05/05/2025

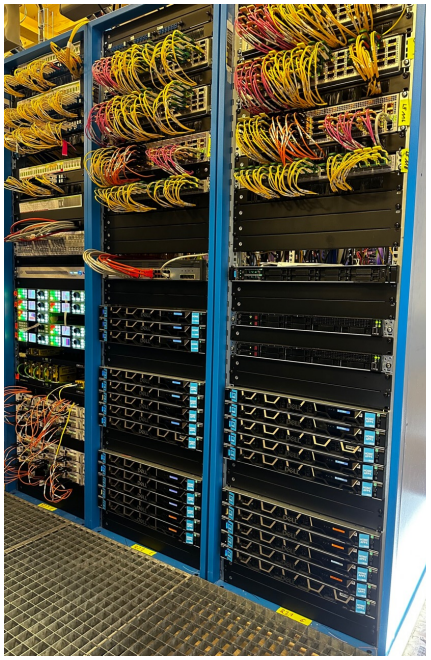
"An Open Approach to a Media Exchange Layer"

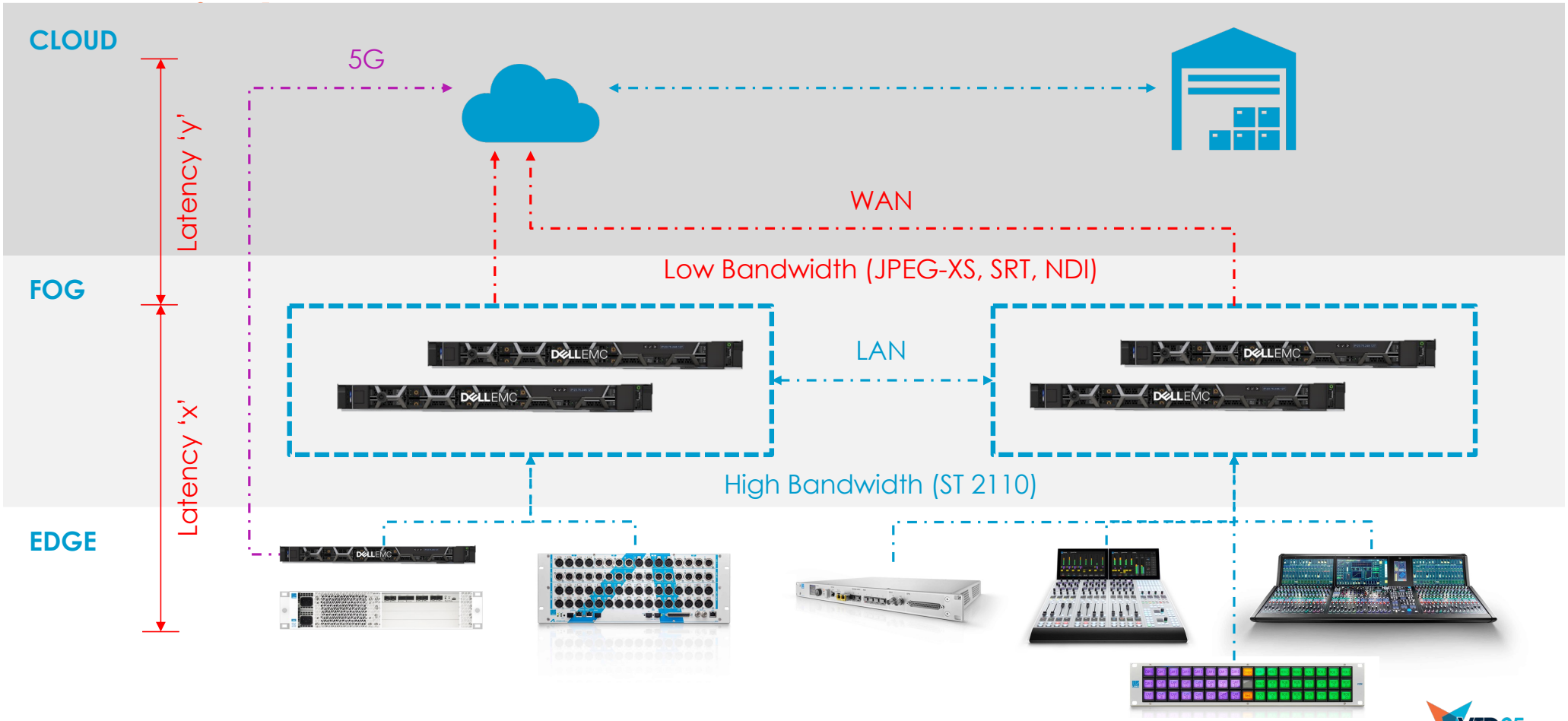
Phillip Myers

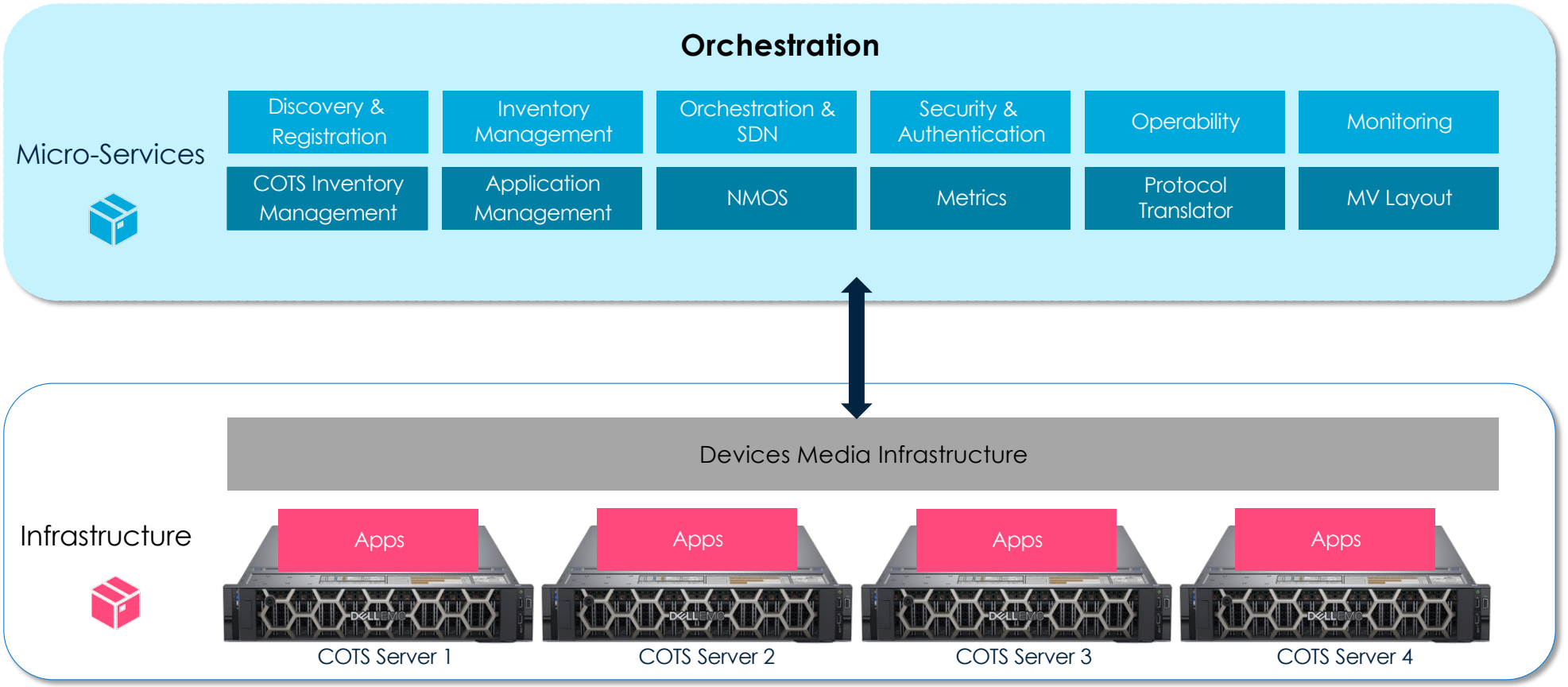
LAWO

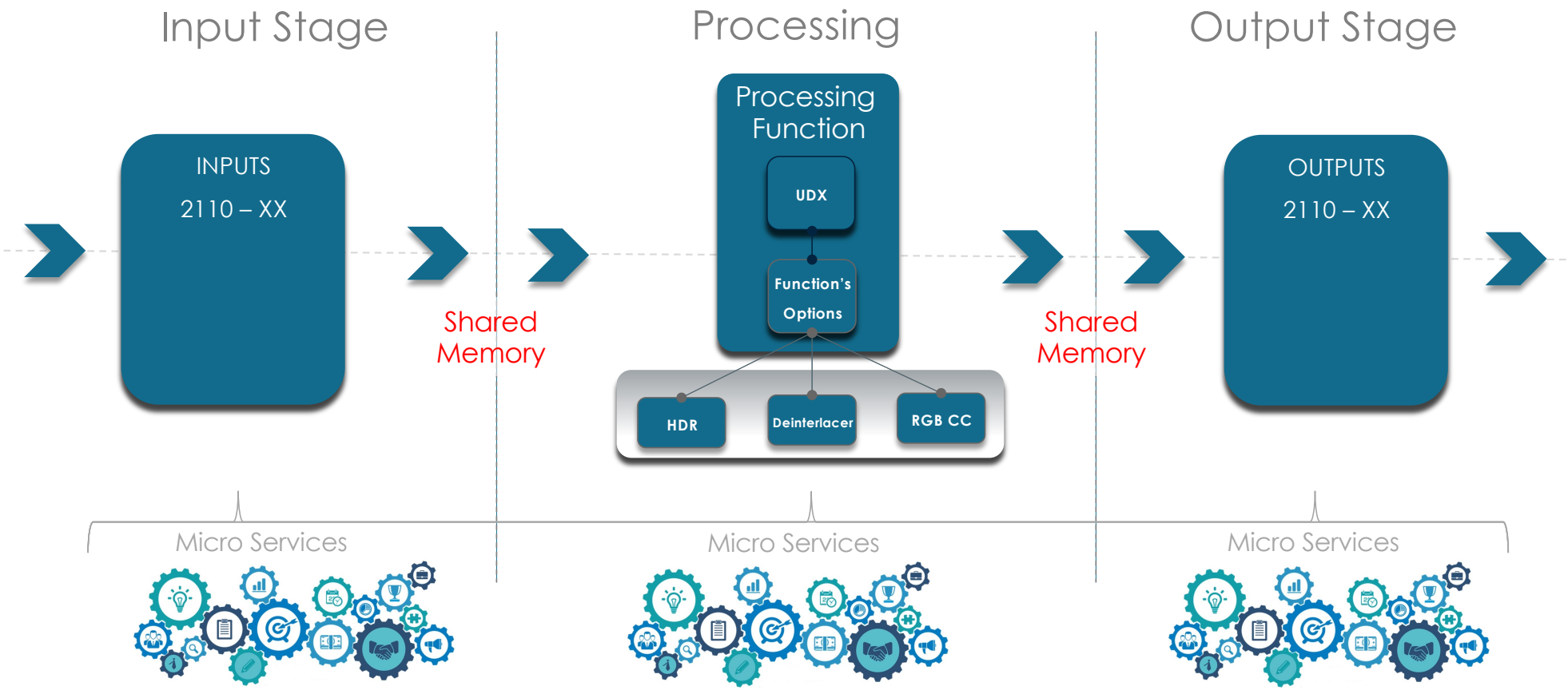
Recap since VidTrans24

- Software defined production tools using standard IT (CPU/GPU) and IP technologies are already here and used today by many customers
- Tier-1 Live Sports Production
- Global Sporting Events
- Major Broadcast Infrastructures
- However, with these new tools, new challenges arise !









Container Image Versioning

(Registry)

Resource

Scheduling

App Output Stage

(SMPTE ST 2110)

```
lawo@candy:~$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS     NAMES
a61482c84fee   10.1.215.66:5200/st2110:v1.1.0         "taskset -c 11,75 /a…"   5 minutes ago    Up 5 minutes              udx-hdr-001-send2110-audio-4
063776d763e6   10.1.215.66:5200/st2110:v1.1.0         "taskset -c 11,75 /a…"   5 minutes ago    Up 5 minutes              udx-hdr-001-send2110-audio-3
f75a2aa6f04c   10.1.215.66:5200/st2110:v1.1.0         "taskset -c 11,75 /a…"   5 minutes ago    Up 5 minutes              udx-hdr-001-send2110-audio-2
12690935270f   10.1.215.66:5200/st2110:v1.1.0         "taskset -c 11,75 /a…"   5 minutes ago    Up 5 minutes              udx-hdr-001-send2110-audio-1
615217c3be93   10.1.215.66:5200/st2110:v1.1.0         "taskset -c 10,74 /a…"   5 minutes ago    Up 5 minutes              udx-hdr-001-send2110-1
487fffbf6a09   10.1.215.66:5200/st2110:v1.1.0         "/app/receive2110 au…"   5 minutes ago    Up 5 minutes              udx-hdr-001-receive2110-audio-4
