FLEXIBLE VIRTUALIZATION OF RADIO PRODUCTION AND PLAYOUT.

J.E.Kivelä

Jutel Oy, Finland

ABSTRACT

A virtual browser-based radio production and playout system in a cloud environment has been built. The system audio processing is done in cloud so that no specific hardware for studio and playout infrastructure is needed.

The system is built in the AWS cloud environment and it is based on HTML 5, Rest APIs, ActiveMQ messaging and WebRTC audio streaming. The system data model is compatible with existing enterprise-level radio production and playout system.

Proof of concept tests have been done with the cloud-based radio production system. The system combines all the functionality of radio studio operations, audio processing, playout control, program production and playout streaming.

INTRODUCTION

Radio needs to be close to the public and on the spot of action. The need to run radio broadcast operations from local or temporary studios on shopping malls, sports centres, schools etc. is increasing. This means that the transition between a studio-controlled playout session and a remote broadcast must be as easy as possible.

A large national broadcaster typically runs multiple national distribution channels and lots of regional operations. The number of workstations can be up to several thousands. In addition to broadcast planning and production applications the workstations have many other applications running. Simultaneously, system maintenance and updates have become more and more complicated.

Up till now large radio broadcast IT systems have been built with multi-tier client-server configuration. Client-Server architecture has been efficient for large fixed installations but it has some drawbacks that can be overcome with new possibilities.

The advent of networked IP-audio has removed the need to place the audio playout close to the audio infrastructure. Web-based architectures move the business logic and interfaces to back-end servers and enable the use of browser-based workstations. Virtualisation and cloud technologies make dedicated servers almost obsolete.

GOALS FOR THE PROJECT

The goal of the project has been to test and build a virtual browser-based radio production and playout system in a cloud environment without any dedicated hardware-based audio studio and playout infrastructure.

Modern cloud and web-based architectures have been used to build up a new generation broadcast production and playout platform that supports workflows from a simple remote broadcast to complex studio-operated mixed programs. Table 1 shows the different targets of the project.

Topic	Goal
Temporary Studio installation	No extra audio hardware needed except the microphone inputs. Just a workstation with a browser.
Regional Studio	Minimal or no extra audio hardware needed. Just a workstation with a browser and decent acoustics
Remote broadcasts	Monitor and control broadcast with a tablet. Audio feed from the tablet or via mobile codec.
Continuity Studio	Integration with browser-based touch-screen virtual studio and IP audio routing.
Office workstations	Browser based. No client software to be installed.
Playout Server	Playout servers to be freely located on fixed devices, virtual machines or in the cloud. Audio output to be either audio cards, virtual IP-audio or cloud-based streaming.
Audio Management	Either interface to the virtual IP studio (VSM) or internal audio engine with mix, duck and sum functionality.
Data Model	Flexible Data Model with backwards compatibility to RadioMan Enterprise system.

Table 1. Project sub-goals

Long term goal of the project is to be able to implement local studios in a way that all that is needed is just an Internet connection and room facilities with decent acoustics. Some local microphone management may be needed in case of multiple persons in the studio. The basic configuration of the system is shown in Figure 1.

