Daisy-chaining AVB

IN THIS DOCUMENT

AVB, Audio Video Bridging over Ethernet, is a set of IEEE standards for transporting Audio and other real-time content over Ethernet. The standards have been adopted by more than 20 manufacturers of FPGAs, microcontrollers, and switch silicon.

AVB is often purported to only serve large-scale applications, such as music venues. In this article we argue that AVB is excellently suited to small-scale applications, such as consumer audio, audio conferencing, or in-car entertainment. For this, we advocate the used of *daisy-chained AVB*: it avoids the need for switches, at the expense of reducing the capacity of the AVB system.

In this article we first give an overview of AVB. After that, we discuss how AVB daisy-chaining works, and show an example daisy-chained network.

1 AVB in a nut-shell

From a high level perspective, AVB works by *reserving* a fraction of the available Ethernet bandwidth for audio traffic. AVB packets are sent regularly in the allocated slots, and as the bandwidth is reserved, there will be no collisions. All nodes in the system share a virtual clock, and AVB packets have a *presentation time* which defines when the audio should be played out.

So, for example, a system may comprise a host-node that is delivering data (the *Talker*) two nodes that comprise the left and right speakers (the *Listeners*) and as all three nodes share a single global clock, the left and the right speaker will produce sound synchronously.

1.1 Reserving bandwidth: the Stream Reservation Protocol (SRP, IEEE 802.1Qat)

The magic behind AVB is that it splits traffic on the network into two groups: real-time traffic and the rest. All real-time traffic is transmitted on an 8 kHz beat, and the rest is scheduled around it. That is, every 125 us all real-time streams send their data; holding up the other traffic, and when no more real-time data is available, other packets are transmitted. This is visualised in Figure 1.

Publication Date: 2015/3/17 Document Number: XM003718A

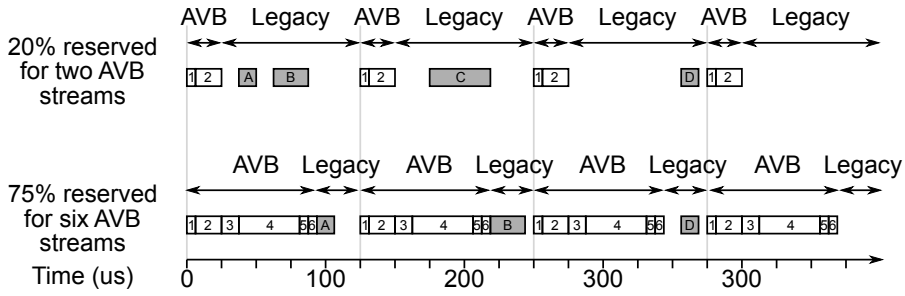


Figure 1: Two traffic shaping scenarios. Top: 20% is reserved for AVB; IP and other legacy traffic is shaped around the AVB slots. Bottom: 80% is reserved for AVB; legacy traffic is delayed and or dropped if it cannot be shaped around the AVB slots.

In order to ensure that there is sufficient room available for all real-time traffic, a protocol is used to *allocate bandwidth*. Figure 2 shows a system comprising two switches and four nodes: nodes A and D reserve a stream between them (say 45 mbit/s), and nodes B and C reserve another stream (say 20 mbit/s). All switches in between those nodes will make sure that sufficient bandwidth is available: 65 mbit/s will be reserved between switches X and Y since both the traffic from A to D and B to C will travel over this link. If this happens to be a 100 mbit/s link, then only 35 mbit/s is available for other traffic, such as web-traffic or configuration messages. If a large web page is requested at D from A, then packets may be dropped at X.

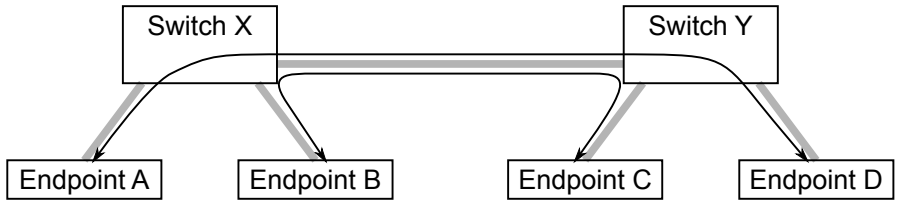


Figure 2: Diagram showing two switches with four nodes

Using allocated bandwidth enables AVB to send data from endpoint to endpoint within a 2 ms window: AVB allows for a maximum of 7 hops to meet this constraint, where each hop adds at most 125 us delay. This means that a node can transmit

audio requesting that it be played 2 ms in the future, and all samples will arrive in time to be played out at the right time.

The protocol for allocating bandwidth is called the Stream Reservation Protocol (SRP, IEEE 802.1Qat) and this forms a fundamental building block of the AVB standard. All nodes in the system (switches and endpoints) must implement SRP and shape traffic by sending real-time traffic at the 8 KHz beat. If one of the nodes was a legacy switch, then it would not treat real-time traffic preferentially, potentially delaying the real-time traffic, and causing jitter in the output.