491f6d38b495   10.1.215.66:5200/st2110:v1.1.0         "/app/receive2110 au…"   5 minutes ago    Up 5 minutes              udx-hdr-001-receive2110-audio-3
5cdf5e205fc0   10.1.215.66:5200/st2110:v1.1.0         "/app/receive2110 au…"   5 minutes ago    Up 5 minutes              udx-hdr-001-receive2110-audio-2
a50a2270db9b   10.1.215.66:5200/st2110:v1.1.0         "/app/receive2110 au…"   5 minutes ago    Up 5 minutes              udx-hdr-001-receive2110-audio-1
74302aa76942   10.1.215.66:5200/st2110:v1.1.0         "/app/receive2110 -p…"   5 minutes ago    Up 5 minutes              udx-hdr-001-receive2110-1
ea5cab80f7de   10.1.215.66:5200/truffle:v1.1.0        "/app/ppmodifier -po…"   5 minutes ago    Up 5 minutes              udx-hdr-001-ppmodifier-1
361f4f8c7d91   10.1.215.66:5200/deinterlacer:v1.1.0   "/app/ppdeint -port …"   5 minutes ago    Up 5 minutes              udx-hdr-001-ppdeint-1
6e0e942aa7f7   10.1.215.66:5200/ppserver:v1.1.0       "/app/ppserver -widt…"   5 minutes ago    Up 5 minutes              udx-hdr-001-ppserver-1
ca8c776f627b   10.1.215.66:5200/kero-warden:1.2.2401c11   "/app/warden --prom_…"   3 days ago       Up 9 hours                home_apps_warden
36617b0032ee   10.1.215.66:5200/kero-bundon:1.2.2401c11   "/app/bundon -e 10.1…"   3 days ago       Up 9 hours                home_apps_bundon
b73e62ae9281   10.1.215.66:5200/kero-tick:1.2.2401c11     "/app/tick ptp -prom…"   3 days ago       Up 9 hours                home_apps_tick
lawo@candy:~$
```

