The Hierarchy of Latency

Greg Shay
CTO, The Telos Alliance
Cleveland, Ohio, USA
Greg.Shay@TelosAlliance.com

Abstract – “Low Latency” is an overused term and hides important technology design concerns and constraints. To bring clarity to the design of modern digital audio, audio networks, and cloud computing, this paper defines and describes multiple aspects of the meaning of latency in the context of modern audio production, with example numerical values. A one-page chart is presented, visualizing the range and meaning of the Hierarch of Latency.

Introduction

The next time the words “low latency” comes out of somebody’s mouth, stop them right there and say:

“What do you mean, how low is *low* latency?”

When considering low latency, what is meant by *low?*   One goal of this paper is to set aside the phrase “low latency.” and replace it with meaningful quantitative numbers that can be used to describe the issues we are talking about and problems we may be trying to solve.

The beginning of the answer is the next question: “*Why* do we have to ask?”

We are analog. We live in a continuous, analog world, and typically interact with persons and objects relatively close to us.   The advent of digital signal processing and digital network communications has added *latency* to much of our everyday lives.

Common Everyday Experience

What is probably the most common everyday experience of audio latency? Simply the (limited) speed of sound. And what may be the most common *first* experience of the latency caused by the speed of sound? Lightning and thunder. We were all taught as kids to count “one one thousand.. two one thousand, up to five one thousand” until hearing the crash of thunder for each mile of distance to the lightning.

Since we are in the audio business, let's first dig into the hierarchy of latency of sound, then we will consider control path latency.

Our Shortest Audio Path

What might you guess is the shortest audio path we experience every day? I say it is hearing our own voice. The distance from our mouth to our ears, about 6 inches, is about ½ millisecond. Bone conduction is faster than air, so our brain has to sort out hearing two copies of our own voice, we are usually not even aware.

An analog mixing console has practically zero audio delay, but not exactly zero. All analog filters have *group delay,* which is manifested as actual time delay [1]. The delay in feedback loop response time creates what is called Transient Intermodulation Distortion (TIM) [2] [3]. So don't let someone tell you "digital has the problem of latency, but analog did not." The design of high performance analog signal processing filters takes quite a bit of careful effort keeping analog group delay carefully controlled.

With the advent of digital signal processing, digital filtering, digital mixing consoles, and now digital network connection of audio signals, it is true that signal latency delays have increased significantly. In the design of digital filters, group delay has to still be managed, just as in the analog case, for proper response and low distortion. But with enough latency, you can begin to *hear* it.

Consider the case of hearing your own voice from a microphone to headphones. With low enough audio latency, nothing is noticeably different (remember the ½ millisecond that we are already used to hearing our own voice). As audio delay increases, at first there is the perception of 'something is different', and with more latency, there are worse effects.

Digital Audio Network Customer Acceptance or Rejection

A key transition in the professional broadcasting industry with the introduction of digital console and digital network audio around the year 2002, was to have *no noticeable difference* from what the talent was used to hearing in headphones. This meant the outright acceptance or rejection of the new digital audio network technology. What are the limits of latency to meet this test?

Experience from digital audio mixers and digital audio networks

Self-contained digital audio consoles, consisting of an analog-to-digital (A/D) sigma delta conversion, a local DSP processor computing audio sample by sample, and a digital-to-analog (D/A) sigma delta conversion, at the common professional 48 kHz sample rate, typically can have an air-to-air (microphone input to headphone output) latency of approximately 1.0 to 1.5 ms. This latency was found to be imperceptible (note how this compares favorably with the human head acoustic delay). Higher speed flash A/D and D/A converters can result in less latency but are typically more expensive and don't have as high dynamic range (critical for audio) and were not found to be necessary.

An early "Live" audio over IP network solution was Telos' *Livewire,* and in *Livestream* mode, delivers air-to-air (mic to headphones), sending audio to the mix engine and back, in just slightly under 3 milliseconds. This was judged to sound very "live," thus the name.

As the mic to headphone latency approaches 6 to 7 ms, there is something that can start to be noticed, however most users find it still quite normal, acceptable and not distracting. I often describe this to be something like ”singing in the shower” where you can perceive the enhanced reverberation (in the reflecting shower stall, bouncing around the physical dimensions of the walls 3 to 6 feet apart).

