AN AUDIO SYSTEM APPLICATION FOR THE ADAPTIVE AVIONICS PLATFORM

M. Brunner, B. Annighöfer, University of Stuttgart, Institute of Aircraft Systems, Pfaffenwaldring 27, 70569 Stuttgart, Germany

Abstract

This paper elaborates and validates an audio system application for an Adaptive Cabin Management System (ACMS) based on the Adaptive Avionics Platform (AAP).

A modern approach for avionics is the self-organizing and almost configuration-free AAP. System functions are configured and instantiated without human interaction during an organization phase of the system. A domain where the benefits of the AAP seem most promising are cabin management systems (CMS), where frequent cabin layout changes and updates occur. The AAP offers scaling and modification of CMS with hardly any configuration effort. This opens up the perspective of reusable systems for different types of aircraft making the need for an exclusive system design for a particular aircraft obsolete. The platform ensures fault-tolerance via reconfigurable network paths and master/slave behaviour of computing modules (CPM).

In this paper, a cabin audio system for the AAP is presented. The audio system uses cabin interphones (CIP) and cabin speakers (CS) as peripherals to support three typical scenarios: (1) CIP to CIP calls, (2) CIP to CS calls (passenger address) and (3) playback of recorded audio using the flight attendant panel (FAP). The audio system is modifiable and scalable due to its integration into the AAP. Therefore, it can be used in different cabin layouts without the need for hardware configuration. The integration also opens up the perspective for saving weight and resources for development and maintenance of the cabin functionalities.

The validation of the audio system concept and its performance is based on an ACMS lab demonstrator and showed that time delay requirements are fulfilled. Time difference requirements for synchronous playback were not met, but comparison with other approaches showed that clock synchronization overcomes this shortcoming.

Keywords

adaptive avionics; self-configuration; avionics platform; cabin management systems; cabin audio;

1. INTRODUCTION

The number of integrated cabin functions and their complexity rise steadily [1]. This trend is accompanied by high development and configuration efforts, especially, if frequent cabin layout updates occur during an aircraft's lifespan. The Adaptive Avionics Platform (AAP) addresses these trends by enabling an almost configuration-free distributed avionics system.

The system topology of an AAP instance can be built almost-freely from generic hardware modules. Peripherals can be connected anywhere in the system and are discovered by the AAP automatically. Failure management and redundancy are transparently provided to system functions by the AAP. System functions concentrate on their core function and are relieved of management tasks.

The AAP allocates the system functions depending on the discovered topology.

One of the key attributes of the AAP is the possibility to reuse system functions for various system topologies. In this way the AAP increases the modifiability of systems and scalability of system functions. Both attributes aid to make cabin management systems (CMS) more flexible. Frequent changes in the cabin layout and the integration of new system functions are easily achieved.

One of the most demanding tasks of today's CMS are the provision of audio capabilities. Typical audio capabilities comprise the playback of recorded audio, announcements in the cabin via cabin interphones and cabin intercommunication between cabin interphones. In modern aircraft audio capabilities are provided by cabin intercommunication data systems (CIDS) using a dedicated network [1, 2]. Aside from the audio capabilities, the CIDS provides the cabin smoke detection, which is a safety-related system function. The separation of the CIDS from other cabin functionalities is due to the relatively strict timing requirements of the audio capabilities as well as the incorporation of the safety-related smoke-detection [2]. The time delay for announcements between microphone and speaker should be less than 100 ms to ensure lip-synchronous announcements [3]. The audio playback of speakers should be synchronous to a maximum time difference of 1 ms to avoid incomprehensible announcements [3]. In switched networks that are shared among cabin functionalities, audio video bridging has been successfully employed to meet these requirements [4].

The AAP targets the almost configuration-free integration of multiple system functions into a distributed system. In this paper, an audio system application is presented as a system function for the AAP. The audio system application uses two distinct peripheral types: cabin interphones (CIP) and cabin speakers (CS) to provide the typical audio capabilities. The audio peripherals are self-descriptive and can be automatically discovered during the organization phase.

The audio system's functionalities are split into different applications, so that the AAP can distribute them appropriately to the discovered topology. This allows the

audio system to become modifiable and scalable due to the self-configuration abilities of the AAP. Therefore, the audio system can be used in different cabin layouts and scenarios. The audio system operates on the same infrastructure as other cabin functionalities without the need for hardware configuration. If this paradigm were to be employed for all functionalities in the cabin, weight and complexity could be reduced and maintenance facilitated considerably [2].
The remainder of this paper is organized as follows. In Chapter 2 the AAP is introduced in detail. Chapter 3 describes the concept of the audio system application as well as its integration into a demonstrator. Chapter 4 evaluates the principal performance of the audio system application and compares the results with applicable requirements and other approaches. Chapter 5 presents the closing conclusions and gives an outlook on further research of the topic.

