INTERNET-DRAFT
Intended Status: Informational
Expires: August 18, 2014

R. Huang
Huawei
Y. Zhang
CoolPad

February 14, 2014

Survey of WebRTC based P2P Streaming
draft-huang-ppsp-p2p-webrtc-survey-00

Abstract

This document presents a survey of current emerging WebRTC based P2P streaming applications available on the nowadays Internet.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1 Introduction

P2P streaming applications usually use standard or private P2P protocol to deliver content between their end users. Some of them uses specific desktop software as the client which end users have to download and install in their OS. The others uses browsers as clients, but users still need to download corresponding plug-ins to enable the P2P transmission.

WebRTC is a new software architecture which enable web browsers with Real Time Communication (RTC) capabilities via simple JavaScript APIs. It provides direct interactive communications using audio, video, etc. between two peers' web-browsers without any additional plug-ins. Currently, WebRTC only focuses on conversational applications such as VoIP and video conferencing, multimedia streaming systems are not considered. However, the interests towards WebRTC is strongly increasing. An ever-increasing number of P2P applications have adopted WebRTC to distribute multimedia content or file sharing from a source to a large number of endpoints only via browsers without installing any plug-ins. This document presents a survey of current emerging WebRTC based P2P streaming applications available on the nowadays Internet.

the remainder of this document is organized as follows: Section 2 introduces terminology used throughout the survey; Section 3 provide the analysis of how traditional P2P streaming applications on browsers work; Section 4 presents current emerging WebRTC based P2P streaming applications.

2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Plug-in: a piece of software which enhances another software application and usually cannot be run independently.

API: Application Programming Interface - a specification of a set of calls and events, usually tied to a programming language or an abstract formal specification such as WebIDL, with its defined semantics.

Peer: A peer refers to a participant in a P2P streaming system that not only receives streaming content, but also caches and forwards streaming content to other participants.

Tracker: A tracker refers to a directory service that maintains a

list of peers participating in a specific audio/video source or in the distribution of a streaming file. A tracker answers queries received from peers for the information of other peers sharing the same streaming content. A tracker is a logical component which could be centralized or distributed.

Tracker protocol: A signaling protocol between trackers and peers.

Peer protocol: A signaling and data protocol among the peers.

3 Traditional P2P Streaming Applications on Browsers

Some of traditional P2P streaming applications also allow their users to obtain their content via browsers. The browsers are required to install some plug-ins provided by these content providers firstly. After that, the browsers of end users download and run specific JavaScript application from content server. The JavaScript application will act as a peer of P2P network to connect to a tracker and get a list of available peers via tracker protocol. With the help of plug-ins, the JavaScript application will then be able to communicate with other peers exchanging content via standard or private peer protocol.