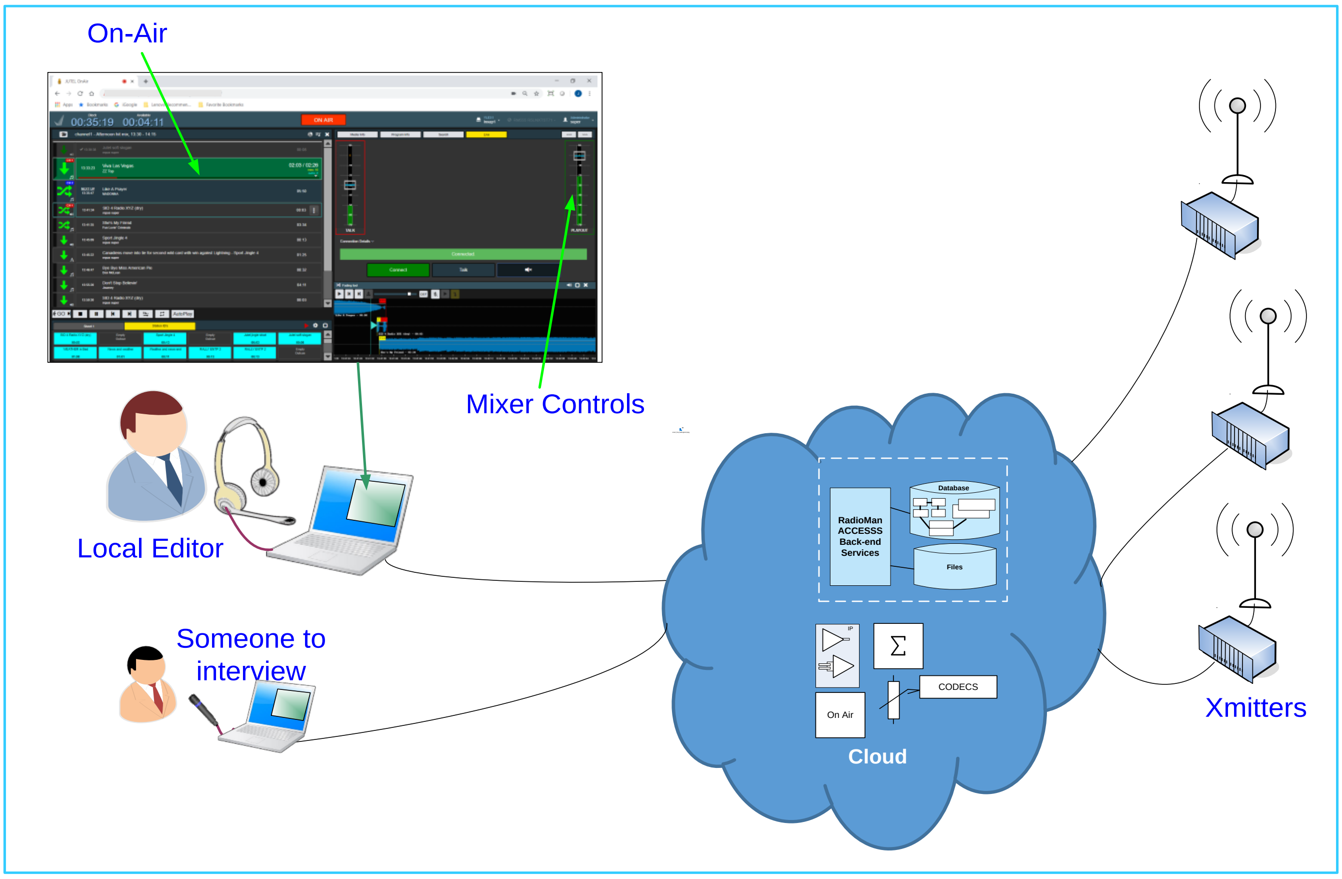

Figure 1 – System Overview

ARCHITECTURE

The basic system architecture utilises HTML5 `Faulkner et al (1)´ browser based front end that supports playout, planning and production tasks. The architecture is shown in Figure 2.

The internal communication is based on HTML5 REST APIs, ActiveMQ messaging and WebRTC `W3C (2)´ audio streaming. All the user interfaces are utilising HTML5 so that the system should be as agnostic of the browser as possible. ActiveMQ messaging is used for fast control purposes so that the control messages do not need to go thru the back-end processes in playout situations.

The audio streaming between the workstation and the back-end processes were implemented with WebRTC and Opus audio coding. First some software-based standard broadcast codecs were tested but with WebRTC the system does not need any installed add-ons or any external software modules. Originally the program output was taken out via an Icecast streaming server but during the test phase also WebRTC streaming and local studio player options were tested.

The back-end services were installed in AWS cloud in a clustered configuration with a virtual load balancer between the browser front-ends and the back-end processes. The playout processes in the AWS cloud were extended with controllable audio processing processes managing level controls, mixing, auto-ducking and audio streams.

The workstations were equipped with USB-connected microphones and loudspeakers/headphones. The HTML5 On-Air control screen was equipped with simple microphone and level controls and the feedback from the playout mix to the workstations supports N-1 configuration.