2. THE ADAPTIVE AVIONICS PLATFORM

The Adaptive Avionics Platform (AAP) is a concept for a self-organizing avionics platform, developed at the Institute of Aircraft Systems, University of Stuttgart, Germany [5, 6, 7, 8, 9].
The system topology can be built almost-freely from generic hardware modules. The generic hardware modules are CPM, IOM and ANS. Peripherals can be attached to any IOM in the system and are discovered automatically. Generic software components are adapted for the present topology to provide platform management functions and communication. Applications of the platform and the system functions are automatically allocated to respective modules in the topology.
The AAP incorporates different phases of operation during which the system possesses different abilities. Predominately, phases of operation divide into the standard phase and the organization phase. In the organization phase, the system has the means for self-configuration and therefore permits the system to be modified. The set of system functions can be altered and the instance size and topology can be changed to meet the present demands. The platform then discovers the system topology and configures the distributed system to execute the set of system functions. When in standard phase, the platform ensures the failure tolerant execution of the system functions. The configuration of the system remains static with the exception of fault isolation reactions. The system operates deterministically as if a human developer configured the system and its reaction to faults.
An AAP instance is built from the generic hardware and software components and is complemented by peripherals and applications. These are mostly specific to a single system function.
Peripherals can be smart or non-smart. Smart peripherals are capable of self-description and can, therefore, be discovered and integrated into the system in a fully autonomous fashion. Non-smart peripherals lack this feature. Information on how non-smart peripherals are connected to a specific IOM in the system must be supplied manually.
The applications divide into C-APPs and P-APPs. C-APPs compute the core functionality of the system function. P-APPs provide the means for the system function to interact with peripherals. C-APPs and P-APPs make use of a platform middleware that enables the communication of the application using a publisher-subscriber pattern.
A generic software component, the platform management (PlaMa) provides system-wide management functionalities. With help of the PlaMa, detection and isolation of faults is achieved in the system. The execution of the system functions is permanently supervised by the PlaMa.
By these means, the AAP seeks to simplify the development of avionics system functions by introducing a high level of abstraction and, therefore, hiding complexity from the function developer. Therefore, any system function becomes useable for many different cabin layouts without the need of hardware configuration or additional extensive development.
A lab demonstrator was realized to validate the AAP concept. The demonstrator integrates cabin management functionalities with the help of the AAP.
In the following, the adaptive avionics platform (AAP) concept is summarized.

2.1. Adaptivity Concept

The adaption of the APP instance requires the knowledge of the present system topology. Modules and their interconnections are automatically discovered. IOMs detect connected smart peripherals. The information on the detected peripherals and possibly connected non-smart peripherals is collected along with the general system topology. This is the starting point to automatically create a configuration for the AAP instance.
The AAP requires the system function to follow the basic concept of division into C-APP and P-APPs. This enables the distribution of P-APPs within the system to those IOM that have a matching peripheral type attached. The C-APPs are allocated to all present CPMs. The AAP transparently enforces a master/slave behavior of all CPMs found in the topology. The system function therefore becomes executable within different system topologies, while fault tolerance is provided by the AAP.
The communication between C-APPs and P-APPs is provided by the AAP via a publisher-subscriber pattern. All exchanged data is organized in topics. All applications provide information which topics are produced or consumed within the application. On grounds of this information, the AAP is able to organize the network during the organization phase. The applications are therefore enabled to produce and consume data without knowledge of its distribution within the system.
The PlaMa is adapted to be able to fulfill the management tasks for the discovered topology. The present network topology, modules as well as their hosted applications are considered during this step.
The integration of the system function is, therefore, possible without human interaction. Multiple system functions can be integrated in this way on the same AAP instance. The configuration of the network and function allocation is carried out automatically by the system. These steps normally require considerable time in instances of realistic size, which is achieved by the AAP in a matter of a few minutes.
The concept of the AAP intends the self-configuration capabilities to be enabled exclusively during a special operation phase dedicated to organization of the system. The exclusion of adaptivity in the standard phase facilitates future qualification of the AAP approach and systems that operate on top of the AAP.

2.2. Architecture of an Adaptive Avionics Platform Instance

All AAP instances rely on the same generic hardware and software building blocks. Specific system functions complement the generic instance with specialized hard- and software to create a system instance.
There are three types of generic hardware modules: (1) Computing Modules (CPM), (2) Input-Output Modules (IOM) and (3) Adaptive Network Switches (ANS). The CPMs host the centralized computing applications (C-APPs). Every system function normally requires one C-APP to perform a meaningful task. On the other hand, IOMs act as a gateway for the system function specific peripheral hardware and host the related peripheral application (P-APPs). P-APPs need to run once per connected peripheral hardware.
Both, CPMs and IOMs connect to the network via the ANSs. The ANSs ensure deterministic network routing and provide network fault-isolation as well as network fault-tolerance.
For these generic hardware modules, a few compilation rules exist. The basic network layout consists of daisy-chained ANSs. The switch at the start of the chain is called access switch, this switch only connects to its neighbor switch and all present CPMs. The last switch on the chain is called end switch. Since the system is intended to offer network fault tolerance, at least two of these chains, called network sides are necessary. Note however, that it is not necessary for a network side to have more than two ANSs (access switches and end switches).
Neighboring network sides are connected via their end switches. This enables alternate routes in case of a network fault, for example in case of loss of a wired connection. IOMs may connect to any switch that is neither an access switch nor an end switch.
Software wise, there are two generic software components, necessary for an AAP instance: (1) the platform middleware and (2) the platform management (PlaMa). The middleware provides data-centric communication via a publisher-subscriber pattern and generic interfaces for C-APPs and P-APPs. The platform management is a generic software that runs on all modules and provides platform management services to hosted C-APPs and P-APPs to provide platform management services like fault-tolerance.

2.3. Organization Phase

The system's adaptivity property is only available during the organization phase. For this additional software and hardware components are required that are unnecessary during the standard phase. Firstly, a local organization service (LORG) is present on all CPMs and IOMs. Secondly, the hardware instance is temporarily extended with a centralized organization controller (CORG). The CORG orchestrates the adaption process during the organization phase and is again obsolete once the standard phase is engaged.
The system can then be extended in hardware and software. For instance, additional network sides, CPMs, IOMs or ANSs can be added. New system functions or system function upgrades can be realized by adding new peripheral hardware and corresponding C-APPs and P-APPs. C-APPs and P-APPs are stored in the CORG repository. Once the changes to the system have been made, the organization phase is started. During the organization phase, the system topology is discovered. IOMs, CPMs, ANSs and their interconnections are determined. Connected peripherals, capable of self-description so-called, smart peripherals, are discovered.
Following the topology discovery, the CORG distributes the C-APPs and P-APPs to their respective modules. C-APPs and P-APPs provide information, which topics they would like to publish or subscribe. The ANSs are updated with routing tables for each publisher-subscriber topic to enable deterministic routing and the alternate routing for fault-tolerance.
For verification purposes of the topology discovery, an automatic verification algorithm allows the comparison of the discovered topology with an expected topology supplied as a model [10].
After the organization phase completes, the CORG is removed from the system and LORG services are stopped. From this point in time, the system can be regarded as a common avionics system, with no self-configuration capabilities at all.

