An Investigation into the AES X170 Project for Control of Ethernet AVB Networks

James Dibley, Richard J. Foss
Department of Computer Science
Rhodes University, Grahamstown 6140
Tel: +27 46 603 8291, Fax: +27 46 636 1915
email: gjld3593@ict.ru.ac.za; r.foss@ru.ac.za

Abstract—The Audio Engineering Society (AES) X170 project aims to define a control and management protocol for audio distribution networks. This paper describes recent investigations and developments in AES X170, including the development of an XFN implementation specification and the implementation of native XFN control over an Ethernet AVB endpoint.

Index Terms—AES X170, XFN, Ethernet AVB, XMOS, audio distribution networks.

I. BACKGROUND

Ethernet Audio/Video Bridging (AVB) is a suite of cooperating protocols developed by the IEEE [1] that extend the Ethernet standard to provide deterministic streaming of uncompressed audio and video data across quality of service (QoS) networks.

In Ethernet Audio/Video Bridging, audio and video data is distributed in streams, transmitted between two or more endpoints that are connected in a network via bridges. Any endpoint that intends to transmit data onto the network is termed a Talker, and must register a stream that will distribute the data it intends to transmit. The registration of this stream is relayed through network bridges to all endpoints on the network. One or more endpoints, termed Listeners, may declare an interest in receiving the stream. If at least one Listener endpoint wishes to receive the stream and the route between the Talker and that endpoint can satisfy the stream’s QoS requirements, the Talker will transmit the stream.

Audio and video data streams are time-sensitive in ways that most data transmissions are not [2], which means that the QoS requirements for a distribution network are extensive, and can be summarised as follows:

- Latency is to be minimal and deterministic
- Bandwidth is to be guaranteed for the lifetime of a stream in advance of the stream’s distribution
- Presentation of the stream at multiple Listeners must be synchronised.

The primary goal for Ethernet AVB is to deliver a distribution network technology capable of satisfying these QoS requirements while permitting the convergence of audio-video stream data and conventional data networking on a common infrastructure [3]. The component protocols of the Ethernet AVB suite include:

- IEEE 802.1Qav defines enhancements to forwarding and queueing logic for network bridges and endpoints that define transmission classes and algorithms to provide self-configuring traffic shaping services. This provides a basis for convergence with conventional data networking and enforces the most efficient possible use of available bandwidth.
- IEEE 802.1AS defines mechanisms and procedures that provide a unified and self-configuring time domain for the distribution network. This provides the foundation for satisfying several essential QoS requirements, including transport synchronisation, regeneration of media clocks, and assessment of transport latency.
- IEEE 802.1Qat defines mechanisms and procedures to broadcast and negotiate QoS commitments for a given data stream. This ensures that for every network path between a transmitting device and potential receivers there are adequate resources to satisfy a data stream’s QoS commitments. If insufficient resources exist then transmission does not take place [4].
- IEEE 1722 defines behaviours, message formats and protocols for transport of audio/video data across a QoS network that provides bandwidth reservation via IEEE 802.1Qat and a time domain via IEEE 802.1AS. IEEE 1722 can encapsulate a number of popular audio / video data protocols.
- IEEE 1722.1 provides a protocol to perform discovery, enumeration, connection management and control of audio distribution devices.

Ethernet AVB is preceded by a number of technologies that provide audio distribution networks with quality of service features. These include CobraNet and EtherSound, which provide QoS audio distribution networks above Layer 2 Ethernet [2]. These proprietary technologies have met with commercial success. However, a 2005 survey of members of the Audio Engineering Society indicated a ‘strong … desire to see open and multi-sourced technology standards for audio networking’ [5]. Ethernet AVB intends to provide such a standard.

The same survey indicated ‘the core importance of control and monitoring capability’ [5]. This paper describes an approach to providing control and monitoring capability.

1 This work was undertaken in the Distributed Multimedia CoE at Rhodes University, with financial support from Telkom SA, Converse, Stortech, Tellabs, Easttel, Bright Ideas 39, THRIP, and Universal Media Access Networks GmbH.
for Ethernet AVB devices through new developments concerning the Audio Engineering Society’s X170 project [6]. It describes the development of an implementation specification for XFN, the project protocol, the investigation of an open-source implementation of Ethernet AVB, the implementation of native XFN connection management on an Ethernet AVB endpoint, and outlines proposals for further work in this area.

II. INTRODUCTION

The purpose of an audio distribution network is to simplify and assist the task of audio distribution by enhancing or replacing conventional non-networked audio distribution systems.

Many of the use cases for these systems are not readily apparent to the layman: in addition to the visible uses in concert venues and theatres, audio distribution systems are also employed in large public spaces (airports, transit systems, hospitals, state buildings). These use cases can be categorised as ‘live sound’ and ‘sound reinforcement’ applications, which share some of their user requirements and are distinguished by others.

