Distributed Audio Mixing Network (DAMN)

Johannes Giani[1], Masih Jakubi[1], Dominik Litfin[1], Andreas Rehbein[1], Johannes Röhn[1]
[1]*University of Applied Sciences Cologne, Germany, Email: johannes.giani@googlemail.com*

Abstract

Networked digital audio mixing consoles and systems have emerged over the last years and experienced perpetual penetration of the professional audio market. Despite the observation that interconnected IT-Solutions are influencing today's audio products, all available mixing consoles share the same principle: central signal processing. In centralistic approaches all the audio signals are transferred via proprietary or standardized transport technologies to one central node within the network environment. At this central point all signal processing takes place. The idea behind the university project Distributed Audio Mixing Network (DAMN) is to erupt such boundaries. The result is an operational prototype system that is decentralized, meaning that processing of audio signals shifts away from one central node towards the edge nodes of the audio network. This enables enormous potential not merely in the matter of scalability. Systems can easily be extended as the processing capacity grows with the number of nodes which hold the audio I/O available. DAMN also provides an intuitive GUI that is accessible through any Web-Browser and is optimized for touch-enabled devices. The GUI gives the user the ability to control the system and to mix the audio signals using the flexible channel concept of the DAMN system. DAMN reveals exemplarily how tomorrow's mixing systems can look like - distributed, flexible and scalable.

1. Introduction

The distributed audio mixing network (DAMN) is a prototype developed by five students of the University of Applied Sciences Cologne, Germany. The project was part of the masters degree program in media technology.

In order to come up with our idea of a distributed, scalable and flexible audio mixing system we first had to analyse how todays systems work. We considered a typical setup for live audio production which commonly consists of the following devices: A mixing console, Stageboxes with I/O functionality and a network that provides the infrastructure for signal transportation. These systems follow a *centralized* approach, meaning that the stageboxes feed an unprocessed "raw" signal to the console (or its external processing unit), which returns the processed signal back to the output interface. In short, all the signal processing and control takes place inside the console. Upgrading the DSP and/or I/O capabilities is a very costly affair, and even if there are no financial limitations, there are certainly hardware restrictions. In addition, such systems very often suffer from inflexible signal processing and routing, which means one cannot choose between e.g. different EQs or change their order in the signal flow. Furthermore it is not possible to built complex signal chains because of constrained chaining options. Finally the broad majority of consoles are designed as a single operator device, making cooperative work practically impossible.

Besides this classic approach of system devices and processing, a new trend regarding signal infrastructure has emerged. A rising number of companies are switching over from baseband technology towards IT networking solutions. In our opinion this is a promising way to overcome the restricted scalability and flexibility we just described. By connecting all system devices over a network and using low latency, real-time capable protocols for signal transport and control messages, it is possible to distribute the processing power over all the nodes in this network, practically *decentralizing* the mixing system. Modern networking technology also allows for much more scalable systems, as it can easily be expanded by adding new switches and routers. With these advantages in mind, our aim was to built a working prototype of a distributed audio mixing network that showcases its possibilities, limitations and potential future applications.

In the following chapters we will introduce our concept for the system, as well as the resulting prototype and required technical solutions, before summarizing the results and pointing out prospects for further research.

2. Concept

In this chapter we will take a closer look on the basic concept of our mixing system. An important part of the project was to provide flexible audio processing and routing. To reach this goal, we came up with a new channel concept that differs from the more traditional approach in most other systems. Whereas normally the operator has to choose between different channel types like Inputs, Auxes or Master channels, there is no such categorization in our approach. A DAMN-channel is designed to be multi-purpose, meaning that it can serve as all of the types mentioned above and it even may have several in- and outputs at the same time. As illustrated in Figure 1, such a channel consists of three sections: The *input*-section mixes all connected inputs together. If multiple inputs are connected, every delay and level parameter of the incoming signals are being adjusted to ensure phase accurate mixing. The resulting signal will then be processed by a maximum amount of four DSP-Plugins per channel (*DSP*-section). The operator can choose between an equalizer, a limiter, a compressor and an expander in no fixed order, meaning that after adjusting all parameters it is

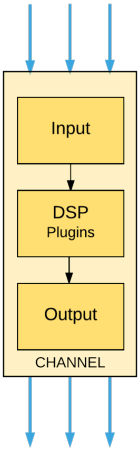

Fig. 1: The DAMN-channel concept consists of three main sections: Input, DSP-Plugins and Output.