2.4. Adaptive Cabin Management System Lab Demonstrator

An adaptive cabin management system (ACMS) based on the AAP was developed at the Institute of Aircraft Systems (ILS), University of Stuttgart, Germany. The lab demonstrator incorporates many of the typical system functions found in CMS today. These include cabin temperature monitoring, passenger service units, cabin illumination, a flight attendant panel (FAP) and the audio system presented herein. However, the system is based on consumer hardware and Linux operating systems with default scheduling. Therefore, the lab demonstrator lacks real-time capabilities and thorough process segregation. These restrictions do, however, not limit the lab demonstrator's capabilities to demonstrate the AAP and its adaptivity mechanisms.
Currently, the available hardware modules limit the maximum ACMS instance size. There are two CPMs, four IOMs and six ANSs, as well as the CORG available. Additionally, two ANSs run as virtual machines. This only enables for the minimum requirement of two network sides and two redundant master/slave CPMs.
The discovered topology as well as the allocated applications are visible, which correspond to the aforementioned system functions and the audio system application as well.The network spanned by the ACMS is referred to as the backbone network. Another network, called the test network enables the development and debugging of the ACMS lab demonstrator. All CPMs, IOMs and ANSs connect to the test network. The test network is accessible from a dedicated test computer. Both networks utilize Gigabit Ethernet. Inter-module communications rely on UDP multicasts. The IOMs and peripherals use Ethernet as the communication interface.
FIGURE 1 shows an actual organization result automatically generated during the organization phase of the ACMS lab demonstrator.

3. AUDIO SYSTEM APPLICATION

On basis of the AAP modifiable and scalable system functions can be developed. The audio system application presented in this paper targets the provision of cabin audio capabilities, while being scalable, modifiable and reusable.

These properties render the audio system application able to support cabin layout changes. Its application in different aircraft types for example across an aircraft family becomes possible. The audio system inherits these properties from the AAP and provides means to maintain this flexibility while considering realistic use cases.

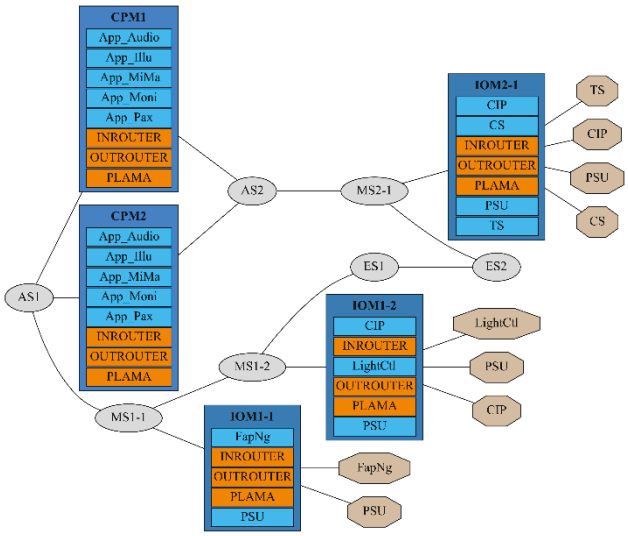

FIGURE 1. Actual organization phase result. The generic software is allocated to all modules. The CPMs host the C-APPs. The IOMs host the P-APPs that correspond to their connected peripherals.

The two peripheral types, cabin speakers (CS) and cabin interphones (CIP) are needed by the audio system to perform its task. The core functionalities of the audio system are: (1) playback of recorded audio, (2) announcements in the cabin via cabin interphones and (3) cabin intercommunication between cabin interphones. The control of audio playback relies on an flight attendant panel (FAP). In all scenarios audio streams are sent from or towards the peripherals. These audio streams are managed by the audio system application.
The tasks of the C-APP and P-APP application components are as follows. The C-APP hosts an audio stream handler that coordinates the audio streams in the system. The stream handler rejects or accepts incoming audio streams and signaling data. The stream handler can therefore limit the concurrent streams in the system or reject an audio stream when the targeted extension is already busy. If a stream is accepted, the stream handler publishes the stream to the network, which routes the data to the recipients. Otherwise, a termination message is returned to the source of the audio stream.
Another purpose of the C-APP is the provision of an audio player functionality. The player is controlled remotely via the FAP. On request, the player publishes recorded audio data as an audio stream via the stream handler to the cabin speakers and reports its state to the FAP, which uses this information for proper display. The player informs the FAP of the available media, which enables adaption of the FAP user interface.
The P-APPs act as input-output gateways to the network for the peripherals and relay the peripheral data to the C-APP. However, P-APP include logic to filter irregular requests before they are published over the network. Irregular requests can occur, when the user of the peripheral dials an extension number that is not present in the system.
FIGURE 2 shows an exemplary architecture of the audio system application. The peripherals are connected to different IOMs, which run the P-APPs. The C-APP is allocated to the CPMs.

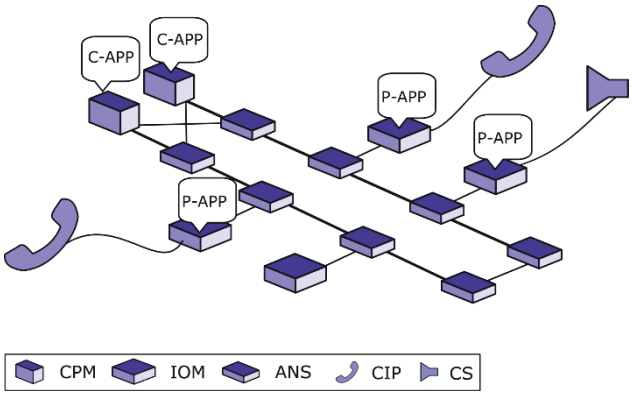

FIGURE 2. Audio system application architecture. The P-APPs manage their connected peripheral. The C-APP supplies the core functionalities.

