This is a sample chapter of "WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web" by Alan B. Johnston and Daniel C. Burnett, Second Edition.
For more information or to buy the paperback or eBook editions, visit

http://webrtcbook.com

1 INTRODUCTION TO WEB REAL-TIME COMMUNICATIONS

Web Real-Time Communications (RTC), or WebRTC, adds new functionality to the web browser. For the first time, browsers will interact directly with other browsers, resulting in a number of architectures including a triangle and trapezoid model. The media capabilities of WebRTC are state-of-the-art, with many new features. The underlying standards of WebRTC are being developed by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

1.1 WebRTC Introduction

WebRTC is an industry and standards effort to put real-time communications capabilities into all browsers and make these capabilities accessible to web developers via standard [HTML5] tags and JavaScript APIs (Application Programming Interfaces). For example, consider functionality similar to that offered by Skype™ [SKYPE] but without having to install any software or plug-ins. For a website or web application to work regardless of which browser is used, standards are required. Also, standards are required so that browsers can communicate with non-browsers, including enterprise and service provider telephony and communications equipment.

1.1.1 The Web Browsing Model

The basic model of web applications is shown in Figure 1.1. Transport of information between the browser and the web server is provided by the Hyper-Text Transport Protocol, HTTP (Section 6.2.1), which runs over Transmission Control Protocol, TCP (Section 6.2.9), or in some new

implementations, over the WebSocket protocol (see Section 6.2.2). The content or application is carried in Hyper-Text Markup Language, HTML, which typically includes JavaScript and Cascading Style Sheets [CSS]. In the simple case, the browser sends an HTTP request to the web server for content, and the web server sends a response containing the document or image or other information requested. In the more complex case, the server sends JavaScript which runs on the browser, interacting with the browser through APIs and with the user through clicks and selects. The browser exchanges information with the server through an open HTTP or WebSockets channel.

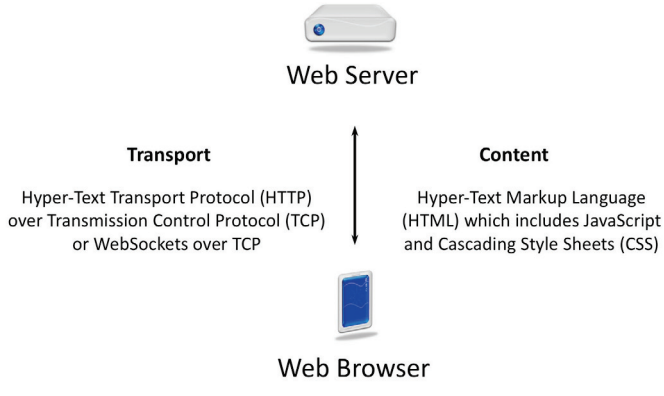

Figure 1.1 Web Browser Model

In figures in this book, we will show an arrow between the web browser and the web server to indicate the web session between them. Since WebRTC can utilize any web transport, the details of this connection, and whether it is HTTP or WebSockets is not discussed.

1.1.2 The Real-Time Communication Function in the Browser

Figure 1.2 shows the browser model and the role of the real-time communication function. The lighter block called "Browser RTC Function" is the focus of this book. The unique nature and requirements of real-time communications means that adding and standardizing this block is non-trivial. The RTC function interacts with the web application using standard APIs. It communicates with the Operating System using the browser. A new aspect of WebRTC is the interaction that occurs browser-to-browser, known as a "Peer Connection", where the RTC Function in one browser communicates using on-the-wire standard protocols (not HTTP) with the RTC Function in another browser or Voice over IP (VoIP) or video application. While web traffic uses TCP for transport, the on-the-wire protocol between browsers can use other transport protocols such as

User Datagram Protocol, UDP. Also new is the Signaling Server, which provides the signaling channel between the browser and the other end of the Peer Connection.