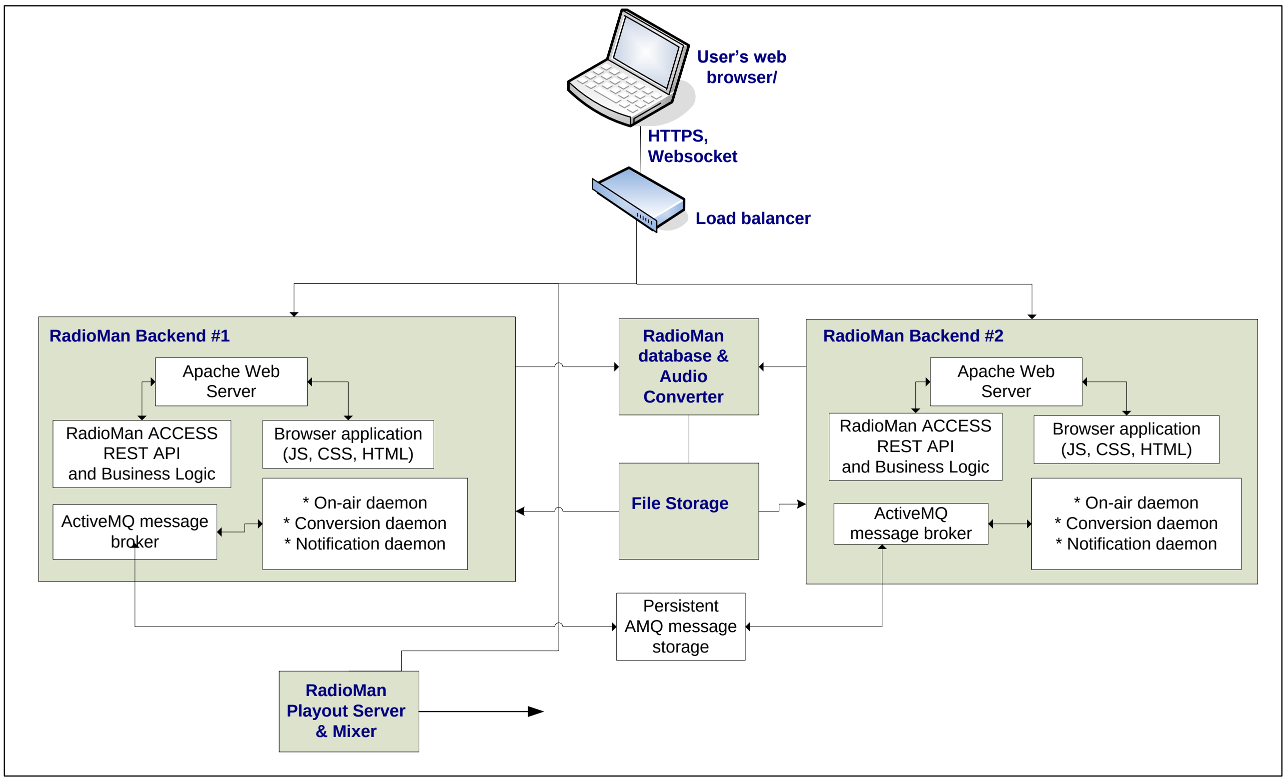

Figure 2 – System Architecture

The system architecture allows the back-end processes to be installed either on a dedicated server, on virtual machines or in the cloud environment. The playout servers can be configured either as stand-alone units, on virtual servers with IP-audio out (AES67, Livewire etc) or they can be implemented in the cloud as in the test installation.

The playout user interfaces can be also placed inside a Virtual Studio Manager as HTML5 components so that the VSM environment and the OnAir control combine an integrated touch-screen environment for larger studio environments.

RESULTS

System tests were done with the cloud-based radio production system that combines all the functionality of radio studio operations, audio processing, playout control, program production and playout streaming. The system was installed in AWS Cloud in Ireland and the tests run in Finland in Helsinki and Oulu. Another test setup was installed as an inhouse system in Finland on major broadcaster facilities.

The cloud-based radio system performance was tested with Opus audio coding in various bitrates. The round-trip ping was 58ms. The average overall delay from microphone to the listener was 330ms varying from 280ms to 450ms. The Opus bitrates in the tests were from 48kbits/s to 256kbits/s. This is shown in figure 3. Figure 3 shows approximated delays as it is difficult to define delays at exact points.

The playout start delay tested is depended on the network delay and is very small. The start delay from reporter's play-command to the sound at the listener was tested to be about 160ms.

The system was also tested with two separate contributors on different locations. On the test system the audio chatting was routed via the server environment and thus the overall delay is more than 600ms which is not practical. In the next phase chatting will be arranged in a peer to peer configuration making DJ and guest operation practical.