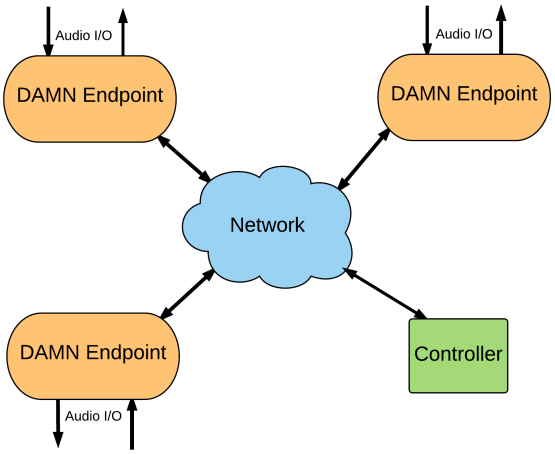

Fig. 2: The DAMN Architecture and its components

still possible to change their arrangement in the signal flow without loosing any settings. At the end, the *output*-section provides connectivity to one or more desired destinations. An adjustment of the signal level at this point is not required, as every new destination provides this functionality as part of its *input*-section.

The following subsections will be discussing function and structure of the three main components: The *Network*, the *DAMN-Endpoint* and the *Controller*. They are illustrated in figure 2.

2.1. Network

All the nodes in our decentralized system are connected via a network with special requirements. One requirement for the network is the capability for the transport of all the relevant signals via a single cable. The connection follows a star-topology for maximum flexibility with a switch in the middle to which all nodes are connected. Open standards ensure transparency and future compatibility with other systems. Most importantly, the network had to be capable of real time audio transport and control for multiple users, which also includes the knowledge of all signal propagation delays at any time in the network to ensure phase accurate mixing.

2.2. DAMN-Endpoint

The DAMN-Endpoint is a device which offers necessary functionality to form the decentralized system together with other devices of its kind. Located on the edge of the network, it provides analogue audio in- and output interfaces with pre-amplification of the input signals, analogue to digital conversion (and vice-versa), level adjustment as well as the possibility to delay the output signals. It is also responsible for signal processing, which is a major benefit. By combining I/O and DSP functionality on the same device we ended up with a very extendible approach, meaning that adding a DAMN-Endpoint to the network will not only provide more physical I/O to the user, but at the same time increase processing capabilities of the whole system. To efficiently manage the resources for internal signal transportation between the endpoints, they are also able to mix signals together in order to combine them in one stream for transmission. This proved far more effective than streaming every signal separately.

2.3. Controller

The controller can be considered as the brain of the DAMN system. It had to be flexible and platform independent, thus we decided to realize it as a web-application using modern web technology standards. It is designed for wireless use on a tablet computer but also works great on any device with a modern web browser and sufficient screen size. By adding multitouch-functionality, it is able to simultaneously control multiple faders at the same time, just as any hardware mixing console. It is responsible for user interaction and network management. Therefore, the application logic was implemented in the controller to efficiently manage DSP and streaming resources of all nodes in the network. The decision which resources are being used is based on several different parameters:

1. *Physical input and output*
   A smart choice of the physical input and output of a signal can minimize its propagation delay in the system and avoid unnecessary streaming of signals from one endpoint to another.

2. *Resource usage*
   Resource usage in the system can differ depending on the current configuration, which has to be taken into account.

3. *Resource availability*
   If there is no more capacity left on one endpoint, the system will try to choose another endpoint in the network to process the signal. In case this is not possible, the user will be notified.

The application logic is also responsible for the compensation of all signal delays that occur during transport and processing within the system. The possibility of setting a separate delay on every endpoint in the network is used to ensure phase accurate mixing and synchronized signal output.

