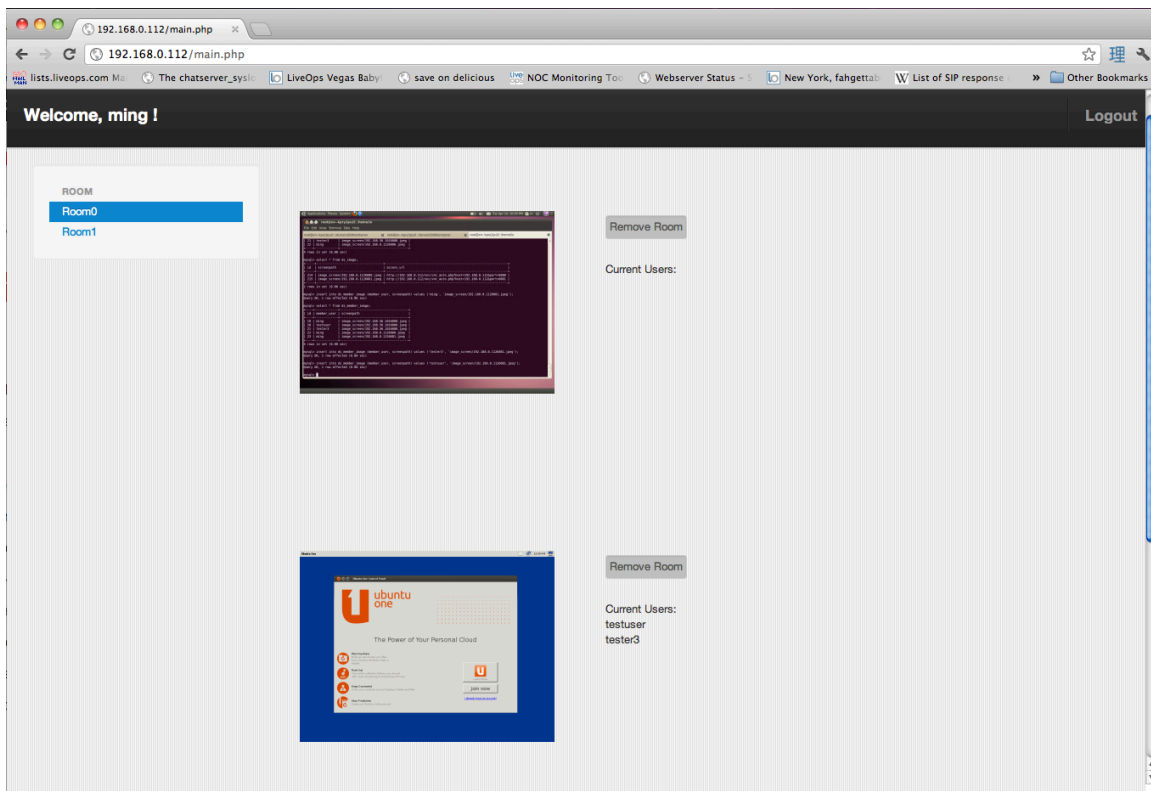


Figure 20 - Main Lobby Room Index Screenshot

3.3 Collaboration Room

Upon entering the Collaboration Room, the desktop environment is immediately ready for user to control. In the Collaboration Room, multiple users are allowed to control the desktop environment together in the page provided in Figure below. Desktop control is only given to the user who is giving the most recent update on keyboard or mouse activities.

