A Comparison of the bandwidth utilisation of Ethernet AVB and IEEE 1394b for streaming real-time audio with QoS using Network Simulation

Fred Otten and Richard Foss
Department of Computer Science
Rhodes University, Grahamstown, South Africa
Tel: 046 6038291 Fax: 046 6361915
g05o5894@campus.ru.ac.za, r.foss@ru.ac.za

***Abstract*—Network Simulation techniques can be used to evaluate and compare networks in different scenarios. IEEE 1394b and Ethernet AVB technologies can both be used for streaming real-time audio while maintaining Quality of Service (QoS). AudioNetSim is a network simulator designed specifically for simulating professional audio networks and provides support for both technologies. This paper uses AudioNetSim to compare the bandwidth utilisation of these two technologies using equivalent networks. It shows that, in general, Ethernet AVB networks are more efficient than IEEE 1394b networks for transmitting audio. At 100 Mbps, Ethernet AVB is able to transmit 11 additional audio channels. It also uses 12.975 percent less bandwidth to send a single stream consisting of 16 audio channels on a 100Mbps network.**

***Index Terms*—Modelling, Simulation, Bandwidth Calculation, Professional audio networks, IEEE 1394, Quality of Service**

I. Introduction

The use of digital multimedia networks for professional audio and video is increasing. The advantages of less cables, dynamic routing and flexible command and control make digital multimedia networks attractive. Professional audio networks are large audio networks used in convention centers, studios and stadiums. They consist of a number of nodes/devices, such as amplifiers, mixing desks and routers, which are connected together using Firewire (IEEE 1394a [21] or IEEE 1394b [18]) or Ethernet (CobraNet [8], Ethersound [5] and AVB [7]). The equipment used in these networks is expensive, so the networks need to be carefully planned by Audio Engineers before deploying them.

Network Simulation can be used to aid them in the designing process [17]. To ensure QoS, they need to take into consideration the bandwidth available and make sure the correct number of audio channels can be transmitted. They also need to consider which technology is best suited for the scenario. Network Simulation also allows the Audio Engineer to compare equivalent networks using different technologies.

This paper uses a network simulator designed specifically for professional audio networks to compare the bandwidth utilised when streaming real-time audio with guaranteed QoS. It begins by discussing network simulation and introducing the network simulator - AudioNetSim - which is a proof of concept implementation of a network simulation framework designed by the author. It then describes Firewire and Ethernet AVB networks and the calculations used by AudioNetSim to determine the bandwidth utilisation. It concludes by analysing the results for a 16 node network and discussing the differences between the technologies.

The work reported in this paper was undertaken in the Telkom Centre of Excellence in Distributed Multimedia at Rhodes University, with financial support from Telkom, Comverse, Tellabs, Stortech, Amatole Telecom Services, Bright Ideas 39, THRIP, DAAD and The Andrew Mellon Foundation

II. Network Simulation

Simulation is essentially the imitation of the operation of a real-world process or system over time. In the case of network simulation, it is the imitation of a given network's activity over time. One of the most important questions in network simulation is how much detail is necessary to accurately represent the network and obtain the required information. As the level of detail increases, the resource requirements increase, which can lead to scalability issues. This introduces a tension between large-scale simulation and realistic simulation.

Since simulation can occur at many levels, different levels of abstraction can be used. To provide an accurate simulation, we therefore need to consider only those aspects of the system that affect the problem under investigation. This is tailored to the requirements of the simulation - i.e. the metrics which we are interested in.

The aim of simulation is to produce an accurate representation of a system's activity/status so that inferences can be made about the use of different network designs and the use of different protocols in that network.

Either a model can be built by analysing the network and determining a set of equations to obtain the required metrics (termed an analytical approach), or a computer-based model can be constructed which mimics the activity of the network [4]. When creating a model, different levels of abstraction can be used depending on the level of detail and accuracy required and what information the simulation needs to produce [3].