At around 10 ms of latency, a bizarre sounding phenomenon can start to occur, often described as ”your ears getting sucked out.” That’s a real description from real users. And it is distracting and not acceptable. My own guess for this odd phenomenon is that since the typical male broadcaster has a deep booming voice with a fundamental frequency of something like 100 Hz (which has a wavelength of 10 ms), the sound pressure waves inside your head via your ear Eustachian tubes end up out of phase with the sound pressure wave outside your ears from the headphones.

At a delay of about 20 - 30 ms, there is an even worse effect, artificially inducing something like stuttering. It is almost impossible to control your own voice and stop yourself from repeating the ends of words over and over (and over over over.!) Try it sometime, it is definitely a “brain hack.” (This also gives me great empathy for folks who deal with this every day.)

Does audio latency always cause problems?

So are digital audio and digital audio networks impossible to use? Does everything have to remain analog to be usable? Obviously not, since DSP and digital audio networks have revolutionized our industry. But with the above identified phenomenon, how?

The answer is we have adapted what we do and how we work, where necessary. In fact, some of the cutting edge progress of the broadcast industry today is to find new workflows and use case patterns that can further adapt to latencies that occur when taking advantage of the latest technologies and techniques.

We are actually rather familiar with audio latency

Consider a normal conversation with someone across the room. 10 ft apart means about 10 ms of delay (the speed of sound is approximately 1100 ft/s, or 1.1 ft per ms). Raising your voice to have a conversation across a large room can involve 20 – 30 ms of delay, without any uncomfortable effects (other than perhaps straining our voice).

Having a conversation over a cell phone typically has 100 to 200 ms of latency each way [4]. We hardly notice (or we have gotten so used to it.)

In the 1960’s through 1980’s, it was common for an international phone call to be transmitted via a communication satellite in the geostationary Clark belt, 22,500 miles out in space, which introduced a round trip audio latency of about 300 ms, due to the speed of light latency, (not audio) this was probably the first common everyday experience of latency due to the speed of light, electromagnetic waves (not sound waves). 300 ms does start to make conversations a little awkward, stepping on each other's beginning of statements, interfering with the natural ”being in the same room” back and forth conversational flow. Today, phone calls between Europe from North America go via fiber optics and have a fraction of this delay.

But clearly, the most sensitivity to audio latency is for the special case of hearing your own voice.

As long as you are *not* hearing your own voice (through speakers or through headphones), the requirements for keeping audio latency down into the single milliseconds, greatly relaxes. This is key for allowing much of the new, modern, network and cloud based audio production workflows.

Remote musical performance

In the early 2000's I was present at a special performance of a string quartet, where the quartet was split, two each in different cities about 450 miles apart (San Diego and San Francisco), connected by a high speed digital IP network. The physical speed of propagation on a network is about ⅔ the speed of light in a vacuum, so 450 miles / 186,000 miles/sec x 150% = about 3.5 ms one-way, plus the 1.35 ms analog conversion and 250 us packetization delay mode of audio over IP (as, for example, used by a Livewire *Livestream)*. This adds up to acoustic latency between the remote performers of approximately 5 ms. How did the performance go? Flawlessly.

Think about it, from the performer's point of view, the 5 ms delay is the same as if their chairs were sitting 5 ft apart from one another, much smaller than the typical seating spread of an orchestra.

Note that this was a special case of a high performance private high speed IP network that could carry and deliver the uncompressed 4000 IP packets per second with low enough jitter that additional jitter buffering (and additional latency due to additional jitter buffering) was not needed.   More later on the relationship between network performance, jitter buffering, and latency.

Back-deriving from the 40 ft seating spread of a decent sized orchestra suggests professional musicians would be comfortable with remote performance delays up to 40 ms, which would be 5000 miles via a high performance network using 1/4 ms packet time IP audio. That size covers the continental U.S.   Note that musicians in a professional orchestra follow the conductor, and are listening less to and not following each other.

I suspect the performance tolerance is much lower for more "play by ear" rock and jazz combo groups, forming a beat and groove with each other. I remember the difficulty during the *Live Aid* concert in 1985, when the Rolling Stones attempted a live remote performance split between London and New York. What I remember is the struggle to find the beat with each other.

