ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

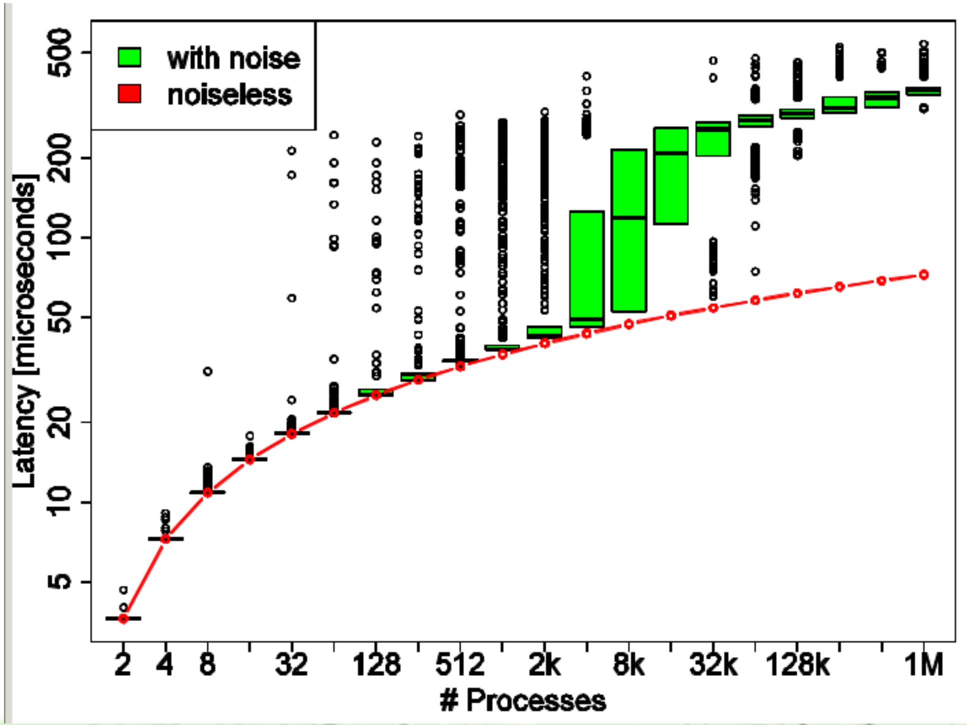
Protocols for Fully Offloaded Collective Operations on Accelerated Network Adapters

Timo Schneider, Torsten Hoefler,

Ryan E. Grant, Brian W. Barrett, Ron Brightwell

WHY (COLLECTIVE) OFFLOAD

- Blocking Collectives:
  - During communication CPU is idle
  - If one process is delayed, all have to wait (OS Noise)



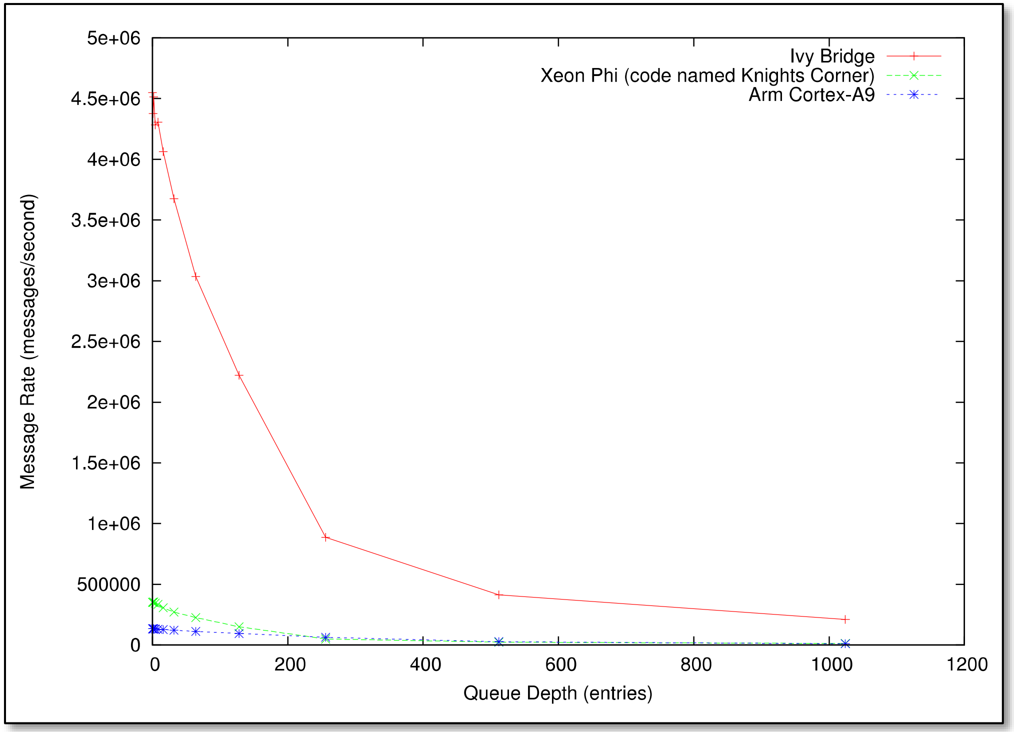
See: Hoefler et. al.: "Characterizing the Influence of System Noise on Large-Scale Applications by Simulation", SC'10

