INTEGRATED CONTROL OF DISTRIBUTED SYSTEMS

Rick Kreifeldt Harman Professional Group, Northridge, California, USA
Mark Bailey JBL Professional, Northridge, California, USA

1 INTRODUCTION

There are many problems associated with the development of a control protocol of a modern distributed audio system. A protocol has be lightweight for simple edge devices such as a microphone so as not to add too much cost, yet still have the ability to manage the thousands of parameters inside of a mixer or configurable DSP device.

In this paper we describe some of the important aspects to be considered in the design of a communication protocol to control audio systems. As the audio industry must piggyback on the developments in the larger telecom and consumer electronics world, we offer both positive examples to be emulated and problems with the existing standards. In each issue, we present the strategy and approach we employed when developing HiQnet™, the Harman Professional communication protocol.

2 TRANSPORT INDEPENDENCE

Transport independence is a desirable network trait because it allows the same core code to be deployed on devices with the appropriate network interface for the intended market and usage. In the installed sound world, Ethernet and serial devices prevail. In the studio, IEEE1394 and USB are common interfaces. For simple control, a USB product can be built more economically than an Ethernet product, allowing inexpensive digital products for the lower end of the market.

There are main methods of assuring transport independence for a networking protocol. The first is to build your protocol upon network standards that enjoy wide industry adoption such that there in an implementation for every transport. In today's world, the choice would be IP. You could tie your protocol to services in the IP world. To do this for non-Ethernet devices, you could employ PPP (Point-to-Point Protocol)[1] in your serial products. While this would afford the software designers a wealth of IP services, the resulting products would be required to implement a TCP/IP stack. We chose a different approach in HiQnet, where the control mechanisms do not strongly rely on any underlying services. We abstract the network transport to a datagram service, a broadcast service, and a reliable service. Using this approach, we are able to design serial products utilizing the least costly microprocessors possible.

3 DEVICE DISCOVERY

3.1 Automatic Addressing

The problem of Device Addressing has plagued any control system designer. Before you can do any meaningful work with a device, that device must have a unique position in a network address space. Many of the features of audio products place extra requirements on the type of addressing that must be performed. By looking at the methodologies employed in the networking world, we can devise plug-n-play systems that automatically self-address and also meet our needs.

3.1.1 Logical vs. Physical

In the network world we talk about Logical vs. Physical addresses. A physical address is the unique identifier on the network. The logical address is usually a higher-level identifier that may be changed. A physical address is usually unchangeable and less user-friendly. Common examples include the name of a computer and the MAC(Media Access Control) address of the Ethernet NIC(Network Interface Card). In a logical sense, the source address of the computer is the name of the computer itself, and users just think of transferring files to that computer. In the physical sense, the data is sent in packets that are transmitted to the correct MAC address.

In our world, it is common for a system designer to develop a complete design offline, that is unconnected from any device, and then arrive at a jobsite where they send the settings down to the devices. If we are using a closed system, then many schemes could be devised. However, today most new products and protocols are Ethernet based, which causes us to deal with the necessary evils of IP addresses.

3.2 Zeroconf

To understand the wrinkle introduced by IP addresses, it is useful to understand how those addresses are assigned on a LAN. The IETF(Internet Engineering Task Force) Zeroconf Working Group[2] devised a scheme to automatically assign IP addresses such that two devices could be connected and they could start talking with no user intervention. The IP addresses are assigned under this basic scheme[2]

1) Wait a little bit to see if a DHCP server will give us one.
2) If we haven't heard from one, then pick one randomly and broadcast out to the world to see if it is used.
3) If no one else tells us that they are using the address, then start using it. If someone says they are using it, go back to step 2
4) Listen to see if a DHCP server ever comes online, if so, let it give us a new IP address, otherwise keep using the address from 3.

What this means to us, is that if we are being good IP network citizens, we should obey the precepts outlined by Zeroconf and do the same with our audio products. As a consequence our IP addresses can only be considered transitory at best, theoretically they could be different every time that we reboot the system. To meet that requirement that the integrator be able to design the system without any physical devices connected and simultaneously be good network citizens, any protocol developed must rely on a higher level logical addressing system whereby the logical addresses from the offline devices may be mapped onto the actual physical devices of the installation.