Core Services

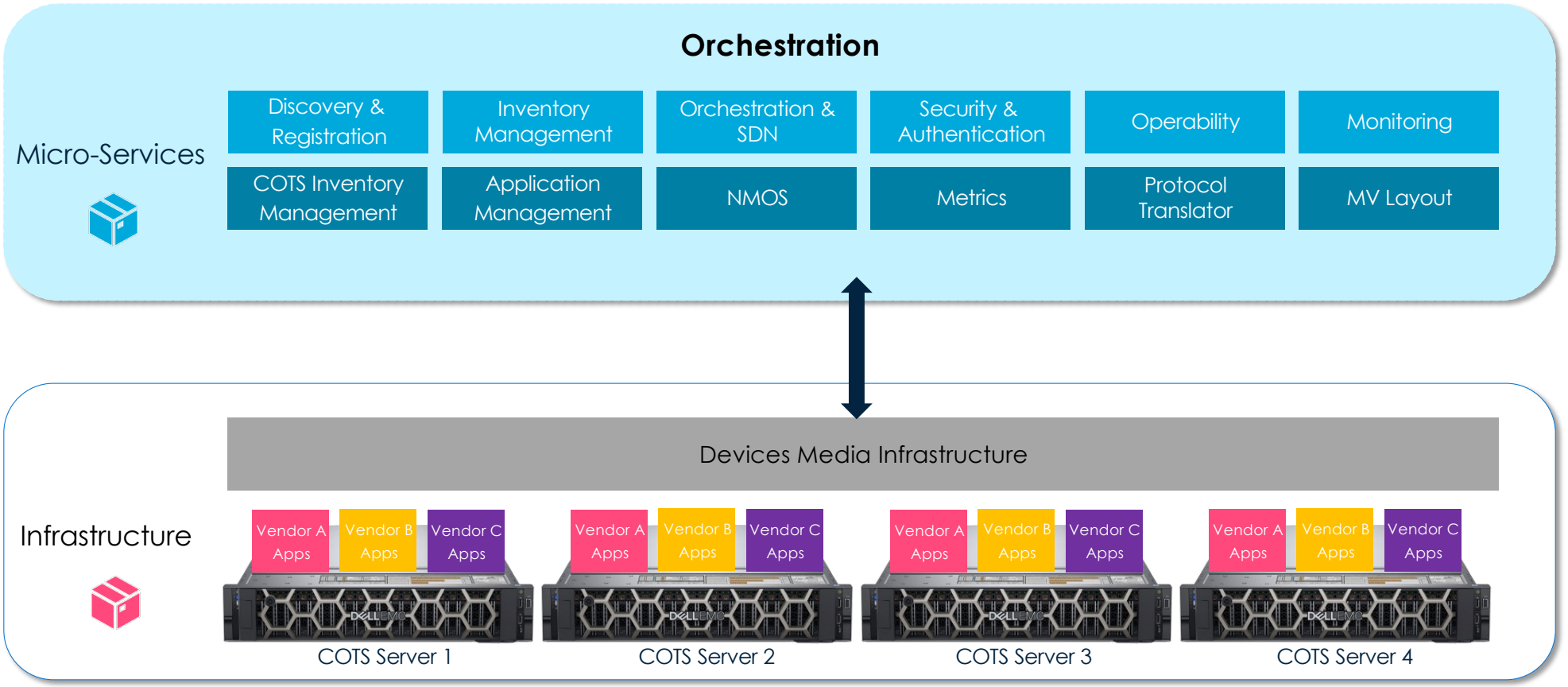
(Orchestration, Timing, Monitoring)

App Input Stage

(SMPTE ST 2110)

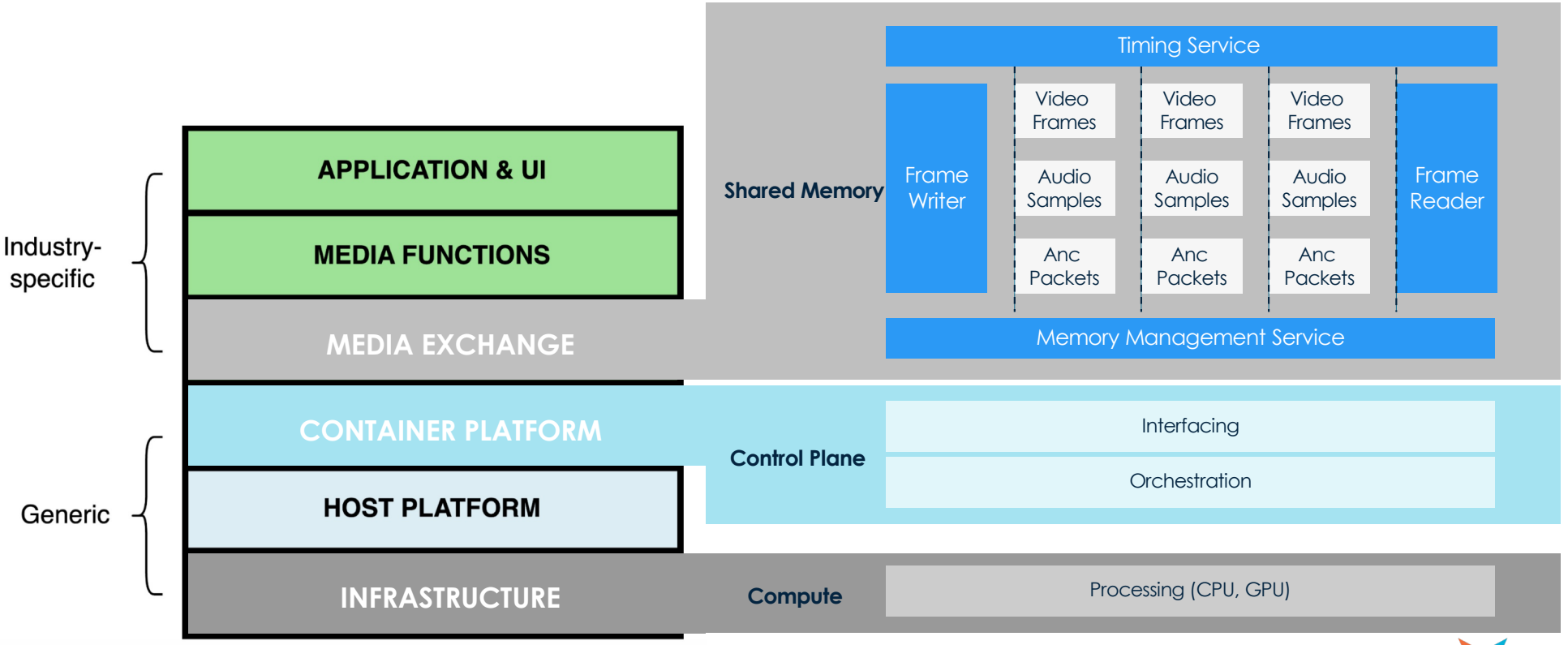
App Processing

(AV store, Deinterlacer)



