GCCG working group Readout to the VSF Fall Meeting – September 22nd, 2021

Goal of the group:

- Document common methods and practices for transferring A/V essence
- From Ground to Cloud, Within-a-Cloud, and Cloud-to-Ground

What are the "operating points" we see (G-C,C-G) (for TV)

- Uncompressed (2022-6 or 2110-20) (already defined)
- **"PremiumCompressed" – super low latency, super high quality**
  - **JXS for (1080i @ 200m?) (UHD @ 1000-1500M)**
  - **Dual-path reliability model for super-good reliability @ *minimum latency***
- J2K-ULL (stripes) @200
- J2K (full-frame) @120
- AVCI @ 100
- **Interactive Latency ~20Mbit(HD) (ULL restricted VBV buffer) $$**
  - **ARQ/RIST or dual-path model or FEC? (Typical 4:2:2/10 profile)**
- **Rate Optimized (longer latency @ lower rate) (HD@3-10 Mbits, more delay)**
  - **ARQ/RIST or dual-path model or FEC? (Typical 4:2:0/8 profile)**
- Zoom / GoToMeeting / Webex – whatever gets a picture on the screen
  - Internet best-effort reliability work

Focus on these

Premium Compressed (JPEG XS)

VSF TR-08: codec & LAN 2110-22 base
VSF TR-09: WAN 2110-x extension
*** reference TR-08**
*** opt: with 2022-7**
*** opt: with FEC (small intl, 1D)**
*** opt: GRE Tunnel (ref RIST)**
*** opt: encryption (ref RIST)**

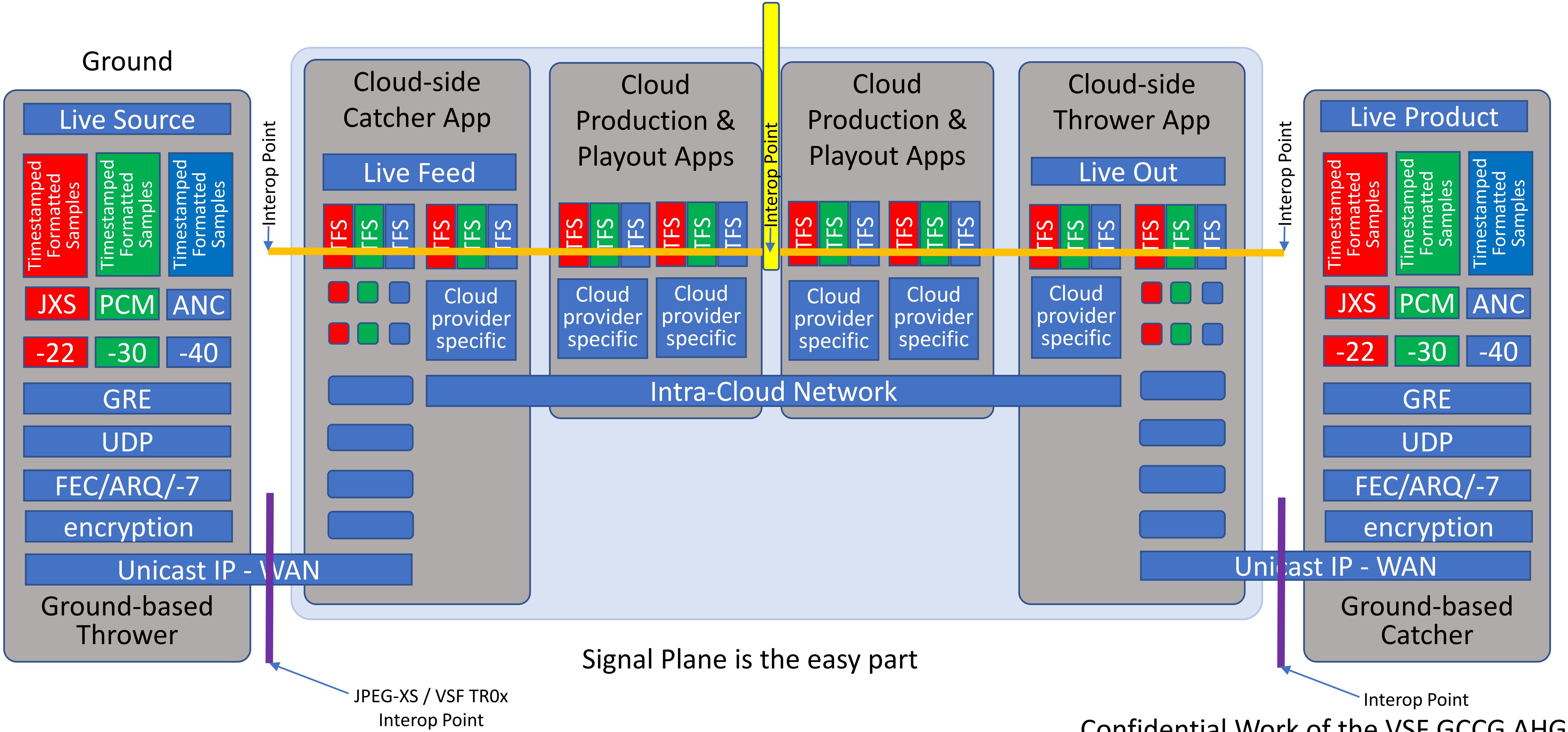
Interactive Latency (small GOP or Prog Refresh)