WHY (COLLECTIVE) OFFLOAD

- Blocking Collectives:
  - During communication CPU is idle
  - If one process is delayed, all have to wait
- Solution: Non-blocking collectives!
  - If the CPU does computations, who ensures progress?

WHY OFFLOAD

- Dedicate cores to communication?



See: Barrett et al.: The impact of hybrid-core processors on MPI message rate, EuroMPI'13

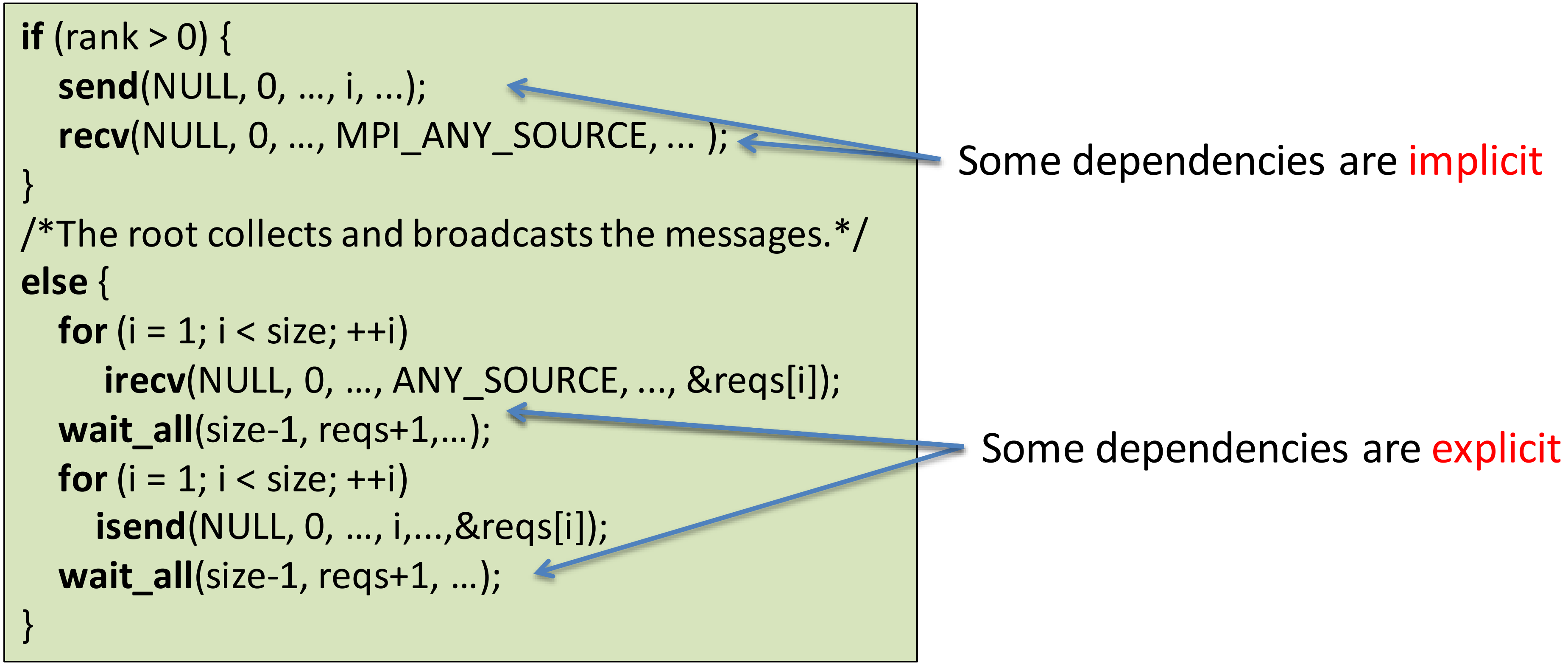
- Speed: CPU <-> NIC can be up to 500ns!