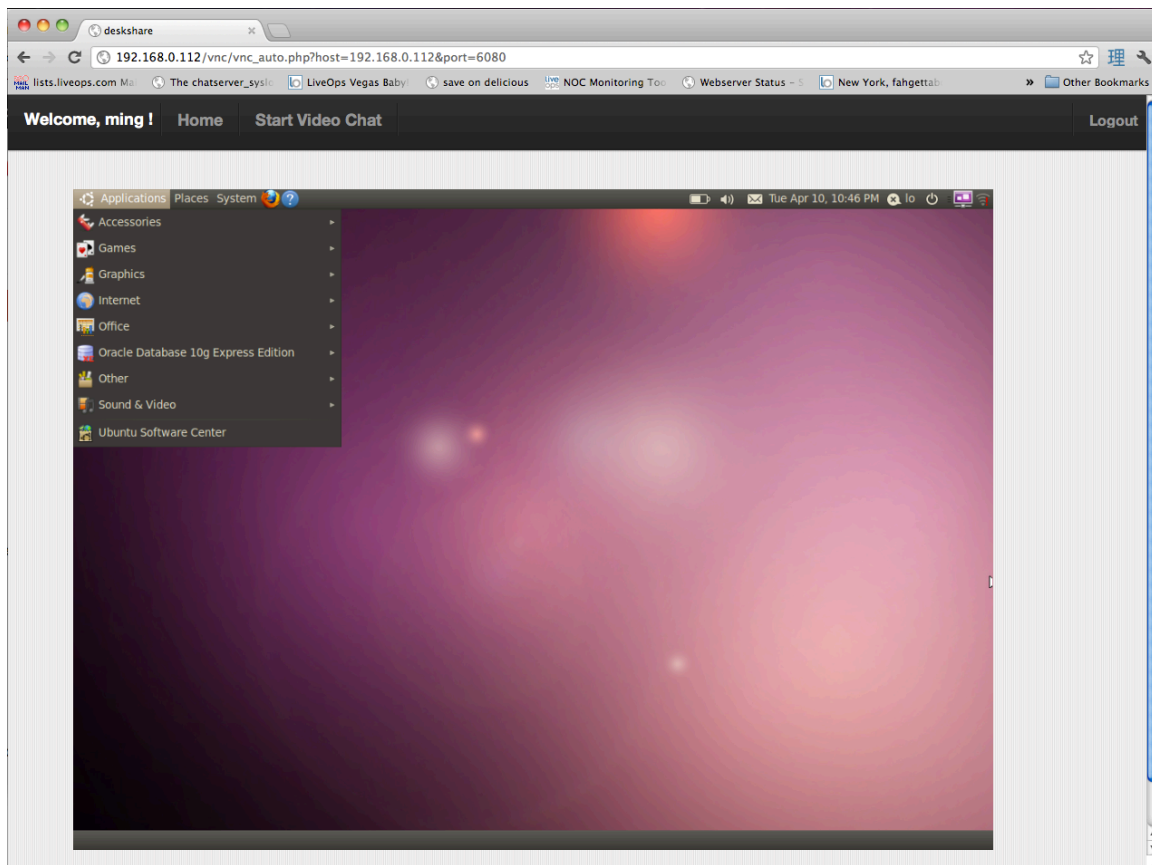


Figure 21 - Collaboration Room

Concerns on speed, latency and responsiveness regarding Room usage is addressed in Evaluation section.

3.4 Video conference

Users who enter a Collaboration Room can start or join an ongoing videoconference by clicking on the “Start Video Chat” button at top of page. Every Room is equipped with a unique Video Chat session id. Note that the actual desktop sharing data channel is separate from the videoconference data channel. In that sense, if videoconference transmission fails due to Video server issue, desktop sharing would still continue.