- **(HD) H264 (constrained VBV, 4:2:2, 10bit)**
- **(UHD) H265 (constrained for latency)**
- **2022-2(TS-RTP)**
- **RIST-FEC (+/- multi-path)**

Bandwidth Optimized

- **(HD) H264 (420/8 standard vbv)**
- **(UHD) H265 ()**
- **2022-2(TS-RTP)**
- **ARQ**
- **Opt: encryption (ref RIST)**

JXS/-22 covers the ultra-low-latency, ultra-high-quality G-C and C-G cases "Premium Compressed"
But what about the C-C case? How to send things between applications within a cloud?



Signal Plane is the easy part

Work for this and adjacent groups

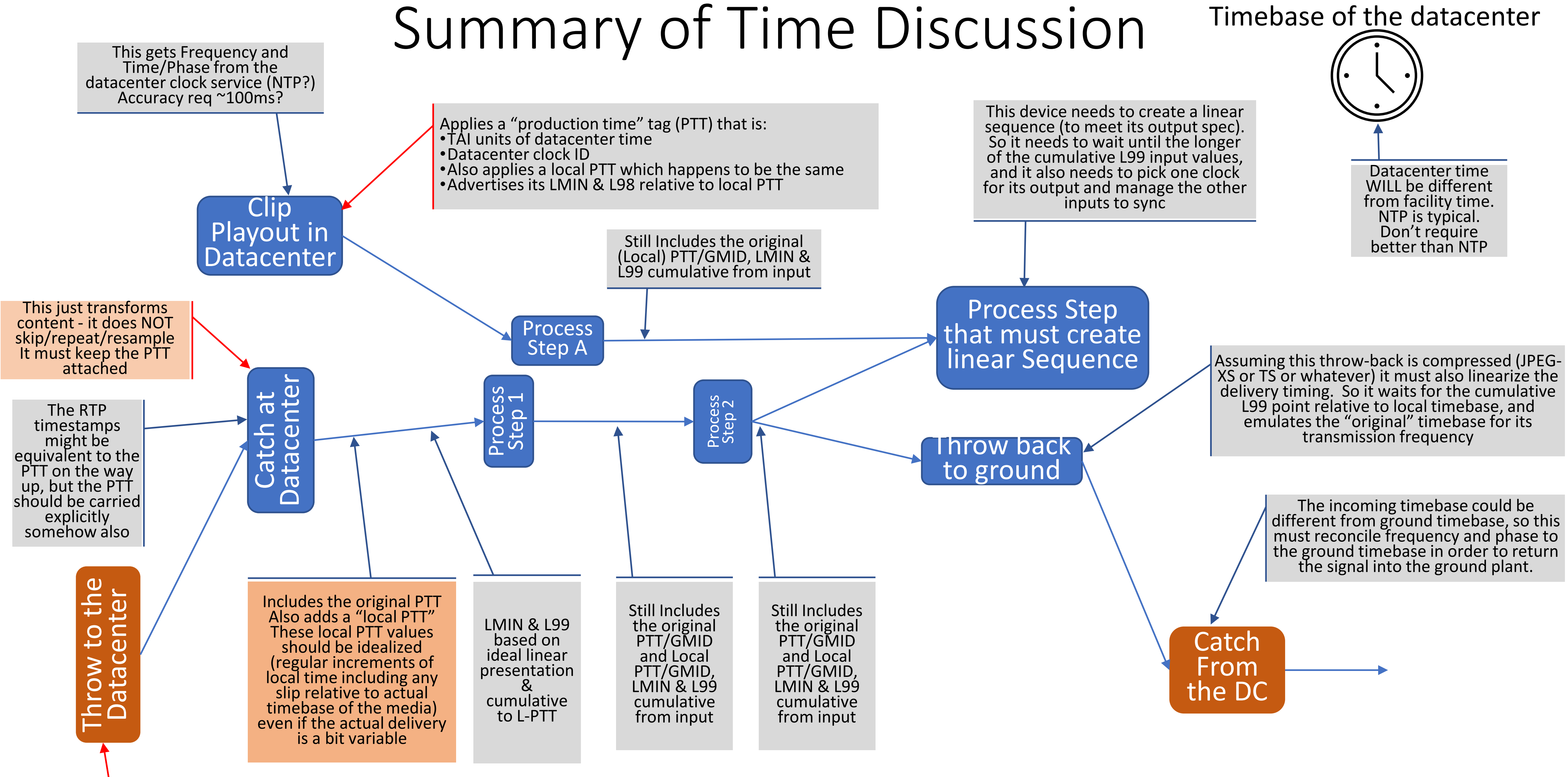
- Define Requirements for (Premium Compressed) TR-08 & TR-09 for G-C and C-G use cases
  - TR-07 & 08 are done and published. Defines JPEG-XS over TS and over 2110-22 with interop points and capability sets
  - TR-09 in progress. RIST elements in the data plane, plus Control plane
  - TR-07 & 08 include a list of “interop points” including some non-tv cases from IPMX

- Is there something more to document about the TS/IP/H.264 case? Or is it good enough already?
  - The standards for the parts are ok, just a list of more details of the operating points within them might be handy.
  - Should we document the supported “interop points” ? (maybe overcome by reality)
  - Is there something to write down? Probably yes, but its mostly pointers to other things and constraints.
  - Can the new data-plane work in TR-09 (ARQ, FEC, -7) be applied to this class of streams ? (yes)
  - How about the control-plane work above? YES, hopefully, but what is the NMOS-equivalent for TS ?

- Time-Flow / Time-Transport / Time-Tagging model
  - How do we treat time in the concatenated virtualized systems
  - How do we integrate “real-time” on the ground with “floating time” in the cloud?
  - This requires a vocabulary and some modeling/specification effort