It is reported that a good professional drummer can subtly shift the timing of the beat by *single milliseconds* to alter the 'feel' of the music. With this sensitivity and resolution, the upper limit of latency for a rhythmic live performance group playing together is likely to be more on the order of 5 ms to 10 ms, where you start to hear yourself delayed in relation to those around you, much more like the earlier examples of hearing your own voice.

Control Latency

So far, the discussion has been about audio latency and its effects.   In the broadcast workflow, control latency can be just as important in producing a professional sounding, well-coordinated show. Control latency, at its simplest, is the time from hitting a button or moving a control, until the audio reacts. As with anything you want to carefully control, too long of a delayed reaction makes careful control difficult or impossible.

So how much control latency can there be, until it starts to be a problem? The answer may be surprisingly larger than you might expect.

Around 2012, the BBC innovated a distributed radio production system, called *ViLor* "Virtual Local Radio" [5], whereby dozens of local radio studios around the UK were connected centrally to two redundant central equipment centers. A primary concern was the effect of the long distance time-of-flight, of both the audio and the control signals. Could you really run a live broadcast show remotely from the other side of the country?

Basic research was done with double-blind tests, to gauge the operator perception, and if there was any impairment on producing the live show with remote facilities. A key finding of this research was that if the audible and visual response to a control action is below threshold of around <u>50ms to 70ms</u>, then there is no impairment of operation. It feels natural and normal to a typical operator. If the delay in control action is longer than this, the user starts to think that the control may be broken, or the button press was missed, and they had to press a second time.

Interestingly, this response time latency threshold is around ten times longer than the latency sensitivity of hearing your own voice.

I had a chance myself to test this kind of control latency, running a "cloud" audio mixer in Amazon Web Services (AWS) in Virginia connected to a physical surface at my office in Cleveland, pushing a control button on a console, putting my nose right up close, and punching the button and observing the light up on/off. I could not perceive a delay and I had no feeling like the light was delayed from my punching action. I was genuinely surprised. I figure the round trip latency was about 20 ms, well below the critical threshold determined by ViLor.

Note that 50 – 70 ms round trip is enough to reach neighboring cities within a geographical region, but does not span the entire planet. Time-of-flight latency to the opposite side of the globe and back is on the order of 200 ms, which is well over the threshold of this kind of careful and precise operation.

Human Reaction Time

With these latency values in mind, it is interesting to compare to the documented human reaction time, like starting a race at the sound of the gun. The average reaction time is between 150 ms to 300 ms, with one source documenting an average of 228 ms for auditory stimuli, and somewhat large 247 ms for visual stimuli [6].

Two very interesting points to note here: audio reaction is faster than visual, and that the latencies described so far in this paper that make a difference are quite a bit smaller than human reaction time. This implies we can *perceive* audio latency effects well before we can *react* to them.

Cautionary note: Musicians are not 'typical' users! Recall that the good professional drummer can subtly shift the timing of the beat by *single milliseconds,* to alter the 'feel' of the music. They would certainly notice the 50 ms latency from hitting a drum pad to the sound. I have a story about the drummer of ZZ Top I can tell, if you ask me…

Latency and Wide Area Networks

There are relationships of time-of-flight, packet jitter, QOS, codecs, and buffering to latency. For AES67 link offset, what value needs to be used?

AES67 link offset is the offset in time at an audio network stream receiver to add to an incoming packet's timestamp for playback. It is another way of describing the receiver buffering required, but referring to the common global clock at the sender and receiver instead of the buffer length directly.

