

Ethernet for Studio Audio Systems

Steve Church

Telos Systems, Cleveland OH USA

steve@telos-systems.com

Those of us involved with radio broadcast studios have seen tremendous evolution during the last decade. Reel-to-reels have been pushed aside in favor of PC editors. PC delivery systems have replaced CD players and cart machines, which had only a few years earlier replaced turntables. And just now, we are smack inside the center of the conversion to digital for mixing, routing, and processing, a trend certain to accelerate with IBOC poised to take off in the USA.

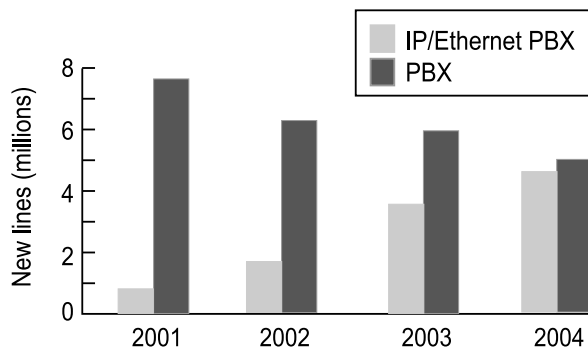
WHAT WE HAVE

But we are still using old-fashioned and limited schemes for connecting all of these new pieces together. With technology in flux, we find ourselves lashing-up a mishmash of:

- Analog, both professional and consumer, with XLR, RCA, DB-9, DB-15, ¼" phone, mini phone, and RJ-45 connectors
- Digital, AES-3 and MADI
- Digital, over proprietary fiber and copper
- Audio file transfer over data networks

Our industry is clearly ready for a new way to interconnect studio components – something that gets the job done simply and reliably as a replacement for the analog and digital methods we are using now, but also ready to take radio studio infrastructure into a future that will require more than audio. HD-Radio's text capability means that song title information, for example, might need to be linked to audio feeds.

The technology side of radio broadcasting is small beans compared to computing and telephony, so we usually borrow and adapt our technologies from those industries. Developments in those fields are significant to our future. So, what's been happening there lately? Clearly, Ethernet and Internet Protocol have taken the computer world by storm, with the vast majority of local networks now using this technology combo. But more interestingly, the telephone world looks to be going there also. Voice over IP is gaining on traditional circuit-switched phone systems, with VoIP gear now taking around 10% of new PBX shipments. Nearly all of the major PBX vendors offer VoIP. Router and Ethernet switch vendors from the computer world have been giving high-profile demonstrations of audio and video being run alongside data traffic and are heavily promoting "converged networks" and LAN-based telephone PBX systems.



Forecast for Ethernet/IP vs. traditional PBX lines
(Source: The Phillips Group InfoTech)

You probably have not been thinking of it this way, but Ethernet is probably already the most widely used digital audio transmission method in larger radio facilities today. Computer audio delivery is very often a client-server system with Ethernet connecting the server and the computer in the studio. You don't think of this as exactly audio transmission because it is a file-based transfer – probably done well in advance of playout. The transmission "latency" (to use the fancy network engineer's term for transmission delay) is seconds or minutes, and the audio is stored (that is, buffered) on the playout machine. But audio over Ethernet, it is.

... AND WHAT WE WANT

So what if we could just speed this up so that delay was in the sub-millisecond range? And find a way to ensure reliability? Then wouldn't we have a low-cost, universal way to connect audio and data for everything we have in our studio facilities? If we could, the advantages would be many:

- Ethernet would be low-cost. Because it leverages R&D and manufacturing scale from the high-volume computer world, cables, plugs, tools, testers, and PC network interface cards are standard and off-the-shelf. With its huge installed base, cost will no doubt continue to fall and capability to increase.
- A single Ethernet network could be used for audio, data, and telephone.
- An Ethernet-based system could scale from very small (two terminals connected to each other) to thousands of channels.
- A wide variety of wiring infrastructure components could assist installation.

- RJ-45 plugs would be fast and easy to install.
- An Ethernet switch inherently could provide audio routing at no cost additional to the basic infrastructure.
- We would be ready for a radio future that includes synchronized visual and text elements, such as for IBOC or the web.

So, it would be most excellent if it could be made to work. But can it?

The tech issues that need to be resolved to bring this concept to reality are: Can we get the delay low enough to fully support live applications? Can we have the solid reliability we need for uninterrupted audio? Can we have full pro-audio quality?

You may have heard about Ethernet’s unpredictability with regard to consistently delivering bandwidth, and maybe even experienced it yourself when an office network you were using slowed to a crawl. But that was your grandfather’s Ethernet. Today’s Ethernet is nothing like the original, which was invented over 25 years ago. The recent introduction of data rates to a gigabit, switching, and full-duplex technology has changed everything.

Ethernet is the “PC” of the data networking world. It started with limited capability and no means for achieving any kind of guaranteed service quality, so it received little respect among network gurus. In particular, it was not seen as suitable for applications involving real-time audio or video. But as with PCs, widespread adoption has led to a dramatic increase in performance. Speed has increased from 10 to 100 to 1000 Mbps. The original bussed coaxial cable has given way to star-configured copper and fiber. Ethernet switches allow each link to own all of the theoretical bandwidth and full-duplex is routine. Just as PCs have grown to be perfectly acceptable audio editing and delivery devices, so has Ethernet increased its performance to be able to support live audio transmission – as we will soon see.

ALTERNATIVES

What are the alternatives for studio digital audio connections?

AES-3 is becoming popular. But this is unquestionably not the future. With its lack of flexibility and one-way single-source-per-cable limitation, AES-3 falls way short of what we need for modern applications. Control and data is possible only for very simple, low rate functions. AES is now over 15 years old and reflects the limitations of its day.

Proprietary audio network techniques are growing in popularity for connecting routers to each other and to terminals. Systems using this approach are certainly more modern and capable than those using analog or AES-3, but the downside is that they are expensive and closed, with the obligation to buy all components from a single vendor. And connections to the “outside world” must still be via analog, AES, or MADI.

ATM is a network technology invented by telephone engineers and used within the telephone infrastructure. At one time, ATM was looking like it would be extended all the way to the desktop, becoming a serious rival to Ethernet for office LANs. Its main advantage is that it naturally provides full “quality of service” for audio and video. It does this by dedicating predictably-occurring cells upon request to create “virtual channels”. Its disadvantage is high cost and mind-numbing complexity.

