WebRTC and RTCWEB – The World Soon Upside Down

Guenter I. Klas
Feb 19, 2013

The 1 minute takeaway

WebRTC is a technology that introduces real-time communication directly in web browsers. This avoids the download and installation of plugins and renders the user experience with audio/video services on the web significantly smoother than today. The WebRTC standard from the World Wide Web Consortium W3C provides new elegant programming interfaces for web developers to leverage the underlying new technology built into web browsers. That lower-layer technology makes use of a suite of protocol standards being bundled together for this purpose by the Internet Engineering Task Force IETF under the name RTCWEB. WebRTC goes even beyond pure audio/video communication peer-to-peer between web browsers. It also supports real-time communication for file and screen sharing as well as gaming. The specifications from IETF are not limited in their use to web browsers as end points. Google and Mozilla have been early adopters of WebRTC and RTCWEB, whereas Microsoft surprised the W3C community with their own proposal elaborated by Microsoft and Skype engineers. Though the standards for WebRTC are not finished yet and several challenges still have to be overcome, both prototypes and early products have come to market that demonstrate the benefit of real-time communication support 'natively' built into web browsers. The next 12-24 months will show to which extent this technology will be disruptive, fuel over-the-top communication services and enhance consumer-facing online web applications. Both small technology companies as well as big telco service providers have jumped on the bandwagon. This space is clearly worthwhile to monitor.

Tags: WebRTC, RTCWEB, W3C, IETF, Skype, Voxeo, Twilio, AddLive, VP8, SDP, RTCPeerConnection, RTCDataChannels, MediaStream, getUserMedia, real-time communication, API

WebRTC, standing for Web Real-time Communication, will gain some well-deserved attention in the months to come for a number of reasons: It may turn out to be disruptive, to drive some fights in the industry, to fuel innovation and to delight software developers and consumers alike. Less well known is RTCWEB, its sort of companion, the engine that works in the background for real-time peer-to-peer communication between browsers.

The purpose

Web technology and browsers have improved dramatically over the past years. More and richer features have received direct support in Web browsers (e.g. graphics). With WebRTC, browsers can do things that before would have required a plugin or a dedicated client to be downloaded and installed by a user (e.g. as with Skype). WebRTC makes browsers support phone calls, video calls, conferencing, sharing of nearly everything (audio, files, screens), gaming and of course all sorts of messaging. All this without the need to handle pop-up blockers, to install plugins or 3rd party software. WebRTC is the German Autobahn for real-time communication straight out of the browser: fast and smooth.

Main elements of this technology are being standardised by two organisations: The World Wide Web Consortium W3C and the Internet Engineering Task Force IETF. W3C defines what web developers will need to know (WebRTC). IETF defines what browser implementers and infrastructure vendors will need to know (RTCWEB).

What are the claimed benefits?

- Free, standardised communications technology built directly into web browsers.
- No need for download of plugins, no issues with their update, smoother user experience.
- Web developers can deploy real-time-related services more easily and more universally across devices around the Globe.

What about the industry support?

First, there is the support which is needed directly in web browsers. Several browser implementers have shown early implementations of WebRTC, including Google, Mozilla and Opera. As usual, the browser companies still put their innovations into desktop browsers first, before they bring the same features to browsers on smartphones. In the meantime, we find WebRTC support e.g. in Google Chrome and Mozilla Firefox Nightly. Solutions (by 3rd parties) for Apple's iPhone operating system iOS and Google's Android smartphone operating system have already been shown too. Support in browsers is generally mixed. Typically not all aspects of the specifications agreed in standards are implemented in equal measures across browsers today, though solutions will become more complete over time. Good news is that WebRTC calls have already been demonstrated between the Chrome and the Firefox browser in Feb 2013 [1]. For a YouTube video about this, see [2].

Second, there are the companies which use WebRTC as a new enabler. They experiment with new types of communication services using WebRTC as a nice browser-built-in feature. Some of those companies are Voxeo Labs, Twilio and Solaiemes.