1.2 The global clock: Precision Time Protocol (PTP, IEEE 802.1AS)

All audio traffic in AVB is synchronised to a global clock; this enables producers and consumers of audio to play and record sound synchronously. The clock is implemented by the *Precision Time Protocol*, or PTP.

PTP assumes that all nodes have a reasonably good clock (say a crystal clock), preferably of a known accuracy (say 25 ppm, equivalent to 2 seconds per day). PTP nodes that are connected using an ethernet cable send regular messages to each other, reporting the time, and calculating the skew between their respective clocks. The node with the most accurate clock is picked as a 'Master' node and all other nodes now estimate their skew relative to the Master clock, enabling all nodes to compute a local clock that is closely kept in sync with the Master clock.

Synchronising the clocks over the network comes at a price. Suppose that a node has an instable clock (for example because it is temperature sensitive), and its frequency is changing rapidly. This node will observe that its frequency is changing relative to the Master clock. It can either gently adjust the local clock to match the new frequency, but this will temporarily cause a phase difference between the Master and the local clock. Alternatively, the frequency can be adjusted faster, but this creates a higher-frequency jitter in the clock signal. For audio, one typically allows for a small temporary phase drift, keeping the jitter at very low frequencies.

The PTP protocol is specified in IEEE standard 802.1AS and is a second building block of AVB. It is also commonly used by networked computers (laptops, servers) in order to provide a synchronised clock.

1.3 Streams, Channels, Talkers and Listeners

AVB is build around *streams of audio*. A stream comprises multiple channels (for example stereo), and each AVB packet contains 125 us worth of samples for all channels that are part of the stream. Streams are produced by *Talkers*; the nodes that produce audio. A microphone or a laptop playing MP3 files are Talkers. *Listeners* can subscribe to a stream: a speaker is an example Listener that will typically pick a single channel out of a stream and play it out.

A typical system may comprise, for example:

- A single talker (a DVD player) with 6 listeners (for 5.1 surround sound)
- Multiple talkers (a group of microphones) with a set of speakers, for conferencing

- A few tens of microphones, a few dozen speakers, and a massive mixing desk (for a music venue)

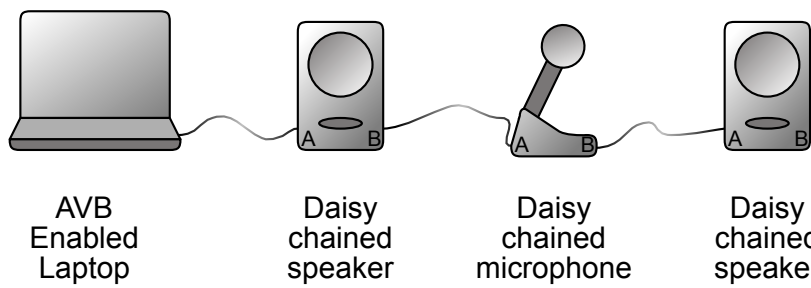
There are no rules on how small or large an AVB system should be. However, there are practical limits: AVB streams have a sizeable overhead, limiting the number of streams that an ethernet cable can carry. A 100 mbit Ethernet cable can carry 9 stereo AVB streams (for a total of 18 channels), or a single AVB stream with 45 channels.

A discovery protocol (IEEE 1722.1) is used to enumerate, discover and control attached devices and their capabilities. This protocol is detached from the actual delivery of data, and is purely used by a host to configure the system.

2 Daisy chaining

Compared to other mechanisms of digital audio distribution (such as USB audio), AVB appears expensive because of the need of AVB-aware switches. For this reason we argue the case for daisy chained AVB: an AVB endpoint with two Ethernet ports (we call them A and B), and a built-in "switch"; quoted as this is not a fully-fledged switch.

Figure 3: Diagram showing an AVB enabled laptop that has an Ethernet port plugged into the daisy chain comprising two speakers and a microphone.



An example layout is shown in Figure 3. A laptop is connected to node 1, which is connected to node 2, which is connected to node 3; where the network ends. Each node comprises two ports (that are symmetrical), and logic that connects the ports as follows:

- If only one port is plugged in, the node acts as an ordinary AVB endpoint.
- If both ports are plugged in, the node mostly acts as a bridge across the two ports: all traffic is passed through as normal. The node itself will tap into any AVB streams that are passing through the device, and occasionally the node will consume or produce a packet: for example when responding to any of the SRP, PTP, or configuration protocols.

This means that the node needs very little in terms of switching capacity. Data that comes in on port A will go to B unless it is destined for the local node, and traffic that comes in on port B will go to port A unless it was destined for the local node. Occasional packets may be generated locally, and the node must have knowledge as to whether these packets should go to A or B. The software that bridges A and B has to be AVB aware, and has to participate in, for example, clock synchronisation.