A number of different techniques have been used to model networks. These include mathematical techniques (such as Markov Chains, Queuing Theory and Graph Theory) and Packet-based or Flow-based models of the traffic travelling over the wire. A number of simulation programs have been developed. Academic Examples include: REAL; INSANE; NetSim (detailed simulation of ethernet); Maisie (which is a C-based language for simulation); NS-2; Harvard Simulator; FlowSim; SSFNet; and NCTU-NS. Commercial examples include: BONeS, COMNET III, and OPNET. These simulation programs are either designed as generic simulations which are able to simulate a large number of different networks (such as NS-2 which can be used to simulate any network) or specialised network simulators which have a specific focus (such as NetSim which focuses on Ethernet networks). Most network simulation research has been performed on Ethernet networks [9], wireless

networks or the TCP protocol [2]. In our case, we are interested in determining the bandwidth for a specific protocol in a specific environment (a professional audio network). The use of a computer-based model or a generic simulation program means that after each alteration to the network state, the simulation needs to be redone using the new data in order to obtain the results. On a large network, this would take a large amount of time and would also calculate much unnecessary information. In an analytical approach, calculations can be performed to determine these metrics based on the network which has been designed, in near-real time for large networks. An example of this approach can be found in Otten and Foss [16] which shows how bandwidth can be calculated for a IEEE 1394b Firewire network. This is the simulation approach which we use since it meets the requirements of the audio engineer for near real-time determination of these metrics. In the next section, we briefly discuss the network simulator framework used and our implementation, AudioNetSim.

III. AudioNetSim

To simulate this environment and determine metrics analytically, we need to consider the network and gather sufficient information to calculate the metrics. To do this, a network simulation framework was developed which consists of four parts:

- Network Model
- Control protocol model
- Graphical User Interface
- Interface for Interaction with the simulator

The network model focusses on the physical, data link and transaction layers and essentially abstracts the network into a form which can use analytical methods to determine metrics useful to an audio engineer. The control protocol model focuses on the control aspect of the network. This separation allows for different control protocols to be used on top of the same network. The graphical user interface part focuses on how the device configuration is layed out by an audio engineer and how bandwidth utilisation can be viewed. This is considered separate from the network technology being used and builds a generic configuration. The interface for interaction with the simulator focusses on different approaches which can be used to interact with the simulated network. This interface enables the audio engineer to use the control protocol to modify parameters, create connections and perform grouping operations on the simulated network. It hence allows different control applications to be used with the simulated network. The framework provides a platform for any professional audio network simulation to be developed. The divisions ensure extensibility and the components are designed to interact with each other seamlessly. It allows easy implementation of different networks and protocols due to this separation.

AudioNetSim is an implementation of this framework for simulating professional audio networks using the XFN control protocol. It consists of four main components:

- Network Models for Firewire and Ethernet AVB networks
- XFN Control Protocol Model
- AudioNetSim Graphical User Interface
- Interface to UNOS Vision [24] Control Application

AudioNetSim was created to meet the requirements of an Audio Engineer. It utilises a GUI and includes a library of devices for the engineer to choose from. Should the device not be found in the library, it can be retrieved from a live network and added to the library. It also includes the ability to group devices and connections into a single entity termed a "cloud". This allows the audio engineer to create and store their racks of devices and makes it possible to easily build large configurations, while keeping the GUI from being too cluttered.

An object model is used to create a virtual representation of the network and its devices, which are XFN capable. When a device is inserted, it is added to the simulated network and an XFN parameter tree is created for that device. This parameter tree is a representation of the parameters in the XFN stack. AudioNetSim is integrated with UNOS Vision, which is a powerful XFN control application which can be used for connection management, command and control [24].

When a parameter is modified within UNOS Vision, instead of sending an XFN message to the device, a call is made to the simulated network with the necessary device information and the value in the device's parameter tree is modified or returned in the same manner as on a real device. This is described further in Otten and Foss [17].

In this paper, two equivalent virtual devices (one Firewire and one AVB) are utilised. AudioNetSim allows an Audio Engineer to build the configuration for the two networks using these devices and compare their bandwidth utilisation. This is described further in Section VI.

IV. Firewire Networks

Firewire is a high speed serial bus based on CSR architecture - ISO/IEC 13213 (ANSI/IEEE 1212) "Control and Status registers architecture for microcomputer buses" [19]. It was standardised by the IEEE as IEEE 1394 in 1995 [20]. Subsequent revisions occurred in 2000 (IEEE 1394a) [21], 2002 (IEEE 1394b) [18] and 2006 (IEEE 1394c) [22].

