electronic design

print | close

Understanding Audio Video Bridging

Electronic Design
Peter Hedinger Andrew Lucas Henk Muller
Andrew Lucas, Peter Hedinger, and Henk Muller
Wed, 2014-01-29 09:59

Audio Video Bridging over Ethernet (AVB) is a set of IEEE standards for transporting audio and other real-time content over Ethernet. More than 20 silicon manufacturers, audio and infotainment companies, and networking vendors have adopted these standards.

AVB often is purported to only serve large-scale applications, such as music venues. However, it also is excellently suited to small-scale applications such as consumer audio, audio conferencing, and in-car entertainment. Daisy-chained AVB fits these applications because it avoids the need for switches without reducing the system's capacity.

Table Of Contents

AVB In A Nutshell

From a high-level perspective, AVB works by reserving a fraction of the available Ethernet bandwidth for AVB traffic. AVB packets are sent regularly in the allocated slots. As the bandwidth is reserved, there will be no collisions.

All of the nodes in the system share a virtual clock. AVB packets have a presentation time that defines when the media packet should be played out. (AVB packets can include all sorts of time-sensitive data. In this article, we concern ourselves only with audio).

So, a system may comprise a host node that is delivering data (the talker) and two nodes that comprise the left and right speakers (the listeners). As all three nodes share a single global clock, the left and the right speaker will produce sound synchronously.

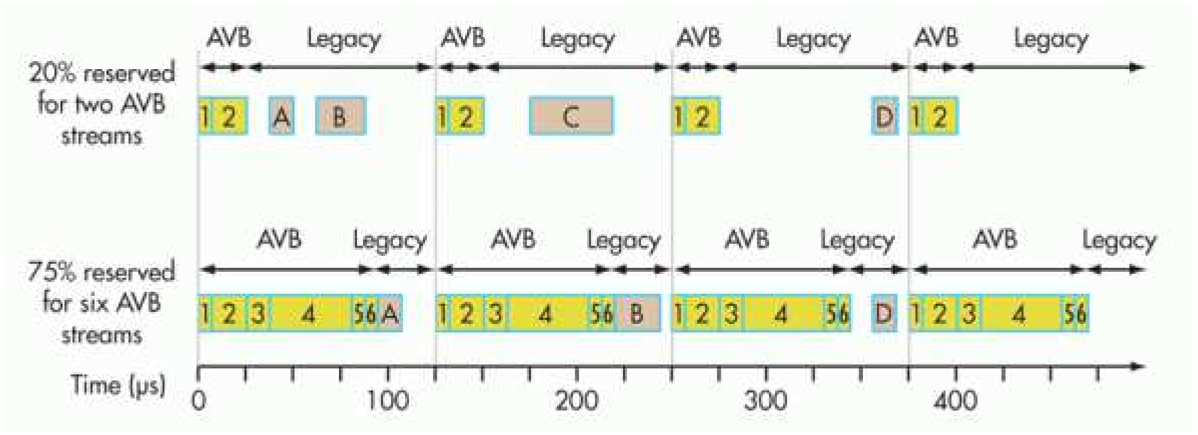
Related Articles

- Smarter Video

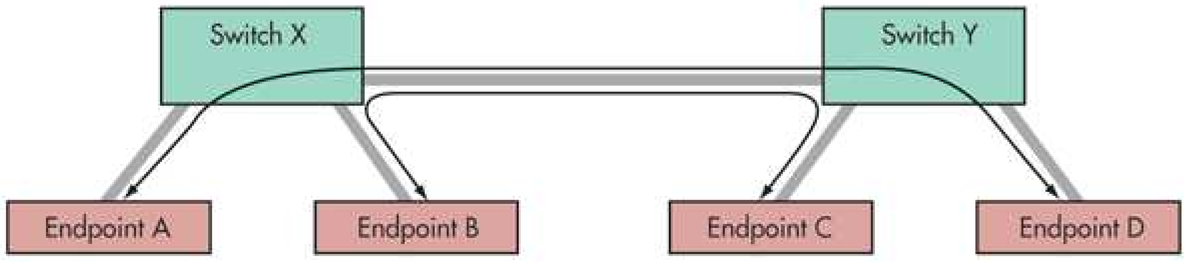
Surveillance Requires The Right Hardware And Software
- HEVC Improves Mobile VIdeo Performance
- Active DisplayPort Cables Extend Video-Interconnect Reach

The Stream Reservation Protocol

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. Every 125 µs, all real-time streams send their data. Other packets are transmitted when no more real-time data is available, holding up the other traffic *(Fig. 1)*.



To ensure 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 Mbits/s). Nodes B and C reserve another stream (say, 20 Mbits/s).



All switches in between those nodes will make sure that sufficient bandwidth is available: 65 Mbits/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 Mbits/s will be 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.

Using allocated bandwidth enables AVB to send data from endpoint to endpoint within a 2-ms window. AVB allows for a maximum of seven hops to meet this constraint, where each hop adds at most 125-µs 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). It 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 were a legacy switch, then it would not treat real-time traffic preferentially, potentially delaying the real-time traffic and causing jitter in the output.