Some definitions:

- **Network time-of-flight**: The precision time protocol (PTP) global clock difference between when packets were sent and when they are received. This is how long it takes for the packet to go through the network, based simply on the velocity of propagation of signals via wires or fiber optics. It is not quite the speed of light in a vacuum (300x10^6 meters/sec), it is roughly ⅔ of that speed. The dielectric constant of the air and the insulation on wires and the index of refraction of fiber optics act to make electromagnetic signals slower.
- **Network packet jitter:** The variation of the time it takes the packet to traverse the network and be received. This also includes any variation in time of the sender, if the sender has some deviation in the time that it sends each packet. Jitter can also include the sum of the errors in the sender and receiver recovered PTP time. (If PTP recovery is operating correctly, this is close to zero).
- **QoS Quality of Service.** To minimize audio packet delay in order to minimize buffering and its resulting latency, audio packets are marked with a high quality of service (differentiated services code point, DSCP, tag) value, *and* IP switches are configured to prioritize the output of high priority packets first. By doing this, the expected maximum packet delay in each switch for an audio packet is the network transmission time of *one* maximum-sized packet (typically 1500 bytes). On a 1 Gbit network, this is 12 microseconds. If QoS is not used, there is no limit to the number of packets that may get ahead of and delay the audio packets. On a lightly loaded network, this may not cause a problem, but as the network gets closer to being heavily loaded, this unlimited potential delay can cause audio clicks and pops. Use QoS !

Synchronous mode vs Syntonous mode

To define, *Synchronous mode* means audio samples are transferred and output at known, defined moments in time, which means the frequency *and* phase of the output audio is under control. You can reconstruct acoustic wavefronts and stereo images with multiple separately network connected receivers and speakers, using synchronous mode.

Syntonous mode, means you have an audio sample rate frequency lock, but only the frequency, not the phase or absolute time. With syntonous mode there is always an unknown absolute phase offset in the output audio. You cannot reconstruct acoustic wavefronts or have consistent stereo images from *separate* network connected speakers using syntonous mode. Although it should be noted the phasing and imaging is stable and correct when carrying all the channels in a multichannel syntonous stream that is output to multiple speakers from one audio network receiver.

AES67 is the Audio Engineering Society industry standard for IP networked audio, published in 2013. In this standard, the terminology "link offset" is used to describe the amount of time that a network receiver *offsets forward* the audio samples it receives, to compensate for the finite time of flight and network delay of that network 'hop', or 'link'. The 'link offset' is conceptually closely related to the size of the receive buffer, but I proposed the clear distinction in terminology as to not further overload the use of the terms ‘delay’, ‘buffer’ or ‘latency’. (I was on the AES committee X192 which then released named AES67).

For synchronous mode: the AES67 link offset must be set larger than the longest packet travel time. The longest packet travel time is the time-of-flight plus maximum jitter amount.

For syntonous mode, the AES67 link offset has a slightly different interpretation and means the size of the receive buffer. The size of the receive buffer must be set larger than the difference of the maximum from the minimum packet jitter. In syntonous mode, the time-of-flight does not affect the required receive buffer size.

The AES67 link offset directly sets the latency of the audio connection. Every millisecond of buffering adds a millisecond of audio latency.

Time-of-flight delays are relatively constant between given locations because the physical distances don’t change. But these delays can be affected by wide area network routing changes. The packets you send from one city to another (for example Ohio to Germany), may be taking an unexpected route (for instance through an international network gateway in Chicago), with additional distance. And that route might change unexpectedly.

What are typical wide area network jitter figures? The key is to realize is that the typical wide-area network (WAN) does not implement QoS (nor does the internet). As mentioned above, without QoS, jitter can be unbounded and is typically related to network load. A packet that is delayed is stored somewhere in memory, either in a switch or a router, and for moments where a burst of traffic is greater than the outgoing link can support, incoming packets are held in memory until they can be sent. Many links for a long distance path, each introducing potential routing delay, can add up to a higher packet delay. As network capacity is increased, there are fewer occurrences of not having enough link capacity, and so with higher capacity networks, jitter is typically less.

A privately managed WAN can in fact implement QoS, and packet jitter then becomes a function of how many intervening routers or switches there are in the path *and* the network transmission time of one full sized packet, per each network link hop. Just how well the public WAN (internet or unmanaged WAN) behaves is a moving target and is in general getting better over time as the networks increase in speed, until it doesn’t. It is unpredictable. As mentioned earlier, if the internet is behaving well at the moment, it’s jitter can be a small number of milliseconds (plus the time-of-flight). But during loaded times it can be seconds. This is why you cannot create truly reliable audio connections over the internet without multiple independent paths to increase the probability that not all paths are slow at the same time.