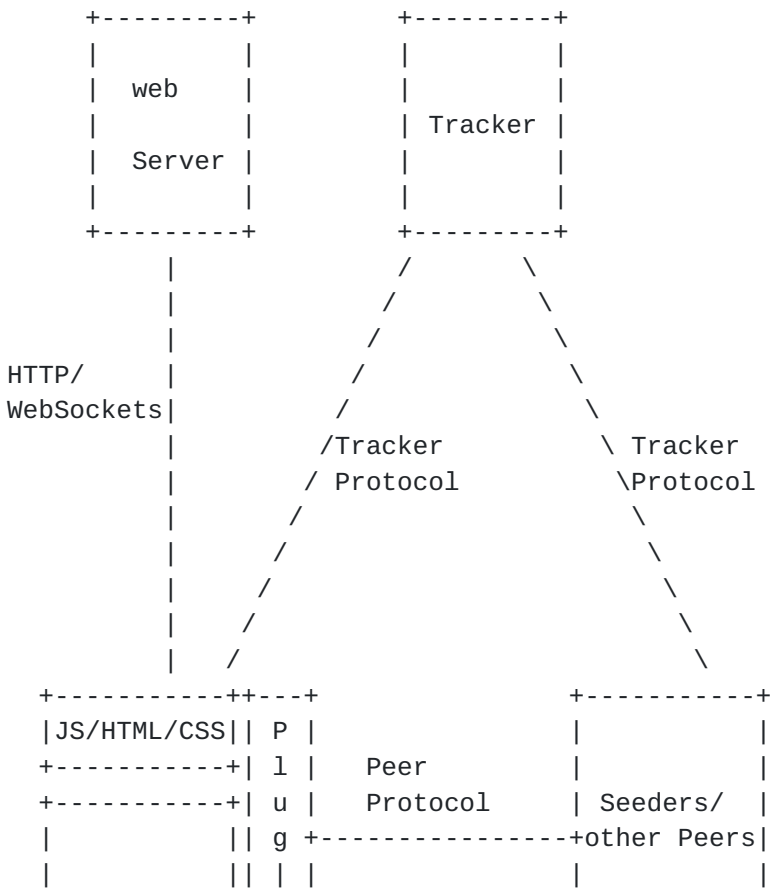

Figure 1 Traditional P2P Streaming Applications on Browsers

Adobe's Flash Player is a typical example of such usage. It has cooperated with many content providers, e.g. Sohu video, to support P2P video streaming. It also provides APIs to help developers to be able to use flash player to build various other P2P applications right within the browsers. But the browsers of these users will be required to install or update the plug-ins of Flash Player. Flash Player has provided the ability of peer-to-peer communication since version 10.1.

4 WebRTC based P2P Applications

WebRTC is a collection of technologies, including a media connection protocol (which is RTP) that can be used to initiate sessions like video conversation, a data exchange protocol that will enable users to transfer data, e.g. chat or files, without the need for a third-party server, and a way to access local resources, e.g. user's microphone and video camera. WebRTC is created to facilitate conversational communications from web browsers. However, it is not the only usage WebRTC turns out to be. Recently, some innovative applications and projects are using WebRTC technology on peer-to-peer connections to accomplish tasks like streaming and sharing, instead of relying on third-party plug-ins or proprietary software. They all rely on WebRTC's data channel protocol, which is designed to allow browsers to exchange data other than audio or video. Some of the typical WebRTC based P2P applications are presented as follow, but not exhaustive. Currently, the data channel protocol has been implemented in Chromes as well as Firefox, which means that current WebRTC based P2P applications are only supported in these two types of browsers.

4.1 Swarm CDN

Swarm CDN is a new peer-to-peer browser based content delivery network using WebRTC technology to save web site owners' money, reduce their bandwidth and reduce server costs. Instead of hosting the content on servers in data centers, Swarm CDN provides a way for users to deliver content to other users on the same site without installing any plug-ins.

Figure 2 demonstrates the rough architecture of this system.