In aircraft, many cabin speakers and multiple cabin interphones are installed. The AAP ensures that all of these peripherals are accessible by the audio system. However, neither the AAP nor the audio system can decide autonomously how these peripherals are used meaningful. In the context of the audio system the operation is only meaningful if interphones at certain physical location can be called or announcement can be played back with multiple speakers at once. This requires that an description is provided from which the audio system can derive how its functionality shall be performed.

3.1. Audio Streaming

To provide all cabin audio functionalities the audio system must transport and handle audio and signaling data. In the following, different details of how the audio system accomplishes this within the AAP are presented.

Stream Transport

The AAP offers messaging via a publisher-subscriber pattern. Four different message topics are used for the audio system: (1) upstream audio and meta data (towards C-APP), (2) downstream audio and meta data (towards P-APPs), (3) audio player requests and (4) audio player status. The C-APP subscribes topics (1) and (3) and publishes topics (2) and (4). The P-APPs subscribe topic (2) and the FAP additionally subscribes topic (4).
The downstream from C-APP to P-APP needs to contain audio only in case a call was established or a broadcast is sent. Only in these cases the peripherals process the audio signal. Therefore, topics (1) and (2) carry the audio signal under these conditions otherwise only meta data is sent.
Ideally, the downstream audio would be routed only toward the receiving P-APPs. However, due to the implementation of the network routing of the AAP, no individual network routes from C-APP to particular P-APPs exist. The P-APPs of the audio system therefore select only data from the downstream relevant for their peripheral. FIGURE 3 depicts exemplary the network routes provided by the AAP.

Therefore, network bandwidth is consumed along the routes toward any audio system P-APP. This plays a role when regarding concurrent audio streaming, because a multiple of the processable bandwidth is then unnecessarily transmitted. Especially for closely related in-flight entertainment systems this behavior quickly drives the network bandwidth against the physical limits.

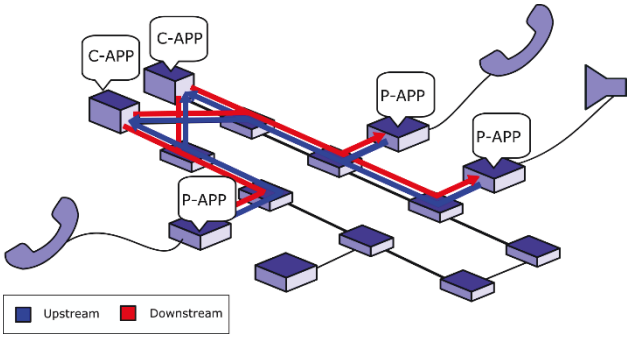

FIGURE 3. Provided network routes for stream transport. The downstream is provided only by the active master CPM. The upstream is received by both CPM.

Stream Meta Data

The stream meta data is used to indicate the state of the stream, its source and its destination. Source and destination are specified via a node identification number.
The state of the stream can be: TERM (terminated), RING (ringing/accept), INCALL (established call) or BROADCAST (broadcast transmission). State TERM is used to indicate the stream termination, e.g. after hanging up or if the stream is declined. State RING signals that a node is trying to call another node. State INCALL is used when the stream has been established between both parties. State BROADCAST is used when audio data is transmitted without the need that the destination is responding with audio data on its own, e.g. when an audio player is broadcasting audio data to a cabin speaker.

Stream Handling

The stream meta data is included in both, upstream and downstream topics. Audio nodes may commence a stream via stream states RING and BROADCAST. The C-APP will decide whether the destination is available and will either publish the stream to the downstream topic or publish a termination message. If a destination is available, broadcast streams will always be forwarded, without the need to wait for an answer of the destination.
For calls however, only if an upstream of the destination with state RING is detected, the stream state is altered to INCALL by the C-APP. Now, audio nodes publish their streams marked with state INCALL until the stream terminates.
Through this mechanism, the C-APP has the full authority to accept and reject streams and is able to prioritize and terminate streams. Termination of a stream is signaled by the C-APP by a downstream topic with state TERM.
While data transmissions from the P-APPs towards the C-APP cannot be limited this way, the C-APP only forwards accepted streams to the downstream topics and limits the total count of concurrent streams towards the P-APPs at any time.
This approach offers the possibility to detect a malfunctioning P-APP and inform the platform with a failure indication, which could in turn passivate or reset the P-APP. However, such mechanisms are not implemented in the current ACMS lab demonstrator.

Stream Audio Format

The audio system uses 16-bit signed integer stereo pulse code modulation (PCM) samples and a sample frequency of 48000 Hz as an uncompressed interchange format. This results in a relatively high bandwidth usage of 1536 kbit/s, but does not necessitate encoding and decoding. Due to the multicast network messaging there is, however, no increase in the used bandwidth, if audio streams are cast to more than one receiver. In fact, the current ACMS demonstrator cannot target audio nodes directly, but distributes every stream to every node in the system. The nodes must filter the stream data, which addresses them.
These constraints are however beneficial for stressing the demonstrator backbone network, which only contains three audio peripherals. Therefore, only a maximum of two concurrent streams are possible with a combined load of 4608 kbit/s of audio bandwidth in total (one duplex + simplex stream). These network loads might occur in AAP instances with a realistic size. A likely scenario where this might become relevant are music streams of in-flight entertainment systems, which are, however, out of the scope of the audio system application.

3.2. Audio Peripherals

The audio system uses cabin speakers and cabin interphones to perform its tasks. The audio peripherals communicate with the P-APP and handle the data towards and from the actual audio hardware. The peripherals are based on similar state machines, which implement the call logic. FIGURE 4 depicts the state machine of the call logic implemented in the audio peripherals.

