Product Design Specification

for

p2pcast

Version 1.00

Prepared by

Group Name: NP Compete

Daniel Vicory	7024755	dvicory@gmail.com
Omar Masri	8646101	marismaro@gmail.com
Jerry Medina	4349882	jerryfoxyt@gmail.com
Nicole Theokari	5179528	nicole@theokari.com
Justin Liang	5217286	justinyliang@gmail.com

Instructor:	Chandra Krintz
Course:	CMPSC189A
Lab Section:	*Wednesday 6:00-6:50PM*
Teaching Assistant:	*Geoffrey Douglas*
Date:	3/4/14

Version	**Primary Author(s)**	**Description of Version**	**Date Completed**
1.01	Daniel Vicory, Omar Masri, Jerry Medina, Nicole Theokari, Justin Liang	Initial draft	3/4/2014

1. Introduction

1.1 Product Overview

p2pcast is a browser-based method of broadcasting video using peer-to-peer technologies. Its purpose is to allow users to stream video easily, quickly, and free of cost. There are two distinct components that make up the p2pcast platform. There is a component that runs in the user's web browser and another component that functions as a server to enable connectivity between users' web browsers.

Current services that provide a similar functionality include Skype and Twitch.tv. Skype uses an external program for its peer-to-peer streaming video. Twitch.tv is a streaming website which uses dedicated centralized servers to support its massive audience. p2pcast proposes to create a simpler de-centralized service to use used with no additional installments and with the propose of being multiplatform. Multiplatform will rapidly grow as support in existing browser and subversion for mobile continue to extend their support of WebRTC.

1.2 Definitions, Acronyms, and Abbreviations

Term	**Definition**
Web Application	The component that runs on end-user's web browsers, composed of HTML JavaScript and is also the part that talks to other peers directly
Application Server	The component that runs on the host, used to index channels, facilitate connections, and serve the web application to web browsers
Broadcaster	A peer who originates video streams to users
Channel	A method of namespacing different broadcaster's video streams. Channels are created by a broadcaster which contains only their own video stream. Users can j a channel to view that broadcaster's video stream, in which they become a peer f that specific channel.
Citrix Online	The online services division of Citrix Systems, Inc.
End-user	A person who uses the p2pcast web application, whether to broadcast or view vi streams
Forwarding/	The process of one peer in a network transmitting data it is receiving from anoth

Rebroadcasting	peer in the network to at least one other peer
Google Chrome	A web browser developed by Google, the primary target of our p2pcast web application
Host	User that runs a p2pcast application server
ICE Framework	ICE is a framework used to connect peers. First tries UDP, then TCP with HTTP then TCP with HTTPS, then TURN servers.
ICE Candidate	An ICE candidate is a network interface and port of a peer that is using the ICE framework
JavaScript	A dynamic computer programming language, the primary development of p2pca done in
NAT	A network protocol used in IPv4 networks that allows multiple devices to conne a public network using the same public IPv4 address.
Node.js	A platform built on Chrome's JavaScript run-time, V8, for easily building fast, s network applications
p2pcast	A web application that allows for peer-to-peer video broadcasting
Peer	A browser that is made available to be connected to by other peers, can be a broadcaster or user
Peer-to-peer	A method of communication, where most data is transmitted between end-users instead of centralized servers
PeerConnection	An object from the RTCPeerConnection API
SDP	Session Description Protocol (SDP) is a format for describing streaming media parameters used in signaling
Signaling	A process to exchange control messages and coordinate communication between peers
Socket.IO	A JavaScript library for real-time web applications
SRS	Software Requirements Specification
Stream	A sequence of data provided in a forward iterable-only manner
STUN	Session Traversal Utilities for NAT (STUN) is a protocol that uses a third-party STUN server to allow peers to discover each other’s IP address even if they are

	behind a NAT
TCP	This provides reliable, ordered, error-checked delivery of a stream of octets betw programs running on computers connected to a local area network, intranet or th public Internet.
TURN	Traversal Using Relays around NAT (TURN) is a protocol that uses a third-part TURN server to allow peers to receive or transmit data over TCP or UDP connections even if they are behind a NAT
UDP	A simple transmission model with a minimum of protocol mechanism.
Web Browser	A software application for retrieving, presenting and traversing information reso on the World Wide Web
Web Server	Computer software that deliver web pages to web browsers which may consist o static and dynamic (JavaScript) components
WebRTC	A W3C draft standard that enables Real-Time Communications (RTC) capabiliti for web browsers via simple JavaScript APIs

1.3 Technologies Used

1.3.1 JavaScript

JavaScript will be the primary core of the p2pcast application, using ECMAScript 5 features within the web application, and ECMAScript 6 (Harmony) features within Node.js on the server, as applicable. WebRTC APIs are made available from JavaScript and all client-side scripting requires JavaScript, so it is a major component of p2pcast.

Primary resource: https://developer.mozilla.org/en-US/docs/Web/JavaScript

1.3.2 Node.js

Node.js is a runtime platform built on Chrome's JavaScript runtime, V8, for easily building fast, scalable network applications. Node.js contains many modules that allow developers to do system-level operations that one is not able to do in JavaScript on the web browser. Node.js will be used to power the application server for p2pcast, handling such things as message routing, tree management, etc.

Target versions: 0.10.26 and 0.11.x

Primary resource: http://nodejs.org/api/

1.3.3 Bootstrap

Bootstrap is a set of predefined CSS classes that make styling much easier. Bootstrap provides the default settings for typography, tables, forms, and buttons. Bootstrap also provides reusable components (navigation bars, pagers, and progress bars) as well as scriptable widgets (tooltips, tabs, and picture carousels). Since Bootstrap is all CSS and JavaScript you can use Bootstrap with any server technology or development environment. The idea behind Bootstrap is to get a easily create a webpage that has consistent and visually appealing styling, without being a designer. Since p2pcast's focuses lie on the technical aspect, using Bootstrap to not get distracted is a natural choice.

Primary resource: http://getbootstrap.com/

1.3.4 Socket.IO

Socket.IO is a JavaScript library designed for real-time web applications. It consists of two libraries, one of which is used on the web browser and the other on the server for Node.js. Socket.IO will provide all client and server communication.

Targeted versions: 0.9.x

Primary resource: https://github.com/learnboost/socket.io

1.3.5 Heroku

Heroku is a cloud platform used for deployment of web applications in several programming languages. For quick, universal, and repeatable deployment, p2pcast will use Heroku.

Primary resource: https://devcenter.heroku.com/

1.3.6 WebRTC

WebRTC is a W3C working draft specification that enables web browsers to have Real-Time Communication between each other via simple JavaScript APIs. It defines primarily three APIs, MediaStream, RTCPeerConnection, and RTCDataChannel.

Targeted versions: 1.0, draft

Primary resources: http://www.webrtc.org/, http://dev.w3.org/2011/webrtc/editor/webrtc.html

1.4 Team Contact Information

Chandra Krintz (Professor)

ckrintz@gmail.com

Daniel Vicory

dvicory@gmail.com

Nicole Theokari

nicole@theokari.com

Omar Masri

marismaro@gmail.com

Jerry Medina

jerryfoxyt@gmail.com

Justin Liang

justinyliang@gmail.com

2. Design Overview

2.1 Components

p2pcast is composed of two separate but highly integrated projects: the client (web application) and the server (application server). The major components consist of channel managing mechanisms (global and individual), tree management, and other peer/user management.

