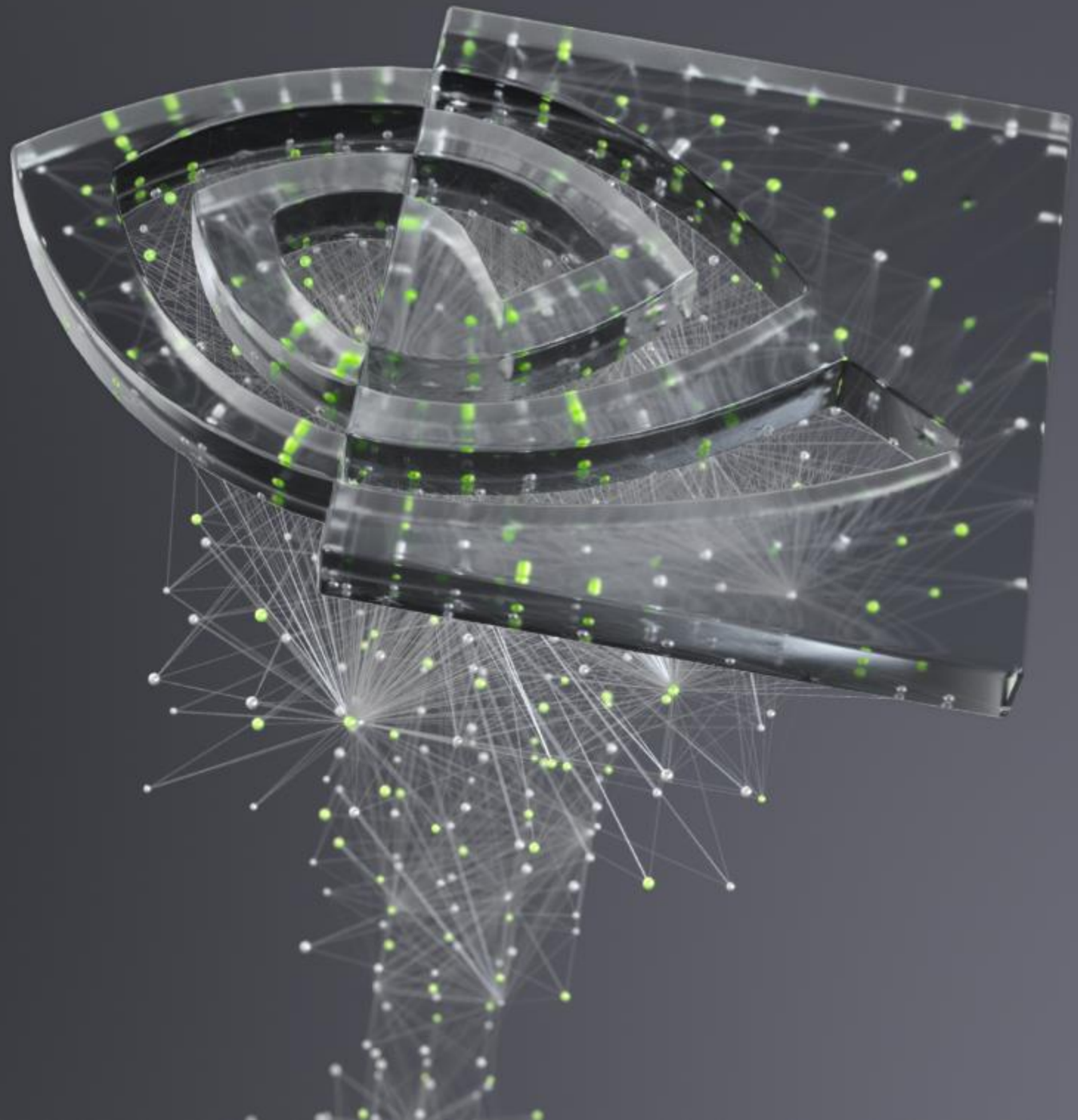
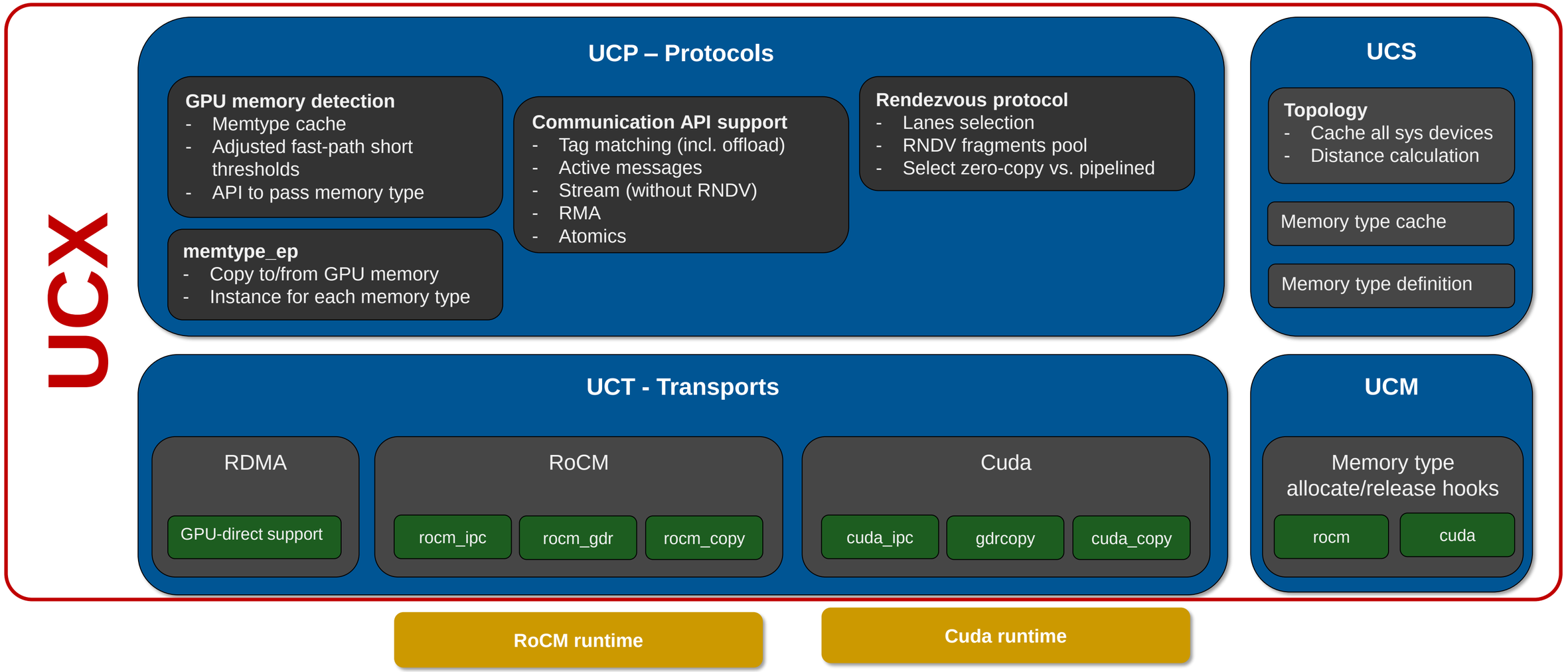


