OPPORTUNITIES AND CHALLENGES OF IMPLEMENTING P2P STREAMING APPLICATIONS IN THE WEB

Irena Trajkovska, Pedro Rodríguez, Javier Cerviño and Joaquín Salvachúa
Universidad Politécnica de Madrid
Ciudad Universitaria, Avda. Complutense s/n
28040, Madrid, Spain

ABSTRACT

P2P applications are increasingly present on the web. We have identified a gap in current proposals when it comes to the use of traditional P2P overlays for real-time multimedia streaming. We analyze the possibilities and challenges to extend WebRTC in order to implement JavaScript APIs for P2P streaming algorithms.

KEYWORDS

P2P, multimedia, streaming, WebRTC

1. INTRODUCTION

In the early days of Internet, the complexity of IP level multicast to stream media to users has triggered the movement of the multicast functionality on the application level. Since then, users in the role of clients and servers called peers, make the core of what is called a P2P technology. Implemented as an organized overlay, it quickly became attractive for deployment of P2P algorithms for applications such as streaming, audio/video conferencing and file sharing. **Real-time P2P streaming** systems as a counterpart of **Video on Demand** have replaced traditional way of "download then watch" content media by "watch on-line". WebRTC on the other hand, is an ongoing web project guided by Google, Opera and Mozilla in order to facilitate JavaScript APIs for real-time communication in the browser. Today, in order to watch a P2P streaming content, a user needs to download an application that bases on some P2P algorithm.

In this paper we analyze the possibilities and challenges to extend WebRTC in order to implement JavaScript APIs for P2P algorithms as part of the WebRTC standardization process. This will leverage the browser with ability to host P2P multimedia streaming applications in a straightforward manner without having to install plugins or download the application. The developer would only implement the multimedia application using APIs of algorithms such as PRIME [2] or LayeredCast [3] as part of the JavaScript library, rather than deploying a complex application that includes these P2P algorithms. Since the number of P2P streaming applications on the Internet takes a significant part among the RTC, we believe this approach deserves to be considered in order to satisfy entirely the necessity of the multimedia streaming community.

2. STATE OF THE ART

To establish the context of our discussion, brief overview of P2P streaming overlays, and current WebRTC trends of real time multimedia on the Web is presented.

P2P streaming systems' classification based on the overlay network topology consists of **Tree-based** and **Mesh-based**. First one is an overlay topology made of one or multiple connected trees. The root represents the streaming server that starts the video dissemination, while the nodes take part of various sub trees. The streaming content is divided into sub-streams, each routed in separate sub-tree. Scribe [1] and Pastry [5] are algorithms that use tree-based overlays for P2P multimedia streaming. In mesh-based approach, the participating peers form a randomly connected dynamic overlay called mesh. Every peer maintains incoming and outgoing connections with several neighbors depending on their bandwidth and the content availability. Then it delivers the media content by push/pull mechanism in a dynamic way. Examples among many applications that use mesh algorithms include CoolStreaming[8], PPLive[1], SopCast[2], etc.

WebRTC is a free, open project, with an idea to enable web browsers with Real-Time Communications (RTC) capabilities and permit development of rich, high quality, RTC applications in the browser via simple Javascript APIs and HTML5[3]. Main goal of this specification is to achieve an interoperability of real-time multimedia applications across multiple web browsers and liberate the application users of the plugin installation requirement. The key driver of these two technologies would be the possibility to implement P2P routing algorithms into JavaScript API through WebRTC and this way facilitate the creation of P2P web oriented real-time multimedia streaming applications.

3. OPPORTUNITIES IN P2P STREAMING OVER WEBRTC

We are going to take a look at the different aspects of WebRTC that can bring new opportunities and features, in comparison to the existent peer-to-peer streaming solutions. Obviously, the most important aspect of WebRTC is the fact that is a novel technology designed for implementations in web browsers accessed via a JavaScript API. However, this means that the protocols used to establish connections and transmit data have to be standardized so that different browsers can communicate.

We will divide the potential pros of using WebRTC for peer-to-peer real-time streaming in two big groups: advantages derived from it being a web technology and pros of the standardized solutions defined for the transport and connection establishment protocols.