  - The use cases include how to catch up / manage drift / manage change / re-integration to timeline

- The Media Containers / Object Format structures for hand-off from application to application
  - The github part of AWS CDI is a step in this direction
  - And also how this handoff interacts with latency and latency accumulation

Summary of Time Discussion



Timebase of the datacenter

Datacenter time WILL be different from facility time. NTP is typical. Don't require better than NTP

This gets Frequency and Time/Phase from the datacenter clock service (NTP?) Accuracy req ~100ms?

Clip Playout in Datacenter

Applies a "production time" tag (PTT) that is:
- TAI units of datacenter time
- Datacenter clock ID
- Also applies a local PTT which happens to be the same
- Advertises its LMIN & L98 relative to local PTT

Process Step A

Still Includes the original (Local) PTT/GMID, LMIN & L99 cumulative from input

This device needs to create a linear sequence (to meet its output spec). So it needs to wait until the longer of the cumulative L99 input values, and it also needs to pick one clock for its output and manage the other inputs to sync

Process Step that must create linear Sequence

This just transforms content - it does NOT skip/repeat/resample It must keep the PTT attached

The RTP timestamps might be equivalent to the PTT on the way up, but the PTT should be carried explicitly somehow also

Catch at Datacenter

Process Step 1

Process Step 2

Throw back to ground

Assuming this throw-back is compressed (JPEG-XS or TS or whatever) it must also linearize the delivery timing. So it waits for the cumulative L99 point relative to local timebase, and emulates the "original" timebase for its transmission frequency

The incoming timebase could be different from ground timebase, so this must reconcile frequency and phase to the ground timebase in order to return the signal into the ground plant.