2.1.1 Client

Each client acts as a peer, whether a broadcaster or a viewer. A peer will rebroadcast their received stream to other peers that connect to them. The connection of peers is dictated by the server, which facilitates the required WebRTC protocol handshakes. As a client, there is little local control for acting as a peer for other clients, which supports the ease of use and transparency for underlying technology.

The client, due to its peer to peer nature, actually shares some of the same components of the server. This lends itself well to code reuse.

2.1.2 Server

The server's main purpose in p2pcast is to function as a signaling hub. As the WebRTC specification leaves open the specifics of setting up WebRTC connections with other peers up to the implementer, it requires a middleman. In our case, p2pcast consists of an application server component which functions as that role. The p2pcast application server facilitates connections between peers that connect to it. If a peer disconnects, it is the server's job to make use of the tree to find new peers for all affected peers.

Beyond that, it must also namespace users between channels, keeping their trees separate. Additionally, it also provides access control mechanisms for the creation and deletion of channels.

2.1.3 Channel Manager

The channel manager is responsible for creating a channel that is addressable by a unique name, and also for making sure duplicate channels are not allowed. The channel manager has, or can acquire, a reference to all channels that exist and is also able to release channels into the available pool (say, by the request of the broadcaster or for some configurable inactive use).

The channel manager is responsible for routing messages to the correct channel. It also requires a way to inspect all of its channels (for example, browsing through all channels, or searching).

2.1.4 Channel

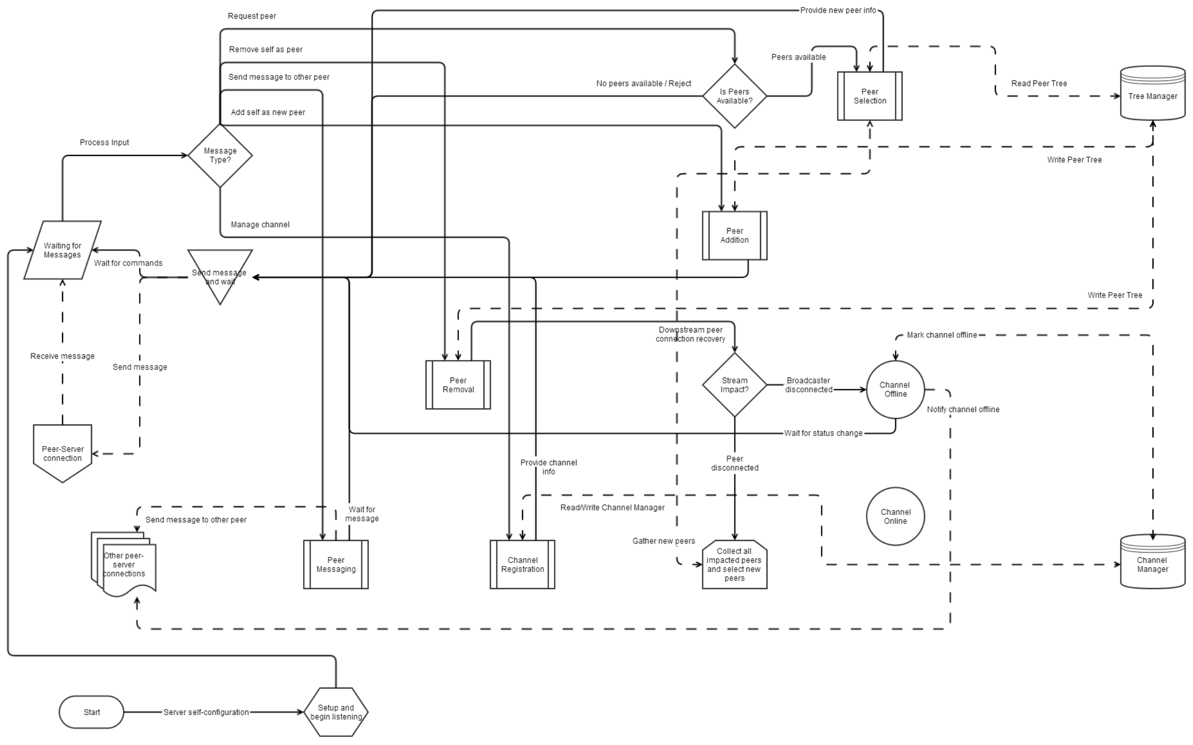
A channel is the logical wrapper for a broadcaster's video stream. It is uniquely named in the global namespace of all channels. It holds a reference to the tree for the peers on that channel as well as metadata about the channel, for instance, creation timestamps, and textual descriptions set by the broadcaster. Channels can be configurably leased by the host, where broadcasters who do not broadcast on their channel for some amount of time can be released to be claimed by other broadcasters.

Channels are also responsible for controlling access to the tree for which they abstract.

2.1.5 Tree Manager

Every channel has a tree manager that handles all connected peers, new peers, the broadcaster, etc. The tree manager is aware of the broadcaster peer and structures the tree such that "consumer" peers are placed below them. Additionally, the tree manager can internally structure the tree to optimize for peer conditions, such as latency between levels in the tree, and available bandwidth per peer. The specific arrangement of the tree is not defined, however, except for knowing who the top-level of the video stream is (the broadcaster) and the ability to find all peers who are affected by another peer disconnect, so that new connections for those affected peers can be facilitated.

2.2 Software Design



Non-exhaustive state diagram for server

2.3 Client (Web Browser)

The p2pcast web application targets the Google Chrome web browser for desktop and laptop. Chrome is required due to it being the browser with the most WebRTC features implemented, most notably stream rebroadcasting.

Users access the p2pcast web application hosted on Heroku and are greeted with the option to create their own channel and become a broadcaster, or join another channel to watch that broadcast. If they join another channel, the p2pcast application server facilitates a new WebRTC peer connection behind the scenes and starts the stream, without the user's explicit knowledge about how it works.

Other variations of the Chrome browser for mobile smartphones or tablets may be compatible. User's experience may vary wildly depending what portion of the WebRTC specification is implemented on these mobile versions.