UCX GPU SUPPORT

UCX vF2F 2021

UCX GPU SUPPORT MAP



WHAT'S NEW IN 2021

- Proto_v2 : Pipeline, Put/Get/Atomic with GPU memory + ucx_perftest
- More accurate topology detection based on sysfs walk
- Cuda driver hooks based on binary parser – supports static link!
- Global memtype cache
- Select cuda stream by memory type
- Auto-register whole allocation
- Select Cuda-IPC based on NVLINK topology
- RNDV fragment pool per device

COMING IN 2022

- Device memory pipeline (for Cuda-IPC)
- Intra-node 2-stage pipeline (based on map-shared host buffers)
    - May require adding remote memory cache for UCP endpoint
- DMA-buf support
- Tuning out-of-box proto_v2 performance
    - Eager/rndv threshold
    - Select pipeline vs. GDR based on topology
- Better multi-context support

DEVICE MEMORY PIPELINE

- Use cases:
  - Processes on same node exchanging Cuda-managed memory; have NVLink
  - Processes on different nodes exchanging Cuda memory over RDMA; GPU-direct is fast;

    User buffer registration is too expensive, or not possible because BAR is small
- Solution approach:
  - Rndv fragment memory pools on every system device (initialize on demand)
  - Protocols select device memory according to user buffer locality
- Challenges:
  - The receiver needs to select a rdnv fragment that is Nvlink-reachable from the sender, or host fragment if not reachable

INTRA-NODE 2-STAGE

- Use case: Processes on same machine exchanging Cuda memory; no NVLink
- Current protocol is:
  - send_buffer -> cudaMempy -> rdnv_frag -> knem/cma/xpmem copy -> rndv_frag -> cudaMemcpy -> recv_buffer
- A better protocol would be:
  - send_buffer -> cudaMemcpy -> shared_rdnv_frag -> cudaMemcpy -> recv_buffer
- Challenges:
  - Manage and reuse shared rndv fragments
  - Cache remotely-mapped fragments per endpoint
- Current plan:
  - Create remote regions hast per UCP ep, to also replace the cache on xpmem