Catch From the DC

Throw to the Datacenter

Includes the original PTT Also adds a "local PTT" These local PTT values should be idealized (regular increments of local time including any slip relative to actual timebase of the media) even if the actual delivery is a bit variable

LMIN & L99 based on ideal linear presentation & cumulative to L-PTT

Still Includes the original PTT/GMID and Local PTT/GMID, LMIN & L99 cumulative from input

Still Includes the original PTT/GMID and Local PTT/GMID, LMIN & L99 cumulative from input

Applies a "production time" tag (PTT) that is:
- TAI units of facility time
- Facility Clock GMID

Summary slide. GCCG Timing Architecture
Confidential Work of the VSF GCCG AHG

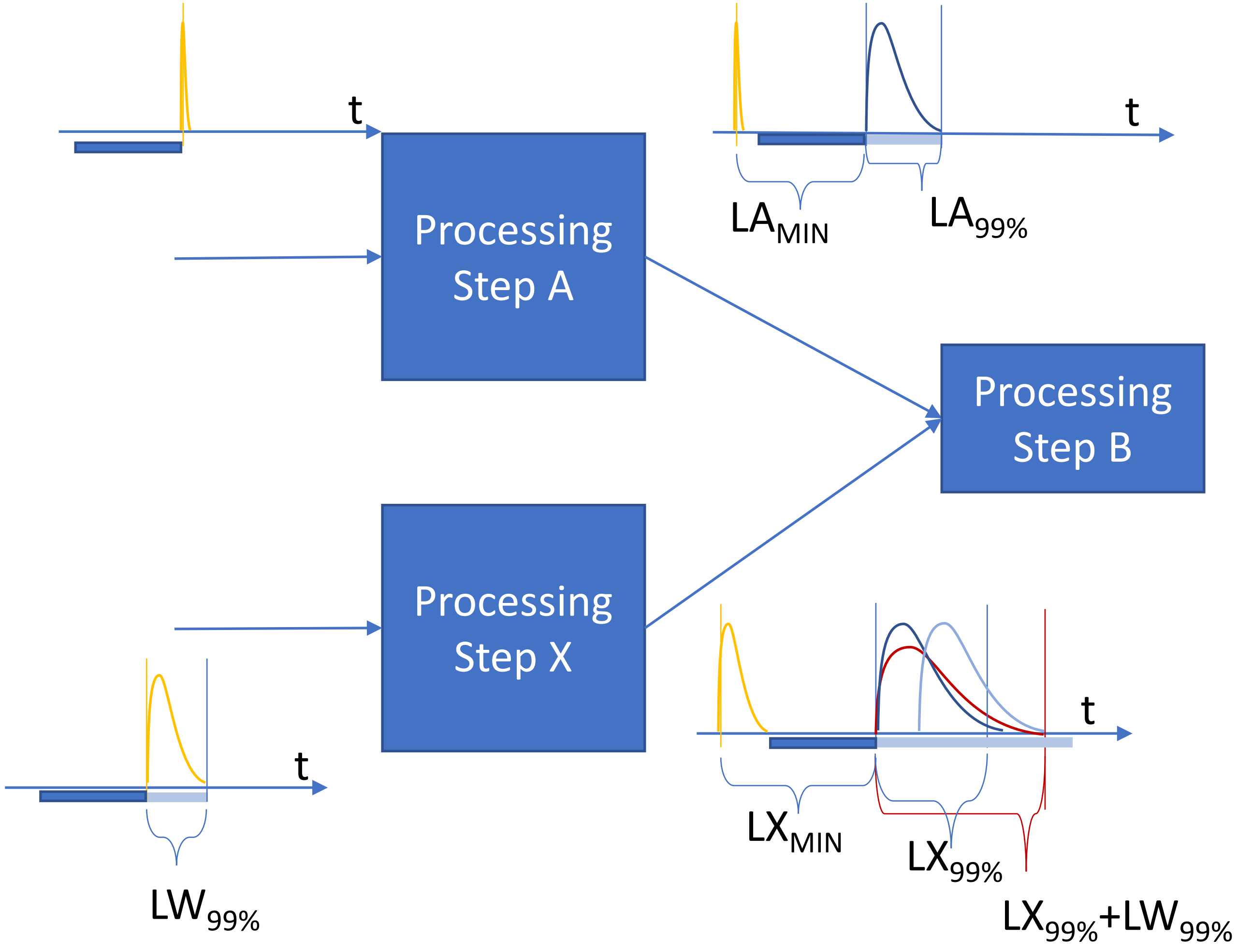
The "time variability / time floating" problem

