Peer to peer networking and data transfer through the web browser.

Peer to peer consumer file/data transfer has traditionally been done through desktop based applications. More recently, the rise of the web has meant that this has moved to client-server based solutions situated in “the cloud” such as Dropbox or WeTransfer. Whilst cloud storage does have it's advantages, it has disadvantages such as server maintenance costs, server load/congestion and the privacy of data in the hands of third party services. The issue of data privacy has been made especially clear over the last few years due to activists such as Edward Snowden and as a result, I think this has made informed consumers more wary of third party cloud storage solutions. Peer to peer networking can be implemented in the browser as a way to solve this issue as the data being transferred does not pass through or remain on any third party servers.

The idea of my project is to utilise emerging web technologies, speficially WebRTC to produce a web application that can securely transfer or stream a file peer to peer (browser to browser) and to compare this approach as an alternative to current and previous solutions.

There have been similar implementations of WebRTC such as www.sharefest.me which allows for anonymous file transfer through browser to browser communication. This works by a peer selecting a file to “upload” on the web page, sharing the URL and then a second peer is able visit the same page where they can then download the file through a connection with the uploading peer.

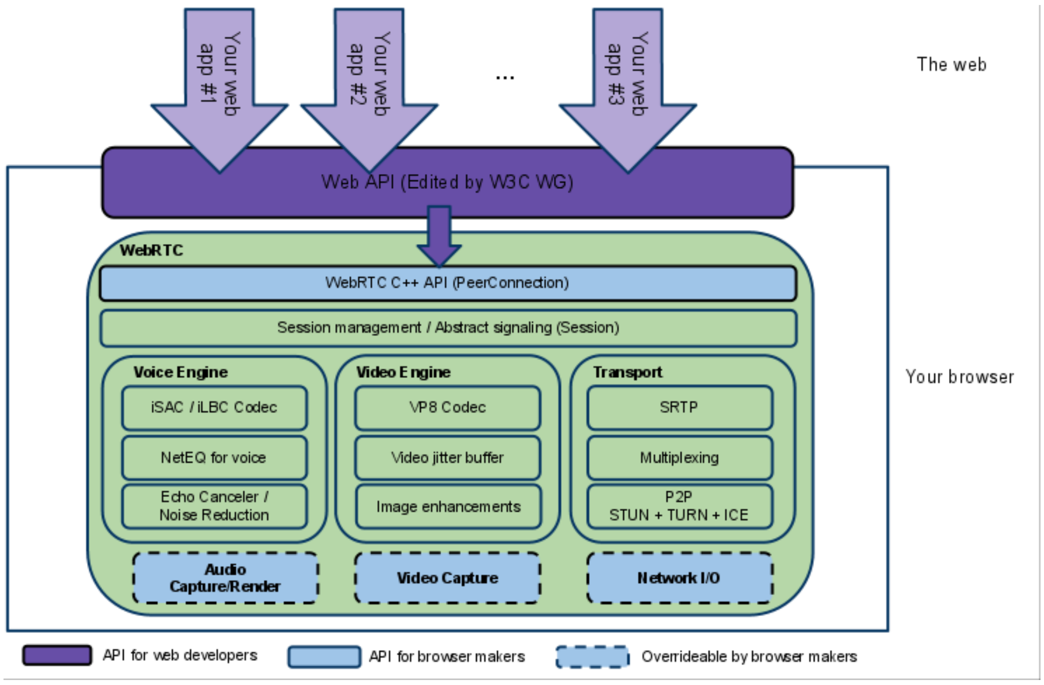
My plan is to expand on this idea by allowing the peer to stream the selected file as well as being able to download it. A potential use case for this is streaming a movie from your computer to a smartTV by having the web page open in a WebRTC supported browser such as Chrome or Firefox. This could also again be expanded to use the network of peers that have already buffered the video to allow the file to continue streaming to new visitors once the original peer has disconnected. This could also be based on the geographic location of the peers in the network to improve performance, similar to how a content delivery network works. On top of this, I plan to introduce metrics into the application in order to analyse how my project and WebRTC in general benchmarks in comparsion to present solutions

Throughout the project, I will be using git as my VCS with my repostories sitting on GitHub.To do the initial step of establishing the connection between peers, a signalling server is required. I plan to develop this server using Java, in particular the Spring framework and associated technologies such as Maven for dependency management and build automation, JUnit for unit testing and Apache Tomcat as the application server. This server will use WebSockets as opposed to using the standard HTTP protocol in order to achieve the bidirectional communication between client and server which is necessary for signalling.

The client-side web application will be developed using a JavaScript framework such as BackboneJS sitting on top of the WebRTC API , HTML5 and CSS3. WebRTC (Real Time Communication) is a recent API standard proposed by W3C and implemented by Google in 2013. It currently has 3 APIs for communication over browsers:

- getUserMedia for camera/microphone/media capture.
- RTCPeerConnection for audio/video calls.
- RTCDataChannel for peer-to-peer data sharing.

The main APIs that will be utilised is RTCDataChannel and potentially getUserMedia.



As peer to peer networking and the protocols behind WebRTC and video streaming are relatively new to me and as you can see from the architecture diagram above, there will be a lot of areas I will need to research in order to make my web application the best it can be.

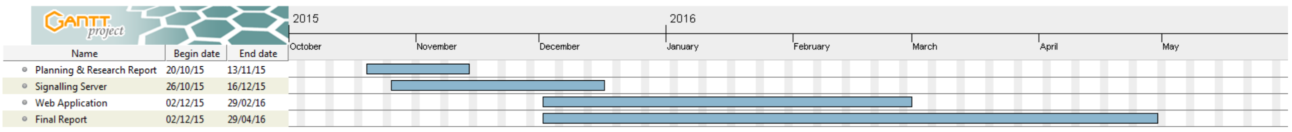
Potentially, the most difficult part of my project will be organising peers to distribute the stream of the video once it has been fully buffered on their clients in order to distribute the load away from the original streaming peer, therefore making it work like an actual peer to peer network rather than multiple peer to peer connectons from the this streaming peer to each client.

The first deliverable will be the signalling server written in Java. This will handle the exchange of client metadata in order to set up the connection between two peers.

The second deliverable will be the client side web application the user interacts with in order to select a file as well as this, it will handle sending the metadata to the signalling server and managing the peer to peer data transfer/streaming.

The third deliverable will be the report containing documentation and analysis using metrics from my application to benchmark my application, comparing how it and the architecture and technologies behind it perform in comparison to others, in particular how peer to peer over the browser (webRTC) compares to other methods of data transfer.

Below you can see a breakdown of the estimated timeline of my deliverables in the form of a gantt chart.



Name	Begin date	End date
Planning & Research Report	20/10/15	13/11/15
Signalling Server	26/10/15	16/12/15
Web Application	02/12/15	29/02/16
Final Report	02/12/15	29/04/16

In conclusion, I think the application I plan to produce and the metrics from is a worthwhile endeavour into examining how well peer to peer networking through browser-to-browser communication performs for transfering and streaming data.