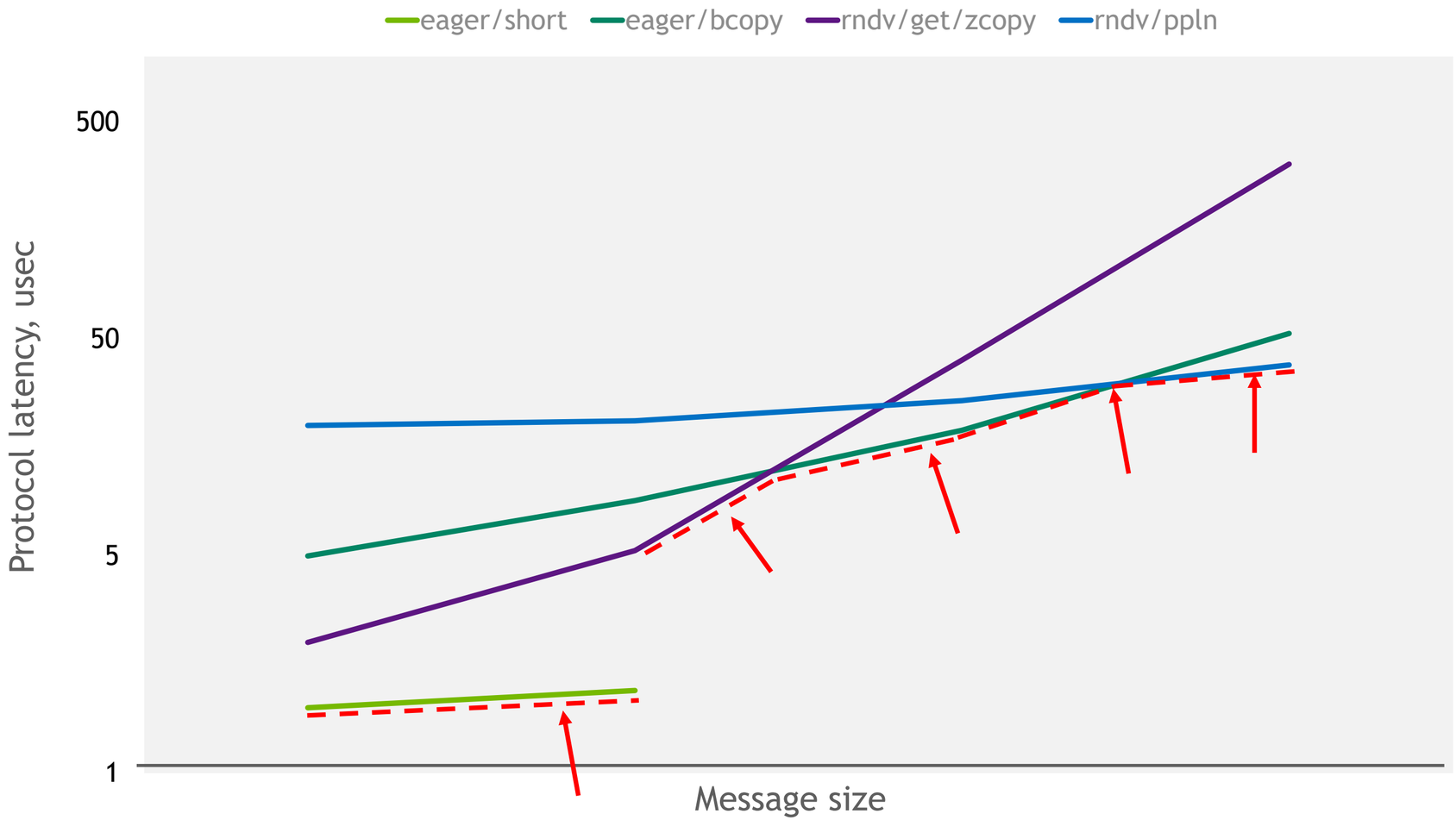
DMA-BUF SUPPORT

- GPU transport to expose querying a memory region’s file descriptor
- RDMA transport to support registering DMA-buf memory with a file descriptor
- UCP to get the file descriptor from GPU, pass to RDMA
    - Close the file descriptor right after, to not exhaust fd space

MOVE REGISTRATION CACHE TO UCP

- Pro
  - DMAbuf-friendly – no need to get fd from GPU if cache is hit
  - Reduce SW overheads for zcopy path
  - Allows buffer-id tracking – since memory detect and memory reg are on different MDs
  - Allows future optimization: Use bcopy initially, and pin memory after N uses
- Con
  - UCT user will have to create registration cache to keep same level of performance
    - Mitigation: keep regcache in UCT as well, at least for a while