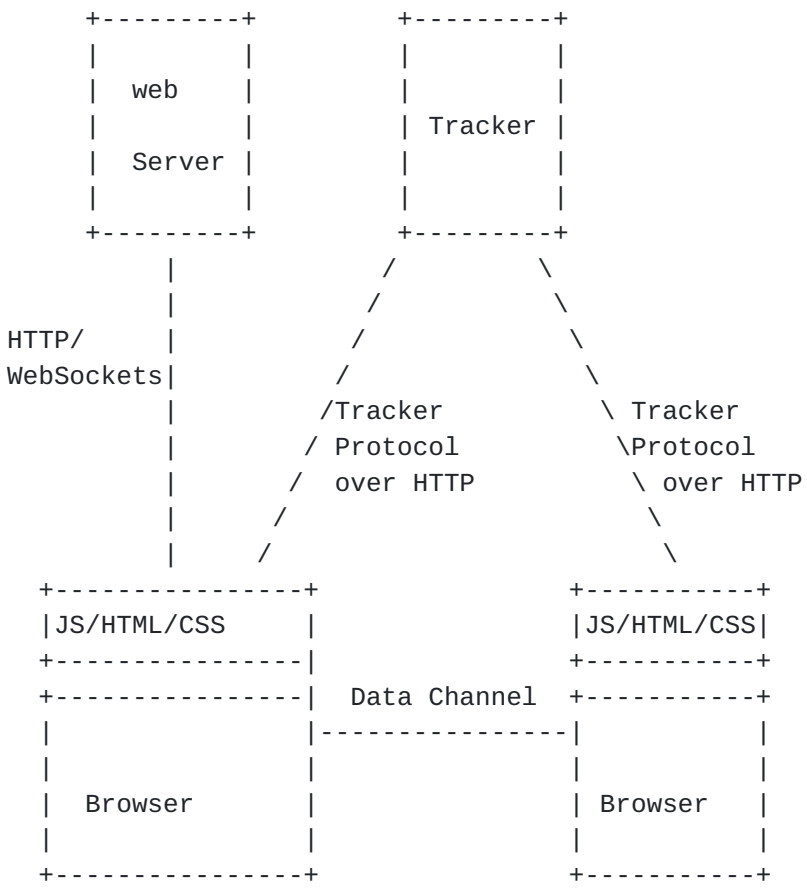

Figure 2 WebRTC based P2P Applications

Swarm CDN provides plug-ins to content providers or modifies their sites' HTML code so that peers made up of the users's browsers will be redirected to a centralized server which is called tracker in this memo. As depicted from the above figure, after the web page is loaded from the original web server, the peer will join the P2P network by establishing a HTTP connection to the centralized server which holds connections to a certain number of peers that have recently joined the network. Afterwards, this HTTP connection is used to signal the WebRTC handshake, resulting in a direct WebRTC data channel between the newly joined client and one of the other clients that has the content. For those users on a non compatible WebRTC browsers, the tracker will detect this and default them to retrieve the requested content from the original web servers.

Swarm CDN could provide service for video and images. Currently, it's free and able to swarm video and images for 20 simultaneous users.

4.2 PeerCDN

PeerCDN helps web applications build a peer-to-peer content delivery

network (CDN) that will make the web faster, more reliable, and help sites to reduce bandwidth costs. The technology of PeerCDN is similar to Swarm CDN, which uses WebRTC encrypted data channel for data transmission. It also requires users to insert a single line of HTML code to enable the usage without any browser plug-ins. Afterwards, a web browser triggers a notification to the peerCDN's tracking server that they are visiting the page. If someone else visits this page at the same time, peerCDN automatically sets up a connection and quietly streams the data between the peers. PeerCDN serves a site's static assets, such as images, streaming videos and file downloads. It only works on all WebRTC browsers.

PeerCDN was founded by Stanford engineering graduate in early 2013. And Yahoo has bought this startup company in December of 2013.

4.3 Sharefest

Sharefest is a file sharing application developed by a video streaming company called Peer5. Different from tools like Dropbox transferring files to the remote server then back again, Sharefest creates peer-to-peer communication between browsers directly using WebRTC.

Figure 3 depicts the architecture of Sharefest.