Firewire networks consist of a number of buses (up to 1024) which are joined together using bridges or routers. A bus can contain up to 64 nodes. These nodes are logical entities on firewire devices (which are also termed modules). Each node can in turn contain a number of ports which can be used to daisy chain devices. One of the attractions of the firewire bus is the dual transmission modes - Asynchronous and Isochronous. Asynchronous transmission allows devices to read and write from/to an addressed location periodically with guaranteed delivery, while isochronous transmission allows devices to perform real-time streaming of data by sending data every 125μs. Isochronous data can use up to 80 percent of the available bandwidth (100μs). Each device is able to transmit a number of sequences within isochronous streams, which are allocated channel numbers.

With the IEEE 1394b project, the IEEE aimed to overcome the limitations of IEEE 1394a (short cable length and bandwidth wasted by the use of idle gaps) in order to broaden the scope of IEEE 1394 and make the interface more valuable to the end user. The IEEE 1394b bus is a backwards compatible bus created to support faster speeds than 400 Mbps (which is supported by IEEE 1394 and IEEE 1394a specifications). This bus supports speeds of 800 Mbps and was created to support up to 3.2 Gbps. It also supports a number of different mediums including: optical, coaxial and twisted-pair cabling. The use of beta mode signalling rather than data strobe signalling also means that full duplex transmission is possible, since a strobe does not need to be transmitted on the second pair. IEEE 1394b still maintains the ability to do data-strobe signalling to ensure legacy compatibility.

A number of arbitration enhancements such as fly-by concatenation, ACK accelerated arbitration, multispeed concatenated packets and priorities are included in IEEE 1394a [1].

These arbitration techniques, however, still include sub action gaps and hence do not make optimal use of the available bandwidth. IEEE 1394b solves this problem by defining a new arbitration technique which takes advantage of the full duplex nature of beta mode signalling. This technique is called Bus Owner/Supervisor/Selector (BOSS) arbitration. In BOSS arbitration, the request signalling is overlapped with data transmission, which removes the need for idle gaps. By removing these idle gaps, extra bandwidth is made available for data transmission.

In legacy arbitration, each device sends a request to a fixed root which determines the winner of arbitration. This means that devices which are further away from the root have to wait longer for arbitration requests to be completed. The last node to transmit is usually in the proximity of the next node to transmit, so instead of having a fixed root which determines the winner of arbitration, BOSS arbitration uses a variable node. When a node begins transmission, it becomes the BOSS. It is the only node in the bus that can be receiving arbitration requests on every active port and is therefore in the best position to decide which node to issue a grant to.

In this paper, the firewire devices which are simulated are IEEE 1394b devices which use BOSS arbitration. Note that in this paper, unless otherwise specified, Firewire refers to IEEE 1394b networks utilising BOSS arbitration.

A. Bandwidth Calculation

When calculating bandwidth, a knowledge of the packet format is essential to determining the overhead. The packet consists of 3 parts - the packet prefix, the packet itself and the packet end.

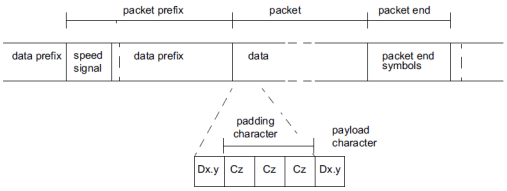


Figure 1: Packet Format

Figure 1 shows a depiction of the IEEE 1394b packet format. The packet prefix incorporates the speed signal followed by a number of DATA_PREFIX tokens. The speed signal consists of a number of control tokens. These tokens are utilised to indicate the packet speed (relative to the port operating speed) to the receiving node. They are also used to indicate whether the packet format is legacy or beta (which it is in this case). For any particular packet speed the speed signal has a constant duration.

In IEEE 1394b, the port operating speed remains constant for all packet speeds. In the case where the packet speed is less than port operating speed, other characters (padding characters or packet delimiters) are transmitted with the payload characters as indicated in Figure 1.

Following the speed signal, a number (p) of DATA_PREFIX symbols are transmitted. In a beta-only network, p is the ratio of the port operating speed to the packet speed (for all packet speeds). A number (e) of DATA_PREFIX symbols are also transmitted after the p DATA_PREFIX symbols to compensate for clock frequency differences. These symbols are called deletable symbols. They are introduced such that the duration is at least 2 symbols at the packet speed (or the duration of 2 symbols at 400 Mbps for packet speeds faster than 400 Mbps).

The packet end consists of a stream of packet end symbols. In a beta-only network, a number (n) of packet end symbols are transmitted at port operating speed where n=2*g (g is the ratio of the port operating speed to the operating speed of the port being transmitted to) if the port being being transmitted to has a port operating speed less than the transmitting port, otherwise n=2*m (m is the ratio of the port operating speed to the packet speed).