As WebRTC is the talking point these days when it comes to real-time communication on the Web, no wonder that telecommunication service providers including mobile network operators have started to take notice and some jumped into action. Partnering is overdue then: Deutsche Telekom announced a partnership with Voxeo, KDDI with Twilio and so did AT&T. Telefonica showed greater appetite and simply acquired a company knowledgeable in WebRTC, namely TokBox.

No doubt, Google and Mozilla with a community of companies in W3C and IETF have been driving WebRTC (and RTCWEB) pretty successfully so far. There will be others whose existing services will be impacted, including Microsoft/Skype. Not surprising that Microsoft has taken a different route through their proposal called Cu-RTC-Web which apparently got first announced in Aug 2012 [3].

As Peter Bright argues in [4], Microsoft appears to dislike the use of SDP (the Session Description Protocol) made mandatory, whereas others would argue it eases interoperability with SIP-based systems. Mind, SDP is also used by mobile network operators in the so-called 3GPP IMS system for IP-based multimedia, which may well ease interworking of WebRTC with the network operators' IMS-based services. Reportedly neither Skype nor Microsoft Lync use SDP in their core. On the other hand, as Peter argues, Microsoft's concern over SDP may be justified, in particular if WebRTC / RTCWEB required the current SDP to be extended to support W3C use cases and if greater demand arose for browser to non-browser communication.

Thus, we need to see, whether some of the issues surrounding SDP turn out to be show-stoppers in the future or not. Though Microsoft's proposal was voted down by WebRTC folks, it doesn't mean the issues Microsoft pointed out have been automatically resolved.

And Apple? Well, shrouded in secrecy so far.

On the computing infrastructure side possibly some kind of *WebRTC Platform as a Service* (PaaS) propositions may get traction (Chad Hart calls it WebRTCaaS [5]). There, WebRTC technology features and services are used by websites in a PaaS/cloud model. As companies active in this space he lists Twilio, Voxeo, Hookflash, AddLive, TenHands, and VidTel.

To which extent will WebRTC cause some headache for telecommunications companies?

AT&T positioned themselves with regard to WebRTC already at their Developer Summit in Las Vegas in early 2013. It seems that a typical carrier's strategy won't be fighting WebRTC. Instead, it will most often be about embracing the new technology and limiting damage to existing telco business models through enriching any WebRTC-enabled services with telco ingredients of value. Interworking with 'legacy' audio/video services or existing VoIP services is one area where value-add can be created. This is also picked up by telco infrastructure vendors like Ericsson. As they say, their solutions will not only support web-to-web communication but also web-to-IMS interworking.

Who does the standards?

The world-wide web consortium W3C has been drafting relevant JavaScript *Application Programming Interfaces* (APIs) for the browser [10]. The W3C specification for WebRTC is expected to be ratified in 2014.

The Internet Engineering Task Force IETF has defined an enabling *protocol specification* in its working group called RTCWEB [11].

Both standardisation efforts are not fully completed yet as of the date of this writing.

Overall, W3C and IETF have somehow managed to agree on a beneficial work split.

W3C has standardised the JavaScript APIs which Web developers are so keen on when coding their web applications nowadays with HTML 5, CSS and JavaScript.

IETF has specified the "over wire protocol" suite called RTCWEB protocol specification. Most importantly it defines which existing protocols should be used and bundled together to support WebRTC services and to expose control over this suite to JavaScript developers via the APIs defined by W3C. IETF's RTCWEB defines protocols and elements to use for data transport, data framing, data formats (including codecs) and connection management (SDP).

The "over wire" or "on-the-wire" protocols from IETF run between supporting servers (for signalling) and between browsers (for the media path).

Some hot debates centred in the past around the type of mandatory video codec to choose for browsers (VP8 versus H.264). Ideally it should be first class and royalty free. The "first call" between Google Chrome and the Mozilla Firefox browser has been made using VP8 in Feb 2013.