Note that neither routing tables nor buffers are required, and no operating system is needed to implement something that simple. This means that cost-wise, a daisy chained AVB endpoint is little more than the cost of a normal AVB-endpoint plus an extra Ethernet PHY and jack.

There are limitations to this approach

- Unlike a switch, a daisy chained network requires that traffic destined for the tail travels through the whole daisy chain; in a switch with 7 nodes, all 7 nodes can in theory receive 100 Mbits of traffic. In a daisy chained system, that would require the head of the node to transport 700 Mbits/s. However, in an AVB system most traffic is multicast audio traffic, and very little traffic is destined to specific nodes. So where the nodes on the chain listen to the same stream, there is little extra traffic in a daisy chain.
- A second limitation is that the AVB standard does not allow for more than seven switches in a network, in order to guarantee a 2 ms end-to-end latency. This limits a single daisy chain to seven nodes. There are two ways around it: first, one can forego the 2 ms guarantee in a closed system. Second, one can use a switch with daisy chains. If a daisy chain of four nodes is connected to each port of the switch, four times as many nodes can be used on a switch, reducing the cost of the infrastructure required.

Because of these limitations, daisy chained AVB is well suited to deal with small scale systems.

3 Example daisy chain implementation

We have developed a daisy chain AVB node on the basis of an XMOS chip with 16 logical cores. The hardware and software architecture of the system is shown in Figure 4. The hardware used for our system comprises:

- An xCORE multicore microcontroller with 16 logical cores
- Two ethernet PHYs with magnetics and jacks
- A low jitter PLL for word-clock generation
- A CODEC with input and output stages

The microcontroller runs seven tasks to control the two Ethernet ports; inputting packets, outputting packets, and routing packets between the two ports. Another six tasks implement the AVB stack, these are the Talker/Listener, PTP and Media Clock recovery, I2S control, SRP/MRP, and 1722.1 discovery and control tasks. All

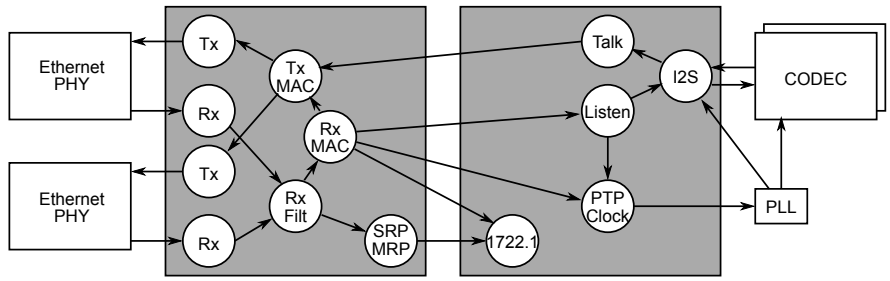


Figure 4: Software and hardware architecture of our example daisy chained AVB solution

13 tasks fit in 128 kByte of on-chip memory, obviating the need for external RAM. An external Flash chip is used to hold persistent data and the boot image.

The software is very similar to the software found in high-channel count AVB products - the only part that differs is the MII interface and buffering.

We have constructed the system using a XMOS sliceKIT with two Ethernet slices and an Audio-slice, a photo of this system is shown in Figure 5. Here the AVB system is connected to a laptop that uses the two nodes as a “left” and “right” channel. (Note that our audio slice comes with MIDI, dual stereo input and dual stereo output as default; for this demonstration we only use a single audio output.)

The laptop can discover the two nodes, and we can redirect our audio output to the two speakers. A scope probe on the two channels show that the two channels are playing without a discernible phase difference. The same hardware/software architecture can be used to, for example, build a conference system, or to drive a P/A system.

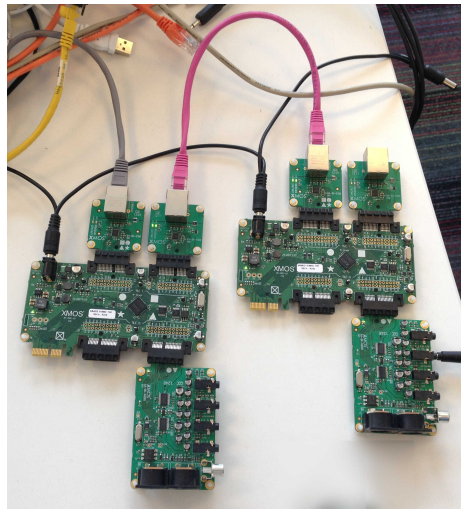


Figure 5: Prototype daisy chain

4 Conclusions

We have shown that we can construct a low-overhead AVB system that obviates the need for full blown AVB switches. This reduces the cost of AVB, and enables daisy chained systems to be constructed

Author Information:

Andrew Lucas, AVB Lead, XMOS Ltd

Peter Hedinger, Technical Director of Applications, XMOS Ltd

Henk Muller, Principal Technologist, XMOS Ltd