webRTC: the premise and the promise
EXECUTIVE OVERVIEW

WebRTC is considered a nascent industry standard for incorporating real-time communications capabilities into web browsers and applications. This paper explores the current status of this standard, reviews WebRTC’s evolution and challenges, and attempts to determine whether WebRTC should be considered a standard for developers seeking to add real-time communications to their applications.

BACKGROUND OF WEBRTC

WebRTC (Web Real-Time Communications) is an open-source project enabling plugin-free, browser-to-browser communications. It includes the fundamental building blocks for multimedia communications such as audio, P2P file sharing, and video components used in mobile and web browser applications.

Prior to the advent of WebRTC in 2010, developers relied on cumbersome plug-ins (like Adobe Flash), or multivendor browser extensions that required download. The vision behind WebRTC was to address these issues by creating an interoperable open standard framework, eliminating the need for the aforementioned plugins and downloads, and enabling web developers to add real-time communications to their applications with simple JavaScript APIs.

This paper investigates the extent to which WebRTC delivers on this promise.

ADVANTAGES OF WEBRTC

WebRTC is free to implement, lowering the barrier to entry for developers. This is especially true for those who specialize in building OTT (over-the-top) consumer apps. Building real-time capabilities for consumer apps can be cost prohibitive, as such business models often generate low revenues per user.

Leveraging WebRTC reduces the complexities of implementing audio, video, and telephony for web developers without domain expertise. By using JavaScript application programming interfaces (APIs), the large community of web developers can add real-time communications quickly, without worrying
about long-development cycles, firewall traversal, licensing codecs, inefficient plugins, or distributing software clients.

WebRTC is not a compression protocol like H.264 or G.729, a communication protocol like SIP or H.323, or a programming language. Instead, it is the embedding of the baseline elements needed for audio and video communication within some browsers. As a result, simple audio and video calling is only a click or two away from users in supported environments.

CHALLENGES FACING WEBRTC

A cornerstone of WebRTC’s vision of interoperability is that it must be supported ubiquitously by all widely-used browsers. With this support, developers could build applications assuming end-user compatibility, regardless of browser. Unfortunately, this is not the case. WebRTC is currently only supported by Google Chrome and Mozilla Firefox. Google has long championed WebRTC, specifically by providing the VP8 & VP9 video codecs WebRTC uses, and ensuring Chrome has the most advanced WebRTC support. Though Google has a vested interest in promoting the open web and standards that support the web as a platform, other vendors may not benefit from WebRTC and have no incentive to capitulate to Google. Both Apple and Microsoft do not currently support these codecs, with Microsoft supporting the H.264 codec and Apple supporting neither. In fact, Microsoft has developed it’s own alternative and possible competitor to WebRTC, oRTC, which it has integrated into the Microsoft Edge browser. Apple has not revealed its stance on WebRTC, but does not currently support it. This is particularly significant in view of Apple’s continued dominance in the mobile sector. It is also important to note that the last time there was an omnipresent video solution for the browser (Adobe Flash), Apple impeded it’s ubiquity by refusing to allow it on iOS.
Another challenge for WebRTC is that it does not take advantage of Apple’s hardware acceleration technology. As a result, it does not optimize the use of available processing power. It also does not adapt to fluctuating network bandwidth, which would need to be built from scratch to adapt to wavering connectivity. This can be frustrating on mobile devices, on which users are frequently moving between high and low bandwidth environments. WebRTC is unable to adjust seamlessly to the available capacity, negatively impacting user experience with degradation of voice and video quality and lost connections. In addition, it does not include multi-party conferencing, call signaling (e.g. XMPP, H.323, SIP), or work behind proxy servers, which can be drawbacks in the case of establishing these features ubiquitously in enterprise software.

Given these limitations, it would be difficult for WebRTC to be considered a standard or to deliver a comparable experience to traditional audio and video conferencing software, most of which offer multi-party conferencing, high-definition, call-speed control, and screen sharing (which is only possible through native C++ applications). This is because developers building such software maintain control over their technology, while WebRTC’s functionality remains tied to the native capabilities of web browsers. Thus, WebRTC is not scalable without significant investment in infrastructure and development, to bridge the gap between WebRTC and traditional video conferencing.