Reliable network protocols vs unreliable : UDP, SRT & TCP

UDP

User datagram protocol (UDP) is the simplest network protocol, but it is unreliable because packets are sent without any regard if they reach their final destination or not.

The number of audio samples in each packet creates an audio latency called the *packetization latency.* Sending smaller packets results in lower latency, but since each packet has around 72 bytes of overhead, network efficiency gets very low if sending a very small number of audio samples per packet.

Some typical packet sizes are the AES67 interoperable standard of 1 ms (48 samples @ 48 kHz), Livewire *Livestream* IP audio packets of 250 us (12 samples), and SMPTE 2110-30 minimum packet size of 125 us (6 samples). To maximize network efficiency, Livewire also can use packets of 5 ms (240 samples), which for stereo 24 bit audio results is a close to maximum ethernet packet of 1500 bytes.

Keep in mind, the end-to-end latency of a digital audio network point-to-point ”hop” is larger than just the packet size, typically 3 times the packetization latency, plus any additional network jitter buffering, if needed.

Using these same packet sizes with reliable protocols can increase the overall latency, due to allowing time to retry sending of missed packets.

SRT with Retry

Using the secure reliable transport (SRT) protocol, if there was a missing packet, it takes 2 times the one-way propagation delay for the information to be communicated back for the sender to resend, and then the resent packet to arrive at the receiver. For a *very* reliable protocol, there might be two retries, or even an unlimited number of retries possible (TCP uses unlimited retries).

Consider the following: If the longest one-way propagation on the planet is around 100 ms (for instance Australia to London), then allowing for one retry would require 100 x 3 = 300 ms. With two retries, 500 ms. So for instance, if you use 1 second of buffering with a protocol with retry, you can have practically error free, reliable audio protecting against packet loss. If your application can accept 1 second of latency, essentially error free audio to and from anywhere on the planet is possible. (See RFC6298 Appendix A).

Now, does everyone believe what I just said? I’m sure you are familiar with the (big) loophole..

Anyone who has used the web, would be happy if all web accesses were less than 1 second.
While slowness using the web could be a slow server, we have all experienced the communication timeout.

The 1 second maximum latency is assuming the long-distance network is operating normally, and each node is forwarding packets as fast as they can without extra delays. The loophole is that of course the network has many reasons when it doesn’t operate normally, and significantly larger delays are possible as the network routing reconfigures itself as links go down and rerouting automatically happens. This is both the special strength and the problem with the global networks that evolved from the military originally designed ARPAnet designed to withstand a war.

TCP

The transmission control protocol (TCP) and hypertext transfer protocol (HTTP) (also HTTPS) are used for most web browsing and most other long-distance access across the network, and *are* designed to expect and work around the typical longer delays of the global network. TCP has something like a *90-second* timeout, during which it keeps retrying (like SRT), until it gives up. Similarly, HTTP gives up with one of the “404” type errors.

TCP also gradually slows down the rate of retrying (using exponential backoff), so that if there were very many TCP connections attempting retries, the retries themselves do not *create* a traffic overload problem.

This TCP 90-second threshold dates back to the creation of the TCP protocol and the ARPAnet itself in the 1970’s. Surely, networks are faster than they were 50 years ago? But things still go wrong today. Cables fail, hardware fails, software gets updated, servers get taken offline for maintenance, usage and traffic congestion have gotten worse not better. So what is the latency of something broken getting fixed? That may be forever. I suggest the 90-second rule represents the threshold of *human patience,* not so much how long the technology takes to correct its operation. It’s how long we are willing to wait to see if the connection goes through before the human being gives up and tries something else.

TCP can almost deliver the same end benefit of retries, like SRT, but the design of TCP to wait up to 90 seconds for the data to come through, makes TCP problematic for real time streams. When packets have not been coming through for longer than the maximum global time-of-flight latency of 200ms, something more serious is wrong, another method might be tried (like an alternate or redundant route), rather than just waiting. This is the difference between TCP and SRT.

Cloud Computing

Which cloud location? It's not really a “cloud” after all, but a collection of servers at various locations. For fault tolerance purposes, all the servers are not at one location (e.g., to minimize impact of earthquakes and storms). Using a nearby server located in the same geographic region limits the round-trip time-of-flight latency, for latency sensitive applications.