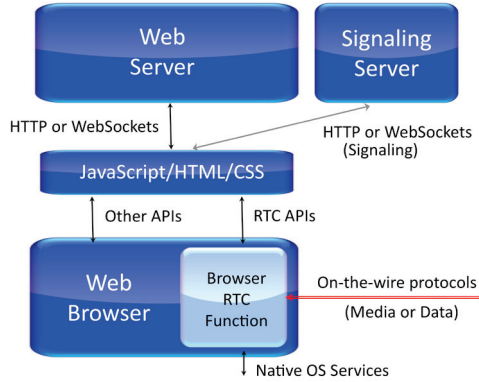

Figure 1.2 Real-Time Communication in the Browser

1.1.3 Elements of a WebRTC System

Figure 1.3 shows a typical set of elements in a WebRTC system. This includes web servers, browsers running various operating systems on various devices including desktop PCs, tablets, and mobile phones, and other servers. Additional elements include gateways to the Public Switched Telephone Network (PSTN) and other Internet communication endpoints such as Session Initiation Protocol (SIP) phones and clients or Jingle clients. WebRTC enables communication among all these devices. The figures in this book will use these icons and elements as examples.

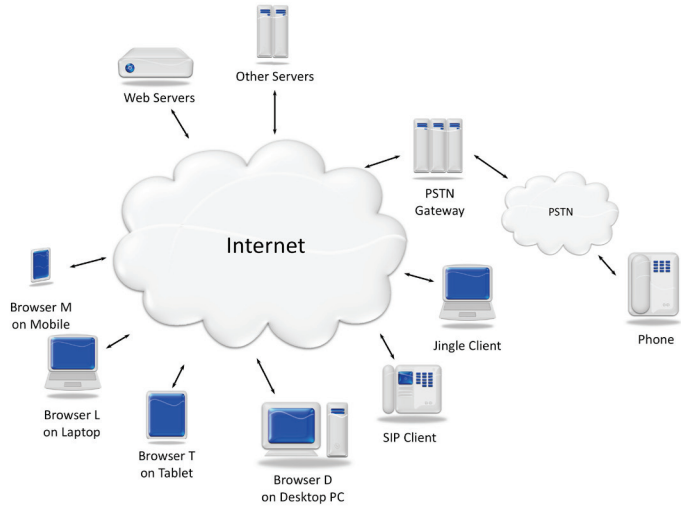

Figure 1.3 Elements in a WebRTC Environment

1.1.4 The WebRTC Triangle

Initially, the most common scenario is likely to be where both browsers are running the same WebRTC web application, downloaded from the same webpage. This produces the WebRTC "Triangle" shown in Figure 1.4. This arrangement is called a triangle due to the shape of the signaling (sides of triangle) and media or data flows (base of triangle) between the three elements. A Peer Connection establishes the transport for voice and video media and data channel flows directly between the browsers.

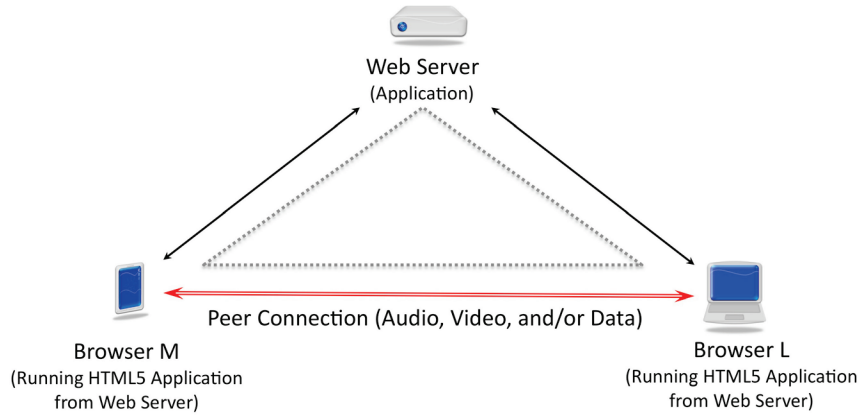

Figure 1.4 The WebRTC Triangle

Note that while we sometimes refer to the connection between the browser and server as signaling, it is not really signaling as used in telephony systems. Signaling is not standardized in WebRTC as it is just considered part of the application. This signaling may run over HTTP or WebSockets to the same web server that serves HTML pages to the browser, or to a completely different web server that just handles the signaling.