3.2.1 Adding Plug-n-Play

The next use case for addressing is the requirement plug-n-play for simpler systems. In this case, the user has a handful of devices that he wants to take out of the box, plug together, and start controlling automatically without any user intervention of the addressing scheme. One simple solution is to have the devices automatically negotiate their higher level protocol logical addressing in the same manner that they negotiate their IP addresses. This is the scheme we have employed in HiQnet. Our HiQnet address can be negotiated in a variety of ways that borrow heavily upon the principles of Zeroconf and work in conjunction with standard Ethernet products that obey Zeroconf rules.

3.3 Device Announcement/Discovery

After a device has an address, it can now tell you "I'm Here". Device announcement typically involves some kind of broadcast mechanism or roll call sent out on the network. Typical issues to be solved include robustly handling devices joining and leaving the network as well as a failsafe

mechanism should a controller miss an announcement of a device. Care must be taken that these schemes scale, a simple scheme that works well for a handful of devices can easily fall apart when hundreds of devices in a large stadium installation flood the network with broadcast traffic.
To see how difficult this is in the real world, just look at the number of additions and corrections to the UPnP spec regarding announcement.

3.4 Network and route mapping

An important aspect of device discovery is to discover its network ports or interfaces and possible routes to control the products. A common problem is a mis-configured IP address or subnet mask. In the computer world, the user is expected to clear or reset these himself, however this is less than user-friendly when your device is not a computer sitting on your desk but an array of speakers already hanging in the air. A robust and user friendly protocol must contain methodologies for discovering these network interfaces and changing them remotely.

As mentioned previously, it is common in our products for there to be multiple network connections. A product may have an Ethernet connection in addition to RS232 or RS422 serial connections. Multiple network connections add complexity to the device discovery mechanism in two ways. First, it would be advantageous to be able to use one network port to configure another. For example, a user may plug a computer directly into a USB or serial connection on a product to configure the IP address on the device's Ethernet port. Second, if there are multiple network ports then it is easily possible for there to be multiple routes that a control message can take to reach the device. If it is possible to bridge control messages from one network to another through a device, the user may not even be aware that he has another route to a device. Rather than force the user to only have one connection, it is beneficial to develop a system that seamlessly discovers all routes to a device and arbitrates to the least costly or most direct route.

Unfortunately, the computer world has not managed to address these issues in a standardized way, thus leaving the problem unsolved. Under the UPnP guidelines, a device that is discovered on multiple networks may be presented to the user in a variety of ways, including both one device with two network interfaces and multiple devices each on the separate interface.[3] By not strongly specifying how they are to be presented and controlled, this essentially leaves the problem unsolved, anyone making a generalized control application will be uncertain of how the embedded device will present itself. Another problem with the UPnP scheme is that the announcement methodology can require at least ten seconds to discover all the interfaces, with each interface discovered individually.[3] Network congestion could delay this discovery time even longer. Because of this, a controller designer either has to wait an additional indeterminate time lag to present the device to the user, or simply allow the device to change its presentation as each interface is discovered.

In HiQnet, after discovery of a network route, we have a simple mechanism that allows us to query for all network interfaces as well as change the configuration of those interfaces even with only one connection to a device. This affords remote management of network configuration as opposed to requiring that front panel or dip switches must accessed on each unit to properly configure it.

4 CAPABILITIES DISCOVERY

4.1 Features

Capabilities discovery has long been the holy grail of networked systems. If we could automatically discover the features and functions of a device we could build human interfaces for a previously unknown object or have one device control another without any actual functional knowledge. This

lofty goal has always proven difficult to implement, as increasingly complex devices resist attempts to be self-describing.

The basic concepts of capabilities discovery are well known. Firstly, a model is developed of a generic device. This model usually offers a way to sub-divide the device into useful objects that contain parameters and operational methods or functions. The functions themselves will require parameters, have specific return values. For example, the now-defunct AES-24 standard used a class hierarchy where objects were defined to have a series of properties, methods, and events[4]. The properties defined the functional state (gain, freq, etc) of the object. The methods were used to perform operations on the objects and the events were used to update other devices that were interested in knowing changes in the device.

