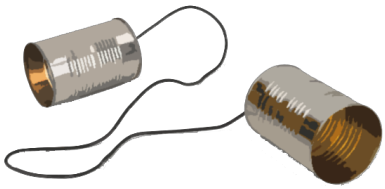

Tin Can

Participants

Henry Davidge (hd96) and Zack Reneau-Wedeen (jzr4)

Proposal Summary

We intend to use WebRTC technology to implement an in-browser decentralized video and chat client. We will call it “Tin Can.” As a foundation, we will study Chris Ball’s serverless-webrtc demo. We will then write our own web application using Node.JS, which, like the demo, brokers a serverless connection between two clients. We will make two substantial additions. First, we will introduce a client discovery engine that allows nodes to act as brokers of offer and accept messages between their friends. For example, if Alice and Bob are friends, and Bob and Eric are also friends, then we would like Alice to be able to find and chat with Eric through our application. Second, we will add support for video chat.

Motivations

In the current consumer technology climate, we believe that a practical decentralized communication system must operate in browser, and we see an opportunity to use cutting edge standards to effect such a protocol. Chatting through our service will be serverless, which we think is especially compelling today. There is a growing population of users who distrust big technology companies with their data, given recent leaks about NSA cooperation. Finally, the prolific javascript community presents a rich environment of potential features that could be incorporated into our system as time allows.

Core Features

These will be a part of our project. Responsibilities are designated in parentheses.

- Reimplement Chat with Node.js (Henry + Zack, with the help of the demo)
- Client Discovery and Introduction Brokering (Henry)
- Video Chatting (Zack)

The Future

These are probably beyond the scope of our project, but could be implemented in the future. If we finish the core features in time, we will incorporate some of these as well.

- Support in Google Chrome (might only be possible in a future version).
- Centralized storage of user information to ease introductions. (Note: the entire chat would remain decentralized, and no application server would ever see the chat data or

even know who is talking with whom.)
- Encrypted chatting.
- NAT hole-punching.
- A well-developed user interface.