SMPTE 2021 ATC Manuscript template

Tighter NIC/GPU Integration Yields Next Level Media Processing Performance

Thomas Kernen

NVIDIA

Thomas True

NVIDIA

Written for presentation at the

SMPTE 2021 Annual Technical Conference & Exhibition

Abstract. *As the media industry further consolidates building services based on Commercial-Off-The-Shelf (COTS) hardware, the requirement for increasing performance continues to accelerate. The processing of video resolutions up to Ultra HD (UHD) and 8K, higher frame rates, increased bit depth, and High Dynamic Range (HDR) imagery, requires tighter integration and optimization between the Network Interface Controller (NIC) and the Graphical Processing Unit (GPU). Combined with simultaneous input and output of multiple streams, this creates the potential for performance bottlenecks that must be unlocked.*

This paper describes how COTS hardware platforms running GPUs alongside ST 2059-2 PTP locked NICs which are accurately pacing packets according to ST 2110-21 requirements may further increase their performance throughput by reducing CPU driven data copy overhead. And in doing so, reduce processing latency and jitter, and more effectively use GPU resources while freeing up CPU resources.

Keywords. GPU, NIC, DPU, PTP, 2110, Timing.

Introduction

As live media production workflows continue to embrace IT-based systems for their transition from SDI to IP, the focus on designing tightly coupled solutions between the different components increases. This is specifically due to the introduction of the SMPTE ST 2110[1] disaggregated essence model whereby audio, video and metadata are transmitted as individual RTP streams, requiring a tight alignment of these flows at source and destination to be maintained via IEEE 1588 Precision Time Protocol[2], as defined in the SMPTE ST 2059-2 PTP Profile[3].

Leveraging the Input/Output function of Network Interface Cards (NIC) designed to offload Data Center class applications at speeds in excess of 100 Gigabits per second per interface is now common practice. In order to unlock such performance capabilities, the entire Commercial-Off-The-Shelf (COTS) based hardware system, i.e: the sum of the individual components, must be set up in an optimal manner, something common to hyperscalers for their demanding applications and workloads. In this context, we need to transpose those capabilities to the world of ST 2110 IP flows.

This is performed via a series of optimizations that range across a broad spectrum of capabilities: A Data Processing Unit (DPU) incorporated into the SmartNIC running the PTP stack to provide an Operating System (OS) agnostic timing reference for accurate packet scheduling of ST 2110 flows or bypassing the CPU and with the direct data transfer between the NIC and the Graphics Processing Unit (GPU). These lead to an increase in throughput for all media processing applications.

ST 2110 & COTS Network Adapters

Ever since the original design for the Network Interface Controller (NIC) in the early 1970s, the packet rate has been slowly but steadily increasing over the years. Today, 100 Gigabit/s NICs are not uncommon in high-performance servers designed to run and accelerate datacenter workloads. The exact same Common Of The Shelf (COTS) hardware is found in modern media processing servers.

As the bandwidth rates increased, the packet processing rate on the host was not able to keep up with the original single queue First In, First Out (FIFO) model. To break this bottleneck requires several innovations that are now common in high-performance NICs. Here are some examples that are used by many NIC implementations :

- Direct Memory Access (DMA) copying data to/from the NIC to memory thereby removing the intermediate CPU processing stage.
- Packet checksum computation offload[4] for send and/or receive messages to the NIC
- Segmentation Offload[5] moves a multipacket buffer to the NIC which then splits this buffer into separate packets thereby freeing up CPU resources. This may apply to TCP and/or UDP packets.
- Receiver Side Scaling (RSS)[6] enables packets to be distributed to separate queues, each queue then being assigned to a different CPU or CPU core thereby reducing the

risk of packets being handled by a CPU core currently swamped by inbound data while others are idle.

- Large Send Offload (LSO)[7] allows the TCP stack to build a longer TCP message and send it in one call down the stack. The adapter then re-segments the message into multiple TCP packets. This offloads a large amount of kernel processing from the CPU to the NIC.
- Large Receiver Offload (LRO), also known as Receive Segment Coalescing (RSC) [8], provides the same kind of functionality as above but for received packets[9].
- UDP Segmentation Offload (USO) provides similar capabilities to LSO for UDP packets.
- Header-Data split[10] improves network performance by splitting the headers and data in received Ethernet frames into separate buffers.

These offloads are some examples of what is achievable with modern high-performance NICs to achieve sending and/or receiving packets at line rate of the NIC port(s).

Additionally, kernel bypass capabilities allow for further performance in the operating system. Reducing the OS kernel overhead by enabling applications to directly access the network adapter resources. Frameworks designed for media workflows such as NVIDIA's Rivermax SDK[11] not only leverage the NIC offloads and the kernel bypass capabilities but also provide a common multi-OSs (Linux and Windows) API layer to program the NIC to send and/or receive ST 2110 media flows.

Timing and pacing constraints in ST 2110-21

SMPTE ST 2110-21[12] defines the traffic shaping and delivery timing for the ST 2110 video flows. These flows must adhere to specific criteria for the sender/source and network model. This is a critical part of achieving the required performance for operating over an IP network. The reason for shaping this traffic is not only to minimize packet delay variations (PDV), which could otherwise lead to jitter, increased latency and/or dropped packets but also to avoid congestion. This is managed via accurate packet pacing at a fixed rate, per flow, derived from the media resolution and frame rate. A detailed analysis of the impact of the ST 2110-21 traffic model on media flows can be found here[13] and measurement results here[14].

Therefore, compliance with ST 2110-21's strict timing accuracy requirements, combined with high throughput interfaces for sending and/or receiving multiple video streams at high resolution/depth//bitrates require a full set of NIC and OS offload hardware capabilities. These shall ensure that the I/O performance is maximized while minimizing the impact on the timing elements such as the PTP stack.

Timing as a Service (DPU)

What is a DPU?

For many years, Central Processing Units (CPUs) were the sole programmable element in most computers. Flexible and responsive they handled the processing of all the compute workloads.