How do Collectives Look Like?

```
if (rank > 0) {
    send(NULL, 0, …, i, ...);
    recv(NULL, 0, …, MPI_ANY_SOURCE, ... );
}
/*The root collects and broadcasts the messages.*/
else {
    for (i = 1; i < size; ++i)
         irecv(NULL, 0, …, ANY_SOURCE, ..., &reqs[i]);
    wait_all(size-1, reqs+1,…);
    for (i = 1; i < size; ++i)
        isend(NULL, 0, …, i,...,&reqs[i]);
    wait_all(size-1, reqs+1, …);
}
```

Open MPI Barrier for small Communicator sizes

How do Collectives Look Like?



```
if (rank > 0) {
   send(NULL, 0, …, i, ...);
   recv(NULL, 0, …, MPI_ANY_SOURCE, ... );
}
/*The root collects and broadcasts the messages.*/
else {
   for (i = 1; i < size; ++i)
       irecv(NULL, 0, …, ANY_SOURCE, ..., &reqs[i]);
   wait_all(size-1, reqs+1,…);
   for (i = 1; i < size; ++i)
      isend(NULL, 0, …, i,...,&reqs[i]);
   wait_all(size-1, reqs+1, …);
}
```

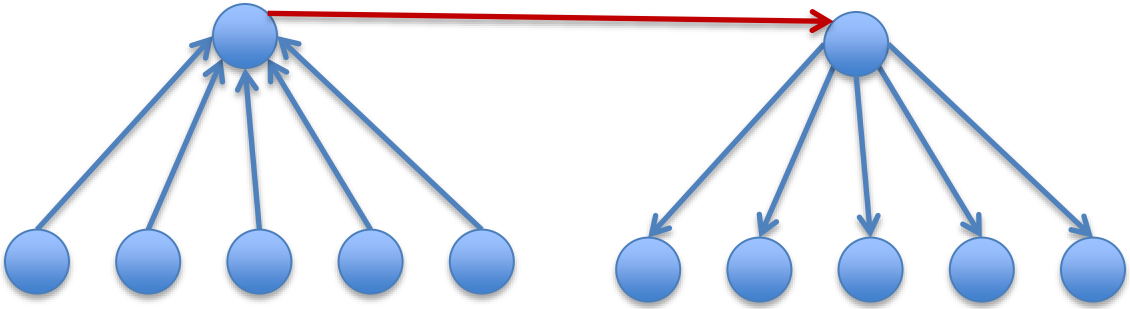
Some dependencies are implicit

Some dependencies are explicit

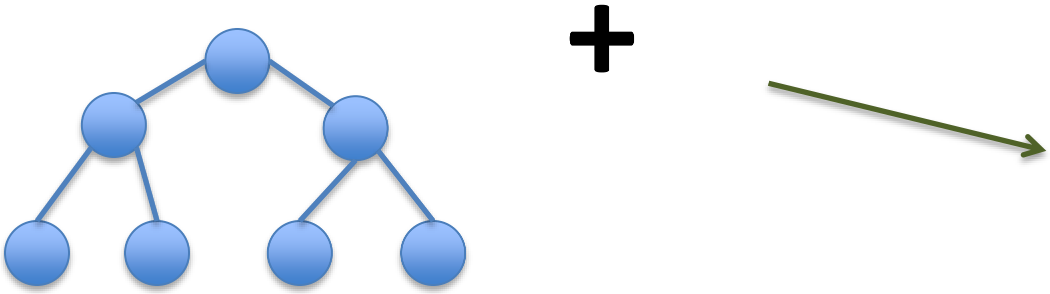
How do I translate this to Portals 4? ConnectX? Quadrics?