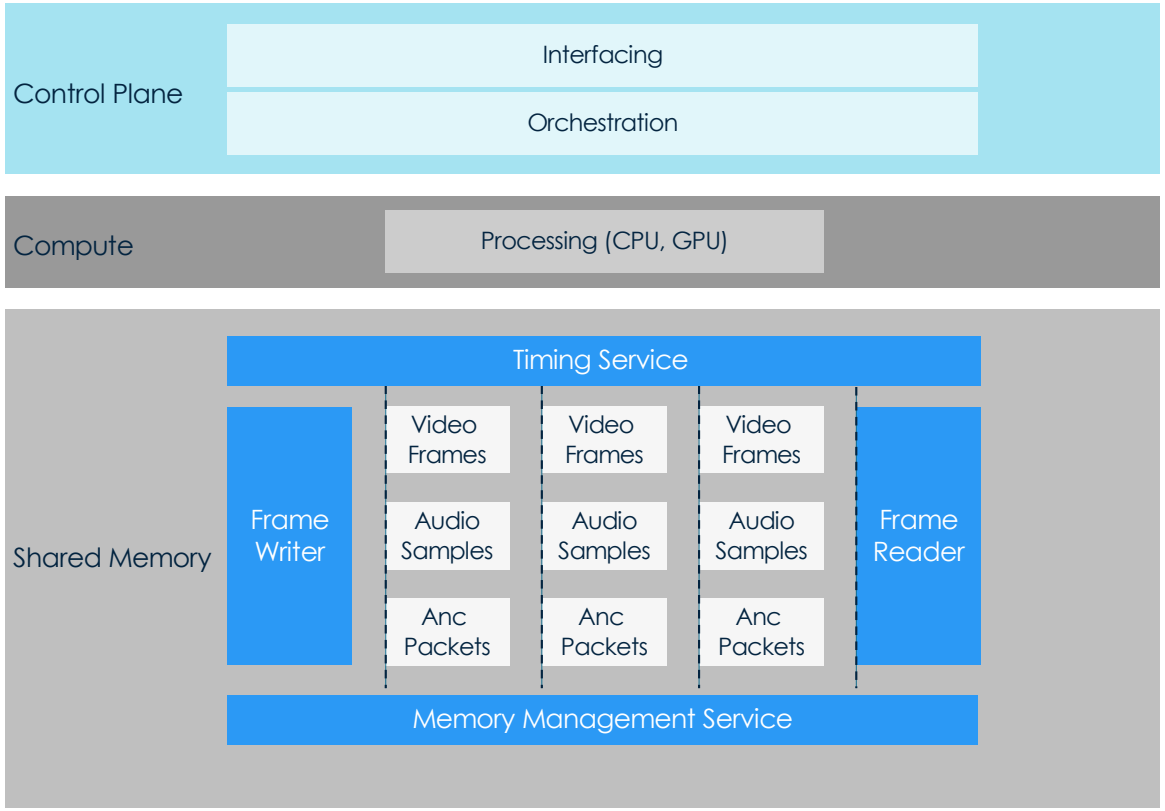
The industry challenge !

Layering – the building blocks



* EBU – The Dynamic Media Facility Reference Architecture

Layering – the building blocks

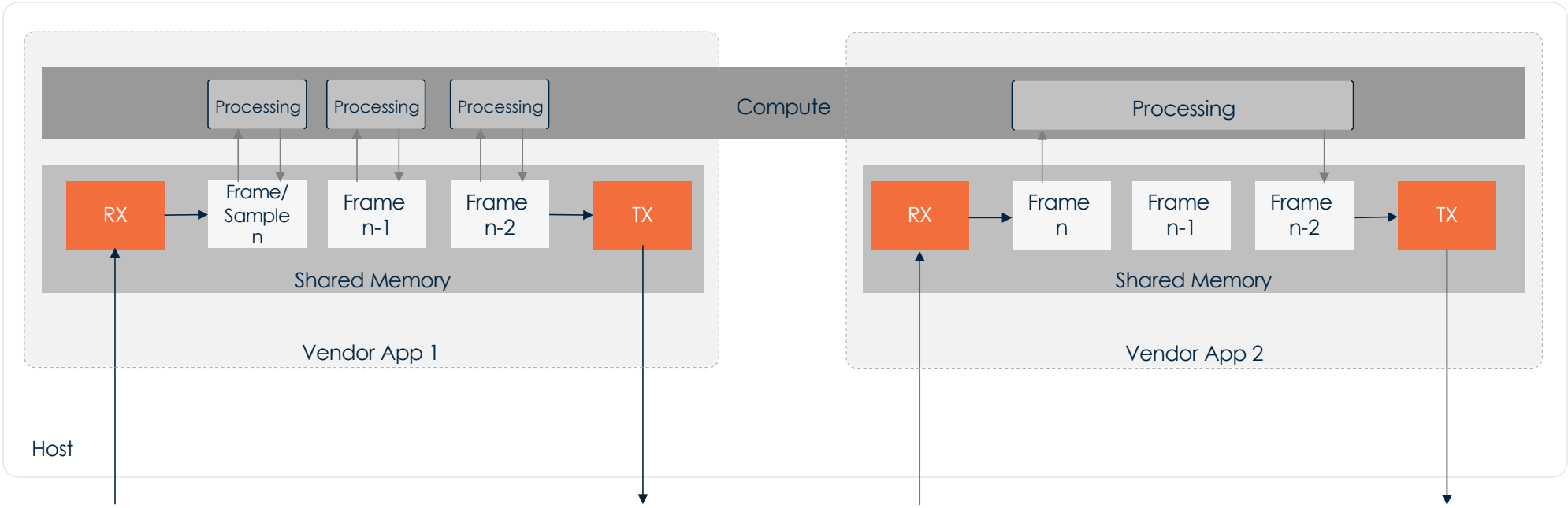


Plane running on every compute instance, providing point of contact. Communicates live control requests to planes and services below and reports back logging, channel status and overall server health