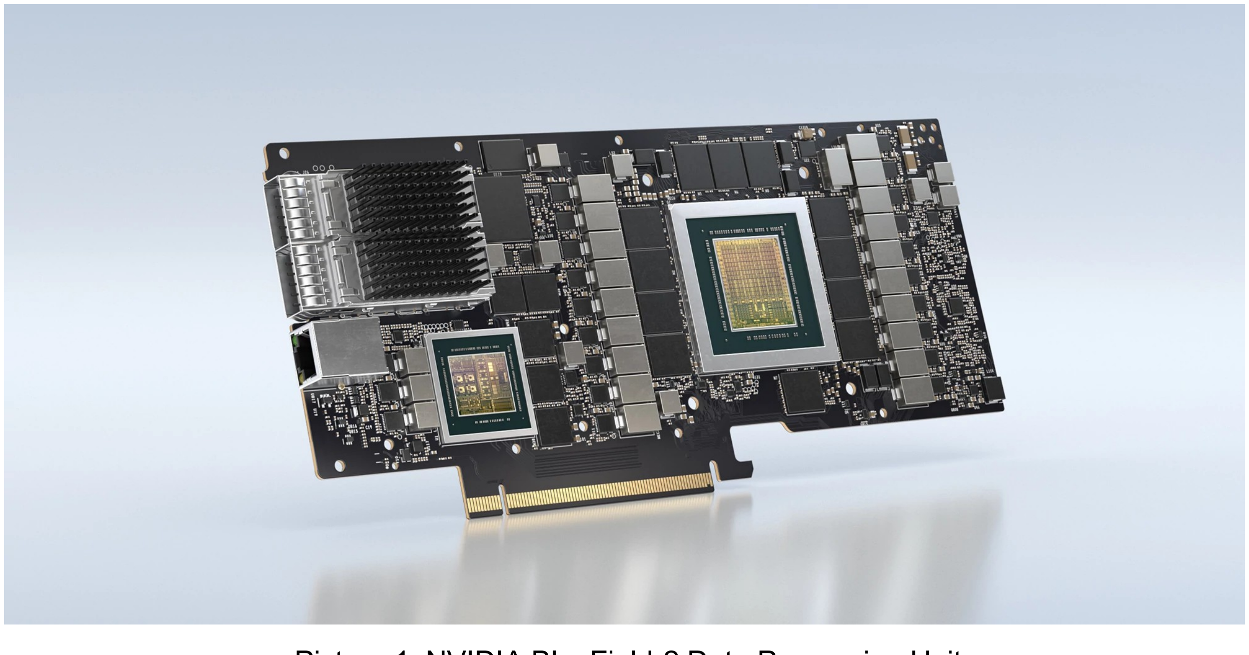
More recently the Graphics Processing Unit (GPU), has taken a central role. Originally used to deliver rich, real-time graphics, their parallel processing capabilities make them ideal for accelerated computing tasks of all kinds. Thanks to these capabilities, GPUs are essential to many types of more recent workloads such as artificial intelligence, deep learning and big data analytics applications.

The introduction of the Datacenter Processing Unit (DPU) as the 3rd pillar of the data-centric accelerated computing model was designed to move the data around between the different parts of the workloads, across the CPU/GPU boundaries between hosts.

As such the DPU is a new class of programmable processors that combines three key elements. As a system on a chip (SOC), a DPU combines:

1. A high-performance, software-programmable, multi-core CPU, typically based on the widely used ARM architecture and tightly coupled to the other SoC components.
2. A high-performance network interface capable of parsing, processing and efficiently transferring data at line rate to GPUs and CPUs.
3. A rich set of flexible and programmable acceleration engines that offload and improve application performance.

While the DPU can be used as a stand-alone embedded processor, it's more often incorporated into a SmartNIC as illustrated in Picture 1 below. As such it provides a programmable network adapter card with programmable offload accelerators and Ethernet connectivity on a single PCIe adapter.



Picture 1. NVIDIA BlueField-2 Data Processing Unit

Challenges in delivering consistent timing across hosts

One of the key pillars of an SMPTE ST 2110 system is a stable and accurate time transfer built upon the IEEE 1588 Precision Time Protocol (PTP). The ST 2059 specifications define the PTP profile parameters for the time transfer and the related timing metadata consumed by the endpoints (PTP Followers).

In a classical bare metal system, the PTP stack is the only entity that controls the PTP timing system via the PTP Hardware Clock (PHC)[15]. When the PTP stack is running as a PTP Follower, it disciplines the PHC via the PTP messages that are received by the host. Further information about PTP behavior in SMPTE ST 2110 networks, its efficiency and monitoring can be found here[16], here[17] and here[18].

Assuming that the host is Linux based and using hardware timestamping capable NIC with a stable timestamping engine such as the NVIDIA ConnectX 6Dx NIC[19], the expected performance will be in line with the SMPTE PTP ST 2110-21 requirements for packet pacing multiple media flows concurrently at all resolutions up to UHD-2/8K.

As real-time media production services evolve towards more complex workflows, based on multiple Operating Systems (OS) specifically Linux and Windows, and sharing hardware resources via virtualization of the host functions (CPU, GPU, NIC, memory, storage, ...) the requirements for a consistent timing reference solution across all the different system combinations increases dramatically.

Excluding all upstream timing-related elements such as GrandMaster(s) and PTP-aware network switches, there are still several components on each host that may introduce variability in the time transfer and accurate and stable representation of time.

Such elements include:

- PTP stack parameters and servo control loop that differ per stack implementation leading to different stack behaviors when processing PTP messages.
- CPU interrupts causing a PTP stack not to process messages during a short period of time due to other tasks preempting the CPU core running to the PTP stack.
- Operating System induced system noise & jitter that further impacts process scheduling.
- PTP timestamping accuracy under system and bandwidth load.
- Hardware and/or software misconfigurations causing NIC and CPU resources for the PTP stack to not be connected to the same Non-Uniform Memory Access (NUMA)[20] thereby introducing additional jitter.
- Hardware timestamping engine accuracy differs from NIC to NIC implementation rounding values across different boundaries.
- Operating System timing capabilities differ significantly between Linux's PTP Hardware Clock Infrastructure with a full and proven framework userspace and kernel API set users by multiple hardware and software implementations vs. Windows' limited framework requiring vendor-specific PTP stack implementations for specific NICs or the upcoming native Windows support that is currently limited to software timestamping[21].

Due to this diversity in the construct of elements that feed into the time transfer capabilities, the end result across hosts will diverge. As accuracy requirements increase, i.e. improved timing resolution & clock stability, these ever so small changes in behavior add an additional burden to maintaining a coherent timing system across all hosts.