As far as the standardisation process is concerned, are we witnessing a perfect collaboration between two organisation? As it says in [6]: "Together, these two specifications aim to provide an environment where JavaScript embedded in any page, viewed in any compatible browser, when suitably authorized by its user, is able to set up communication using audio, video and auxiliary data, where the browser environment does not constrain the types of application in which this functionality can be used."

Fair to say that IETF go even further: RTCWEB is of course the great companion of WebRTC, but not limited to it. "From RTCWEB protocol point of view, the endpoint can also be something different from a browser". Thus, it could be for instance some native application (say in iOS or Android) or a machine-to-machine terminal software that collects sensor data and transmits it via the RTCWEB protocol suite.

What are the bits and pieces needed for a good WebRTC service implementation?

First, we need certain features and application programming interfaces which get built into browsers and which are finally used by web service developers. These JavaScript APIs shield Web developers from the underlying complexity. This includes the RTCPeerConnection API that supports audio/video communication between browsers, the RTCDataChannels API that has been specified to enable high performance, low latency, peer-to-peer communication of arbitrary data directly between browsers including file and screen sharing or gaming, and the MediaStream API to capture the user's media on a browser from microphone and camera.

Then we need mechanisms to agree on sessions, like the JavaScript Session Establishment Protocol JSEP.

We need mighty functions inside the browsers that handle everything related to real-time services like voice and video: Audio/video engines that include e.g. codecs like Opus for wideband music quality audio (what a new phone conversation quality we can expect now via the browser!) and VP8 (or H.264) for video (stunning video full-screen on my 24'' monitor), jitter buffers, image enhancers, noise reduction etc.

Then we need a protocol that handles real-time communication including data framing, loss detection, lip synchronization, error recovery (RTP) and one that provides statistics about that (RTCP). On top, security (confidentiality and integrity) is catered for by encrypting data and signalling and by shipping keys. For this secure protocols are used (SRTP for symmetric key cryptography, DTLS over UDP to exchange session keys). To support generic data streaming between peers e.g. for gaming, additional protocols like SCTP are needed. Finally the browser will support mechanisms to deal with NAT/firewall traversal.

Second, we need server support for various reasons (despite WebRTC enabling peer-to-peer real-time data streaming): To enable signalling between the remote browsers, to support NAT/firewall traversal using the ICE framework with protocols like STUN and TURN involved to connect peers directly or via a relay server, and to discover real-time service users around the Globe (who is out there ready to be communicated with!).

Finally, WebRTC-enabled services may need to tap into identity solutions in order to allow communicating users to verify each other's identity.

Looking under the hood: How does it actually work?

A browser together with the JavaScript code of a web page that offers an out-of-the-box WebRTC audio/video service (or gaming service etc.) must do a number of things:

1. First, the web page's code gets streaming audio, video or other data from the desktop, laptop, tablet or smartphone or any other gadget that runs a Web browser. For this purpose, the web page developer will use the MediaStream API which uses a method unsurprisingly called getUserMedia(). The captured media stream will be typically passed on to an HTML 5 element of the same page to output this local audio/video via the user's browser. The same media stream will be passed to the other party via the RTCPeerConnection API.
2. Second, some signalling happens between the local web application and the remote peer application via a server in-between.
   - Network configuration: The local web application gets its own network information (IP address and TCP/UDP port), and exchanges this with other WebRTC clients via signalling to enable a connection.
   - Session control: It then coordinates signaling communication to initiate or close a session.
   - Exchange of media capabilities: The local system exchanges information about media and client capabilities with the far end (e.g. screen resolution, codecs available on the local device). The well-known Session Description Protocol SDP (see SIP, IMS) is used for this purpose. One party offers the capabilities it can support, the other party answers with what it feels is best to choose. This offer/answer architecture is realized with the JavaScript Session Establishment Protocol JSEP.