The Precision Time Protocol

All audio traffic in AVB is synchronized to a global clock so audio producers and consumers can play and record sound synchronously. The Precision Time Protocol (PTP), already commonly used in networked computers such as laptops and servers to provide a synchronised clock, implements the AVB clock.

PTP assumes that all nodes have a reasonably good clock such as a crystal clock, preferably of a known accuracy like 25 ppm, equivalent to 2 seconds per day. Specified in IEEE standard 802.1AS, PTP is a second building block of AVB.

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. All other nodes estimate their skew relative to the master clock, enabling them 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 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.

Streams, Channels, Talkers, And Listeners

AVB is built around streams of data. If the data is audio, a stream comprises multiple channels such as stereo audio. Each AVB packet includes 125 µs worth of samples for all channels that are part of the stream. Streams are produced by talkers, which are the nodes that produce audio. A microphone or a laptop playing MP3 files would be a talker. 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 such as a DVD player, with six listeners, for 5.1 surroundsound

• Multiple talkers such as a group of microphones, with a set of speakers, for conferencing

• A few dozen microphones, a few dozen speakers, and a massive mixing desk for a music venue

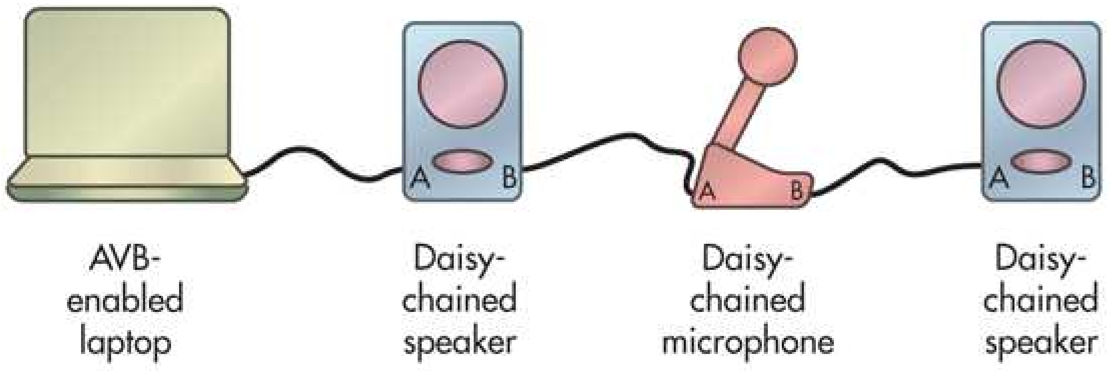
There are no rules on how small or large an AVB system should be, but there are practical limits. AVB streams have a sizeable overhead, limiting the number of streams that an Ethernet cable can carry. A 100-Mbit Ethernet cable can carry nine stereo AVB streams for a total of 18 channels or a single AVB 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. Hosts use it purely to configure the system.

Daisy Chaining

Compared to other mechanisms of digital audio distribution such as USB audio, AVB may appear expensive because of the need for AVB-aware switches. Daisy-chained AVB solves the cost issue. It uses an AVB endpoint with two Ethernet ports (we call them A and B) and a built-in "switch," which really isn't a fully fledged switch.

In a typical layout, a laptop is connected to node 1, which is connected to node 2, which is connected to node 3, where the network ends *(Fig. 3)*. Each node comprises two ports that are symmetrical and logic that connects the ports. If only one port is plugged in, the node acts as an ordinary AVB endpoint.



However, 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. Occasionally, the node will consume or produce a packet, such as when it's responding to any of the SRP, PTP, or configuration protocols.

The node, then, 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. 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 it 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 physical layer (PHY) and jack. There are limitations to this approach, though.

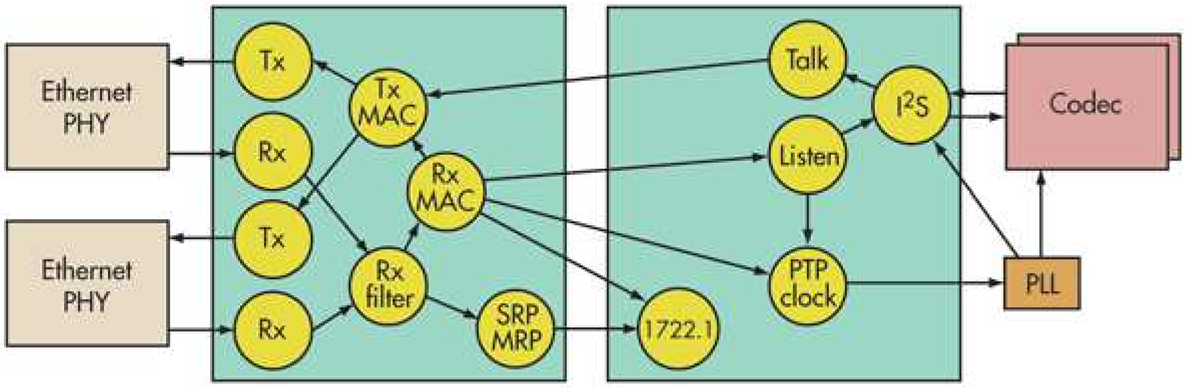
First, unlike a switch, a daisy-chained network requires traffic destined for the tail to travel through the whole daisy chain. In a switch with seven nodes, all seven nodes can in theory receive 100 Mbits/s of traffic. In a daisy-chained system, that would require the head of the node to transport 700 Mbits/s. But 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.