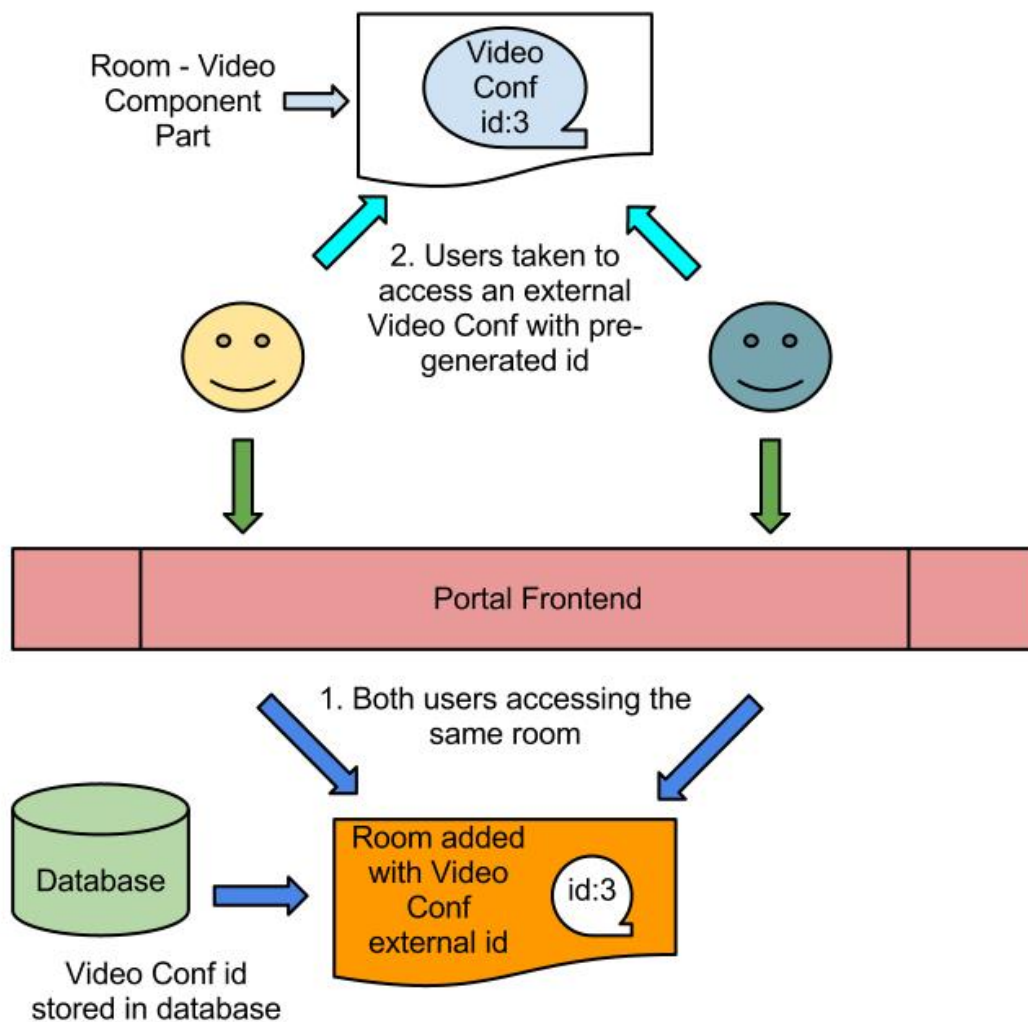


Figure 22 - Video Conference Workflow

The Figure below shows an ongoing desktop sharing session with 2 videoconferencing users. The number of desktop sharing participants is at least 2, whose video screens are shown, but possibly more if they do not have Video Chat turned on. Users are free to join and leave the room as they desire. No disruption occurs to members in the room when any participant leaves.