The duration of the packet prefix and packet end in a beta-only network is less than their duration in an IEEE 1394a network. This, along with the lack of idle gaps, are the reasons why there is more bandwidth available for transmitting data in a beta-only network than in an IEEE 1394a network.

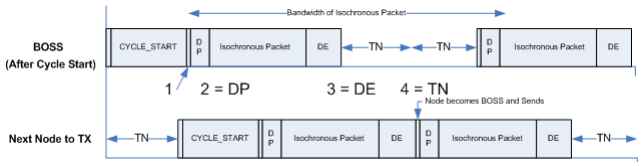


Figure 2: Isochronous Transmission with BOSS Arbitration

Figure 2 shows the isochronous interval from the perspective of two devices - The BOSS and the next node to transmit.

There are a number of overhead factors which need to be taken into account. They are as follows:

- The time taken to transmit the Speed signal (1)
- The time taken to transmit the Data Prefix (2)
- The time taken to transmit the Data End (3)
- The delay to next transmitting node (4)

These factors are labeled in Figure 2 using the numbers indicated in brackets.

As mentioned, a speed signal is included in the packet prefix. The duration of this speed signal is $\frac{8000}{x}$ ns where x is the packet speed in Mbps. There is no information on the PHY delay in the IEEE 1394b spec. It is, however, faster than the PHY delay in a IEEE 1394a node, and varies from manufacturer to manufacturer. For the purposes of this calculation, the same value is used as is used for IEEE 1394a (144ns) since we know that the PHY delay for IEEE 1394b is less than or equal to this value. The packet end consists of a ISOCH_GRANT which is sent to the next node to transmit.

The Isochronous packet contains audio data and packet headers. The packet headers consist of a CIP header for each stream (20 bytes). Certain audio chips, such as the DICE II chips utilised by the firewire evaluation board [23] only send data when the buffer is full (8 data blocks), this leads to 32 bytes being reserved when transmitting audio at a sampling rate of 48KHz.

The overhead can vary according to the topology since this will effect the delay to the next transmitting node. Consider an overhead of o ns.

For the transmission of a stereo stream at 48 KHz using 400 Mbps IEEE 1394b, the amount of bandwidth reserved would be $\frac{8o}{20.35} + 20 + (2 \times 32)$ bytes for every cycle (125 μs)

More information on this bandwidth calculation and the calculation of overhead can be found in Otten and Foss [16].

V. Ethernet AVB Networks

Ethernet AVB is an extension of current Ethernet (802.1) networks to provide quality of service at the data link layer (layer 2 in the ISO Stack). Before the development of Ethernet AVB, there were only proprietery ethernet solutions such as Dante, CobraNet and EtherSound which provided Quality of

Service. Quality of Service has, however, also been made possible in Ethernet using higher level protocols such as RSVP. In order for Quality of Service to be provided, we have to have the ability to allocate resources, as well as ensure that timing and synchronisation can be performed.

The IEEE 802.1 Audio Video Bridging Task Group is responsible for developing a set of standards commonly known as audio video bridging (AVB). These specifications aim to allow for time-synchronised low-latency streaming services to take place through bridged IEEE 802 networks. These standards augment the current Ethernet standards to provide Quality of Service and provide a layer 2 method of ensuring that there is sufficient bandwidth within an Ethernet (802.1) network. The following standards form part of Ethernet AVB:

- 802.1Qat - resource reservation [12]
- 802.1Qav - forward and queueing of time sensitive streams [13]
- 802.1 AS - timing and synchronisation [11]
- 802.1 BA (which is currently still under standardisation) - features, options and configurations for AVB systems [14]

This section introduces Ethernet AVB networks and discusses the various protocols which are used to ensure that the audio will be delivered.

AVB Networks are essentially an extension of standard Ethernet (802.1) networks. The main difference between standard ethernet networks and AVB networks are that AVB networks provide precise synchronization, traffic shaping to ensure that media streams are delivered and only utilise the resources allocated to them, admission controls for devices, and the identification of non-participating devices. This is done by introducing extra features to the MAC layer of Ethernet networks.

For additional information, Foulkes and Foss [7] and Foulkes [6] provide extensive reviews of Ethernet AVB.

