A theoretical “Recipe Book” for AV over IP networks

Disclaimer/apology/motivation

I’ll make it clear – I loathe “step by step” “cookie cutter” config guides. However, I do appreciate a bit of guidance can be a catalyst for a conversation. Please please please, don’t follow this blindly! Please ask questions, please challenge my views, and understand THIS IS NEVER DONE!

I’m going to describe how to go from a “flat” network, through small segmentation, to greater, more secure, and more reliable methods to approach the zen of “fire and forget”.

Step1 layer1

The Physical Layer – the “problem” that used to exist here was Collision Domains. This was solved at Layer 2. Aside from the infinitely fascinating world of transmission over media. Day to day – cat cable 1Gbps 100m max, >1Gbps (for pro) fibre – and in 2025, single mode please. Wavelength Division Multiplexing is fascinating, as are PSK and FSK; refractive indices; wave propagation; transistor dynamics; impedance characteristics; intermodulation and a million other detailed aspects. Some of these are considerations that are essential in practice. At the risk of deviating inappropriately, we should press on!

Step 2 layer 2

This is where (of course) switching happens. Notwithstanding marketing, we have a mapping between layer 1 and layer 2 – essentially, we map MAC Addresses to physical connections. This manifests in a MAC address table. We can do VLANs by segmenting a MAC address table. This is where we can segment “broadcast domains” in short, we have broken the network up into different regions that can’t pass broadcast traffic to other regions.

OK – just to make it abundantly clear – we don’t have IP addresses yet. Forget about them. You now have a network that functions MAC address to mac address. This is a network. AVB in its 1722 incarnation works like this, as did cobranet… but… we see IP addresses as an intrinsic part of networks. Why?

To be honest – this is because we use desktop or laptop operating systems. These are “full stack” implementations – up as far as layer 7, and beyond! The essential issue is that this “necessary evil” confusion is the root of most “network problems. We’ll get onto how we work with L3 and above, as long as we are entirely conscious of this.

Step 3 Layer 3

This is where IP addresses live. Whether it is IPv4 or IPv6 is entirely irrelevant. Layer 3 describes “packets” that are contained within “frames” (layer 2) are moved between different network segments. There has been much trepidation about “letting this loose” in AV o IP. This is because (traditionally) “routing is hard”. Why this perception? 1. Having a full stack appliance – like a Dante card, or a RAVENNA interface, we have some very specific challenges. These are only partially to do with Layer 3 (the Network layer), they aren’t so much to do with Reality in 2025. They have been (completely legitimately) a much larger issue with older networking hardware.

Pause – we’ve covered the high-speed part of the stack!

This seems an appropriate time to pause and take stock. This is deliberate. 1. This is where my preference for IPv6 is explained 2. We have to deal with what is available to you today (likely IPv4). This isn’t some kind of mocking the old banger – car paralell. We are concerned with delivering the best to our clients. Networking is only a small component of this, so if you don’t need a “moonshot” today… do something that adds more value! However, the jump to “moonshot” is actually very (scarily) small. Wherever you are in your own evolution, you are more equipped to make some “cold light of day” choices, without growing the sticking plaster layer. The best thing is- likely 90% of your existing devices can be re deployed in a modern context!

The ”split”

Initially we’ll look at IPv4 and IPv6. IPv4 – I’ve done it- I’ve taught IP subnetting to thousands of people. (in IPv4). I’ve taught VLSM (variable length subnet mask) as this followed a more traditional networking course. There is nothing wrong with this. Now we will show why, whilst useful, we really only have simpler choices.

1. Broadcast domain – at the IP layer we (theoretically) have broadcast IP address for an IP subnet. In IPv6, we don’t have IP broadcast traffic! This means that – we have to scale safely in IPv4 – remember every device can issue broadcast, which is flooded throughout the segment. It is possible that Broadcast can be “ignored” by stack implementation, but in IPv4, it still takes bandwidth on the wire. Practically we are “safe” in a /24 subnet (254 useable addresses). This means that this sets the upper bound of our number of devices in a subnet.
2. Multicast – With various “boltons” notably IGMPv3, IPv4 is much more like IPv6. This is because “source specific multicast” is possible in IPv4 (theoretically). However, few IPV4 implementations actually adopt SSM! IGMPv3 doesn’t mean SSM!

“Typical” AVoIP implementation in the field.

I’m describing “typical”. If you are doing something more sophisticated – please skip on. This is the findamental problem with “cookie cutter” solutions – none of them fit! Sadly, many things are misunderstood.

An IP address is NOT a “spare”/additional MAC address!

