Report from RTC-Web Workshop

Mountain View, October 6, 2010

(Publication note: This document is intended to be publicly visible, and will be called to the attention of all workshop participants.)

Executive summary

Extending the browser with functionality that allows interactive audio and video directly between users is an idea that has great value. It is also possible to do.
All present agree that a standardized platform of APIs and protocols, that allows applications running in a Web page in one browser to communicate using audio and video with another application running in another browser, will greatly facilitate the development and deployment of such applications.
We will work together to try to define and realize such a platform.

Workshop goals

The goal of the workshop was to start the discussion among a (relatively) small set of key actors in the industry to see if it was possible to reach agreement on whether building a standardized platform for real time collaboration in the browser was a Good Thing, and if it was possible to reach some common understanding of what properties such a platform needed to have to be useful.
The workshop did not have as a goal to reach any decisions about what the standards should be; this is an exercise best left to work in existing standards organizations.

Workshop participants

The participants came from across the industry; Microsoft, Apple, Google, Skype, Mozilla, Intel, IBM, Ericsson, Cisco, Opera and Logitech were among those represented.

1. Bernard Aboba, Microsoft
2. Harald Alvestrand, Google
3. Francois Audet, Skype
4. Richard Barnes, BBN
5. Adam Barth, Google
6. Christopher Blizzard, Mozilla
7. Chris Cavigioli, Intel
8. Alissa Cooper, CDT
9. Jeremy Doig, Google
10. Lisa Dusseault, Linden Lab

11. Niklas Enbom, GIPS
12. Ian Fette, Google
13. Pat Galvin, IBM
14. Roar Hagen
15. Stefan Håkansson, Ericsson
16. Tom Harper, Logitech
17. Ian Hickson, Google
18. Magnus Hiie, Skype
19. Joe Hildebrand, Cisco/Webex
20. Markus Isomaki, Nokia
21. Philip Jägenstedt, Opera
22. Cullen Jennings, Cisco
23. Matthew Kaufman, Skype
24. Piotr Kessler, Ericsson
25. Bastiaan Kleijn, GIPS
26. Serge Lachapelle, Google
27. Philippe Le Hegaret, W3C
28. Rian Liebenberg, Google
29. Jan Linden, GIPS
30. Henrik Lundin, GIPS
31. John Luther, Google
32. Debargha Mukherjee, Google
33. Mark Nottingham, Yahoo!
34. Jon Peterson, Neustar
35. Hubert Przybysz, Ericsson
36. Eric Rescorla, Skype
37. Aron Rosenberg, Logitech
38. Jonathan Rosenberg, Skype
39. Umesh Shah, Intel
40. Jonas Sicking, Mozilla
41. David Singer, Apple
42. Tim Terriberry, Mozilla
43. Hannes Tschofenig, Nokia-Siemens
44. Justin Uberti, Google
45. Jean-Marc Valin, Octasic
46. Koen Vos, Skype
47. Richard Winterton, Intel

Workshop format and discussions

Before the workshop, most of the participants sent in short papers to stimulate the participants' thinking. These papers are all available at http://rtc-web.alvestrand.com/papers and were made available to participants prior to the meeting.
At the workshop itself, some participants were invited to introduce various topics; see http://rtc-web.alvestrand.com/slides for the slideware they used (if any), and the group then spent most of the time in a free-form discussion, followed by a short summary at the end.

Workshop conclusions

Scope of work

The scope of work is real time client-to-client communication between apps running in browsers - using audio and video, but not necessarily limited to those media formats. The media exchange may, but need not, be mediated via a server (servers usually imply a huge latency cost, from the time it takes to get there and back).
Interworking with non-browser devices is of interest, but only if the non-browser device supports a compatible protocol set - this will, at minimum, involve use of STUN for connection establishment (see "security" below).

The question of whether we're "integrating a softphone in the browser" (that is, putting signalling functionality in the browser) or "providing P2P sockets" (media only, leaving signalling to Javascript and backends) is still not completely settled - it's reasonably clear that if the effort is to be successful, the hosting website + the browser functionality have to be able to function as a softphone or a videoconferencing unit together, but this does not dictate the division of labor between the website, the browser and the active content that the browser executes.
If we want to implement, for instance, a softphone-in-a-browser that interworks with SIP phones, we still need the interdomain interworking protocols we all know, including public endpoint identifiers and all that. This places certain constraints on the semantics of the functionality that is in the browser - it has to support at least functionality enough to work in this scenario, whether the signalling functionality is built into the browser, executed in Javascript or performed by the backend Web server.