Cloud computing performance depends on shared vs. not shared computing resources. When you pay for cloud computing, there are typically different costs for different tiers of service. The least expensive tiers can mean sharing computing resources with others, simultaneously. For security reasons, these context switches can be considerably slower than the typical operating system context switches. When you hire a ‘non-shared’ compute resource, the computing latency is essentially the same as a local virtual machine, typically < 150 us. Experiments with shared cloud computing resources have in the past shown latencies upward of hundreds of milliseconds, to almost 1 second. This depends completely on the cloud provider's service level agreement (SLA) and is sometimes not well defined.

Cloud computing also may reserve the right to perform maintenance at any time and take the CPU you might be using at the moment offline, to be migrated to another. When an application is designed around performing work in stateless increments, a server will always be available for the next work increment, regardless of maintenance.

Plesiochronous Clouds

FIGURE 1: A PLESIOCHRONOUS PLESIOSAURUS …!?

No, not a dinosaur !

From Wikimedia:
In telecommunications, a plesiochronous system is one where different parts of the system are almost, but not quite, perfectly synchronised. [7]

When something is plesiochronous, it means the timebase for that part of the system is different from the global timebase, but the difference is bounded. It is not drifting continuously in one direction or another (that would be asynchronous). The plesiochronous time may wander, getting ahead or getting behind, but it does so within a limited range ahead and behind. For example, a device that is plesiochronous may drift a few milliseconds ahead or behind, but overall track global time, always staying within a few milliseconds.

In reality, all systems are technically plesiochronous, because no time synchronization is perfect, there is always some synchronization time error.. But with a typical PTP follower, the sync control loop is always acting to drive the synchronization error to zero (the error is zero mean average). In a plesiochronous device, the time offset may stay nonzero (but bounded) for a long time (the synchronization error is not zero mean average.)

A practical example of the plesiochronous situation is with remote server or cloud computing. All the media locally is precisely synchronized to a PTP grandmaster that is synchronized to the global GPS timebase. The central or cloud server may not have access to PTP, yet its timebase may be able to be synchronized to the global atomic clock through some other method, using for example network time protocol (NTP), Chrony (a better behaved NTP), or some other private method. The result is a plesiochronous system. The clocks between the local media and central server do not drift apart over time, but neither are exactly matching each other, as they each have their own bounded tracking and wandering behavior.

I think of this as ”islands of time.” ”Islands of time,” internally self-consistent, but with a possible time offset from everything else. If this offset is comparable in size to the time-of-flight round trip to and from the cloud (or remote server), the plesiochronous unknown offsets can effectively be absorbed into the buffering, without affecting the workflow.

An example with numbers. If the roundtrip from on premises to the cloud data center in another city was for example 30 ms, and the cloud servers are using Chrony which keeps the time within, for example, 5 ms, then the exchange of AES67 streams would need to use a link offset of the sum of the time-of-flight plus the expected maximum cloud server time error. The plesiochronous time error gets absorbed into the link offset buffering of the time-of-flight. And if the time-of-flight is the dominant factor, the fact that the cloud server is not precisely PTP synchronized has little penalty.

Audience experiencing latency effects

There are a couple of relatively new special cases of the audience being affected by latency, which may be generally yet unresolved.

Watching sports games within earshot of your neighbors

If you and your neighbors (apartments next door or backyard deck parties) are within earshot of each other, and each watching the same game through different delivery channels (different streaming services, or one off-air and one streaming), it can take away some of the anticipation excitement of a game when you hear your neighbors cheering before you see the end of the play. Hard to put an exact limit figure on this, but it is probably related to the human reaction time, cited above.

Sports betting

As in the classic motion picture “The Sting,” a delay in hearing or seeing a real time betting event may open a window for someone to place a bet after knowing the outcome. With some streaming delays of typically 30 seconds to a minute (and the betting house wanting to allow betting until the last possible moment), this becomes a real issue.

Attending an event in person, while listening on your personal device

This is a rather recent modern technology use case, but it has really been around for decades when choosing to listen to a favorite commentator on a local radio station broadcast rather than (or in addition to) the in-stadium announcer. For a sports event, the delay while listening to your personal device must be small enough for the play by play to be relevant, perhaps ½ second.

