30-03-2016

SA8T2 Internal Deliverable Technology Scout: Stream and record lectures with WebRTC

SA8T2 Internal Deliverable

Contractual Date:	30-04-2016
Actual Date:	30-03-2016
Grant Agreement No.:	691567
Activity:	12/SA8
Task Item:	Task 2 – WebRTC
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	NORDUnet (UNINETT)
Authors:	Simon Skrødal (UNINETT)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 691567 (GN4-1).

Abstract

This document reports on results and findings from a technical investigation into WebRTC's maturity for lecture recording, in the context of European higher education and research. The technology scout was conducted by the Service Activity 8 (SA8, Real Time Communication and Media), Task 2 (WebRTC) team as part of the GN4-1 project. This report should, as such, be read in context of the related work produced by GN4-1 SA8-T2.

Table of Contents

Table of Figures

Executive Summary

A host of well-established, as well as emerging, technologies are used to facilitate online teaching and learning. Solutions pertaining to web conferencing, lecture streaming and recording are changing the way education is perceived, delivered and consumed.

By removing the barriers of time and location, online technologies offer unprecedented flexibility to teaching, learning and research. Education is made accessible to a segment of our population that would otherwise find it difficult, or even impossible, to access their field of study. Lectures may be streamed from one campus to another, or indeed from one kitchen bench to the other, and recorded for later review at the learner's convenience.

Much attention is given to WebRTC and its potential for online education by means of live streaming of audio/video/data in a web-conferencing context. Less known is the standard's ambition to facilitate desktop capture (acquire and stream what happens on the screen) as well as the recording of all media content.

With this in mind, this technology scout set out to investigate WebRTCs maturity regarding all facets required to implement a zero-install, browser-based, *screencast* (web)application for streaming and recording combined feeds from microphone, web camera and screen — utilising only WebRTC and HTML5 standards.

Our findings suggest that browser implementations of WebRTC standards, specifically pertaining to recording and screen capture, are insufficient. Implementations are, however, progressing at a very rapid pace. We therefore recommend that this area is revisited in the near future, e.g. early 2017.

1 Introduction

1.1 About this document

This report documents an investigation into the technical feasibility of using WebRTC technology for lecture streaming and recording services in the context of European higher education and research. It also provides a comparison with how traditional services for lecture recording are constructed, and what implications a WebRTC-based equivalent may have.

The technology scout was undertaken as part of the Geant4 Phase 1 project by the WebRTC Task 2 (T2); one of three tasks of the Real Time Communication and Media activity (SA8). This report should, as such, be read in context of the related work produced by GN4-1 SA8-T2.

The WebRTC task ran from 1 May 2015 to 30 April 2016.

1.1.1 Target audience

This document targets technical management and specialists, in particular those working in the fields of real time communications, eLearning and eResearch.

1.1.2 About the author(s)

Simon Skrødal is very passionate about ICTs and their value-adding potential for teaching and learning. Following degrees in computer science, Simon was awarded the University Doctoral Research Medal for his PhD research and development of a computer simulation for pre-service teacher training.

Simon has worked as a Senior Engineer/Advisor for the national UNINETT eCampus services deployment programme since 2011, where he is heavily involved in establishing infrastructures and services for teaching and learning in the Norwegian Higher Education sector.

1.1.3 Important reading note on WebRTC technology maturity

Research and writing of this document took place in February/March 2016 and provides an accurate account of the WebRTC technology status from this period in time. However, keep in mind that WebRTC, particularly in regard to browser support, is in a continuous state of flux.

1.2 Background

1.2.1 WebRTC

WebRTC (Web Real-Time Communication) facilitates a new generation of communication applications that, in many cases, promises considerable benefit to research and education. WebRTC gives web browsers access to the host user's microphone and webcam without plugins/extensions and provides functionality to efficiently and securely stream these feeds (and other data) over the internet in real-time. The WebRTC technology roadmap includes features for screen capture as well as a facility to record and store content.

Benefits of WebRTC include simplified (i.e. better) user experience (no additional installs/dependencies), cross-platform support (one solution fits all), open and standards-based technology (that is simple to use), security (peer-to-peer encrypted data) and performance (reduced latency).

WebRTC is platform independent and, at its core, requires no dedicated software or extensions (other than a web browser). The standard lowers the threshold to develop applications able to transport synchronised multimedia and data over the internet in real-time.