3. Prototype

The exemplary implementation of the DAMN concept lead to a working prototype system which verifies the approach in terms of a proof of concept. The prototype system is separated into three parts as described in section 2 namely the network, the endpoints and the controller. The following subsections are going to describe all of the three implementations in greater detail and will gfive insights into occurred problems and obstacles.

3.1. Network

The network part of the DAMN concept is the central communication point and accomplishes several tasks. At first it should transport uncompressed audio data between the DAMN-endpoints of the system in real time. Secondly, it must manage the control data for several platform-independent users. These points lead to two key components of the DAMN-network, which are the real-time audio network and the server. These are going to be described subsequently.

3.1.1. Real-Time Audio Network

The utilization of a real-time audio standard derives from the demands the interface has to meet. It must carry sensitive real-time audio as well as control data whilst remaining scalability. State of the art digital audio interfaces such as MADI are not able to handle all of these requirements and are inflexible in terms of the usable topology.
The Audio Video Bridging (AVB) standard by the Institute of Electronics and Electrical Engineers (IEEE) fulfills the required capabilities while being of low costs (license free). In spite of that, alternative technologies like Audinate *Dante* or Alc NetworX *RAVENNA* would also be eligible, but there was a lack of available development boards for the prototyping purposes of this limited university project.
AVB guarantees a deterministic delay of 2 ms over up to seven network hops between a source (AVB talker) and a sink (AVB listener) by enhancing the widely used Ethernet (IEEE 802.3) standard [1]. The three AVB extensions adapt the OSI layer 2 of standard Ethernet for synchronization (gPTP; IEEE 802.1AS), quality of service (FQTSS; IEEE 802.1Qav) and fixed bandwidth reservation mechanisms (SRP; IEEE 802.1Qat), as well as an integrated OSI layer 2 transport protocol IEEE 1722 [2]. Another aspect of the AVB standard are the integrated AVDECC control protocol, which is also used in the DAMN prototype [3].
In contrast, the main drawback of AVB are the special requirements to the hardware, which lead to dedicated AVB-capable switches and endpoints. In case of the DAMN prototype, the 5 port Fast Ethernet AVB switch of DSP4YOU (*AVB-SW* [4]) was used, which was totally sufficient for the prototyping purposes. The special endpoints hardware will be described in section 3.2.

3.1.2. Server

The server component of the network is responsible for the DAMN control management. It consists of a central server device which processes several tasks. On the one hand, it has to deliver the web content to the clients. On the other hand, it has to manage all the communication between multiple DAMN-endpoints and possible several controller instances.

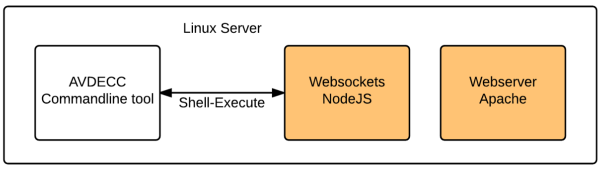

Fig. 3: DAMN Server component

In terms of the prototype system the server utilizes a virtualized Linux distribution on which the two server instances run. An Apache webserver delivers all the content needed for the client device to display the GUI for mixing. Furthermore, the application logic is served via HTTP as part of the GUI. The central communication server is realized by a *Node.js* server. *Node.js* is a JavaScript based server approach for the handling of multiple parallel connections in a real-time context, which is required in the DAMN system. The server deploys the *WebSockets* protocol for low latency communication with the controller devices using low overhead HTTP communication via the UDP transport protocol and JSON data elements. [5]
Besides, the communication between the server and the DAMN-endpoints is realized via the AVB integrated AVDECC protocol. The server translates all commands received via the *WebSockets* interface to the AVDECC layer by shell-execution of the *avdecc-cmd* command line tool by J.D. Koftinoff [6] and the other way around. This open-source tool was extended in terms of the DAMN command types, so the special vendor defined AVDECC GET and SET control payloads could be executed and evaluated. The response is always sent back to the responsible controller device via the JSON data type. The basic parts of the server are described in figure 3.
The final part of the network consists of a WLAN router device which enables wireless access for e.g. touch devices such as an Apple *iPad*. The router additionally handles the DNS resolution of the HTTP webserver.