PROTOCOLS CUTOFF

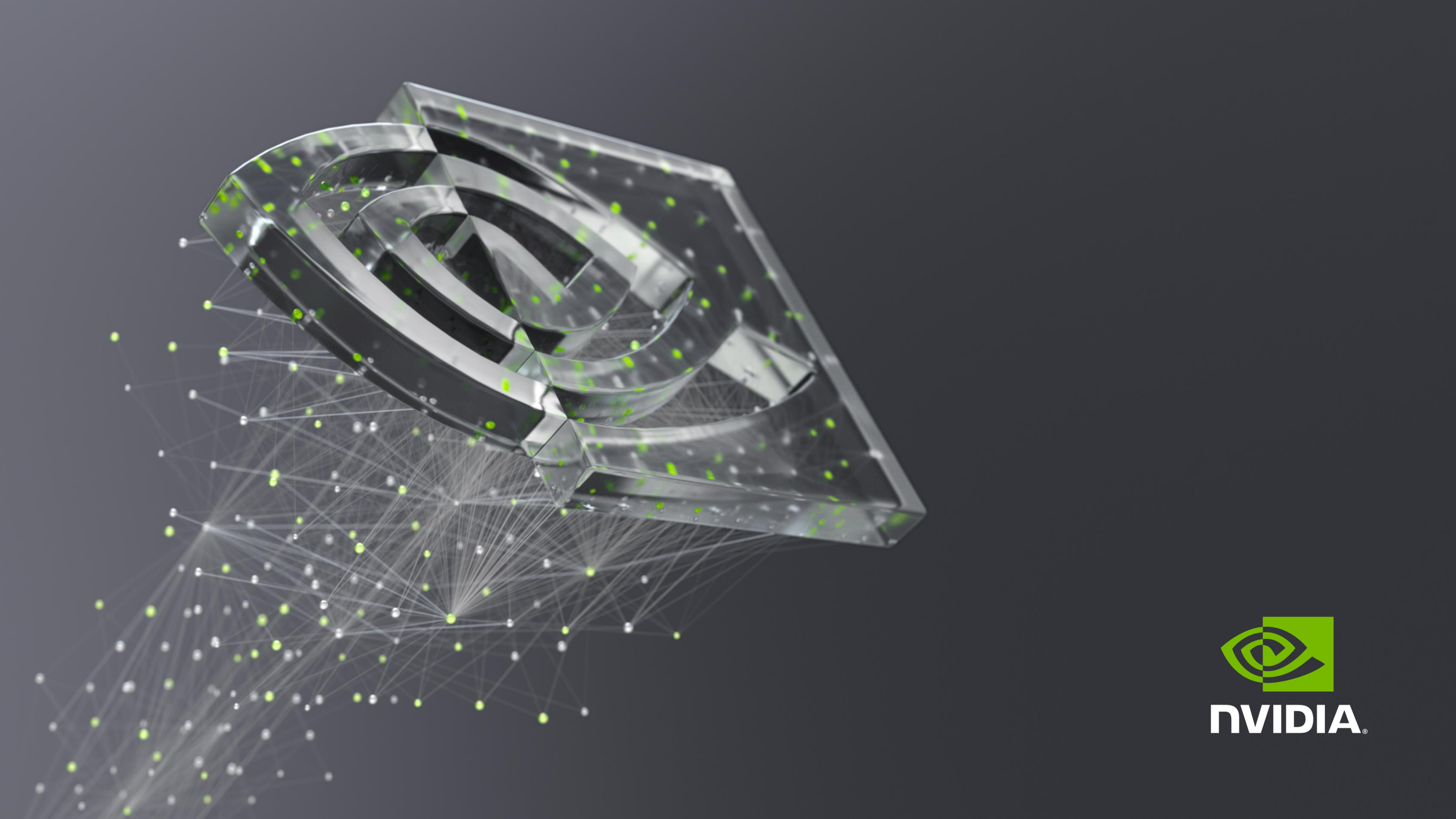


PROTO_V2 PLAN

What	Status
Infrastructure	Done
Tag match protocols	Done
Put/get/atomics	Done
Rendezvous pipeline	Done
Topology awareness	Done
Active messages	WIP for v1.13
Stream	Not started
Client/server connection	WIP for v1.13
Error flows	Not started
Tune out-of-box performance	WIP, continuous effort

PROTO_V2 NEXT STEPS

- Implement all API with new protocols
- Remove exiting protocol and ep config code
- Rendezvous protocol with IOV list
- Protocol versions and wire compatibility
- Detailed traces of “why” specific protocol was selected
- Fine tune performance estimation model
- Topology “injection”


NVIDIA