Internet streaming looks pretty close to what we want, so can’t we just scale that up? Internet streams are usually compressed to very low bitrates, but there is no inherent reason to do that – and one could transmit full uncompressed CD quality were there enough bandwidth. A subtle but important problem remains: there is no coupling of the sampling clocks at the send and receive sides, causing problems with delay and clock-slip glitches. (more on this later) We also don’t want to be stuck with PCs as our only choice for audio input/output terminals.

USB and **Firewire** are good ways to get audio and video into and out of a PC, but there is no way to extend these to the multiple ports we need for a studio plant.

CABLES ‘N BITS: NETWORKS DEFINED

Networks are defined by their structure – bus, star, or some combination, and the organization of their bits – packets, continuous, or some combination. Traditionally, networks for (telephone) audio were distinct from data networks.

Telephone Digital Links: The First Audio Nets

Telephone engineers invented the first digital audio transmission system in the 1960s to save on copper. Two pairs carrying digitized connections in the “T-1” format could handle 24 channels, a savings of 22 pairs for each link.

2 bits	8 bits	8 bits	...	8 bits
Sync Flag	Slot 1	Slot 2	...	Slot 24

T-1 bitstream

The pattern above repeats at an 8kHz rate, the audio sample period. And each time, the 8-bit sampled amplitude for each active channel is placed on the wire in its own timeslot. There is a byte for each slice of audio, and every bit is in its place. T-1s are plugged to ports on Telco Central Office switches, and the switch knows what channel belongs to which telephone by a combination of the port and the offset within the bitstream. Newer digital telephone transport systems, such as ISDN, are similar.

Digital telephone switches connect the channels by sending the audio byte to a timeslot on a “backplane” bus and pointing the receiver to it. Note that there is no

information about the audio contained within the stream itself, either on the cable or the bus – there is no way of knowing the telephone number associated with a channel on a T-1 by examining the bits. Rather, the correspondence between the T-1 ports/timeslots and the backplane timeslot must be maintained entirely “out of band” within the switch’s software. This style of switching is called TDM (Time Domain Multiplex) because the channels are multiplexed on the backplane according to a timed offset. This is in contrast to “Space Division Multiplexing,” which devotes a separate cross-point to each connection, the way it was done in analog days gone by. (In practice, TDM switches have many parallel busses in order to handle the required volume of connections.)

Borrowing from the telecom world, modern broadcast audio routers are TDM devices, very similar to digital telephone switches. And the proprietary interconnects, whether copper or fiber, are similar in style to T-1s with regard to bit structure.

These cables and switches/routers taken together make a “circuit-switched” network, so-called because the result of an active connection is an emulated analog circuit. During the time of a connection, the two ends are as if wired-up to each other, owning all of the bandwidth, whether they need it at a particular moment or not.

Packet Networks

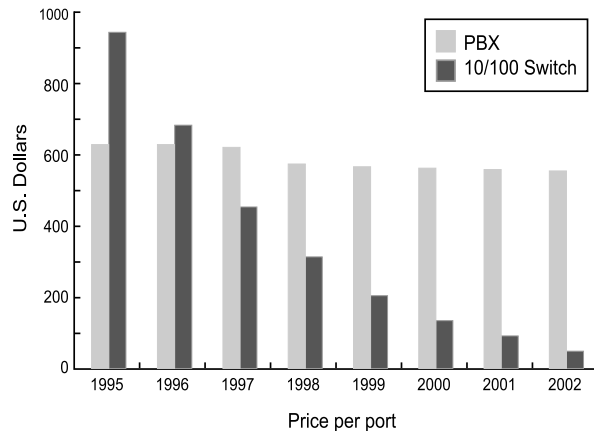
Computer networks are almost always packet-based, because data is naturally bursty. When you first open a web page, you cause a lot of data to flow. But while you are reading it, there is no data moving. Another reason packets became popular for data is that they let a number of data sources share the same wire using *statistical multiplexing*.

Our interest here is audio, so why do we want to think at all about data networks? Aren’t circuit networks perfectly fine for audio?

The answer lies primarily in the tremendous scale of manufacturing in the data network world and the flexibility such networks offer.

- Computer network components are much cheaper these days than their circuit-oriented counterparts owing to their ubiquity and high-volume.
- We often want to have both audio and data simultaneously on the same network.
- Computers are nowadays very often either the source or destination for audio signals and data networking capability is built-in.

We see the convergence of the two network styles most clearly in the VoIP telephone application that is rapidly gaining on old-style PBXs. The idea is that you need only one cable to connect both your office PC and telephone. And the switch that does the work for both is a cheap commodity Ethernet switch rather than an expensive proprietary PBX. The cost benefit is significant.



PBX vs. Ethernet switch per-port pricing trend

ETHERNET

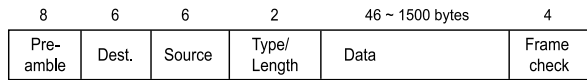
Enter Ethernet. Ethernet is a packet network, but by convention, Ethernet packets are called *frames*. (I will use “frames” when referring to Ethernet low-level functions, but will use “packets” when the general concept or application level is being described.)

The original Ethernet was based on a single shared coaxial cable – the *Ether* in Ethernet’s name. The very first versions used a 1/2” thick coaxial cable with physical taps into it – you actually had to cut a little piece out of the jacket and screw in a metal part that made contact with the ground and center conductors. Later, the coax cable was smaller and T-connectors were used at the back of connected computers, but the principle remained the same. When Ethernet eventually transitioned to telephone-style twisted-pair wires with a central hub, the coax snaking around the office disappeared – but the medium was still shared. All of the connected terminals received all the traffic and the receivers filtered out all frames except those addressed to them.

When a terminal is transmitting, it owns the full capacity of the cable. That means that there has to be some method to arbitrate access so that data from the various terminals don’t interfere with each other and that all have a chance to get on the wire and use their fair piece of the available bandwidth. This is done by the MAC – Media Access Controller – in each terminal. Bob Metcalf invented the method at Xerox PARC in 1973. His mechanism senses when a collision occurs – this is *collision detect*. Upon detecting a collision, both terminals choose a random back-off time and then retransmit their frames with a good probability for success. The system includes a listen-before-talk function to reduce collisions – a *carrier sense* function. Using these, all terminals could share access to the channel – a *multiple access* scheme. Put these all together and you can understand why Ethernet is called a Carrier Sense Multiple Access with Collision Detect (CSMA/CD) system.