3.2. Endpoints

The endpoints of the DAMN system are responsible for the actual decentralized audio processing in terms of DSP as well as in- and output of audio signals. The flexible channel concept of DAMN is realized here. All of the endpoints are inter-connected via the real-time AVB network for low latency streaming between all endpoints. Endpoint control is realized via AVDECC messages as described above.

3.2.1. Development Boards

The implementation of the endpoint device in terms of the DAMN prototype is based on XMOS *AVB Audio Endpoint [7]* development boards. These offer basic AVB capabilities such as two unbalanced analog in- and outputs and AVB streaming of up to eight channels in one AVB listener respectively talker stream. The provided open-source firmware written in C and

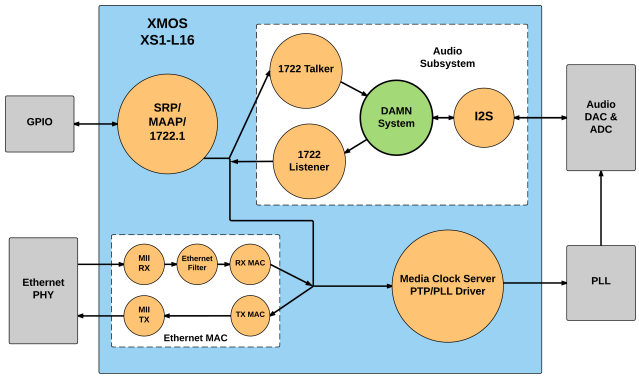

Fig. 4: DAMN Endpoint High Level System Architecture

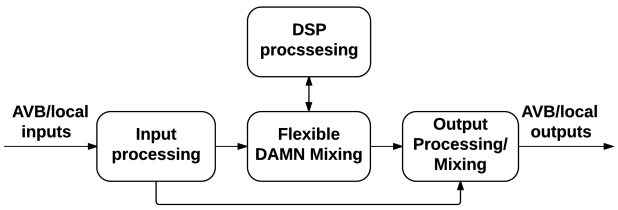

Fig. 5: DAMN Endpoint processing

the special XMOS *XC* language already implement the basic functionalities needed for streaming and control while still being expandable. The XMOS *XS1* dualcore multithread DSP microprocessor provides spare processing power and memory which is used for DAMN system purposes. Figure 4 shows an overview of the extended DAMN firmware. The DAMN parts of the implementation are located between the I2S component and the AVB listener and talker as part of the audio subsystem.

3.2.2. DSP Implementation

The DSP implementation pursues different targets. On the one hand it has to manage the flexible mixing concept of DAMN. On the other hand it should offer DSP processing for signal manipulation. Unfortunately, the DSP implementation was highly dependent on the processing power of the XMOS *XS1* DSP, which had several limiting issues in terms of available memory.

The DSP implementation is based on a parallel multi-threaded architecture on the XMOS multi-core processor. Figure 5 illustrates the DAMN-endpoint processing. At the input processing stage, all the needed delays for in-phase mixing are applied to the local or AVB input samples. In the next step, they are handed over to the central DSP mixing process where the flexible mixing and routing concept of DAMN is realized. Depending on the current configuration, each of the processed DAMN channels may have several inputs like AVB or local inputs or even other DAMN channels. These signals are all mixed together into one output signal per channel which can be delayed as well. For each of the channels, the DSP processing is triggered in a separate parallel process. Furthermore, the output processing applies additional processing such as delays before the samples are sent through the network or local outputs. The AVB output channels of the endpoints also received separate mixing capacity to reduce the amount of totally used AVB channels, as there are only eight available per prototype device.

