FASTSOCKETS OVERVIEW

A UNIQUE, PLATFORM INDEPENDENT MIDDLEWARE FOR FINANCE, CAPTURE AND MEDIA

Dr. Markus Fischer |
Director, NEIO Systems, Ltd. |
markus.fischer@fastsockets.com

VERSION 1.201 – 09/14/2025 - CONFIDENTIAL

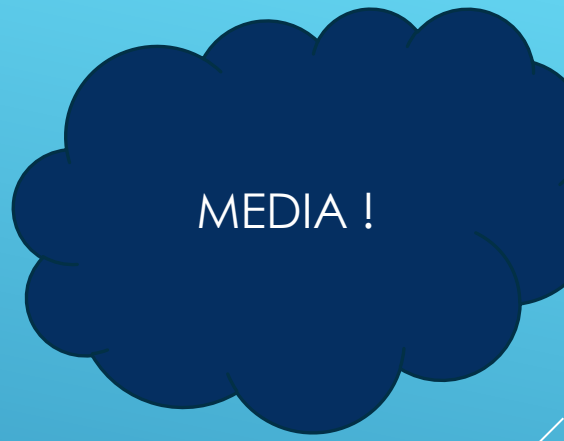
NETWORK IO AS THE BOTTLENECK

- Upcoming network speeds beyond 1Gbps introduce high CPU loads, message drops, jitter.
- Legacy interfaces are cumbersome giving access to feature like hardware based RX or TX timestamps
- Legacy OSI layers introduce additional overhead with multiple copies of data

FASTSOCKETS – AN INNOVATIVE LIBRARY ADDRESSING FUTURE NETWORK TRENDS

- Kernel bypass avoids unnecessary data copies
  - Additional GPU Direct support for improved number crunching
- Improved (user level) TCP/IP stack for faster Transmit (Trade Execution)
- RAW interface for capture - LOSSLESS
- Access to hardware based RX and TX timestamps
- One API – multiple network Vendors
  - Scale from 1Gbps to 100Gbps
  - IN Place – Depict GigE Vision with Zero Copy
- Linux and Windows support

FASTSOCKETS – AN INNOVATIVE LIBRARY ADDRESSING FUTURE NETWORK TRENDS



- Kernel bypass avoids unnecessary data copies
  - Additional GPU Direct support for improved number crunching
- Improved (user level) TCP/IP stack for faster Transmit (Trade Execution)
- RAW interface for capture - LOSSLESS
- Access to hardware based RX and TX timestamps
- One API – multiple network Vendors
  - Scale from 1Gbps to 100Gbps
  - IN Place – Depict GigE Vision with Zero Copy
- Linux and Windows support

FASTSOCKETS – AN INNOVATIVE LIBRARY ADDRESSING FUTURE NETWORK TRENDS

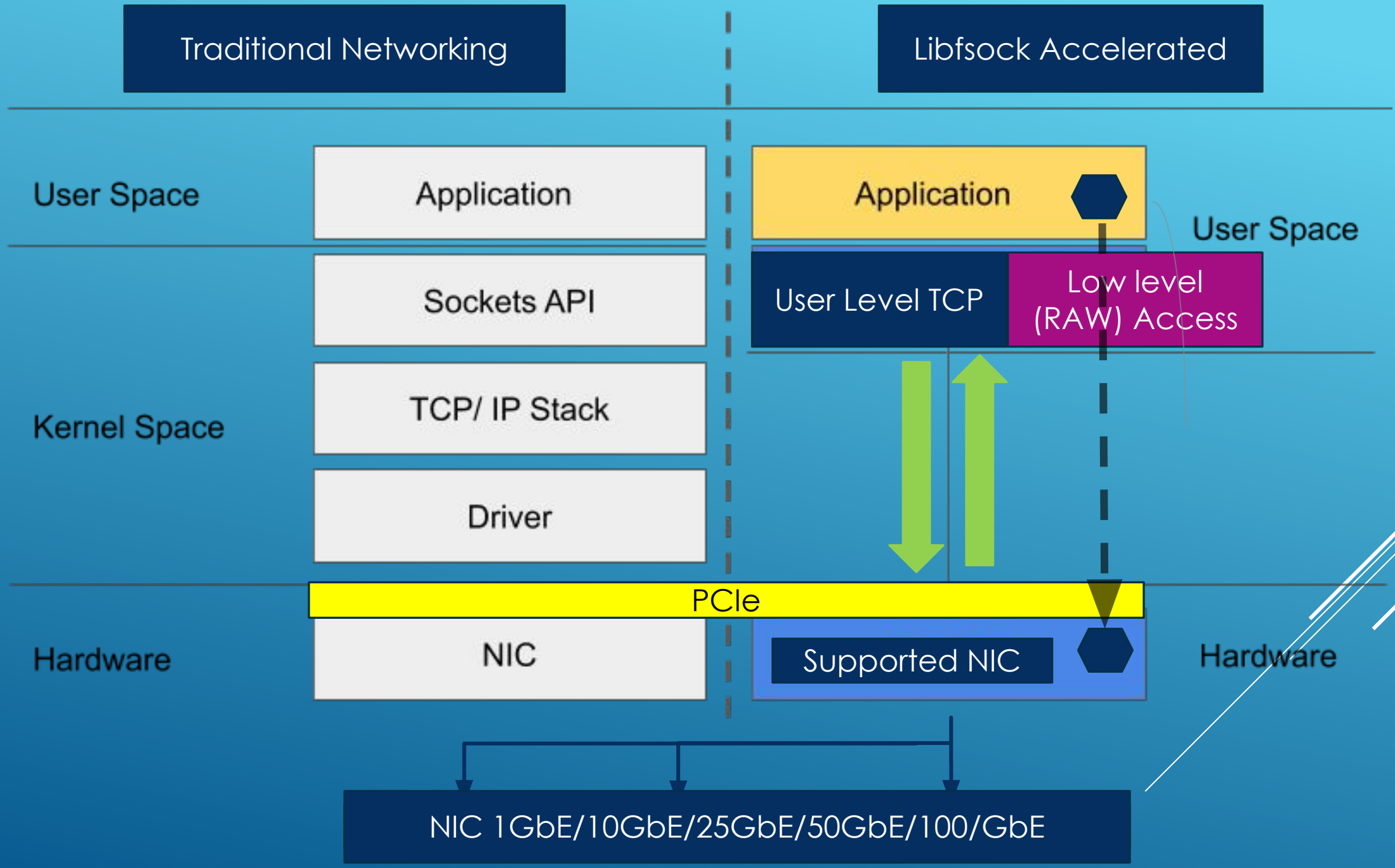
- Kernel bypass avoids unnecessary data copies
  - Additional GPU Direct support for improved number crunching
- Improved (user level) TCP/IP stack for faster Transmit (Trade Execution)
- RAW interface for capture - LOSSLESS
- Access to hardware based RX and TX timestamps
- One API – multiple network Vendors
  - Scale from 1Gbps to 100Gbps
  - IN Place – Depict GigE Vision with Zero Copy
- Linux and Windows support

