S. Stickland, R. Athauda, and N. Scott, "Design and Evaluation of a Scalable Real-Time Online Digital Audio Workstation Collaboration Framework"
J. Audio Eng. Soc., vol. 69, no. 6, pp. 410–431, (2021 June).
DOI: https://doi.org/10.17743/jaes.2021.0016

Design and Evaluation of a Scalable Real-Time Online Digital Audio Workstation Collaboration Framework

SCOTT STICKLAND, *AES Student Member*, **RUKSHAN ATHAUDA, AND NATHAN SCOTT,** *AES member*
(scott.stickland@uon.edu.au) (rukshan.athauda@newcastle.edu.au) (nathan.scott@newcastle.edu.au)

The University of Newcastle, Australia

Existing designs for collaborative online audio mixing and production, within a Digital Audio Workstation (DAW) context, require a balance between synchronous collaboration, scalability, and audio resolution. Synchronous multiparty collaboration models typically utilize compressed audio streams. Alternatively those that stream high-resolution audio do not scale to multiple collaborators or experience issues owing to network limitations. Asynchronous platforms allow collaboration using copies of DAW projects and high-resolution audio files. However they require participants to contribute in isolation and have their work auditioned using asynchronous communication, which is not ideal for collaboration. This paper presents an innovative online DAW collaboration framework for audio mixing that addresses these limitations. The framework allows collaborators to synchronously communicate while contributing to the control of a shared DAW project. Collaborators perform remote audio mixing with access to high-resolution audio and receive real-time updates of remote collaborators' actions. Participants share project and audio files before a collaboration session; however the framework transmits control data of remote mixing actions during the session. Implementation and evaluation have demonstrated the scalability of up to 30 collaborators on residential Internet bandwidth. The framework delivers an authentic studio mixing experience where high-resolution audio projects are democratically auditioned and synchronously mixed by remotely located collaborators.

0 INTRODUCTION

Professional audio mixing of multitrack music recordings is intrinsically collaborative and reactive. It demands effective, timely communication between all collaborators to audition audio material and processing techniques, typically via a Digital Audio Workstation (DAW) software application. The ideal audio mixing environment is a studio where engineers and clients can mix and monitor high-resolution audio assets. Recently many efforts have explored remote and online modes of operation to facilitate such shared activities using the Internet. These platforms have made significant progress in developing online collaborative modes of operation, including remote recording, creative production, and remotely supported music composition. We believe there is potential for further improvements to collaboration models to facilitate a more authentic and inclusive collaborative experience, especially for audio mixing in professional and educational contexts.

We present two hypothetical use-case scenarios to demonstrate the need for an enhanced, scalable, and more realistic online DAW collaboration environment.

Scenario 1: An international touring front-of-house sound engineer renders a live multitrack recording of an American-based band. At the end of the tour, the sound engineer returns to Australia, their home country. The touring band, impressed with the live mixing, wants the engineer to work with them and their local studio engineer to mix a live album. The sound engineer accepts the work but is unable to return to the United States in time. Instead of the engineer creating mixes for delayed offline review, the team completes the work by setting up a real-time, online, collaborative session using a DAW application. All of the geographically isolated collaborators join the collaboration session and interactively and simultaneously audition, debate, and discuss the project and contribute to the process by collectively mixing and monitoring the high-resolution, uncompressed DAW

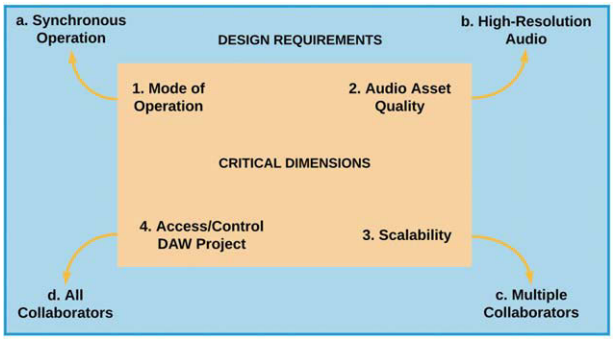

Fig. 1. The four dimensions for designing an online collaborative DAW environment and the requirements for collaborative audio mixing and mastering.

project. Mixing the album in this democratized environment allows all of the stakeholders to provide in-the-moment, practical, and immediately enacted contributions and arrive at a final mix in a more synergetic, timely, and mutually agreeable manner.

Scenario 2: A college of music offers an online course in advanced audio mixing. The online tutor, who has a class of 25 remote students, has shared access to high-resolution audio files and a DAW project via a cloud storage folder. The tutor and students, who have already downloaded the DAW project and high-resolution audio files to their local computer, open a local instance of the DAW project with remote control facilitated by an online DAW collaboration application.

All of the class communicates via a videoconferencing feature, and the tutor and students connect their local DAW to the collaboration application, linking all of the remote instantiations. Initially the tutor demonstrates the use of a compressor plug-in, which the students observe in their local DAW. Any adjustments to the compressor plug-in are executed on the local DAW instantiation and then mirrored on all other DAWs within the collaboration session. Wishing to gauge the students' level of understanding and proficiency, the tutor individually invites students to apply compression to another of the audio tracks in the DAW project. Everyone else observes the changes with live comments managed via videoconferencing or a text-based chat.

In the above scenarios, each real-time collaborator can:

- View and control a local DAW instantiation of the collaborative audio mixing project;
- Interact and communicate synchronously with all other stakeholders (via videoconferencing and online chat); and
- Locally monitor the mixing project's progress (which includes remote collaborators audio mixing actions) through uncompressed, high-resolution (> 44.1 kHz, 16-bit) audio.

We derive four critical dimensions that need addressing in developing an online DAW collaboration framework in audio mixing (see Fig. 1):

1. *Mode of Operation*: Either a simultaneous (synchronous) or delayed (asynchronous) mode of operation and interaction;
2. *Audio Asset Quality*: The quality, or fidelity, of the project's audio assets accessed by each collaborator;
3. *Scalability*: An ability to scale the collaboration session to well beyond one-on-one interactions; and
4. *Access/Control of DAW Project*: An ability to access and control/operate the DAW project by each collaborator.

For an online DAW collaboration framework to facilitate the needs of the two scenarios presented above, it must deliver the following requirements for the four critical dimensions (see Fig. 1):

a. *Synchronous operation*: A mode of operation, where all collaborators can interactively communicate/participate during the collaboration session;

b. *High-Resolution Audio:* Access to high-resolution audio assets by each collaborator. High-resolution audio consists of a sample rate of at least 48 kHz and resolutions of at least 24-bit, the generally accepted benchmarks for pre-mastering, professional-grade music production [1,2];

c. *Multiple Collaborators:* Scalability to encompass multiple collaborators (for example, up to 26 collaborators including the online tutor in Scenario 2); and

d. *All Collaborators:* Access to and control of the collaborative DAW project that is equitable and localized for all of the collaborators, allowing for real-time monitoring of remote changes and a means of contributing to the project through DAW-enabled actions by each collaborator.

Existing DAW collaboration platforms fall short of meeting at least one of the above requirements (see detailed discussion in SEC. 1). Thus, in the paper, we present a framework for online collaborative professional audio mixing, meeting all four requirements presented above. The framework provides a unique approach and broadens the paradigm for remote online collaborative audio mixing.

The organization of this paper is as follows: SEC. 1 provides context and discusses related work in this area. In SEC. 2, we present the design of the framework in detail. SEC. 3 outlines the implementation of the framework and a detailed evaluation of the framework is provided in SEC. 4. SEC. 5 concludes the paper with a discussion of future research directions.

1 RELATED WORK

In this section we review existing DAW collaboration platforms, including mainstream standalone and web browser-based applications [3,4], and evaluate their capacity to meet the requirements outlined in SEC. 0, based on the four dimensions.

Existing platforms with a synchronous mode of operation typically forego even a modest number of simultaneous collaborators in favor of streaming compressed, high-quality (sub-high-resolution) real-time audio to effectively minimize the demands on available bandwidth. Source Elements' DAW-agnostic *Source-Connect Pro* [5] and Steinberg's *Cubase*-centric *VST Connect Pro* [6] are successful pioneers of this design ethic. They are perhaps better characterized as remote performer/recording platforms, wherein the "studio" collaborator is the only one with access to the DAW project. Similarly recent platforms, including *Soundwhale* [7], Audiomovers' *Listento* and *Listento Receiver* [8], *Sessionwire Studio* [9], and *ConnectionOpen* [10], variously offer "studio-quality," "high-resolution," "high-quality," or "uncompressed" audio streaming directly into a collaborator's remote DAW platform. Of the platforms mentioned above *Source-Connect Pro*, *VST Connect Pro*, and *Soundwhale* provide synchronized playback and recording between the application and a remote DAW; nevertheless none can currently provide an environment for multiparty, collaborative, and democratized audio mixing.

A synchronous environment can indeed function with multiple collaborators; however audio streamed over a network, and especially audio streamed over the Internet, is more susceptible to latency and jitter [11,12]. For those users unable to access high-speed connections, platforms such as *Listento* offer the option of streaming sub-high-resolution lossy/compressed audio between all participants to reduce the likelihood of jitter, dropped packets, and latency [8,13].

Inserting a multiparty audio streaming platform as a plug-in into a DAW's main stereo mix bus gives an engineer the ability to stream a mix directly to multiple remote locations, thereby facilitating a means of synchronous auditioning and providing feedback, provided that there are also synchronous lines of group communication established. In such a scenario, however, only the one mixing engineer has access to the DAW; therefore only one person in the collaboration can execute changes to the mix as the mixing progresses. While this arrangement may suit some audio mixing scenarios that are centered on the mixing skills of a single sound engineer, this approach does not enable additional collaborators at different locations to be simultaneously involved with the practicalities of mixing.

Some approaches have chosen to forgo synchronous operation in favor of asynchronous audio and project file sharing via cloud storage, consequently facilitating many collaborators to work with high-resolution audio assets. Avid's *Cloud Collaboration* [14] and Steinberg's *VST Transit* and *Transit Join* [15] are examples of this asynchronous approach. Spotify's *Soundtrap* [16] and BandLab Technologies' *BandLab* [17], both archetypes of a web browser-based DAW platform, are similar in design and adopt Web Audio and Web MIDI application programming interfaces (APIs) to implement their functionality and audio processing.

Naturally an asynchronous mode of operation is at odds with the reactive context of collaborative audio mixing in the studio, which values in-the-moment auditioning and spontaneity. Asynchronous environments are typified by collaborators who contribute to a DAW project in isolation, saving and uploading these changes to an updated version of the project in cloud storage. There is typically no mechanism by which to closely observe, audibly monitor, evaluate, and discuss remote contributions and changes instantaneously. Instead changes to the collaborators' local version of the project are uploaded to the cloud then synchronized with a downloaded version at each location. Asynchronous communication is often characterized by text-based messaging within the collaboration platform or co-opting existing messaging through social media platforms, short message service (SMS), or email. Although these platforms are successful for collaborative music creation and remote contributions in, for example, the composition of a piece of music, they do not necessarily achieve an authentic, collaborative studio mixing context as discussed for professional and educational audio mixing scenarios presented in SEC. 0.

Table 1 presents the existing DAW-agnostic and DAW-specific collaboration platforms and identifies their capacity to deliver on the four critical dimensions. It is essential to observe that not one of the platforms is capable of meeting all four dimensions. That is, no platform can synchronously link and provide access for many collaborators to a shared DAW project while also allowing every collaborator to mix and process high-resolution multitrack audio files without requiring post-synchronous collaboration uploading, downloading, auditioning, and decision making.

Both *Source-Connect Pro* and *VST Connect Pro* implement lossy audio codecs, version 2 of the advanced audio coding (AAC) enhanced low delay codec, with sample rates from 44.1 to 192 kHz and bit rates from 32 to 364 kbps [18], and a proprietary process utilizing *Vorbis*, with sample rates from 11.025 to 192 kHz and bit rates from 45 to 500 kbps [19], respectively, for streaming during the collaboration session. These codecs minimize real-time latency and buffer sizes during the session [20,21]; however both platforms also offer an integrated post-collaboration session repair or replacement process using the remote performer's locally stored pulse code modulation (PCM) recording of the session.

Neither platform scales well, with *Source-Connect Pro* offering a maximum of four simultaneous collaborators or four additional audio streams using the multi-connect feature, which provides three additional instantiations within a DAW project. *VST Connect Pro* is strictly a one-to-one collaboration model but does have the additional benefit of an accompanying video and chat capability and can stream up to sixteen simultaneous audio channels for multichannel recordings, such as discrete drum kit components [21].