3.1 Web Applications

Being WebRTC a web technology destined to work within web browsers, it inherits most of the known benefits of web applications. Some of them are:

Manageability: In terms of actualization and patching, managing web applications is very easy as users always obtain the last version of the client application when they browse the web page. As a side effect, there is no possibility a peer can connect to our new P2P streaming application with an old, incompatible version.

Multiple platform development: Clients of WebRTC applications are coded in JavaScript and are run by a web browser. If browser developers follow the standard, the application should run without modifications, regardless of the specific browser that the user has chosen. Furthermore, the operative system or even the hardware platform is also irrelevant as long as there is a compatible browser developed.

No installation: Today's peer-to-peer based streaming services require the installation of dedicated applications or, at least, browser plugins. By using WebRTC we avoid this step, reaching a larger user base.

3.2 Standardized Protocols

The protocols used in WebRTC are defined by the IETF working group (rtcweb). Having a standard definition of the technologies and protocols being used provides opportunities in terms of interoperability with new and existing applications. Furthermore, the current state of the drafts points towards already established and proven protocols and mechanisms. While following this standards can seem like a loss of flexibility as some of the decision have already been taken, it can also be seen as an advantage as developers can focus on the overlay network and the way the media is distributed and take the lower protocol levels for granted. Here we will discuss some of these aspects:

RTP/SRTP: Real-Time Transport Protocol [6] and its secure counterpart. RTP is extensively used in real-time applications, especially in real-time audio and video communication tools as well as real-time streaming solutions.

[1] www.pplive.com
[2] www.sopcast.com
[3] www.webrtc.org

ICE: Interactive Connectivity Establishment [4] provides the mechanism needed to perform NAT (Network Address Translation) traversal using both STUN and TURN to achieve this purpose. NATs can make peer-to-peer communication difficult in some cases; the advantage of using ICE in P2P streaming is that peers can always connect to each other.

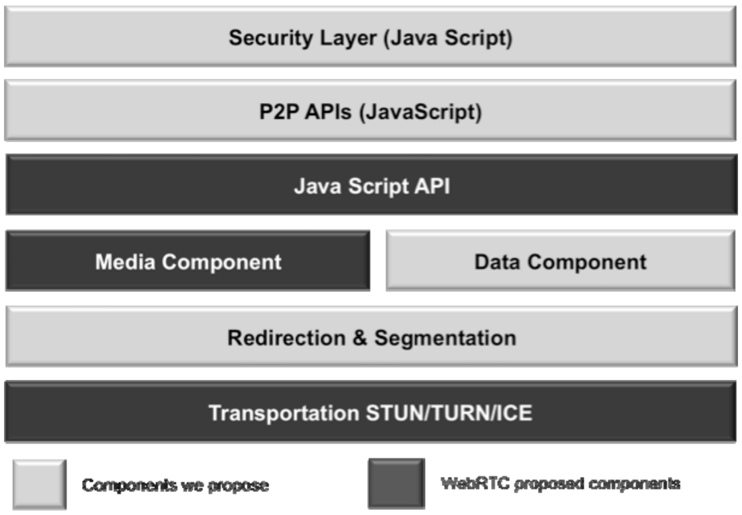
Security Layer (Java Script)
P2P APIs (JavaScript)
Java Script API
Media Component
Data Component
Redirection & Segmentation
Transportation STUN/TURN/ICE
Components we propose
WebRTC proposed components

Figure 1. Additional components proposed in WebRTC for P2P streaming support

4. CHALLENGES IN P2P STREAMING OVER WEBRTC

This section discusses the key open questions that WebRTC group must solve in order to support P2P streaming in the web browsers. For that purpose, as crucial components to be implemented as parts of the specification are: redirection and segmentation of media streams in clients, security aspects, data channel support and implementation of P2P algorithms in JavaScript. Figure 1 depicts a diagram of already scheduled for implementation components by the WebRTC project and the ones we propose based on the challenges we discuss. We think that these questions need to be addressed prior to the implementation of P2P streaming using standard mechanisms based on WebRTC.