FASTSOCKETS – AN INNOVATIVE LIBRARY ADDRESSING FUTURE NETWORK TRENDS

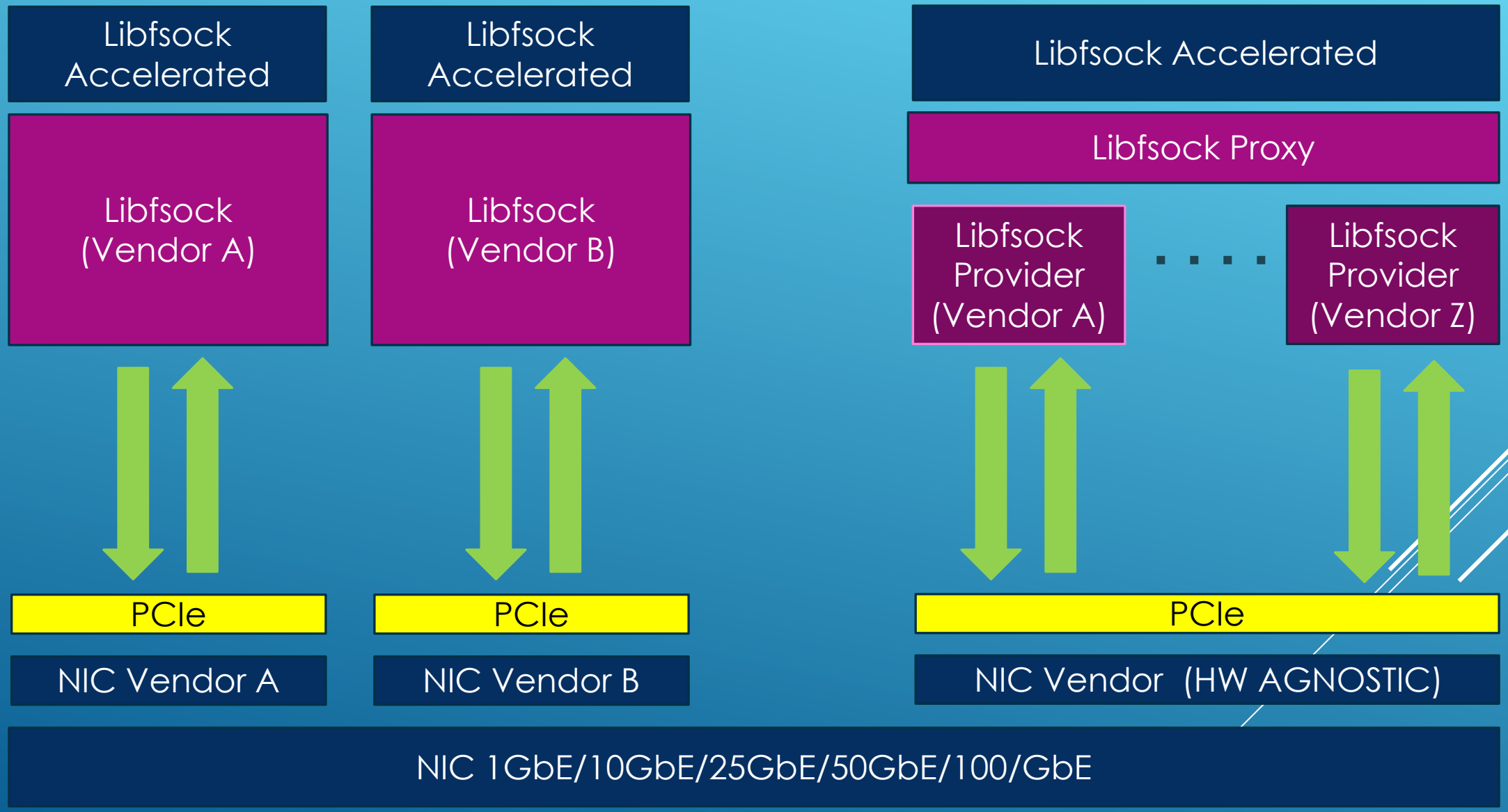
- Kernel bypass avoids unnecessary data copies
  - Additional GPU Direct support for improved number crunching
- Improved (user level) TCP/IP stack for faster Transmit (Trade Execution)
- RAW interface for capture - LOSSLESS
- Access to hardware based RX and TX timestamps
- One API – multiple network Vendors
  - Scale from 1Gbps to 100Gbps
  - IN Place – Depict GigE Vision with Zero Copy
- Linux and Windows support

FASTSOCKETS – AN INNOVATIVE LIBRARY ADDRESSING FUTURE NETWORK TRENDS

- Kernel bypass avoids unnecessary data copies
  - Additional GPU Direct support for improved number crunching
- Improved (user level) TCP/IP stack for faster Transmit (Trade Execution)
- RAW interface for capture - LOSSLESS
- Access to hardware based RX and TX timestamps
- One API – multiple network Vendors
  - Scale from 1Gbps to 100Gbps
  - IN Place – Depict GigE Vision with Zero Copy
- Linux and Windows support


Traditional Networking
Libfsock Accelerated
User Space
Application
Application
User Space
Sockets API
User Level TCP
Low level
(RAW) Access
Kernel Space
TCP/ IP Stack
Driver
PCIe
Hardware
NIC
Supported NIC
Hardware
NIC 1GbE/10GbE/25GbE/50GbE/100/GbE

LIBFSOCK Single Vendor vs Multi Vendor support via PROXY and Providers



LIBFSOCK PROXY ENHANCEMENTS

- The concept of PROXY and Providers allows for easy integration to the libfsock API and autodetection of Hardware specific providers
  - PROXY for BOTH LINUX and WINDOWS
- Proxy will detect providers
- Proxy will choose the correct provider based on IP address
- Finally, this allows for mixed use of different vendors in the same application