1.2.2 Screencast

A screencast is a multimedia composition typically comprised of a feed from the computer screen, accompanied by an audio narration. The screencast may also include a camera feed ('talking head') and system audio (see Figure 1: Screencast layout sample).

Screencasts are particularly suitable for presenting content that requires excellent representation of information otherwise difficult to capture with a video camera. Hence, it is a popular solution for capturing lectures centred around digital content, such as delivered by overhead projectors (e.g. slide-based) and interactive whiteboards. Many educators (and indeed students) also make use of screencasts to create educational content outside of the classroom (e.g. step-by-step tutorials or student assignments).

Traditional screencast solutions run as native applications, developed for dedicated operating systems (OS). A WebRTC-based alternative, on the other hand, is OS-agnostic and may run in modern web browsers on any system.

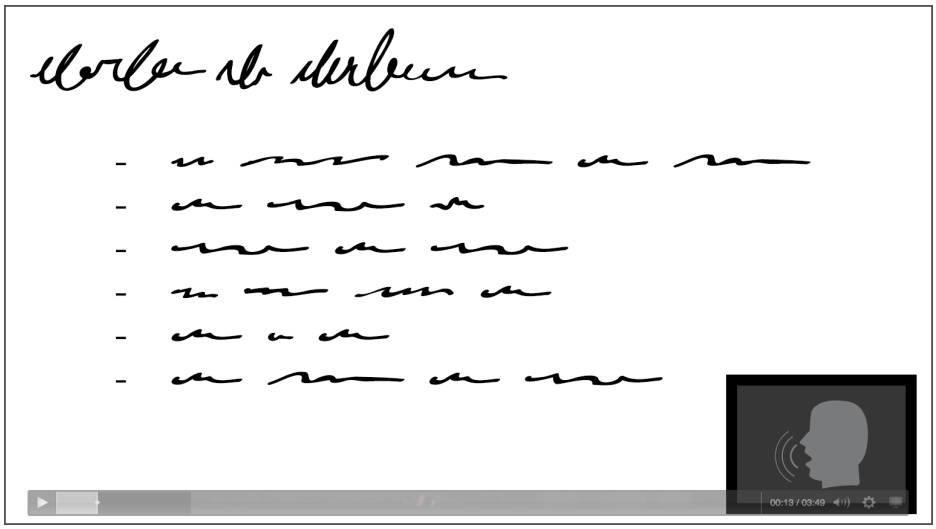

Figure 1: Screencast layout sample

1.3 Technology scout objective

Although WebRTC enjoys advanced support in many major web browsers, the standard is still a work in progress. Further, many components or extensions of the standard are in their infancy.

This technology scout set out to investigate WebRTCs current state of maturity to support a browser-based screencast application to achieve the following:

- Acquire the following media sources from the host system:
  - Screen
  - Video (e.g. web camera)
  - Microphone
- Record these sources to a flat video file
- Support live stream broadcast (with recording) as well as 'off line' (e.g. in office) production
- No web browser extensions/plugins required!

The very name "WebRTC", *Web Real-Time Communications*, implies at least two endpoints performing synchronous communication. In the scope of this technology scout, we also want to gauge WebRTC/HTML5 standards to allow for the production of screencasts without necessarily broadcasting the content live others.

An absolute requirement of the solution is that it will function without browser extensions/plugins of any kind. Dependencies on such extensions take away the desired "nothing to install" benefit as well as interoperability across browsers (extensions are not browser-agnostic).

1.4 Rationale

Keywords associated with WebRTC include "open", "free", "standard", "accessible", and "platform-independent". It is supported and given tremendous momentum by superpowers of the web (e.g. Google, Mozilla, Microsoft and Opera), with underlying protocols being developed jointly at the W3C and IETF. WebRTC is thus an emerging technology that will present to the R&E community a new set of standards and applications for real-time communication and collaboration.

Lecture streaming and recording is a growing market that does not seem to coherently follow any standards; not with capture agents/clients, not with metadata, not with content formats (codecs), not with delivery and not with playback. Screencasts provide an interesting case because, if following certain base-standards, not only can they be made as simple or complex as desired, but they may also evolve over time without leaving "legacy" content behind.