Conventional audio distribution systems present difficulties and constraints to each of these applications. The cost, weight, operating tolerances and scarcity of audio cabling are all significant factors, as is scalability, but perhaps the most significant difficulty is the coupling that exists between a channel of audio signals and its physical cable. This dramatically limits the resilience of the distribution system and makes the provision of transport redundancy all but impossible.

Audio distribution networks promise to remove these limitations by decoupling the work of audio distribution from the topology of the distribution system.

An audio distribution system may be a permanent or temporary installation. A sound reinforcement system is likely to be a permanent installation, and also likely to be a large diameter network, whereas a live sound system is most likely to be a temporary installation over a relatively confined area. In sound reinforcement applications, security, scalability and integration with other presentation technologies (video, data) are paramount. In live sound applications, portability, modularity and ease of maintenance are immediate and routine concerns.

However, one requirement is accepted as universal: the need for control and monitoring of the network. As Gross reports:

Sixty-four percent of respondents considered monitoring or remote control mandatory. Among installed-sound respondents the call was even louder at 84 percent … this may turn out to be the most important factor driving adoption of audio networks. Especially in installed sound, the primary draw of an audio network may be in the control and monitoring capabilities. [5]

As indicated earlier, the draft IEEE 1722.1 protocol provides services for device control, connection management, discovery and enumeration on Ethernet AVB-compliant networks. The Audio Engineering Society’s X170 project provides an alternative control protocol that has been demonstrated by prior research to provide suitable encapsulation of Ethernet AVB functionality [7] in addition to interoperability with further disparate control protocols [8].

A. XFN

XFN is an IP-based peer-to-peer protocol for control and management of devices on audio/video distribution networks, and a core component of the Audio Engineering Society’s X170 project [6].

An XFN audio distribution device is represented within the protocol as an array of device parameters. Device parameters may represent the audio distribution features of the device in terms of control parameters and display parameters; they may also represent more general information, such as the configuration of the device’s network interface(s). Control parameters expose device features as read/write controls to other X170 devices; display parameters provide read-only indications of device states.

XFN devices may interact by sending control messages between parameters. XFN control messages are defined in terms of verbs (the ‘Command Executive’) and attributes (the ‘Command Qualifier’), and the object of a control message may either be a device parameter or the device itself. Example verbs include ‘Get’, ‘Set’, and ‘Create’; example attributes include ‘Value’, ‘Name’, and ‘Flags’.

XFN defines a sophisticated representational schema that allows the parameter representation of a device to closely map its functional layout. Each device parameter is addressed in terms of seven hierarchical levels, each successive level defining a more specific functional classification. Figure II-1 displays this hierarchy for a subset of configuration parameters:

---

---

2 XFN is an acronym for ‘Cross-Fire Network’. This name is explained in Annex E of the AES X170 specification. [9]
the operating practice of audio engineers. These include a sophisticated implementation of parameter grouping, which enables parameters to track value changes on other parameters (not necessarily on the same device; cf. Figure II-2), a ‘push’ notification system that provides responsive metering of display parameters, and mechanisms intended to render the network discovery and device enumeration processes effectively invisible to the end user.

Figure II-2 Parameter grouping in X170.

XFN was originally employed to control devices within IEEE 1394 (‘Firewire’) audio distribution networks:

“At the time of its development, Firewire was the only standards based network technology that allowed for deterministic transmission of audio … this has now changed … and Ethernet AVB is now the network technology of choice …” [9]

At the time of writing, XFN has been implemented within a core protocol stack (‘UNOS Core’) and a controller application (‘UNOS Vision’) [10].

Research has shown that XFN is readily capable of encapsulating Ethernet AVB device parameters for connection management, [7] and the flexibility of the XFN representational schema has also enabled a number of proxy implementations providing interoperability between XFN and other control protocols [8].

A common hindrance to this research, however, has been the limited availability of Ethernet AVB-compliant devices for testing and evaluation. Since certain components of Ethernet AVB (notably IEEE 802.1Qav) demand substantial revisions of conventional networking technology, there is a clear obstacle to practical research. Prototypes of the kind described in [7] can perform emulation of components of the standard but rely, in part, on conventional networking, hindering investigation of XFN monitoring and control over the full functionality of an Ethernet AVB endpoint.

The following section describes a solution to this problem.