In contrast to circuit-switched structures, Ethernet

frames carry source and destination addresses as part of the header. This means a switch can examine this to know where each came from and where it is intended to go.



Basic Ethernet frame

Another important difference is that the frame size ranges from 72 to 1526 bytes, depending on the amount of data to be carried. This means it can be flexibly adapted to the application, and various length frames may dynamically coexist.

Switched Ethernet

Switched Ethernet is a fundamentally different technology from the original, despite the name and the compatibility at the terminal level. With a dedicated full-duplex connection from each terminal and a central switch that routes traffic, switched Ethernet is no longer a shared medium system, and therefore does not need or use a Media Access Controller and the associated CSMA/CD scheme. Network interfaces automatically disable these functions when they are plugged into switches.

It is the recent arrival of this technology that makes pro audio over Ethernet possible.

ETHERNET IN STUDIO ACTION

Telos has been developing a studio audio transport system called *Livewire*. The technical ideas at the heart of this system are:

- Audio, control, and any needed non-audio data are conveyed via a common Ethernet.
- An Ethernet switch is used to isolate links and route audio.
- Links are full-duplex, audio streams are prioritized, and bandwidth per link is limited. Together, these techniques maintain fully reliable transmission.
- A clock signal distributed over the Ethernet allows precise synchronization and very low delay.
- Two audio modes are possible: 1) A very low-delay mode for links that involve live monitoring, and 2) An Internet Standard medium-delay mode for connection to PCs, satellite receivers, etc.
- Audio terminals advertise their streams to the network so that all connected receivers know what is available.

Livewire transports 50 uncompressed professional audio channels (25 stereo channels) in each direction, with additional capacity for non-audio data transmission. 1000BASE-T or gigabit fiber can support over 250 stereo channels.

The native Livewire audio format is 48kHz sampling-rate and 20-bit resolution. Other rates may be supported with conversion in terminals.

Keeping Delay Down

Broadcast studios have the requirement that DJs be able to listen to themselves in headphones. Maximum tolerable delay before the perception of annoying “comb-filtering” becomes a problem is around 10-15ms, and greater delays cause echo to be heard. So our microphone-to-headphones delay budget is around 10ms. But we can’t burn this all on one link because there may well be multiple links and maybe devices like processors in the path that also have delay. So the contribution of each link must be kept very small so that the cumulative result is below the audible threshold. Our goal in Livewire was to keep delay below 1ms per link, and we have accomplished this. There are two keys to live audio success:

- Use a sufficiently small frame length.
- Use a clock and PLL system to synchronize the audio bit-level transmit and receive clocks in terminals.

Frame length is an important trade-off in packet networks that are used for live audio. Smaller packets mean shorter buffers at the transmitter and receiver, leading to lower delay. But longer packets are more efficient because the header overhead is shared by more data bits. Fortunately, here is where Ethernet’s flexibility helps us: we can choose the length we want dynamically, so we don’t need to settle for one compromise value. We take advantage of this in Livewire by supporting two stream types: a very low-delay “Livestream” mode, and an “IP Standard” mode.

The **Livestream** mode is intended for live audio signals such as microphones and monitors that need the lowest delay. Livestreams have 16 audio samples per frame, or 250µs (a quarter millisecond) at our 48kHz sampling rate. We need a one-packet transmit buffer and a two-packet receive buffer to cover the worst case, so the end-to-end link delay is less than 1ms. To put this into context, the usual professional-grade analog-to-digital converter has 500µS delay, and one meter of audio travel in air is about 1ms delay.

The **IP Standard** mode is intended for feeds from PC-based delivery systems and remote networks such as from satellites. We pack 240 samples into each IP Standard frame, using nearly all the 1500 bytes possible in an Ethernet frame, and making for around 5ms link delay, including buffers. IP Standard streams will usually be generated and consumed by PC delivery systems and editors, so the main motive for this format is to reduce the interrupt rate to one that a PC can handle. These have more efficiency with regard to bit usage because the overhead from the header is spread over more data. This higher efficiency lets us use the full IP protocol for live media according to the internet RTP (Real Time Protocol)

format defined in the IETF (Internet Engineering Task Force) standards document RFC1889. We want to keep Livestreams lean and mean, so we stick there to raw Ethernet frames, but here we have no problem with carrying the extra IP load.

More buffers mean more delay, so we want to have as few as possible. With careful design, we can keep the total to three: one in the transmitter and two in the receiver. This requires a high-accuracy clock to be distributed to all terminals. If each terminal were to have an independent clock, the slight differences between the two would mean that more buffering to cover the wander would be needed at the receiver – and even so, eventually the buffer would over or underflow and the audio would be interrupted. (Sample Rate Converters could also resolve this problem, but they would impose their own significant delay, as well as adding unnecessary cost, complexity, noise, and other problems. And they don't help with delay.)

The terminal with the lowest assigned IP number serves as the master clock source and all the others are locked to it. If it is unplugged or fails, another will automatically and seamlessly take its place. This “master” terminal sends a clock packet to the network at regular intervals.

Our clock packet is *not* used to create time slots or to order the outputs of the transmitting terminals. We don't need this because we are not using the cable as a shared medium and have no need for timeslots. Our links are all dedicated and full-duplex. And the clock packet is not transmitted at the beginning of a sequence of audio packets. Rather, it is transmitted at a much lower rate and a PLL (Phase Locked Loop) circuit is used to increase the rate to provide a synchronized clock in terminals. Because switched packet networks can introduce variable delay, we use a sophisticated method for transmitting and recovering the clock.