Avid Cloud Collaboration and *VST Transit/Transit Join* utilize the *WavPack* codec, which encompasses sample rates from 6 to 192 kHz and bit depths of 16, 24, and 32-bit and 32-bit floating [22] and Free Lossless Audio Codec, capable of sample rates from 1 to 655.35 kHz and bit depths of 4, 8, 16, 24, and 32-bit [23], respectively, to maximize cloud storage capacity. Employing these codecs reduces the size of the audio files without detrimentally affecting

Table 1. Comparison of DAW collaboration platforms.

Platform	Synchronous Interactions & Communication	High-Resolution Audio (> 44.1 kHz, 16-bit, Lossless, Uncompressed)	Scalable (> 2 Peers)	Equal Access to/Control of the DAW Project
Source-Connect Pro (v.3.9) [20]	√	√ (Using asynchronous Auto-Restore/Auto-Replace)	√ (Limited to 4)	×
VST Connect Pro (v.4) / VST Connect Performer [21]	√	√ (Using asynchronous HD replacement option)	×	×
Avid Cloud Collaboration [10]	×	√	√ (Limited to 11)	√
VST Transit / VST Transit Join [11]	×	√	√	√
Soundtrap [12]	×	×	√	√
BandLab [13]	√	×	√	√
Soundwhale [3]	√	√ (Using asynchronous third-party file-sharing of local PCM recordings)	×	×
Listento/Listento Receiver (v.20200416) [4]	√	√	√	×
Sessionwire Studio [5]	√	√	×	×
ConnectionOpen (v.3.7.0) [6]	√	× (44.1 kHz, 16-bit; 48 kHz, 16-bit possible)	√ (Limited to 3)	×

their resolution and maximizes download data limits. The asynchronous nature of these platforms translates to much-improved scalability, with *Avid Cloud Collaboration* limiting the number of collaborators at any given time to eleven and *VST Transit* encompassing unlimited but asynchronous participation.

The Web Audio API is at the center of the audio processing capabilities of browser-based DAW applications, such as *Soundtrap* and *BandLab*. In contrast, existing professional offline DAW platforms (e.g., *Pro Tools*, *Cubase*, *Logic*) employ long-established, low-latency Audio Stream Input/Output (ASIO) [24] or Core Audio [25] protocols for Windows and macOS-based PCs, respectively. Consistent implementation of these protocols, together with digital signal processing interface technologies such as Virtual Studio Technology (VST) [24], AudioUnit (AU) [26], and Avid Audio eXtension (AAX) [27], has given rise to an ever-expanding market of audio processor plug-in applications.

While the browser-based user environment and cloud storage are well suited to online collaboration, browser-based DAWs have yet to see mainstream acceptance in professional mixing of recorded music, mainly owing to their incompatibility with existing VST, AU, or AAX-based plug-ins. Furthermore, while developer efforts have resulted in progressive improvements to input and output latencies, Web Audio cannot provide the same low-latency response expected in professional audio mixing [28]. Additional limitations include relatively small track counts (*BandLab*) and use of the *Vorbis* (lossy) codec in processing audio assets before their storage in the cloud (*Soundtrap*).

Recent DAW-agnostic collaborative streaming platforms utilize audio codecs such as AAC [29] (*Soundwhale* [30]; *Listento/Listento Receiver* [8]) and *Opus* [31] (*Sessionwire Studio*) at bit rates below that of uncompressed audio; however they can encounter jitter, dropped packets, and latency issues. With a fixed bit depth of 16 bit, *ConnectionOpen* allows the user to nominate a sample rate compatible with their system's chosen ASIO or Core Audio driver. Significantly, the *Listento/Listento Receiver* standalone and plug-in applications currently offer three "PCM" audio stream

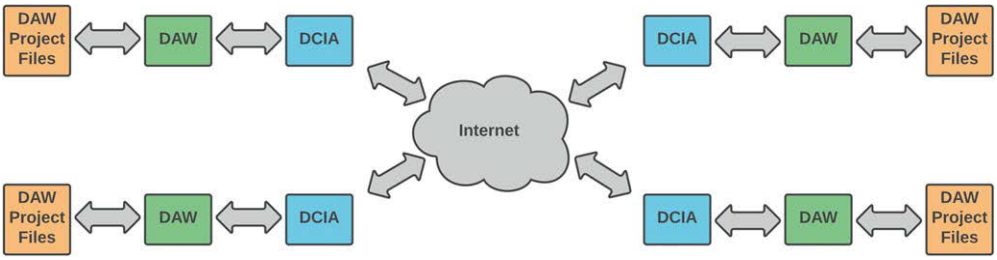

Fig. 2. The overarching design of the DAW Collaboration Framework (DCF).

options with varying bit depths, specifically 16-bit, 24-bit, and 32-bit float, paired with a user-defined sample rate, usually dictated by the associated DAW project's sample rate [8]. Consequently this platform provides high-resolution audio streaming and remote recording, albeit with the risks associated with media streaming over the Internet (see SEC. 4.2 for details on media streaming protocols). Furthermore the plug-ins cannot synchronize playback/recording between remote DAW instantiations; therefore streamed recordings require post-collaboration alignment with existing project events.

In summary current online DAW collaboration platforms fall short of meeting all four of the dimensions outlined: a synchronous mode of interactions and communication, an ability to mix and process high-resolution audio assets, scalability to encompass many collaborators, and equitable control of and access to a DAW project for every collaborator. In the next section we present the design of our proposed framework that is capable of meeting the requirements on all four dimensions with a potential to lead to new online collaboration and education models and practices in professional audio mixing of recorded music.

2 DAW COLLABORATION FRAMEWORK (DCF): DESIGN

The existing synchronous collaborations platforms have taken two distinct design approaches:

1. Streaming sub-high-resolution audio to cater for many remote collaborators; or
2. Streaming high-resolution audio to cater for a limited number of remote collaborators only.

These approaches are primarily due to the limited bandwidths available in today's Internet for streaming high-resolution audio assets to multiple collaborators during a collaboration session.

To address this limitation the proposed framework, termed DAW Collaboration Framework (DCF), avoids streaming high-resolution audio data in real time during the collaboration session. Instead the DCF streams low-bandwidth control data, generated by each collaborator's actions, to extend DAW control to all other collaborators' DAW project instantiations over the Internet. This approach allows the platform to scale to many collaborators, as the bottleneck of network bandwidth is addressed by only transmitting low-bandwidth control data while providing each collaborator with control of the DAW project and access

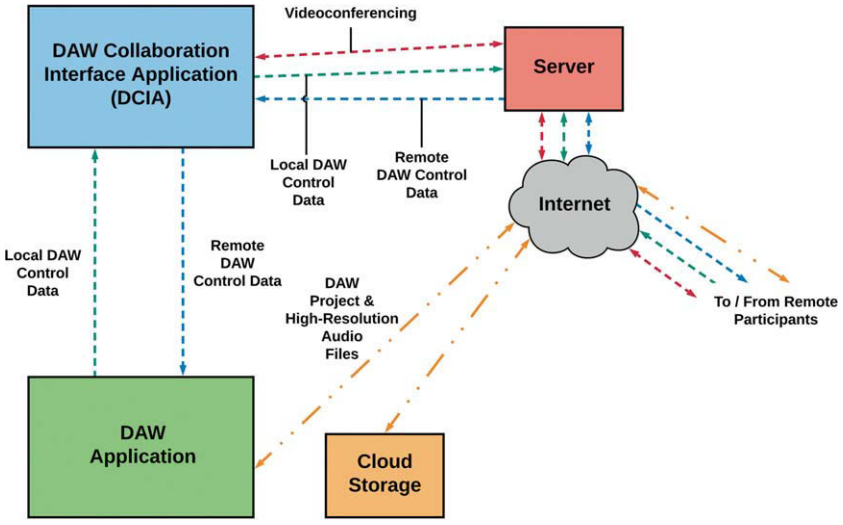

Fig. 3. The role of the DCIA: locating and connecting to remote collaborators and streaming communication media (audio and video streams and optional text-based messages) and the MIDI control data messages to and from remote collaborators and their local DAW via the Internet.

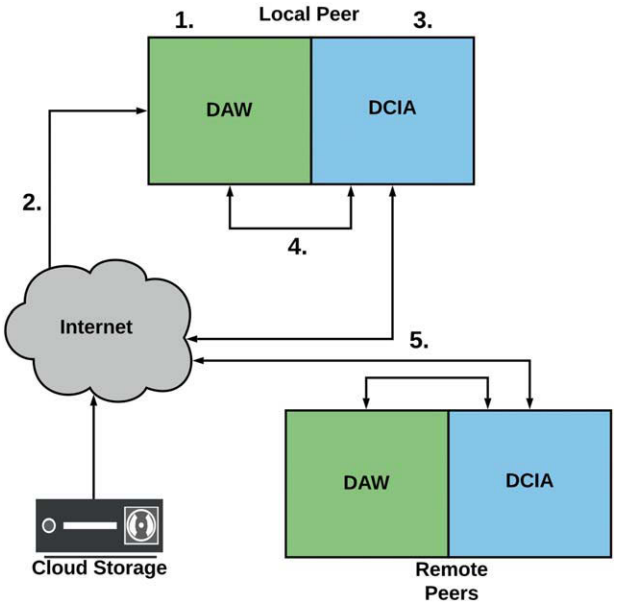

Fig. 4. The overall process in a collaboration session, integrating cloud storage.

to high-resolution audio assets shared before the collaboration session. This framework enables real-time interactions and communications with multiple online peers, all with the capacity to access and mix a shared DAW project and monitor the audio playback in high resolution. To our knowledge this approach has not been previously successfully applied over the public Internet. Brock et al. [32,33] have identified control data streaming, in conjunction with high-resolution audio over IP, as a novel and promising approach to distributed remote audio post-production in the film industry. However due to the Internet's bandwidth limitations and lack of high-level Quality of Service, successful deployment of the collaboration solution was restricted to a privately managed research network [34].

Fig. 2 presents the overarching design of the DCF. A Web application termed DAW Collaboration Interface Application (DCIA) interacts with each collaborator's DAW application to transmit and receive control data to synchronize DAW operations and the session's real-time videoconferencing and chat communication streams. Multiple collaborators synchronously interact with their local DAW project, accessing, mixing, processing, and monitoring high-resolution audio assets.

2.1 DCIA and DAW Application Interfacing

The DCIA performs two essential roles:

1. Establishing a crucial link between the Internet and DAW (see Fig. 3) to enable relaying of DAW-generated control data in the form of Musical Instrument Digital Interface (MIDI) messages; and
2. Locating, connecting, and streaming communication media (audio and video streams) and MIDI control data messages to and from the remote collaborators.

By running concurrent instantiations of the DCIA and a DAW application on one machine, every peer can interface with potentially any DAW platform that accepts and transmits remote control MIDI messages and links that DAW instance with all other remote instantiations in the collaboration. Furthermore a music production project file and its associated high-resolution audio assets can be distributed and downloaded by the participants before or at the outset of a collaboration session by existing asynchronous file-sharing or cloud storage access methods.

Fig. 4 outlines the overall process in a collaboration session. A participant opens the DAW application (Step 1) and downloads the DAW project and high-resolution audio files from cloud storage (Step 2). The participant then navigates to the DCIA browser-based application (Step 3) and selects two virtual MIDI ports to interface the DCIA with the DAW application (Step 4). Finally the DCIA establishes network connections with remote participants to create a collaboration session and stream locally generated, and receive remotely generated, communication media and DAW control data (Step 5).

2.2 Connection Architectures

There are several ways in which to design the communication architecture between collaborators [35]. In a decentralized architecture each site directly communicates with the other sites (in a peer-to-peer manner), forming a mesh architecture for group interactions [36,37] [see Fig. 5(a)]. In a centralized architecture a central server (such as a media server) facilitates communication among collaborators. The central server can integrate communication channels to form a mixing architecture that provides a single channel between each collaborator and the central server [see Fig. 5(b)] or route communication channels to and from each collaborator to form a routing architecture [see Fig. 5(c)]. Deployment of the central server can be as a standalone hardware component [38] or a cloud-based service [39]. We discuss the implementation and evaluation of different connection architectures in SEC. 4.