Overall the implementation was limited by the small amount of memory of only 64 KB per core. A second limitation was the full occupancy of the available 16 threads on the XMOS chip. The memory had to be allocated dynamically to host all of the required variables. Furthermore, the AVB playout buffer had to be reduced from 2 ms to only 0.5 ms to handle all the data. The limited processing power results in a maximum number of processable channels per endpoint of eight and a maximum amount of 72 inputs. The maximum delay was defined to 833 μs for AVB inputs respectively 1.3 ms for local inputs and DAMN channels.
The following DSP algorithms have been implemented: A four band EQ, a compressor, a limiter and an expander. The number of DSP units per channel was defined to four. In total, up to six EQ, respectively 13 dynamics DSP algorithms can be processed. The EQ implementation is based on IIR Biquad filters.

3.2.3. Control

The control of the endpoints including the DSP and mixing processing was realized via the AVB integrated AVDECC protocol. These were already implemented in the provided firmware and had to be extended by special DAMN commands.
The AVDECC protocol is made up of three sub protocols: ADP (discovery), AECP (enumeration and control) and ACMP (connection management). The ADP is responsible for the detection of AVB capable devices in the network and ACMP manages the connections between these. Moreover, the AECP protocol defines how the functional structure of AVB devices is described by the AVDECC Entity Model (AEM). This model describes an endpoint in detail based on the so called descriptors for each of the functions e.g. volume control of a signal or stream input/output port. Additionally, AECP defines how these properties/descriptors can be controlled and modified via dedicated commands.
For the DAMN prototype we used vendor defined CONTROL descriptors to represent the current DSP and mixing configurations of each endpoint. CONTROLS were defined for e.g. a mixer/channel structure of an endpoint or a DSP unit. This CONTROL can be modified and read out via dedicated GET and SET CONTROL commands with special payload. This payload is built as well as evaluated in a dedicated application task of the XMOS firmware. The process has direct on-the-fly access to the DSP and mixing variables to modify the current audio processing without any delays.
Furthermore, the ADP DISCOVER functionality is used by the controller to detect available DAMN devices in the network. The ACMP messages are used without any modifications to connect the AVB streams of the DAMN endpoints with each other for the streaming of real-time audio.

3.3. Controller

The controller section of the DAMN concept is the most important part as it manages and controls all of the DAMN endpoints via the network. It can be considered as the ”brain” of the DAMN system and it is possible to have multiple instances. Furthermore, the GUI also played a major role during the implementation as it needs to provide an easy way of handling the flexible channel concept on multi-touch enabled devices. As the implementation of the controller was devided into two main parts, subsequently the GUI and the application logic, our web application will be presented in detail.

3.3.1. Graphical User Interface

The GUI of the DAMN-controller is the visible interface to the user. It does not reveal the decentralized aspect of the DAMN system and is usable as a flexible and scalable mixing application. Nonetheless, it is roughly based on a traditional mixing consoles interface to provide instant familiarity and quick access to the user. The GUI employs the advantages of a software-based interface in combination with touch-sensitive devices like tablet computers to improve the usability of the whole system. Because of the vast amount of different screen sizes, the application is designed to adapt itself graphically. Nevertheless we recommend a minimum screen size of 7.9 inch (e.g. Apple iPad mini). Furthermore, the GUI supports multitouch gestures for simultaneous handling of multiple graphical elements. This improves usability especially when mixing audio signals with the faders and adjusting equalizer parameters. For non-mulitouch enabled devices the handling has been adapted accordingly, but note that a simultaneous operation of the faders will not be possible.

The GUI’s job is to provide the control interface and display the current state of the system graphically. This can be further categorized into the following tasks:

- Display and control of in- and outputs of a channel
- Display and control of a channel’s DSP
- Display and control of mixing functionality of a channel
- Display of the current signal level of a channel

To provide a functional and clear usage we defined several further requirements for the GUI: It has to display the faders in a reasonable size at any time during operation, which is why the maximum amount of faders displayed depends on the screen size. The operator must be able to add and delete channels dynamically, rename and color-code them. The displayed channels and all of their functions must always be selectable. Moreover the exact numerical volume must be visible at any time.