There is a notion that 1. All devices should have static IPv4 addresses, 2. Any device needs to be deployed on any configuration of a network. The MAC address IS the GUID (Globally Unique Identifier). IP addresses are best deployed dynamically for hosts. Whilst I fully Appreciate the desire to take ones entire inventory, and make a “big” subnet (maybe /16) this is quite dangerous. By using /16 instead of /24 you create a broadcast domain of 65,534 host addresses. This means that a huge amount of broadcast traffic can exist, largely unmanaged!

I like when people challenge ideas with "what ifs" this is very positive. Fundamentally it means that I have explained sufficiently simply, that when introduced to a new concept, you can formulate an objection - this (to me) means that my message is getting through, rather than "authority through complexity" thank you, I like this!

DHCP Dynamic Address Configuration Protocol

There are many options in DHCP. We’ll (initially) deal with the simple function of “how do I get an IP address?”. We’ll deal with some long-held myths, and why they actually were real, how they have been addressed, and how it is kept (apparently) simple.

A host (a laptop of AVoIP) device issues a dhcp discover message. For ancient reasons, this is sent as a broadcast Frame (layer 2). This arrives “everywhere” – this is a common reason for wanting to scale the segment – eg to a /24.

Imaging a cold power up event, which launches broadcast frames from 254 devices! That is a LOT! It is done this way to maintain backward compatibility with the Bootstrap protocol (a predecessor to DHCP).

The DHCP server then (should) check for devices with the IP address it is going to offer the discovering device. Generally if the device has been “seen” before, the offer should be the last IP address offered to that MAC address.

The receiving host gets the offer, and then Must checo for other devices with that IP address – this is by issuing an ARP_discover message (again Broadcast). If it receives no response, it issues a dhcp_ack (acknowledge) message. Then carries on using the address it has accepted. IF it gets 2 offers (maybe there are 2 Servers) it will send a DHCP_NACK message (non-acknowledge), this tells the server that it wont use that address. This is sent broadcast, so if there are 2 or more servers, the server being rejected knows to not carry the record. The other servers become aware that the NACK was not meant for them.

Now, I did say there were historical “bugs” in DHCP server implementations. Now they have been fixed, we shall spare the blushes of those that “did it right” (in the end) and acknowledge (in a more multicast sense) that this process was addressed in about 2012! So long standing myth addressed!

“ok, so someone fixed an error”, looking at the process, how do I know that my dhcp server is this thorough?

This is a very good question – some lower cost hardware exploits the “should” word (ie for should, read “doesn’t bother!”) for sanity, I recommend using the ISCDHCP4 server implementation! This is tested, and deployed on millions of networks worldwide. It is open source from the Internet Systems Consortium, and is implemented in amongst others OPNSense, as well as on many Linux Stacks, and downloadable from many repositories.

DNS – on a “show” network?

DNS is a useful system. It’s the thing that gives an IP address to your browser when you type www.google.com Hand writing the famous bind9 dns server, can be painful (especially if using vi like I did for a long time. Thankfully, this can coordinate with your dhcp server! This means that a server can (attached to its offer) ask for dhcp option that asks the receiving client to reveal its hostname. This can then be placed into the A record list on a DNS server – such as unbound DNS (which is also in opnsense) or it can be used through some work, to populate your very own bind9 service (if you like command lines!

Wait- DNS – that’s like google right? I don’t want the world knowing about my IP addresses – hackers! Security!

That is a good point, and I’m glad you are paying attention to security! If setting any IP address scheme, please use IETF RFC1918 private IP addresses! These WILL NOT be passed to the public Internet – these ranges are 10.0.0.0/8 172.16.0.0/12 and 192.168.0.0/16 this gives plenty of IP addresses. It is also why your home router may well have the IP address 192.168.1.1, millions of others do, because it falls within rfc 1918 range (it is your private IP address, not your public IP address.

An important clarification on the “pause”

DHCP, and DNS are “Services” they don’t live in the OSI 7 layer model “stack” these are software applications, this means that the messages have to leave the “network stack”, bounce around in a server, do some computational work, and return results to the client via the “network stack”. So we’ve skipped over ports, and some other stuff in “session” layer, “presentation layer” and “application layer”. On most regular AVoIP networks, we don’t need to do much/any work above layer 3 (Network/IP layer).

OK – a deviation, and a chance to digest! (I also like to “go on”)

This is possibly somewhat different to most “networking courses”. Please indulge me – the IETF (the Internet Engineering Taskforce) is the “authority” that “governs” the Internet. It is a “standards” body like few others! An RFC a “request for comment” is the IETF version of a “standard”. ANYONE can submit one! Feel free. However, it is a “consensus led” group. You can post an RFC – if you are talking nonsense, you will likely be ignored, and will disappear into obscurity! Worst case, someone might attack you (unlikely) the RFCs that gain traction are written by top tier researchers, but it maintains academic purity by being an equal forum, but as Orwell noted – “some pigs are more equal than others”. There was an April 1st RFC tradition. This shows the underlying humour of IETF – notable RFCs include RFC 2549 https://www.rfc-editor.org/rfc/rfc2549.html this is funny (to me) and contains many humourous quips (dad jokes). This is how these things should be!