Not a good programming model for collective offload!

How Should Collectives Look Like



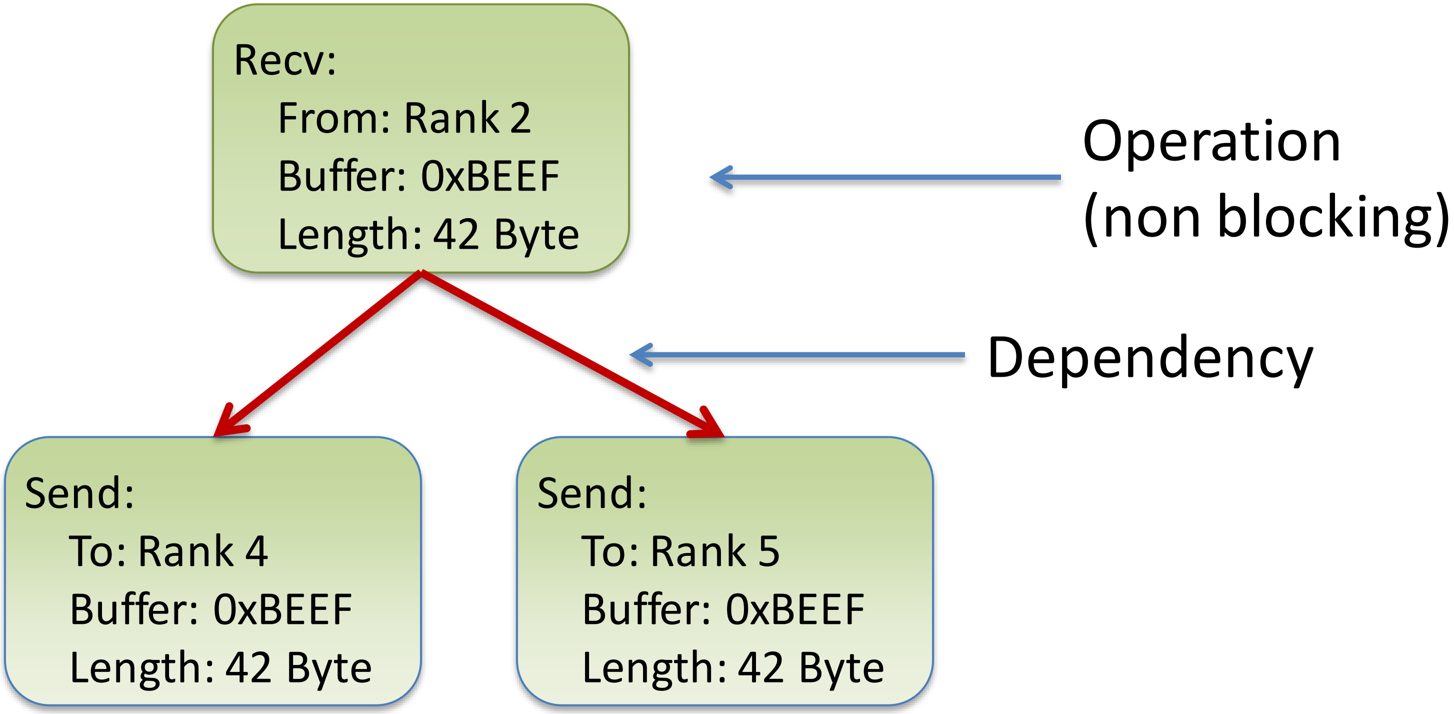
Data Movement
+
Dependencies



Communication Topology

"Schedule" of operations (and dependencies among them) for each rank

cDAG



Recv:
From: Rank 2
Buffer: 0xBEEF
Length: 42 Byte

Operation
(non blocking)