Streams redirection and segmentation between multiple *PeerConnection* in the client: WebRTC specification does not clearly state that an incoming media stream in a *PeerConnection* can be redirected to another *PeerConnection*. This feature would enable the creation of P2P overlays using current WebRTC APIs for media streaming. If these overlays were tree-based, clients would send incoming data to peers, and these peers would redirect the same traffic to other peers in the overlay. In mesh-based overlays the solution would be more difficult for video and audio streaming because it would force WebRTC implementations to support splitting of the streaming content into chunks. In other P2P scenarios (file sharing, gaming, etc.), the implementation of the overlay would be simpler since the stream only consists of data (different from video and audio) that makes it easier to get processed by custom JavaScript libraries.

Data channel must be finally supported: At the time of writing this paper, WebRTC specification does not completely support the transmission of data. Its group is only making progress for providing audio and video transmission that is quite interesting for multimedia P2P scenarios, such as simple streaming or conferencing. But there are additional P2P cases in which the transmission of other type of data would be also mandatory: on-line gaming for distributing game events, file sharing, collaborative distributed systems, signaling for advanced multimedia streaming, etc.

P2P algorithms must be ported to JavaScript libraries: In section 2 we mentioned several P2P algorithms and architectures that are currently implemented in different systems. These algorithms support the creation of P2P overlays: mesh-based or tree-based. Assume that previous challenges have been successfully solved; P2P developers would need to port existing algorithms to JavaScript language in order to make them work for web applications. In other words, algorithms such as Scribe [1] or Pastry [5] will have to be implemented in JavaScript. With this, the web applications could provide complete P2P services through the web browsers.

Security problems related to JavaScript and P2P Systems: Apart from the functionality, if we manage to solve previous problems, P2P developers would be concerned about another important aspect: security. In this case there are several vulnerabilities, such as lack of both JavaScript code encryption and efficient trust management in new P2P scenarios. Nowadays JavaScript source code can be downloaded and the end users can read it. This can be considered as global flaw in JavaScript and can especially affect P2P scenarios because participating users could access foreign data of other users that is redirected through their clients. This could be solved if JavaScript code was ciphered and browsers prevent the user of accessing this data. Finally, P2P applications should implement some kind of mechanism for trust management in order to support faithful traffic redirection among clients.

5. CONCLUSION

We analyzed the opportunities and challenges of supporting P2P streaming applications using WebRTC, suggesting an implementation of current P2P algorithms in JavaScript. Possible use case of this outcome could be a web adaptation of various P2P architectures such as for example the one described in [7]. Further proposal could include JavaScript APIs for QoS management in P2P architectures. However additional changes would be required in the current WebRTC standard to overcome the challenges described throughout this paper and facilitate the creation of more developer- and user-friendly P2P applications on the web.

ACKNOWLEDGMENT

This research was conducted with the funding from the project SAAN: TIN2010-19138

REFERENCES

[1] M. Castro, et al, 2002. Scribe: a large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, volume 20, issue 8, pages 1489-1499.

[2] N. Magharei, et al, 2009. PRIME: peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM Trans.* Netw. 17, 4, pages 1052-1065.

[3] M. Motamedi, et al, 2010. LayeredCast-a hybrid peer-to-peer live layered video streaming protocol. *Telecommunications (IST)*, 2010 5th International Symposium on, pp. 663-668.

[4] J. Rosenberg, 2010. RFC 5245: Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. *IETF RFC*.

[5] A. Rowstron and P. Druschel, 2001. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. *IFIP/ACM Int. Conf. on Distributed Systems Platforms*. Heidelberg, Germany, pages 329-350.

[6] H. Schulzrinne, et al, 2003. RFC 3550: RTP: A Transport Protocol for Real-Time Applications. *IETF RFC*.

[7] I. Trajkovska, et al, 2010. A novel P2P and cloud computing hybrid architecture for multimedia streaming with QoS cost functions. *Proceedings of the international conference on Multimedia*. Firenze, Italy, pages 1227-1230.

[8] X. Zhang, et al, 2005. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. *INFOCOM. 24th Conf. IEEE Computer and Communications Societies.* Miami, USA, pages 2102-2111.