Finally – the "how to"

Assumptions

Remember the old saying – "assume makes an ass out of u and me!" This "how to" read without the prior content will perfectly achieve this! This is probably the most truthful thing anyone will ever tell you, when it comes to data networks!

Step 1 – get vlans sorted!

Lets define what we need to separate. Lets then consider what we want to separate. Often we feel ne "need" to separate more than we actually do! Here is the method.

1. Which services actually conflict with each other in 2025?
    a. Actually surprisingly few!
        i. Dante Primary, and Dante Secondary MUST be separated
        ii. Most other things will be happy – some implementations of "redundancy" may work slightly similar to Dante, check for hard technical requirements!

2. OK, what "might" I want to separate
    a. It makes sense to separate areas with zero overlap – eg – just because lighting might well work in same segment as audio… why would anyone need to do both at the same time. Even if they do, there must be a way to maintain separation.

This uneasy feeling of wanting to separate, despite there actually being no technical reason, is entirely understandable. Actually, it shows a security mindset. This is very positive! However – please don't just get another network interface for your machine, connect that to another VLAN and think you have achieved separation (you Haven't). and you have done the most dangerous thing imaginable, through admittedly good intentions!

Having multiple network connections would put an enterprise network engineer into a cold sweat. Using a desktop operating system in this manner effectively creates a Frankenstein router. The desktop operating system doesn't think it's a router. However, if it sees multiple connections to the public Internet, it will start to behave like one. Have you ever had "intermittent control?" yes this is the desktop OS doing precisely that!

Sanity – its time to add a router!

What!? Add a router – you are mental! Routers are expensive, complicated, and unnecessary

No, a "router" is free. Just set up a software router on a virtual machine, and enjoy!

Shadow boxing – virtual machines are expensive and complicated!

They can also be free. It is true that ideally one would have a dedicated server (or a cluster of servers) to mimic enterprise networks. However, this recipe book aims to introduce the concepts without undue financial risk. You can install vmware player (free) run an opnsense VM (free) and because you know "all about VLANS" you can start routing!

Software routers? Latency!

Yes – you would have been right (in 2005). In 2025, a software router like opnsense works at "wirespeed". Yes, you may (at this stage) consider a second-hand enterprise server – this has a much larger process capability, the option to install a type 1 hypervisor (like proxmox – also free, but please pay them – its not much, and they deserve it!) you could run vmware player – just to try it out. My main concern would be power supplies. True, the old "enterprise" cpus (xeon etc) are good to just be left switched on for years, but modern CPUs are also very resilient. The power supplies are the issue, and a second-hand dual power supply case is a very good answer, its not so much (anymore) the longevity of the CPU itself.

OK so do I need to understand Routing protocols, these seem hard!

Well, in essence – NO. for most AVoIP networks, static routes are all you need. You aren't dealing with a hyperscaler datacentre (and these guys run IPv6 natively, with BGP – they like simple too!)

My network is really big. This seems too simple.

If your network is really big… you need help. If you are following a step by step "how to" guide, you are failing to prepare, and therefore preparing to fail. "Really big" is hyper scale – this is millions of endpoints. If you have invested in this, you can afford an architect! If you are running a few hundred (big in AV) you are fine!

What about security? I still don't get why I shouldn't have separate cables for separate services?

Fine. Maybe the best way to stay secure is to use remote desktops. Have a dedicated control OS that lives in the network with your Dante devices, another for NDI tools, etc. Then, enable remote desktop, and run another VM that can do Apache Guacamole, or rust desk. This gives you a safe link through your router (and firewall, if running opnsense). This is your golden protection against hackers and malware! By having an RDP server between your actual control machine, and your desktop/laptop/tablet, any email attack may infect your viewing machine (nasty) but… it doesn't attack the "real machine" which stays running, and doesn't know anything happened!

You can start to secure your network from a sanity perspective. And the best thing is (windows license fees notwithstanding) its free!

You said this is a recipe where are the actual instructions?

Right below – there is no point me re publishing my own "spin" here are some people that do this all the time!

Proxmox - Installation - Proxmox Virtual Environment

Opnsense: Get started - OPNsense

Guacamole Unlocking Remote Access: A Comprehensive Guide to Installing and Configuring Apache Guacamole on Ubuntu | by Anshumaan Singh | Medium

Rust desk RustDesk Server Pro – Documentation for RustDesk

And of course you can always email me website@kdwalsh.com