2.3 Features and Benefits of the Platform

The design of the DCF and DCIA derives several benefits. The notable features of our approach include:

- *Scalability*: A significant reduction in the volume of streamed data during a collaboration session around which the successful operation of the collaboration hinges. The streaming of MIDI control messages and the choice to relegate the importance of data-intensive streamed audio/video to only support videoconferencing and chat between peers have meant reduced bandwidth usage and thus the capacity to significantly increase the number of participants that can interact in real-time;
- *Access to high-resolution audio:* By avoiding high-resolution audio streaming during a collaboration session, each peer monitors the DAW project's audio assets directly from their local DAW instantiation;

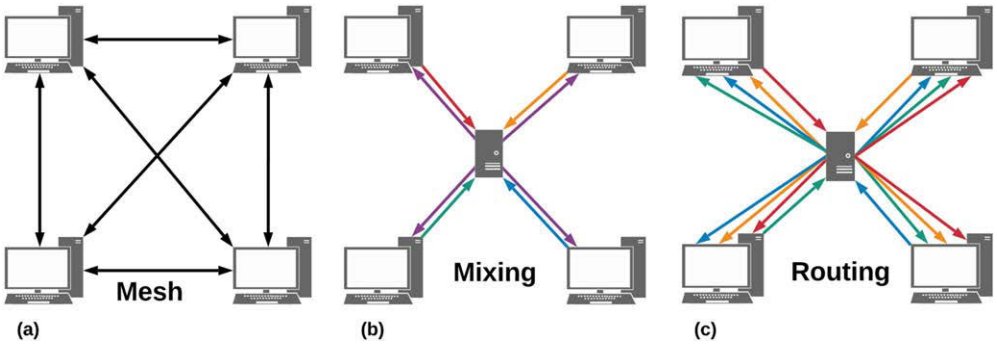

Fig. 5. Communication architectures for the DCF: (a) decentralized mesh architecture; (b) centralized mixing architecture; and (c) centralized routing architectures.

therefore the resolution of the DAW audio playback is only limited by:
  - The project's local audio file characteristics, particularly sample rate and bit depth;
  - The specification of a peer's audio A-D/D-A interface; and
  - A peer's monitoring equipment;
- *Access to and remote control of the DAW project:* Through mapping and the triggering of MIDI messages, local DAW operations and functions generate the control data streamed to all other peers, upon the reception of which the remote DAW instantiations mirror the exact operations and functions. The DCF provides every collaborator with full control of their local DAW project while also receiving and replicating all remote collaborator actions; and
- *Use of the public Internet:* Given that the participants in the collaboration monitor their local DAW-generated audio playback throughout the audio mixing work, each is unaware of the slight differences in the inherent latencies between them, rendering the public Internet more than capable of providing the required bandwidth, throughput, and Quality of Service. Also, today's Internet bandwidths are capable of providing videoconferencing facilities to utilizing unreliable protocols such as User Datagram Protocol (UDP).

The design choices of the platform and features have envisaged a professional audio mixing platform that can scale to a significant number of participants using typical Internet bandwidths with each collaborator having access and control of a shared DAW project with high-resolution audio assets, all the while communicating via videoconferencing and chat with every participant in a collaboration session. This design meets the four critical requirements for Scenarios 1 and 2 outlined in SEC. 0. The following section discusses the implementation and evaluation of the DCF.

3 DCF IMPLEMENTATION

We begin this section by discussing the choice of DAW platform and the features that enable us to integrate the DAW into the DCF architecture. We then discuss how the DCIA utilizes the WebRTC API to facilitate real-time online group communication, followed by an examination of how the Web MIDI API and WebRTC data channels transmit and receive MIDI control messages to and from the DCIA. To conclude this section we present three implementations of various connection architectures.

3.1 Remotely Controlling a DAW Application

In examining the remote-control capabilities of a professional-grade DAW application we chose two platforms: Steinberg's *Cubase Pro* [40] and Cockos' *REAPER* [41]; both are representative of the current professional

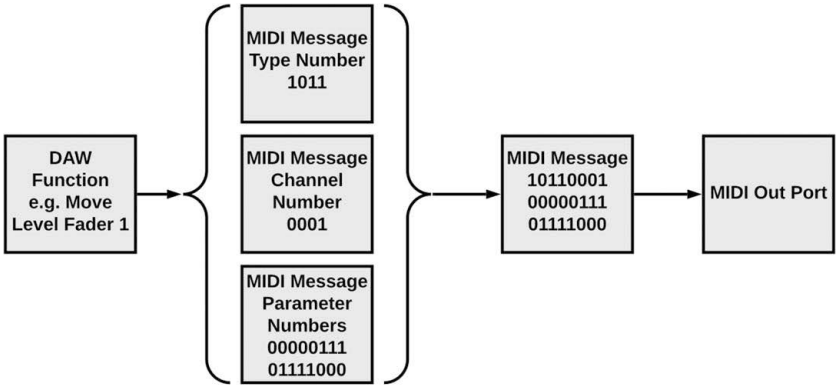

Fig. 6. An elementary DAW function mapped to a MIDI 1.0 protocol message and sent to a MIDI Out port.

```
<?xml version="1.0" encoding="UTF-8"?>
<remotedescription version="1.1">
<ctrltable name="Standard MIDI">
<ctrl>
  <name>Fader 1</name>
  <stat>176</stat>
  <chan>0</chan>
  <addr>7</addr>
  <max>127</max>
  <flags>3</flags>
</ctrl>
<ctrl>
  <name>Fader 2</name>
  <stat>176</stat>
  <chan>1</chan>
  <addr>7</addr>
  <max>127</max>
  <flags>3</flags>
</ctrl>
```

Fig. 7. An example of the XML file format for MIDI mapping DAW functions.

DAW market. Of especial importance was their ability to integrate with an external remote controller, such as a control surface.

Remote controllers, particularly control surfaces, are hardware devices that integrate with DAW software that provides users with a tactile, analogue control over mixing and music production functions. Surfaces generally do not receive or transmit audio signals but instead receive and transmit data commands and parameters that map to a DAW's various functions, employing data protocols including Musical Instrument Digital Interface (MIDI) 1.0 [42] and Open Sound Control [43].

Both *Cubase Pro* and *REAPER* feature external controller integration and include factory default configurations for some of the mainstream MIDI-based devices and control protocols, such as Mackie Control Universal and Human User Interface. These protocols are predetermined, mapping specific MIDI controller data to DAW functions and commands (see Fig. 6). *Cubase Pro* further includes a *Generic Remote* feature that allows users to create a customized MIDI mapping to and from many of the DAW's mixing and audio processing operations. Importantly this mapping can be imported and exported as a standard XML file (see Fig. 7) for sharing between various instantiations of *Cubase Pro*, delivering a uniform response to incoming MIDI control data and the generation of control data when mixing functions, such as level, panning, and plug-in parameter changes, are implemented.

Currently however choosing MIDI 1.0 messages as DAW control data does provide some limitations in functionality. Binary DAW functions, such as mute and solo buttons/switches, which are either on or off, can be readily mapped to MIDI Note On and Note Off events. Similarly DAW functions with values on a continuum, such as level faders, pan-pots, and plug-in parameters, can be mapped to MIDI Continuous Controllers (CCs), but values are restricted to 128 gradations owing to the MIDI 1.0 specification determining that CC values range between 0 and 127 [42]. To illustrate this limitation the operation of a level fader, for example, which is capable of significant positional variations, can only generate a maximum of 128 different fader-position control data messages.

3.2 Real-Time Online Group Communication

The web browser, as an interactive platform for real-time communications, has seen a rapid rate of development and expansion in capabilities in support of online collaboration. In particular the WebRTC standard provides peer-to-peer connectivity, media device capture and streaming over User Datagram Protocol (UDP), and binary data streaming over Stream Control Transmission Protocol (SCTP) [38,44]. UDP, an inherently unreliable 'best-effort' protocol, is commonly used in media streaming over the Internet owing to its low-latency delivery of data packets [45]. UDP however is susceptible to lost and dropped packets and out-of-order packet delivery. SCTP on the other hand offers reliable or semi-reliable packet delivery, providing the ability to determine a maximum number of retransmissions or a maximum packet lifetime and ordered or unordered packet delivery [46].

WebRTC allows for the establishment of real-time online, operating system-agnostic connections between web browsers, and an effective means of communication through synchronous videoconferencing and an asynchronous text-based chat facility. With the work of the Internet Engineering Task Force and World Wide Web Consortium (W3C) in recent years, the WebRTC API has been increasingly integrated and standardized across most of the popular web browser platforms [47]. Though primarily designed with a peer-to-peer connection architecture the inclusion of an external media server significantly increases the number of simultaneous connections enabling expanded group communications and interactions.

For all its advantages, WebRTC does include one notable drawback for professional music production collaboration. WebRTC's media streaming API utilizes the *Opus* audio compression codec [48] across its audio streams, producing sub-high-resolution audio [49]. Therefore while developers can exploit WebRTC's APIs to identify online peers, establish connections, and provide videoconferencing communication between the peers, it cannot deliver a streamed, high-resolution audio source for the collaboration.

3.3 Utilizing and Integrating the Web MIDI and WebRTC APIs

Grounding the DCIA in a web browser environment not only allows the application to exploit the WebRTC API but also integrate the Web MIDI API [50]. Developed by the Web Audio Working Group of the W3C, Web MIDI gives *Chromium*-based browsers [51] access to the local computer's MIDI devices and ports and facilitates MIDI data flow between them. Given that mapped MIDI commands can remotely control a DAW it was, therefore, advantageous

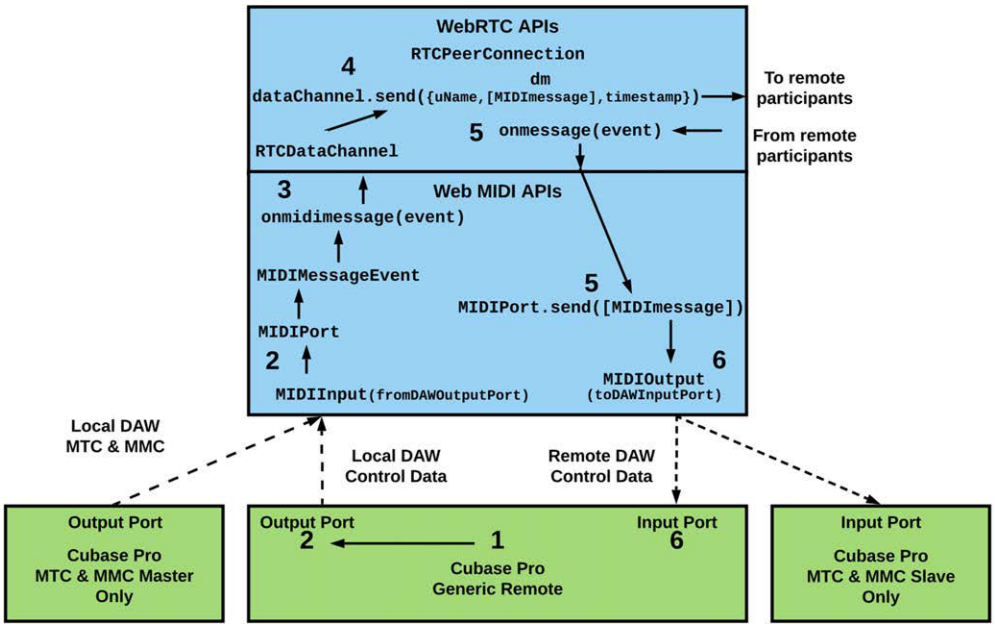

Fig. 8. The six steps in the data flow between Cubase Pro (the DAW) and the Web MIDI and WebRTC APIs, methods and handlers.

to explore the functionality that Web MIDI presents. Of crucial importance is the application's facility to disseminate received MIDI events to all participants in a collaboration.

The WebRTC API can establish an online group and deliver audio/visual communications in the browser environment. In addition, it can optionally establish bi-directional data channel connections that carry arbitrary, binary data between the peers in parallel to, but separate from, the media channels [52]. RTC data channels use SCTP for reliable or semi-reliable message delivery and therefore provide the necessary infrastructure to direct a MIDI data flow through the data channels to every participant in the collaboration session. The DCIA's basic design, therefore, has a dual purpose:

1. To establish WebRTC peer connections for streaming media for videoconferencing among collaborators and data for streaming MIDI control data and chat communication; and
2. To gain access to a DAW via the DCIA's Web MIDI ports and the DAW's MIDI-based remote-control feature.

Fig. 8 shows the interplay between the WebRTC and Web MIDI APIs, their methods, and handlers. In this instance the chosen DAW is *Cubase Pro* so that the *Generic Remote* feature can interface with the collaboration application, generating MIDI data events that represent the DAW's local, user-instigated, mixing functions and operations and executing these functions upon receipt of data events from the application.

Described below is the process of interactions between local and remote collaborator DAW instantiations:

Step 1: Operation of the DAW creates a MIDI message that conforms to the XML MIDI mapping file. The MIDI message consists of:

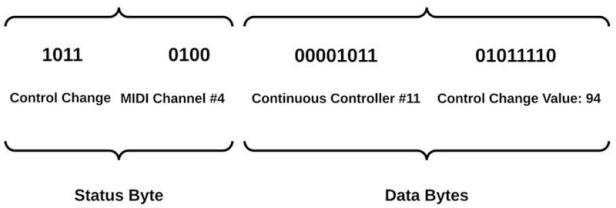

Fig. 9. A typical 3-byte MIDI message format.

1. A status byte, comprising of the message type number [typically a 'Note On' event (1001) or 'Control Change' event (1011)] and the message's selected MIDI channel (0000 to 1111); and
2. A number (typically two data bytes), consisting of the event's parameters, both ranging from 00000000 to 01111111 (see Fig. 9).