1.1.5 The WebRTC Trapezoid

Figure 1.5 shows the WebRTC Trapezoid [draft-ietf-rtcweb-overview], based on the SIP Trapezoid [RFC3261]. The two web servers are shown communicating using a standard signaling protocol such as Session Initiation Protocol (SIP), used by many VoIP and video conferencing systems, or Jingle [XEP-0166], used to add voice and video capability to Jabber [RFC6120] instant messaging and presence systems. Alternatively, a proprietary signaling protocol could be used. Note that in these more complicated cases, the media may not flow directly between the two browsers, but may go through media relays and other elements, as discussed in Chapter 3.

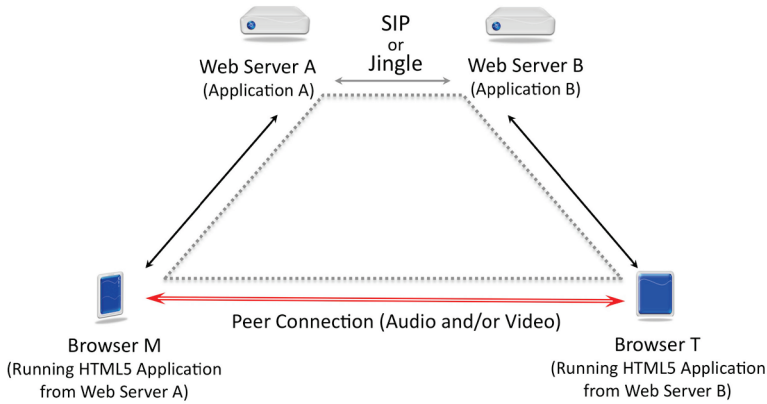

Figure 1.5 The WebRTC Trapezoid

1.1.6 WebRTC and the Session Initiation Protocol (SIP)

Figure 1.6 shows WebRTC interoperating with SIP. The Web Server has a built-in SIP signaling gateway to allow the call setup information to be exchanged between the browser and the SIP client. The resulting media flow is directly between the browser and the SIP client, as the Peer Connection establishes a standard Real-time Transport Protocol (RTP) media session (Section 6.2.3) with the SIP User Agent. Other ways of interoperating with SIP are covered in Section 2.2.6.

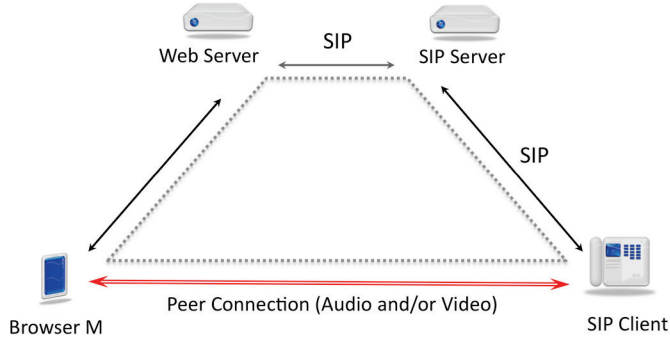

Figure 1.6 WebRTC Interoperating with SIP

1.1.7 WebRTC and Jingle

Figure 1.7 shows how WebRTC can interoperate with Jingle. The Web Server has a built-in Extensible Messaging and Presence Protocol, XMPP [RFC6120], also known as Jabber, server which talks through another XMPP server to a Jingle client.

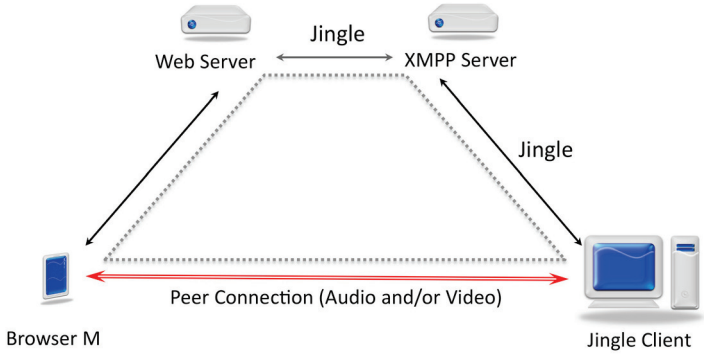

Figure 1.7 WebRTC Interoperating with Jingle

1.1.8 WebRTC and the Public Switched Telephone Network (PSTN)

Figure 1.8 shows how WebRTC can interoperate with the Public Switched Telephone Network (PSTN). The PSTN Gateway terminates the audio-only media stream and connects the PSTN telephone call with the media. Some sort of signaling is needed between the Web Server and the PSTN Gateway. It could be SIP, or a master/slave control protocol.