Dependency

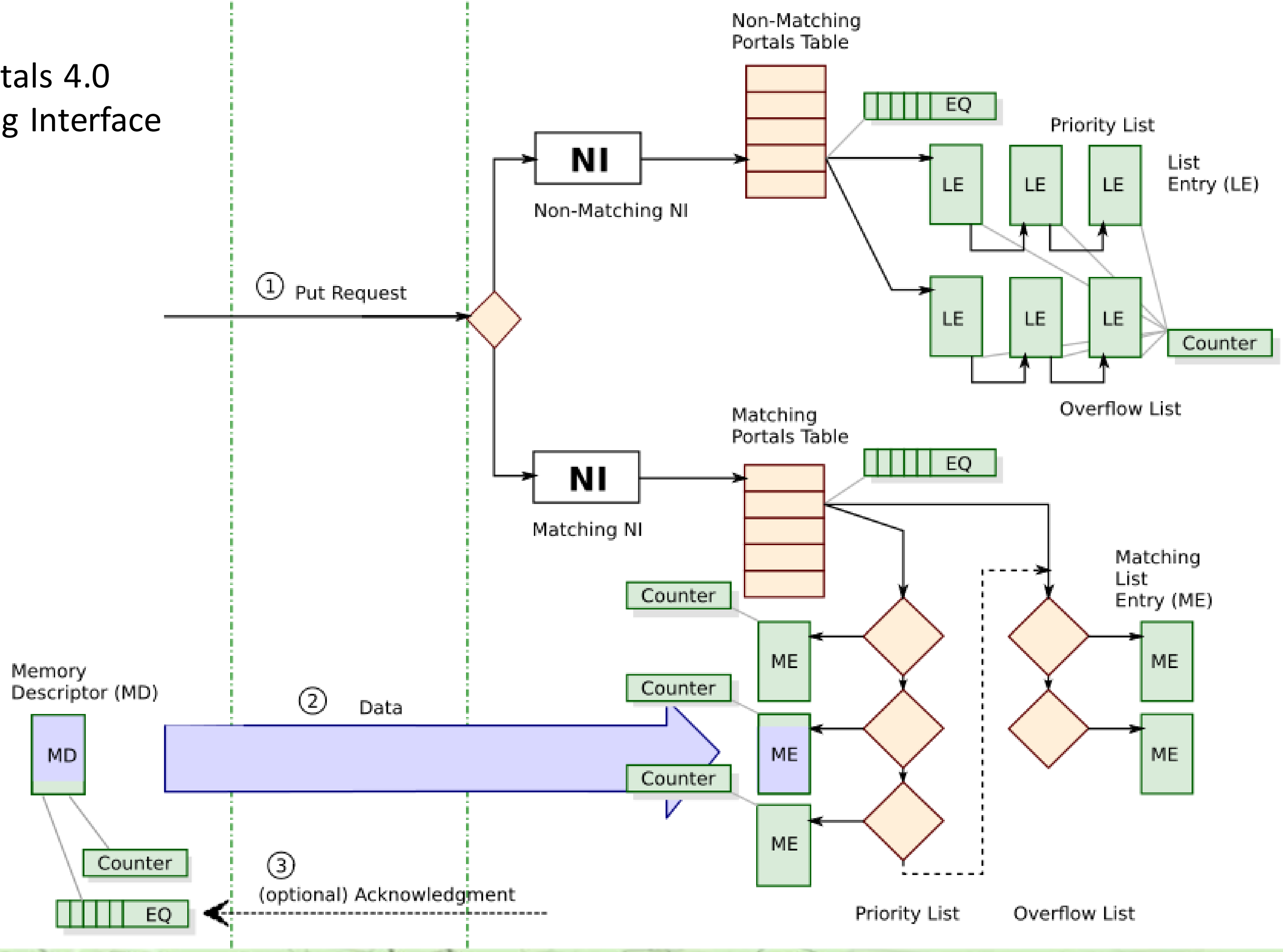
Send:
To: Rank 4
Buffer: 0xBEEF
Length: 42 Byte