Ethernet AVB networks are a form of bridged Ethernet networks [10]. A bridge is used to connect together different LANs. These can be using different physical layers or media access control entities. A bridged LAN refers to a number of these LANs which are interconnected using bridges.

Ethernet AVB uses bridged Virtual LAN (VLAN) networks. In a VLAN, the end points communicate with each other as if they are attached to a single broadcast domain regardless of where they are situated on the network or how many bridges are between them in the bridged LAN. These VLANs are uniquely identified by a VLAN identifier (VID). A bridge in the network can also be referred to as a switch. Information about the VLAN is contained in a VLAN tag which is appended to the Ethernet frames.

Ethernet AVB frames are standard Ethernet frames which have a VLAN tag appended to it [10]. The use of VLAN tags allow the priority information to be carried within the frames. This is done using the VLAN tag's priority code point (PCP) field. These priority values are used to classify the streams into different traffic classes which make it possible for its transmission requirements to be met. Ethernet AVB frames are transmitted on VLANs, with the default value of the VID being 2.

The Ethernet AVB specifications also detail a number of protocols which are used to register streams, manage streams and broadcast their characteristic through-out the network. A protocol called Multiple Registration Protocol (MRP) is defined in 802.1Qat [12]. This protocol allows the registration of attributes on a port and allows applications to manipulate and interpret these attributes. A number of MRP applications - Multiple Stream Reservation Protocol (MSRP), Multiple VLAN Registration Protocol (MVRP) and Multiple MAC Registration Protocol (MMRP) are used to all to facilitate the allocation of bandwidth for realtime streams. To send streams, the terminology Talker and Listener is used for the endpoint which sends the stream and the endpoint that recieves a stream respectively. 802.1QAS defines mechanisms to ensure timing and synchronisation of audio which is transmitted between the Talker and Listener.

Using these MRP applications and the timing and synchronisation mechanisms, AVB Networks are capable of transmitting both realtime and non-realtime data while guaranteeing quality of service for streams which are time sensitive. Ethernet AVB limits the amount of bandwidth which can be allocated for realtime streams to 75 percent of the total available bandwidth. This ensures that bandwidth is still available to transmit other traffic (such as command and control data) on the network.

A. Bandwidth Calculation

A field calles the TSpec within the Talker attribute used in MSRP contains the traffic specification for the stream. This consists of the maximum frame size and the maximum frame rate for a given stream. From this, the actual bandwidth required can be calculated by taking into account the per frame overhead.

The following calculation can be used: Actual bandwidth = (per frame overhead + maximum frame size) × maximum frame rate. The maximum frame size and the maximum frame rate can be determined from the TSpec for the stream, however the per frame overhead needs to be calculated based on the headers which need to be transmitted. In Ethernet AVB, streams are transported using audio/video transport protocol (AVTP), which is described in IEEE 1722 [15], while audio samples are packaged using IEC 61883-6. These packets are encapsulated in Ethernet frames which are VLAN tagged. The overhead associated with Ethernet VLAN tagged frames is as follows:

- Inter frame gap (IFG): 12 bytes
- Preamble: 8 bytes
- Ethernet header (with VLAN tag): 18 bytes
- Trailer: 4 bytes

This is a total of 42 bytes for Ethernet VLAN related headers. To transmit streams using AVTP, an AVTP Common header and AVTP Common Stream Header needs to be transmitted. The overhead for these headers is as follows:

- AVTP Common Header: 12 bytes
- AVTP Common Stream Header: 10 bytes

This is a total of 22 bytes for AVTP headers. The streams are transmitted using IEC 61883-6 which requires a CIP header to be transmitted. This introduces an addition overhead of 10 bytes. This leads to a grand total overhead of 74 bytes for each stream packet.

The calculation is therefore: Actual bandwidth = (74 + maximum frame size) × maximum frame rate

The total packet size for the transmission of a multicore would be $74 + xn$, where x is the amount of streaming data bytes transmitted at the requested frame rate and n is the number of streams being transmitted. The maximum frame size in this case is xn. The frame rate depends on the traffic class which is being used. In traffic class A, packets are transmitted every 125 μs (i.e. a frame rate of 8000 per second) with a maximum of 7 hops and a maximum latency of 2 μs, while in traffic class B, packets are transmitted every 250 μs (i.e. a frame rate of 4000 per second) with a maximum of 9 hops and a maximum latency of 20 μs.