To synchronize the playback and navigation of every DAW instantiation in the collaboration it is critical that the DCIA additionally receives and transmits MIDI timecode (MTC) and MIDI Machine Control (MMC) messages. The MTC paradigm uses 2-byte quarter frame messages for synchronization (see Fig. 10). MMC 1.0 is a form of system exclusive MIDI message and as such adopts the universal system exclusive format, shown in Fig. 10.

Step 2: The DAW directs the MIDI message to the Generic Remote's MIDI Output port. A virtual MIDI port is necessary to facilitate the interfacing between the DAW and DCIA and we have used the *LoopBe30* virtual port software for this purpose [49]. The DCIA's Web MIDI `MIDIPort` interface is assigned the same `MIDIInput` port as the Generic Remote's MIDI Output port (see Fig. 11).

Step 3: On receiving a MIDI message event the Web MIDI `onmidimessage(event)` handler executes a function that converts the event data into an array, along

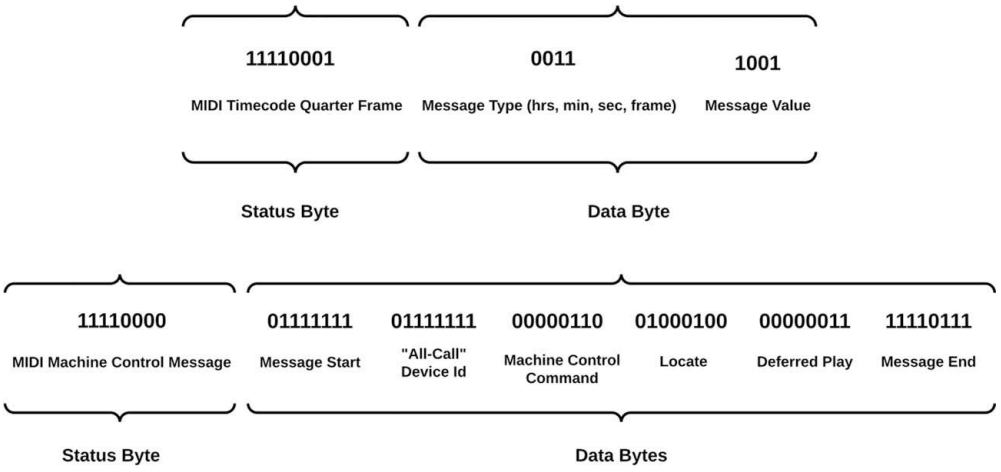

Fig. 10. An archetypal MTC quarter frame message (top) and an example of an MMC message (bottom).

with the peer's username and a timestamp, then converts the array to a string.

Step 4: The DCIA establishes an `RTCPeerConnection` data channel and sends the string, `dm`, consisting of the three elements: the username of the peer who created the message [0], the MIDI message's status and data bytes [1], and a timestamp [2].

Step 5: The DCIA's `onmessage(event)` handler executes a function that parses the incoming string and reconstructs the MIDI message from the second element [1] and sends it to the `MIDIOutput` port (`toDAWInputPort`).

Step 6: The MIDI message travels to the Generic Remote's MIDI Input port, another virtual MIDI port allocated as the DCIA's `MIDIOutput` port.

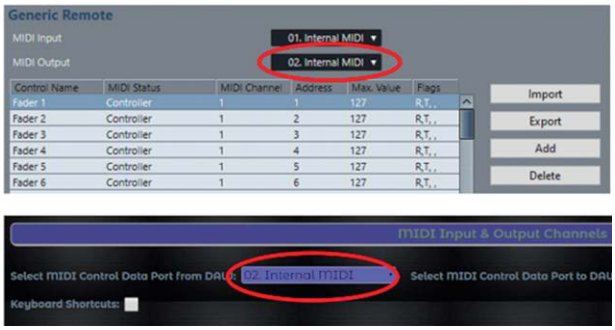

Fig. 11. The Generic Remote MIDI Output port (02. Internal MIDI) assigned to the interface application's MIDI Input (MIDI control data port from DAW) port.

3.4 Connection Architectures

There are several ways to connect collaborators. Firstly we discuss the implementation of a decentralized peer-to-peer mesh architecture, followed by the implementation of a centralized media server-enabled architecture.

3.4.1 Decentralized Mesh Architecture

Deploying the DCIA entailed the creation of numerous P2P connections, facilitated by a simple Node JS signaling server to identify Interactive Connectivity Establishment candidates and perform Network Address Translation traversal (see Fig. 12). This mesh architecture is the most elementary form of WebRTC-enabled group communication, which avoids a central media server in favor of direct, low-latency connectivity.

3.4.2 Centralized Connection Architecture: Mixing and Routing

To implement a centralized connection architecture we chose the *LiveSwitch* [53] WebRTC media server developed by Frozen Mountain. The intrinsic advantages of using *LiveSwitch* in the development of the DCF were threefold:

1. Its ability to operate as a multipoint connection unit (MCU) to establish a mixing architecture, selective forwarding unit (SFU) to establish a routing architecture, or a hybrid of both. This flexibility allowed us to experiment and investigate which model of operation provided the best user experience;
2. Its ability to route RTC data channels through the server rather than to establish multiple P2P connections; and
3. The option of creating multiple servers to distribute and serve the media and control data streams separately.

Our implementation utilized Frozen Mountain's *LiveSwitch Cloud* server, which hosts the media and data servers on their cloud infrastructure, to avoid the cost of setting up a hardware-based media server. Accessing the *LiveSwitch* server requires the generation of a secure web token so that users can register with the server, enter the collaboration group videoconference, and participate in data transfer between DAW instantiations. We implemented a Node JS web server for this purpose and locally hosted it on a Windows 10 laptop PC (Intel Core i7-3610QM CPU

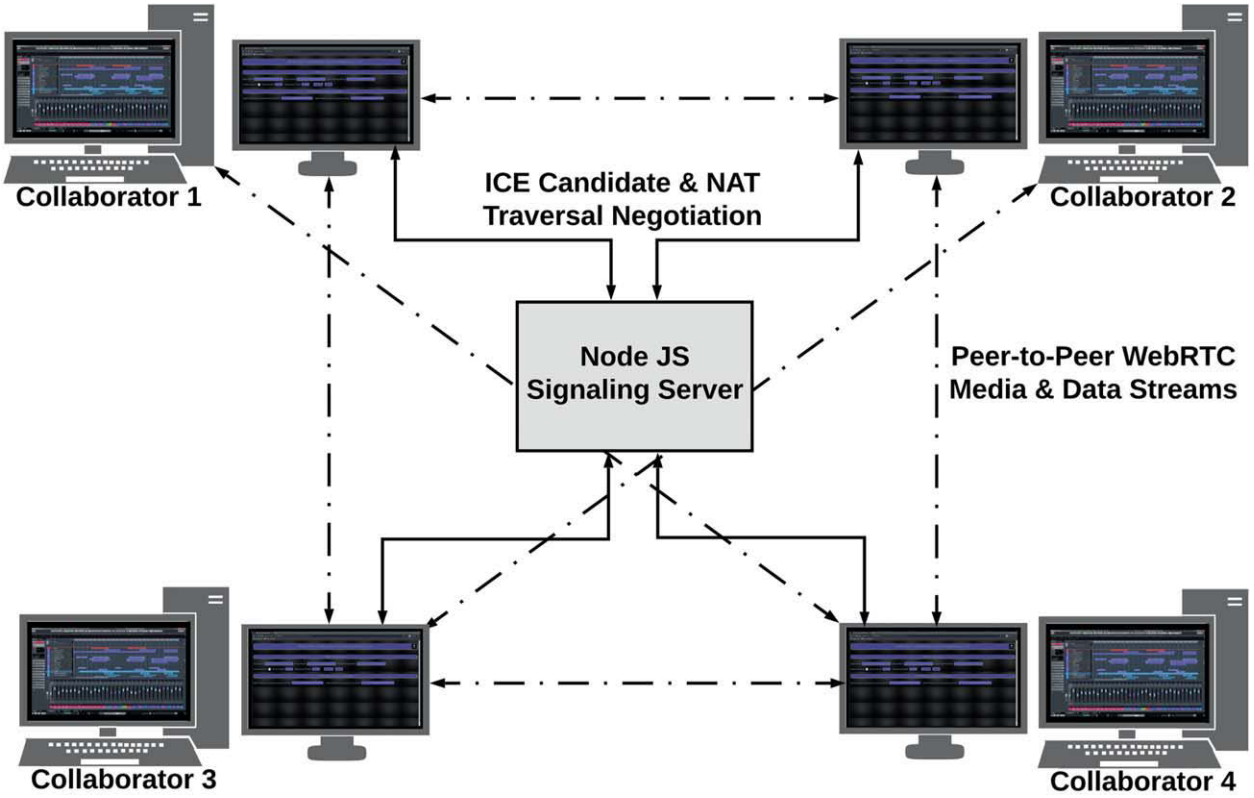

Fig. 12. Peer-to-peer mesh architecture with a Node JS signaling server.

@ 2.3 GHz) as the architecture's *Authorization Server* for our experiments.

Fig. 13 illustrates the architecture and signal flow supporting the online collaboration process. The DCIA, as in the mesh architecture, interfaces with *Cubase Pro* directly through the *Generic Remote* feature DAW-side and Web MIDI ports application-side, routing control data messages to and from the DCIA's WebRTC data channels. The *Authorization Server* generates JavaScript Object Notation (JSON) web tokens (JWTs), directs the DCIA to the *LiveSwitch Gateway* server, which functions as the signaling server, and registers the DCIA with the *LiveSwitch Cloud* servers using the respective JWTs. Generating a JWT requires a combination of client-side and server-side objects, the combination of which ensures a secure means of accessing a collaboration session.

Two separate instantiations of the *LiveSwitch* media server were employed (see Fig. 14): one for audio-visual streams for videoconferencing (termed *Media Server*) and the other for control data and text streams (termed *Data Server*). The *Media Server's* configuration uses UDP to stream audio-visual data streams efficiently while the *Data Server* uses SCTP to stream control data and text data reliably. Using two servers provides a division of the DCF's data streams and allows for experimentation with hybrid MCU/SFU architectures. The dual-server model also delivers a level of redundancy if one server goes offline, permitting either the videoconferencing or text-based chat feature to continue. The Authorization Server generates two JWTs: one for the collaboration's media (audio and video) streams and the other for the collaboration's data streams.

4 DCF EVALUATION: RESULTS AND DISCUSSION

This section presents the evaluation of DCF. The organization is as follows: Firstly we present the metrics and environment used for the evaluation then outline the client-side CPU performance results, client-side and server-side bandwidth usage results, and scalability of the DCF to multiple collaborators, using different connection architectures. We conclude this section with a discussion of the results and limitations identified through the DCF implementation.

4.1 Metrics

The metrics we use to evaluate the DCF across three connection architectures, specifically mesh, mixing, and routing architectures, are CPU performance, bandwidth usage, and scalability.