WinSock Sockets Application

WinSock Sockets 2.2 Application

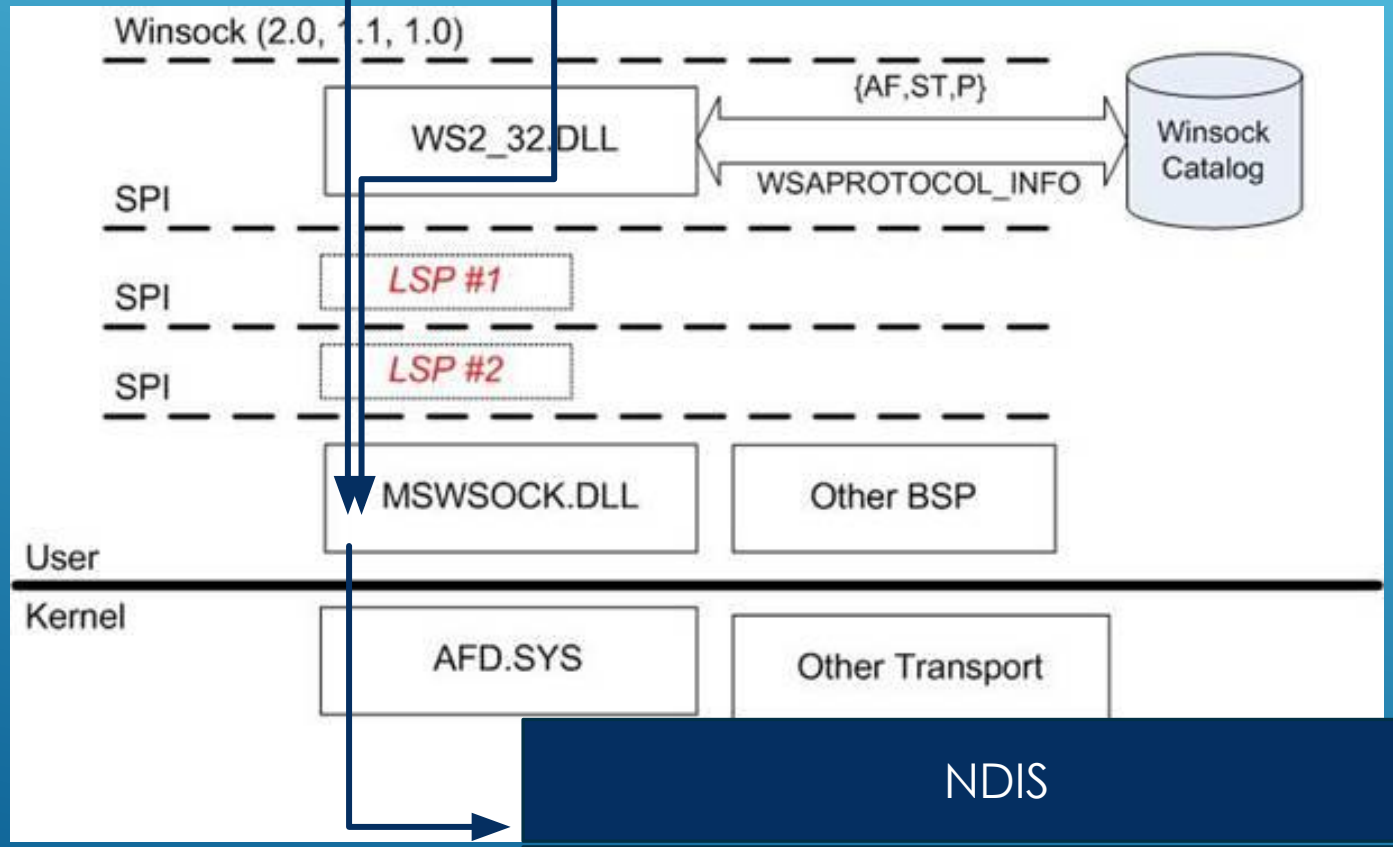
WinSock / libfsock Application

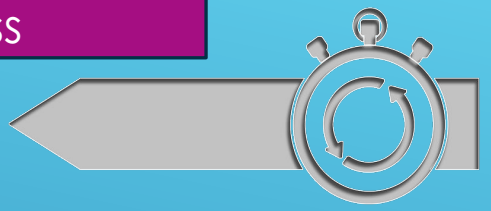
Winsock DLL

WSOCK32 DLL

User Level TCP

Low level (RAW) Access





Kernel Bypass!