B. The XMOS XS1 & implementations of Ethernet AVB

The XMOS XS1 is a general-purpose multi-threaded processor architecture that provides hardware implementations of threading, scheduling, process communication and timing:

The processors are general-purpose [author’s italics] in the sense that they can execute languages such as C; they also have direct support for concurrent processing (multi-threading), communication and input-output. [11]

The implementation of these features within hardware provides deterministic execution of multi-threaded applications. The thread scheduler is capable of recognising a wide variety of conditions as events; as a result, an event-based processing model may largely replace hardware-based interrupts, promoting determinism. In addition, each hardware thread has dedicated registers, intelligent input-output, a number of configurable hardware timers, and a guaranteed allocation of processing power.

Threads may interact via synchronising channels, providing message-passing interprocess communications. Channels may also be used to implement asynchronous and streaming (unsynchronised) forms of communication between threads.

A stated goal of the XS1 architecture is to ‘make it practical to use software to perform many functions which would normally be done by hardware’ [11].

The XS1 architecture has been implemented in a number of ‘XCore’ processor configurations, from single-core to quad-core processors. Each processor core provides eight hardware threads and 64KB of onboard SRAM, plus extensive software-configurable I/O. A variety of development boards are available (Table II-1), while XMOS provide royalty-free open-source repositories of common software components (development tools, Ethernet and IP stacks, general purpose I/O drivers) for developers and designers to adapt and use within their own applications.

Table II-1 XS1 processor configurations [12]

<table>
<thead>
<tr>
<th>XCore (XCores)</th>
<th>XS1-L1</th>
<th>XS1-L2</th>
<th>XS1-G4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threads (MIPS)</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>400/500</td>
<td>800/1000</td>
<td>1600</td>
</tr>
<tr>
<td>16</td>
<td>64KB per XCore</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The flexibility of the XS1 architecture and the relative accessibility of the design capture process have facilitated an open-source implementation of an Ethernet AVB endpoint. A number of versions of this Ethernet AVB endpoint application are available for demonstration on a variety of XS1 development boards. The most economical implementation can be deployed on a dual-core XCore processor, providing an IEEE 802.1Qav-compliant Ethernet stack, a TCP/IP stack, an I2S audio driver, media clocks, an IEEE 1722 Talker, an IEEE 1722 Listener and a proprietary remote control implementation in a total of fifteen threads.

However, the innovative nature of the XS1 architecture precludes any realistic opportunity to ‘port’ an existing implementation of the XFN protocol stack onto these devices. In order to investigate XFN control over an XMOS Ethernet AVB endpoint, it became necessary to approach a native implementation of the XFN protocol stack.

III. DESIGN

These investigations have produced an implementation specification for XFN and a design for native XFN control over XMOS Ethernet AVB endpoints. The following subsections describe how this was achieved.

A. An XFN implementation specification

The Audio Engineering Society specification for XFN ‘specifies an approach to the control and monitoring of professional audio devices’ [9]. As a consequence, it
provides a specification of protocol messaging, the parameter representation hierarchy, a detailed set of messaging structures, and details a set of significant abstractions.

It was found that there were parts of the specification where there was insufficient detail to enable requirements analysis. For example, the XFN specification states that ‘the nature of [an XFN] command [is] specified in the Command Executive and Command Qualifier fields [of an XFN message]’ [9]. However, while the XFN specification defines the valid identifiers for both the Command Executive and Command Qualifier fields of an XFN command, it does not define:

- the valid combinations of those identifiers
- the meaning of a given combination
- details of any data that must be provided with the command
- details of valid response(s) that a recipient may make.

Several techniques were used to capture requirements for an XFN stack implementation. These included packet analysis of communications on production XFN networks, and an extensive review of the UNOS Core XFN implementation. Packet analysis enabled the documentation of network discovery, device enumeration and other serialised procedures in XFN. The UNOS Core review was carried out over an extended period and enabled the specification of:

- XFN command semantics, including expected data and consequences (Table III–I provides an example of this, defining the meaning of three valid XFN commands; details of data and responses are omitted for space);
- the XFN representation schema, its component data structures, and their attributes;

in addition to enabling the resolution of discontinuities in the AES specification.

**Table III–1** XFN command definitions (extract)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET VTBL</td>
<td>Param Request. Gets the target parameter's value table, if it has one, from the XFN Application.</td>
</tr>
<tr>
<td>IS ALIVE</td>
<td>Device Request. Asks an XFN Device whether it is still in contact with the network.</td>
</tr>
<tr>
<td>JOIN MSTSLV</td>
<td>Param Request. Sets up a MSTSLV relationship between the addressed parameter and a supplied slave parameter.</td>
</tr>
</tbody>
</table>

The process of producing the requirements specification drew on the procedures outlined by the Real-Time Structured Analysis & Design (RTSAD) methodology [13]. RTSAD recommends the production of a ‘behavioural model’ based on identifying the boundaries and input-output events of the system to be designed. As such, developing a complete specification of XFN commands was paramount.