Audio processing and DAW mixing operations are intrinsically CPU-intensive and a machine's CPU capability and available RAM can limit such operations. While DAW manufacturers recommend minimum system requirements, for example Steinberg recommends an Intel Core i5 or faster CPU and 8 GB RAM [54], it is not unusual for professional DAW users to work on machines that feature > 3 GHz, $\geq$ 8-core/8-thread processors, and ≥ 32 GB RAM [55]. There-

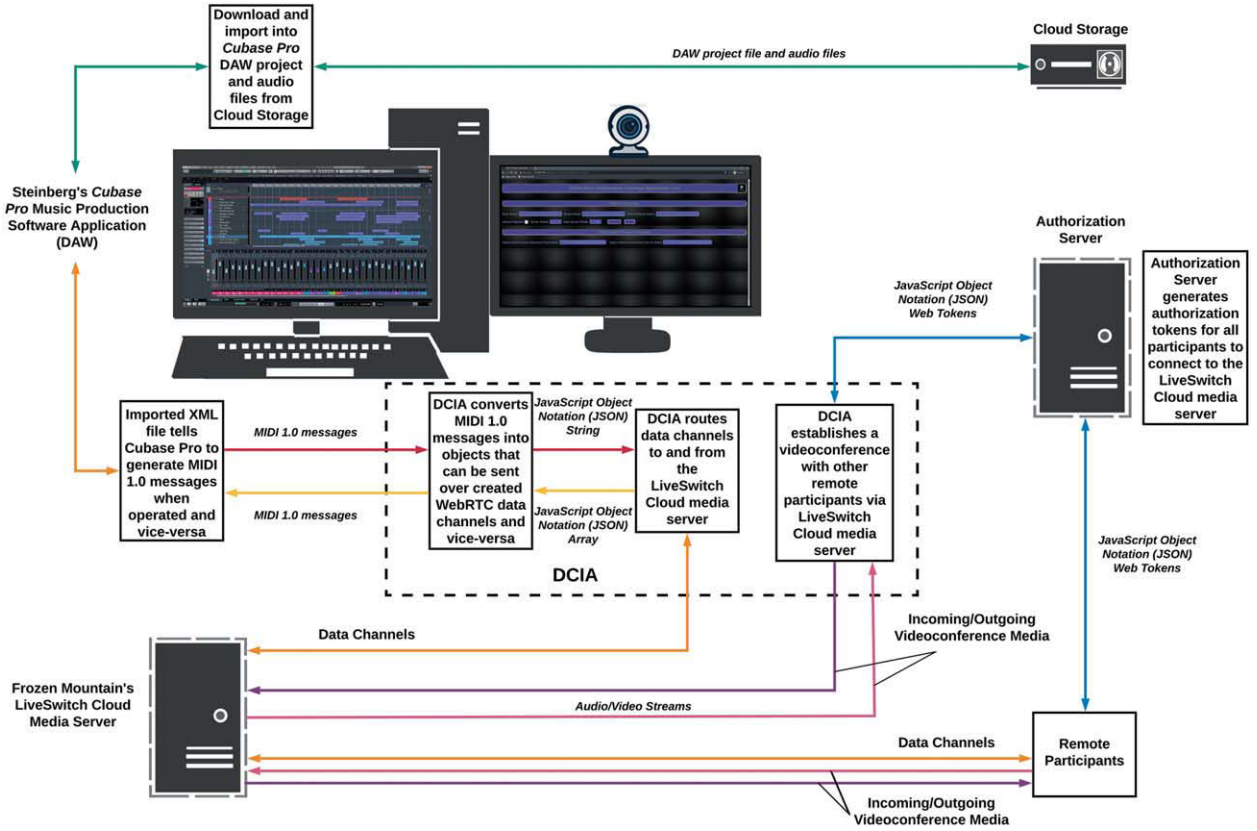

Fig. 13. The centralized architecture and signal flow, using Frozen Mountain's LiveSwitch media server, and a NodeJS authorization server.

fore we sought to find an architecture that would have the smallest impact on the test computers' CPU.

In Australia the National Broadband Network, the only public wired Internet service, aims to provide at least 50 Mbps downstream bandwidth to 90% of the population [56,57]. Currently the top speed tier delivers bandwidth no higher than 100 Mbps. We limited the experimental environment to 50 Mbps and 100 Mbps as typical client and server Internet connection speeds, respectively, and consequently examined the implications these limits present for the deployment of the DCF with and without a central server.

4.2 Testing Environment

Table 2 summarizes the three computers used to conduct evaluations of the DCF in the various connection architectures. Each computer accessed the Internet via discrete networks, specifically a standard residential Internet connection, the University of Newcastle's VPN, and a wireless 4G connection.

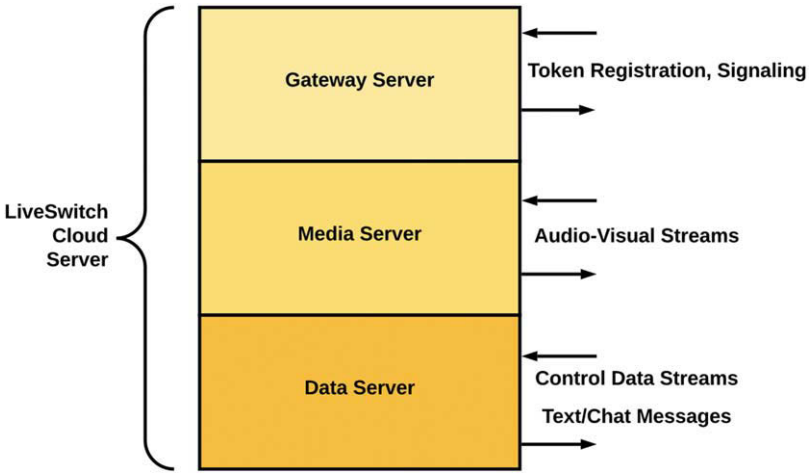

Fig. 14. The LiveSwitch Cloud server stack, consisting of a Gateway Server, Media Server, and Data Server.

Table 2. Specifications of computers used for experiments.

	Specifications	Network Access
Local Client	Dell Latitude 5480 PC Intel Core i5-7300U CPU @ 2.60 GHz 16 GB RAM Windows 10 1909 Enterprise 64-bit OS Chrome 83 Browser Cubase Pro 10.5.20 DAW	VDSL Hybrid Fiber Coaxial Max. 50/ (download/upload) Mbps Residential Internet access
Remote Clients	Custom-Built ASRock PC Intel Core i7-8700K CPU @ 3.70 GHz 32 GB RAM Windows 10 1909 Pro 64-bit OS Chrome 83 Browser Cubase Pro 10.5.20 DAW	4G Wireless Broadband Max. 110/15 Mbps
Authentication Server	Samsung PC Intel Core i7-3610QM CPU @ 2.30 GHz 12 GB RAM Windows 10 1909 Home 64-bit OS	VDSL Hybrid Fiber Coaxial Max. 50/ Mbps VPN

Table 3. Maximum VP8 codec parameter values for real-time streaming.

Parameter	Value
Frame Width	640 pixels
Frame Height	480 pixels
Target Framerate	15 fps
Bitrate	500 kbps

Table 4. Bitrate "sweet spots" for the *Opus* codec (frame duration = 20 ms).

Configuration	Bitrate "Sweet Spot" (kbps)
Narrow-band Speech	8 – 12
Wide-band Speech	16 – 20
Full-band Speech	28 – 40
Full-band Mono Music	48 – 64
Full-band Stereo Music	64 – 128

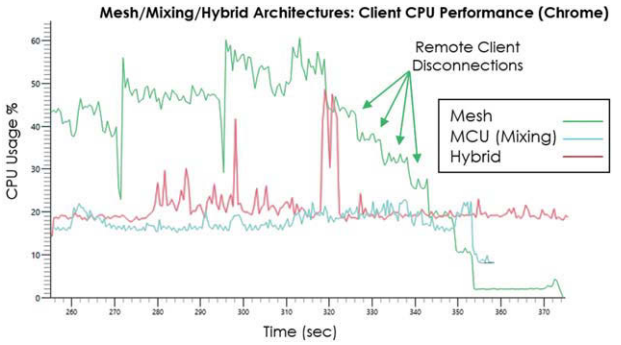

Fig. 15. The percentage of CPU usage by the web browser (Chrome 83) in each of the three connection architectures.

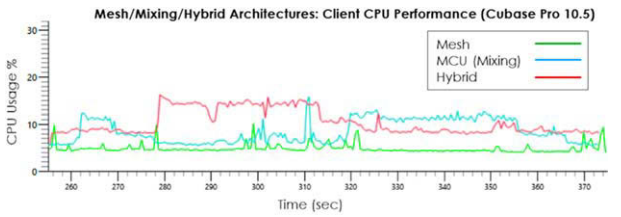

Fig. 16. CPU usage by the DAW (Cubase Pro 10.5) in each of the three connection architectures.

We determined at the outset that the Media Server (see Fig. 14) was to operate as an MCU. This approach provided the flexibility in the on-screen video layout and also outsourced the video transcoding to the *LiveSwitch Cloud* server to free up as much local CPU as possible. Furthermore we restricted the Media Server to only implement the VP8 video codec [58] due to its high optimization rate and compatibility with group videoconferencing [59]. Maximum parameter values were set based on The WebM Project's recommended settings for real-time constant bitrate encoding and streaming (see Table 3) [60].

Similarly the Media Server was restricted to only implement the *Opus* audio codec with a maximum bitrate of 48 kbps, slightly higher than the maximum rate recommended for full-band speech given a frame duration of 20 ms (see Table 4) [61].

4.3 Client-Side CPU Performance

Testing of the mesh architecture demonstrated that DAW-based real-time collaboration was indeed possible, and further, multiple peer connections could provide equitable access and editing capabilities to all the collaborators [4]. Mesh configurations however can place considerable strain on a collaborator computer's available CPU capacity when adding peers to the online collaboration environment. At one point during the test the processes of the two software applications *chrome.exe* and *cubase10.exe* alone consumed a maximum 66.43% of the local client computer's CPU (see Figs. 15 and 16).

The inclusion of the cloud server in the DCF's architecture provides enormous benefits for the client compared to a mesh architecture. Outsourcing the audio-visual mixing, transcoding, and routing to the Media Server generally results in significantly lower demands on a client's CPU, primarily due to the reduced amount of processing the web browser is required to undertake. Fig. 15 also plots *Chrome*'s rate of processing with both Media and Data servers functioning as an MCU and with a hybrid architecture of an MCU Media Server and SFU Data Server.

Across a test with seven peer connections, *Chrome*'s baseline CPU usage is similar in both the MCU and hybrid environments; however, interestingly, the inclusion of the SFU did see some spikes in CPU consumption not seen in the purely MCU-based configuration. Whereas the mesh architecture demonstrated peak CPU usage of 60%, the deployment of the MCU and hybrid architectures resulted in

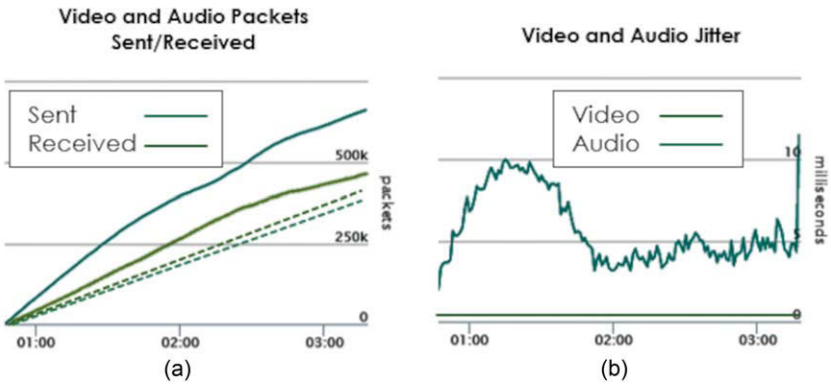

Fig. 17. (a) The Media Server's video and audio packets sent and received; and (b) video and audio jitter for a typical peer connection over a 10-peer connection test.

peaks of 22.4% and 49.8% on the local client, respectively. This reduction, particularly that of the MCU-based architecture, provides sufficient enough headroom for the DAW application to perform relatively intensive audio processing before exceeding the computer's CPU capacity.

Across all three architectures, as expected, there was no difference in *Cubase Pro*'s CPU consumption, as illustrated in Fig. 16. In all three instances CPU usage hovered between 5% and 12% while performing basic playback, level fader, pan pot, and audio compression parameter changes, irrespective of the number of peer connections at any given moment.

4.4 Bandwidth Usage

This section presents the results of bandwidth usage, beginning with server-side bandwidth usage results generated by an MCU Media Server, and both MCU and SFU configurations for the Data Server. We then present client-side bandwidth usage when deploying the client in a mesh architecture, with MCU-configured Media and Data Servers, and with a hybrid MCU Media Server and SFU Data Server architecture.

4.4.1 Server-Side Results

In analyzing the server-side *LiveSwitch Cloud* server requirements, we present the Data Server's bandwidth usage when operating as either an MCU or SFU. The Data Server receives, mixes, and transmits the DCF's MIDI event messages and text messages generated by the chat feature.

Audio/Video Transmission by Media Server (MCU Mode):

Video Performance: A test with ten simultaneous peer connections demonstrated that a typical connection encountered consistent video streaming performance, with steady video streaming rates, minimal received packet loss, and exceptionally low jitter (see Figs. 17 and 18). The results confirmed the suitability of the *LiveSwitch Cloud* Media Server's video configuration settings (see Table 3), particularly the choice of the VP8 codec and maximum parameter settings.

Audio Performance: The test as mentioned above also produced positive results for the Media Server's audio streaming, with the *Opus* codec (see Table 4) providing consistent streaming rates across the entirety of the test for a typical peer connection, as seen in Fig. 17. The connection had more than 400,000 packets of audio data received during the test and only just over 150 packets lost (see Fig. 18), resulting in a loss percentage of approximately 0.038% spanning the entire collaboration session.