NDIS

PCIe

NIC 1GbE/10GbE/25GbE/50GbE/100GbE

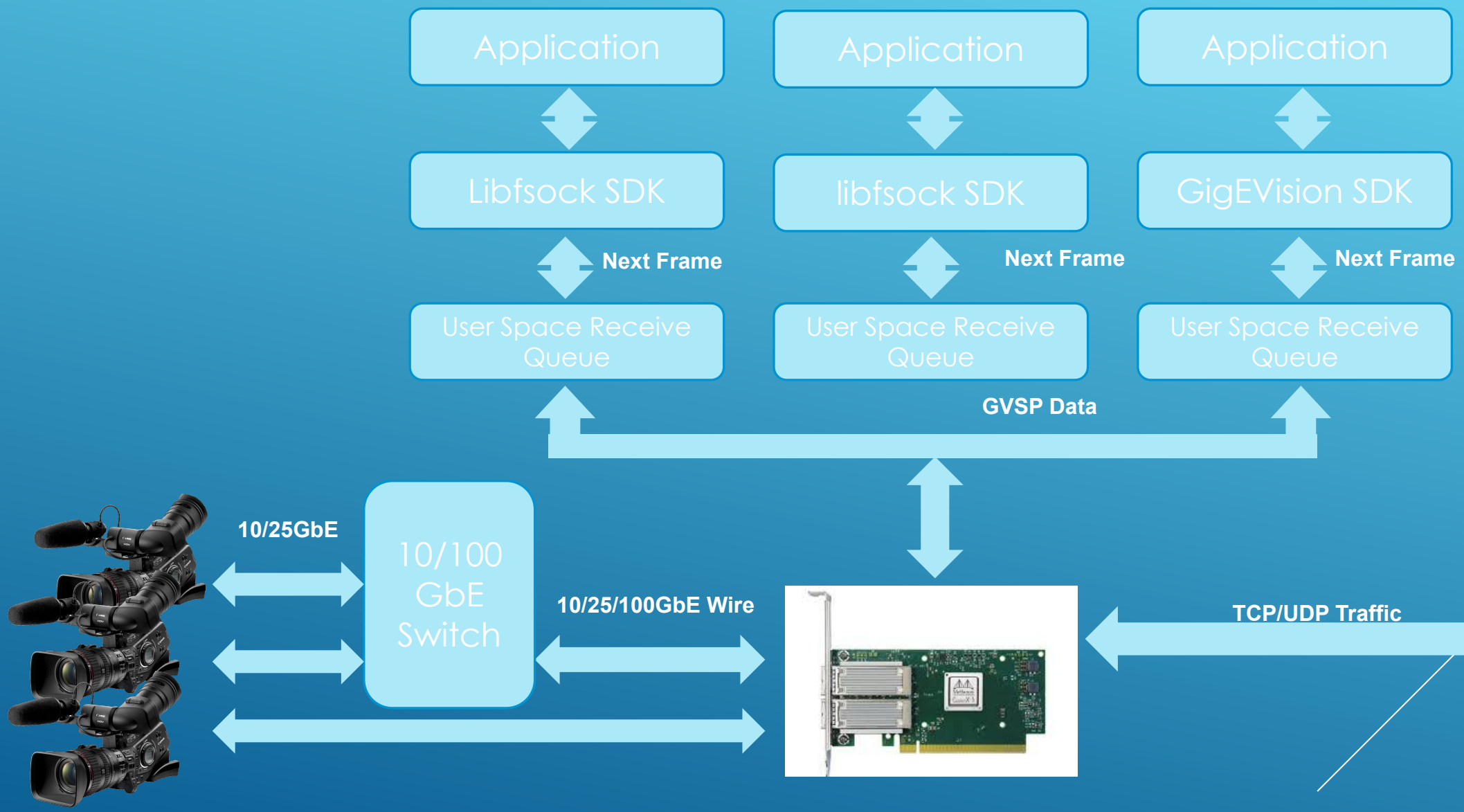
LIBFSOCK FOR FINANCE

- 1/10/25/50/100 GbE support on supported HW
  - Take NIC for best purpose
- Map / transfer applications to kernel-bypass
  - Lowest latencies, Lowest jitter
  - Lowest tick to trade
  - Tools like exanic-capture / **exanic-replay**
- Best in class timestamping
  - HW timestamping for RX and TX for both UDP and TCP protocols
- **Windows** and **Linux** Support
  - Any recent linux distribution
  - Windows 11/ Server 2016 and later

LIBFSOCK GIGE VISION ACCELERATION

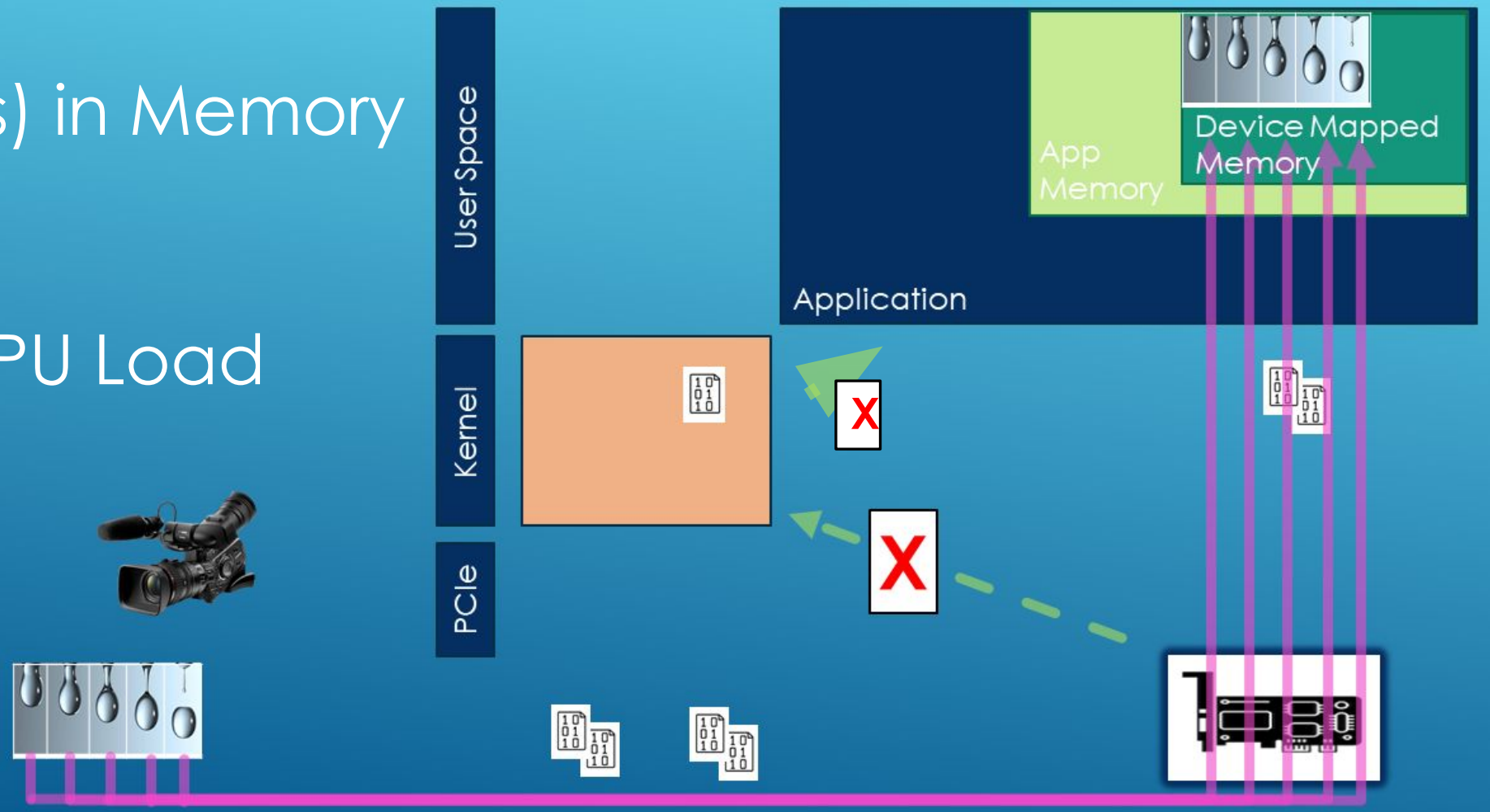
- 10/25/50/100 GbE support on supported HW
- Efficient handling of GigE Vision Protocol
  - Support of Version 1.1 and 2.1
- Hardware assisted separation of HDR and DATA (frames)
- Multiple Camera support per server:
  - 8 * 10/25GbE cameras
- Optional time stamping
- Aggregated Receives: Low CPU overhead
  - Blocking, timeout
- Windows and Linux Support
  - Windows 10 and later
  - Any recent Linux distribution