The transmission of a stereo 48KHz stream using traffic class A translates into 24 bytes being sent 8000 times every second for each of the two channels. In this case, the actual bandwidth would be $(74+(24\times2))\times8000 = 976000$ bytes every second. This is equivalent to 0.931 MB/s. Within Ethernet AVB, 75 percent of the total bandwidth can be reserved for streams, which means that 100.698 of these stereo 48Khz streams could be sent on a gigabit ethernet link.

VI. Comparison Network and Testing Methodology

The maximum amount of evaluation board devices which can be daisy-chained in a firewire network using the evaluation boards is limited to 16 devices by the manufacturer. We therefore construct a network consisting of 16 evaluation boards using AudioNetSim. Each of these evaluation boards has the potential to output 2 streams each with 16 audio channels. This network makes it possible to saturate the network links at all speeds and provide a comparison between firewire and Ethernet AVB networking technologies. Audio streams are routed between devices by connecting multicores (which represent an audio stream that consists of a number of channels) using UNOS Vision. UNOS Vision is also used to increase the amount of audio channels within each stream.

For our testing purposes, all audio channels are mono, 48KHz. Within firewire networks, a single multicore connection is made since each device is only able to receive two streams. This causes the stream to be broadcast over the entire network. Within AVB networks, however, connections are made to every device since the bandwidth needs to be evaluated at an end point rather than on a subnet as is the case for firewire. To test the network, connections are made in such a manner that a device is broadcasting.

The firewire and AVB networks are setup within AudioNetSim with the firewire devices being daisychained and the AVB endpoints all being connected to a single switch. UNOS Vision is then utilised to make multicore connections. Once the connection is created, the number of audio channels is increased until the maximum, 16, is reached. At each point, the bandwidth utilisation is recorded. In this manner, a matrix is constructed with the number of streams (1-16) and the number of channels (1-512). These results will be analysed in the next section.

VII. Results and Analysis

This section presents and analyses the bandwidth utilisation results for the Firewire and Ethernet AVB networks obtained from AudioNetSim using the networks and testing methodology presented in the previous section.

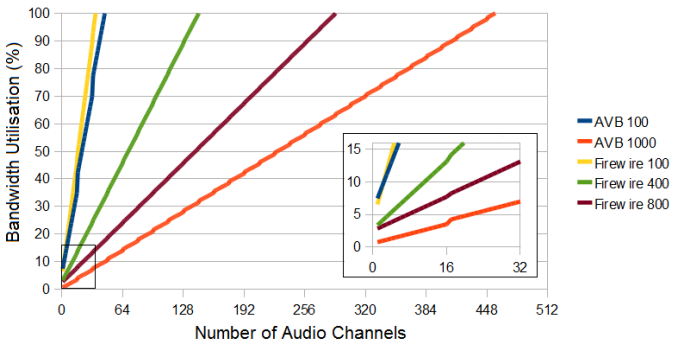


Figure 3: Bandwidth Utilisation when transmitting minimum amount of streams

Figure 3 shows the percentage of bandwidth utilised when transmitting a certain number of audio channels (shown on the x-axis) utilising the minimum amount of streams using each of the technologies. Each stream from the evaluation board is able to contain up to 16 audio channels. The inset figure shows a zoomed-in view of the bandwidth utilised when transmitting less than 32 channels. This figure shows that for equivalent speeds (i.e at 100 Mbps), Ethernet AVB utilises less bandwidth than Firewire when there is more than 2 audio channels being sent.

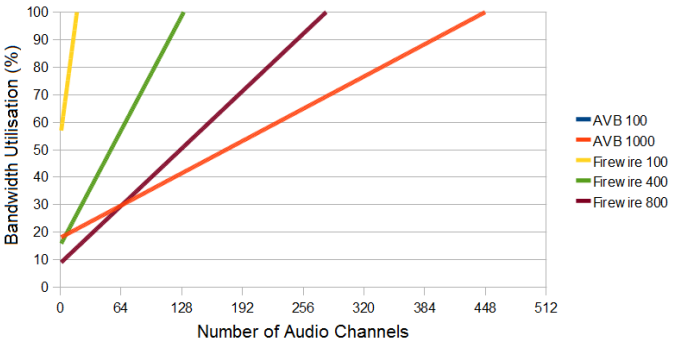
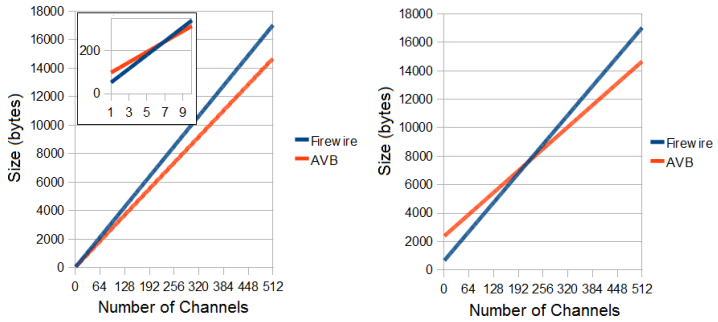