Capacity Counting & Priority Tagging

Normal Ethernet, even when switched, is a “best efforts” system. A computer tries to send data as fast as it can and all of the others on the network are doing the same. In the face of this, we must take some steps to ensure that we will always have the bandwidth we need for each active audio stream. We do this by:

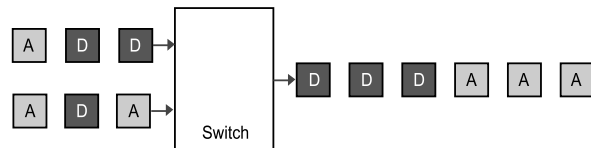
- Never allowing links to be overfilled. Terminals are in control of the streams they transmit and also the ones they request the switch to send them for reception. They have a function that calculates the available link capacity and decides if there is enough space remaining before connecting any new audio channel. Of course, the system is engineered so that this limit is never noticed by a user.
- Tagging audio frames with a higher priority value than data so that network interfaces and switches can distinguish them and put them ahead in their queues. We do this on a per-frame basis, not by assigning

particular Ethernet switch ports permanently to high priority. This lets each link pass both high-priority audio and lower-priority data.

These are the essential procedures if we desire to achieve full audio reliability from Ethernet. Because we have a switch port dedicated to each link, we know definitively what bandwidth is available. We can calculate *to the bit* what the utilization of the link will be for our audio streams. And we have full-duplex; each direction is isolated and independent from the other.

If we weren't interested to share audio links with non-audio data, capacity counting and limiting alone would be sufficient to have 100% reliability. But we *do* want to have non-audio data – we need to take care of the clock synchronization and control packets at the least. At the most, we could well want to have a PC with all the usual file-transfers, emails, web browsing, etc. This is where prioritizing the audio packets comes in. This is how a packet-oriented data network like Ethernet is able to offer us the QoS (Quality of Service) we need for audio, even when data is contending for the available bandwidth.

Computers are not careful with their data rate – like Italian drivers, they want to go as fast as they can. (Those who have visited that country will understand.) So we make a rule: Our audio is like an ambulance, causing the road hogging speedsters to pull over and get out of the way.



Audio frames (A) are tagged with high priority, so the switch causes data frames (D) to wait.

Eventually, when there is space, the PC's network driver and the switch will allow the low-priority packets through. If there is persistently not enough capacity over a long period, the driver and/or switch will drop packets – but this is not at all a problem. Indeed, it is the usual way the internet works.

Part of the network driver in every PC is a function that detects the available bandwidth and automatically adjusts the data flow to match. This is the TCP (Transmission Control Protocol) part of the TCP/IP combo. When a connection is first made, and continuously during transmission, TCP probes the link to determine the appropriate speed. When packets are dropped, it takes this as a signal to slow down. Any such dropped packets are recovered with a request for retransmission and subsequent response with the lost packet. This is how your home PC and 56k modem manage to work. The PC

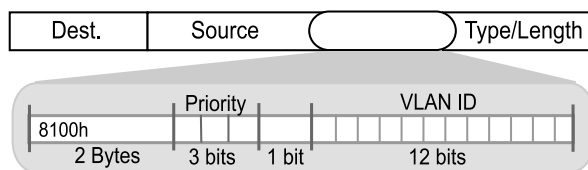
wants to go fast, and can certainly run faster than 56k, but TCP automatically makes the required adjustment.

So, in the end, we have exactly what we want: The audio gets the bandwidth it needs without fail, and the other data naturally and automatically adapt to perfectly fill the remaining bandwidth.

We can set an upper limit on bandwidth devoted to audio in order to reserve some capacity for other data. In a 100BASE-T, if we set the limit to 85%, there will be plenty enough capacity for 25 audio streams, while still having 15Mbit free for everything else.

Priority is new to Ethernet, having come along with switching. IEEE is the body responsible for Ethernet standards, and they added this function with the 802.1Q and 802.1p extensions to the basic Ethernet definition in 1998, mainly to support VoIP telephones and other real time media. This really is a very simple and clever way to mix high QoS services with normal data without going to all the complexity and rigidity of circuit-switching. You get a very open, flexible, and boundaryless transport medium with an uncomplicated addition to the basic Ethernet.

Implementation of priority is through an additional 4 bytes of data inserted into a frame's header. Within these 4 bytes is a field for the 3-bit priority flag, providing eight possible values. The new fields are inserted into a frame's header immediately following the source and destination address fields and before the 802.3 "length" (or the Ethernet II "ethertype") field. The first 2 bytes are where the original "type" field was and are fixed to a value specifying that tag control info follows.



Modern 802.1Q/p Ethernet frame with priority and VLAN tagging

A high-end switch will support all eight priority queues, but some simpler ones have only two, with the top four levels being mapped to the high-priority buffer and the bottom four to the low-priority buffer.

Some switches allow "port-based" priority, where a configuration setup can be used to assign a port full-time to a given level. But we don't want this. Rather, we want to take advantage of the 802.1Q/p tagging on a per-frame basis so that a single link can be shared for both audio and data. All Livewire terminals tag audio and clock frames with priority 6, and control frames with priority 3. (The top level, 7, is reserved for "network control" messages.)

How Ethernet Switches Morph into Audio Routers

The idea behind switching is pretty simple, at least for unicast (point-to-point) transmissions. The switch builds up a table of what addresses are attached to what ports. It does this merely by examining sent frames. When a terminal sends a frame, there will be the source address in the header. If the association is not already recorded in the Source Address Table, it is added. If a connection is unplugged or there is no data for a long time (usually days), the entry is removed. When frames come in, the switch looks into the table, discovers what port owns the destination and forwards the data only to that port. In the rare case that no entry exists for an address, frames destined for that address are "flooded" to all ports to be sure the intended recipient will receive them.

Multicast (one-to-many) transmissions are used for Livewire so that an audio source can be received at any number of locations. A multicast Ethernet frame has a special destination address, one that is not associated with a particular port and terminal. This is a "virtual" address that is just stopped inside the switch if there are no interested receivers. When a receiver wants to tune-in, it sends a message to the switch telling it to turn on the stream. This message can use the IEEE standard GMRP (Group Multicast Reservation Protocol) or "IGMP (Internet Group Management Protocol) snooping". In either case the result is the same: an entry is made in a table that routes multicast frames only to subscribed destinations.

The switch knows what frames are multicasts because the destination address belongs to a pool set-aside and defined within the Ethernet standard for this purpose. Interestingly, Ethernet has set aside *half* of all destination addresses for multicast – 140,737,488,355,328 addresses, which should be enough for even the very largest broadcast facility! (The distinction is made in the first transmitted bit of the 48-bit address: a 1 in this position signifies a multicast.) The designers clearly had big plans for multicast that have not yet been realized.