Libfsock GVA with Multiple 10/25G Cams



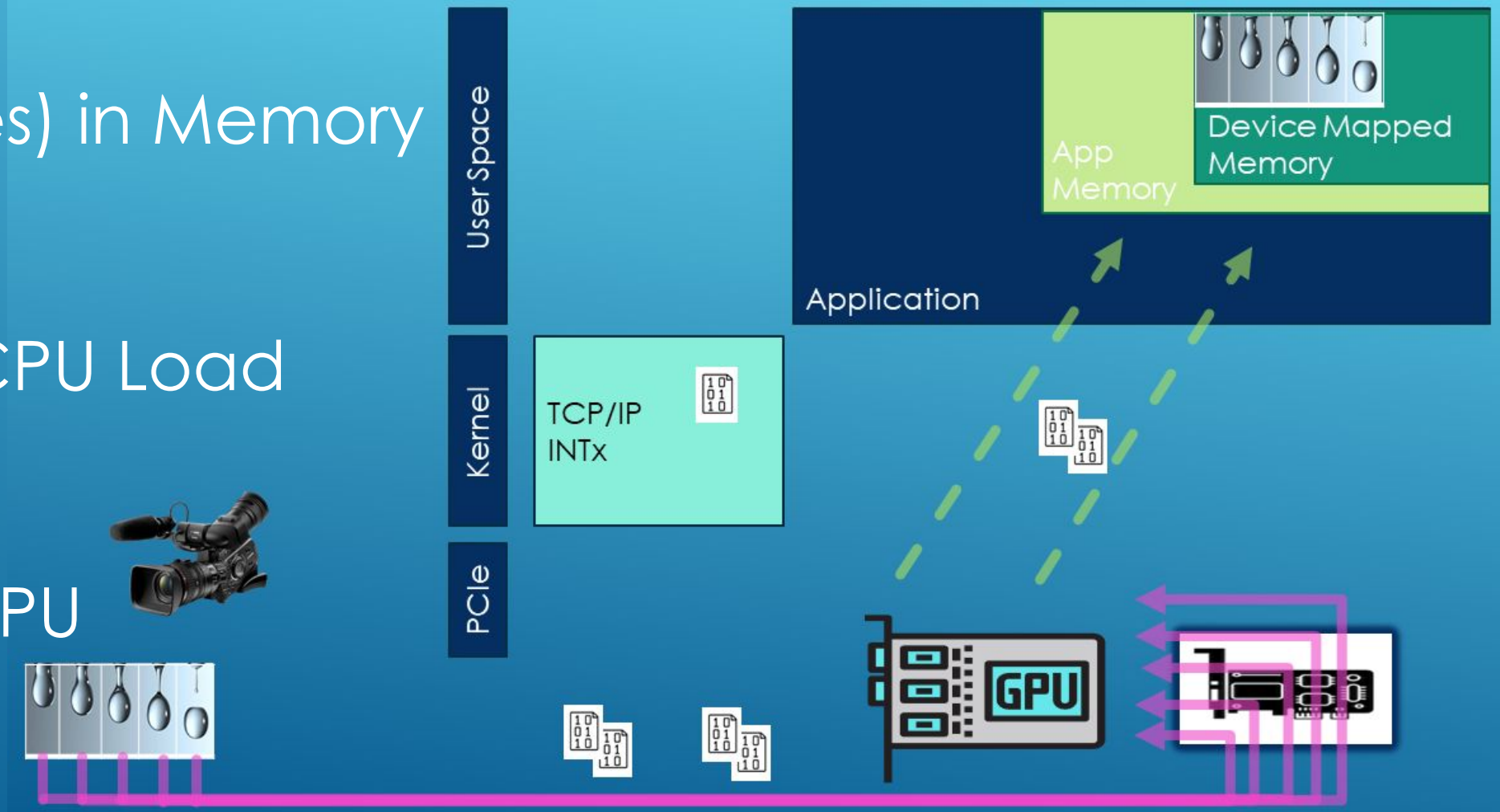
LibFSOCK GigE Vision Data Flow

- Zero Copy
- Data (Images) in Memory Placement
- Single Digit CPU Load



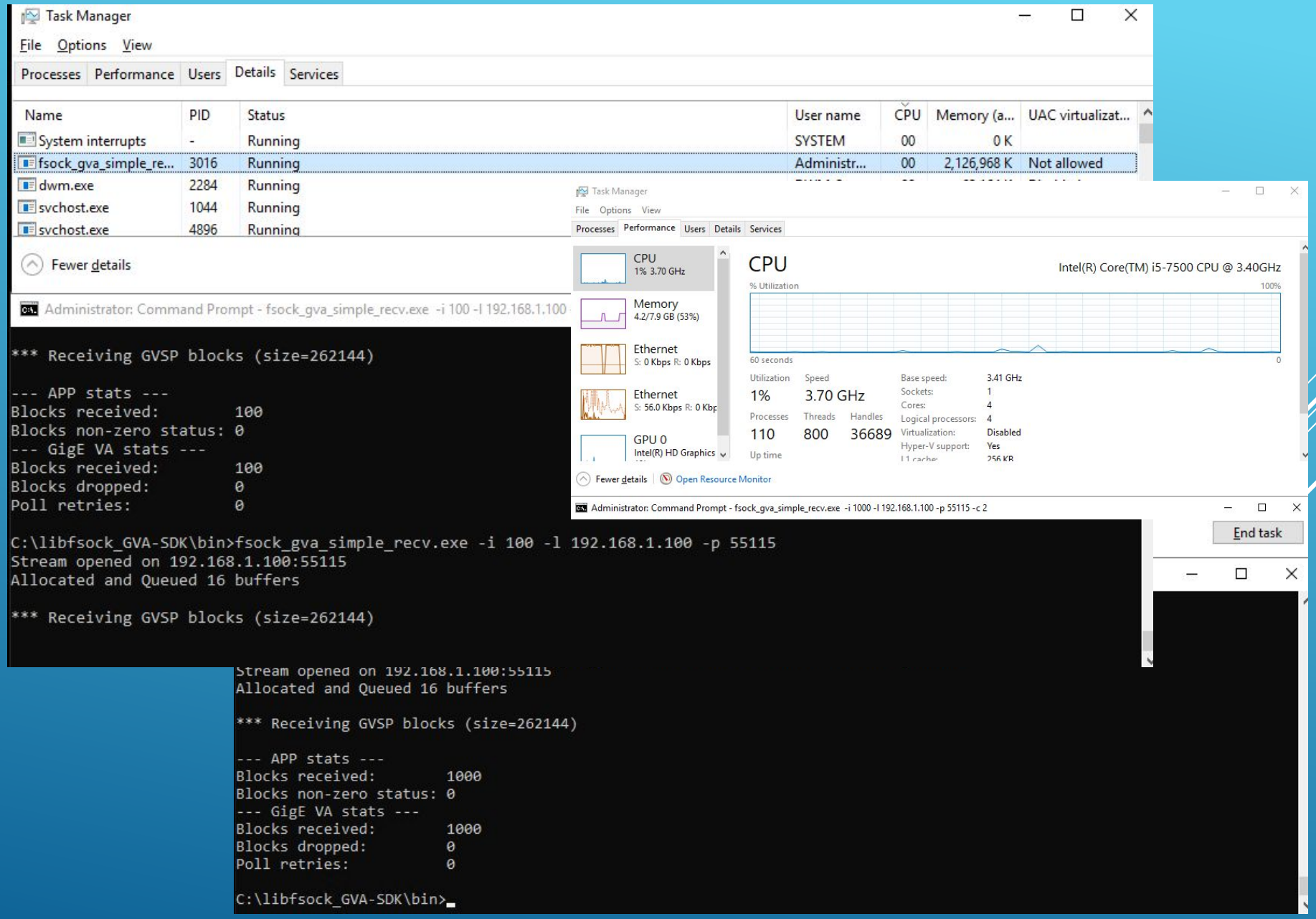
LibFSOCK GigE Vision Data Flow with GPU Support