Send:
To: Rank 5
Buffer: 0xBEEF
Length: 42 Byte

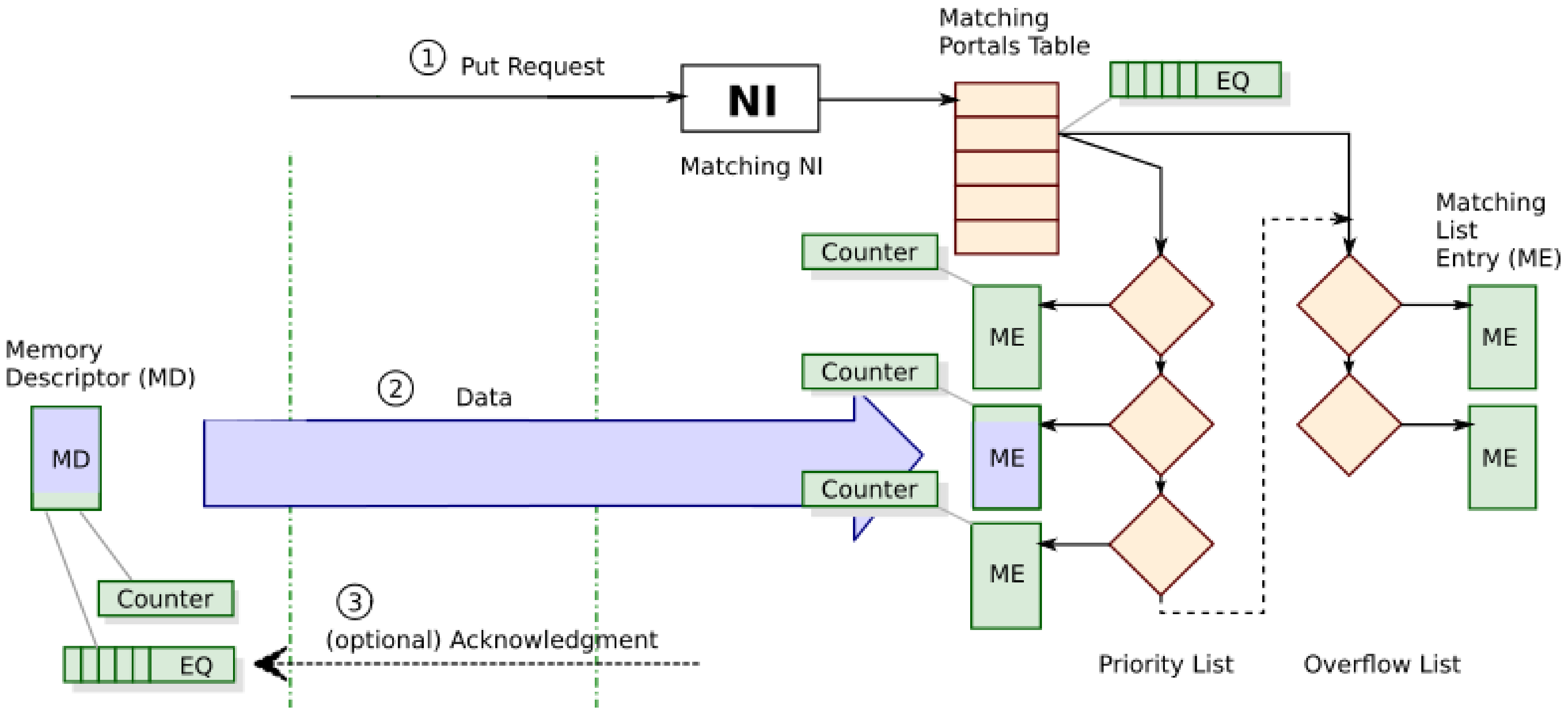
- Backends for cDAG: MPI, DMAPP, Portals 4 Triggered Operations