There is also a 'broadcast' address in Ethernet. Data sent to this address is received by all stations. This is usually used for "where are you?" messages such as for Ethernet-to-IP address resolution and file sharing systems.

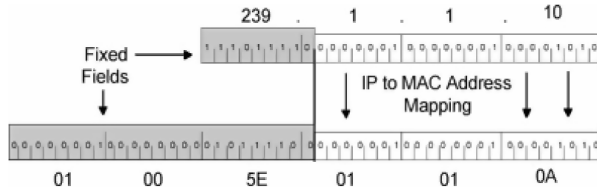
Each audio terminal has a normal Ethernet address and IP number assigned to it to be used for control and configuration, similar to the usual PC network setup.

There will also be a set of contiguous multicast addresses assigned for the audio, one for each send audio source. You assign these numbers during configuration using either a terminal's on board user interface or a connected PC with web browser. These values are stored along with the text names for the terminal and each audio stream in nonvolatile memory in the terminal.

Livestreams are Ethernet multicast as described above. IP standard streams are multicast at both Ethernet and IP layers using the set-aside multicast addresses at each

layer. IP addresses will be mapped into an Ethernet MAC layer multicast, according to a de facto standard process for this procedure. This process is as follows:

1. Identify the low order 23 bits of the IP Class D address.
2. Map those 23 bits into the low order 23 bits of an Ethernet address with the fixed high order 25 bits of the IEEE multicast addressing space prefixed by 01-00-5E.



IP to Ethernet mapping (example for IP Class D address 239.1.1.10)

For example, the mapping of IP address 239.1.1.10 to Ethernet is done by placing the low order 23 bits of the Class D address into the low order 23 bits of the reserved MAC layer multicast address 01-00-5E-xx-xx-xx. Since only 23 bits are mapped, the 24th significant bit is fixed at 0. The final MAC address that is utilized by the multicast group 239.1.1.10 is 01-00-5E-01-01-0A. (Ethernet addresses are written in “dashed-hex” form, while IP addresses are written in “dotted-decimal” form. Both are ways to represent information contained in bytes.)

For our IP Standard streams, we use the IP address range from 239.128.0.0 through 239.255.255.255. This choice is based on the assigned numbers from the IANA (Internet Assigned Numbers Authority) allocation of this range for use within organizational and site specific scopes. These addresses are to be used for multicast applications that are not used across the global internet. Since our application will be used within a single organization and is not intended to be placed on the public internet without translation, this range is appropriate.

We will assign IP Class D addresses sequentially so that no two complete addresses within the range are the same. Over 8 million unique Class D multicast addresses will be available with each address mapping to a globally unique MAC layer multicast address.

Advertising

Livewire audio terminals advertise their streams to the network so that receivers know what is available. When a terminal is first connected and each 10 seconds thereafter, a special message is multicast describing its streams. This includes addresses, characteristics, and text names. Receivers build local tables with this information that can be displayed to users for selection. If an advertisement is missed three times, receivers remove the associated entry from their tables. This happens when a terminal is disconnected from the network, powered-down, etc. There

is also an explicit *off* message that immediately signals that the audio is no longer present.

We send this advertising “out of band” rather than burdening the actual streams with any descriptive information because we want to keep the audio frames as clean as possible and the efficiency maximized.

For these messages, we use a Telos-developed object-oriented protocol carried over IP in a special format called *RUDP* (Reliable User Datagram Protocol). This allows the construction of messages ranging from simple to complex in an open, extendable way.

VLANs

Livewire audio, clock, and control may be assigned to a VLAN (Virtual LAN) not used by normal data traffic. A VLAN is a logical grouping of nodes, consisting of clients and servers that reside in a common “broadcast domain”. Remember that Ethernet has a special address for broadcasts that go to all terminals. In very large LANs this traffic can become quite big, and VLANs are a way to contain that traffic. Anything sent on a particular VLAN will not be seen on others. When there is a common network for audio and general data, it could make sense to isolate Livewire to its own VLAN to keep broadcasts away from audio terminal links.

VLANs also provide a measure of security because it would be impossible for a hacker to cross from a VLAN connected to the internet to another dedicated to audio.

As with priority, VLANs may be established on a per-port or per-frame basis. We will generally want to do this on a per-frame basis.

VLANs have no effect on priority. This must still be handled with the separate priority mechanisms.

Network “Layers”

When perusing data sheets from switch vendors, you may encounter the terms “layer 2 switch” or “layer 3 switch”. These are referring to the OSI (Open Systems Interconnect) network layers. (nothing at all to do with MPEG audio layers)

- Layer 1 is the physical, hardware layer
- Layer 2 is the Data Link Layer, corresponding to Ethernet
- Layer 3 is the Network layer, corresponding to Internet Protocol
- Layer 4 is the Transport layer, corresponding to TCP
- Layers 5-7 are the Session, Presentation, and Application layers

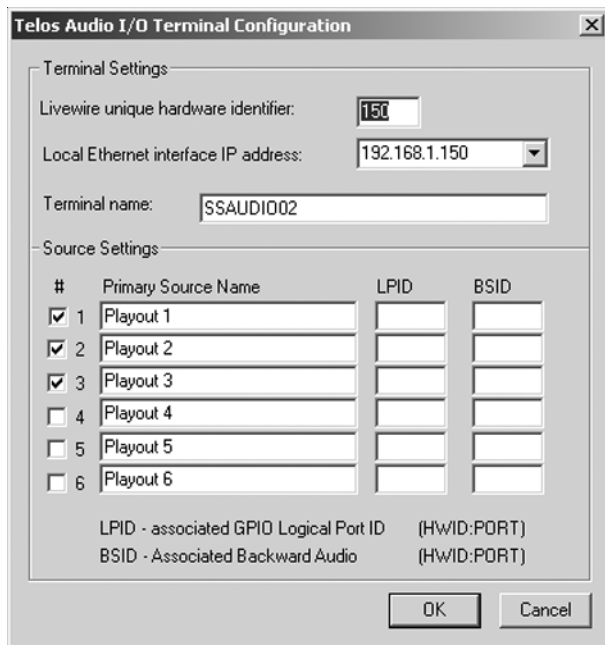
So, a “layer 2 switch” is a basic Ethernet switch that operates at the Ethernet frame and address level. It has no knowledge of any upper layers, such as TCP/IP. A “layer 3 switch” is able to look deeper into the frames to the IP level, something that routers have traditionally done. Livewire needs only a basic layer 2 switch. However, a

layer 3 switch could be useful in some installations where the LAN serves multiple functions. It could be used to bridge VLANs, for instance.

