Leveraging WebRTC for P2P Content Distribution in Web Browsers

Christian Vogt, Max Jonas Werner, Thomas C. Schmidt
Department of Computer Science
Hamburg University of Applied Sciences, Hamburg, Germany
christian.vogt@haw-hamburg.de, maxjonas.werner@haw-hamburg.de, t.schmidt@ieee.org

I. Introduction

WebRTC enables web applications to establish a direct communication channel between two browsers without relaying the data through a web server. It consists of an API [1] defined by the W3C and a set of underlying protocols defined by the IETF Rtcweb Working Group [2]. The possibility of establishing peer-to-peer channels between two browsers and the expected broad deployment opens the opportunity for new use cases that were not possible until now.

In this demo, we present a distributed content sharing facility using WebRTC Data Channels as well as an emulation component for test and measurement purposes. This library provides an API for applications to store content in and retrieve content from the underlying DHT; it can be used as a drop-in for existing web applications. The integration of WebRTC into a majority of browsers on the market can immediately deliver the benefits of such an approach to a huge user base, without the need of installing any additional software.

II. Architecture

The architecture proposed in this paper consists of two main components: *a bootstrap server* that serves as a central instance for joining the P2P network and the *peers* (made up of the users's browsers) forming the actual network.

The implementation is based on the built-in capabilities of web browsers, leveraging the WebRTC API. So called Data Channels [3] allow for the transfer of generic data (text, binary data) in a P2P manner. Point-to-Point Data Channels comprise the main transfer mechanism used to build the P2P system described in this paper. On top of these Data Channels we implemented a protocol used for joining the WebRTC network and for maintaining connections between clients.

A. Functionality

1) Join: Clients join the P2P network by establishing a WebSocket connection to a bootstrap server that holds a connection to a certain number of peers that have recently joined the network. Afterwards this WebSocket connection is used to signal the WebRTC handshake, resulting in a direct WebRTC connection between the newly joined client and one of the other clients, forming a mesh of peers. Once a client has joined the network it may disconnect from the server without losing the capability of transferring data to/from other peers.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

2) Message exchange: Figure 1 demonstrates the exchange of a message between client 1 and client 3. Here, client 1 sends a message via the Data Channel to client 2 indicating the sender (client 1) and receiver (client 3). Our approach is designed so that client 2 forwards all messages dedicated to other peers via a routing layer to the designated receiver. Messages that are meant for client 2 itself are passed to the local application layer on client 2.

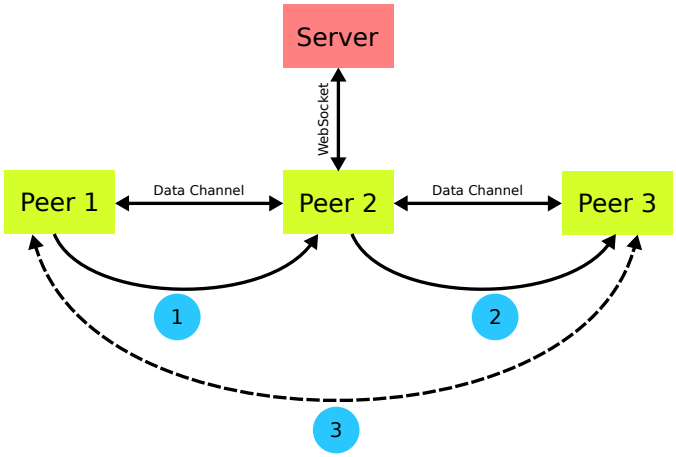

Fig. 1. Client 1 wants to communicate with client 3. Therefore it sends the data via the open Data Channel to client 2 which in turn routes it to client 3. The server is not involved in this process at all.

3) Layered approach: To enable the functionality explained above we used a layered approach similar to that proposed by Dabek et al. in [4]. Figure 2 shows the different layers and APIs. The central component – called PeerManager – provides an API to applications that would like to make use of the P2P network. Underneath the PeerManager lies a routing layer that can also be implemented in different ways, later allowing us to form a structured Peer-to-Peer overlay like Chord [5].

Our next step is to concretize the component's APIs to be able to assemble the three layers from Figure 2 into a working solution. Using a structured P2P system like Chord we are planning to achieve our goal of implementing a pure browser-based P2P network that allows us to investigate possibilities of running information-centric networking components on top.