Figure 4: Bandwidth Utilisation when transmitting 32 streams

Figure 4 shows the percentage of bandwidth utilised when transmitting a certain number of audio channels (shown on the x-axis) utilising 32 streams using each of the technologies. This figure shows that when transmitting few audio channels within a stream and multiple streams, Firewire can utilise less bandwidth than Ethernet AVB. When transmitting 32 streams, Firewire 800 utilises less bandwidth than Ethernet AVB running over Gigabit Ethernet for up to 64 channels.



(a) Transmitting minimum amount of streams

(b) Transmitting 32 streams

Figure 5: Amount of data sent each 125μs

Figure 5a shows the amount of data (in bytes) transmitted every 125μs when transmitting a certain amount of audio channels utilising firewire and ethernet AVB and the minumum amount of streams. The inset figure shows a zoomed-in view of the data transmitted when transmitting less than 10 streams. This figure shows that the amount of data sent by Ethernet AVB when transmitting the miniumum amount of streams is less than the amount of data sent by firewire when transmitting more than 7 channels.

Figure 5b shows the amount of data (in bytes) transmitted every 125μs when transmitting a certain amount of audio channels utilising firewire and ethernet AVB and 32 streams. This figure shows that the amount of data sent for firewire is less than the data sent for Ethernet AVB for up to 216 channels when utilising 32 streams.

The gradient of the lines for firewire is steeper than the gradient for AVB in both Figure 5a and Figure 5b. This means that as the amount of audio channels increase, the increase in data sent is greater for firewire than it is for AVB. This is because the data reserved (i.e. utilised within a cycle) is 32 bytes per audio channel rather than 24 bytes for Ethernet AVB. This is because the firewire utilises a blocking method for transmission, i.e. data is only sent once eight samples are in the buffer. This means that in some isochronous cycles, data is not sent, however it is reserved and hence considered utilised. In Ethenet AVB, traffic class A is utilised which sends 8000 packets per second. The streams are not transmitted using a blocking mechanism every 125 μs as in firewire, but rather as they are available, and if the traffic shaper algorithm has a positive or zero credit.

In terms of overhead per stream, Ethernet AVB has significantly larger packet headers than Firewire. Firewire does, however, have additional overhead such as data begin and data end tokens in which the data is encapsulated. This explains the reason why firewire utilises a smaller percentage of bandwidth than Ethernet AVB when multiple streams are being sent with minimal channels of audio per stream.

	Firewire			Ethernet AVB	
	100	400	800	100	1000
% 1 stream (16 channels)	47.909	15.348	10.026	34.934	3.493
max. channels < 50%	16	66	133	21	226
max. channels < 100%	34	140	281	45	456

Table I: Summary of results

Table I shows a summary of the bandwidth utilisation for Firewire and Ethernet AVB. It shows the maximum number of audio channels that can be transmitted before the bandwidth utilisation exceeds 50 percent and also before it reaches 100 percent. It also shows the percentage of bandwidth which is utilised when transmitting a single streams with 16 audio channels. This table shows that Ethernet AVB networks are able to transmit a greater amount of audio channels than equivalent Firewire networks. It is able to transmit 11 more channels than Firewire at 100 Mbps. The percentage bandwidth utilised for transmitting a single stream with 16 channels is 12.975 percent lower for Ethernet AVB.

From the results presented, we can conclude that the firewire network is more efficient when transmitting multiple streams with few channels to all devices. This might occur in an audio network for digital home audio. Generally, Ethernet AVB utilises less bandwidth than Firewire. It also has other advantages - Ethernet AVB is able to utilise existing Ethernet infrastructures and streams can be transmitted point-to-point and are not limited to being broadcast to every device.