Operation The GUI’s main window is divided into three main sections: The selected channel on the left, the channel strip area in the middle and the control-panel on the right. Depending on the system’s state the channel strips will be further divided into an upper and lower part, as displayed in figure 6:

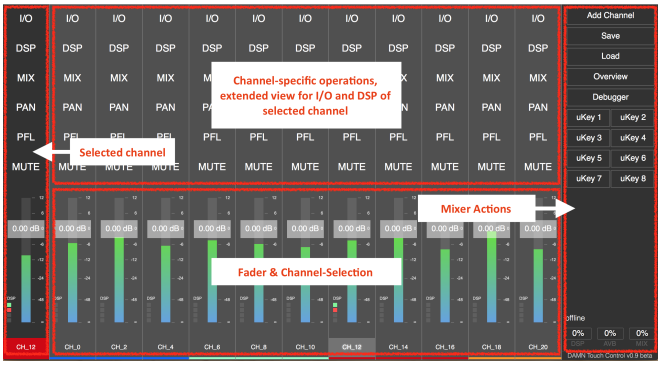

Fig. 6: Partitioning of the GUI into four areas

Each of the channel strips provides control of the following functionalities:

- I/O: Manage in- and outputs of the channel
- DSP: Add, remove and adjust signal processing algorithms
- MIX: Manage the level of all the channels inputs
- PAN: Control the channel’s panning (not implemented in the prototype)
- PFL: Pre-fader listening (not implemented in the prototype)
- MUTE: Mute the channel
- FADER: Control the level, display metering
- SELECT: Select the channel and show its parameters

The operator can either select a channel by tapping on its name field on the bottom, or directly tapping on one of the described software buttons which will instantly display the desired function and trigger the channel selection at the same time for convenience. The current selected channel is always displayed on the left of the screen for quick access, whereas the channel strip area can be moved left and right by horizontal swiping. Furthermore, the GUI offers several views for different control scenarios like the flexible mixing, DSP customization or input/output routing of a channel. Figure 7 shows the DSP-view on top of the faders, which remain movable. The control-panel on the right hand side provides buttons to add a channel, save and load configurations, display an overview of all current system parameters, open a debugger window with system alerts and eight user defined ”uKeys” which are fully configurable. In addition, this panel provides three indicators for system resources, namely the currently used DSP, AVB and MIX capacities in percent.

Implementation As the controller had to be platform independent, the application needs to work with any modern web browser. Thus, it was realized as a web application based on HTML5, JavaScript and CSS3. Because the DAMN mixing system consists mainly of dynamic elements, the static HTML index page only provides three div-containers for the main sections described in this chapter and includes all required CSS and JavaScript files, which generate all other components dynamically during runtime. For the complex

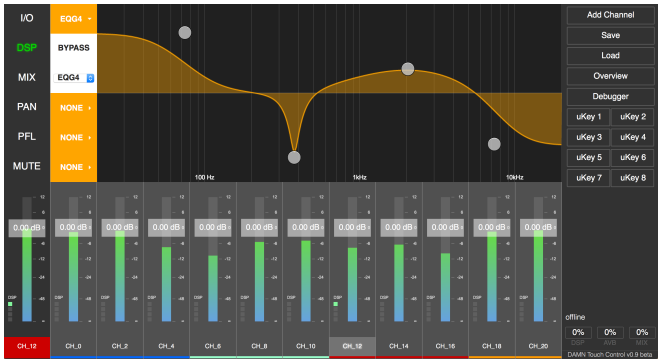

Fig. 7: DSP-view with active equalizer

GUI constructions we used the *jQuery* library and to add multitouch functionality on touch devices the *Hammer.js* JavaScript library was utilized. [8] [9] Finally, the whole logic was also integrated into the web application, which executes the actual commands on the network involving the server and the endpoints.