Delivering timing with a DPU

Since a DPU SmartNIC can provide both a highly efficient NIC with offload capabilities and excellent timestamping resolution combined with a host independent set of computing resources that can be tuned for specific tasks, this lays the groundwork for delivering a timing framework that is completely independent of the processes running on the host itself.

A host can therefore offload and contain the timing components independently of the number of applications that may be consuming the time transfer capabilities such as for the following use cases:

- Bare Metal server running an operating system with native PTP capabilities and one or multiple applications consuming time
- Bare Metal server running an operating system with limited or no native PTP capabilities and one or multiple applications consuming time
- Virtualized server running one or more operating systems, with or without native PTP capabilities and one or multiple applications consuming time.

Figure 1 below illustrates the third use case (Virtualized server with multiple OSs) sharing the timing capabilities managed and exposed by the DPU to the host.

The NVIDIA BlueField-2 DPU[22] uses the NIC component (based on the NVIDIA ConnectX-6 Dx NIC) for the hardware timestamping of the sent and received PTP packets from the Ethernet interface. The PTP Hardware Clock (PHC) still resides on the NIC portion of the DPU card and will be disciplined by the PTP stack as it would for a bare metal PTP native OS running a PTP stack.

Since the PTP stack now resides in an OS-independent portion of the host, i.e.: on an arm core of the DPU. The communication between the PTP stack and the PHC is done over a Physical Function (PF) allowing for the PTP stack sitting in the DPU arm core user space to communicate with the NIC. This is done to discipline the PHC as well as sending and receiving the PTP messages to/from the PTP stack to the NIC. This is the only read/write to/from the PHC that is allowed on the host and done so to avoid any other application(s) from attempting to control the PHC.

All operating systems consuming time will have a read-only capability to/from the PHC, either via a Physical Function (PF) or a Virtual Function (VF) from the NIC. Once again this prevents any other application than the PTP stack residing on the arm core of the DPU to discipline the common PHC.

Applications will then be presented with a “get_time” like function for obtaining the time for their own usage.

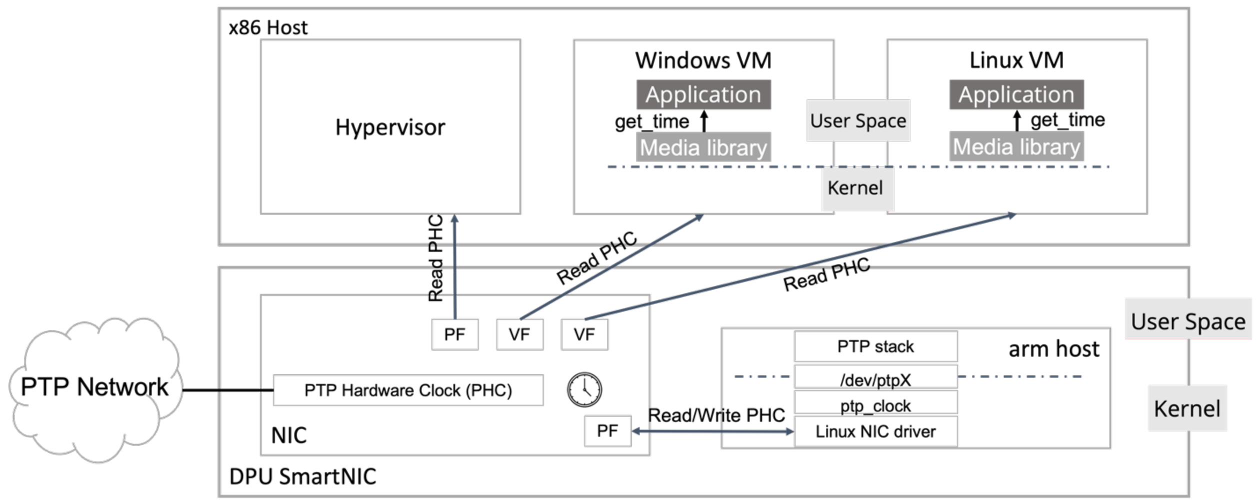


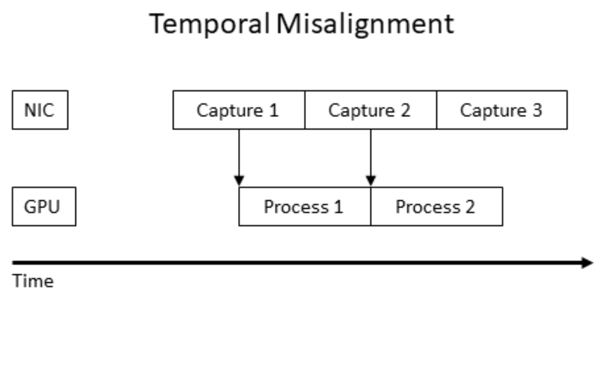
Figure 1. DPU-Based Timing Delivery in a Multi-OS Virtualized Environment

By doing so, we can ensure that only one entity oversees disciplining the PHC per host, therefore preventing misconfiguration or administrative abuse of the common timing resource shared across the different operating systems and applications served by a single host.

Syncing GPU from PTP service

In a GPU-based video processing application, frame/field synchronization of the GPU processing to the stream timebase is required for both video capture and playout. At the time of this writing, there is no hardwired connection between the GPU and the DPU providing the PTP time service and furthermore the GPU as a device does not maintain a hardware clock in sync with the PTP time. As a result, GPU synchronization at frame granularity is purely software-controlled based upon the offset between the RTP packet timestamps and the current PTP time as reported by the PTP time service running on the DPU. Direct hardware synchronization of the GPU to the PTP time is a topic of future research.