- Allow variability in the timing of handoffs in software, but still with an ability to predict the outcome
- Some processes must generate a consistent output – encoders, mixers, etc.
- Must bound the input buffering (latency) yet accommodate the variability
- What are the sources of delivery (arrival) variability?
  - Short-term variability (upstream process steps with variable latency)

- Vocabulary (about each process step in the datacenter)
  - TX Variability – bound on how early/late the signal may egress, variance above baseline
  - TX Planning Delay – the baseline in the above statement – the smallest value of delay
  - Actual delay is planning delay plus some amount of variability

What is the “spec” of latency and variability?

- Push model (not backpressured)
- Based on (uniform?) content “chunks”
  - might be frames, fields, stripes
  - might be blocks of audio samples
- L_{MIN} = the soonest/shortest amount of time from input to output
  - Input time = buffer 100% arrived to me
  - Output time = buffer left me 100%
  - Includes the egress transit time
- $L_{99\%}$ = the amount of variability *beyond* the L_{MIN} for 99th percentile case
- What about continuity? Do processing steps need to maintain cadence if input not there?
  - Maybe not – only if it “has to” for its own processing purposes. Otherwise best to just be late or missing and let downstream do the best it can with what it gets when it gets it.



Variability on the input accumulates into the output!!!

What is the “contract” at the input of a (in-cloud) receiver ?

- Data content spec (e.g. 4:2:2/10 SDR 709 1080p 50Hz whole-frames)
  - Setup time: everything
  - Runtime variable: (HDR format?)

Mix of setup-time and in-band

- Data format spec (frames or stripes, and how formatted in object?)
  - Declare at setup time

Application convention, *setup time*

- L_{MIN}, L_{99} parameters of each incoming signal
  - These are relative to the local PTT value
  - These may include any L_{MIN}, L_{99} of the upstream signals, depending on the sender

C-C: What is the right way to organize the data?

- Packed like 2110 PGROUPS ?   (easy to say, painful to do)

- What is actually Software-Friendly ?
  - 10-bit values mapped into 16i? 8-bit values plus a packed trailer with more?
  - Planer or Interleaved? Stripes or whole frames (or both)?
  - What about metadata (static or dynamic) ?
  - What about audio and other stuff ?

- What does “Connection Management” Look Like?
  - Are connections only made at application startup, or are they dynamic?
  - Can there be a recommended interface for connection management?

- We will probably create a TR that documents a C-C method including data structures and CM interface

GCCG: Whats the Path to Completion?