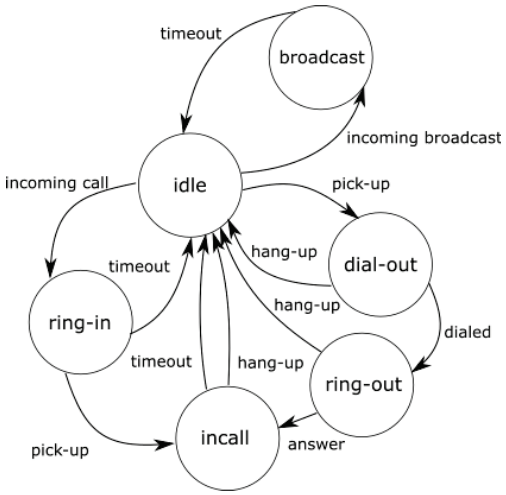

FIGURE 4. Call logic state machine for audio peripherals. The entry state is the state idle. Certain events trigger the transition into the other states.

The initial state idle is left either, if an incoming call is signaled by incoming data or, in case of the CIP, a call is initiated via the handset. In the former case, the state

machine transitions into the ring-in state, which signals the incoming call and is either left by picking up the call or the caller has canceled the call. The state machine of the CS transitions automatically from the ring-in state to a pickup state.
For the ACMS lab demonstrator, the peripherals were implemented on single-board computers. The cabin speakers use the integrated audio chip to drive amplified loudspeakers. The cabin interphone rely on USB phone handsets and a numeric keyboard.

3.3. Flight Attendant Panel Integration

The integrated audio system is controlled and its status is displayed via the flight attendant panel (FAP). Most importantly, audio playback is operated from the FAP. As mentioned previously, the C-APP subscribes a topic dedicated to audio playback commands. The FAP publishes this topic, which can contain the different types of commands to control the audio playback feature of the C-APP. Through this play, pause and stop commands as well as media change commands can be issued. In turn, the C-APP publishes audio playback information, containing the identifier of the loaded audio resource, the playback state and the playback time position. The FAP displays the current audio network layout. The display serves as a phonebook, displaying all extensions numbers and their description. All active audio streams are listed alongside their state, connection ends and the time they have been active, based on the audio stream topics read by the FAP. The stream information is used to mark the extensions busy that currently process a stream. FIGURE 5 displays the FAP interface with the audio playback controls on the right and node and status information on the left. The FAP of the ACMS lab demonstrator is displayed and controlled via a consumer tablet.

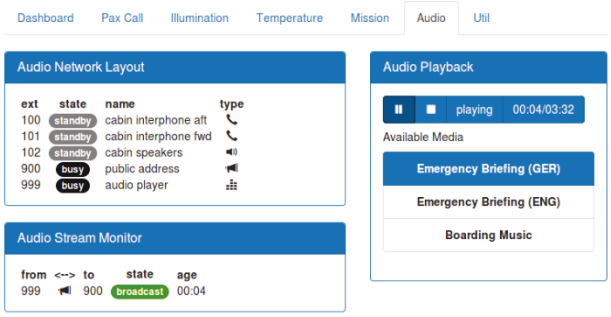

FIGURE 5. FAP integration of audio system application. Audio playback can be controlled in the right section. The left sections displays the status of the audio system.

3.4. Location and Functionality Mapping

Virtually every real-life application depends on the knowledge of the physical location of its peripherals. This holds true for the cabin audio functionalities. One way to make the audio system easily useable by the crew, are pre-defined extension numbers for the peripherals located at certain physical locations. The peripherals should be line-replaceable without further configuration effort. Different zones of the aircraft might be targeted during public address. Therefore, due to the lack of algorithmic discovery, the locations of the peripherals installation points are supplied by external means. In reality, this could be achieved through pin programming. For the ACMS lab demonstrator a preset meta file on the filesystem of the peripheral contains information of its physical location.
The meaning of the physical locations of the peripherals for the operation of the audio system is given by an audio node model. This model enables the audio system to work under different contexts. Essentially, these parameters represent all the currently undiscoverable parameters needed for operation. Research has been done on the determination of the physical location with the help of ultrasonic and radio frequency signals, an approach without the need of additional hardware is not available [11]. Therefore, the provision of these parameters by the developer is presently indispensable.

Audio Nodes and Extensions

The distributed manner of the audio system necessitates a generalized scheme of the distributed peripherals connected to the AAP instance. Every audio device, that is an audio receiver or transmitter is called an audio node. Audio nodes are identified by their aircraft-wide unique node identification number.
There are two different node types for the audio system: physical and virtual. All physical nodes refer to peripheral installation points. Virtual nodes are nodes that can have an arbitrary number of physical nodes associated. Additionally, virtual nodes can be nested. This facilities the introduction of zones in the aircraft, which can be addressed in announcement or audio playback independently from other zones. A special case of the zonal configuration is the passenger address, which includes every cabin speaker available in the aircraft.
Virtual nodes aid in the definition of separated cabin layouts. For instance, for the separation of economy class and business class. FIGURE 6 shows an exemplary cabin layout with two distinct zones. Cabin Speaker and Cabin interphone mount points are depicted. Zones A and B form separate virtual nodes, that can be accessed individually. A superordinate virtual node can group the zones so that simultaneous playback in both zones is possible.

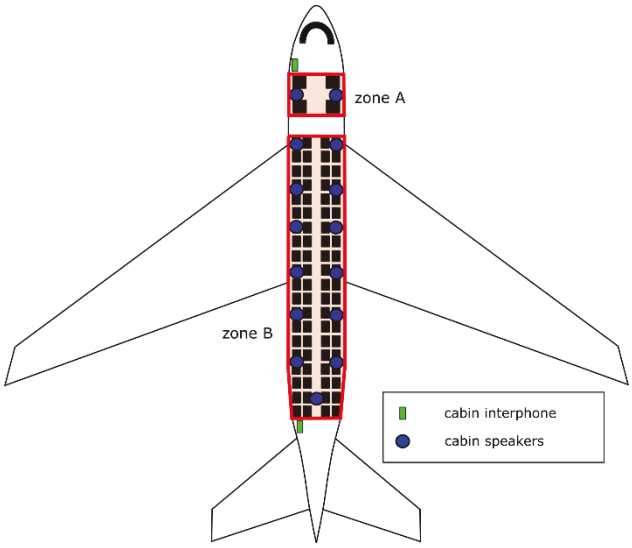

FIGURE 6. Possible cabin layout for the audio system. Zone A groups the speakers in the business class, while zone B groups speakers of the economy class.