In the case of SMPTE ST 2110-20[23] and ST 2110-22[24] video capture, the RTP timestamps are the same for all packets composing an image frame and represent the image capture time. The GPU frame processing must be aligned to the frame boundaries as determined by the RTP timestamps, otherwise, frame misalignment of the GPU processing will result in image tearing as processing is performed on two partial frames that are not temporarily aligned as shown in Figure 2. This frame alignment is achieved by detecting the Marker bit in the RTP header and ensuring that the RTP timestamps for all packets in the frame are at the same offset from the current PTP time queried from the DPU time service as illustrated in Figure 3. Furthermore, if this offset drifts between successive frames, the GPU is out of sync with the video rate resulting in a non-constant latency in the GPU processing that can be problematic in a real-time broadcast or virtual production pipeline. This non-constant latency can be avoided by delaying the start of GPU processing until the current PTP time is a constant offset from the capture time and ensuring that the GPU processing always completes within the duration of the current frame. It should also be noted that this alignment is for GPU-based video essence processing only and not for direct display of the video frames by the GPU as the frame timing and alignment is not guaranteed to be aligned with the vertical retrace of the GPU display and could result in tearing or other display artifacts. Synchronization of the GPU vertical retrace to the IEEE Start of Frame Delimiter (SFD) will not be possible until a direct hardware connection between the NIC and GPU is available.



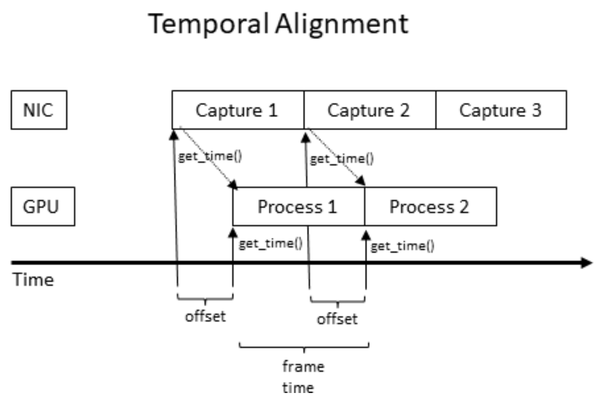
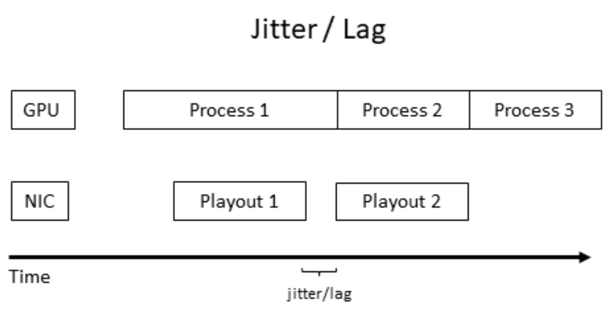


Figure 2. Temporal Misalignment between the GPU and NIC

Figure 3. Temporal Alignment between the GPU and NIC

Video Playout

The RTP timestamp of a playout frame is defined as the time point at which the first packet of the frame is presented for transmission by the NIC and all packets constituting a frame have the same timestamp. If the GPU processing is not synchronized to the video frame rate then the video playout will exhibit jitter and/or lag as shown in Figure 4. GPU processing time must consume less than a frame time to ensure GPU synchronization. The delta between successive queries of the PTP time from the DPU time service should be equal to or less than the frame rate of the transmit stream. When the frame time is exceeded, a duplicate frame may be transmitted, or there may be a stutter in the ST 2110-20 stream depending upon the application. Alignment of the frame on the outgoing transport stream is always guaranteed by the hardware scheduling of the NIC with the RTC synced to PTP. Figure 5 illustrates smooth playout when the GPU and NIC are in frame alignment.



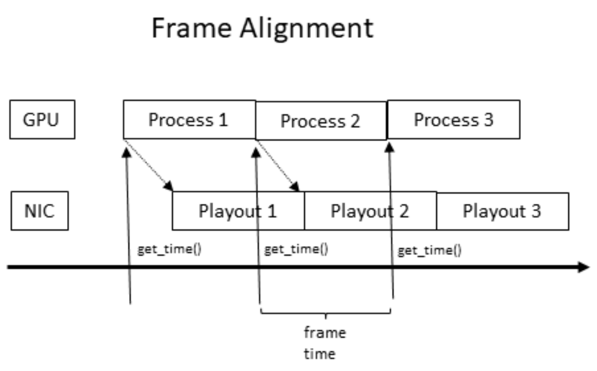


Figure 4. Jitter and/or Lag Resulting from. Misaligned GPU and NIC

Figure 5. Temporal Alignment between the GPU and NIC

Direct Data Transfer

In modern computer systems based upon the PCI Express (PCIe) architecture GPUs and NICs function as endpoint devices on the PCIe switch fabric connected to a CPU and system memory via a PCIe root complex.[25] The root complex may contain multiple PCIe ports with directly connected endpoints in addition to multiple switch devices connected directly or cascaded. Each CPU, associated root complex, local host memory, PCIe switch fabric and endpoint devices are commonly referred to as a Non-Uniform Memory Access (NUMA) node. In a system with multiple CPUs or NUMA nodes they are inter-connected via Intel QuickPath Interconnect (QPI)[26] or HyperTransport (HT)[27] as illustrated in Figure 6.

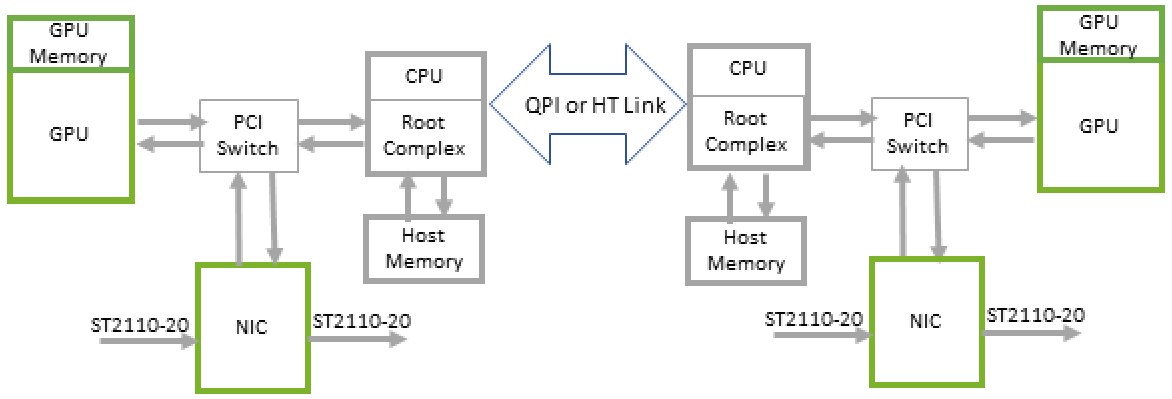
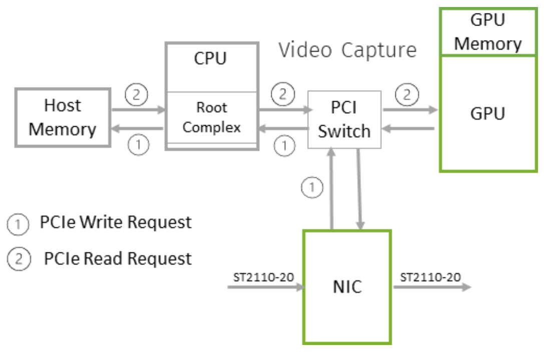