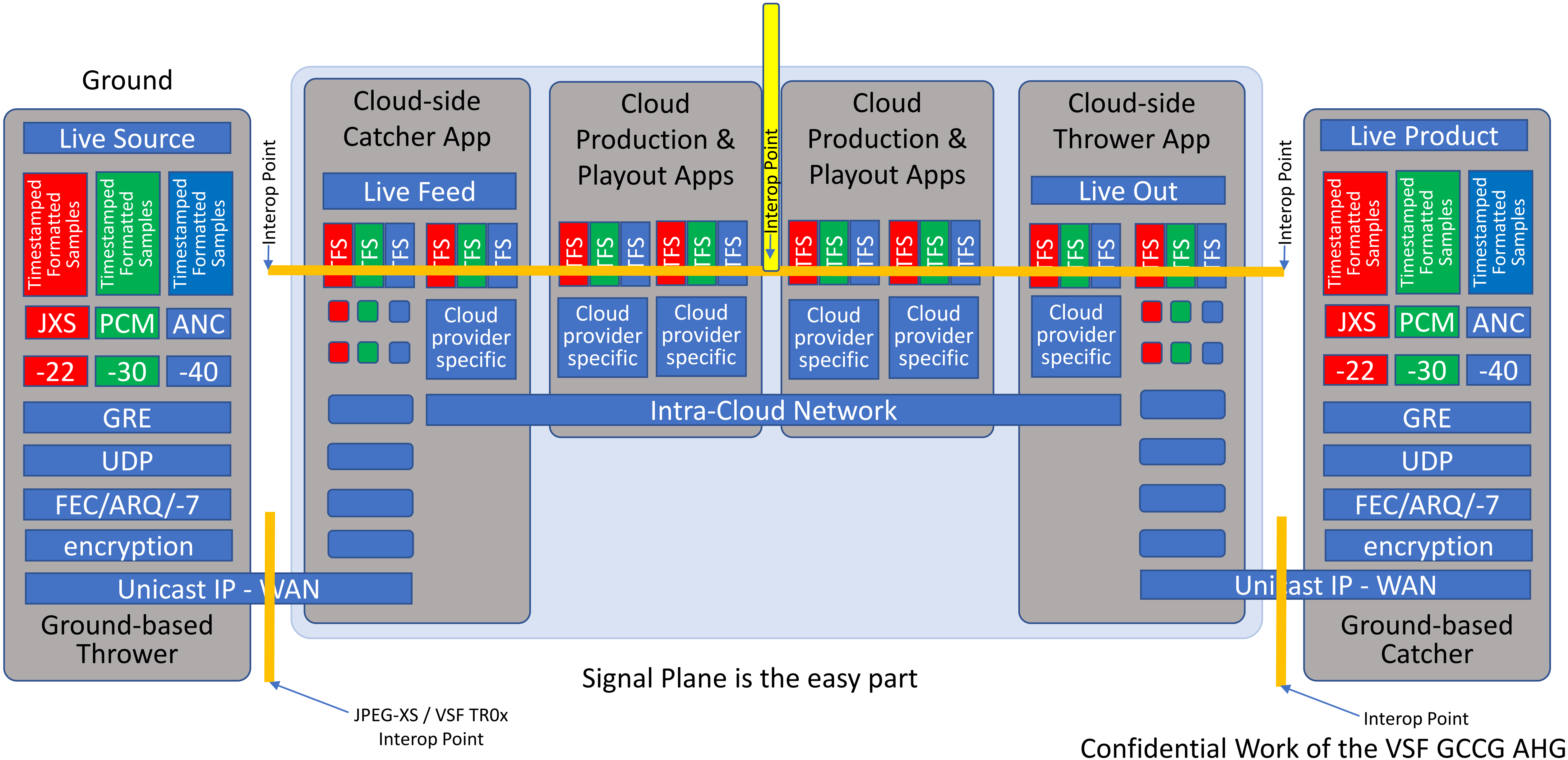
- Buffer formatting scheme(s) for video handoffs
  - 8’s plus trailer scheme seems nice and handy for the buffer-transfer use case
  - Alignment points (ensure planer start points are aligned)
  - Stripe size if not whole frame/field (balance transaction overhead –vs- latency)
  - Include an optional traincar(s) above/below (without messing up the alignment) for audio or metadata or both

- What about audio and other stuff as separate essences
  - Memory transfer is not worth bothering, just use tcp
  - define a structure inside that (maybe common with the traincar block above)

- Path to publication of the C-C data format spec
  - VSF TR would be a reasonable vehicle for the C-C spec

- Connection Management Interface – what is common, what is specific?

Interoperability Points: G-C, C-C, C-G



Signal Plane is the easy part

Confidential Work of the VSF GCCG AHG