After a model has been developed, a query mechanism is designed to allow interested parties to discover the pieces of the model. UPnP uses Simple Service Discovery Protocol (SSDP). A device, when plugged into the network, uses an SSDP announcement to advertise a pointer to an XML descriptor description file, which has information on the different properties and services of the device. [5]

4.2 The problems with the standards

The problem of capabilities discovery has been attacked many times from many different angles. Unfortunately the complexity of our devices and systems as well as their operational characteristics has not usually fit with the design of any of the existing standards. Most of the standards are aimed at consumer electronics, lured by the potential volume of business. Simple problems crop up right away. For example, UPnP only allows a 20K descriptor file, completely inadequate to describe most DSP matrix problems, let alone a large console. Also, if your main devices are DVD players and TVs, you never imagine that the product fundamentally changes its functionality. If we were to use UPnP for a DSP product that changed its configuration with a preset change, we would be required to issue a ByeBye. Essentially we tell the world that we are going away, and then come back as a new device. Imagine that in the middle of a show, you change a preset and the device goes off the network then comes back in ten seconds or so.

HiQnet has been designed to handle capabilities discovery with a robust handling of changing configurations. One of the patent-pending features of HiQnet is its scalable mechanisms for discovering the required details of the device. From simple wall controllers that employ a handful of messages, to a sophisticated self-building graphical interface, we have the messages in place to describe audio products.

5 ERROR REPORTING

Besides providing for discovery and control, it is also critical for a distributed control system to provide robust error reporting. Besides providing error messages for basic protocol errors, the system should also be able to notify of significant events. Things like loss of AES sync, clip, or thermal overload should be transmittable on the network. An control devices increasingly becomes distributed , multiple clients need to be able to register for these events, so that a client anywhere in the system can notify a user of potential problems anywhere in the world.

6 ALPHABET SOUP

There are many competing industry initiatives and standards, with multiple alliances and consortiums. For the home, the Digital Living Network Association (DLNA) champions UPnP and has driven the development of standards for many things such as AV and HVAC devices. UPnP is fundamentally built upon IP, so naturally favours Ethernet networking. HANA (High-Definition

Audio-video Networking Alliance) champions the use of IEE1394.[6] Jini™, championed by Sun Microsystems and built upon Java™, is another competitor to UPnP, seeking to also simplify the networking of devices.[7] Many other initiatives have come and gone, leaving in their wake a host of bold vision statements and promising outlooks for the future home, but little actual working hardware.

7 HIQNET™

Due to the deficiencies of the existing standards, we have developed HiQnet™, a control protocol designed for audio systems. HiQnet is transport independent, we have implementations running on Ethernet, USB, and RS232. HiQnet provides a mechanism to negotiate logical addressing that works seamlessly with the auto-negotiation of IP addresses as advocated by Zeroconf. In addition, we provide a rich discovery mechanism and error reporting that meets the requirements of configurable, changeable products.

8 CONCLUSION

Designing a distributed, intelligent networked system of audio products includes many different challenges. Competing priorities of ease-of-use and flexibility add to the issue. Unfortunately, no industry standard at this time seems to fit our needs. These standards have been mostly designed with fewer and simpler devices in mind, not allowing for the complexity we require. By looking at industry standards, we may both adopt the practices that fit our needs and allow us to work better with standard computer products as well as model our behaviour on the standards that may not be as flexible as we require, yet still retain the important characteristics. In particular, Ethernet products should abide by the Zeroconf standards and practices. HiQnet was designed for complete audio systems control and follows the concepts outlined in this paper.

9 REFERENCES

1. W. Simpson, IETF RFC 1661, Internet Engineering Task Force, 1994
2. Stuart Cheshire, “How Does Zeroconf compare with ViiVv/DLNA/DHWG/UPnP”, Zero Config.org
3. UPnP Device Architecture 1.0, UPnP Forum
4. AES24-2, Audio Engineering Society, (1999).
5. “Understanding UPnP”, Microsoft, (2000).
6. Bill Rose, “Entertainment Networking for Consumers, A Reality Check”, HANA.org, (2006).
7. Frank Sommers, “Dynamic Clustering with Jini Technology”, JINI.org, (2006)