Figure 6. Multiple NUMA Nodes Connected via a QPI or HT Link

Traditional computer-based video capture or playout systems utilize GPUs to perform video processing and NICs to perform the SMPTE ST 2110-20 video I/O. Video essence data moves between the GPU and NIC PCIe endpoint devices via Direct Memory Access (DMA) transactions to or from host memory. In this scenario, an application must allocate host memory buffers for video I/O transfers to and from the NIC and GPU device memory for pre- and post-processed frames and manage the two-step DMA transfer processes between the two memory locations. In the case of video capture as shown in Figure 7, the NIC issues a memory write request, the root complex receives and processes the request writing the data to host memory, the GPU then issues a memory read request to fetch the data into GPU device memory. For video playout, as shown in Figure 8, the GPU issues a write request, the root complex receives and handles the request writing the processed video data into host memory. The NIC then issues a PCIe read request to fetch the data for video transmission. This methodology is typically referred to as peer-to-host-to-peer transfers.



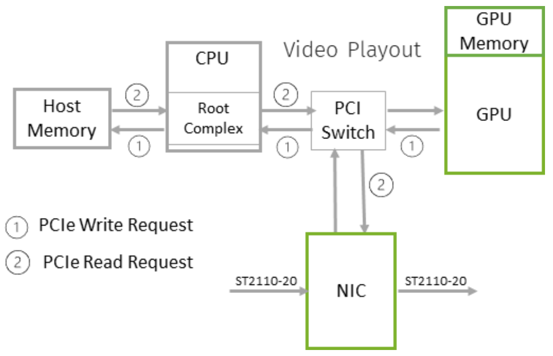
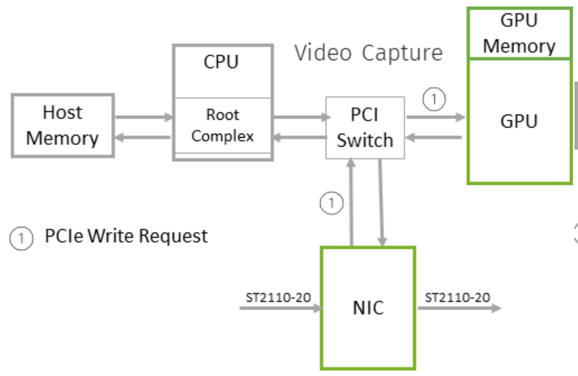


Figure 7. Traditional Video Capture to GPU Through System Memory

Figure 8. Traditional Video Playout from GPU Through System Memory

In a Direct Data Transfer system, the GPU and NIC endpoint devices perform peer-to-peer DMA transactions. In this case, the application needs to allocate only GPU device memory and register this memory with the NIC device driver. For video capture as shown in Figure 9, the NIC issues a write request directly to GPU device memory while for video playout in Figure 10, the NIC issues a read request of video essence data directly from GPU device memory bypassing the host memory. The PCIe switch on which both the GPU and NIC reside handles the PCIe write and read requests for Direct Data Transfers.



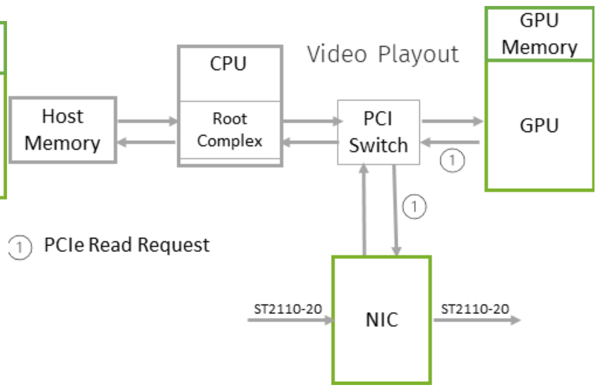


Figure 9. Video Capture Directly to GPU Device Memory

Figure 10. Video Playout Directly from GPU Device Memory

Optimum performance of Direct Data Transfers is achieved when the GPU and NIC devices are installed on the same PCIe switch device. In this case, the maximum realizable bandwidth can be achieved between devices. When the devices are not installed on the same PCIe switch device, but on different ports of the root complex, maximum performance cannot be achieved because the root complex cannot efficiently handle peer-to-peer transactions. Furthermore, when the GPU and NIC devices are attached to different NUMA nodes, data transfers fall back to the traditional peer-to-host-peer methodology.

The benefits of Direct Data Transfer to a broadcast video processing or post-production application include:

- 50% reduction in the number of PCIe bus transactions for each data transfer between NIC and GPU.
- Reduction in host memory usage as payload essence data transferred directly to/from the GPU without the need to allocate host staging buffers.
- Elimination of host memory bandwidth consumed by transfers to host staging buffers.
- Potential reduction in GPU usage previously used to DMA or copy the video essence data from host memory.
- Reduction in the transfer latency between the NIC and GPU.

Flexible Data Conversion Between NIC and GPU

Core GPU-based image processing for creative as-well-as deep-learning applications requires that the image essence is in an uncompressed planar format where each component (i.e. RGB or YCbCr) is stored as an 8-bit integer (INT8) or normalized 16- or 32-bit floating-point (FP16 or FP32) value. Meanwhile, the video essence is transmitted uncompressed in an ST 2110-20 stream where the integer color components are interleaved and packed into pgroups (an integer number of octets or bytes that does not cross a packet boundary) or compressed in a format such as JPEG XS[28], H.264/AVC[29] or H.265/HEVC[30] in an ST 2110-22 stream. In addition, colorspace conversions and other data transformations may be required between the video input and output streams and the linear color processing of the GPU-based workflow. The compute cores and on-board GPU codec hardware is utilized to convert the video essence to and from the format required for GPU processing.