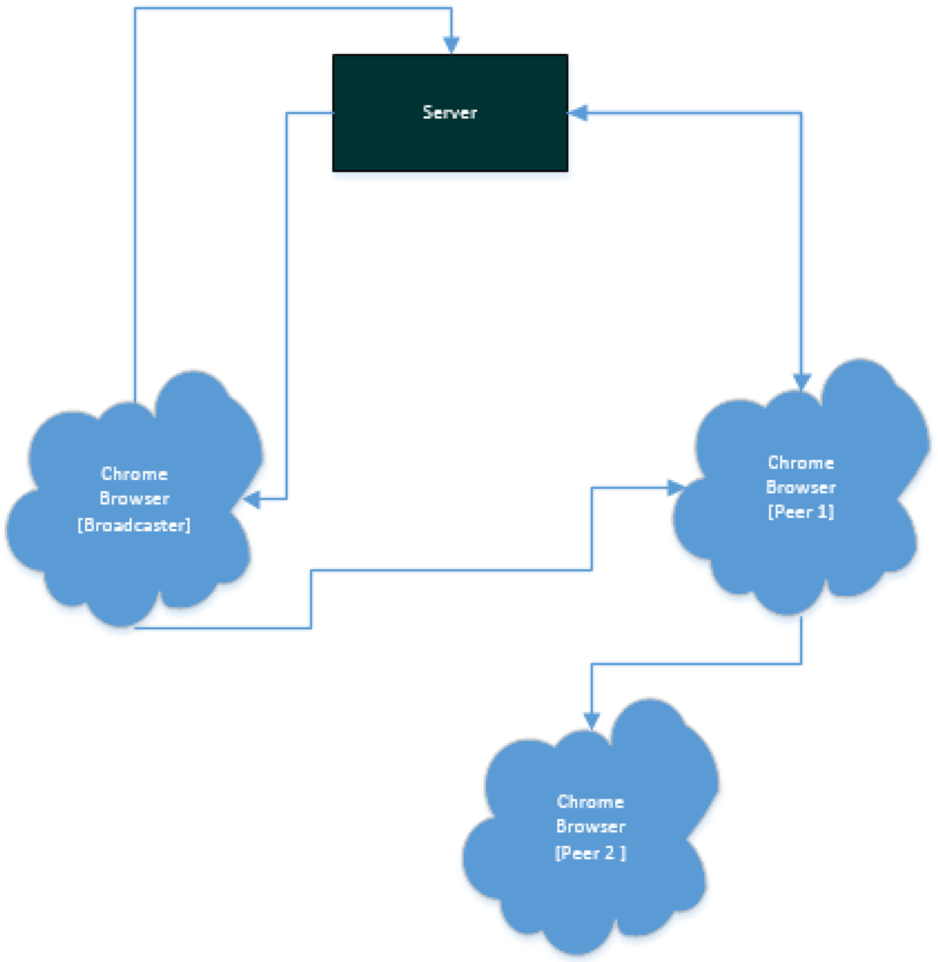
Each client will be designated as a peer, including the broadcaster. Peer disconnections signal an event to the server. In the case of sudden interrupt your downstream peers will signal the server about your disconnect. Peer membership in the tree is handled transparently, setting up and tearing down connections with minimal disturbance.

2.4 Server (Host)

The server is involved in all aspects of peer management, from the beginning of the lifecycle when peers connect to the end when they disconnect. Once the broadcaster has opened a channel and begun to stream and a new peer tries to connect to the channel, the server will facilitate the exchange of the WebRTC connection handshake to each of the two peers involved. Disconnection of a peer will require server intervention to reconnect one or more peers via the tree manager in a given channel. It is the server's duty to match peers together intelligently. For example, avoiding using newly joined peers as a rebroadcaster may be unwise since they are more likely to disconnect soon. The server should also know the optimal peer pairing, using the power of the tree manager.

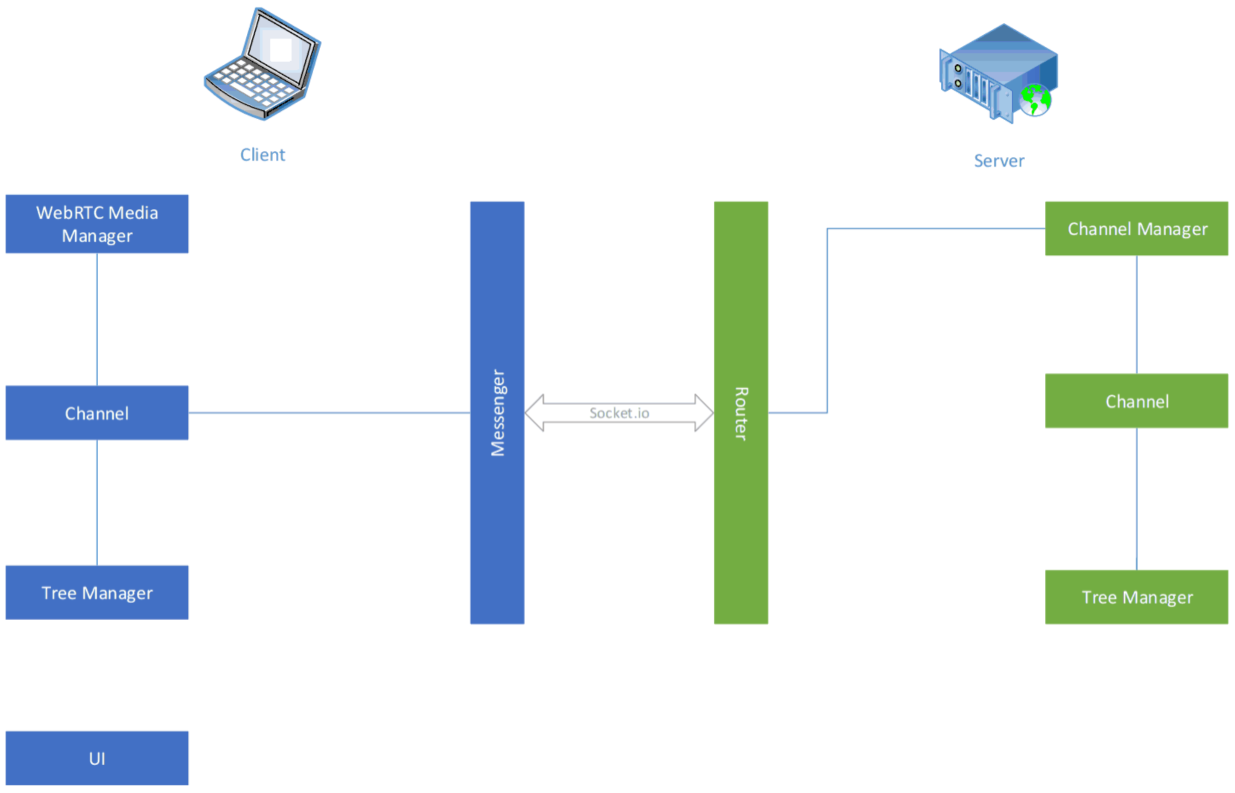
The server keeps a connection open to all peers currently using the p2pcast web application, exchanging status updates, and other such messages to gauge the quality of the connections, or for setting up and tearing down peer connections.

2.5 Data Flow Diagram



3. Design Specifications

3.1 High-Level Overview



The major components of p2pcast shown together

3.2 Channel Creation and Begin Broadcast

3.2.1 Successful Channel Creation for Broadcaster

Channel creation is successful if the user inputs a channel name that has either been leased to them or chooses one that is currently not leased to another broadcaster. The broadcaster will be given access to change metadata of the channel, such as channel description.

3.2.2 Failed Channel Creation for Broadcaster

Channel creation fails under various circumstances: the channel lease time has run out for that broadcaster, another broadcaster already has a lease on that same channel name, or there is an interruption in the p2pcast application server causing a loss of information. The broadcaster will be notified of the issue at fault.