PORTALS 4

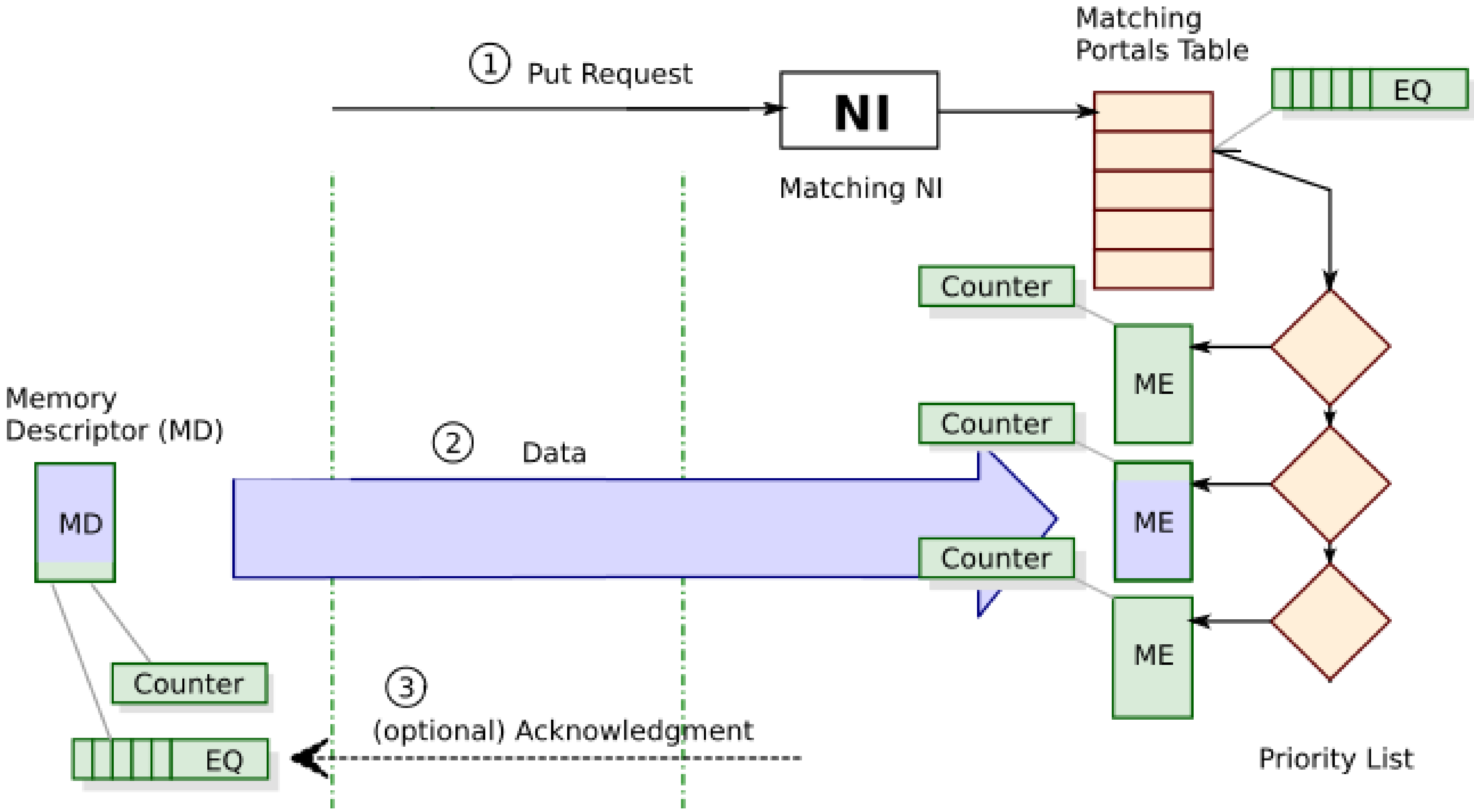
Source:
Barrett et al.: The Portals 4.0 Network Programming Interface