The identification of an audio node and the assignment of a dialable extension numbers are logically separated in the audio system. Therefore, extension numbers have to be defined that map to a specific audio nodes. These extension numbers can be entered on a cabin interphone handset to make a call.
The zonal membership and extension numbers is not defined by autonomic adaption. The rationale behind this is that the system architect supplies these parameters as a pre-definition to ensure a useable and deterministic mapping. An autonomic alternative, where the system is able to conclude how it should configure itself to maximize usability, is currently unavailable.
Updates to these parameters are possible without the need for organizing the AAP anew, because they only affect the operational behavior of the audio system. Additionally, the actually connected nodes can be determined. Therefore, a single model can fit similar cabin layouts and increases reuse across an aircraft family or fleet. In FIGURE 7 the audio node model of the audio system instance of the ACMS lab demonstrator is shown, see FIGURE 2 for comparison.

```
<audioMeta>
  <audioNodes>
    <virtualNode node_id="0" vid="PUBLIC_ADDRESS" name="public address">
      <member node_id="3"/>
    </virtualNode>
    <physicalNode node_id="1" hwid="CIP-001" name="cabin interphone aft"/>
    <physicalNode node_id="2" hwid="CIP-002" name="cabin interphone fwd"/>
    <physicalNode node_id="3" hwid="CS-001" name="cabin speakers"/>
    <virtualNode node_id="9999" vid="AUDIO_PLAYER" name="audio player"/>
  </audioNodes>
  <extensions>
    <extension ext="100" assoc_node="1" callable="true"/>
    <extension ext="101" assoc_node="2" callable="true"/>
    <extension ext="102" assoc_node="3" callable="true"/>
    <extension ext="900" assoc_node="0" callable="true"/>
    <extension ext="999" assoc_node="9999" callable="false"/>
  </extensions>
</audioMeta>
```

FIGURE 7. Exemplary audio node model. The audio system's physical and virtual nodes are defined in the upper section. The lower section defines the extension number for the present audio nodes.

4. PERFORMANCE ANALYSIS

Cabin audio is one of the most demanding tasks of current CMS. The reason for this is that strict timing requirements for cabin audio exist. For example, the time delay between the audio source and the actual played back audio should be below 100 ms to ensure lip-synchronous announcements [3].
Even more severe are the requirements on synchronicity of audio emitting from different speakers. To avoid audio incomprehensibilities a maximum of 1 ms of time difference for simultaneous playback on cabin speakers is required [3].
In this Chapter, the audio system application is analyzed towards these requirements with the help of a lab demonstrator.

4.1. Performance Aspects

As previously mentioned the ACMS lab demonstrator lacks real-time capabilities for network and task scheduling respectively. Therefore, the ACMS lab demonstrator offers only best-effort timing. This means that no deterministic behavior is ensured and hence no guarantee can be given for the performance of any integrated system. This limited analysis is intended to enable a comparison of the current ACMS with actual cabin audio systems and to demonstrate that some associated requirements can be met nevertheless by an audio system integrated on the current AAP.
Foremost, latency and synchronicity are the properties that are mandatory for cabin audio. In this paper, the latency describes the end-to-end time delay between two audio peripherals (e.g. CIP-to-CIP or CIP-to-CS). The synchronicity describes the time difference of the audio playback of two cabin speakers.
By measuring audio latency and synchronicity, the performance of the integrated audio system can be benchmarked. The measurements were carried out with consumer grade USB audio devices for this analysis.

4.2. Measurement Scenarios

To record audio signals, a Laptop and two consumer USB audio devices were used. However, due to the nature of USB, these devices show considerable asynchronicity during recording. To achieve meaningful results, at the start of each measurement both audio devices record the same physical audio source, therefore exposing the present measurement asynchronicity. During the same record, the audio sources are switched manually by rewiring to the actual sources to be compared. During post-processing, the record is split due to silence during switching of the audio sources. The asynchronicity of a measurement can be determined through cross-correlation. Calculating the difference between the asynchronicities in the reference measurement and the actual measurement, yields the asynchronicity of the audio sources. For the measurements recorded speech is used as audio.
Two scenarios are taken into account for the measurements. FIGURE 8 and FIGURE 9 depict the scenarios.

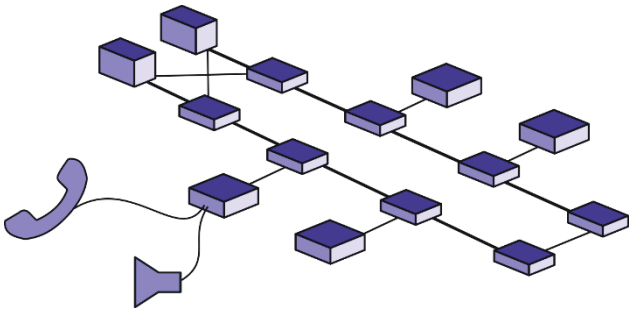

FIGURE 8. Setup with same IOM for measurements. The peripherals are connected to the same IOM. All streams are processed by a single IOM.

At first, the audio peripherals are connected to the same IOM, which is reached from the CPM via two switches. During the second scenario, one peripheral is connected to the previously mentioned IOM, but this time, the second audio peripheral is connected to an IOM on the other network side, which is reached by the CPM via three switches.The measurements can provide information whether the system topology plays a role for the performance values of the audio system.
In the case of audio latency, an external audio source

connects to an audio peripheral. The audio travels through the audio system toward another audio peripheral. The output of the audio peripheral is then compared to the original external source to measure the audio system's latency. FIGURE 10 shows the setup for the measurement of the latency.

