Smuxi - Feature # 943: WebRTC support

Status:	New	**Priority:**	Low
Author:	Andrés G. Aragoneses	**Category:**	
Created:	01/18/2014	**Assigned to:**	
Updated:	01/19/2014	**Due date:**	
Complexity:			
Found in Version:			
Subject:	WebRTC support		

Description:

I've been thinking a lot lately about WebRTC.

It's a good thing because it's P2P-based, it let's us go one step farther in avoiding dependance over servers.

However, WebRTC normally still requires three elements to work properly, 100% of the time:
a) A STUN server.
b) A signaling server, also called TURN.
c) A browser.

More info: http://blog.vline.com/post/63765098884/webrtc-if-its-p2p-why-do-i-need-a-server

Nowadays, (a) is not really a problem. There are already lots of free STUN servers out there. It's a service that is easy to provide because it's cheap, as it's just a reply to a message, no more bandwidth involved.

The problems are (b) and (c).

(b) is basically a server which acts as a proxy. The bad thing about depending on this is not middle-man surveillance or anything (as that can easily be solved by using encryption), but about the fact that server-bandwidth costs money. However, there's a possible solution to this problem: http://blog.printf.net/articles/2013/05/17/webrtc-without-a-signaling-server/ (serverless WebRTC: https://github.com/cjb/serverless-webrtc)

The catch about Chris Ball's solution above is twofold:
1) You still need a browser.
2) You still need to have some pre-communication between the parties (he says this can be done "via IM").

IMHO, it's kind of bullshit that to take advantage of the awesomeness of WebRTC, you need a browser. WebRTC is mainly about bi-directional communication! Its main target should be actually be IM apps! The only "web" related bit about WebRTC should be the fact that it should always be transported via HTTP, to avoid arrogant firewall policies. But nothing impedes us from implementing WebRTC in smuxi, which would make (c) and (1) issues go away.

Then, (2) problem explained above becomes a chicken&egg problem because smuxi is an IM app ;)

However, I think the pre-communication that should be done here is simply e-mail.

Let's assume both parties have smuxi installed in their computers. X wants to send a file to Y, or start a chat. Then X starts smuxi and clicks a button that says "Invite to chat/filetransfer via email". When clicking that button, a dialog pops up that lets you insert an email address, and ComboBox where you can choose two options:
- [Relay directly via STMP] (and you would need to provide SMTP details)
- [Generate a mailto: link that your email client can understand]

The result could be an email with a smuxi-webrtc:// link, for example.

Then, when Y receives and opens the email, she would click it, and smuxi would understand what to do, in the same way that ChrisBall's demo works.

If this strategy is implemented, and smuxi's approach begins to be "liked" a lot, one could even turn this into an RFC, where smuxi-webrtc:// would become something more official.

And then, we would have a browserless-and-even-more-serverless WebRTC.

History

01/18/2014 06:00 PM - Andrés G. Aragoneses

Maybe the simplest and easiest proof-of-concept (or rather, minimum-viable-feature) here would be to forget email for now and implement a basic alternative to "DCC Send File", called "Smuxi Send File"*.

If this works, then the email-based negotiation explained in comment#0 would serve the purpose of starting a chat with a user that we don't have in any of our configured networks in Smuxi.

* (or rather, label it simply as "Send file", and smuxi under the hood would figure out if both parties have the necessary version of smuxi for WebRTC support)

01/19/2014 01:10 PM - Mirco Bauer

- Tracker changed from Bug to Feature

01/19/2014 01:27 PM - Mirco Bauer

Yes with p2p there is the always the problem of the "rendezvous point". WebRTC solves this with a website were both peers meet each other before they go p2p. For this reason I like the TorChat approach very much, as it is p2p and relies on hidden services for peer to find each other, see: https://www.meebey.net/research/torchat_protocol/ I plan to make TorChat a built-in feature of Smuxi and this could also be used to initiate WebRTC communication without a website. WebRTC itself I don't find interesting because the streams it supports are bound to video/audio only as far as I know because they are elements handled by the browser itself! I found this talk about palava.tv interesting (which is WebRTC based): http://media.ccc.de/browse/congress/2013/30C3_-_5562_-_en_-_saal_g_-_201312281245_-_lightning_talks_day_2_-_nickfarr.html @ 1:31 Hm actually after watching _that_ talk, he mentions an arbitrary stream... so Smuxi could use this but making it Smuxi specific. Take a look at what the TorChat protocol gives you, I think it is much better still (more "freedom" features).

01/19/2014 01:31 PM - Mirco Bauer

You haven't really given a use-case for WebRTC. was your point that people can talk to others (including Smuxi) via WebRTC? I still see the issue of content type, it is limited to audio and video, while I keep Smuxi text-only for now. (it needs to be good at text before entering new communication domains IMHO).

01/19/2014 01:42 PM - Andrés G. Aragoneses

Use case 1: sending files without a central server (a-la IRCish 'DCC Send').
Use case 2: chat without a central server/network.

I still honestly don't get you when you say that WebRTC has a limitation of only allowing audio/video, because:
a) As I explained, you can check out and test the serverlessWebRTC demo, which is just chat and file transfer: https://github.com/cjb/serverless-webrtc
b) In the talk you link, at 1:31hr, they precisely mention that you can transfer raw data, not just audio/video.

Disclaimer: I don't mind that what comes out of this ticket (if anything comes out) is smuxi-specific (at least at first). If other IM clients like the idea, we could later define a spec and try to interoperate.