This Plane performs any required video, audio or ancillary processing

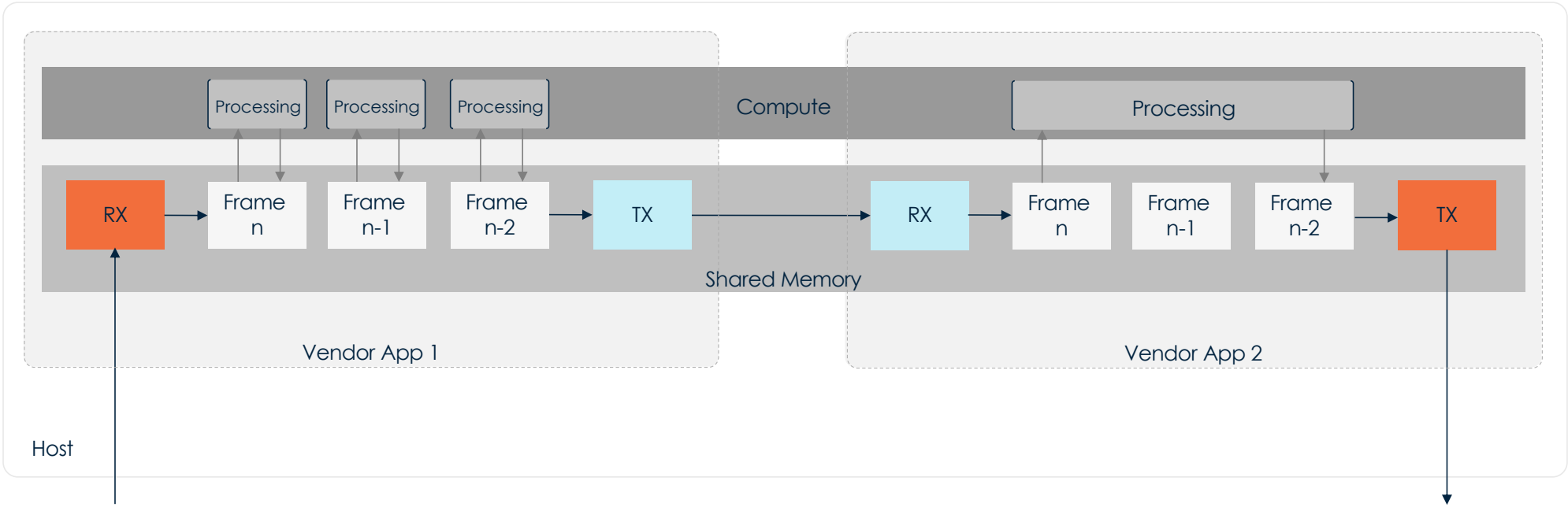
This Plane hosts various services responsible for depacketizing and writing receiver video, audio and ancillary data, quantising it into video frames for processing, before reading processed data from memory and packetizing it into senders.

App with local shared memory



ST2110, NDI, SRT type RX/TX

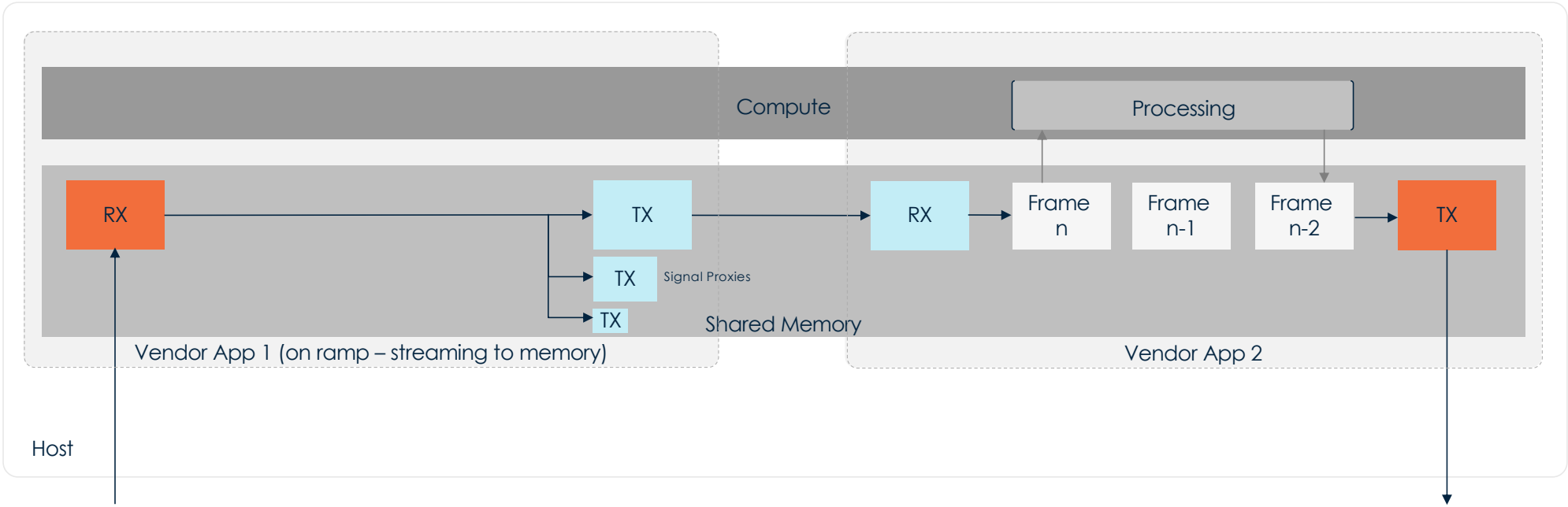
Shared memory across apps (same host)



ST2110, NDI, SRT type RX/TX

Memory Reader/Writer

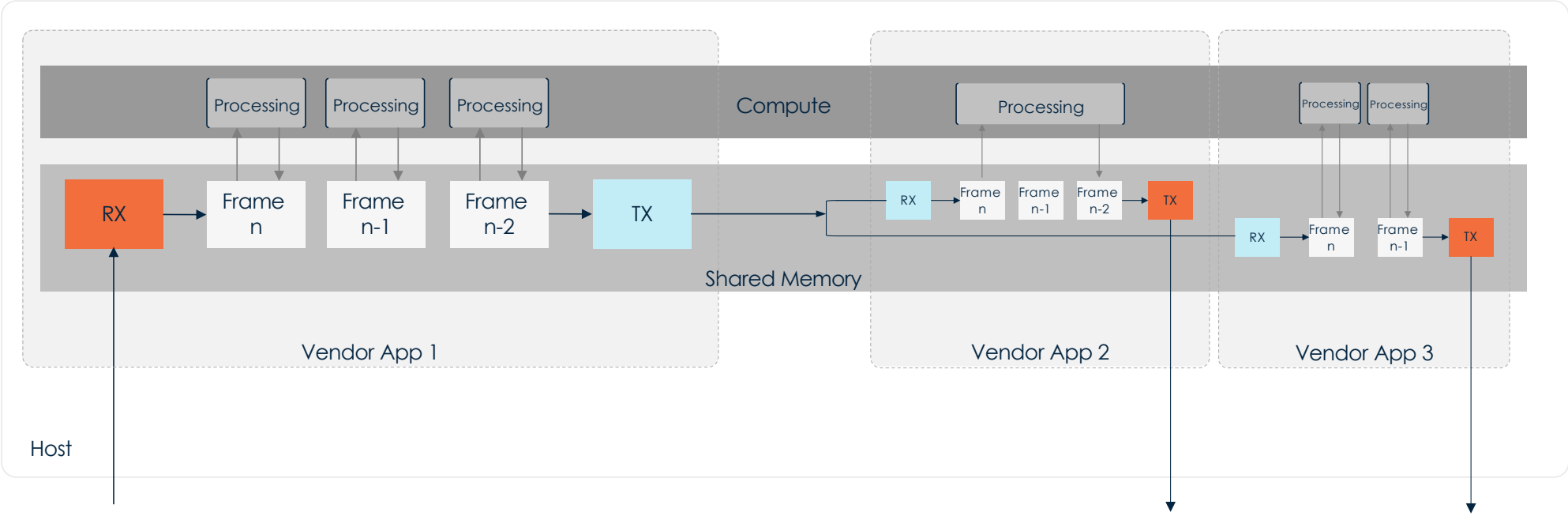
Stream Transcoder On / Off into memory (same host)