- Zero Copy
- Data (Images) in Memory Placement
- Single Digit CPU Load
- Additional Offload to GPU



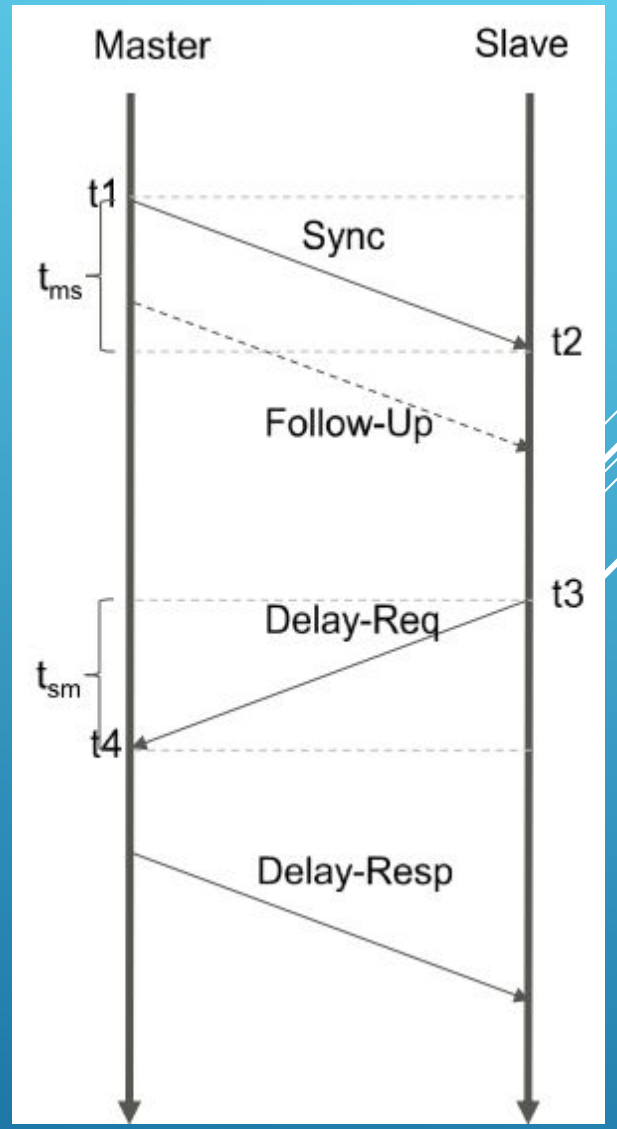
Example: fsock_gva_simple_recv.exe with Zero Load

- Dropless
- Zero Copy
- Data (Images) in Memory Placement
- Single Digit CPU load



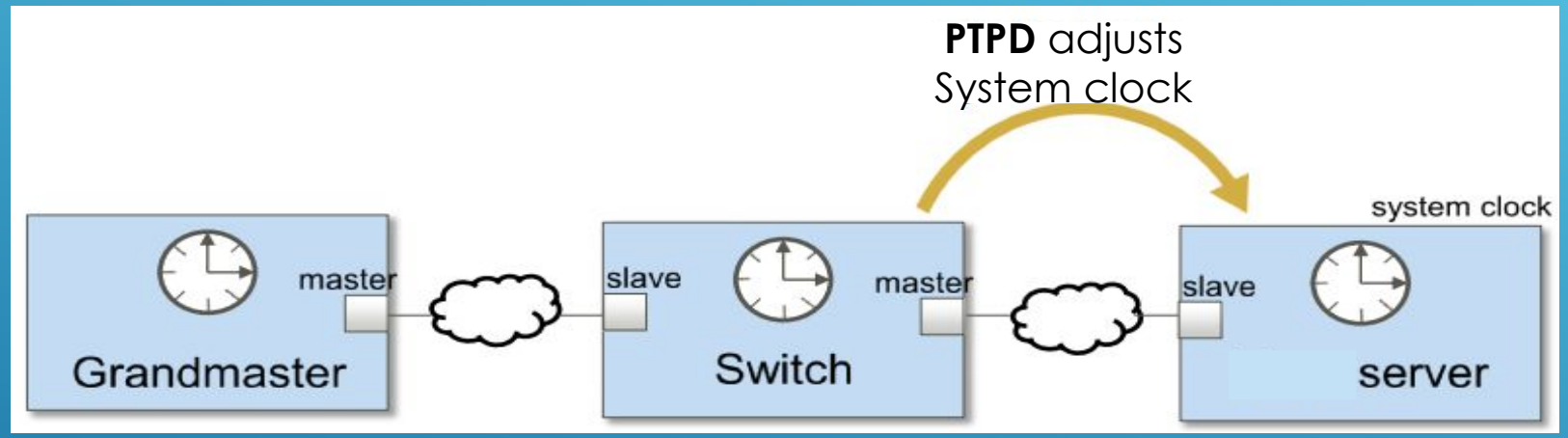
LibFSOCK PTP Solution with Hardware based Timestamps for best accuracy