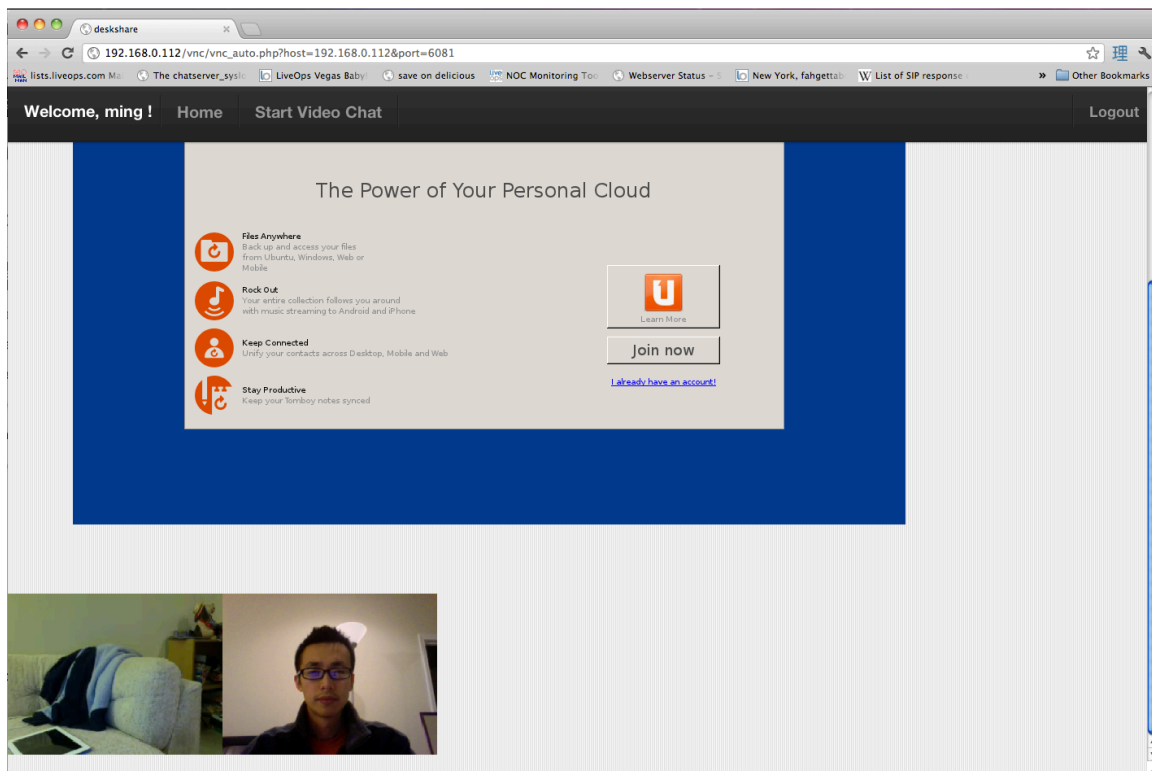


Figure 23 - Room with Video Chat

3.5 Audio streaming

In addition to desktop sharing and videoconference data channels, Audio Streaming provides audio that is sourced from the sharing environment through yet another transmission channel. Similar to videoconference, the unique Audio Streaming session ID is tied to the room. All the users who requests for Audio Stream that are in the same Room are given the same audio stream.