ST2110, NDI, SRT type RX/TX

Memory Reader/Writer

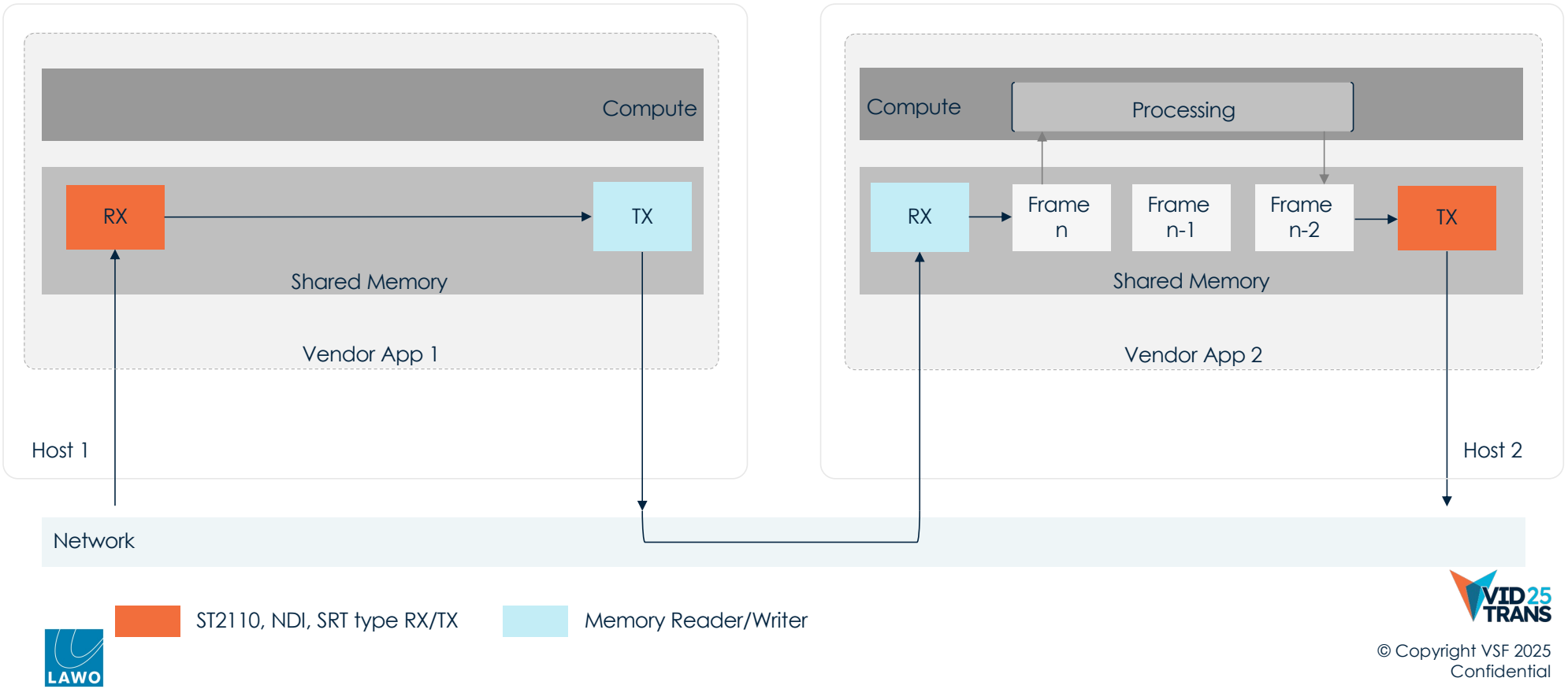
Shared memory across multiple apps (same host)



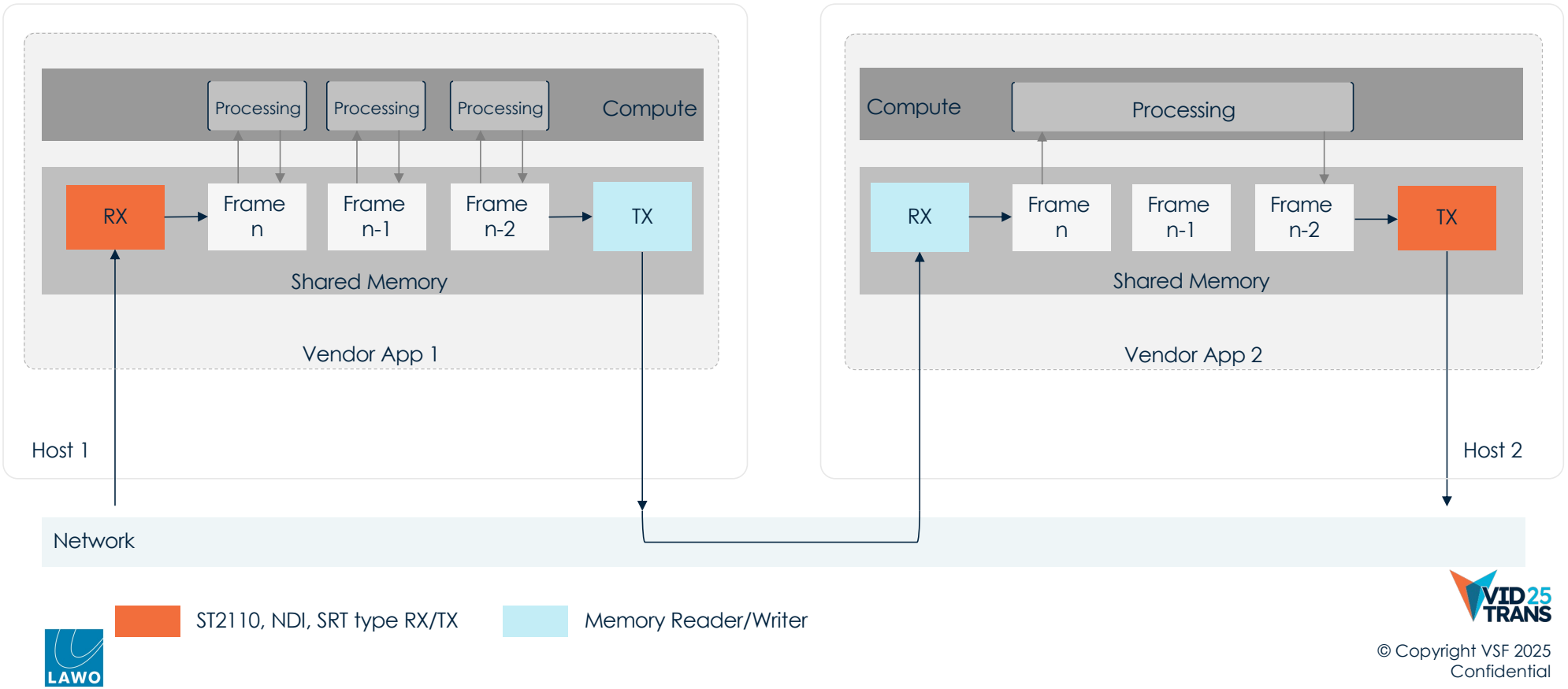
ST2110, NDI, SRT type RX/TX

Memory Reader/Writer

Shared memory across multiple compute nodes

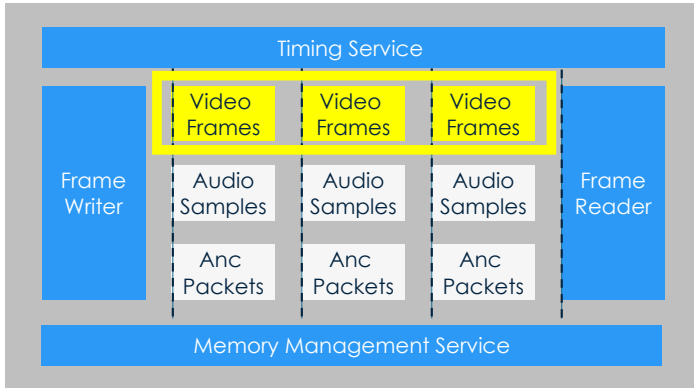


Shared memory across multiple compute nodes



Media Exchange Layer (Common Memory Sharing)

