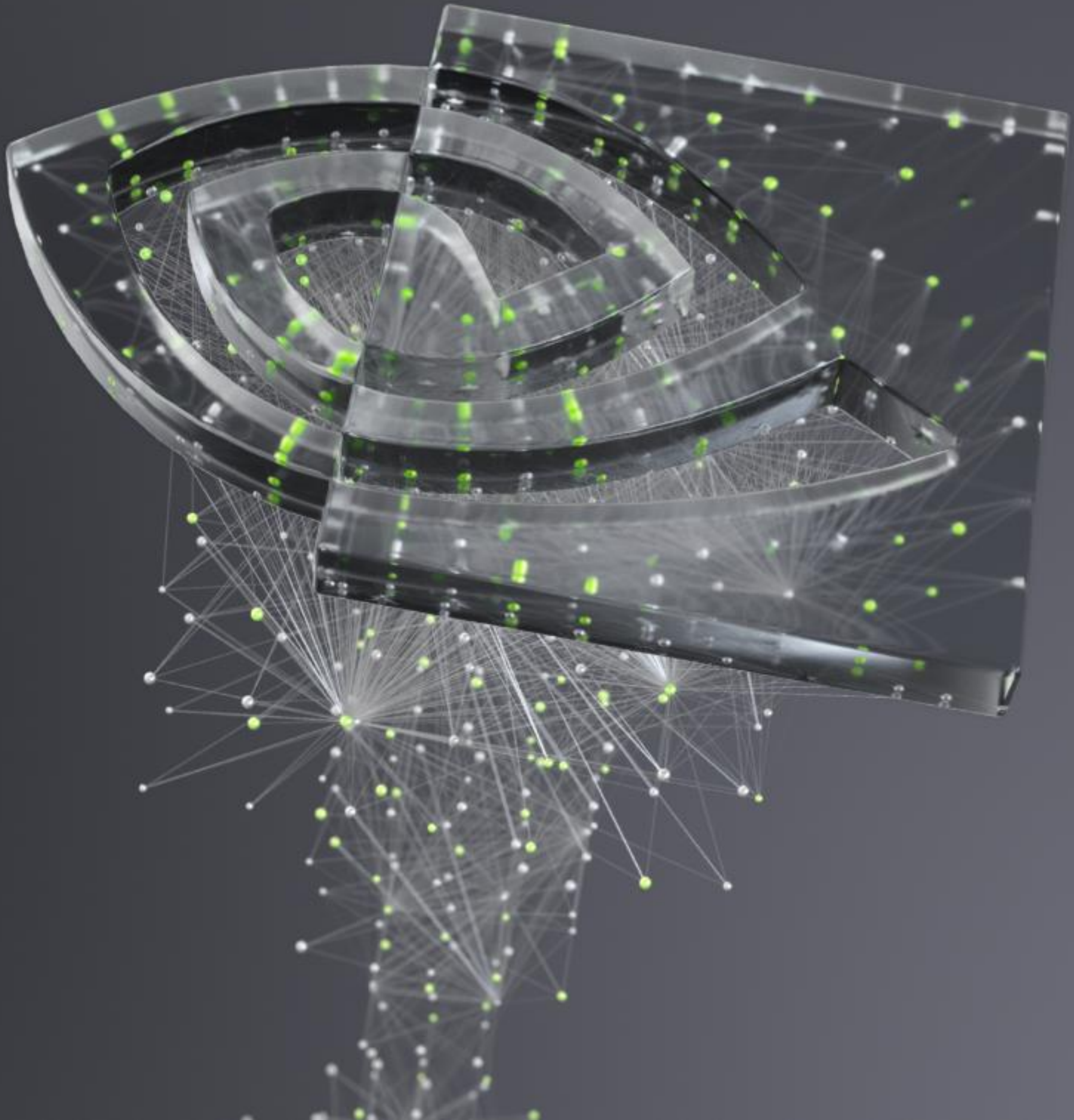


UCX PROTOCOLS

Yossi Itigin, UCF workshop 2021

ISSUES WITH CURRENT UCP PROTOCOLS

- Scattered and complicated logic for protocol and thresholds selection
- No support for protocol selection per memory type / locality
- For non-inline case: many data-path checks for message size, datatype, memory type
- Bad handling of endpoint configuration change while send operation is inflight
- Incomplete handling of “aborting” send requests in case of endpoint error
- Can’t reuse common code (e.g multi-rail) between protocols