WEBRTC ON MOBILE

WebRTC’s support for mobile devices is an important factor when evaluating its practical implementation. The growth of mobile device usage has been exponential, with phones and tablets being used widely to access applications and services. Consequently, when formulating a real-time communications strategy, mobile performance must be considered.

The current industry practice for delivering a mobile experience is primarily through native applications that are written specifically for a particular platform or operating system. WebRTC is not supported on iOS devices (regardless of browser), and is supported for Android devices (on Chrome). This support is not powered natively within the application, but through a peer-to-peer connection in the browser. This is because WebRTC’s open source libraries cannot be embedded into an application, as they would be with a mobile SDK. As a result, development can be challenging and time consuming. Several differences
between device types need to be addressed individually by the application, including display resolution, CPU power, microphone settings, camera quality, etc.

APPLICATIONS OF WEBRTC

Ultimately, determining whether WebRTC is the optimal choice is dependent on the nature of the application and the intended target audience.

Specifically, WebRTC could be a good fit for an OTT consumer desktop application that would benefit from adding basic peer-to-peer conferencing, an application in a locked down environment that doesn’t allow software installations, or a website that needs a click to call button (e.g. contact centers, customer support).

However, for the enterprise, where security, mobility, reliability, and quality of service are business-critical, WebRTC is not ideal. WebRTC is defined for basic peer-to-peer communications between browsers. It is not a framework for implementing an enterprise grade system. In addition to WebRTC’s issues with interoperability, mobility, and quality, there are questions being raised about security, particularly with the potential for denial of service attacks and fraud via the browser interface.

HOW MOXTRA CAN HELP

For developers seeking an engine more suited to an enterprise environment, purpose-built and off-the-shelf mobile collaboration platforms are more suitable. Real-time collaboration requires more than just voice and video calling. It should involve seamless experiences for multiple users, (including content sharing, desktop sharing, and/or whiteboarding) regardless of device type, browser, or operating system. Additionally, the value of on demand, asynchronous communication should be noted. WebRTC only operates in a synchronous communications mode (e.g. everyone is in communication at the same time), but lacks the ability to record, store, and playback voice or video content for users.

Moxtra provides a collaboration ecosystem that goes far beyond communication. The incorporation of both asynchronous and synchronous communication in one platform, accounts for the natural variance
in communication styles. Moxtra offers four distinct embeddable SDKs for iOS, Android, and JavaScript, as well as comprehensive REST APIs. The modular architecture enables developers to select and embed the capabilities they need with just a few lines of code, on any device. Furthermore, the Moxtra plugin allows for complete browser and operating system ubiquity as well as enterprise grade security with multi-layered encryption for all data and traffic. Additionally, Moxtra leverages SVC (scalable video codec), enabling intelligent adaptations to fluctuating or low network bandwidth situations. Moxtra also optimizes CPU resources for low latency and clarity in audio and video.

CONCLUSION

Though WebRTC aims to become a standard in real-time communications, it suffers from significant quality issues and it lacks native interoperability with standards-based systems. This is specifically due to discord amongst the major players in deciding on a standard method of adoption, or in conceding to a codec proposed by another company. So when does WebRTC make sense today?

When evaluating which real-time components to integrate, there are a few questions to consider: What are the goals for the product and company? What caliber of functionality is necessary? If a product needs basic 1:1 voice or video calling on a specific set of browsers, WebRTC would be a good solution. However, if a product would benefit from any combination of multi party voice, video, messaging, or screen sharing, a multi layered solution like Moxtra would be ideal. This is especially true with users in excess of 3, or with global participants.

Presently, the key issue with opting for a WebRTC driven solution is that users would be impacted by browser incompatibility and possible losses in quality. Both of these can negatively affect the user experience of a product, and ultimately tarnish a brand. Thus, until WebRTC is able to achieve its promise of ubiquitous, secure, and high quality real-time communications, it may not be a prudent choice for enterprise grade solutions.