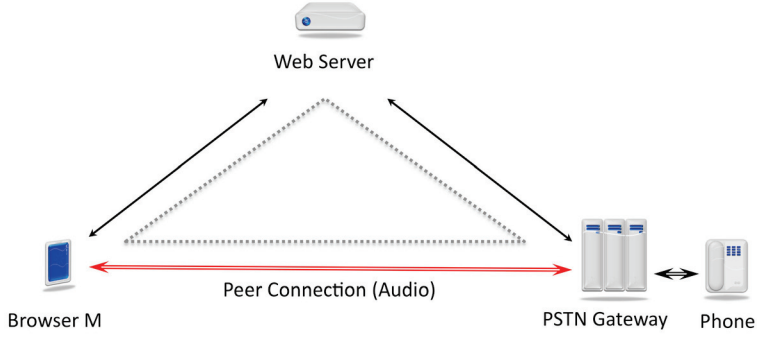

Figure 1.8 WebRTC Interoperating with the PSTN

It is not expected that browsers will be assigned telephone numbers or be part of the PSTN. Instead, an Internet Communication service could assign a telephone number to a user, and that user could use WebRTC to access the service. As a result, a telephone call to that PSTN number would "ring" the browser and an answered call would result in an audio session across the Internet connected to the PSTN caller. Other services could include the ability to "dial" a telephone number in a WebRTC application which would result in the audio path across the Internet to the PSTN.

Note that the phone in Figure 1.8 could be a normal PSTN phone ("landline" or "black phone") or a mobile phone. The fact that it might be

running VoLTE (Voice over Long Term Evolution) or other VoIP (Voice over Internet Protocol) protocol doesn't change this picture, as the Peer Connection will terminate with a VoIP gateway.

Another interesting area is the role of WebRTC in providing emergency services. While a WebRTC service could support emergency calling in the same way as VoIP Internet Communication services, there is the potential that the Public Service Answering Point (PSAP) could become a WebRTC application, and answer emergency "calls" directly from other browsers, completely bypassing the PSTN. Of course, this raises all kinds of interesting security, privacy, and jurisdiction issues.

1.2 Multiple Media Streams in WebRTC

Devices today can generate and consume multiple media types and multiple streams of each type. Even in the simple point-to-point example shown in Figure 1.9, a mobile phone and a desktop PC could generate a total of six media streams. For multiparty sessions, this number will be much higher. As a result, WebRTC has built-in capabilities for dealing with multiple media streams and sources.

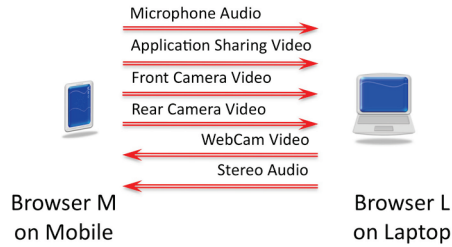

Figure 1.9 Multiple Media Streams in a Point-to-Point WebRTC Session

1.3 Multi-Party Sessions in WebRTC

The preceding examples have been point-to-point sessions between two browsers, or between a browser and another endpoint. WebRTC also supports multi-party or conferencing sessions involving multiple browsers.

One way to do this is to have each browser establish a Peer Connection with the other browsers in the session. This is shown in Figure 1.10. This is sometimes referred to as a "full mesh" or "fully distributed" conferencing architecture. Each browser establishes a full mesh of Peer Connections with the other browsers. For audio media, this might mean mixing the media received from each browser. For video, this might mean rendering the video streams from other browsers to different windows with appropriate labeling. As new browsers join the session, new Peer Connections are established to send and receive the new media streams.