Workflows

Given the hierarchy of latency, what does this mean for audio use cases and workflows?

Outlined below are common live audio production patterns, and how they relate to the different latencies involved. New technology has enabled new use cases and new work patterns to be innovated.

- **Mixing a live show:** The main latency affecting mixing a live show is the control latency, with the threshold of 50 - 70 ms. As you make changes and adjustments, you critically rely on audible and visual control feedback. This latency range enables remote mixing, from another city, generally in the same geographic region, but not from anywhere on the planet.

- **Mixing a live show including your own voice:** This has the tightest limits on latency, 5 to 6 ms. Taking into account the latency of the digital mixer, converters, and audio network hops, leaves little ‘latency budget’ for time-of-flight remote operation. When mixing with your own voice, you are generally limited to the same physical location, on the local LAN.

  There is a way around this limitation, so that you can mix and produce a live remote show with your own voice, using a mix-minus. A mix-minus is a mix from the remote mixer including the complete mix you are controlling but without your own voice. A secondary small utility mixer, physically local to the operator, mixes the mix-minus with the voice and delivers to the headphones.

  Note for this to work correctly, to have the voice level in the final program correct, the full program mix and the local utility mix have to use the same voice mix fader level, as controlled by the operator. This is easily accomplished with modern digital network-based mixing equipment, as the audio, the control, the mix-minus, and the voice mix level that goes to the two mixers, are all carried on the network.

- **Mixing a live show with a conversation between multiple people:** When mixing a live conversation, the latency between the individuals can be what is typical of cell phone delay, 100 - 200 ms. Indeed, they may be on cell phones, remote, or they may be in separate studios, on mic with headphones.

  Each person typically has a mix-minus of the program without each of their own voices coming back to them, and a local mix of their own voice to hear themselves comfortably in their headphones. Note it is *not* critical in this case that the relative mix level of their own voice matches the final program mix. The overall producer mixer will make sure the voices are balanced with each other and to the program as a whole. The individuals just need to hear themselves.

  The mixing operator needs to be within the 50 – 70 ms control latency limit. If the mixing operator is part of the conversation, then they will need a mix-minus of the program and a local mix, as in the previous case.

- **Delivering a live show to a listening audience:** In principle, there is no real limit to the latency when delivering a show remotely to an audience, as long as the latency of the audio and any video are the same. Many streaming services operate under this assumption and use streaming buffering delays of tens of seconds or even a minute or two, to simplify the work of duplication of streams to an audience of potentially millions. That is, until the above mentioned audience latency effects started to be noticed. In the context of streaming delivery, “low” latency can mean 15 seconds, one of the tortured overuses of the meaning of “low.”

In summary, Figure 2 is an illustration map of the range of latencies discussed.