Audio from PCs

A driver software component is used to get audio to and from Windows PCs to the Livewire network. It makes the network look like a sound card, so can adapt any audio software such as delivery systems and editors to the Ethernet.

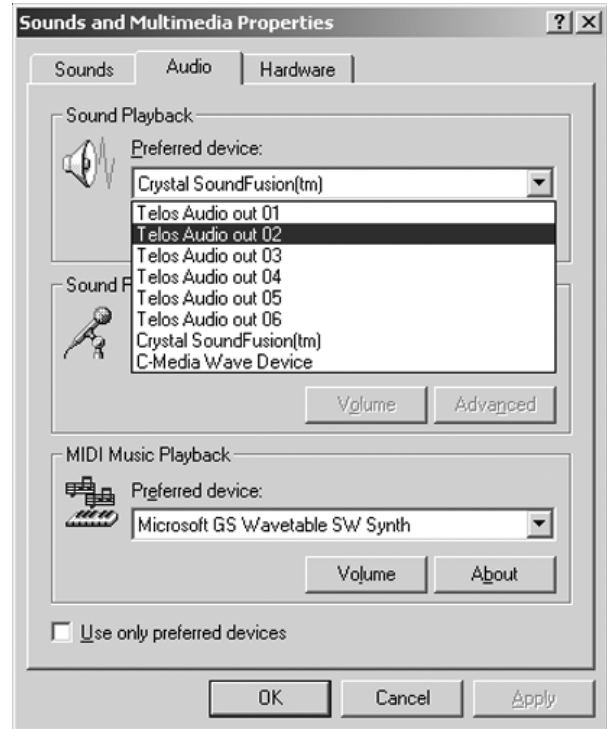
Because the packet rate would be too high, general-purpose PCs are not able to handle the very low delay Livestreams. But this is no problem because we have the IP Standard mode ready-made for the task. Longer packets with more audio samples in each mean that the packet rate is reduced. The increased delay caused by the bigger packets is not a problem because PCs are playing out files, not transmitting live microphone signals.



The Windows PC driver interface lets users assign addresses to audio sources

Multiple channels are supported so that, for example, delivery systems are able to send an independent output from each 'player' to a channel on the network and control surfaces can have a fader for each.

The driver also provides a simple API (Applications Programming Interface) that delivery software developers can use to access the stop-start functions from control surfaces that have traditionally been done via GPIO connections.



After the driver is installed and configured, a Livewire Ethernet channel looks like any audio device to PC audio applications

AN ETHERNET-BASED RADIO BROADCAST FACILITY

A horse that can count to ten is a remarkable horse – but not a remarkable mathematician. So, which do we have here? We've seen that Ethernet can be made to work as a satisfactory audio transport medium, but should it be pressed into this unusual service? Is this really practical and ready for the real world?

The main alternative is analog over copper. If you are reading this, you probably know everything there is to know about this technology. You've soldered thousands of XLRs and maybe more than a few RCAs. You know to reach for a resistor or a gain control when you hear distortion, the snips when you hear hum, and the cans when you hear nothing. You are learning how to use AES-3, and it seems to work most of the time. You even know a bit about Ethernet because you're using it to plug the studio PCs into servers, etc. and anyway the GM has you running the station's data network and fixing the errant PCs in the accounting office.

But the notion presented here – putting your live audio on Ethernet – just seems, well, weird. From the perspective of an experienced XLR installer, sure. But imagine if you were coming fresh to radio from the computer world; wouldn't soldering your first XLR convince you that there

must be a better way? Wouldn't RJ-45s suggest themselves immediately? Indeed, are there not broadcast vendors selling devices to wire-up analog with RJs already? Well, then, what keeps us from taking the natural next steps: make the audio digital, make it bidirectional, allow a bunch of channels on one plug and cable, and combine with all the necessary control and other functions? And while we are at it, why not label each audio channel with a numeric and text ID? And let's do all of this really cheap – and get all the routing we need for nearly free. Doesn't this make a lot of sense? And doesn't it start your imagination going about how you can benefit?