Screencasts utilising WebRTC technology can provide standards-based live streaming (and interaction) out of the box, as well as facilitate "off-line" recordings (e.g. sans the live stream). Audience (students) may participate live or consume recorded content on any platform/OS without any extra (potentially licensed) software/extension installations.

Web-based apps need no rollout, are easier to update, simplify the user experience (require no action from the end user) and are platform agnostic. The underlying technology makes it easier to extend/alter functionalities, and the service can be moulded to suit the particular workflows/policies/requirements of any organisation.

2 Technology Overview

A screencast application requires access to the following media sources on the host system, as well as the capacity to stream and record these:

- Microphone
- Video camera
- Screen (desktop)

The following WebRTC specifications are pertinent to achieve the above: `Media Capture and Streams`, `Screen Capture` and `MediaStream Recording`. This section takes a closer look at each of these specifications.

2.1 Media Capture and Streams

- Editor's Draft: http://w3c.github.io/mediacapture-main/
- Demos and code: https://webrtc.github.io/samples/

With the exception of Apple's Safari, all modern web browsers (e.g. Google Chrome, Mozilla Firefox, Microsoft Edge and Opera) implement W3Cs proposals in the `Media Capture and Streams` specification. This specification defines a set of JavaScript APIs, specifically the `getUserMedia` API, that can request access to local multimedia devices, such as microphones and video cameras.

Browser implementations of these APIs are advanced and well documented, with numerous open-source demonstrations available for app developers.

2.2 Screen Capture

- Editor's Draft: http://w3c.github.io/mediacapture-screen-share/
- Relevant article: https://webrtcstandards.info/webrtc-screen-sharing-discussion-in-w3c/
- Chrome Extension: https://developer.chrome.com/extensions/desktopCapture

Access to the screen is provided by an extension to the `Media Capture and Streams` specification, `Screen Capture`, that enables the acquisition of content from a user's display. The draft refers to the different types of screen-based content as "display surfaces" and groups these into the following four categories:

1. Monitor — a display surface that represents a physical display. Some systems have multiple monitors, which can be identified separately. Multiple monitors might also be aggregated into a single logical monitor.

2. Window — a single contiguous surface that is used by a single application.

3. Application — might have several windows available to it, and those can be aggregated into a single application surface, representing all the windows available to that application.

4. Browser — the rendered form of a single document (i.e. a single tab). Not strictly limited to HTML documents.

The proposed specification defines that the capture of displayed media be enabled through the `getDisplayMedia` API, which to the application developer will closely resemble the aforementioned `getUserMedia` API used for accessing audio and video. The draft can be traced back to 2014, but have undergone a number of revisions and changes since.

While the Screen Capture specification promises a complete set of functionalities required by our proposed service, it is presently **not** implemented by any browser. There is also very little information to be found from browser vendors around this particular specification.

Some experimental implementations of screen capture have, at some point or other, been available in some browsers, but these have long since been revoked due to security concerns:

> *...the security implications of this functionality are much harder for users to intuitively analyze than for camera and microphone access.*
>
> — Security Considerations for WebRTC (http://tools.ietf.org/html/draft-ietf-rtcweb-security-07#section-4.1.1)

Indeed, the `Screen Capture` specification itself devotes a whole chapter to debate the potential risks associated with users (inadvertently) sharing their screen content:

> *The immediate and obvious risk is that users inadvertently share content that they did not wish to share, or might not have realized would be shared.*
>
> — w3c.github.io/mediacapture-screen-share/#security-and-permissions

Little is known about when browser vendors will begin to offer an implementation of the `Screen Capture` specification. Presently, the only way to access the screen is by use of purpose-built browser extensions (e.g. for Chrome and Firefox) or to manually configure browser flags to enable screen sharing and add the hosting domain to a whitelist (Firefox only). This fact puts serious limitations to the proposed service, since such extensions must be built and maintained for each individual browser (and thus kills the ideal of a no-install, no-fuss and easy to use service for the end user). Development of browser extensions also require that all browser vendors provide a supporting API to allow access to a screen capture service (something that is lacking today).

2.3 MediaStream Recording

- Editor's Draft: http://w3c.github.io/mediacapture-record/MediaRecorder.html
- Demo and code: https://webrtc.github.io/samples/src/content/getusermedia/record/

An extension to the `Media Capture and Streams` specification that *is* enjoying browser implementation (Chrome, Firefox and Opera — Edge is 'under consideration') is the W3C `MediaStream Recording` specification.