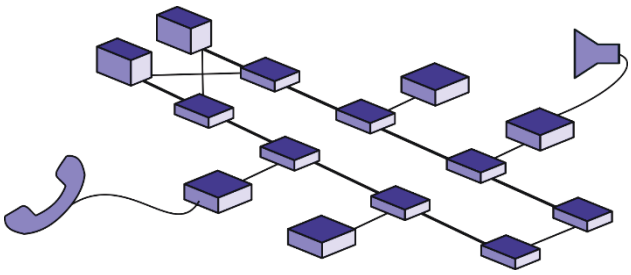

FIGURE 9. Setup with different IOMs for measurements. The peripherals are connected to different IOMs, which process the audio streams individually.

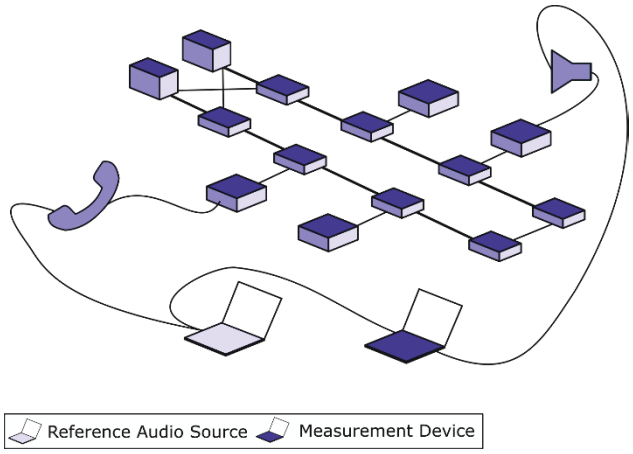

FIGURE 10. Setup for measuring audio latency. The reference's audio signal travels through the audio system to the measurement computer and is fed directly

The audio time difference is measured with the help of audio playback from the C-APP. The output of two audio peripherals is then compared to measure the audio time difference of the two peripherals. For each measurement setup one hundred measurements were obtained.
FIGURE 11 depicts the setup for measuring the audio time difference.

4.3. Audio Latency

One hundred measurements for the audio latency were obtained for each of the two setups. The statistical properties of the measurements are given in TABLE 1.
The comparison of the measurement results shows, that in the considered cases the position of the audio peripherals within the system topology does play a minor role concerning audio latency. When both audio peripherals are connected to the same IOM, a slightly higher latency is observed.
The mean value of the measured latencies indicates that lip-synchronicity is achieved in most cases. However, outliers above 100 ms and a rather huge standard deviation of 10 ms have been observed.

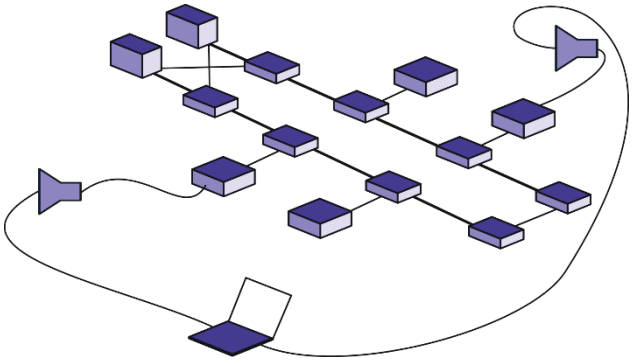

FIGURE 11. Setup for measuring audio asynchronicity. The audio signal is sent from the C-APP towards the speakers. The audio signal of the speakers is compared on a measurement computer.

TABLE 1. Audio Latencies
(SD=Standard Deviation)

Case	**Mean** [ms]	**Median** [ms]	**SD σ** [ms]
Same IOM	87.9	87.3	10.0
Different IOM	85.3	84.1	9.7

The latency of the considered cases differs due to higher loads and schedule uncertainties, when the same IOM is used.
For the implementation a buffer period of 16 ms was chosen. This buffer period showed to work well with the hetereogenous audio hardware of the demonstrator. Buffering is one contributor to latency, hence, the larger this buffer period is chosen the larger the resulting latency is.
The observed standard deviation is close to the buffer period and might, therefore, indicate sporadic loss of audio data within the platform. Overall, the audio system uses five times the buffer period. The relatively large number of buffers in the current implementation has shown to provide better resilience towards network and scheduling uncertainties, especially when audio is played back directly from the C-APP. All buffering occurs within the peripherals not in the platform itself.
In total, at least 80 ms of end-to-end time delay from microphone to speaker is introduced. Additionally, network transport, which can be attributed to the platform and processing in the C-APP and P-APPs adds further delay. On the audio peripheral, processing in the audio hardware's driver might add to the overall delay as well.
The added delay by network transport, C-APP processing and drivers is small in relation to the delay introduced by buffering. The measurement results indicate however, that further lowering the buffer period and the amount of buffers in the implementation can yield considerably lower latencies if needed. The platform concept introduces no prohibitive delay to achieve such low latencies.
Often, a more rigorous 10 ms upper bound for the audio

latency is applied [4] however, even though the platform is capable to host such low-latency applications, the ACMS demonstrator is not suitable to demonstrate this for audio applications at the moment due to audio hardware limitations in the demonstrator setup.

4.4. Audio Asynchronicity

One hundred measurements of the audio synchronicity were obtained. The statistical properties of the measurements are given in TABLE 2.

TABLE 2. Audio Asynchronicities
(SD=Standard Deviation)

Case	**Mean** [ms]	**Median** [ms]	**SD σ** [ms]
Same IOM	17.2	16.6	4.7
Different IOM	21.3	16.8	16.0

For both setups, the mean of the observed time difference are slightly above the median of roughly 16 ms. For the second setup a higher standard deviation was observed, which might indicate that outliers contribute to the higher mean. Both setups show a similar result, therefore the asynchronicity was not influenced by the different setups. Overall, the time difference required by cabin audio systems was exceeded considerably.
For the audio system application, a time difference of the audio playback occurs because neither the playback peripherals (cabin speakers) nor the audio player (C-APP) are real-time capable or share a common time basis. The audio playback of the peripherals can therefore jitter.
The audio system does currently not satisfy the strict synchronicity requirements according to the made measurements. The asynchronicity is in average slightly above the buffer period of 16 ms. The median is even closer to 16 ms.
The playback jitter can, therefore, mainly be attributed to different audio frames being played by the peripherals. Therefore, the asynchronicity measurements are subject to jitter within the scope of the buffer period of 16 ms. Drift and further jitter are introduced by the audio hardware that drives the audio playback.
A comparison with the findings in [4] shows, that synchronicity below the buffer period can be achieved with clock synchronized audio peripherals. Synchronization is an important platform task, which the AAP implementation is currently lacking. Lowering the buffer period increases synchronicity, but the desired limit of 1 ms cannot be achieved this way due to the audio hardware's lower limit of the buffer size used in the ACMS demonstrator.