3.3.2. Logic

The application logic of the controller is the actual brain of the system and knows all the states of the available endpoints in the system. It enables the interaction of the endpoints and always keeps the current latency and resource usage in mind. The following responsibilities have to be managed by the logic:

- Connection of the different endpoints and AVB streams
- Resource optimized placement of the channel processing on the DAMN endpoints
- Execution of commands to modify channel parameters such as volume and DSP
- Delay management

The implementation was realized entirely in object based JavaScript language as part of the web application. Furthermore, the most complex task of the logic turned out to be the correct assignment of the channel processing to a specific DAMN-endpoint. This decision depends on several factors such as the number of inputs, the main output and the remaining DSP capacities of the endpoints. Hence, the utilized resources such as the number of AVB streams and the resulting latency of the system should be minimized. Therefore, a dedicated function checks all possible setups and chooses the best one to be used. This function also gathers all relevant information to build a list of parametrized commands, that will be executed sequentially. Those commands trigger any action needed to build a channel, e.g. connection establishment. When every command of the list is executed successfully, the new channel is created and can be operated.

The delay management also was part of the logic as the delays of signals in the system should be compensated and self regulatory without user interaction. For this purpose the following steps are processed when a new channel is added or the setting of an existing channel is edited:

1. Calculation of channel flows (flexible channel concept)

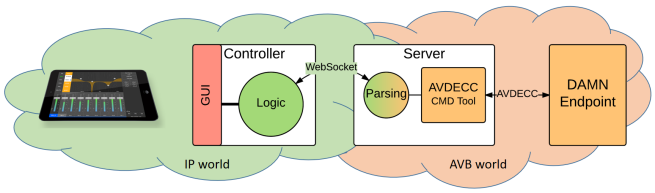

Fig. 8: DAMN control flow

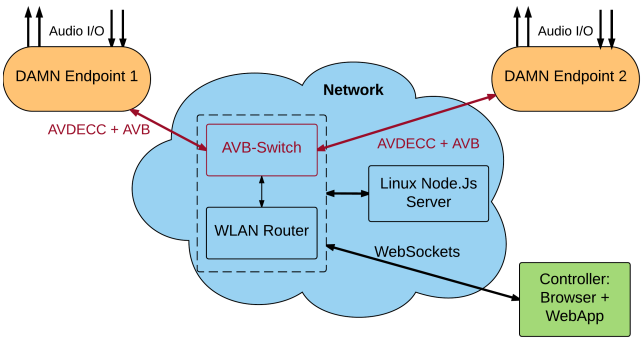

Fig. 9: DAMN prototype architecture

2. Calculation of the minimal total delay
3. Calculation of the delay of physical outputs
4. Setting of the calculated delays on the DAMN-endpoints

The execution of the commands such as volume control or channel assignment is realized via JSON objects which are sent through the *WebSockets* interface to the server. The server then parses and translates the received commands and interacts with the DAMN-endpoints by using OSI layer 2 AVDECC commands. The response of the endpoints is then translated back into JSON objects and sent to the controller and so on. The whole control flow of the DAMN system is shown in figure 8.

To sum up the implementation, figure 9 shows an overview of the finished prototype and the used technologies. The most significant problems were the limitation of the development boards in terms of memory and processing power, which limited the maximum number of channels and inputs. Additionally, the huge amount of AVDECC messages which have to be sent and evaluated by the server and controller were critical, but this problem could be solved.

3.4. System delay measurements

This section lists the results of our system delay measurements. Every measurement was performed several times to ensure significant test results.

The system delay test was performed as follows: Via a DAW-Software we send a unit-impulse function into the local input of a DAMN-Endpoint. Here the local input was forwarded directly and without mixing functionality to a local output. The output signal was recorded in the DAW-Software and the delay was measured. We used the RME Fireface UC audio interface. This procedure was repeated with the signal passing a DAMN channel as well as streaming it via the AVB-network. The result showed 24 samples (500 μs) delay when sending the signal directly from local in- to output and

27 samples (562.5 μs) when sending it through a DAMN-channel. This means that the channel produced 3 samples of delay. When streaming the signal via the AVB-Network the result showed an average delay of 50 samples (1041.6 μs), which equals a delay of 25 samples (520.8 μs) caused by AVB. These delays were measured without the usage of the delay compensation of the system, which would compensate all delays resulting in the same delay for every signal.

4. Conclusion

This paper's subject was the conception and the prototyping of a distributed audio mixing system based on network technologies abbreviated as DAMN. Mainly it was supposed to show the potential of a *decentralized* and highly scalable approach in contrast to almost all state of the art *central* mixing consoles with fixed capacities. Additionally, a flexible signal processing and routing as well as a multi-user concept were in the scope of this project.