- **Intelligent memory store, self describing media (metadata)**
- **Video (frame) store:**
  - Packing format - YUV422P
  - 16-bit 4:2:2 with separate Y, U/Cb and V/Cr planes
  - No interleaving of Y and C, or Cb and Cr, fully planar
  - Initially 4:2:2 but easy to extend to 4:4:4 (if required) and add Alpha support
  - Arrangement in memory:
    - Frame 0 - Y plane, U plane, V plane
    - Frame 1 - Y plane, U plane, V plane, etc…



Media Exchange Layer (Common Memory Sharing)

- **Metadata is stored per frame, and has the following structure:**
  - int32_t Width; // the number of active pixels per line
  - int32_t Height; // the number of active lines
  - int32_t FieldRate; // integer representation, i.e. 59.94 = 59
  - ScanType scan; // field/frame scan type
  - ScanType origScan; // original field/frame scan type (prior to deinterlacing)
  - VidDataFormat Format; // the frame data format (i.e. YUV422P)
  - uint64_t Timestamp; // source timestamp converted to PTP time
  - Rectangle ActiveArea; // the active area of the image within the frame

Media Exchange Layer (Common Memory Sharing)

- **Metadata is stored per frame, 'Scan' type is enumerated as follows:**
  - SCAN_UNDEFINED = 0,
  - SCAN_PROGRESSIVE, // full progressive frame
  - SCAN_INTERLACED_TOP_F1, // interlaced, first field, top lines
  - SCAN_INTERLACED_BOT_F1, // interlaced, first field, bottom lines
  - SCAN_INTERLACED_TOP_F2, // interlaced, second field, top lines
  - SCAN_INTERLACED_BOT_F2, // interlaced, second field, bottom lines
  - SCAN_SEGMENTED, // progressive segmented frame

Media Exchange Layer (Common Memory Sharing)

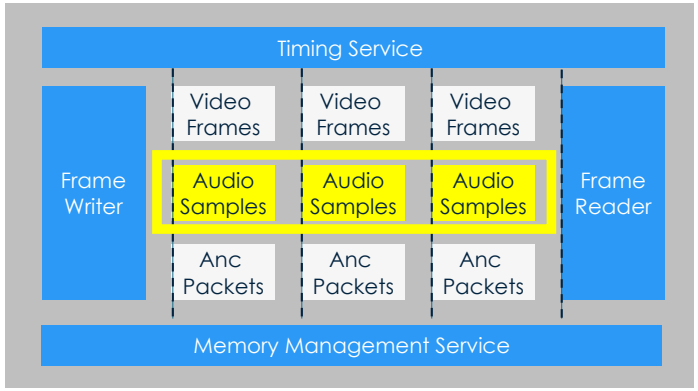
- **Metadata is stored per frame, The 'Active Area' rectangle is defined as:**
  - int32_t Xstart; // first active pixel
  - int32_t Ystart; // first active line
  - int32_t Xend; // pixel after the last active pixel
  - int32_t Yend; // line after the last active line
  - So {0, 0, 1920, 1080} would be a full 1080p active frame

Media Exchange Layer (Common Memory Sharing)

- **Audio store:**
  - 32-bit integer (2's complement) audio samples, MSB aligned
  - Memory layout is a 2D array: channels x samples
  - Samples are addressed in the store using the LSBs of the RTP timestamp (which updates at sample rate)
  - Store could be any size as long as it's a power of 2

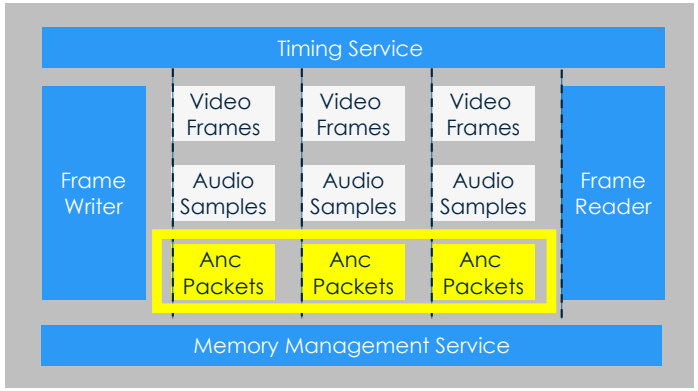
- Different area of the shared memory has the metadata for each channel, this allows the different channels to have different characteristics (sample rate, bit depth etc.)

- For both video and audio stores, you need a mechanism to indicate when the data is stale, or you need the writer to clear the store when the input is lost (Read black frames and silence, for example)



Media Exchange Layer (Common Memory Sharing)

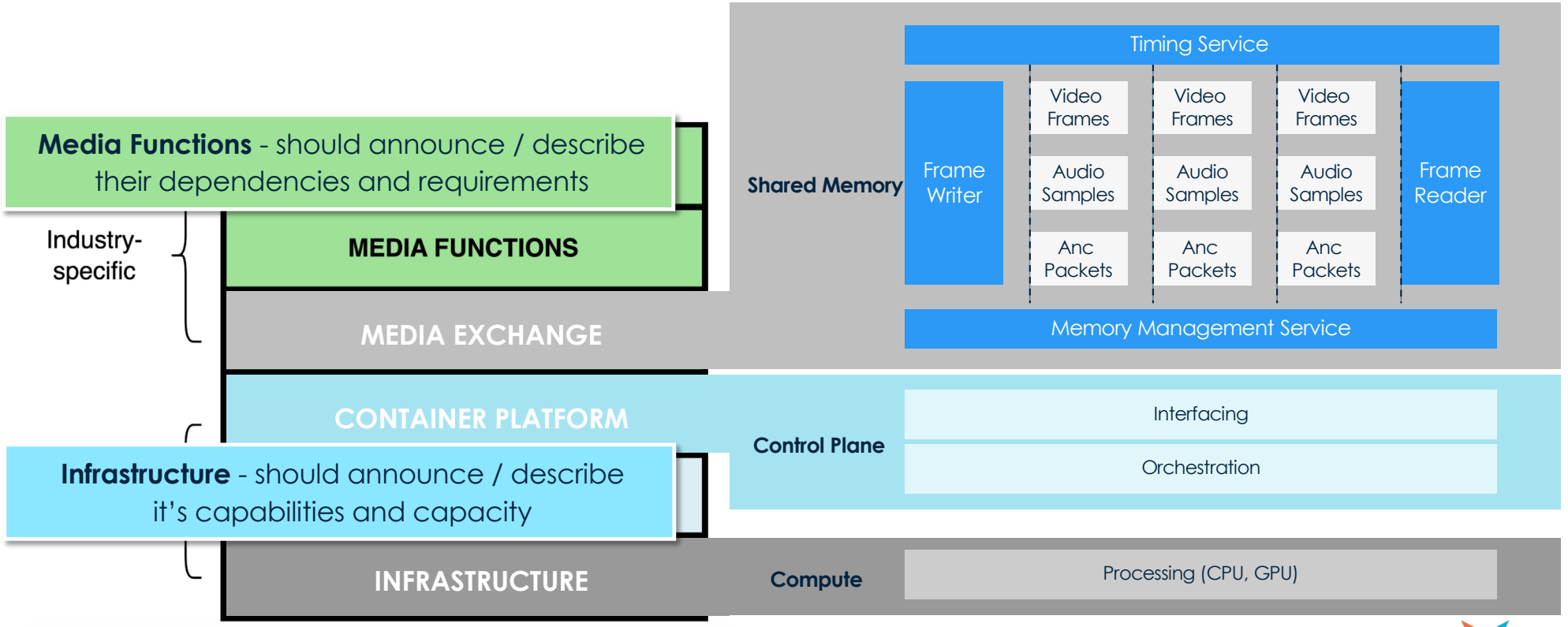
- **Ancillary Data store:**
  - Ancillary data is video frame aligned, so the store is 3-dimensional (streams x frames x packets)
  - Within the store, each packet keeps all its RFC-8331 data:
  - **uint8_t yc**; // C: 1 bit
  - **uint16_t lineNumber**; // t Line_Number: 11 bits
  - **uint16_t horzOffset**; // Horizontal_Offset: 12 bits
  - **uint8_t streamFlag**; // S (Data Stream Flag): 1 bit
  - **uint8_t streamNumber**; // StreamNum: 7 bits