In the case of an uncompressed ST 2110-20 stream the GPU with 1000s of Single Instruction Multiple Data (SIMD) compute cores and very high bandwidth (> 400 GB/sec) to local device memory performs the required transformation instantly to all pixels in parallel. As a result, once the video essence data arrives from the NIC into the GPU device memory, GPU processing commences with an efficient transformation to the required internal application format prior to processing. For transmission, the resulting video essence data is transformed back to the required format. The unit of work is a pgroup. For example, common uncompressed 10-bit YCbCr 4:2:2 video essence is transmitted in pgroups of 5 octets for each two pixels. For efficient GPU processing, a conversion to normalized FP16-per-component RGB values in the range [0.0 to 1.0f] is performed where each parallel compute thread converts the 5 octets to 2 pixels. The GPU can work on any number of pgroups and does not have to wait until a complete frame is present in GPU device memory. This removes latency from the pipeline.

Separate from the GPU compute cores used for generic parallel processing, GPUs contain on-board video codecs for common compressed formats. For ST 2110-22 streams, the onboard GPU codec can be utilized to decode the incoming stream and re-encode the outgoing stream prior to GPU processing if the stream compression is supported by the codec hardware.

Otherwise, for other non-supported codecs, the GPU compute cores can be utilized to implement all or some of the required codec processing. This flexible GPU processing pipeline for both uncompressed and compressed video streams is illustrated in Figure 11.

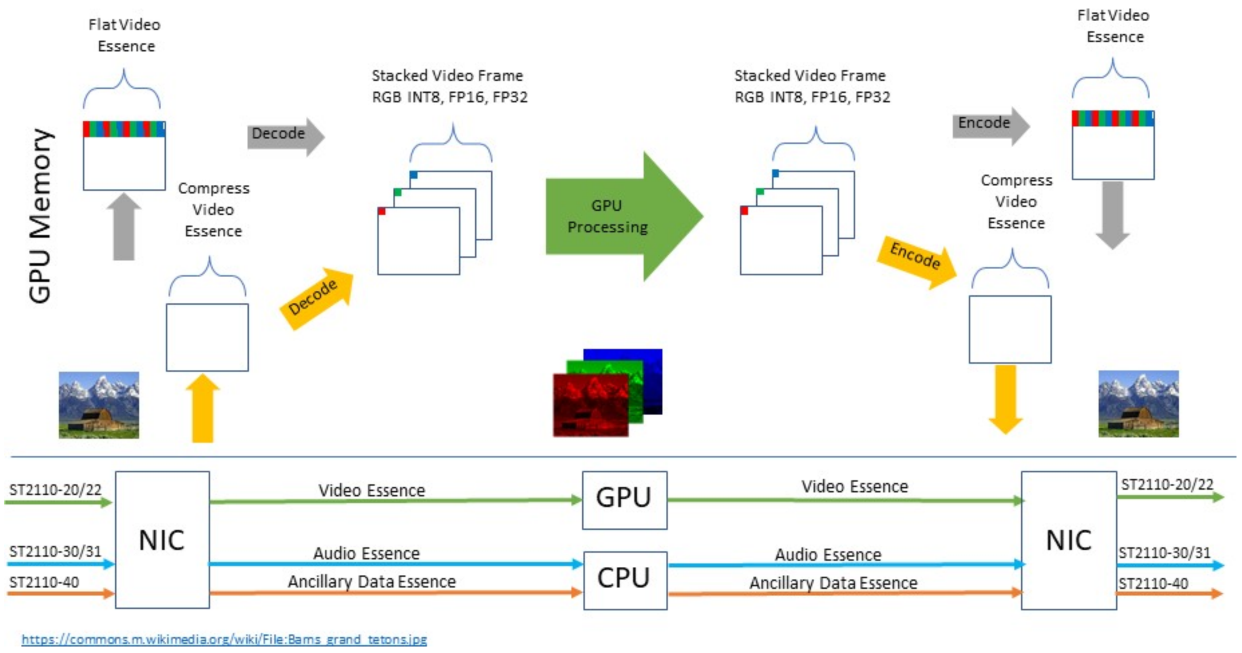


Figure 11. GPU Processing Pipeline

Bringing It All Together With a Technology Demonstrator

As a technology demonstrator, “RivermaxDisplay” combines the technologies discussed above to create an ST 2110 IP network-attached virtual display on the workstation desktop that can be utilized as a standards-conformant high-quality video reference display in a post-production workflow or for a live-to-air-application in a broadcast pipeline. High-quality color-accurate frames for display by the Windows operating system display manager are rendered on the GPU prior to transmission by the NIC as a SMPTE ST 2110-20 uncompressed video stream. Accompanying audio data is transferred to the NIC for transmission as a SMPTE ST 2110-30[31] stream. IEEE 1588-2008 Precision Time Protocol (PTP) as defined in SMPTE ST 2059-2 is used for synchronization while industry-standard Session Description Protocol (SDP)[32] manifests are used to set up the technology demonstrator media processing and transmission pipeline. The system architecture is illustrated in Figure 12.