VIII. Conclusion

AudioNetSim is network simulator designed by the author for simulating aspects of professional audio networks and calculating metrics such as bandwidth utilisation. It has been used to simulate both Firewire and Ethernet AVB networks. This paper has given a brief introduction to these technologies and the analytical calculation which are used by AudioNetSim to calculate bandwidth utilisation. In this paper, AudioNetSim was used to compare the bandwidth utilisation of these two networking technologies. This paper has presented the results of this comparison using equivalent networks. It showed that, in general, Ethernet AVB is more efficient then Firewire networks for transmitting audio. It also has a number of advantages, including only broadcasting streams to the listeners which saves bandwidth and the ability to utilise existing Ethernet infrastructure. At 100 Mbps, Ethernet AVB is able to transmit 11 additional audio channels. It also uses 12.975 percent less bandwidth to send a single stream on a 100Mbps network. Firewire is only shown to be more efficient than Ethernet AVB when transmitting multiple streams with few channels to all devices

References

[1] Don Anderson. *Firewire System Architecture*. PC System Architecture Series. Mindshare, 2nd edition, October 2000.

[2] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving simulation for network research. Technical Report 99-702, University of Southern California, March 1999.

[3] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *Computer*, 33(5):59–67, May 2000.

[4] Xinjie Chang. Network simulations with OPNET. In *Proceedings of Winter Simulation Conference*, volume 1, pages 307–314, Phoenix, Arizona, United States, December 1999. ACM Press.

[5] Digigram. An Introduction to the ES-100 Technology Rev. 3.0b, August 2006.

[6] Philip Foulkes. *Audio Video Bridging Networks*. PhD thesis, Rhodes University, 2011.

[7] Philip J. Foulkes and Richard J. Foss. Providing Interoperability of, and Control over, Quality of Service Networks for Real-time Audio and Video Devices. In *South African Telecommunications and Networking Applications Conference*, 2009.

[8] Kevin Gross. CobraNet Technology Datasheet. Technical report, Peak Audio, April 2001.

[9] Alefiya Hussain, Aman Kapoor, and John Heidemann. The effect of detail on ethernet simulation. In *Proceedings of Principles of Advanced and Distributed Simulation*, pages 97–104, Kufstein, Austria, May 2004. ACM Press.

[10] IEEE. 802.1q: Virtual bridged local area networks, 2005.

[11] IEEE. 802.1as: Timing and synchronisation for time-sensitive applications in bridged local area networks, 2009.

[12] IEEE. 802.1qat: Stream reservation protocol, 2009.

[13] IEEE. 802.1qav: Forwarding and queueing enhancements for time-sensitive streams, 2009.

[14] IEEE. 802.1ba: Audio video bridging (avb) systems, 2010.

[15] IEEE. Draft standard for layer 2 transport protocol for time sensitive applications in a bridged local area network, 2010.

[16] Fred Otten and Richard Foss. Ensuring QoS in Professional Audio Networks which use IEEE 1394b. In *Proceedings of South African Telecommunications and Networking Applications Conference*, 2010.

[17] Fred Otten and Richard Foss. Enhancing the Configuration and Design of Sound Systems through Simulation. In *Proceedings of the 130th Audio Engineering Society Convention*, London, UK, May 2011. Audio Engineering Society.

[18] IEEE Society. IEEE Standard for a Higher Peformance Serial Bus - Amendment 2, January 2002.

[19] IEEE Computer Society. ISO/IEC 13213 (ANSI/IEEE 1212) - Control and Status Registers (CSR) Architecture for microcomputer buses, October 1994.

[20] IEEE Computer Society. IEEE Standard for a High Peformance Serial Bus - Firewire, January 1995.

[21] IEEE Computer Society. IEEE Standard for a High Peformance Serial Bus - Amendment 1, January 2000.

[22] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus - Amendment 3, January 2006.

[23] TC Applied Technologies. *Digital Interface Communications Engine (TCD2210/2220) Users Manual - Revision 0.14*, February 2007.

[24] UMAN. UNOS Vision. Online: http://unosnet.com/unosnet/index.php/unos-vision.html (Last Accessed: 10 March 2011), 2011.

Fred Otten holds a Master of Science Degree in Computer Science from Rhodes University. He is currently reading for a PhD in Computer Science at the same institution and has a keen interest in real-time multimedia and network simulation.