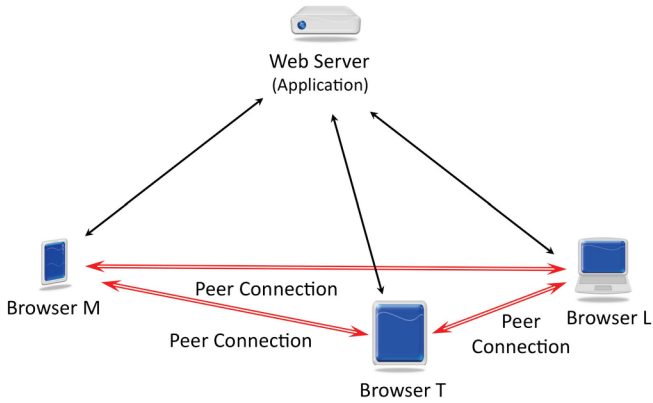

Figure 1.10 Multiple Peer Connections Between Browsers

An alternative architecture to the full-mesh model of Figure 1.10 is also possible with WebRTC. For a multiple browser conference, a centralized media server/mixer/selector can be used; this requires only a single Peer Connection to be established between each browser and the media server. This is shown in Figure 1.11. This is sometimes referred to as a "centrally mixed" conferencing architecture. Each browser sends media to the server, which distributes it to the other browsers, with or without mixing. From the perspective of browser M, media streams from browser L and T are received over a single Peer Connection from the server. As new browsers join the session, no new Peer Connections involving browser M need to be established. Instead, new media streams are received over the existing Peer Connection between browser M and the media server.

The full-mesh architecture of Figure 1.10 has the advantages of no media server infrastructure, and lowest media latency and highest quality. However, this architecture may not be suitable for a large multi-party conference because the bandwidth required at each browser grows with each new participant. The centralized architecture of Figure 1.11 has the advantage of being able to scale to very large sessions while also minimizing the amount of processing needed by each browser when a new participant joins the session, although it is perhaps inefficient when only one or a small number of browsers are involved, such as in peer-to-peer gaming.

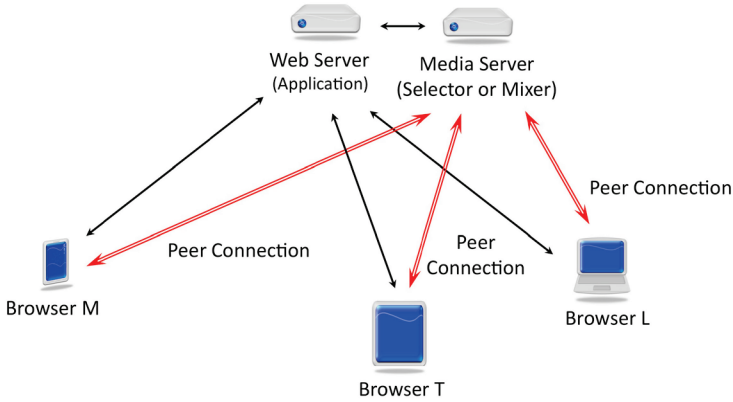

Figure 1.11 Single Peer Connection with Media Server

1.4 WebRTC Standards

The WebRTC standards are currently under joint development by the World Wide Web Consortium (W3C) [W3C] and the Internet Engineering Task Force (IETF) [IETF]. W3C is working on defining the APIs needed for JavaScript web applications to interact with the browser RTC function. These APIs, such as the Peer Connection API, are described in Chapter 5. The IETF is developing the protocols used by the browser RTC function to talk to another browser or Internet Communications endpoint. These protocols, for example, extensions to the Real-time Transport Protocol, are described in Chapter 8.

There are pre-standard implementations of many of the components of WebRTC in some browsers today. See Chapter 8 for details.