Ward & Mellor characterise structured analysis as an iterative process [13]. The process of developing the XFN implementation specification confirmed this characterisation. Individual approaches contributing to the analysis included code review and analysis of the UNOS Core implementation, behaviour capture and analysis of production XFN control systems, and iterative applications of selected modelling formalisms described by Ward & Mellor [13]. The products of these processes were combined and cross-referenced to verify the analysis.

The final implementation specification consists of:

- a complete data specification for an X170 device implementation, defining the composition, type and purpose of every attribute of device and parameter representations and every message field defined by the protocol
- a messaging specification that provides a synopsis, classifications and a message format for every valid XFN command and its associated response(s)

This process also produced an ‘essential model’ of an XFN protocol stack, comprising the following components:

- the aforementioned data specification, which defines every data structure by type or composition and exact meaning.
- a set of RTSAD-derived transformation schema that represents all data flows and control interactions within the XFN protocol stack, including timing constraints. The transformation schemas are ‘levelled’ by access to data stores to provide granular, functionally-grouped views of the transformations.
- a set of Program Design Language (PDL) process specifications for the components of the transformation schema.

![Figure III-1 An example of the transformation schema](image)
perfect communications. It provides a re-usable reference for future implementations of the XFN stack on any software platform.

B. Design for native AES XFN control over an XMOS Ethernet AVB endpoint

The completion of the XFN implementation specification provided a stable foundation for further design work. The Design Approach for Real-Time Systems (DARTS) methodology [14] was selected as defining accessible and efficient procedures for producing coherently structured modular designs with well-defined interfaces, a requirement that becomes increasingly relevant when developing software for an XS1 operating environment.

The transformation schema prepared as part of the implementation specification had been clustered and levelled by functional groupings as recommended in RTSAD practice. Combined with this practice, the linear procedures provided by DARTS mean that selective reductions (‘profiling’) of the original XFN features map directly from the XFN implementation specification to design and implementation.

IV. IMPLEMENTATION

During development of the XFN implementation specification, it became clear that a small subset of the XFN command space would be sufficient to provide connection management over an XMOS Ethernet AVB device.

In the case of the ‘most economical’ Ethernet AVB endpoint described in B, the audio functionality attached to the device is minimal: the Talker and Listener threads have access to one full-duplex two-channel audio interface. Since this design decision (justifiably) restricts the functionality of the endpoint as a whole, the prospect of providing a full-scale implementation of the X170 messaging stack for a device of this (low) level of functionality was unnecessary. Also relevant is that the available resources on this particular device configuration are tightly constrained, with fifteen of sixteen hardware threads already allocated to the baseline Ethernet AVB implementation.

As a result, a decision was taken to implement a small selection of XFN features on this endpoint, as a milestone to providing a full-featured XFN implementation on XMOS Ethernet AVB endpoints with greater resources. This approach has produced a ‘low-profile’ implementation of XFN control.

The low-profile implementation of XFN control is intended for use on devices with small feature sets. As such, some XFN features motivated by issues concerning more complex devices have been omitted from the implementation. In particular, wildcard parameter addressing, network discovery and device enumeration are not supported at this time.

Existing research [7] was consulted to review prior applications of the XFN representational schema to Ethernet AVB endpoints (Figure IV-1). These applications of the schema have been adapted and extended to provide further monitoring and control services, facilitating performance evaluations of Ethernet AVB networks.

Figure IV-1 Preliminary XFN representation of connection management for Ethernet AVB endpoints.

A control and prototyping application was developed using Python to support the development of the ‘low profile’ XFN implementation. This application incorporates a testing framework to validate the messaging and behaviour produced by the ‘low profile’ implementation against the XFN implementation specification.

The control application may perform connection management on any number of Ethernet AVB endpoints. As Figure IV-1 indicates, an Ethernet AVB endpoint’s connection management properties can be represented in four parameters. Further research has indicated that connection management can be satisfactorily performed using only three of these (Figure IV-2).

Figure IV-2 Revised XFN representation of connection management for Ethernet AVB endpoints.
XFN control of an XMOS for Ethernet AVB audio distribution, monitoring and connection management on a novel platform protocol, enabling the development of XFN implementation specification for the AES X170 Ethernet AVB networks. Further work is underway to implement a full-featured XFN protocol stack for XMOS-based Ethernet AVB endpoints.

REFERENCES


James Dibley is an MSc candidate and a member of the Audio Research Group in the department of Computer Science at Rhodes University. He received his undergraduate degree in 2003 from the University of Surrey.

Richard J. Foss is an Associate Professor in the department of Computer Science at Rhodes University. His research interest is the networking of audio and video devices for live concerts, sound installations and recording studios. Over the past three years he has chaired various networked audio committees within the Audio Engineering Society.