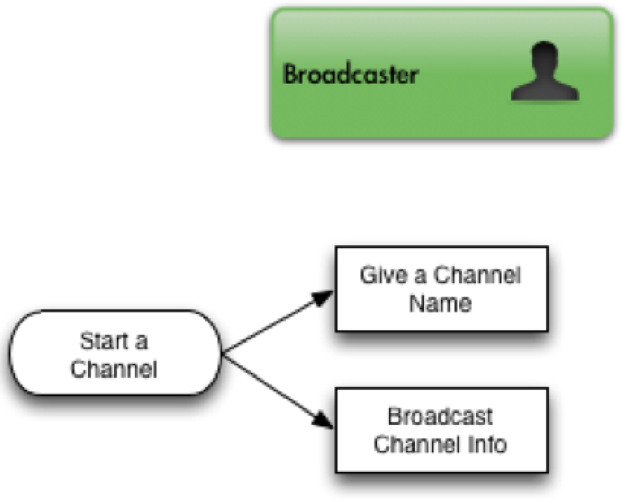
3.2.3 Successful Join Channel as a Viewer

Channel join is successful if the user inputs a channel name directly, or by selection from a list, and that channel exists. The channel selected should have whatever metadata the broadcaster has assigned. If the broadcaster is currently streaming, then the viewer will be able to see a video stream within a few seconds.

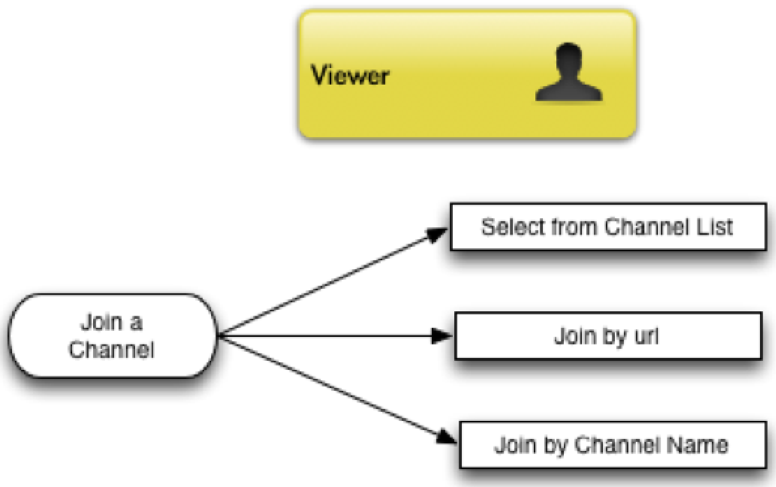
3.2.4 Failed Join Channel as a Viewer

Channel join fails under the various circumstances: the channel specified does not exist after selection, or has metadata that is not identical to what the broadcaster set. Channel join also fails if a video stream is not able to be started and does not indicate why to the user. The only valid reason for not being able to start a video stream is if the server decides that no peer has enough available resources to support another viewer.