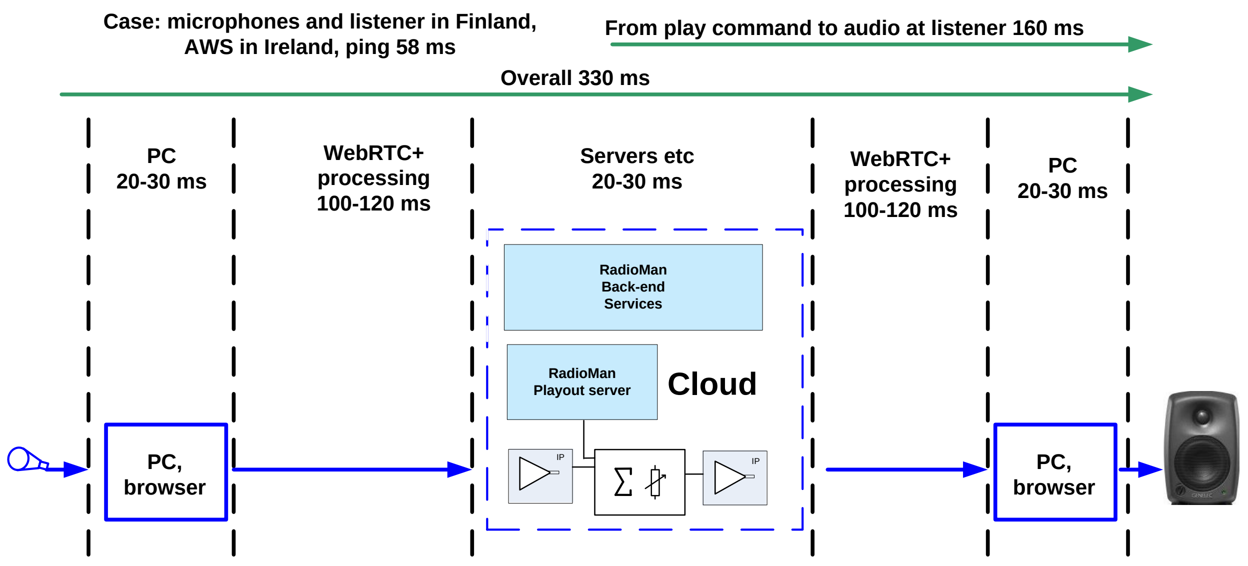
Case: microphones and listener in Finland, AWS in Ireland, ping 58 ms
From play command to audio at listener 160 ms
Overall 330 ms
PC 20-30 ms
WebRTC+ processing 100-120 ms
Servers etc 20-30 ms
WebRTC+ processing 100-120 ms
PC 20-30 ms
RadioMan Back-end Services
RadioMan Playout server
Cloud
IP
IP
PC, browser
PC, browser

Figure 3 – Delays

Part of the tests were done with a major national broadcaster. During the test phase the audio control was introduced with an auto-ducking feature so that the playout level was automatically controlled by the microphone control level on the control screen.

The system was also tested with broadcaster in-house configuration on a virtual server environment. In this configuration the delays encountered are much smaller than with the cloud.

The system has been tested successfully with Chrome and Firefox browsers both in PC and Android tablet / phone environments. Safari browser support is under tests.

CONCLUSION

The results of the tests showed that the concept is feasible for live broadcast production purposes. The On-Air playout control and audio monitor delay was found to be small enough for semi assisted radio with music, jingles and talk.

The system was also tested in mobile radio reporting case with local PA-sound reporting taken from the cloud feedback. This configuration showed clearly that the PA-sound must be taken from a local mix with minimal delay.

The system was tested as a proof of concept but the results have been so promising that the system is now available as a service in the cloud. A process is also under way to install the system on fixed regional services. Further enhancements to be introduced are peer-to-peer chatting for multiple operators and integrated connectivity with mobile contribution devices.

References

1. Faulkner, Steve; Eicholz, Arron; Leithead, Travis; Danilo, Alex; Moon, Sangwhan; Doyle Navara, Erika; O'Connor, Theresa; Berjon, Robin, eds. (14 December 2017) [2016]. "HTML 5.2 W3C Recommendation". Revised version. World Wide Web Consortium (W3C). Retrieved 26 July 2018.

2. *"WebRTC 1.0: Real-time Communication Between Browsers". World Wide Web Consortium. 27 September 2018. Retrieved 25 March 2019*