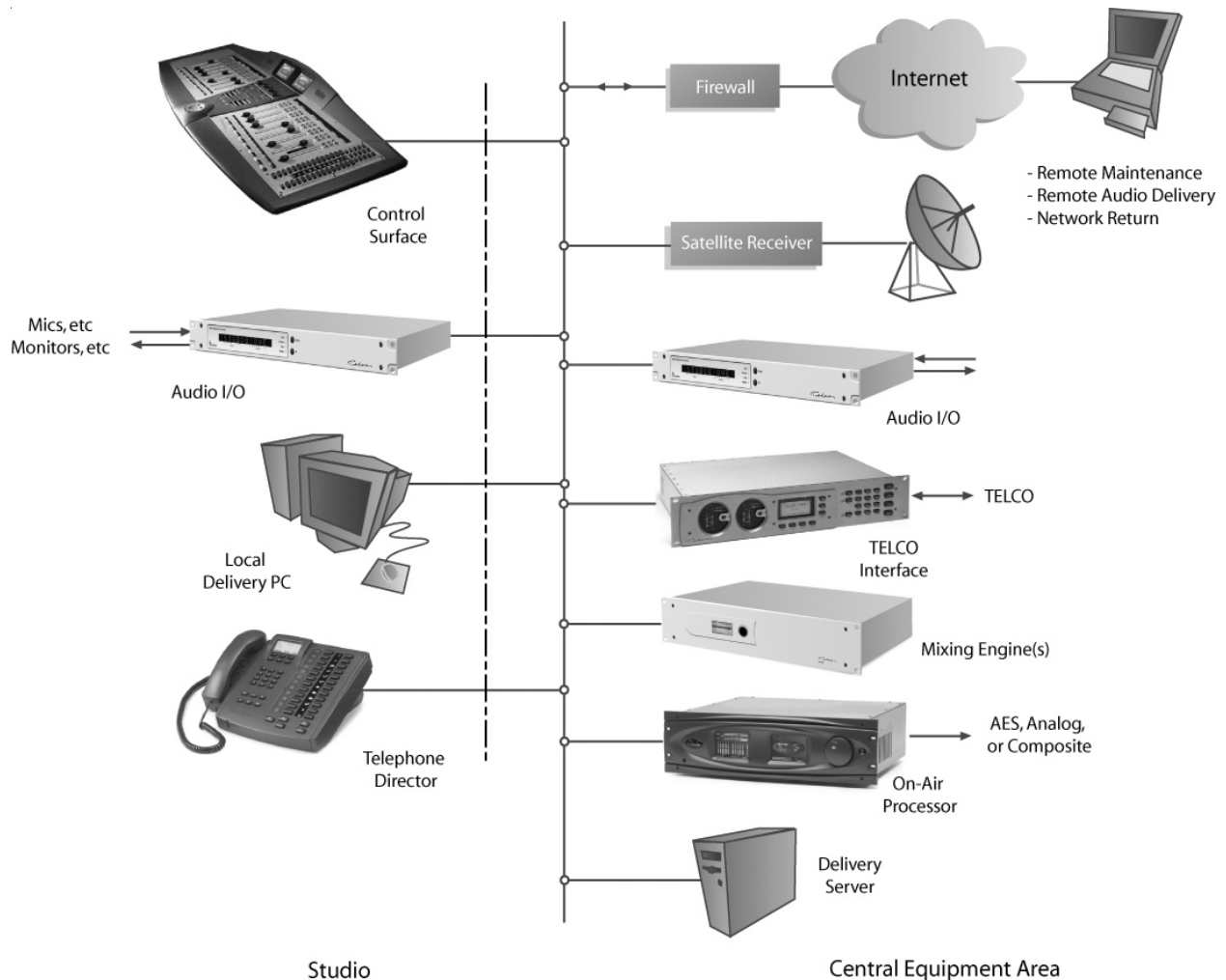
Cat 5 & RJs: Getting it Together

There is much detail described here. *Please don't let that lead you to think that this network stuff is hard to use.* Just as you don't have to know how to write C code to send email on your PC, you don't need to understand the

workings of Ethernet frames and PLLs to connect things. Indeed, making a piece of gear play is actually pretty simple:

- Connect it.
- If a source, assign a numeric ID and a text name.

There will be an Ethernet switch in a central area, along with some terminals to attach legacy audio, processing engines, Telco interfaces, etc. The switch connects by Cat 5 to everything studio-side, which includes additional audio terminals and control surfaces. The delivery PC has no sound card, but instead connects for both audio and the server to a single Ethernet. CD players, phone interfaces, codecs, etc. connect via audio terminals. (But eventually, much equipment comes with a direct Livewire port.) Newsrooms have mini-surfaces that talk to the associated studio's engine. There is a selector panel in the production studio that looks a lot like a traditional audio router controller and provides an equivalent function.



A broadcast facility taking full advantage of Ethernet for live audio, remote audio, associated control, data, telephony, and remote maintenance

Audio Terminals

Ever more audio in broadcast facilities is originating from or being sent to PCs and these plug directly to the net.

But, no doubt, we will have “legacy” audio for some time to come. So we need audio terminal devices that convert analog, AES-3, or other formats to and from the Ethernet. To reduce the cost per channel, these will usually be designed for six or more channels, input and output.

Eventually, with the cost of electronics constantly falling, it may be a reasonable idea to have single-channel “stick-ons” that could be dedicated to a particular audio source. The new IEEE P802.3af standard describes a way to power Ethernet links (including a scheme to switch on the power only when a terminal needs it). This is certain to catch-on for telephone sets as VoIP grows, so commodity components will be available for us to use to power small interfaces without having to plug them into AC locally.

Terminals can be placed near the audio and may be distributed throughout a facility according to convenience. A unit placed within a studio can collect audio from microphones and deliver audio to monitors, while another in the central equipment area can enter network feeds, codecs, Telco remotes, etc. into the system. Because of the inherent audio routing function provided by the Ethernet switch, any audio source from whatever location may be received everywhere.

A terminal version that looks and works like a traditional broadcast audio router control panel is used in places like production studios, newsrooms, and monitoring areas. This has the comfortable and familiar LCD and knob select capability, along with some assigned channel buttons.

Some terminals and a switch would make a low-cost functional equivalent to traditional TDM routers.

Control Surfaces

While our focus has been so far on audio, an important benefit of Ethernet is that it lets us easily combine data. The currently-popular (and sensible) configuration for a broadcast mixing console is to separate the user interface part from the engine that does the actual switching, fading, mixing, etc. Ethernet stands ready to support this connection. When surfaces need audio locally, such as for cue listening, this can be received from the same wire. Only a single connection is needed.

Audio Processing Engines

With a computer network at the heart of the studio, we can take the next obvious step: use a PC as an audio processing engine, the back-end for the Surfaces described above. This would be plugged into the network and associated to the desired Surfaces with a software configuration process. As with the motivation to use Ethernet, PCs offer a lot of power at low cost due to their being manufactured at very high-volume. A single PC has plenty of DSP power to do everything a typical radio studio needs if the software is carefully designed with an eye to efficiency. The

designer must the approach the project as if the PC hardware platform was an embedded DSP device. There is no room for cycle-hogging operating systems with pretty user interfaces or sloppy code.

As with the network links, an important goal is to minimize delay, which means that both the network interface and DSP software have to be carefully coded for efficiency.

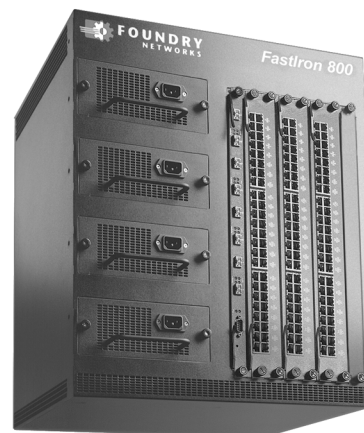
Is this possible? In a word: Yes. Telos has developed such an engine based on an off-the-shelf Pentium 4 motherboard and a reduced and real-time modified version of the Linux OS. It is able to support a full-featured 16-channel broadcast on-air control surface, including per-channel EQ, mix-minus sends, talkback, etc. with less than 1ms throughput delay.