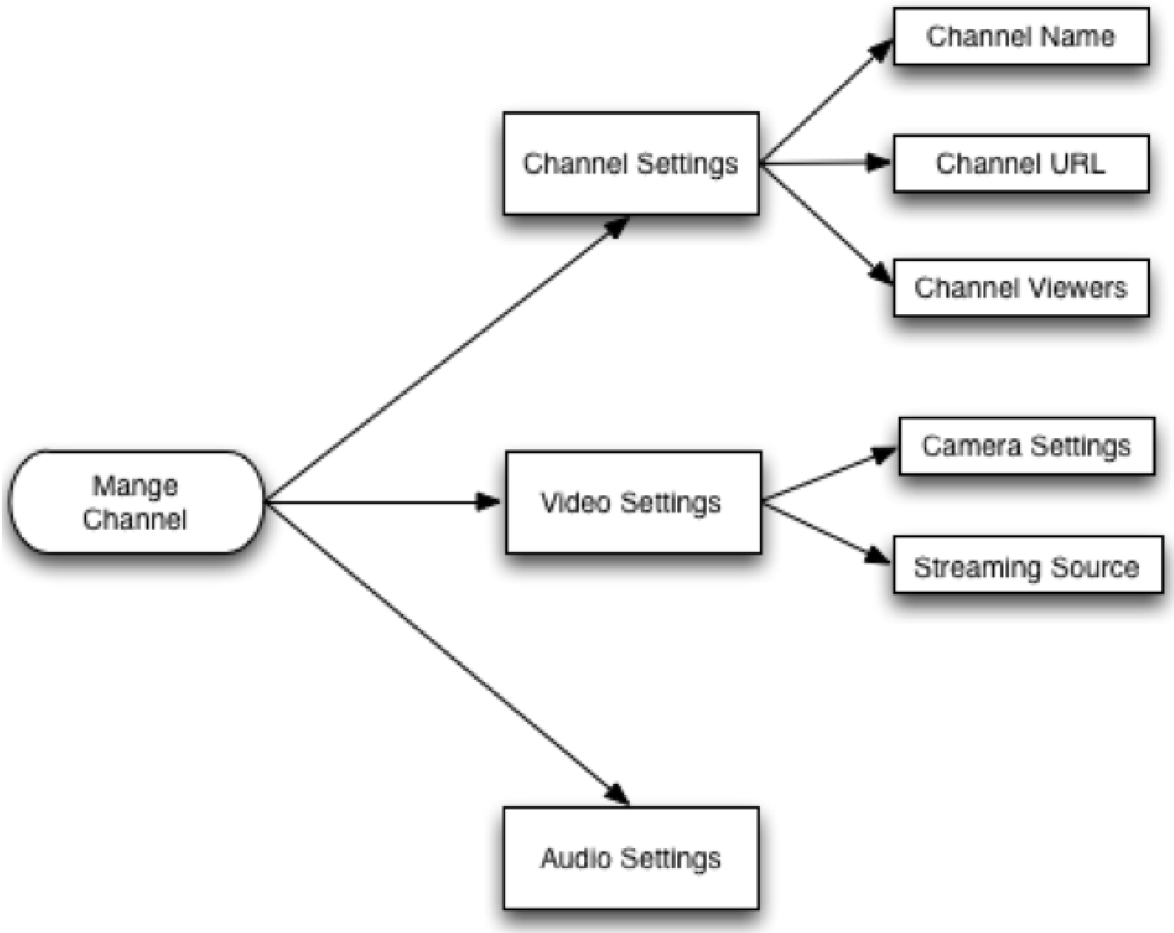
3.3 Detailed User Interaction Design



Broadcaster interaction



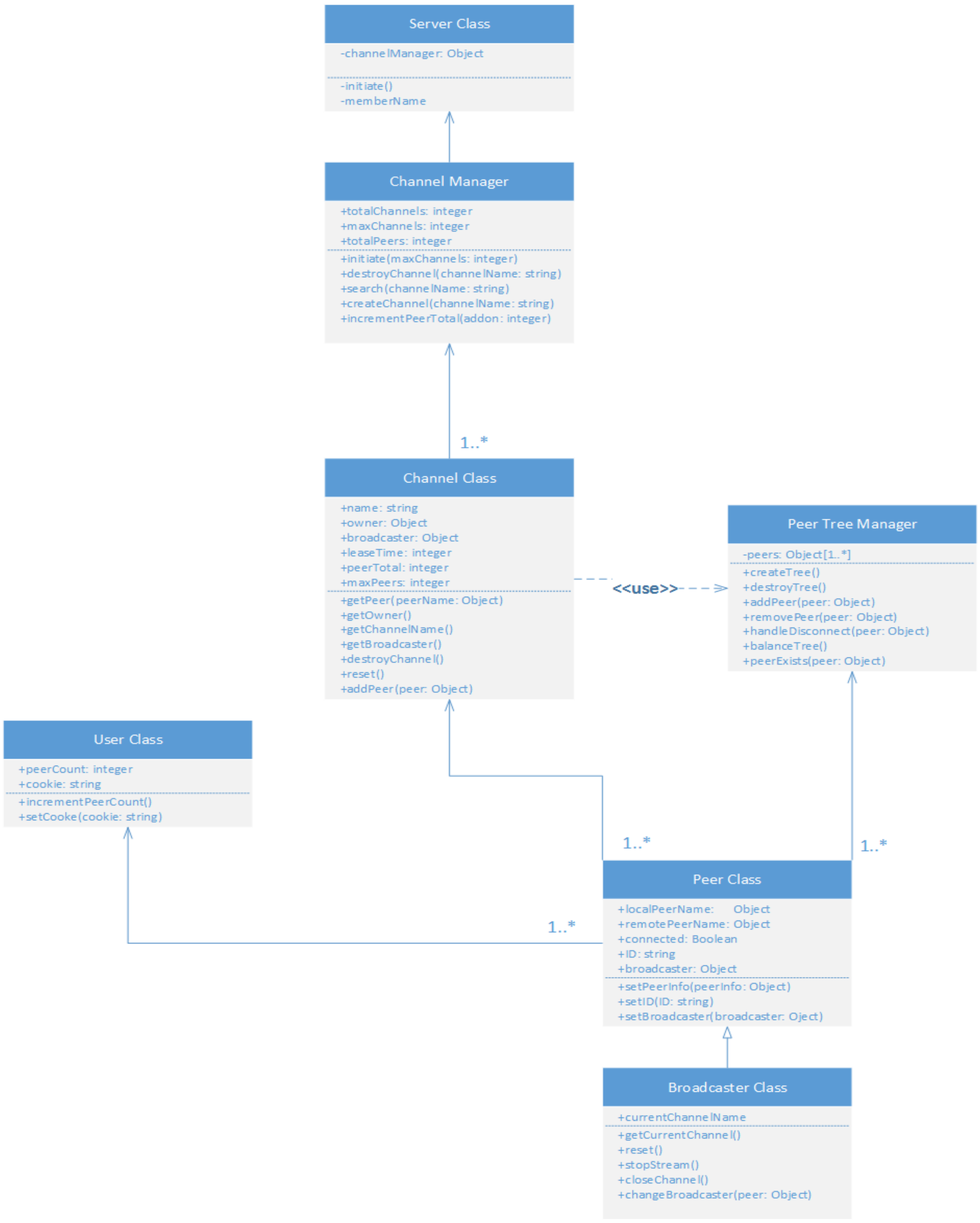
Viewer interaction



Channel management interaction

4. Diagrams and Technical Specifications

4.1 Class Overview



Overview of p2pcast classes abstracted from the client and server side.

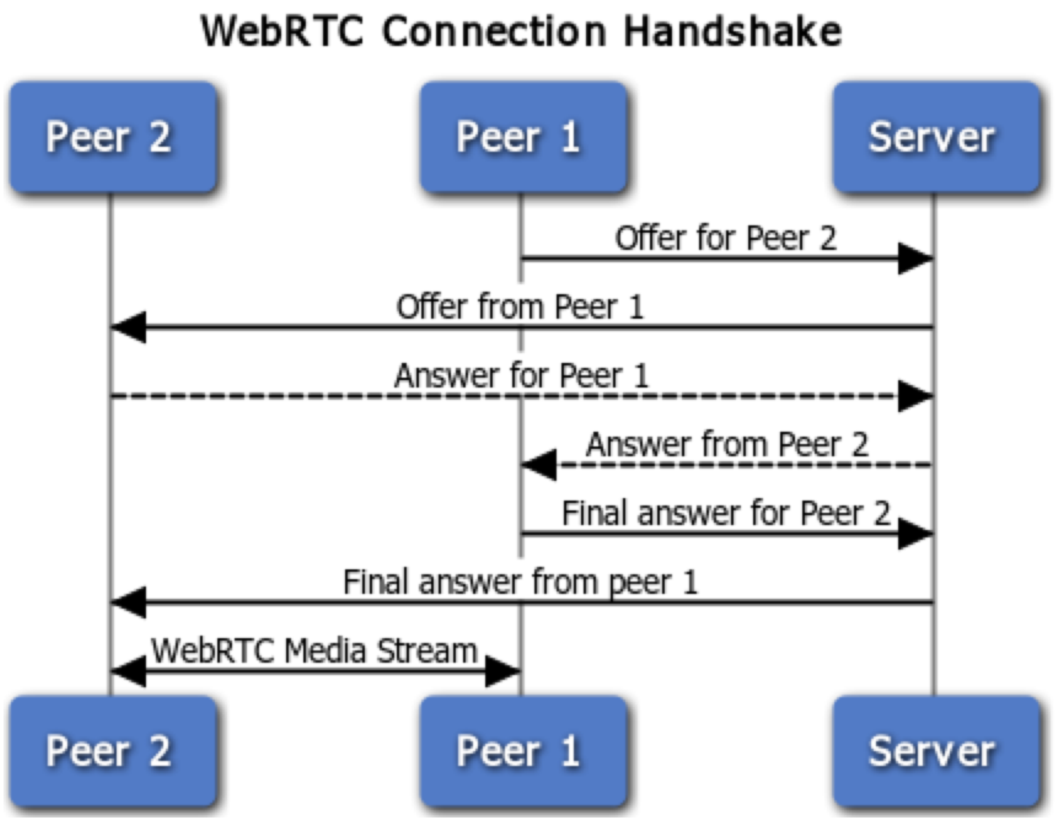
4.2 Client-Server JSON Interface Specification

Subset of client-server JSON specification with TypeScript interfaces. This is meant to be a guide demonstrating what kind and how messages between the client and server are designed.

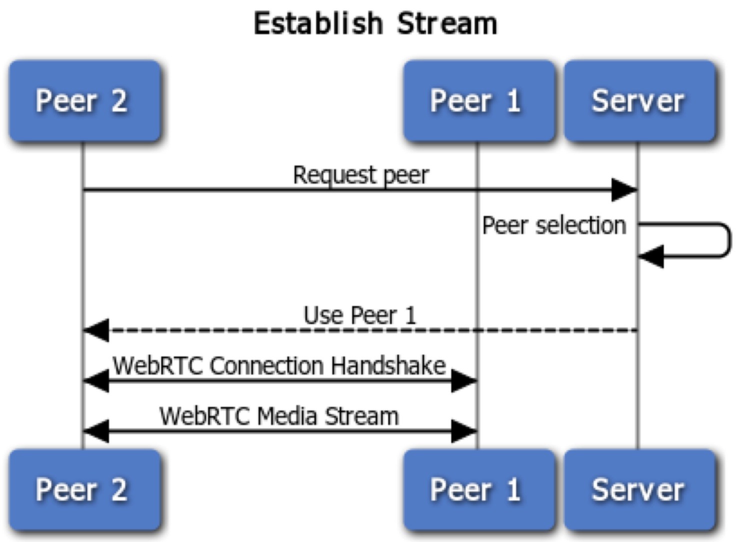
```
enum MessageEvent {
      Connect,
      Disconnect,
      RequestNewPeer,
      MessagePeer
}

enum PeerStateChange {
      PeerConnection,
      PeerDisconnection,
      PeerDegradation
}

enum WebRTCEvent {
      Offer,
      Answer,
      ICECandidates
}

interface Message {
      event: MessageEvent;
}

interface ConnectionSetup {
      type: WebRTCEvent;
      payload: any;
}

interface Peer {
      id: string;
}

interface PeerConnectionInfo extends Peer {
      upstreamPeers: string[];
      downstreamPeers: string[];
}
```

```
interface PeerTree {
      peers: { [key: string]: PeerConnectionInfo };
}

interface PeerEvent extends Peer {
      event: PeerStateChange;
}

interface Connect extends Message {
      cookie: string;
}

interface Disconnect extends Message {

}

interface RequestNewPeer extends Message {
      reason?: PeerEvent;
      peerTree?: PeerTree;
}

interface MessagePeer extends Message, ConnectionSetup {
      destination: Peer;
      data: ConnectionSetup;
}
```