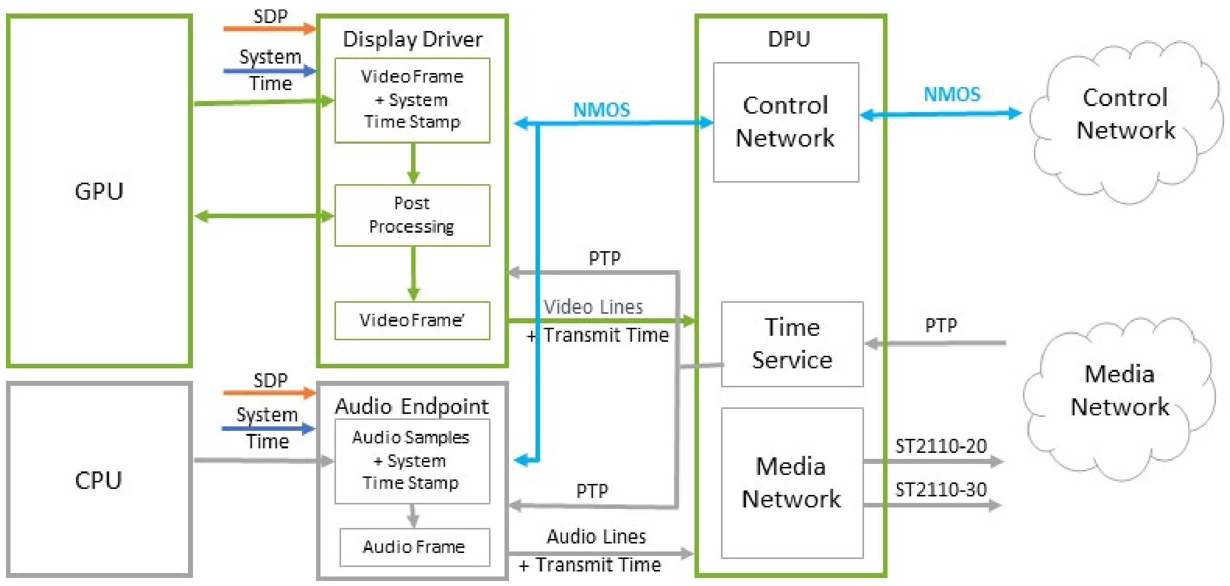


Figure 12. Technology Demonstrator Architecture

Architecture

The Windows Display Driver receives the RGB pixel data of the rendered frame for the display in GPU device memory. Prior to packetization and ultimately network transmission, the RGB pixels are converted by the GPU from the native Windows operating system format to the sampling, depth and colorspace specified for the outgoing video stream in the SDP manifest. Taking advantage of header-data split feature, the final video essence pixels are then transferred directly from GPU device memory to the NIC via Direct Data Transfer for transmission as RTP packet payload on the created SMPTE ST 2110-20 stream while the headers are transferred via DMA from host memory buffers.

Audio sample data is processed by a Windows virtual audio endpoint device.[33] When this audio device is selected in the operating system control panel to be the active audio device, the audio endpoint receives the uncompressed 16- or 24-bit PCM audio samples in a host memory buffer. The NIC then DMA’s this audio essence data as RTP packet payload along with the RTP headers for network transmission in the SMPTE ST 2110-30 audio stream as specified in the SDP manifest.

Synchronization

Integration of this technology demonstrator as a content generation node within an all-IP post-production or broadcast facility requires the audio and video essence streams to be synchronized via RTP timestamps aligned to the media clock as defined in SMPTE ST 2110-10[34]. As a synthetically generated essence, the RTP timestamp of each video frame is the time point at which the first video sample of the frame arrives at the NIC for transmission while for the audio stream, the RTP timestamp is a sample of the audio RTP clock when the first audio sample arrives at the NIC for transmission.

Synchronization of the Windows GPU-rendered frames with the audio essence buffers within technology demonstrator is performed by first mapping the Windows operating system timestamps that accompany each rendered frame buffer and audio buffer to the PTP time. The mapped PTP times are then used by the NIC to schedule the transmission of the ST 2110-20 and ST 2110-30 streams. The mapping function between the Windows system clock and the PTP time returned from the DPU time service is updated periodically over a sequence of frames by the time mapper. It is currently not possible in software to perform the required timer queries and update the map with sufficient accuracy per frame to avoid jitter. These PTP time values become the RTP timestamp inserted by the NIC into the corresponding packet headers. This alignment of video and audio is illustrated in Figure 13.

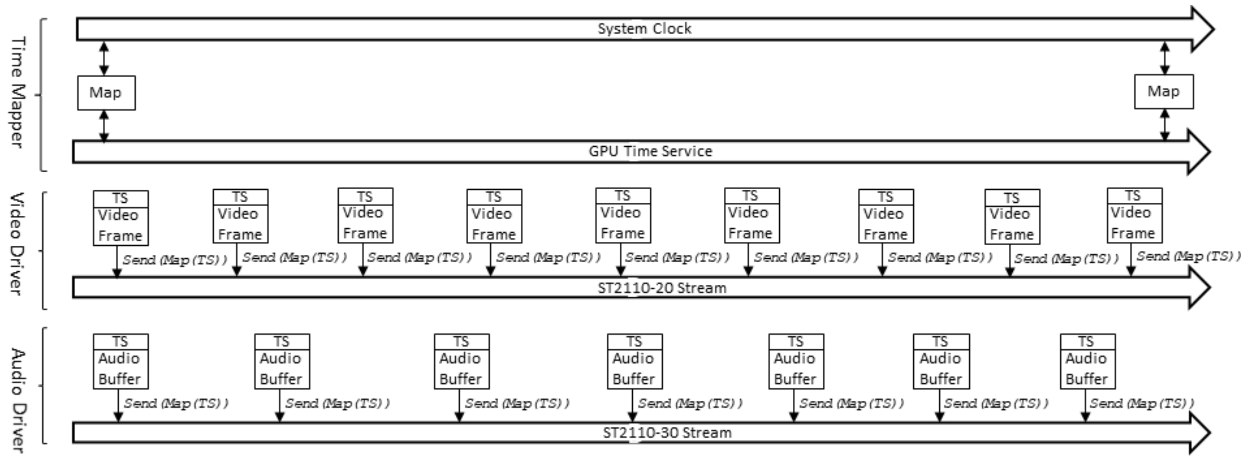


Figure 13. Technology Demonstrator Video-Audio Synchronization

Conclusion

The technology advancements in COTS NICs provide a whole range of capabilities that are applicable to offloading the host-based processing of high bandwidth flows such as those encountered in ST 2110 systems. Additionally, the precise timing and packet scheduling capabilities of the NIC ensures compliance with the strict ST 2110-21 model.

The Data Processing Unit is now the third pillar of the datacenter architecture alongside the CPU and the GPU. In the context of this paper, we demonstrate how this new class of devices enables a PTP timing platform to provide a highly accurate timing solution independently of the host operating system fulfilling the ST 2110 timing requirements.

These capabilities combined with the flexible high-bandwidth parallel processing capabilities of the GPU cores create an ideal real-time video processing system for broadcast and post-production workflows. Furthermore, the Direct Data Transfer functionality optimizes the essence transfer between the DPU and GPU to reduce system overhead and remove latency from the video processing pipeline.

The technology demonstrator brings all these pieces together to create an ST 2110 IP network-attached virtual display on the computer workstation desktop that can be utilized as either a standards-conformant high-quality video reference display in a post-production workflow or as a scan converter for sending a computer graphics desktop application live-to-air in a broadcast pipeline.