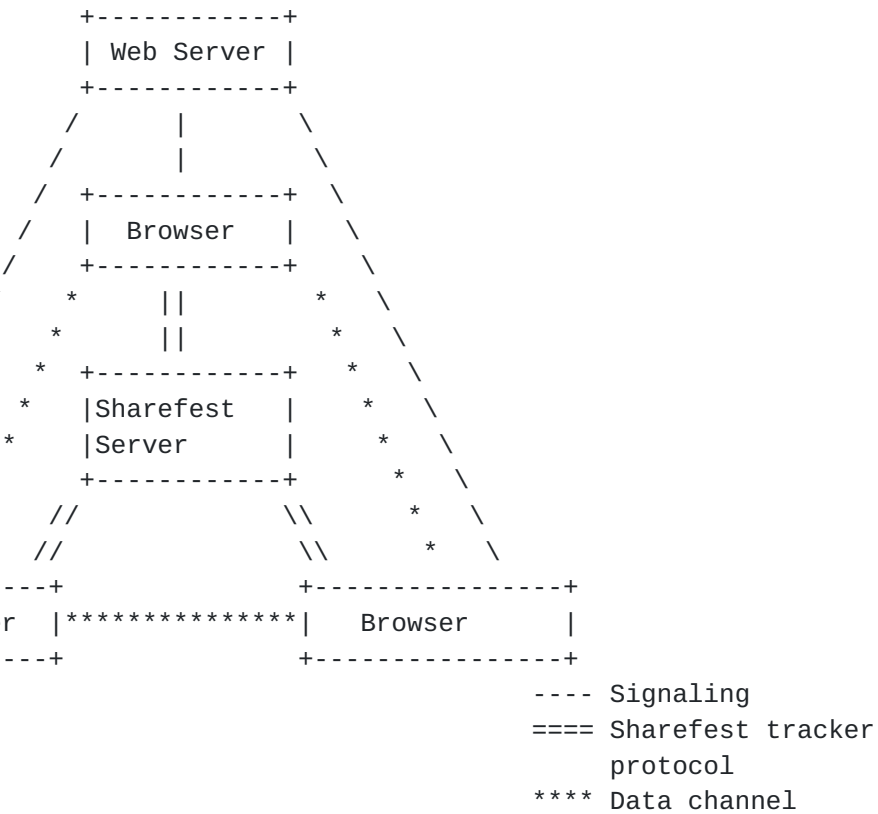

Figure 3 Architecture of Sharefest

The Sharefest server presents an auto-generated short-URL for each file shared by the client wishing to share. When a client browser visits the URL, the server connects it to the sharing client's browser and the sharing client's browser transfers the file over an WebRTC data channel. If multiple clients connect from different locations, the server attempts to pair nearby clients to each other to maximize the overall throughput. Sharefest utilizes some simple p2p technologies like slicing. It does its best to maximize efficiency by dividing each file into slices that the downloading peers exchange with each other.

Currently the size limit for shared files is around 1 GB and only the recent versions of Chrome and Firefox are supported.

4.4 P2P Media Streaming with HTML5 and WebRTC

Some students from Aalto university have proposed an experimental system for P2P VoD service applying WebRTC standards. The experiment mirrors the BitTorrent architecture and the same entities - tracker, seeder and peers - are used. Peers and seeders could be WebRTC browsers. It can be seen in Figure 4.