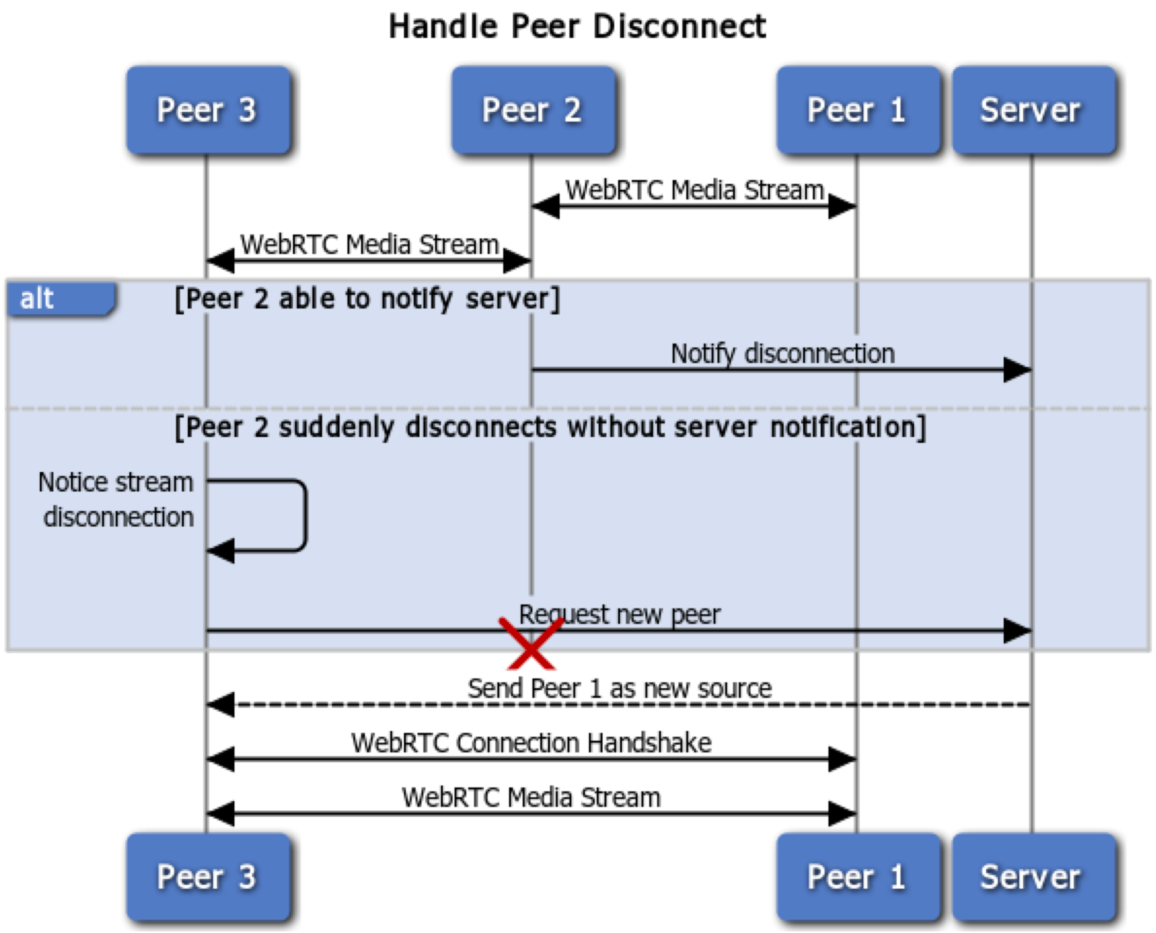
4.3 Sequence Diagrams



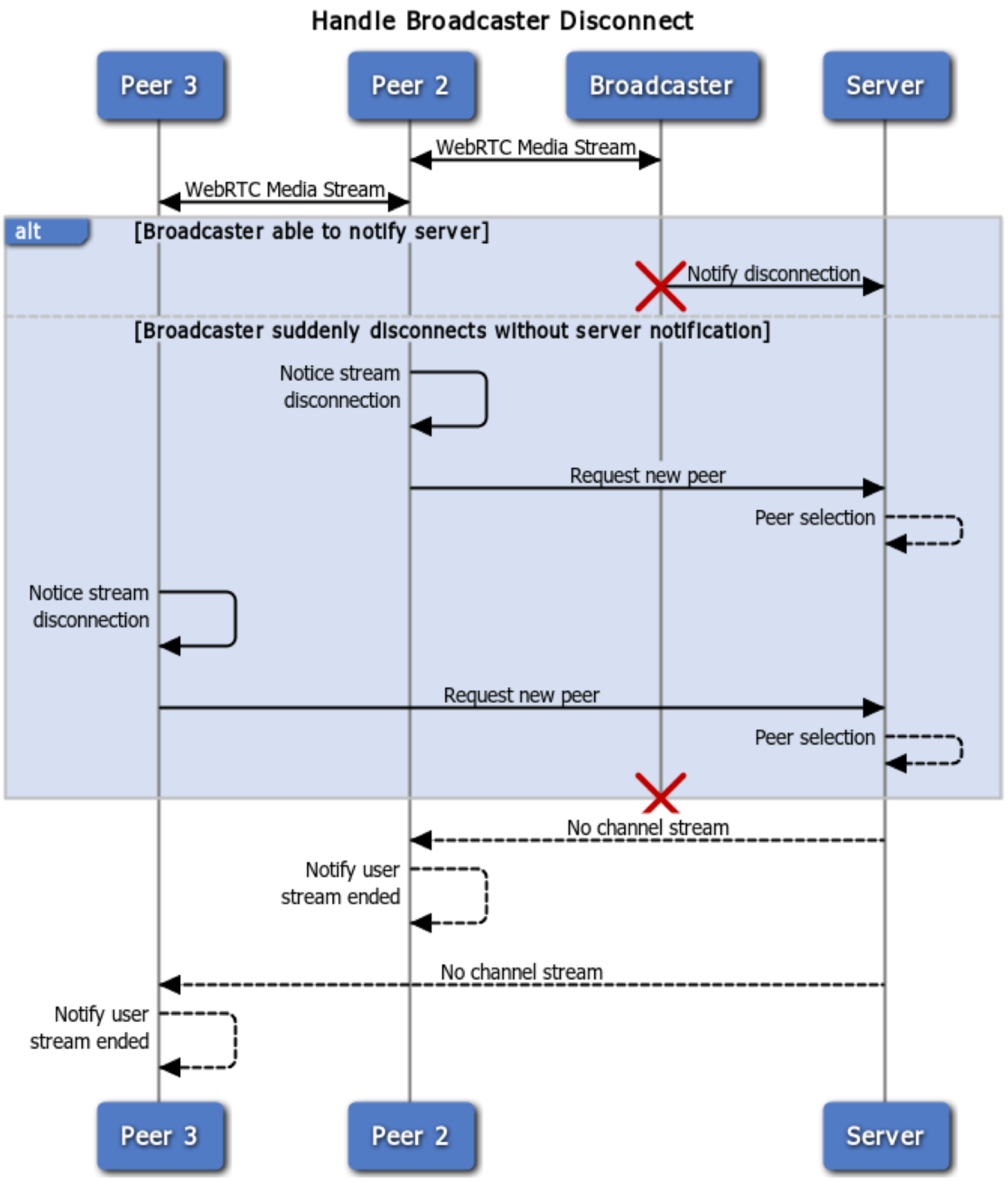
WebRTC Connection Handshake: The WebRTC three-way handshake. This will abstract the WebRTC connection setup in the following diagrams.