In contrast to the Media Server's video performance, the audio stream connection did encounter some entirely expected jitter. Fig. 17(b) demonstrates that after an initial peak jitter of around 10 ms the typical jitter encountered

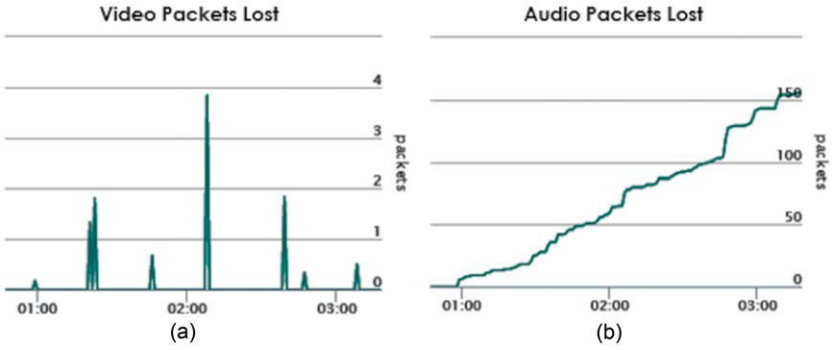

Fig. 18. The Media Server's lost (a) video and (b) audio packets for a typical peer connection over a 10-peer connection test.

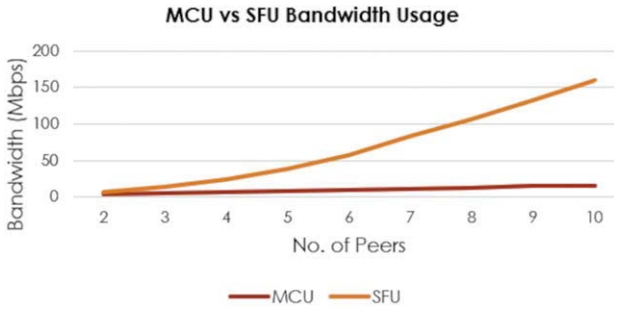

Fig. 19. Bandwidth usage of Data Server serving up to ten simultaneous peers using an MCU and SFU modes.

by the audio connection sat around 5 ms. It is significant to note that the quality of the videoconferencing audio is of secondary importance to the collaboration since its usage is for videoconferencing purposes only.

Data Transmission by Data Server (MCU vs SFU): When deciding on the Data Server's data transmission modes, test results produced markedly different bandwidth usage between the MCU and SFU server modes of operation (see Fig. 19). In an MCU architecture each endpoint establishes a single bidirectional connection with the Data Server:

$$c_t = p_t$$

where c_t specifies the total number of connections and p_t specifies the total number of peers.

In contrast the SFU architecture establishes a 'single-up, many-down' framework:

$$c_t = (p_t)^2$$

Testing was conducted with up to ten simultaneous client connections to determine the average bandwidth usage with the Data Server operating in MCU and SFU modes of transmission. The DCF created a maximum of ten connections in the MCU architecture and saw a linear increase in bandwidth usage with the addition of each new connection (see Fig. 19). In stark contrast, with the Data Server functioning in SFU mode, the framework created a maximum of 100 discrete connections to accommodate the 10 simultaneous clients, resulting in an exponential increase in bandwidth usage. To illustrate the point further, at 10 connected peers, the MCU used an average 15.4 Mbps bandwidth, whereas the SFU used an average of 160 Mbps.

4.4.2 Client-Side Results

In this section we report on a test client's bandwidth usage in the following connection architectures:

1. A mesh architecture consisting of several P2P connections;
2. A mixing architecture using an MCU for audio/video communication and DAW control and text/chat data transmissions; and
3. A hybrid architecture using an MCU for audio/video communication transmissions and an SFU for data transmissions.

Mesh Architecture: Each new addition of a client to the collaboration in a Mesh architecture deployment saw an exponential increase in both bandwidth usage (see Fig. 20) and latency contributing to the marked instability exhibited when a seventh and eighth client joined the group. Despite some evident jitter when the fourth client joined the architecture the DCF demonstrated relative stability up to and including a mesh of six clients, constituting 21 discrete connections across the collaboration.

MCUMedia and Data Streams: Employing an MCU for both media and data transmission in the DCF provides for a less bandwidth-intensive environment for each client. An MCU connection architecture only requires the establishment of one outgoing and one incoming connection with each of the *LiveSwitch* servers (Media and Data). Rather than observing an exponential increase in bandwidth usage as seen in the mesh architecture (see Fig. 20), Fig. 21 shows that average media stream transmission and reception over UDP at the client is comparatively steady, with bandwidth usage of 0.544 Mbps for outgoing and 0.626 Mbps for incoming audio-visual channels. The media streams also include a small component comprising of Session Traversal Utilities for NAT (STUN) requests and responses to identify and monitor the presence of the remote collaborators.

With the Data Server operating as an MCU, data stream bandwidth usage sees a spike at the establishment of the connection between the client and server as they complete STUN request/response and Datagram Transport Layer Security version 1.2 (DTLS v1.2) handshake protocols, as illustrated in Fig. 22. The traces confirm the success of the JavaScript code written to avoid feedback loops, with the server sending the client a higher bandwidth of remotely generated DTLS v1.2 application data with each new client addition than that sent by the client. Again, the MCU architecture provides the DCF with a modest bandwidth requirement when transmitting and receiving DAW-related control data, even when, as is the case in Fig. 22, the client is acting as the MTC and MMC master and generating quarter frame and machine control timing and navigation messages for the collaboration. Average outgoing and incoming streaming rates sit between 3.56 and 5.35 kbps and 4.12 and 10.21 kbps, respectively.

HybridMCU (Media) and SFU (Data): When configuring the Data Server as an SFU, incoming and outgoing data stream bandwidth usage mirrored each other directly (see Fig. 23). Additionally the bandwidth usage, while significantly lower than that seen in the mesh architecture, demonstrated a linear increase with the addition of each new client to the collaboration for the client and server alike, peaking at 37.22 kbps for outgoing and 42.3 kbps for incoming data streams during a test with a maximum of 9 clients.

4.5 Scalability

For scalability we use the following limits in terms of CPU and bandwidth. The CPU usage at the client should not exceed a point where degradation in performance and connection drops occur. For bandwidth usage we kept an

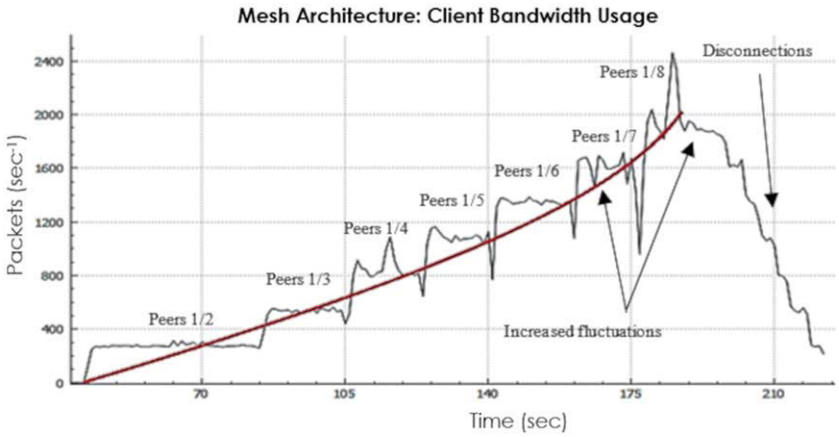

Fig. 20. The bandwidth usage with the addition of new clients to the mesh connection architecture.

upper limit of 50 Mbps client-side and 100 Mbps server-side in total for both of the *LiveSwitch* Media and Data servers. Using the 100 Mbps bandwidth limit in total for both server-side data and media streams is conservative for the *LiveSwitch Cloud* server. However a 100-Mbps connection is reasonable if hosting a *LiveSwitch* media server in-house.

As a result of the Mesh architecture test we determined a maximum of six peers were possible before the performance of the DCIA was noticeably and detrimen-

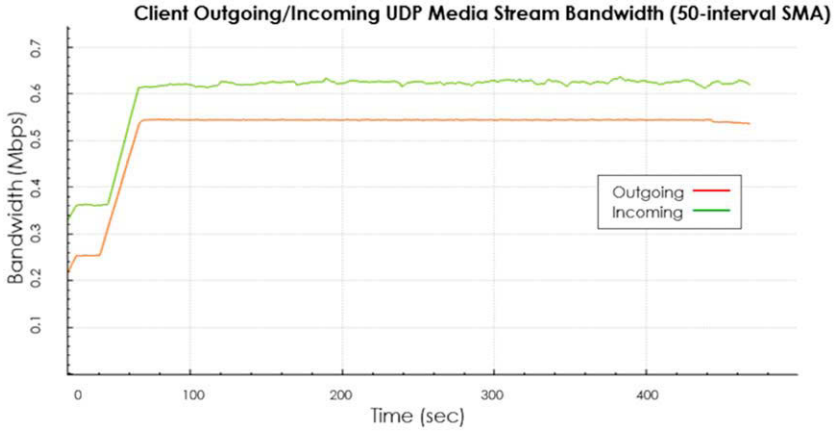

Fig. 21. Outgoing and incoming average UDP media stream bandwidth usage at the client to and from an MCU Media Server (50-interval simple moving average).

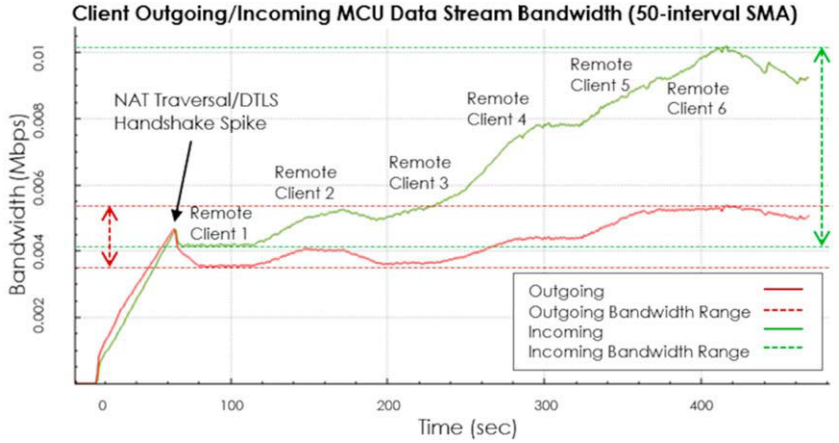

Fig. 22. Outgoing and incoming average SCTP data stream bandwidth usage at the client to and from an MCU Data Server (50-interval simple moving average).

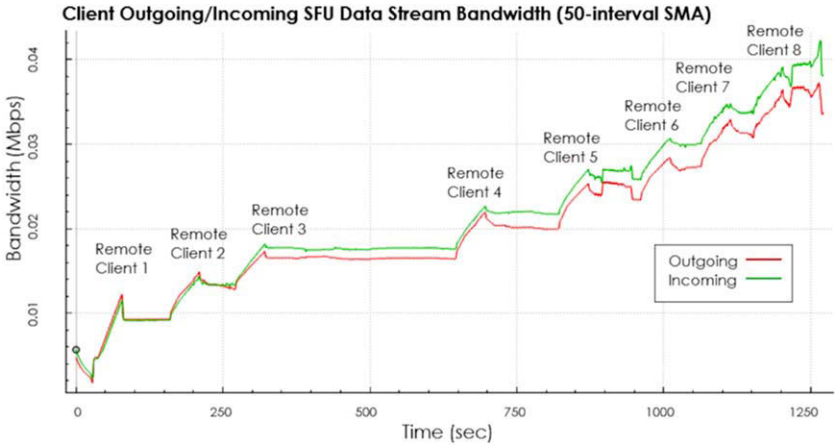

Fig. 23. Outgoing and incoming average SCTP data stream bandwidth usage at the client to and from an SFU Data Server (50-interval simple moving average).

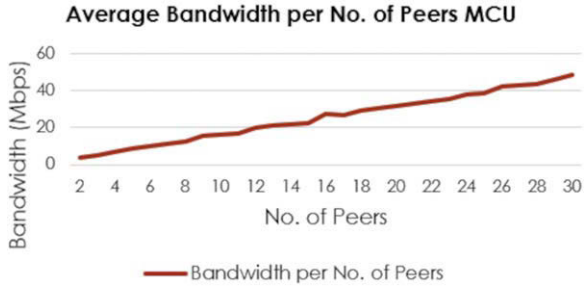

Fig. 24. Average bandwidth usage at a client to serve up to 30 simultaneous peers.

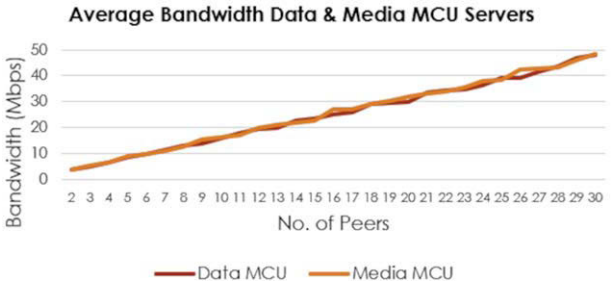

Fig. 25. Average bandwidth usage of the Data and Media Servers operating as MCUs at the server-side.

tally affected, as illustrated in Fig. 20. When utilizing an SFU architecture for the Data Server we reached the 100-Mbps limit at the server-side after the 7th peer connection and before the 8th peer connection joined the collaboration session (see Fig. 19). Employing an MCU architecture for the Data Server, the bandwidth usage scaled linearly at the client-side with a maximum average bandwidth of 48.5 Mbps (see Fig. 24) for a test with up to 30 simultaneous peer connections. This increase in scalability was a significant improvement from the mesh architecture, which saw an exponential increase in bandwidth usage as the number of simultaneous peer connections increased (see Fig. 20).