Switches

Ethernet switches are available from dozens of vendors with varying capabilities and price-points. While some are very simple and cost less than \$200, others offer large numbers of ports, performance monitoring, redundancy, and many other sophisticated features. These range in price from a few hundreds of dollars to many thousands. You may start with a simple, low cost system and scale it up as needs progress. All of the peripheral devices would remain compatible with newer and/or more sophisticated switches.



A “Stackable” Ethernet switch.



Modular switches allow expansion within a single chassis and can serve hundreds of ports.

In a large facility with many studios it may be desired to use a number of smaller switches rather than one big one in order to have redundancy and to potentially save on cable runs. You could have a switch per studio. Ethernet switches have the inherent ability to be cascaded with the various multicast and other control signals being appropriately propagated. These switches need not be co-located, but rather can be Ethernet linked and placed wherever convenient.

Telephones

With the rise of VoIP and VoEthernet PBXs, you can easily integrate your station's telephones into the Ethernet backbone. The sets and central equipment could just be plugged into additional ports on the switch.

Security

Those very concerned with protecting the studio system will keep the local audio network 100% isolated from the internet, though they may well decide to connect it a private network linking co-owned or otherwise affiliated stations. There will be advantages to having the stations general data network and the audio network linked and there is no reason the two cannot be served by the same switch. Those very cautious might prefer to keep the nets independent, but link them with an IP router. As mentioned before, separate VLANs on a common switch would accomplish very nearly the same result. Of course you already know that whenever a connection to the internet is desired, a firewall will be required.

Radio Networks

Referring here to those networks that provide programming, not the infrastructure sort we've been discussing so far. Satellite systems with IP capability and receivers with Ethernet connections clear a trail that leads way beyond the live audio and mailed discs network model we've had since the 1930s.

The idea of integrating live audio, file-based audio, and data on a common packet-based "pipe" is useful beyond local networks. The structure can be extended by satellite or other means to provide radio networks and affiliate stations a distribution system with much more programming flexibility than is now possible. It could well open the door to a new era of programming that takes advantage of the possibility to smoothly blend national and local elements.

For example, audio packages may be sent for storage on the local server along with text descriptions of the content, suggested promotion and lead-in lines, etc. These may then be played at will from that station delivery system as if it were any other locally-produced content.

Making Maintenance Life Easier

True, you can't hang a pair of cans on an Ethernet link to see if you have audio. But, with computer networks

everywhere, there are a lot of tools from that world that can be used to track down problems. There are cable and plug testers, packet sniffers, and more.



Ethernet cable testers

Most Ethernet switches have built-in diagnostics that can be accessed by a browser-equipped PC. "Port mirroring" is a useful function that lets you effectively parallel a test port to another you want to observe.

Audio terminals and processing engines will have diagnostic features as well, some of which will be remotely accessible and some of which will be on local front panels.

Livewire terminals are designed so that they can be plugged directly one to another (without a switch) so that basic "offline" audio flow testing may be quickly done.

Audio terminals in the "router panel" style may be used to quickly check audio feeds. One of these permanently installed in the central equipment area and another "rover" should be adequate to debug most problems. No doubt, as networked audio takes hold, there will appear specialized testing gear in various formats.

IMPOSSIBLE?

Nearly two decades ago, I wrote in the introduction to the Telos 10 manual that DSP applied to broadcast telephony would slam-dunk solve a problem that had been around from the beginning of phones and broadcast studios (hybrid leakage). I predicted that the reaction would be forthcoming in the following order, as with almost all innovation:

1. It would be attacked as "ridiculous" and "impossible" – no chance it will work as the inventor claims.
2. Begrudging acceptance that the technology works, but with the arguments shifting to, "There is no need for change, the new approach is too risky, etc."
3. Imitation.

This is almost certain to happen with the suggestion that data network technology be applied to studio audio. But, whether or not the particular implementation described

here makes its way successfully into the marketplace, there will surely be a similar audio/data network in our future.

Today, most radio facilities have at least four networks already in place: An Ethernet for the computers, a proprietary PBX for the office phones, dedicated on-air telephone system wiring, and traditional audio wiring. This last is the least modern and most difficult to install and maintain, with its thick multi-conductor wires, punch blocks, soldered-on plugs from yesteryear, and ad-hoc mixture of digital and analog in both pro and consumer forms. And now there is a fifth creeping in: AES-3. And a sixth: proprietary digital. Ethernet has the potential to relieve this complication and make broadcast engineering life a little easier. It seems somehow fitting that that a network technology with the name *Ethernet* finally gets applied to radio broadcasting.

Acknowledgements

Many of the deep technical issues involved with timing, synchronization and latency were carefully considered and ultimately resolved by Greg Shay at Telos.

The PC driver and much other Livewire software was written by Maciej Szlapka at Telos.

Much contribution to the ideas and the implementation of Livewire – the mixing engine, in particular – has come from Maris Alberts, Gints Linis, Artis, Normunds, and all the team at the LU Department of Mathematics and Information Science in Riga.

Michael Dosch, in his position as Director of R&D at Telos, has done much to keep the Livewire work on track and the focus on users.

References

IEEE Standard 802.1Q. 1998. Virtual Bridged Local Area Networks.

IEEE Standard 802.3x. 1997. Specification for 802.3x Full Duplex Operation.

IEEE Standard 802.1p. 1998. Supplement to Media Access Control (MAC) bridges: Traffic Class Expediting and Dynamic Multicast Filtering. Incorporated in new edition of IEEE Std. 802.1D-1998.

IEEE 802.3af. Standard Supplement to CSMA/CD access method and physical layer specifications - Data Terminal Equipment (DTE) Power via Media Dependent Interface (MDI)

IETF (Internet Engineering Task Force) RFC 1889. Real Time Protocol.

Breyer, Robert and Riley, Sean. 1999. *Switched, Fast, and Gigabit Ethernet*. San Francisco: New Riders.

Davidson, Jonathan and Peters, James. 2000. *Voice over IP Fundamentals*. Indianapolis: Cisco Press

Metcalfe, Bob. 1993. Computer/network interface design: Lessons from Arpanet and Ethernet. *IEEE Journal on Selected Areas in Communications* (February) vol. 11, no. 2:173-179.

Hersent, Olivier and Gurle, David. 2000. *IP Telephony: Packet based multimedia communications systems*. London: Addison Wesley

Spurgeon, Charles E. 2000. *Ethernet: The Definitive Guide*. Sebastopol, CA: O'Reilly & Assoc.