Establish Stream: A new viewer connecting and wishing to view a broadcast for a channel.


Handle Peer Disconnect
Peer 3
Peer 2
Peer 1
Server
WebRTC Media Stream
WebRTC Media Stream
alt
[Peer 2 able to notify server]
Notify disconnection
[Peer 2 suddenly disconnects without server notification]
Notice stream disconnection
Request new peer
Send Peer 1 as new source
WebRTC Connection Handshake
WebRTC Media Stream
Peer 3
Peer 1
Server

Handle Peer Disconnect: How peer disconnections are handled, whether the peer being disconnected is able to notify the p2pcast application server, or if they disconnect suddenly.



Handle Broadcaster Disconnect: p2pcast handling the scenario when the most important peer, the broadcaster, disconnects.

4.4 Other Optional Features

There are several optional features that NP Compete has in mind in addition to the standard expected functionality of p2pcast. These of course are only suggested features given the chance of additional time. They are list in no particular priority or importance order.

4.4.1 Mobile Support

Currently Google Chrome has the widest support for the most recent draft specifications of WebRTC. Although only the desktop version of Chrome has the p2pcast required WebRTC features implemented, as time progresses iOS and Android flavors of Chrome will likely gain equivalent functionality.

At this time Google Chrome for some Android versions support having a peer be designated as a broadcaster only. The option also exists for this mobile broadcaster to select the prefered front or back facing camera source for broadcasting.

4.4.2 Screen Sharing

Screen sharing is currently supported by Google's Chrome Canary, which allows you to broadcast your screen to others. Allowing screen sharing as a video source for broadcasters will be a simple task, but the eventual goal would to integrate this with simulcasting screen sharing and a webcam. GoToMeeting from Citrix Online supports screen sharing and serves as an example of this feature's importance.

4.4.3 Chat

Since the WebRTC implementation in Chrome does not yet fully support audio rebroadcasting, there is a plan for allowing for alternate communication between the broadcaster and all participants. The broadcaster would be able to configure whether such an option is allowed for their channel and other administrative functions related to live chat.

4.4.4 Audio Broadcasting

Audio rebroadcasting support is not yet implemented in Google Chrome due to architectural decisions within their audio handling framework. The specification, however, does require this feature. Once this feature is implemented, adding audio support to all streams will be as simple as configuration change.

4.4.5 Mozilla Firefox

Mozilla Firefox has, at the moment, the second most complete implementation for the WebRTC specification. Users should be able to, in the future when Firefox's WebRTC support catches up, be able to view broadcast streams and also be able to fully act as a peer by rebroadcasting.

4.4.6 User Registration

User registration would carry immediate benefits. Broadcasters could have better control over their channels in setting metadata, and also make it more realistic to allow broadcasters to permanently or semi-permanently claim channel names. In the case of chat, users would allow for better administrative control. It would also bring value to browsing available channels, as users could rate channels.

4.4.7 Multiple Camera

Broadcasters having the option to broadcast multiple cameras simultaneously would bring enormous use cases for p2pcast. Such functionality would allow for webinars. And, more excitingly, broadcasters could stream a concert from multiple angles all at once. Broadcasters should have the option of choosing which audio stream is the played one (dependent on audio support being implemented in Chrome), or leaving that option to the viewers. Broadcasters could also rearrange videos in a grid on the channel page.

If user support were added, this feature could be greatly expanded. Instead of a single broadcaster owning a channel, a single channel owner could delegate broadcast capabilities to one or more users. This would allow for separate video streams to be broadcasted from different broadcast sources.

5. UI Mockups

The user interface is designed with simplicity and functionality in mind. Figure 5.1 shows the broadcaster's view. On the top right hand corner, the broadcaster can choose to configure the channel options. The viewer's view is similar to the broadcaster's view but without channel controls. In both the broadcaster and the viewer's perspective, the video stream is in the middle of the screen. The name of the current channel will be on top of the video stream. There are also counters that display the current number of viewers that are watching the stream and the current number of peers that are directly connected to you. Later there will be implementations of features such as a channel list and descriptions for each channel.