Fig. 25 traces the bandwidth usage for the two MCU-configured Media and Data servers scaling to 30 peers, with the Data Server requiring a maximum average bandwidth of 48.08 Mbps. As a combined server architecture the *LiveSwitch Cloud* server demonstrated an ability to accommodate 30 simultaneous peers while requiring less than 100 Mbps of bandwidth (see Fig. 26).

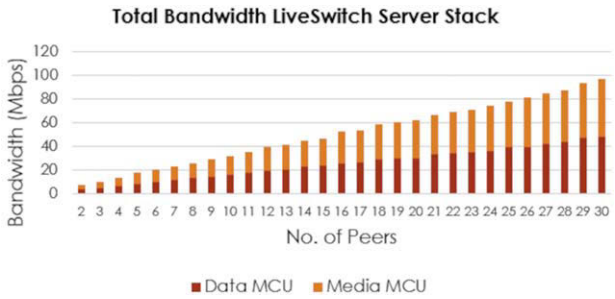

Fig. 26. The LiveSwitch server architecture's total bandwidth usage for serving up to 30 simultaneous peers.

4.6 Discussion

This section presents the results of the experimental DCF implementation using typical computers and public Internet connections (see Table 2) for mesh, mixing, and hybrid mixing/routing connection architectures. The design and implementation of the DCF delivered all four of the requirements identified in SEC. 0:

1. A synchronous mode of operation;
2. Scaling to multiple collaborators;
3. Access to high-resolution audio assets by each collaborator; and
4. Access/control of DAW project for all collaborators.

The DCF meets all of the requirements above (see Table 5) and facilitates novel online DAW collaboration models that realize use-case scenarios in professional and ed-

Table 5. The DCF and dimensions for online DAW collaboration platforms for music post-production.

DCF Connection Architecture	Synchronous Interactions & Communication	High-Resolution Audio (> 44.1 kHz, 16-bit, Lossless, Uncompressed)	Scalable (> 2 Peers)	Equal Access to/Control of the DAW Project
Mesh (Peer to Peer)	√	√	√ (≤ 6 peers)	√
Hybrid (MCU mode - Media Server and SFU mode - Data Server)	√	√	√ (≤ 7 peers)	√
Mixing (MCU mode - Media and Data Server)	√	√	√ (≤ 30 peers)	√

ucational contexts, such as those described in Scenarios 1 and 2.

The mesh architecture avoided the overhead of using a centralized media server but could scale to only six collaborators in the experimental environment. The mesh architecture does however have an overhead of a signaling server to establish the various P2P connections.

For the mixing and hybrid mixing/routing architecture experiments, we used the *LiveSwitch* Cloud server for the centralized connection architectures. Deploying the *LiveSwitch* as an MCU-configured Media Server, together with an SFU-configured Data Server, scaled up to 7 peers before reaching the 100-Mbps bandwidth threshold at the server-side (see Fig. 19). In contrast deploying *LiveSwitch* as MCU-configured Media and Data Servers scaled to 30 peers before reaching the server-side bandwidth threshold of 100 Mbps (see Fig. 26) and 50 Mbps at the client (see Fig. 24). Therefore this result demonstrates the advantage of employing an MCU-configured Data Server over an SFU in a centralized architecture for higher scalability.

4.7 Limitations

The design of DCF requires that all collaborators share the DAW project, including high-resolution audio assets, before the collaboration session. This approach enables to scale to many collaborators in a synchronous mode of collaboration without the need to transmit bandwidth-intensive high-resolution audio data. However this design does limit the collaboration session to work with already shared DAW project files and audio assets. Incorporation of new high-resolution audio assets or DAW project files cannot occur during a collaboration session without synchronizing these data files among collaborators during the session, which can cause delays. However note that the DCF enables use-case scenarios such as discussed in SEC. 0, which shares the DAW project and high-resolution audio assets among all collaborators before the collaboration session.

We identified some practical limitations in DCF implementation. As discussed earlier in SEC. 3.1.1, we anticipated the MIDI 1.0 paradigm, which mandates a range of 0–127 for all CC values, to essentially restrict the resolution of any of the DAW mixing functions mapped to CC events. This restriction produced slightly different fader and pot positions or values between the DAW instantiation generating the MIDI control data and those DAWs receiving the control data via the DCF.

During testing we also identified a concurrency issue when two or more collaborators make simultaneous remote changes to the same DAW function, such as a level fader, plug-in parameter, or mute button. In such circumstances execution of the various changes was according to the order the control messages were received by each collaborator's DAW instantiation. Consequently we observed erratic jumps in fader and pan-pot positions, buttons engaging and disengaging, and variable parameter values. While communication among collaborators via the DCF's video-conferencing can assist with resolving such situations as they arise, we believe the framework should also include features for a more elegant solution to resolve conflicts due to concurrently executing opposing actions among collaborators, which is discussed as a part of future work (see SEC. 5 for further discussion).

The range of DAW functions the DCF can remotely control and execute is determined mainly by the DAW platform's capacity to generate MIDI control data when operated locally. At this time Steinberg's *Generic Remote* feature, integrated into *Cubase Pro*, only transmits MIDI event messages for those functions one would reasonably expect to see on control surface hardware. Therefore, while most of the mix console operations (level faders; pan-pots; mute, solo, and select buttons; plug-in parameters; and transport functions) did indeed generate MIDI event messages and transmit them to the DCIA in testing, other non-mixer functions, such as quantization, adding new tracks, and opening various editors, for example, will not. Nevertheless if Steinberg were to expand the *Generic Remote* feature to transmit MIDI event messages mapped to any of *Cubase Pro*'s functions, the DCF would remotely control and execute these functions as well.

The current *LiveSwitch* data channels are not the full implementation of WebRTC data channels but are instead "optimized for small chunks of binary data or text messages" [62]. Frozen Mountain is presently developing their implementation by incorporating dynamic congestion control and working on the partial-reliability feature set provided by SCTP, including the maxPacketLifeTime and maxRetransmits unsigned shorts and ordered Boolean. Until dynamic congestion control is in place there exists the potential for the DCF's control data transmission to encounter static throttling [62]. In testing, such throttling was noticeable on occasion during extended periods of playback that necessitated long MTC quarter frame streams. This

issue however does not arise when deploying the DCF in a Mesh architecture, since the data channels in this iteration are fully implemented WebRTC data channels, with a low number of maximum retransmissions, and permitted unordered delivery of packets.

5 CONCLUSION AND FUTURE WORK

In summary, this research has identified a shortfall in existing online collaboration platforms to cater to use-case scenarios such as in professional and educational audio mixing of recorded music contexts. Existing collaboration models cannot simultaneously provide real-time interactions among collaborators while providing access to audio mixing techniques via a collaborative DAW project with the ability to mix, process, and monitor high-resolution audio assets in all locations. This paper has presented a unique approach that addresses this gap by providing an online DAW collaboration framework that enables a collaborative session that:

- Delivers synchronous mode of interactions with videoconferencing and chat communications;
- Provides a means of monitoring mixed, processed, and mastered high-resolution audio assets by each collaborator;
- Delivers an environment that is scalable to multiple participants using standard residential Internet connections; and
- Offers real-time access to and mixing of a linked DAW project by all participants.

The platform was successfully implemented over mesh (decentralized) and routing and mixing (centralized) connection architectures and evaluated on CPU performance, bandwidth usage, and scalability. The experiments achieved scalability of up to 30 simultaneous, synchronous collaborators on standard client computers and 50-Mbps Internet client connection, with the incorporation of an MCU Media/Data server architecture. The results also exhibited feasible collaborations of six or fewer participants in a decentralized P2P mesh architecture. This work demonstrated the DCF's ability to meet the four critical dimensions for online collaborative professional and educational audio mixing activities, opening novel ways to collaborate in online music production activities in the future.

There are several areas we are working on to improve the DCF in the future:

- *User evaluations*: Presently we are deploying the DCF in real-world applications, with Australian professional mix engineers and musicians, and evaluating its capabilities from a user perspective.
- *Cross-DAW Interoperability*: We plan to commence investigations into other DAW platforms to determine their capacity to generate and receive control data event messages mapped to DAW functions. These investigations have the potential to lead to the generation of cross-DAW translation templates, which the DCF can implement to link like-functions across a range of DAW platforms.
- *High-resolution control for MIDI commands*: We also plan to improve the resolution issues that MIDI 1.0 CC values currently present by accommodating the MIDI 2.0 Universal MIDI Packet and Capability Inquiry protocols [63], once implemented in the various DAW platforms. Work is already underway to realize this implementation; for example Steinberg recently updated their VST 3 software development kit to support the MIDI 2.0 standard [64]. MIDI 2.0 provides a 32-bit resolution, compared with the 7-bit resolution MIDI 1.0 currently delivers. MIDI 2.0 also provides us with the opportunity to investigate and experiment with its Profile Configuration and Property Exchange capabilities, which have the potential to enhance the DCF's cross-DAW platform compatibility.
- *Handling conflicting actions by collaborators*: We are investigating implementing two approaches to address the issue of collaborators conducting opposing actions simultaneously:

1. Centralized approach: In a centralized approach an assigned "host" of the collaboration session 'controls' the transmission of control data by collaborators. The host can enable/disable each collaborator to transmit control data by implementing permissions at the Data Server in a centralized connection architecture. Such a feature can allow only the dissemination of 'enabled' collaborators' control data while all other control data is filtered. We see the implementation of this approach in situations such as the educational use-case scenario, where the online tutor can be the "host" enabling/disabling student control data transmissions.
2. Decentralized approach: In a decentralized approach each collaborator can filter other collaborators' actions by determining which of the control data streams will transmit to their local DAW instantiation.

In the above approaches the possibility of conflicting actions remains if more than one collaborator is "allowed" to transmit control data. We plan to investigate means to automatically identify conflict actions and resolve them, such as buffering recent control data and filtering out conflicting actions from different collaborators.

6 ACKNOWLEDGMENT

This research is supported by the Australian Government Research Training Program Scholarship.

7 REFERENCES

[1] V. R. Melchior, "High-Resolution Audio: A History and Perspective," *J. Audio Eng. Soc.*, vol. 67, no. 5, pp. 246–257 (2019 May). https://doi.org/10.17743/jaes.2018.0056.

[2] A. Pras and C. Guastavino, "Sampling Rate Discrimination: 44.1 kHz vs. 88.2 kHz," presented at the *128th Convention of the Audio Engineering Society* (2010 May), paper 8101.

[3] S. Stickland, N. Scott and R. Athauda, "A Framework for Real-Time Online Collaboration in Music Production," in *Proceedings of the ACMC2018: Conference of the Australasian Computer Music Association*, pp. 79–86 (Perth, Australia) (2018 Dec.).

[4] S. Stickland, R. Athauda and N. Scott, "Design of a Real-Time Multiparty DAW Collaboration Application Using Web MIDI and WebRTC APIs," in *Proceedings of the International Web Audio Conference*, pp. 59–64 (Trondheim, Norway) (2019 Dec.).

[5] Source Elements, "Source-Connect Pro Version 3.9," http://source-elements.com/products/source-connect/versions (accessed Apr. 15, 2018).

[6] Steinberg Media Technologies GmbH, "VST Connect Pro Version 5.5," https://new.steinberg.net/vst-connect/ (accessed May 14, 2021).

[7] Soundwhale, "Soundwhale: Audio Post and Music Collaboration Software," https://soundwhale.com/ (accessed Jul. 14, 2020).

[8] Audiomovers LLC, "Audiomovers Listento," https://audiomovers.com/ (accessed Jul. 8, 2020).

[9] Sessionwire Communications Inc., "Sessionwire," https://www.sessionwire.com/ (accessed Jul. 20, 2020).

[10] ConnectionOpen, "ConnectionOpen | Online Recording and Collaboration," https://www.connectionopen.com/ (accessed Jul. 27, 2020).

[11] N. Bouillot, M. Brulé and J. R. Cooperstock, "Performance Metrics for Network Audio Systems: Methodology and Comparison," presented at the *127th Convention of the Audio Engineering Society* (2009 Oct.), paper 7940.

[12] C. Chafe, "Tapping Into the Internet as an Acoustical/Musical Medium," *Contemp. Music Rev.*, vol. 28, no. 4–5, pp. 413–420 (2009 Aug.). https://doi.org/10.1080/07494460903422362.

[13] Production Expert, "Streaming Mixes From DAW to Web Browsers Using LISTENTO Plug-in by Audiomovers," https://www.youtube.com/watch?v=jlgYNOQnGnE (accessed Jul. 16, 2020).