B. P2P mechanisms

P2P systems can be classified by two main categories: *Unstructured* and *Structured* systems. Unstructured P2P systems

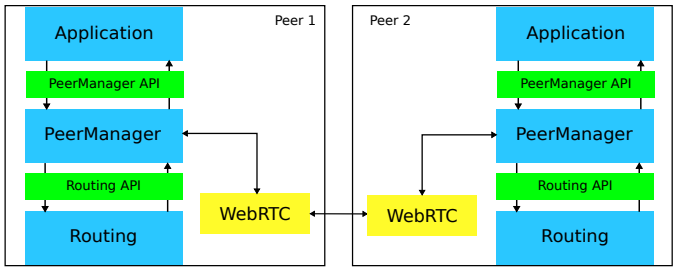

Fig. 2. Our layered approach mimics the proposal towards a common API for P2P networks by Dabek et al. [5].

use a flooding-based approach or a centralized component for lookup and management purposes. In contrast to unstructured P2P systems, structured P2P systems can achieve logarithmic scalability in search and lookup complexity without requiring any central component. Therefore, we concentrate our work on implementing a structured P2P system based on a distributed hash table (DHT). Such a DHT (like Chord or CAN) will be the foundation of the routing layer mentioned above.

C. Demonstration

In our demonstration we will show the following components: A working server component which acts as bootstrap for newly joined nodes and takes care of assigning node IDs. Moreover we've developed a rough client-side application that implements the PeerManager component from Figure 2. We will show how different browsers join the P2P network and may establish connections to each other even after shutting down the signaling server.

For test and measurement purposes, we also developed an emulation component that uses Node.js and the WebRTC implementation contained in the Libjngle library. This emulation is able to run all of the client code written in JavaScript so that we can conduct functionality and performance tests as well as measurements. The status of this emulation layer will also be demonstrated.

III. Related work

Research on leveraging native browser technologies – each achieving a different set of goals – is already being conducted: [6] examines a way to distribute the load and stream video content between browsers using WebRTC, thus reducing the bandwidth cost of content providers. The author uses a BitTorrent-like architecture involving a tracking server for discovering content.

Zhang et al. describe the implementation of a browser-based CDN in [7]. The authors have researched on the possibilities of building such a CDN service using the Flash plugin by Adobe. Their implementation is centered around a coordinator that holds mappings between peers and the data stored on these peers. Web site owners eager to use this solution have to modify the site's HTML code so that every asset that shall be distributed in the P2P network is fetched via JavaScript from one of the participating peers using the proposed mechanism. Similar approaches are pursued by PeerCDN[1] and SwarmCDN[2].

IV. Conclusion/Future Work

Our current experience researching on WebRTC and content distribution in combination with Peer-to-Peer networking leaves us confident that a deeper exploration of the possibilities provided is well worth it. We have already planned next steps with regards to the implementation of the proposed component layers – especially building a working DHT-based CDN prototype on top of WebRTC – as well as investigating possible overlapping issues of browser-based P2P networks and information-centric networking [8].

Topics that are to be investigated more thoroughly with regards to WebRTC are those of security and privacy of users. We will further cover these as part of our ongoing research and implementation efforts. The current specification drafts already cover a wide range of these aspects, especially focussing on end-to-end encryption and peer authentication.

Since both W3C/IETF as well as browser vendors have not agreed on a final specification the APIs and protocols used are very much in flow. This could lead to problems especially when trying to lead our implementation to a consistent state that can be used productively. We are intensely watching the ongoing research activities by W3C and IETF as well as the browser vendor's implementations.

References

[1] A. Bergkvist, D. C. Burnett, C. Jennings, and A. Narayanan, "WebRTC 1.0: Real-time Communication Between Browsers," W3C, W3C Editor's Draft, Jun. 2013.

[2] H. Alvestrand, "Overview: Real Time Protocols for Brower-based Applications," IETF, Internet-Draft – work in progress 01, June 2011.

[3] R. Jesup, S. Loreto, and M. Tuexen, "RTCWeb Data Channels," IETF, Internet-Draft – work in progress 04, February 2013.

[4] F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica, "Towards a Common API for Structured Peer-to-Peer Overlays," in *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22, 2003, Revised Papers*, ser. LNCS, M. F. Kaashoek and I. Stoica, Eds., vol. 2735. Berlin Heidelberg: Springer–Verlag, 2003, pp. 33–44.

[5] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 2001, pp. 149–160.

[6] A. J. Meyn, "Browser to Browser Media Streaming with HTML5," Master's thesis, Technical University of Denmark, DTU Informatics, E-mail: reception@imm.dtu.dk, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012.

[7] L. Zhang, F. Zhou, A. Mislove, and R. Sundaram, "Maygh: building a cdn from client web browsers," in *Proceedings of the 8th ACM European Conference on Computer Systems*, ser. EuroSys '13. New York, NY, USA: ACM, 2013, pp. 281–294.

[8] C. Vogt, M. J. Werner, and T. C. Schmidt, "Content-centric User Networks: WebRTC as a Path to Name-based Publishing," in *21st IEEE Intern. Conf. on Network Protocols (ICNP 2013), PhD Forum*. Piscataway, NJ, USA: IEEEPress, Oct. 2013, accepted for publication.

[1] http://www.peercdn.com
[2] http://www.swarmcdn.com