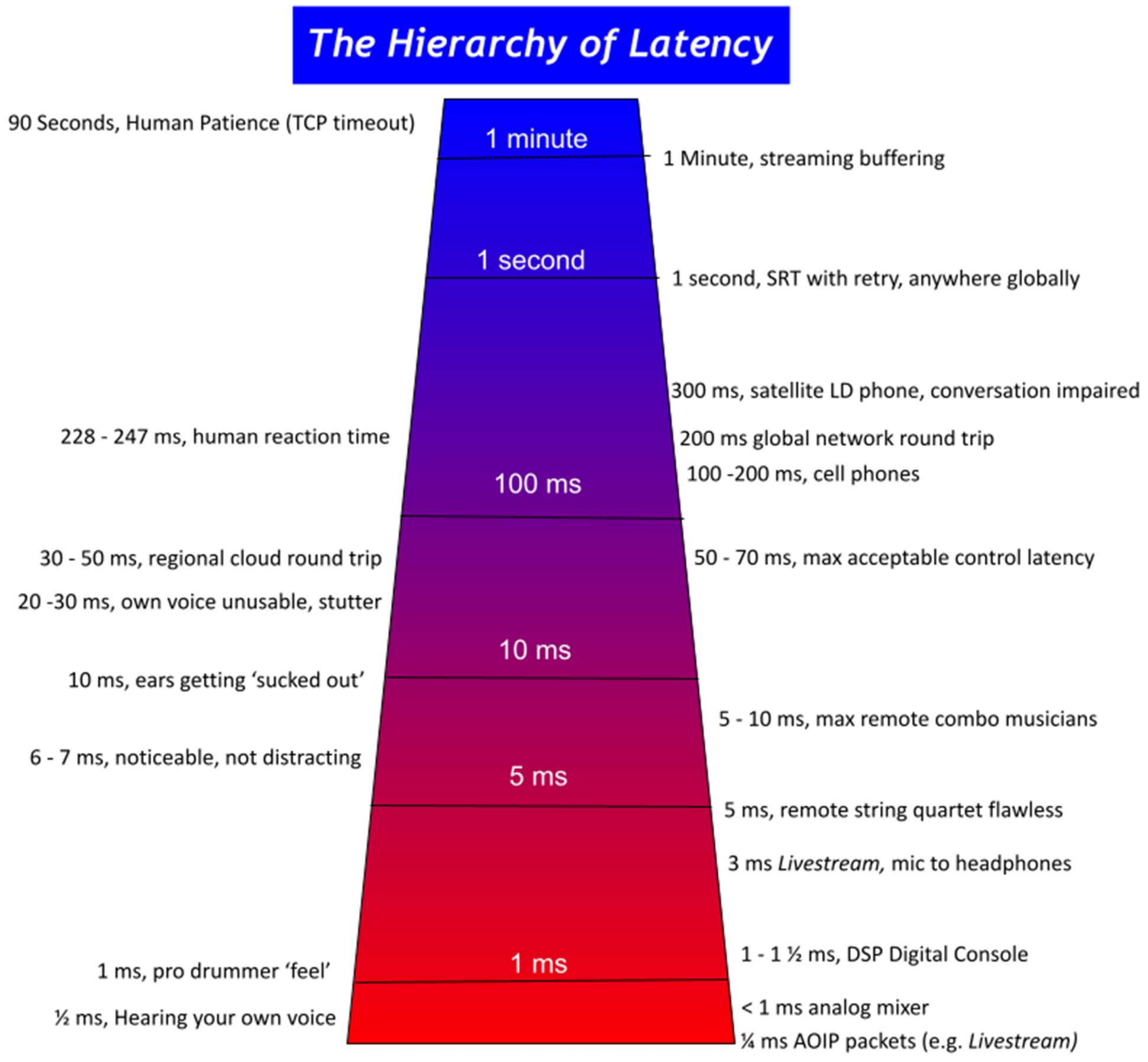
The Hierarchy of Latency
90 Seconds, Human Patience (TCP timeout)
1 minute
1 Minute, streaming buffering
1 second
1 second, SRT with retry, anywhere globally
300 ms, satellite LD phone, conversation impaired
228 - 247 ms, human reaction time
200 ms global network round trip
100 -200 ms, cell phones
100 ms
30 - 50 ms, regional cloud round trip
50 - 70 ms, max acceptable control latency
20 -30 ms, own voice unusable, stutter
10 ms
10 ms, ears getting 'sucked out'
5 - 10 ms, max remote combo musicians
6 - 7 ms, noticeable, not distracting
5 ms
5 ms, remote string quartet flawless
3 ms *Livestream,* mic to headphones
1 - 1 ½ ms, DSP Digital Console
1 ms, pro drummer 'feel'
1 ms
< 1 ms analog mixer
½ ms, Hearing your own voice
¼ ms AOIP packets (e.g. *Livestream)*

FIGURE 2: THE HEIRARCHY OF LATENCY

References

[1] https://en.wikipedia.org/wiki/Group_delay_and_phase_delay

[2]
https://web.archive.org/web/20190302100958id_/http://pdfs.semanticscholar.org/b3c0/a892a982ebde91f83f228905dac30186f827.pdf

[3] https://www.analog.com/en/resources/glossary/transient-intermodulation-distortion-tim.html

[4] https://en.wikipedia.org/wiki/Latency_(audio)

[5] https://tech.ebu.ch/publications/presentations/network-technology-seminar-2012/what-is-vilor

[6] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4456887

[7] https://en.wikipedia.org/wiki/Plesiochronous_system