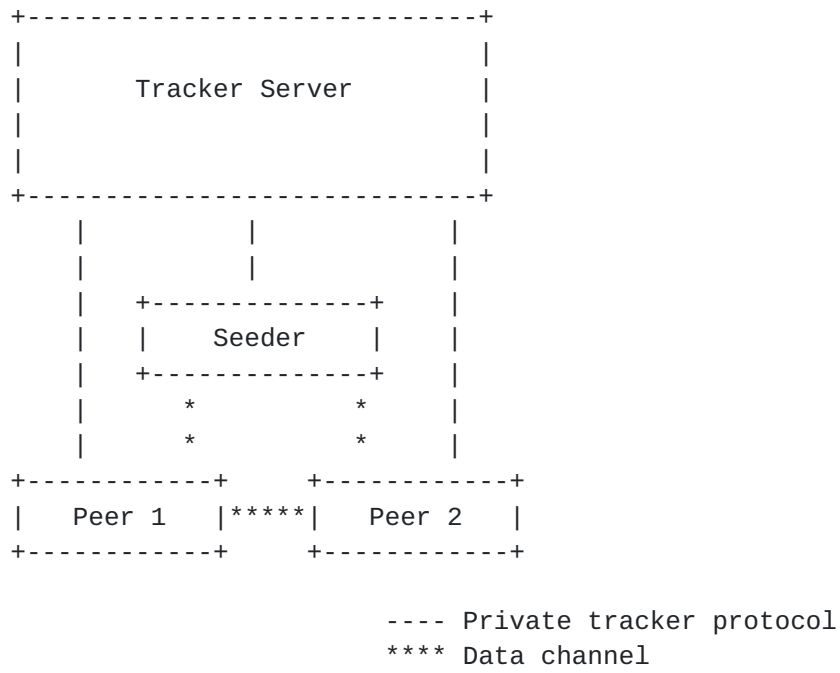

Figure 4 Network Architecture for the P2P VoD service with HTML5 and WebRTC

Initially users or content provides create a torrent file containing the metadata of the video, hashes of each piece of the video and uploads the torrent file to the tracker. Media content is uploaded to

the seeder simultaneously. When a peer requests this content, it will obtain the metadata from the tracker and start to download the content slices from the seeder or original peer uploading the content or both. As the content becomes popular, subsequent peers could get the content from the nearby peers, which decrease the load on the seeder and original peer uploading the content.

The experiment raises some considerations on this kind of implementations. The main concern is that the performance of MD5 on JavaScript and how much load could be handle in current solution generated. It points out that the resources available through browsers are much more limited than for native applications. The calculation of MD5 hashes use too much CPU consumption which will adversely affect the speed of getting content to play, especially for mobile devices. This experiment also indicates that handling two streams,incoming and outgoing, is manageable with current desktop and mobile devices. Increasing peers number will severely degrade performance and result in a bad user experience.

As the capability of computing for both desktop and mobile devices is rapidly increasing, it can be anticipated that the above problems will be eliminated soon.

5 Discussion

5.1 Incorporating P2P Streaming in RTCWEB Scenarios

As we have discussed in the above sections, there are more and more efforts on enabling p2p streaming using WebRTC technologies. According to the statistics of LTE usage in Verizon [Verizon], video streaming traffic is accounting the largest part of the LTE traffic and it slows down the wireless download speed by 20 percentages. Using p2p streaming is an efficient means to solve the traffic pressure not only in the core network, backbone but also in the air interface especially when Device to Device mode (D2D) is deployed between the browsers of end nodes, which is in the standardization process in 3GPP. Although RTCWEB is created to facilitate JavaScript APIs for real-time communication in the browser without implementing any plug-ins, the current scope of RTCWEB only includes conversational applications not streaming systems. We can see that web applications supporting p2p streaming without plug-ins are more and more appealing. So is it possible to incorporate P2P streaming in RTCWEB scenarios so as to implement JavaScript APIs for P2P algorithms as part of the WebRTC standardization process?

5.2 Open Issues

Several open issues need to be addressed for the purpose described in Section 5.1:

1. Do we need WebRTC to support P2P algorithms? If yes, how to support?

2. Do we need WebRTC to support the tracker protocol? If yes, how to support?

3. Do we need WebRTC to support the peer protocol? If yes, how to support?

4. Is there any security problems?

6 Security Considerations

This document has no security concern.

7 IANA Considerations

This document has no actions for IANA.

8 Acknowledgments

TBD.

9 References

9.1 Informative References

[RFC6972] RFC 6972, "Problem Statement and Requirements of the Peer-to-Peer Streaming Protocol (PPSP)"

[SwarmCDN] Swarm CDN web site, http://swarmcdn.com/

[PeerCDN] Peer CDN web site, https://peercdn.com/?

[Sharefest] Sharefest web site, http://www.sharefest.me/?

[PPMSHW] J. Nurminen, A. Meyn, et al., "P2P Media Streaming with HTML5 and WebRTC", February 2013

[Verizon] http://www.fiercewireless.com/tech/story/verizon-spending-millions-network-upgrades-counter-growth-video-traffic/2013-12-16

Authors' Addresses

Rachel Huang
Huawei
Phone: +86-25-56623633

EMail: rachel.huang@huawei.com

Yunfei Zhang
Coolpad

EMail: hishigh@gmail.com