> Note that there is an important distinction between 'pre-standard' and 'proprietary' implementations. Pre-standard implementations emerge during the development stage of standards, and are critical to gain experience and information before standards are finalized and locked down. Pre-standard implementations often follow an early or draft version of the standards, or partially implement standards as a 'proof of concept'. Once the standard has been finalized, these pre-standard implementations must move towards the standards, or else they risk becoming a proprietary implementation. Proprietary implementations fragment the user and development base, which in an area such as communications can greatly reduce the value of the services.

The W3C work is centered around the WEBRTC Working Group and the IETF work is centered around the RTCWEB (Real-Time

Communications Web) Working Group. The two groups are independent, but closely coordinate together and have many common participants, including the authors.

The projected time frame for publication of the first version of standards in the IETF and W3C is in 2014. However, these dates are most likely overly optimistic. (When do engineers ever realistically estimate level of effort?) Standards-compliant WebRTC browsers are expected to be generally available sometime in late 2014.

1.5 What is New in WebRTC

There are many new and exciting capabilities in WebRTC that are not available even in today's VoIP and video conferencing systems. Some of these features are listed in Table 1.1. The rest of this book will explain how these are achieved using the WebRTC APIs and protocols.

1.6 Important Terminology Notes

In this book, when we refer to the entire effort to add standardized communication capabilities into browsers, we shall use WebRTC. When we are referring to the W3C Working Group, we will use WEBRTC. When we are referring to the IETF Working Group, we will use RTCWEB. Note that WebRTC is also used to describe the Google/Mozilla open source media engine [WEBRTC.ORG], which is an implementation of WebRTC.

In addition, because the main W3C specification is titled "The WebRTC Specification" [WEBRTC 1.0], we use its full title to reference this particular W3C document, which is a key part of WebRTC, but by no means the entire specification.

Also note that the World Wide Consortium refers to itself as "W3C" and not "the W3C". We have adopted this convention throughout this book.

Feature	Provided Using	Why Important
Platform and device independence	Standard APIs from W3C, standard protocols from IETF	Developers can write WebRTC HTML5 code that will run across different OS, browsers, and devices, desktop and mobile.
Secure voice and video	Secure RTP Protocol (SRTP) encryption and authentication	Browsers are used in different environments and over unsecured WiFi networks. Encryption means that others can't listen in or record voice or video.
Advanced voice and video quality	Opus audio codec, VP8 video codec, and others	Having built-in standard codecs ensures interoperability and avoids codec downloads, a way malicious sites install spyware and viruses. New codecs can adapt when congestion is detected.
Reliable session establishment	Hole punching through Network Address Translation (NAT)	Direct media between browsers is noticeably more reliable and better quality than server-relayed media. Also, the load on servers is reduced.
Multiple media streams and media types sent over a single transport	Real-time Transport Protocol (RTP) and Session Description Protocol (SDP) extensions	Establishing direct media using hole punching can take time. Sending all media over a single session is also more efficient and reliable.
Adaptive to network conditions	Multiplexed RTP Control Protocol (RTCP), Secure Audio Video Profile with Feedback (SAVPF)	Feedback on network conditions is essential for video, and will be especially important for the high definition, high bandwidth sessions in WebRTC.
Support for multiple media types and multiple sources of media	APIs and signaling to negotiate size/format of each source individually	The ability to negotiate each individually results in most efficient use of bandwidth and other resources.
Interoperability with VoIP and video communication systems using SIP, Jingle, and PSTN	Standard Secure RTP (SRTP) media, Standard SDP and extensions	Existing VoIP and video systems can work with new WebRTC systems using standard protocols.

Table 1.1 New Features of WebRTC

1.7 References

[HTML5] http://www.w3.org/TR/html5

[SKYPE] http://www.skype.com

[CSS] http://www.w3.org/Style/CSS

[draft-ietf-rtcweb-overview] http://tools.ietf.org/html/draft-ietf-rtcweb-overview

[RFC3261] http://tools.ietf.org/html/rfc3261

[XEP-0166] http://xmpp.org/extensions/xep-0166.html

[RFC6120] http://tools.ietf.org/html/rfc6120

[W3C] http://www.w3c.org

[IETF] http://www.ietf.org

[WEBRTC.ORG] http://www.webrtc.org

[WEBRTC 1.0] http://www.w3.org/TR/webrtc