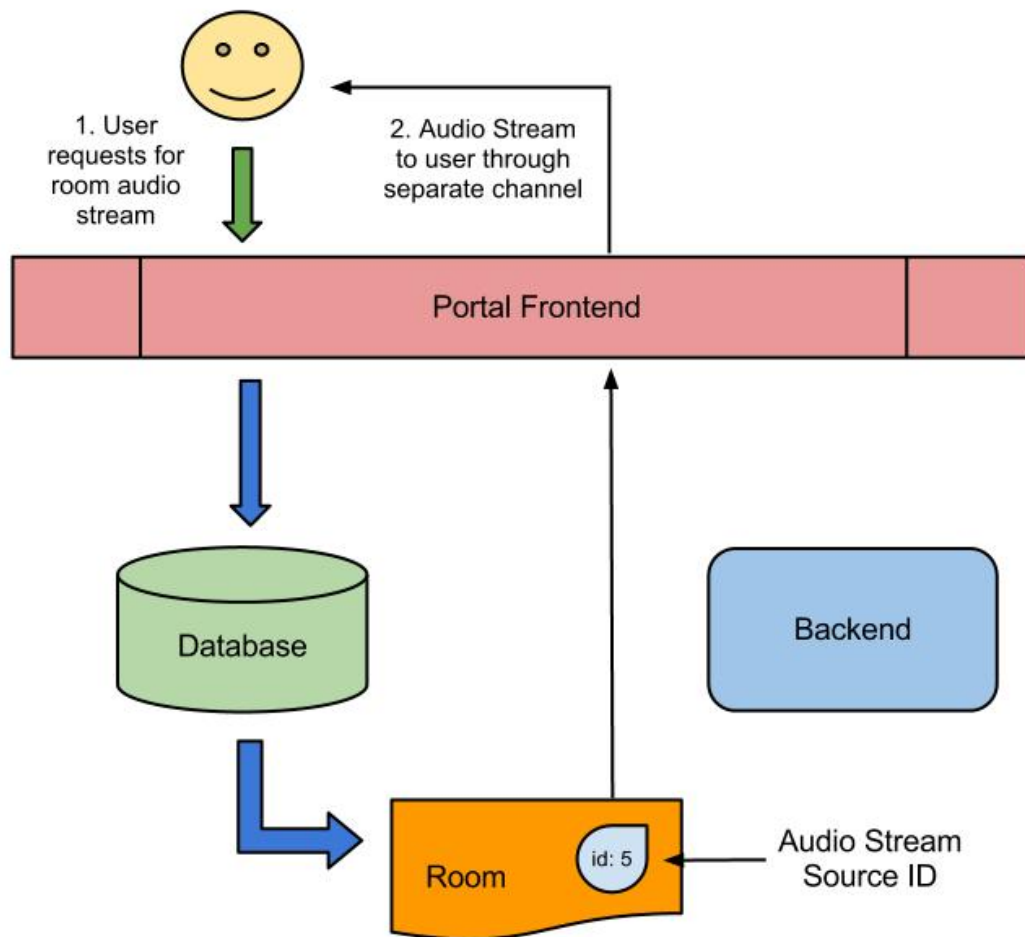


Figure 24 - Audio Streaming Workflow

4. Evaluations and Usability

User evaluations were conducted on typical Internet users for duration of less than an hour for each session. Many questions were raised from these sessions. The following is the list of items that was accepted and contributed to revamp of the portal and pushed the portal to its current form:

4.1 Sharing Environment

The sharing environment is found to be easy-to-use by most users. With the exception of users on mobile devices where they touch-based controlling and smaller screen-size might be more difficult for them to participate

- After a room is created, how can it be easily shared to my social profiles?
 - In the portal's current form, simply by copying the Address Bar Url will be sufficient. Social profile integration can also be achieved to enable this. This option shall be explored as a further research item.
- Do I must have Adobe Flash player installed in order to start/join a videoconference?
 - The current implementation relies on TokBox, which unfortunately is a Flash-based solution. Google's WebRTC [25] is a project that could change that. One of its goals is to remove the need for 3rd-party plug-in installation for web-based multimedia content.

4.2 Video Conference Popup

A minor annoyance brought up by most users is the “Adobe Flash Player Settings” popup when video chat is started. There is unlikely to be a way around this popup. This is a mandatory check is required on user side for preventing any unwanted usage of client's device hardware. This could possibly be removed with a global system wide permission setting that eradicated the need for approving on each device access instances.

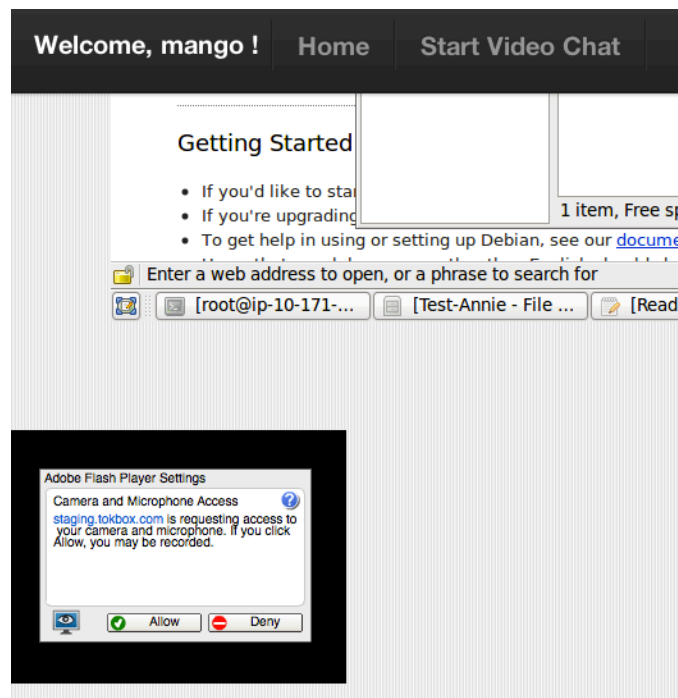


Figure 25 - Video Device Access Request Popup

4.3 Audio Stream Delay

Due to the multi-layer components involved in delivering audio stream, multiple staged buffering is causing a long delay of audio streamed. This resulted in non-synchronized desktop multimedia experience, where the audio is often 10 seconds behind the display. This remains an issue to be addressed in the current portal, but otherwise is acceptable to participants who do not require display to be synchronized with audio stream.