Media Exchange Layer (Common Memory Sharing)

- **Ancillary Data store:**
  - Ancillary data is video frame aligned, so the store is 3-dimensional (streams x frames x packets)
  - Within the store, each packet keeps all its SMPTE ST291-1 data:
  - **uint8_t did**; // data identification
  - **uint8_t sdid_dbn**; // secondary data identification (type 2) or data block number (type 1)
  - **uint8_t dc**; // the count of user data words
  - **uint16_t udw[255]**; // user data, up to 255 10-bit words
  - **uint16_t checksum**; // 9-bit checksum of DID, SDID, DC and UDW

Layering – the building blocks



* EBU – The Dynamic Media Facility Reference Architecture

Understanding the ‘Upper’ layers

- Multi-vendor applications (Live and non-live media functions)
- Orchestration – central container registry, provisioning / scheduling of compute and network
- Roles and Permissions (don’t forget to clean-up – orphan services and garbage)
- Well documented API (open source)
- Security – first class citizen by design (intra service communications and external API’s)

LAWO

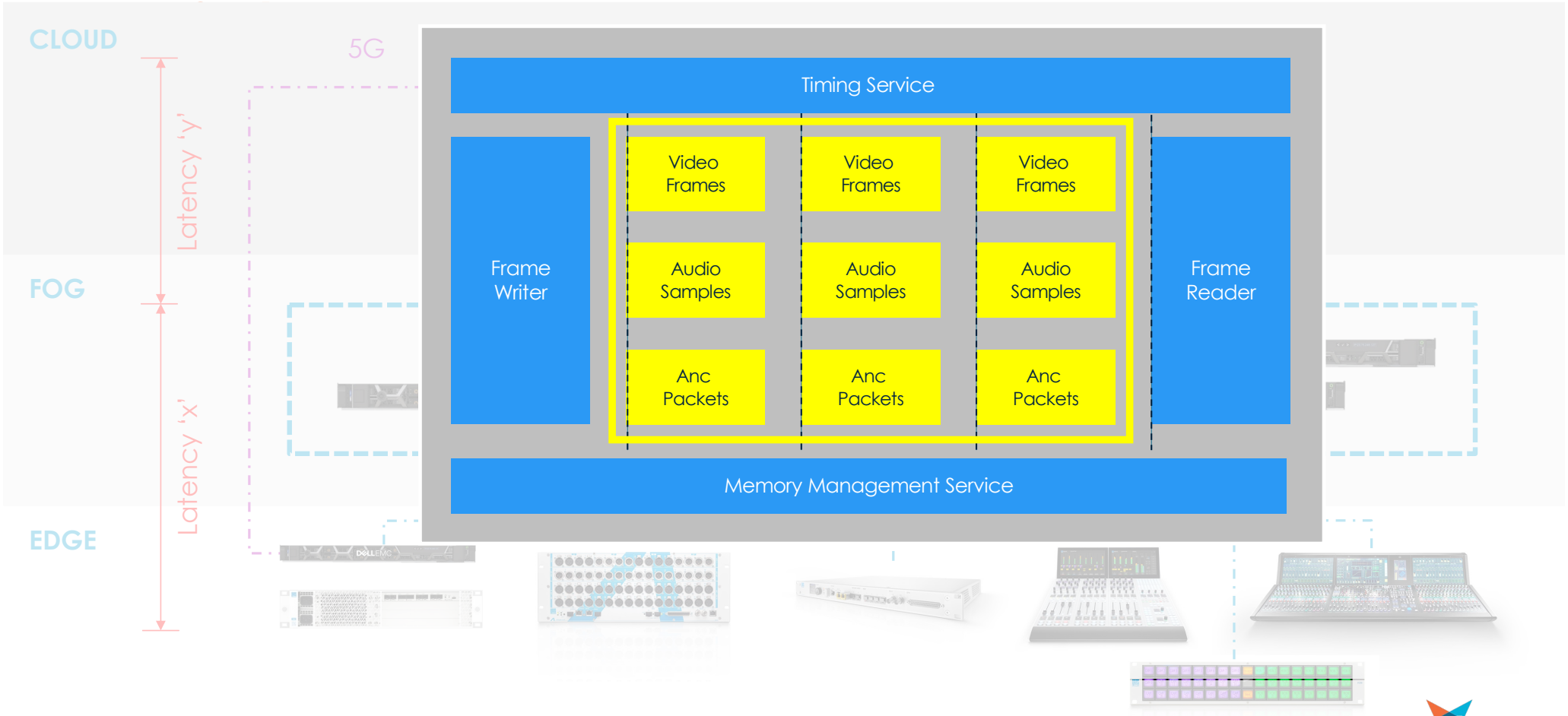
VID25 TRANS

Understanding the ‘Lower’ layers

- Built on containerised infrastructure (Docker, Kubernetes)
- Compute (CPU – x86 / Arm and GPU – discrete and integrated)
- Network topology – North / South (stream transport formats) and East / West (memory sharing)
- Multi-vendor solution – Nvidia Holoscan, Intel Tiber, etc. (discrete and clustered nodes)
- Private and Public cloud – linear expansion (cloud native, not lift and shift)
- Embrace the use of HPC technologies such as Libfabric (RoCE, EFA)

LAWO

VID25 TRANS



Edge to cloud Media Exchange Layer !

Conclusions

- Built for the future, not inherently built on legacy (v210 10-bit video is a good starting point)
- AV media functions should not be grounded by the lowest common denominator – i.e. Audio and Video, Audio only, Video only – not one size fits all (different applications, have different needs – i.e. latency)
- Convergence is inevitable – live and non-live workloads within the same infrastructure
- Standardisation will slow the progress – an open-source approach is needed (Apache 2 ?)
- Call to action to the industry – please get actively involved in the EBU work around the DMF
- Important – customer projects are here and are being planned (time critical, no time to lose)

LAWO

VID25 TRANS

Thank you !

Phillip Myers
phillip.myers@lawo.com