5. CONCLUSION AND OUTLOOK

This paper presented an audio system for the AAP. The AAP allows the audio system to be freely distributed within a system topology. The audio system is therefore modifiable and scalable and can be reused in different cabin layouts in an almost configuration-free manner.
The audio system was integrated into an ACMS lab demonstrator. Interaction with other components like a flight attendant panel were used to enable control and supervision of the audio system functionalities. A subsequent performance analysis has shown, that the AAP concept is able to host the audio system and can provide acceptable latencies. The synchronicity requirements of modern CMS were not achieved by the audio system. Comparable approaches show however, that clock synchronization is able to provide audio with the desired synchronicity.
The exemplary implementation in the ACMS lab demonstrator has shown, that more flexibility in the choice of the architecture of integrated systems as well as better network orchestration are needed for real-life applications. Fully adaptive systems are probably still decades away from being used in aircraft. Today, some parameters still need to be supplied by developers. Furthermore, the development costs for safety related functions are mainly driven by safety requirements, not by the function itself. Both problems call for a higher self-consciousness of the platform and a high-level task description. Then, the safety of functions can be ensured by a platform middleware. This approach is targeted with the research on Plug & Fly avionics (PAFA) [12]. FIGURE 12 shows the basic PAFA concept.

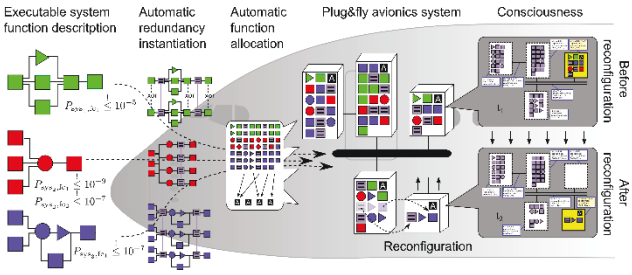

FIGURE 12. Plug and Fly Avionics Concept. Simplex task descriptions are translated into redundant instances that are automatically allocated (left). A system consciousness enables (re-)configuration (right).

The system knowledge is concentrated in a self-consciousness model, which holds all information about the system, such as system topology and the tasks to be executed by the system. The tasks are supplied as model-based system function descriptions. These descriptions contain desired failure probabilities, timing demands, synchronicity demands and data flow between application elements. Function developers only concentrate on the description of the function itself and supply basic failure mode and effects analysis (FMEA) results along with it. The platform is then able to decide how the function can be executed safely. This is achieved through automatic redundancy instantiation and automatic function allocation within the distributed system. The concept allows for live reconfiguration whenever the need arises to ensure and continue safe execution of system functions.
In the context of the audio system the PAFA concept allows the audio timing requirements to be specified in the task description, therefore facilitating the development of such time critical functions.

ACKNOWLEDGEMENTS

The German Federal Ministry for Economic Affairs and Energy (BMWi) has funded this research within the LUFO V-2 programme and the KOMKAB project.

REFERENCES

[1] S. Burger, O. Hummel and M. Heinisch, "Airbus cabin software," *IEEE software,* vol. 30, pp. 21-25, 2013.

[2] E. Heidinger, "Performance Bounds in Switched Ethernet Onboard Networks," 2013.

[3] F. M. Leipold, "Wireless UWB aircraft cabin communication system," 2011.

[4] E. Heidinger, F. Geyer, S. Schneele and M. Paulitsch, "A performance study of Audio Video Bridging in aeronautic Ethernet networks," in *Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on*, 2012.

[5] M. R. Ahmadi, Adaptive Architecture for Fault-Tolerant, Hard Real-Time and Integrated Plug-and-Fly Avionics, Dr. Hut, 2018.

[6] R. Ahmadi, O. Marquardt, M. Riedlinger and R. Reichel, "An adaptive software architecture for future CMS," *SAE International Journal of Aerospace,* vol. 8, pp. 260-272, 2015.

[7] O. Marquardt, M. Riedlinger, R. Ahmadi and R. Reichel, "An adaptive middleware approach for fault-tolerant avionic systems," in *Aerospace Conference, 2015 IEEE*, 2015.

[8] O. Marquardt, M. Riedlinger, R. Ahmadi and R. Reichel, "Autonomous peripherals integration for an adaptive avionics platform," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, 2016.

[9] M. Riedlinger, O. Marquardt, R. Ahmadi and R. Reichel, "An adaptive self-managing platform for cabin management systems," *CEAS Aeronautical Journal,* vol. 7, pp. 483-498, 2016.

[10] B. Schulz and B. Annighoefer, "A Verification Algorithm for the Automatic Topology Discovery of the Adaptive Avionics Platform," in *Digital Avionics Systems Conference (DASC), 2018 IEEE/AIAA 37th*, 2018.

[11] M. Gerdes and D. Scholz, *DESIGNING AND EVALUATING A LOCATION DETECTION SOLUTION FOR THE AIRCRAFT CABIN WITH MODEL BASED SYSTEMS ENGINEERING,* Hamburg: Hamburg University of Applied Sciences Aero-Aircraft Design and Systems Group, 2010.

[12] B. Annighoefer, M. Riedlinger and O. Marquardt, "How to tell configuration-free integrated modular avionics what to do?!," in *Digital Avionics Systems Conference (DASC), 2017 IEEE/AIAA 36th*, 2017.