References

1. "OV 2110-0:2018 - SMPTE Overview Document - Professional Media over Managed IP Networks Roadmap for the 2110 Document Suite," in OV 2110-0:2018 , vol., no., pp.1-4, 24 Jan. 2019, doi: 10.5594/SMPTE.OV2110-0.2018.
2. "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," in IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002) , vol., no., pp.1-269, 24 July 2008, doi: 10.1109/IEEESTD.2008.4579760.
3. "ST 2059-2:2021 - SMPTE Standard - SMPTE Profile for Use of IEEE-1588 Precision Time Protocol in Professional Broadcast Applications," in ST 2059-2:2021 , vol., no., pp.1-22, 4 June 2021, doi: 10.5594/SMPTE.ST2059-2.2021.
4. Linux Kernel - Checksum Offloads https://www.kernel.org/doc/html/latest/networking/checksum-offloads.html
5. Linux Kernel - Segmentation Offloads https://www.kernel.org/doc/html/latest/networking/segmentation-offloads.html
6. Microsoft - Introduction to Receive Side Scaling https://docs.microsoft.com/en-us/windows-hardware/drivers/network/introduction-to-receive-side-scaling
7. Microsoft - Offloading the Segementation of Large TCP Packets https://docs.microsoft.com/en-us/windows-hardware/drivers/network/offloading-the-segmentation-of-large-tcp-packets
8. Microsoft - Receive Segment Coalescing (RSC) https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh997024(v=ws.11)
9. Microsoft - UDP Segmentation Offload (USO) https://docs.microsoft.com/en-us/windows-hardware/drivers/network/udp-segmentation-offload-uso-
10. Microsoft - Header-Data Split Architecture https://docs.microsoft.com/en-us/windows-hardware/drivers/network/header-data-split-architecture
11. NVIDIA Rivermax SDK https://developer.nvidia.com/networking/rivermax
12. "ST 2110-21:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: Traffic Shaping and Delivery Timing for Video," in ST 2110-21:2017 , vol., no., pp.1-17, 27 Nov. 2017, doi: 10.5594/SMPTE.ST2110-21.2017.
13. Kernen T., Vermost W.,"The art of conforming to SMPTE 2110-21 traffic model", Broadcast Engineering and Information Technology Conference (BEITC) Proceedings, April 2018, Las Vegas
14. Kernen T., Vermost W., Kerö N. "Measurement methodology and real-world compliance results for ST 2110-21 devices", Broadcast Engineering and Information Technology Conference (BEITC) Proceedings, April 2019, Las Vegas

15. PTP hardware clock infrastructure for Linux https://www.kernel.org/doc/html/latest/driver-api/ptp.html
16. Kerö N., Kernen T., Using PTP for Time & Frequency in Broadcast Applications Part 1: Introduction, EBU Technical Review March 2018, https://tech.ebu.ch/docs/techreview/trev_2018-Q2_PTP_in_Broadcasting_Part_1.pdf
17. N. Kerö, T. Kernen and T. Müller, "Efficient Monitoring of ST2059-2 Based Time Transfer Performance," in SMPTE Motion Imaging Journal, vol. 126, no. 4, pp. 1-8, May-June 2017, doi: 10.5594/JMI.2017.2680560.
18. T. Kernen and N. Kerö, "Monitoring and Analysis of SMPTE ST 2059-2 PTP Networks and Media Devices," in SMPTE Motion Imaging Journal, vol. 130, no. 6, pp. 10-19, July 2021, doi: 10.5594/JMI.2021.3082992.
19. NVIDIA ConnectX-6Dx NIC https://www.nvidia.com/en-us/networking/ethernet/connectx-6-dx/
20. Sergey Blagodurov, Sergey Zhuravlev, Alexandra Fedorova, and Ali Kamali. 2010. A case for NUMA-aware contention management on multicore systems. In Proceedings of the 19th international conference on Parallel architectures and compilation techniques (PACT '10). Association for Computing Machinery, New York, NY, USA, 557–558. DOI: 10.1145/1854273.1854350
21. Insider preview - Windows Time service in Windows Server 2019 https://docs.microsoft.com/en-us/windows-server/networking/windows-time-service/insider-preview
22. NVIDIA Bluefield Data Processing Unit https://www.nvidia.com/en-us/networking/products/data-processing-unit/
23. "ST 2110-20:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: Uncompressed Active Video," in ST 2110-20:2017 , vol., no., pp.1-22, 27 Nov. 2017, doi: 10.5594/SMPTE.ST2110-20.2017.
24. "ST 2110-22:2019 - SMPTE Standard - Professional Media Over Managed IP Networks: Constant Bit-Rate Compressed Video," in ST 2110-22:2019 , vol., no., pp.1-6, 14 Aug. 2019, doi: 10.5594/SMPTE.ST2110-22.2019.
25. Budruk, Ravi, An Introduction to PCI Express, https://www.mindshare.com/files/resources/MindShare_Intro_to_PCIe.pdf.
26. Intel, An Introduction to the Intel QuickPath Interconnect, https://www.intel.com/content/www/us/en/io/quickpath-technology/quick-path-interconnect-introduction-paper.html
27. HyperTransport Consortium, Meeting the I/O Bandwidth Challenge: How HyperTransport Technology Accelerates Performance in Key Applications, https://www.techonline.com/wp-content/uploads/2020/09/media-1035173-hypertransport_apps.pdf
28. JPEG XS (ISO/IEC 21122-1) Whitepaper, http://ds.jpeg.org/whitepapers/jpeg-xs-whitepaper.pdf

29. T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003, doi: 10.1109/TCSVT.2003.815165.
30. G. J. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1649-1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.
31. "ST 2110-30:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: PCM Digital Audio," in ST 2110-30:2017 , vol., no., pp.1-9, 27 Nov. 2017, doi: 10.5594/SMPTE.ST2110-30.2017.
32. Internet Engineering Task Force (IETF), 2006. RFC 4566 SDP: Session Description Protocol, https://www.ietf.org/rfc/rfc4566.txt
33. Microsoft - Universal Windows Drivers for Audio, https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-universal-drivers
34. "ST 2110-10:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: System Timing and Definitions," in ST 2110-10:2017 , vol., no., pp.1-17, 27 Nov. 2017, doi: 10.5594/SMPTE.ST2110-10.2017.