The implementations are, however, very fresh (e.g. Chrome only added support for audio recording in version 49, Mar. 2016) and buggy (e.g. duration limits, frame drops/buffer issues, mime type errors, periodic blinking, …). There are also a number of limitations still pertaining to control/tuning of recording parameters and codecs (e.g. sample rate, bitrate, only webm, …).

Shortcomings and issues aside, the standard implementation is progressing at a rapid pace and is already suitable for preliminary development.

3 Implications of WebRTC in lecture streaming and recording

We are witnessing a new era of teaching and learning unfold, as a growing number of educational providers are making online classrooms and lecture streaming/recording mainstream (perhaps best illustrated by the rise of Massive Open Online Courses (MOOC), Open Universities and the like).

Lecture streaming and recording is still an emerging market, with commercial vendors and open source projects still working hard to make their systems work well with the many facets demanded from teaching and learning. From a production perspective, systems range from the fully automated that run on expensive proprietary hardware and demand little from the lecturer (but more from the administrator), to the more affordable clients that are run and operated by the lecturer on standard computer/mobile equipment. Any solution will, however, need to facilitate content production, publication and consumption to some extent.

Typical production considerations include:

- platforms, hardware, automation, input (media), processing (conversions), metadata, access, updates/rollouts and storage (and management).

Typical consumption (and publication) considerations include:

- formats (codecs, plugins), access, platform support, seek & search (metadata), playback and user experience.

3.1 Two different views for storing lecture recordings

How, where and for how long recorded material should be stored is often subject to financial, philosophical and practical arguments. The "pragmatic" view is to treat recorded lectures as "fresh goods" that may be thrown away at the end of a semester/year. Another, and perhaps more "conservative", view considers the many potential ways a recorded lecture may be of value in the future, thus demanding that nothing should ever be discarded for this reason. These opposing views demand different approaches and requirements from a capturing solution.

The pragmatic view has many luxuries not afforded by the conservative. While the former typically enjoys lower cost of entry/storage/delivery and may avoid vendor lock-ins (both in terms of contracts and proprietary content formats), the latter must put serious consideration into long-term

conservation of digital content. Even today, many systems still rely on browser plugins (e.g. Flash or Silverlight, and thus dedicated apps for mobile devices) for video playback and to support extra features (e.g. inbuilt quizzing, bookmarks, subtitles, etc.). A lecture recording composition may be incredible complex, composed by one or more videos (and separate audio tracks), hundreds of pictures (e.g. for thumbnails or high-res screen grabs), subtitles with time codes and supporting files to keep audio/video/thumbnails/quiz/other in sync. The conservation of such content, and its interactive nature, becomes a challenge as technologies are being replaced and made redundant.

Nonetheless, it is not possible to deem one view better or more suitable than the other without knowing more about the context.

3.2 WebRTC implications for these two different views

A WebRTC-based solution need not make any assumptions about pragmatism or conservatism at all. It could mainly concern itself with the production aspect, and hardly need to consider the management/publication-side. The reason being that a well put together, open and standards-based web application for screencast production may be extended to suit any dogmas concerning content management and distribution. Extra web-based modules could provide upload/publishing features, e.g. to social video platforms, LMS, cloud storage (e.g. Box, OneDrive, Google Drive), or indeed an existing lecture capture system that supports import from external sources. Cloud storage integration would certainly add value to NRENs that offer these services to their communities. Likewise, modules for basic editing features (e.g. trimming of start/endpoints) could also be added to the solution if needed.

By decoupling production from media management, strategies are no longer dictated by the regulations enforced by proprietary (and indeed most open) solutions. That said, we are not blind to the fact that large-scale fully automated systems offer features that eclipse the scope of this scout. What we are proposing, however, is an end-user operated web-app limited to broadcasting/recording (with any decoupled publishing extensions) of screencasts.

3.3 Considerations

Some NREN communities offer cloud storage solutions for its users. Many of these offer APIs for extending functionality and integrations. A web-based screencast application need not provide a media management system or its own publication system, but rather, via APIs, publish to such cloud storage solutions, or, indeed, social or educational video platforms (e.g. YouTube/Vimeo/iTunes U), local storage, network storage, LMS, CMS, MOOCs, other capture systems, and so on.