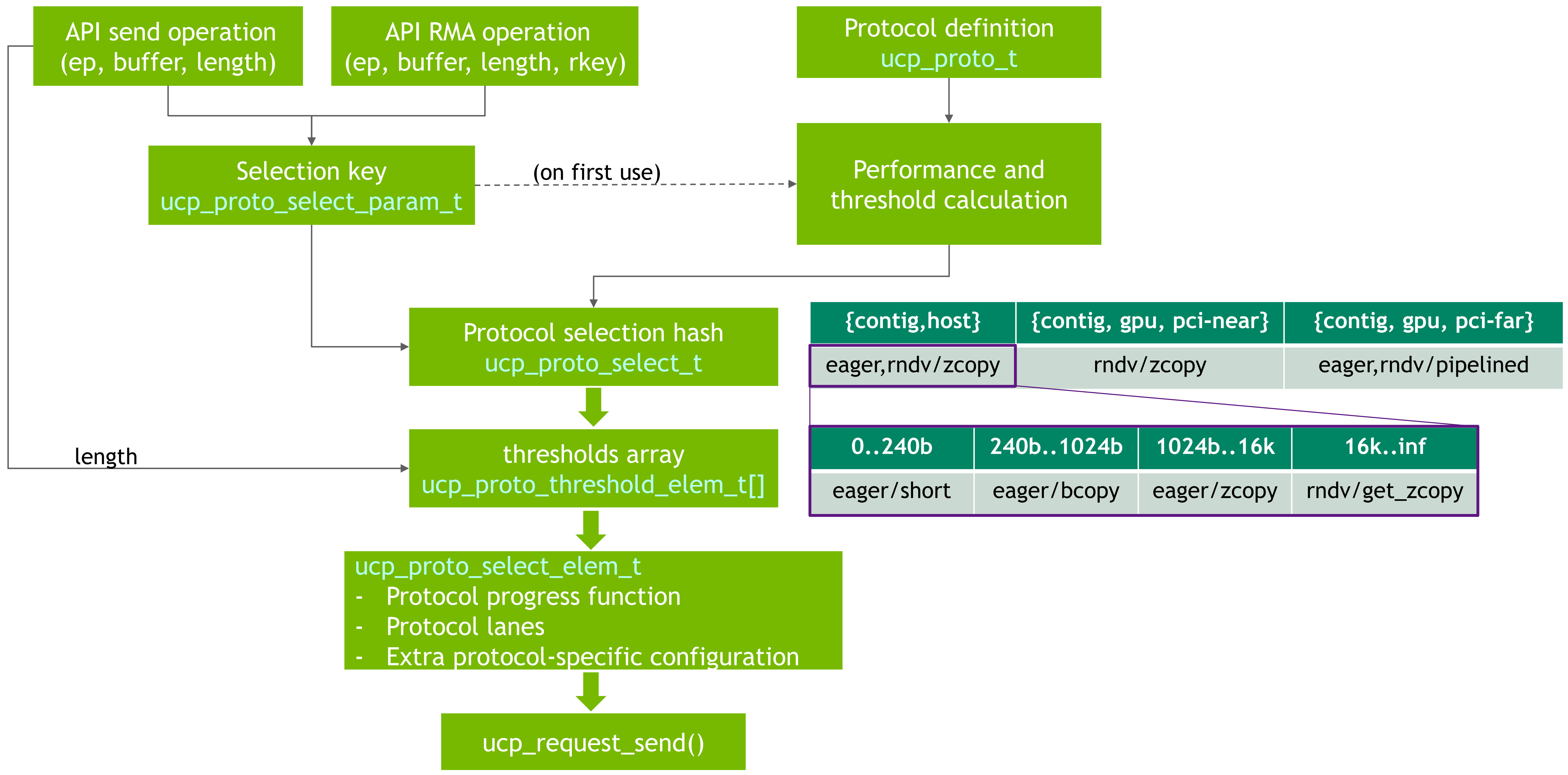
ISSUES WITH CURRENT UCP PROTOCOLS

- Scattered and complicated logic for protocol and thresholds selection
- No support for protocol selection per memory type / locality
- For non-inline case: many data-path checks for message size, datatype, memory type
- Bad handling of endpoint configuration change while send operation is inflight
- Incomplete handling of “aborting” send requests in case of endpoint error
- Can’t reuse common code (e.g multi-rail) between protocols

SOLUTION APPROACH

- Separate protocol definition from protocol selection engine
- Generate protocol cutoff values in a generic way
- Create a “protocol selection key” based on operation properties:
    - Operation, datatype, memory type, memory locality, extra flags
- UCP endpoint and R-key point to protocol selection hash table
    - Similar endpoints/rkeys share the table
- Protocol hash table entries are initialized on first use
- A send operation creates selection key, finds protocol in the hash, and starts sending

DATAFLOW



API send operation (ep, buffer, length)
API RMA operation (ep, buffer, length, rkey)
Protocol definition ucp_proto_t
Selection key ucp_proto_select_param_t
(on first use)
Performance and threshold calculation
Protocol selection hash ucp_proto_select_t
length
thresholds array ucp_proto_threshold_elem_t[]
ucp_proto_select_elem_t
- Protocol progress function
- Protocol lanes
- Extra protocol-specific configuration
ucp_request_send()

{contig,host}	{contig, gpu, pci-near}	{contig, gpu, pci-far}
eager,rndv/zcopy	rndv/zcopy	eager,rndv/pipelined

0..240b	240b..1024b	1024b..16k	16k..inf
eager/short	eager/bcopy	eager/zcopy	rndv/get_zcopy

MAKING A PROTOCOL

- Protocol definition:

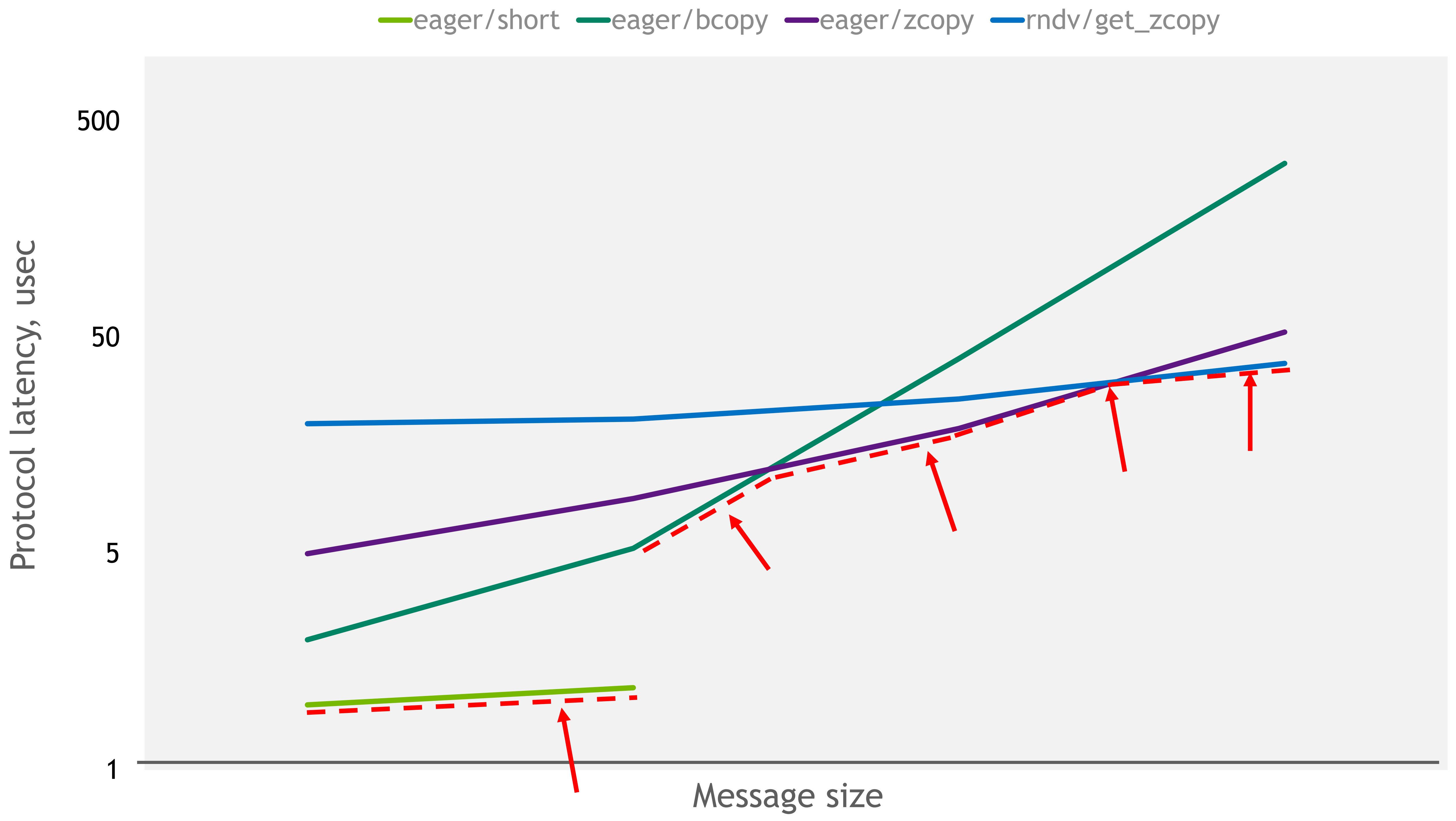
```
struct ucp_proto {
    const char                     *name;      /* Protocol name */
    unsigned                       flags;      /* Protocol flags for special handling */
    ucp_proto_init_func_t          init;       /* Initialization function */
    ucp_proto_config_str_func_t    config_str; /* Configuration dump function */
    uct_pending_callback_t         progress;   /* UCT progress function */
};
UCP_PROTO_REGISTER(&my_proto)
```

- Protocol init() function is called for every new key (=op,dtype,..) :
    - Returns the estimated performance for every message range, or ERR_UNSUPPORTED if cannot run
    - Initializes protocol’s “private data” configuration space
- Define “progress” function to send a request, given that all request fields and the “private data” are set

PROTOCOLS CUTOFF

- Common protocol logic combines the results of init() calls for all protocols
- Select best available protocol for each interval by using linear function intersect
- The performance of a protocol is “time to send” as function of message size
- Find the best protocol for every “interval” by walking on the linear intersections

PROTOCOLS CUTOFF



SEND PROGRESS

- API calls select a protocol and initialize send request fields (e.g tag)
- Protocols define progress functions which use these fields to perform UCT send operations
- New set of common inline functions for multi-rail, rkey resolve, fragmentation, ...
- Protocol responsible for calling completion callback and releasing the request

IMPLEMENTATION STATUS

Done for v1.10:

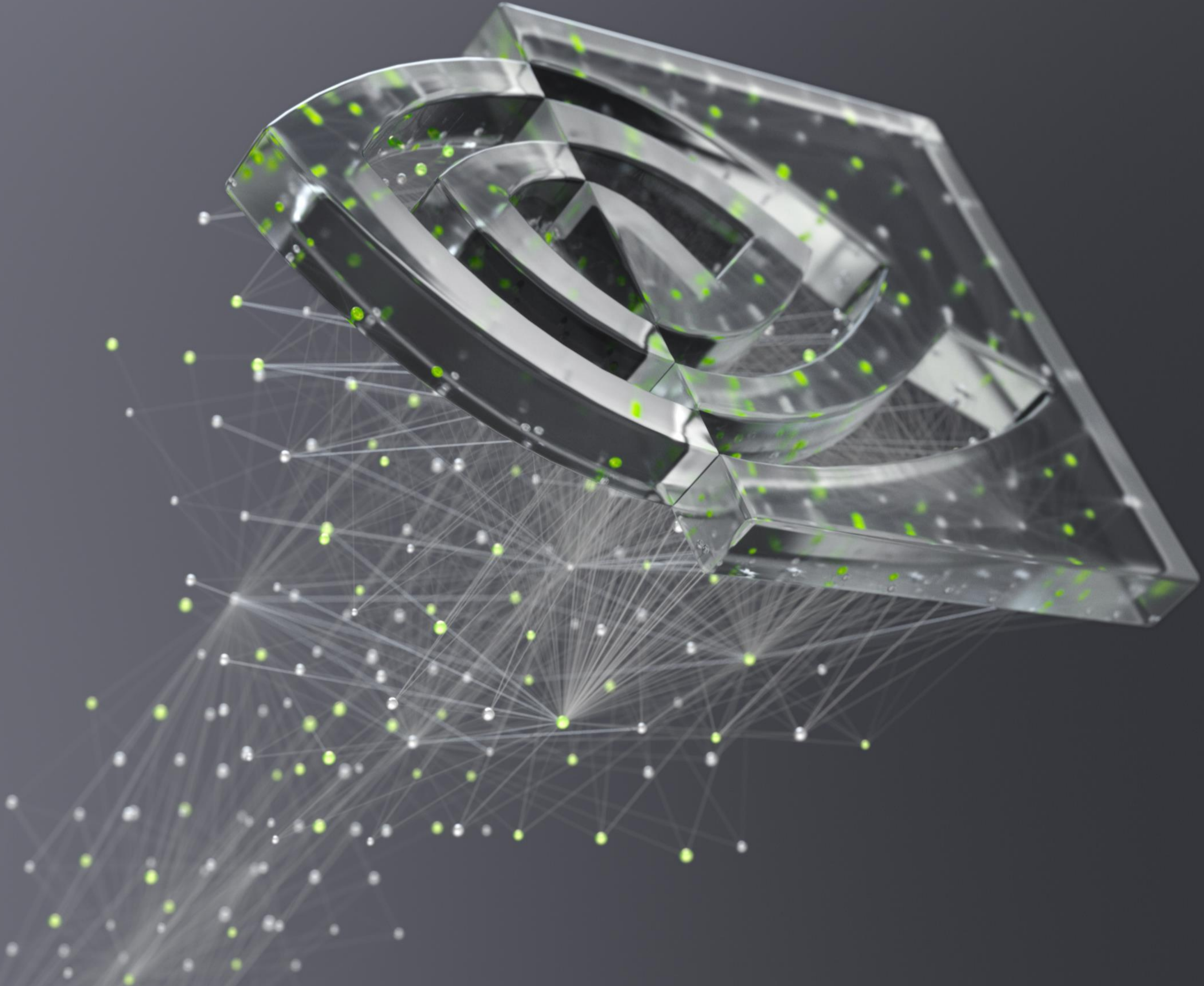
- Protocols common infrastructure
- Eager and RMA protocols with basic GPU support
- Off by default, turn on by UCX_PROTO_ENABLE=y

Planned for v1.11:

- Rendezvous protocols
- GPU pipelined
- Active messages

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
- Fine tune performance estimation model



NVIDIA®