Importantly, signaling methods and protocols are not specified by WebRTC. This means, a WebRTC service developer can choose any suitable messaging protocol for signalling purposes (e.g. SIP, XMPP, WebSocket or others).

3. Once the signalling has successfully completed, the browser can ignore the in-between server and send streaming audio, video or generic data (e.g. for screen or file sharing, gaming…) directly to the peer browser at the far end (thus peer-to-peer). For this, it uses the RTCPeerConnection API for audio and video (RTCDataChannel API for generic data like screen sharing or gaming).

Where is the money flowing and what products can we expect?

WebRTC in Web browsers constitutes an enabler, the one on consumer devices and usable by web and web apps developers. Other products emerge like gateways to support WebRTC communication with the good old telephony service and equally with the now classic Voice over IP services. Finally, there is no limit to building real-time service support (be it voice, video, or screen sharing) into all sorts of consumer-facing web applications. This could be related to customer care and service assistance (e.g. in a medical scenario, for patients or elderly people). More generally, voice/video/data sharing capabilities can be added to existing business, enterprise and collaboration software where it makes sense (e.g. for on-line instant advice) with low hurdles for deployment. Finally, as mentioned above, companies will enter the market with cloud-based solutions that assist service developers in bolting together a new WebRTC-enabled web application.

Is it really new?

VoIP and video conferencing over the Internet have been here for a while, and so related services like Skype, Cisco’s WebEx or Citrix’s GoToMeeting. Equally, as pointed out in [5] things like Real-Time Messaging Protocol (RTMP) as available in Adobe Flash may be regarded as existing predecessors. WebRTC in browsers, however, eliminates some of the complexities and inconveniences enterprise users and consumers had to face so far: downloading a Skype client (and downloading it again when it says a new version is available), handling pop-up windows and security alerts when starting WebEx or simply needing to wait several minutes until the web conferencing tool is ready to go. With WebRTC that should be a matter of the past.

Overall, WebRTC and RTCWEB (worthwhile to mention both) constitute a major development. It’s still early days as the first WebRTC call between Google and Mozilla teams showed in Feb 2013 [7]. More needs to be done and will be done. And more news will hit the web about companies like Voxeo [8] or AddLive [9] innovating in this space.

References

[1] Sam Shead, Chrome and Firefox showcase video chat via WebRTC. http://www.zdnet.com/chrome-and-firefox-showcase-video-chat-via-webrtc-7000010915/ 6 Feb 2013.

[2] WebRTC conversation between Google Chrome's director of product management, Hugh Finnan, and Mozilla's chief of innovation, Todd Simpson. http://www.youtube.com/watch?feature=player_embedded&v=MsAWR_rJ5n8 3 Feb 2013.

[3] Microsoft’s proposal for WebRTC. http://html5labs.interoperabilitybridges.com/cu-rtc-web/cu-rtc-web.htm

[4] Peter Bright, Microsoft goes its own way with Web audio/video spec, despite W3C rebuff. http://arstechnica.com/information-technology/2013/01/microsoft-goes-its-own-way-with-web-audiovideo-spec-despite-w3c-rebuff/. 19 Jan 2013.

[5] Chad Hart, Is there a WebRTC ecosystem? http://lphs.acmepacket.com/blog/bid/168799/Is-there-a-WebRTC-ecosystem, Jan 2013.

[6] http://tools.ietf.org/pdf/draft-ietf-rtcweb-overview-05.pdf

[7] Ian Elliot, First Contact - Firefox & Chrome WebRTC. http://www.i-programmer.info/news/86-browsers/5421-first-contact-firefox-a-chrome-webrtc.html, 5 Feb 2013.

[8] Voxeo Labs's Phono: http://phono.com/webrtc. Demonstrating phone calls between Web browser and old legacy phone network.

[9] Some WebRTC service inspiration: http://www.addlive.com

[10] Documents on webRTC at W3C: http://www.w3.org/2011/04/webrtc/

[11] Documents on RTCWEB at IETF: http://tools.ietf.org/wg/rtcweb/