5. Conclusion

In this project, I have explored and implemented some of the frontier collaboration technologies that exist on the Internet today. By analyzing the various remote display protocols, implementing the newly evolved HTML5 web standards, leveraging Amazon AWS' cloud infrastructures, and incorporating early-stage Internet videoconferencing service. The portal is built with careful considerations using the latest and up-to-date Internet technologies.

The Desktop Sharing Portal I developed in this research is designed to be a collaboration tool on the Internet. The portal presented provides an easy-to-use web based social environment where desktop-sharing activities can occur. Compared to other sharing services, the portal is a smarter infrastructure that places concern on user's experience.

The advantages include - no technical setup steps required, collaboration room access speedup by serving a room quickest to the user's location, real-time statistics on Room statuses, multi-party conferencing capability during ongoing sharing session, and audio streaming. In summary, the presented portal provides a platform that allows desktop sharing collaboration in an efficient and convenient manner.

5.1 Future Research

Google WebRTC integration for audio and video conferencing capability without requiring 3-rd party plug-in would be a beneficial improvement that could be made to the portal. But this will be largely dependent on what WebRTC evolve to. Social Profile Integration with 3-rd party websites can also be considered to allow easy user lookup and Room activity information sharing.

References

- [1] Tae-Ho Lee, Hong-Chang Lee, Jung-Hyun Kim, and Myung-Joon Lee, "Extending VNC for effective collaboration," *Strategic Technologies, 2008. IFOST 2008. Third International Forum on*, 23-29 June 2008, page 343-346.
- [2] VNC Client using HTML5 (Websocket, Canvas).
<http://kanaka.github.com/noVNC/>
- [3] HTML5 Client-less Remote Desktop. <http://guac-dev.org/>
- [4] RealVNC. <http://www.realvnc.com>
- [5] The RFB Protocol. <http://www.realvnc.com/docs/rfbproto.pdf>
- [6] OpenTok API. <http://www.tokbox.com/>
- [7] Microsoft Remote Desktop Protocol. "MSDN Network Documentation"
<http://msdn.microsoft.com/en-us/library/aa383015.aspx>
- [8] SPICE Protocol. http://spice-space.org/docs/spice_protocol.pdf
- [9] Amazon Web Services. <http://aws.amazon.com/>
- [10] VMWare. "Understanding Full Virtualization, Paravirtualization, and Hardware Assist."
<http://www.vmware.com/resources/techresources/1008>
- [11] "Comparison of Remote Desktop Software", Wikipedia.
http://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software
- [12] Open Source Relational Database Management System. MySQL.
<http://mysql.com/>
- [13] Live Audio Streamer. <http://darkice.org/>
- [14] Streaming Media Server. <http://www.icecast.org/>
- [15] The Network Computing Company. <http://nomachine.com/>
- [16] High Speed Remote Desktop. <http://blog.x-row.net/?p=47>
- [17] Citrix XenDesktop.
[http://www.citrix.com/English/ps2/products/product.asp?contentID=163057
&ntref=prod_cat](http://www.citrix.com/English/ps2/products/product.asp?contentID=163057&ntref=prod_cat)
- [18] Microsoft Windows Server 2008 Remote Desktop Services Features.

- <http://www.microsoft.com/en-us/server-cloud/windows-server/remote-desktop-services-features.aspx>
- [19] YouOS. <http://www.youos.com/html/static/manifesto/what.html>
- [20] Synaptop. <http://www.synaptop.com/>
- [21] Kenneth Van Surksum, "Microsoft RemoteFX for VDI Architectural Overview," 17 Feb 2011,
<http://www.microsoft.com/download/en/details.aspx?id=13864>
- [22] Google+ Hangouts.
<http://www.google.com/tools/dlpage/res/talkvideo/hangouts/>
- [23] VoxWire Webconferencing. <http://www.voxwire.com/>
- [24] FlashMeeting Webconference. <http://flashmeeting.e2bn.net/>
- [25] Google WebRTC. <http://www.webrtc.org/>