Security and privacy

The privacy issues involve, among other things, that people's devices be turned on with their consent, and not without it - but UI design experience strongly suggests that there are no foolproof ways to achieve this. The history of the "lock icon" shows some success, but also some clear limits to such an approach; the current "camera light is on when camera is on" has variable implementation (some of them subvertable), and is in fact sometimes overlooked by the user even when it works.
The nature of peer-to-peer communication is such that the same origin policy that is presently used for Javascript is simply not appropriate; clients have to communicate using multimedia to someone who does not, themselves, run a website.

On the security side, a recipient of calls should not fall victim to a multimedia slam attack (sending unrequested media); it was felt that the capabilities of ICE (Interactive Connectivity Establishment, RFC 5245) are probably an adequate mechanism for authentication and authorization of media connections.
The discussion highlighted the fact that part of the ICE exchange has to be in the browser for security reasons; someone just watching the Javascript go by (which may be another script running in the same browser) should not be able to fake the ICE exchange from another computer. Just keeping the transaction IDs in the browser may be sufficient, but this warrants further study.
SRTP is a natural fit for protecting media streams, given present integration with RTP, and the fact that it is actually deployed in places, but the key establishment properties of DTLS (perfect forward secrecy, among others) makes that a more attractive mechanism for use with non-media data.

Devices

At a minimum, it is necessary to enumerate the devices available for the video function (cameras, mikes), and allow the devices to be activated, subject to privacy issues mentioned above. Part of this is already present in drafts for the HTML5 <device> interface, but we need to study this carefully to make sure it fulfils the requirements we have (and those requirements need to be clearly enumerated).

Codecs

The workshop participants strongly supported the idea that there should be a “minimum implemented subset” of codecs avaialble in the browsers - everyone should be free to implement more codecs as needed, but there should be a well known baseline.
The VP8 video codec and the IETF Harmony audio codec (since renamed Opus) had a lot of support for being in that subset, but some worries were raised - both that both codecs are new and haven’t been out long enough for us to assume that all patent concerns are flushed, and that we may need to add G.711 to ease intercommunication with non-Web-browser endpoints - but one participant stated that it will have trouble in coping with congestion when G.711 is attempted over TCP, since it is a very inflexible codec.

Notifications

It was clear to all that notifications need attention and work. Unless we can alert an user to an incoming call even when the browser windows are closed, the functionality of the browser-embedded application is strictly weaker than a non-browser “installed app”.
This means, at a minimum, that an audible and visible alert needs to be raised - exactly what form this alert will take, how it can be triggered, what the security requirements for it are, and how it relates to alert mechanisms already present in the OS, are very much unclear at this point.

Audio functionality

There are more audio functions than just codecs required for an adequate audio experience. These include automatic gain control, mute functions and echo cancellation. It’s not necessarily required that these be fully standardized, but it seems good if we can place some minimum requirements on such functionality (“if a sound comes in, it shouldn’t result in a louder sound

coming back out"). Unlike the audio codec area, there isn't that much open source code available for these functions, but that situation may change if companies with implementations make decisions to change it.

Other pieces

Bandwidth estimation was mentioned. Some codecs have support for doing this in-band, some implementations use bandwidth estimation based on RTCP, some have proprietary schemes. Again, there might be a need to separate the requirement that such a function be present and have some defined characteristics ("don't crowd out TCP"), but the actual control scheme might be allowed to vary between implementations. For further study.

Further work

The next steps involve surfacing this effort officially with the IETF and the W3C, which seem like the main standards organizations related to this space, defining the document sets that we need for a full specification, and starting to field candidates - both in the form of drafts, and in the form of working implementations that people can experiment with.

Specific work items include (not in sequential order):

- Defining a common data model for the "characteristics" of a session that covers both the browser/server interface and the interdomain interface - IETF activity
- Defining interfaces to ICE-encapsulated streams (which may be UDP, RTP, TCP) - IETF activity.
- Enumerate requirements for telephony interaction - IETF activity
- Define what's needed to define on bandwidth estimation - IETF DISPATCH WG activity
- Define privacy indication in browsers - W3C activity
- Support UI primitives required for incoming calls - W3C activity
- Support device discovery and capability exploration - W3C activity
- Extensions as needed to <video> and <audio> tags - W3C activity
- Decide on the interface styles - protocol stacks, socket interfaces
    - Joint W3C/IETF responsibility
    - Trial balloon implementations would really help here
- Decide on Mandatory To Implement codecs - discuss on list, not clear where to go
- Produce a workshop report (this document)