At the beginning, the new DAMN concept was presented and the three parts, namely the endpoint, the network and the controller were introduced. The DAMN-endpoint handles the decentralized digital signal processing as well as in- and output of the actual analogue signals. The central scalable network part manages the low latency audio connections as well as control traffic through a star topology. At last, the controller device is responsible for the user interaction and the network management while being platform independent and multi touch enabled.

The exemplary implementation was based on XMOS development boards using the AVB low latency standard. These accomplish flexible signal processing and routing in the firmware, allowing a maximum of eight channels per device. The network was based on a server/client topology involving a Linux *Node.js* server which interacts through WebSockets with the controller. Furthermore, it communicates with the endpoints via AVDECC. All of the devices are interconnected per a central AVB 5 port switch plus a WLAN router for wireless control. Finally, the controller device was implemented in a multi-touch optimized web-application which can be executed on any current web browser.

The DAMN project showed that a decentralized approach of a mixing system is functional. It has to be said though, that the implemented prototype consisted of just two low cost AVB endpoints due to budget limitations. Nevertheless, it revealed the feasibility of a scalable mixing system based on a real-time network. The scalability enables users to flexibly choose the desired setup and to be independent from fixed mixing consoles, which finally safes expenses. Furthermore, the introduced flexible channel concept enables the user to use a universal channel for all kinds of purposes by simply modifying its properties. In addition, the innovative web application GUI makes it possible to dynamically control the system with multiple control instances and offers a high level of comfort through multi touch support.

Further developments should prove the concept with more than two endpoints and one switch, which in theory is possible with the current implementation. The system should be tested in a more practical environment as the concept is eligible for live or studio context. Besides, a more powerful hardware should be used to erupt the limitations in signal processing. The PFL and panning functions have not been implemented and should be realized. The functionality of the server may be moved into the endpoints, effectively decentralizing it as well. This would also have positive effects on redundancy. For a more user friendly haptic control of the system, the integration of a hardware controller would be desirable. Other relevant aspects are: User administration, an open plug-in architecture and the future of the AVB technology, which will be succeeded by TSN. Nevertheless, DAMN does not dependent on a specific network technology, as the concept can be translated to any real-time capable audio network technology.

5. References

[1] Hyung-Taek Lim, Daniel Herrscher, Martin Waltl, and Firas Chaari. Performance analysis of the ieee 802.1 ethernet audio/video bridging standard. In George Riley, Francesco Quaglia, and Jan Himmelspach, editors, *Fifth International Conference on Simulation Tools and Techniques*.

[2] Microprocessor Standards Committee of the IEEE Computer Society. Ieee std 1722-2011, ieee standard for layer 2 transport protocol for time-sensitive applications in bridged local area networks.

[3] Microprocessor Standards Committee of the IEEE Computer Society. Ieee std 1722.1™-2013, ieee standard for device discovery, connection management, and control protocol for ieee 1722™ based devices.

[4] DSP4YOU. Product brief - avb-sw. URL: http://www.dsp4you.com/downloads/Product%20Brief%20-%20AVB-SW.pdf.

[5] Peter Leo Gorski, Luigi Lo Iacono, and Hoai Viet Nguyen. *WebSockets: Moderne HTML5-Echtzeitanwendungen entwickeln*. Hanser, München, 1. aufl. edition, 2015. URL: http://ebooks.ciando.com/book/index.cfm/bok_id/1835005.

[6] Jeff Koftinoff. avdecc-cmd. URL: https://github.com/jdkoftinoff/avdecc-cmd.

[7] XMOS Ltd. Avb-audio-endpoint-product-brief_1.5. URL: http://www.xmos.com/support/boards?product=14769&component=14430.

[8] Jorik Tangelder Alexander Schmitz, Chris Thoburn. Hammer.js javascript library. URL: https://hammerjs.github.io/.

[9] The jQuery Foundation. jquery javascript library. URL: https://jquery.com/.