5.1 Channel Page (Broadcaster)

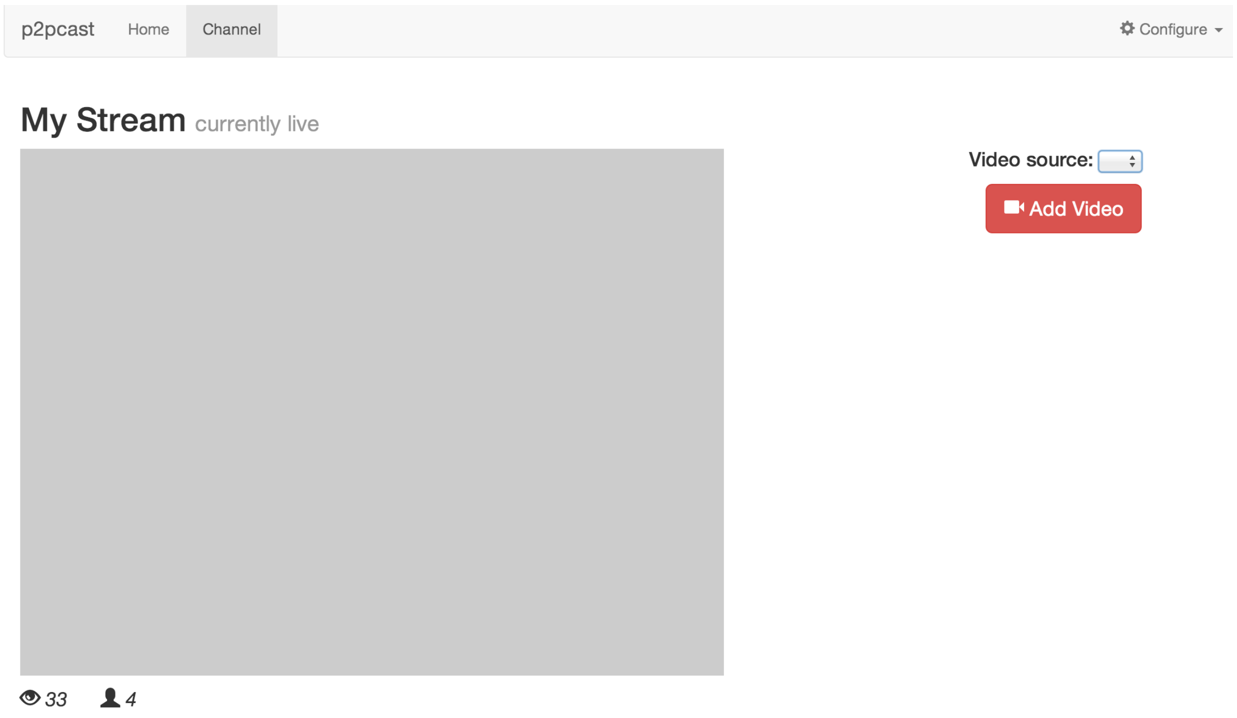


Figure 5.1: The view page from the broadcast's point of view.

5.2 Channel Page (Viewer)

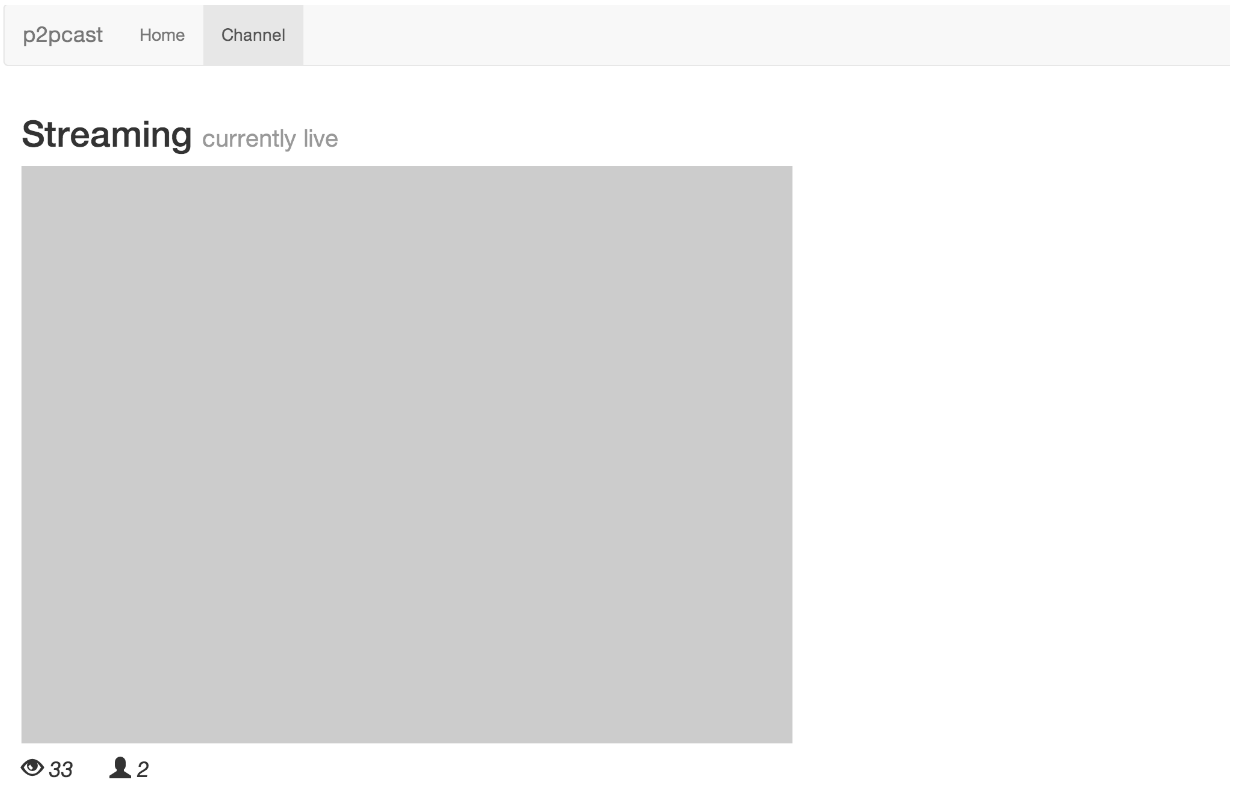
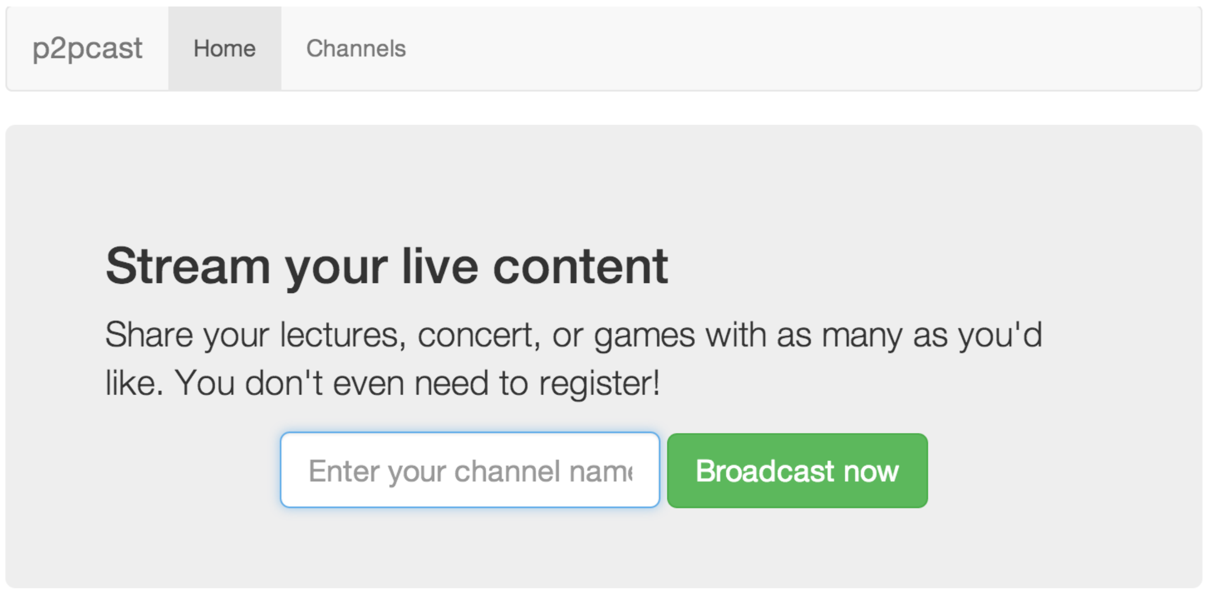


Figure 5.2: The view page on the viewer’s side with automatic connection to a broadcaster.

5.3 Homepage



Stream your live content

Share your lectures, concert, or games with as many as you'd like. You don't even need to register!

Figure 5.3: Homepage for p2pcast where user can select channel name and begin broadcasting.