Federated identity and single sign-on (SSO) are central in the R&E community, but is at best difficult achieve with traditional desktop software. A web-app for screencast production, however, can implement SSO with ease to provide authentication, authorisation and access control. Benefits of account management are obvious, but in an increasingly connected world SSO also presents a number of opportunities for integrations across existing and future services.

Most commercial and open source systems produce content compositions (e.g. mix of audio, video, screen, queue points, bookmarks, thumbnails, metadata, etc.) that may only be represented in

dedicated, sometimes proprietary, players (developed in e.g. Flash/Silverlight). To maintain and manage legacy content from these systems can present a challenge as the technology progresses. Many institutions may therefore feel “trapped”, with no other choice than to continue using outdated solutions in order to protect prior investments and legacy content. WebRTC, on the other hand, is by definition guided by open standards that support future proofing through transfer and transformation of legacy content.

4 Lessons Learned

This technology scout somewhat overlaps a similar research at UNINETT by the author in early 2015. Looking at my notes from one year ago, the WebRTC standards and browser implementations have undoubtedly made progress, but not sufficiently so to realise the proposed service.

Online documentation (even from browser vendors), articles and demonstrations are quickly outdated and becomes not only irrelevant, but in some cases misleading (e.g. demo code, although fully functional, may be using deprecated APIs). Detailed information about the latest standards implementations is often difficult to find or poorly documented. Due to the freshness of some APIs (e.g. MediaStream Recording), there are few implementations and use, making the technology less stable and prone to bugs.

It is important to be aware that standards, APIs and implementations (demos and browser support) may well have changed drastically overnight. To illustrate; in late 2015, with the release of Google Chrome 47, the web browser started to actively block WebRTCs `getUserMedia` (as well as other technologies, such as geolocation and screen sharing) not served from a secure (HTTPs) site or localhost by default. This change has caused a number of WebRTC-related demos and services to stop working for Chrome-users with an up-to-date browser (http://www.tokbox.com/blog/the-impact-of-googles-new-chrome-security-policy-on-webrtc/), and observed frequently in working on this tech scout.

A web browser's support of any WebRTC component may only be partial and behave differently to other web browsers. E.g. one version ago, Chrome's implementation of the MediaStream Recording API only supported video recording (audio was left out).

5 Conclusions

A WebRTC/HTML5-based screencast service would offer a number of advantages to the organisation and its users. Attention to open standards, which is central to WebRTCs specifications, facilitates ease of implementation, great opportunities for integration, future proofing, user-friendliness, broad platform support and so on.

The WebRTC APIs required to realise a screencast application as per our definition are, however, not yet sufficiently implemented in any web browser. The technology scout has nevertheless established that all required building blocks are available, at the very least as specification drafts.

1. **Access to microphone and camera feeds**

   Audio and video capture provided by the `Media Capture and Streams` specification, which defines the `MediaStream` API, enjoys broad and stable implementation in many modern web browsers. It is an essential interface to enable audio and video conferences.

2. **Recording of streams**

   The `MediaStream Recording` specification, which defines the `MediaRecorder` API, permits recording of audio/video and is already implemented and functional in some web browsers. These implementations are very recent, though, with Chrome first supporting the API in version 49 (released in March 2016). Demonstration tests confirm that recording may now be achieved, though lacking in codec (variety) support and the implementations are buggy/inconsistent between browsers.

3. **Access to screen (desktop) feed**

   The real showstopper is screen capture. The W3C is working on extension to the `MediaStream` API (getUserMedia), the `Screen Capture` API (`getDisplayMedia`), that "enables the acquisition of a user's display, or part thereof, in the form of a video stream". The API is, however, not yet fully implemented in any browser. In order to achieve screen capture, a browser-specific add-on/extension (or a manual override of browser configurations in Firefox) must therefore be developed which the end users would have to install in order to use the service.

Early browser previews (alpha/beta versions) and rapid release cycles suggest, however, that a WebRTC-based screencast service may be realised, perhaps in the near future. The main source of uncertainty at present is the `Screen Capture` API, which represents a core requirement for a screencast application. No browser vendors make any mention of implementing this proposal as of yet.