- PTP works by using a two-way exchange of timing messages, known as "event messages"
- Time offset between master and slave clocks is calculated based on timestamps at packet sending and receiving
  - $offset = t2 - t1 - 1\ 2\ tms + tsm$
    $= t2 - t1 - 1\ 2\ \{\ t4 - t1 - t3 - t2\ \}$
- Packet timestamp accuracy is important for PTP



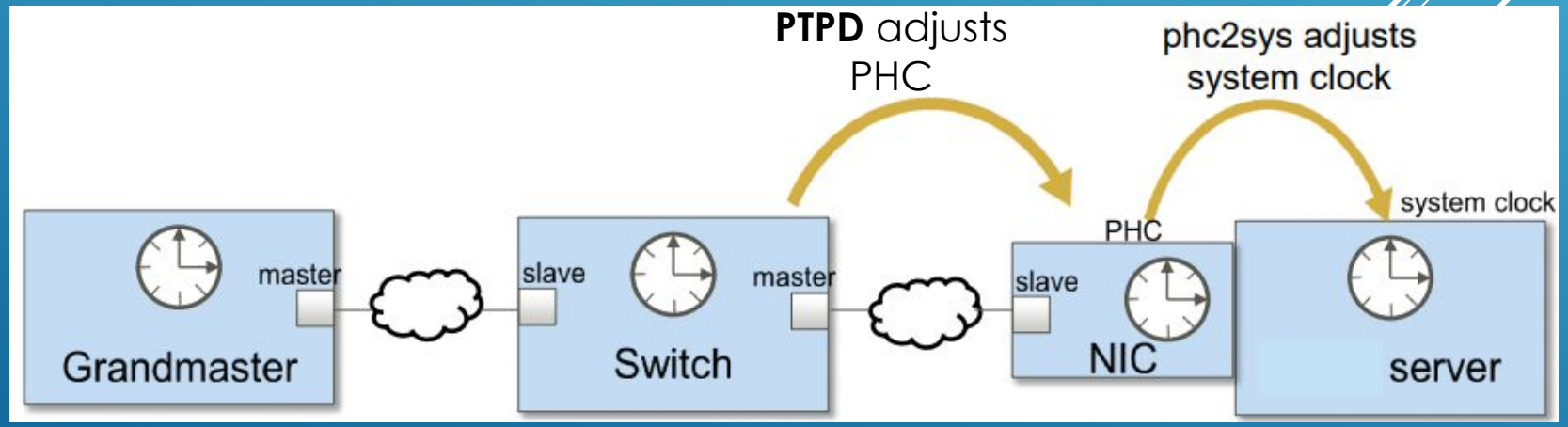
LibFSOCK PTP Solution with Hardware based Timestamps for best accuracy

- HW Timestamping allows for best offset synchronization



Legacy Approach achieves microsecond accuracy only

Approach with oscillators and hardware timestamps for nanosecond accuracy



LIBFSOCK FOR KERNEL BYPASS

LibfSock: NIC agnostic Sockets a-like API (Linux and Windows)

- **fsock_init()** – initialize library
- **fsock_open()** -- create an endpoint, can multiplex several multicasts
- **fsock_bind()** – allocate ports
- **fsock_connect(), fsock_accept()** – establish connection
- **fsock_recv(), fsock_send()** – send and receive data
- **fsock_close()** – close endpoint

Aspects and Concepts :

- Focus on fast RX (HFT/MEDIA/SECURITY), fast TX (HFT)
- Provide additional functionality like RX and TX timestamps
- Arista Switch Timestamping support (HFT)

LIBFSOCK – GVA : SIMPLE RECEIVER

```
/* init fsock lib */
rc = fsock_init(FSOCK_VERSION_API);
if (rc)
  printf("Could not init libfsock", rc);

sin.sin_family = AF_INET;
sin.sin_addr.s_addr = inet_addr(local_addr);
sin.sin_port = port;

rc = fsock_open(&sin.sin_addr, FSOCK_GVA, &gva_dev);
if (rc)
    printf("Could not open a libfsock GVA device", rc);

/* request alternate RING BUF SIZE (default 128MB) */
rc = fsock_set_ep_attribute(gva_dev, FSOCK_RINGBUF_SIZE, NULL, 0);
if (rc)
    printf("Could not set ring buf size", rc);

rc = fsock_bind(gva_dev, FSOCK_BIND_DEFAULT, port, NULL, &gva_chan);
if (rc)
    printf("Could not open a libfsock GVA Channel (stream)", rc);

/* open an mcast stream */
if (mcast) {
    struct ip_mreq mreq;
    mreq.imr_interface.s_addr = 0;
    mreq.imr_multiaddr = maddr.sin_addr;

    rc = fsock_setopt(gva_chan, FSOCK_JOIN_MCAST, &mreq, sizeof(mreq));
}
```