Second, 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.

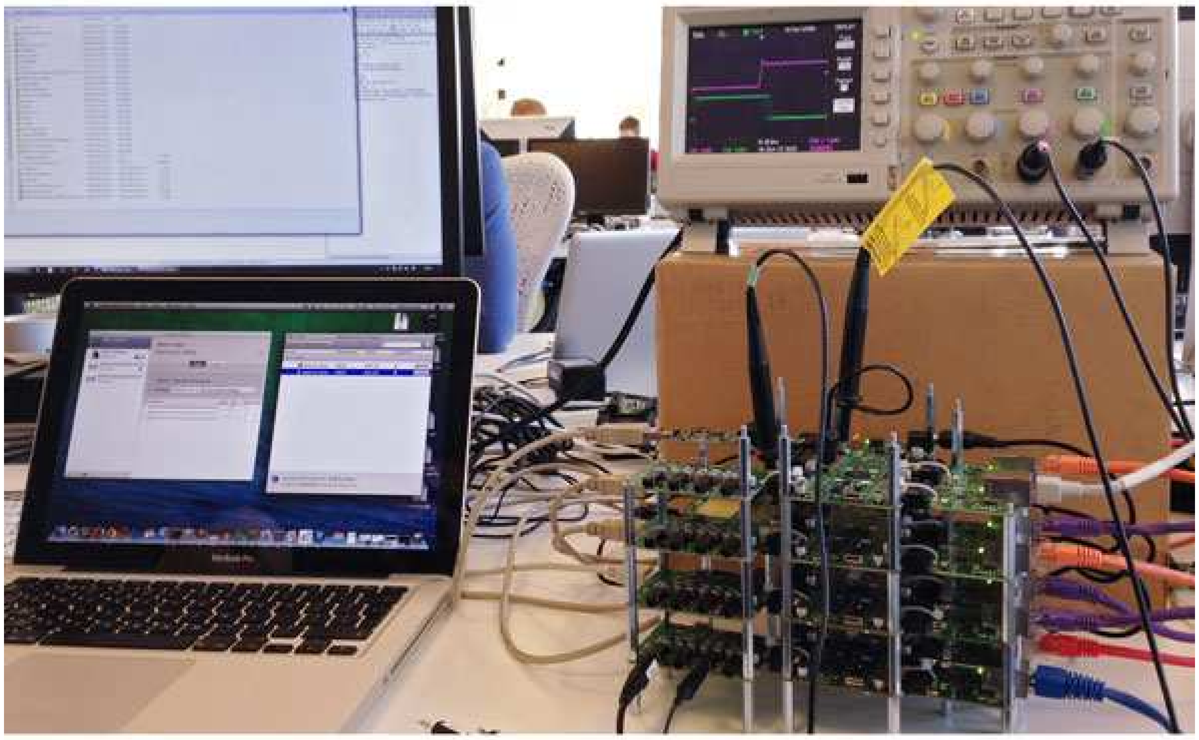
Example Daisy-Chain Implementation

For example, one daisy-chain AVB node can be based on an XMOS chip with 16 logical cores *(Fig. 4)*. The hardware used for the system comprises an xCORE multicore microcontroller with 16 logical cores, two Ethernet PHYs with magnetics and jacks, a low-jitter phase-locked loop (PLL) for word-clock generation, and 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: the talker/listener, PTP and media clock recovery, I^{2}S control, SRP/MRP, and 1722.1 discovery and control tasks. All 13 tasks fit in 128 kbytes 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, except for the media independent interface (MII) and buffering.

The system is constructed using an XMOS sliceKIT with two Ethernet slices and an audio slice. The stack of daisy-chained nodes is connected to a laptop that uses two of the nodes as “left” and “right” channels *(Fig. 5)*. Our audio slice comes with 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 each of the clocks shows 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.

Conclusions

You 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.

Andrew Lucas *leads AVB applications engineering at XMOS Ltd. He is responsible for the design and development of AVB reference designs and IP for the consumer, professional AV, and automotive markets. He also serves on behalf of XMOS on the AVnu Alliance, a consortium of companies working together to establish and certify the interoperability of AVB standards. He holds a first class (honors) master's degree in computer science and electronics from the University of Bristol.*

Peter Hedinger *is technical director of applications at XMOS Ltd. He is involved in the development and testing of AVB and USB audio products. Prior to that, he spent 13 years working on software tools, microprocessor architectures, and 10/40Gb Ethernet switches.*

Henk Muller *is the principal technologist at XMOS Ltd. In that role he has been involved in the design and implementation of hardware and software for real-time systems. Prior to that, he worked in academia for 20 years in computer architecture, compilers, and ubiquitous computing. He holds a doctorate from the University of Amsterdam.*

Source URL: http://electronicdesign.com/communications/understanding-audio-video-bridging