[14] Avid Technology, Inc., "Producing Software for Music - Cloud Collaboration - Pro Tools," https://www.avid.com/pro-tools/cloud-collaboration (accessed Mar. 7, 2020).

[15] Steinberg Media Technologies GmbH, "VST Transit | Steinberg," https://www.steinberg.net/en/products/vst/vst_transit.html?et_cid=15&et_lid=22&et_sub=VST%20Transit (accessed Mar. 7, 2020).

[16] Spotify USA Inc / Spotify AB, "Soundtrap - Make Music Online," https://www.soundtrap.com/ (accessed Sep. 17, 2020).

[17] BandLab Technologies, "BandLab: Make Music Online," https://www.bandlab.com/ (accessed Sep. 9, 2020).

[18] Fraunhofer-Gesellschaft, "The AAC-ELD Family for High Quality Communication Services," https://www.iis.fraunhofer.de/content/dam/iis/de/doc/ame/wp/FraunhoferIIS_Technical-Paper_AAC-ELD-family.pdf (accessed Feb. 21, 2018).

[19] Xiph.Org, "Xiph.org Vorbis Audio Compression," https://xiph.org/vorbis/ (accessed Feb. 20, 2020).

[20] Source Elements, *Source-Connect Pro 3.9 User Guide* (Source Elements, Evanston, IL, 2016).

[21] Steinberg Media Technologies GmbH, *VST Connect 5.5 Operation Manual* (Steinberg Media Technologies GmbH, Hamburg, Germany, 2021).

[22] WavPack, "WavPack: Hybrid Lossless Audio Compression," http://www.wavpack.com/ (accessed Jul. 23, 2018).

[23] Xiph.Org Foundation, "FLAC: Free Lossless Audio Codec," https://xiph.org/flac/index.html (accessed Oct. 26, 2020).

[24] Steinberg Media Technologies GmbH, "Technologies | Steinberg," https://www.steinberg.net/en/company/technologies.html (accessed Jun. 17, 2020).

[25] Apple Inc., "What Is Core Audio?" https://developer.apple.com/library/archive/documentation/MusicAudio/Conceptual/CoreAudioOverview/WhatisCoreAudio/WhatisCoreAudio.html (accessed Jun. 16, 2020).

[26] Apple Inc., "AudioUnit | Apple Developer Documentation," https://developer.apple.com/documentation/audiounit (accessed Jun. 16, 2020).

[27] Avid Technology, Inc., "AAX Connectivity Toolkit," https://www.avid.com/alliance-partner-program/aax-connectivity-toolkit (accessed Jun. 16, 2020).

[28] wac2017 qmul, "Web Audio Conference 2017 (WAC2017 QMUL) Day 1 - Afternoon," https://www.youtube.com/watch?v=OpUeyRRPpCo&feature=youtu.be&t=957 (accessed Jun. 2, 2020).

[29] K. Brandenburg, "MP3 and AAC Explained," in *Proceedings of the Audio Engineering Society 17th International Conference: High-Quality Audio Coding* (1999 Sep.), paper 17-009.

[30] Soundwhale, "Soundwhale Open Source Components," www.soundwhale.com/opensource (accessed Jul. 14, 2020).

[31] Xiph.Org Foundation, "Opus Interactive Audio Codec," http://opus-codec.org/ (accessed May 28, 2018).

[32] N. Brock, M. Daniels, S. Morris and P. Otto, "Audio-Video Synchronization for Post-Production Over Managed Wide-Area Networks," presented at the *128th Convention of the Audio Engineering Society* (2010 May), paper 8040.

[33] N. Brock, M. Daniels, S. Morris and P. Otto, "A Collaborative Computing Model for Audio Post-Production," *Future Gen. Comp. Syst.*, vol. 27, no. 7, pp. 935–943 (2011 Jul.). https://doi.org/10.1016/j.future.2011.02.005.

[34] N. Brock, M. Daniels, S. Morris and P. Otto, "Long-Distance Uncompressed Audio Transmission Over IP for Postproduction," presented at the *127th Convention of the Audio Engineering Society* (2009 Oct.), paper 7945.

[35] M. Wenzel and C. Meinel, "Full-Body WebRTC Video Conferencing in a Web-Based Real-Time Collabora-

tion System," in *Proceedings of the IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 334–339 (Nanchang, China) (2016 May). https://doi.org/10.1109/CSCWD.2016.7566010.

[36] S. Petrangeli, D. Pauwels, J. van der Hooft, M. Žiak, J. Slowack, T. Wauters and F. De Turck, "A Scalable WebRTC-Based Framework for Remote Video Collaboration Applications," *Multimed. Tools Appl.*, vol. 78, no. 6, pp. 7419–7452 (2019 Mar.). https://doi.org/10.1007/s11042-018-6460-0.

[37] K. Fai Ng, M. Yan Ching, Y. Liu, T. Cai, L. Li and W. Chou, "A P2P-MCU Approach to Multi-Party Video Conference With WebRTC," *Int. J. Future Comp. Comm.*, vol. 3, no. 5, pp. 319–324 (2014 Oct.). https://doi.org/10.7763/ijfcc.2014.V3.319.

[38] R. A. Kirmizioglu and A. M. Tekalp, "Multi-Party WebRTC Services Using Delay and Bandwidth Aware SDN-Assisted IP Multicasting of Scalable Video Over 5G Networks," *IEEE Trans. Multimedia*, vol. 22, no. 4, pp. 1005–1015 (2020 Apr.). https://doi.org/10.1109/TMM.2019.2937170.

[39] S. Yoon, T. Na and H.-Y. Ryu, "An Implementation of Web-RTC Based Audio/Video Conferencing System on Virtualized Cloud," in *Proceedings of the 2016 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 133–134 (Las Vegas, NV) (2016 Jan.). https://doi.org/10.1109/ICCE.2016.7430552.

[40] Steinberg Media Technologies GmbH, "Cubase Pro Version 10.5.20," https://www.steinberg.net/index.php?id=14935&L=1 (accessed May 5, 2020).

[41] Cockos Inc., "REAPER Version 6.15," https://www.reaper.fm/download.php?from_reaper=1 (accessed Oct. 20, 2020).

[42] The MIDI Association, "MIDI 1.0 Detailed Specification," https://www.midi.org/specifications-old/item/the-midi-1-0-specification (accessed Feb. 21, 2020).

[43] M. Wright, "OpenSound Control Specification," https://web.archive.org/web/20030914224904/http://cnmat.berkeley.edu/OSC/OSC-spec.html (accessed Mar. 21, 2020).

[44] Google WebRTC Team, "WebRTC," https://webrtc.org/ (accessed Feb. 20, 2020).

[45] J. Postel, "User Datagram Protocol (RFC 768)," https://tools.ietf.org/pdf/rfc768.pdf (accessed Feb. 18, 2018).

[46] R. Stewart, ed., "Stream Control Transmission Protocol (RFC 4960)," https://tools.ietf.org/pdf/rfc4960.pdf (accessed May 30, 2019).

[47] M. Adeyeye Oshin, I. Makitla and T. Fogwill, "WebRTC Using JSON via XMLHttpRequest and SIP Over WebSocket: Initial Signalling Overhead Findings," in *Proceedings of the 9th International Conference on Web Information Systems and Technologies (IEEE WEBIST)* (Auchen, Germany) (2013 May).

[48] S. Shankland, "Google Hitches Opus Audio Technology to WebRTC Star," https://www.cnet.com/news/google-hitches-opus-audio-technology-to-webrtc-star/ (accessed Jun. 10, 2018).

[49] J.-M. Valin, G. Maxwell, T. B. Terriberry and K. Vos, "High-Quality, Low-Delay Music Coding in the Opus Codec," presented at the *135th Convention of the Audio Engineering Society* (2013 Oct.), paper 8942.

[50] World Wide Web Consortium, "Web MIDI API," https://www.w3.org/TR/webmidi/ (accessed Jul. 30, 2018).

[51] Google LLC, "Chromium OS - The Chromium Projects," https://www.chromium.org/chromium-os (accessed Oct. 26, 2020).

[52] World Wide Web Consortium, "WebRTC 1.0: Real-Time Communication Between Browsers," https://www.w3.org/TR/webrtc/ (accessed Oct. 26, 2020).

[53] Frozen Mountain, "Massively Flexible Video, Voice, & Messaging | Frozen Mountain," https://www.frozenmountain.com/products-services/liveswitch/ (accessed Mar. 28, 2020).

[54] Steinberg Media Technologies GmbH, "Compare the Versions of Cubase," https://new.steinberg.net/cubase/compare-editions/ (accessed Jun. 3, 2020).

[55] C. Cunningham, "How to Build the Best PC for Music Production and Audio Editing," https://www.logicalincrements.com/articles/build-pc-music-production-audio-daw (accessed Jun. 3, 2020).

[56] L. H. Campbell, S. Suessspeck and K. Hinton, "The National Broadband Network: What Difference Will It Make to Broadband Availability in Australia?" *Aust. J. Telecomm. Digit. Econ.*, vol. 6, no. 1, pp. 1–25 (2018 Mar.). https://doi.org/10.18080/ajtde.v6n1.141.

[57] M. A. Gregory, "How to Transition the National Broadband Network to Fibre to the Premises," *Aust. J. Telecomm. Digit. Econ.*, vol. 7, no. 1, pp. 57–67 (2019 Mar.). https://doi.org/10.18080/jtde.v7n1.182.

[58] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins and Y. Xu, "VP8 Data Format and Decoding Guide (RFC 6386)," https://tools.ietf.org/pdf/rfc6386.pdf (accessed Mar. 5, 2020).

[59] T. Levent-Levi, "The Challenging Path to WebRTC H.264 Video Codec Hardware Support," https://bloggeek.me/webrtc-h264-video-codec-hardware-support/ (accessed Mar. 4, 2020).

[60] The WebM Project, "The WebM Project | VP8 Encode Parameter Guide," https://www.webmproject.org/docs/encoder-parameters/ (accessed Mar. 29, 2020).

[61] J.-M. Valin, K. Vos and T. B. Terriberry, "Definition of the Opus Audio Codec (RFC 6716)," https://tools.ietf.org/pdf/rfc6716.pdf (accessed May 29, 2018).

[62] Frozen Mountain, "JavaScript - LiveSwitch: Working with Data Channels," https://help.frozenmountain.com/docs/liveswitch/clients/javascript#WorkingwithDataChannels (accessed Mar. 24, 2020).

[63] The MIDI Association, "Details about MIDI 2.0TM, MIDI-CI, Profiles and Property Exchange," https://www.midi.org/midi-articles/details-about-midi-2-0-midi-ci-profiles-and-property-exchange (accessed May 14, 2021).

[64] Steinberg Media Technologies GmbH, "About MIDI in VST 3 - VST - Steinberg Developer Help," https://developer.steinberg.help/display/VST/About+MIDI+in+VST+3 (accessed Oct. 30, 2020).

THE AUTHORS


Scott Stickland


Dr. Rukshan Athauda


Nathan Scott

Scott Stickland is a fourth-year PhD (Music) candidate in the School of Creative Industries at The University of Newcastle (UoN), Australia. He previously completed a Master of Music Technology (UoN) and Bachelor of Education (Sec) – Music (Melbourne) and taught and coordinated music programs in Australian secondary schools for 16 years. Scott has presented papers at the Australasian Computer Music Conference in 2018 and 2020 and the Web Audio Conference in 2019 since commencing his PhD. He currently teaches audio and music production through his business, *Monty Sound Production*, and plays keyboards in the Australian touring band, *Cool Change – The Ultimate Tribute*.

•

Dr. Rukshan Athauda is a Senior Lecturer in the School of Electrical Engineering and Computing at The University of Newcastle (UoN), Australia. Dr. Athauda's research interests span Database Systems, Technology-Enhanced Learning, Cloud Computing, and IT Security. Dr. Athauda has published over 60 peer-reviewed research articles internationally. He has supervised four PhD completions at UoN and also undertaken a number of administration roles, including Head of Discipline and Program Convenor. Prior to joining UoN, Dr. Athauda worked at Microsoft Corporation, USA; the High-Performance Database Research Center at Florida International University, USA; and the Sri Lanka Institute of Information Technology, Sri Lanka.

•

Nathan Scott is a Lecturer in the School of Creative Industries at the University of Newcastle, Australia. He has interdisciplinary research interests spanning creative arts, technology, science, health, and education. Nathan has presented and performed internationally and published in the areas of music, technology, education, gaming, and the human voice. He has presented workshops in regional NSW (2003) and developed an online international postgraduate program supporting the use of technology in music contexts. He participated in the CHASS Expanding Horizons forum in Canberra (2006) and undertook a sub-project as part of an ALTC National Teaching Fellowship (2010).