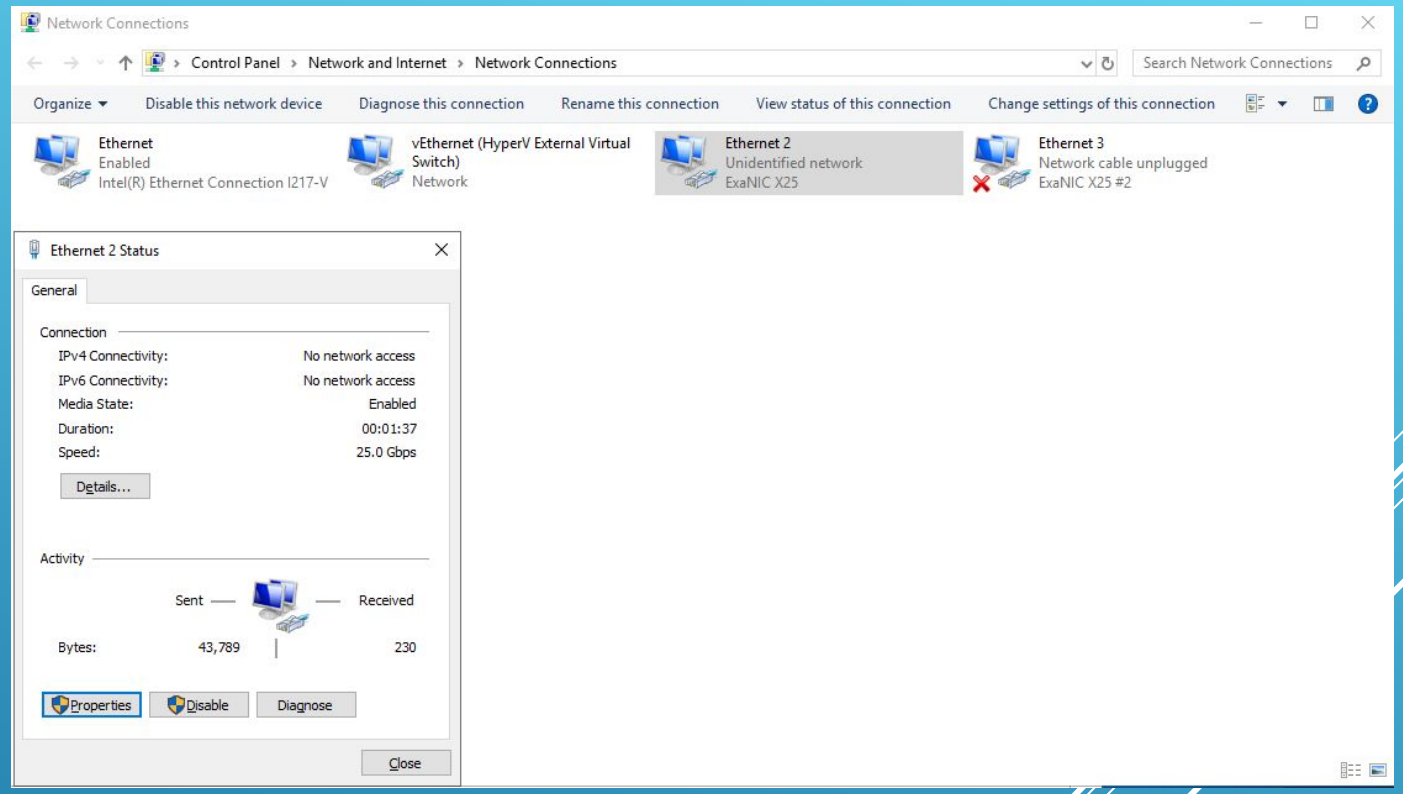
```
 fsock_gva_alloc_type_t gva_alloc;
gva_alloc.bufsize = 128;
gva_alloc.num = num_bufs;
rc = fsock_setopt(gva_chan, FSOCK_GVA_ALLOC, &gva_alloc, sizeof(gva_alloc));

printf("Allocated and Queued %u buffers\n", num_bufs);

/* timeout */
if (block_timeout)
   rc = fsock_setopt(gva_chan, FSOCK_RX_TIMEOUT, &block_timeout, sizeof(block_timeout));

while (blocks_recvd < iters) {
  /* wait for data blocks */
  rc = fsock_recv(gva_chan, FSOCK_RECV_DEFAULT, buf, sizeof(buf), &rxinfo);

  rc = fsock_recv(gva_chan, FSOCK_RECV_DEFAULT | FSOCK_RECV_GVABLOCK_INPLACE, &gva_blk, sizeof(gva_blk), &rxinfo);

  rc = fsock_recv(gva_chan, FSOCK_RECV_NONBLOCK | FSOCK_RECV_GVABLOCK_INPLACE, &gva_blk, sizeof(gva_blk), &rxinfo);
  if (rc != 0) { /* check for errors */ }

  printf ("Received data block %u, type:%u, len:%u, status=0x%x",
          gva_blk.block_id, gva_blk.payload_type, gva_blk.payload_length, gva_blk.status);
  blocks_recvd++;
  /* requeue this block */
  rc = fsock_setopt(gva_chan, FSOCK_GVA_QUEUE_BUF, &gva_blk.gva_buf, sizeof(gva_buf_t));
  if (rc != 0) { /* check for errors */ }
} /* while */
```

MULTIPLE (GENERIC) VENDOR SUPPORT

- Mellanox / Nvidia – ConnectX 5,6,7
  - Allows for Linux/Windows Support
  - GPU Direct Support
  - 1,10GbE – 100GbE
- Solarflare – X2552
  - Windows (libfsock-XIO) , Linux
- Exablaze / Cisco – ExaNIC X10, X25, X40
  - Linux and Windows
  - 10/25/40 GbE support
  - Best in class Windows Solution
- Broadcom, Intel, … (libfsock-XIO)

FAST SOCKETS FOR EXANICS:

- ExaDriver for Windows
- Creates Network Environment known from standard Ethernet NICs
  - NDIS 6.X, Windows Server and Clients
  - Additional Kernel Bypass
  - IP connectivity
    - Ping is still your friend
    - Use CIFS, DHCP, …
- VLAN support
- Teaming support, multiple bandwidth , HA, fault tolerance !
- Lowest Latency as of today
- PTP support – The first hardware timestamps based PTP Solution on Windows



Windows Server 2016

Windows 10

Windows Server 2019

Windows Server 2012 R2

Windows Server 2008

FastSockets Types

- libfsock-exa
  - native driver and library for ExaNICs
- libfsock-mlx
  - native driver and library for ConnectX
- libfsock-xio
  - driver and library for any vendor

NIC BENEFITS

	Acceleration TX	Acceleration RX	Tick To Trade/ Low Latency	MEDIA EXTENSIONS	OS
Cisco, e.g ExaNIC X10/25	UDP: YES TCP: YES MODUS: PIO	UDP: YES TCP: YES MODUS: DMA	SUITED 720ns t2t	NO	Windows, Linux
MLX, e.g Connect-X5,6,7	UDP: YES TCP: NO MODUS: DMA	UDP: YES TCP: NO MODUS: DMA	NOT SUITED, large RX RingBuffers. **Suited via XIO Proxy**	YES: GVA	Windows, Linux
Solarflare	UDP: YES TCP: YES MODUS: PIO/DMA	UDP: YES TCP: YES MODUS: PIO/DMA	SUITED 800ns t2t	NO	LINUX
Intel/Solarflare/ Broadcom/MLX/ NVIDIA [**XIO**]	UDP: Yes TCP: Yes MODUS:PIO	UDP: YES TCP: Yes MODUS: DMA	SUITED, large RX RingBuffers	YES: GVA	**Cloud** / Windows/ Linux (v2)

Questions ?