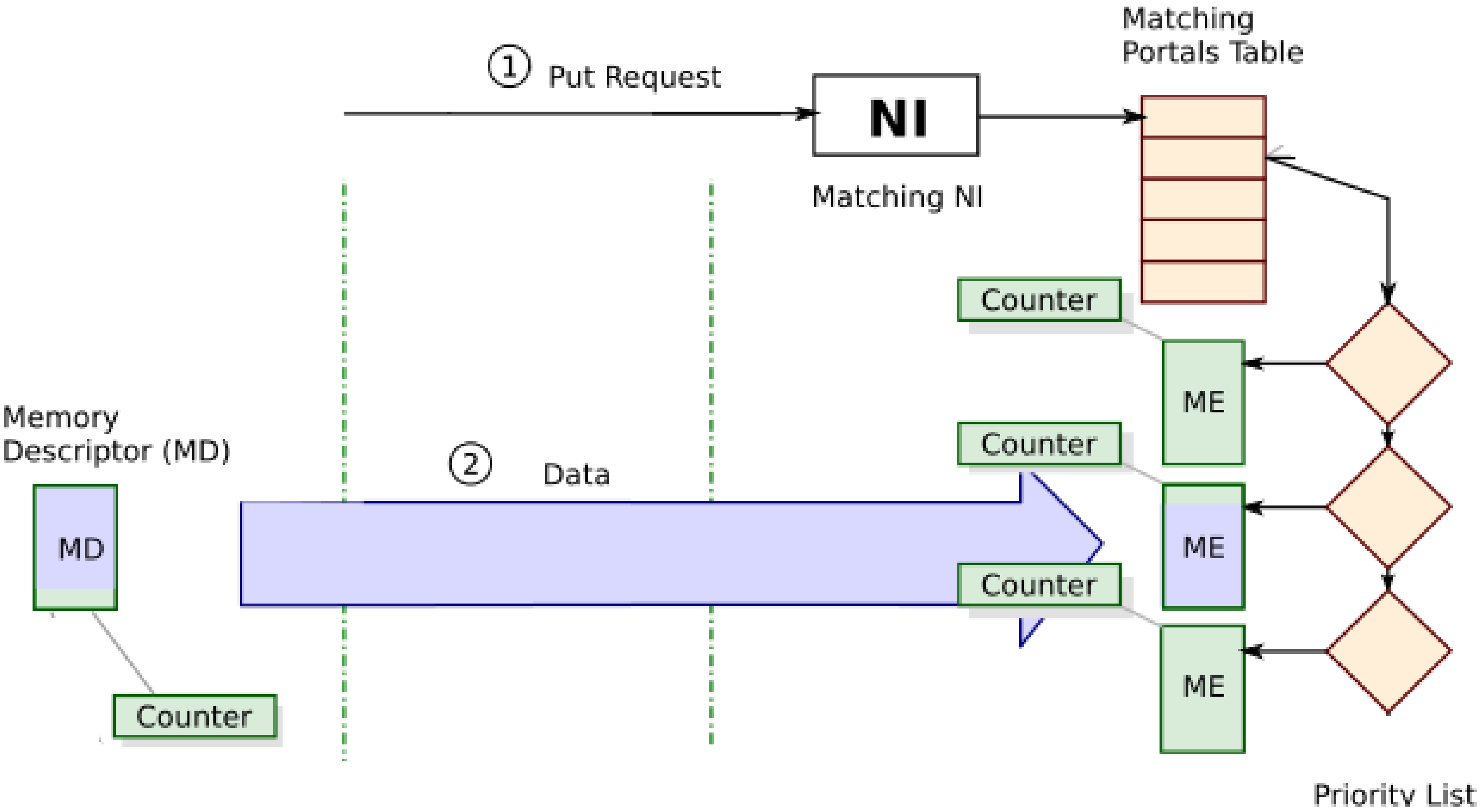
PORTALS 4


Matching
Portals Table
① Put Request
NI
Matching NI
EQ
Matching
List
Entry (ME)
Counter
ME
Counter
ME
Counter
ME
ME
ME
Memory
Descriptor (MD)
MD
② Data
Counter
EQ
③
(optional) Acknowledgment
Priority List
Overflow List

PORTALS 4

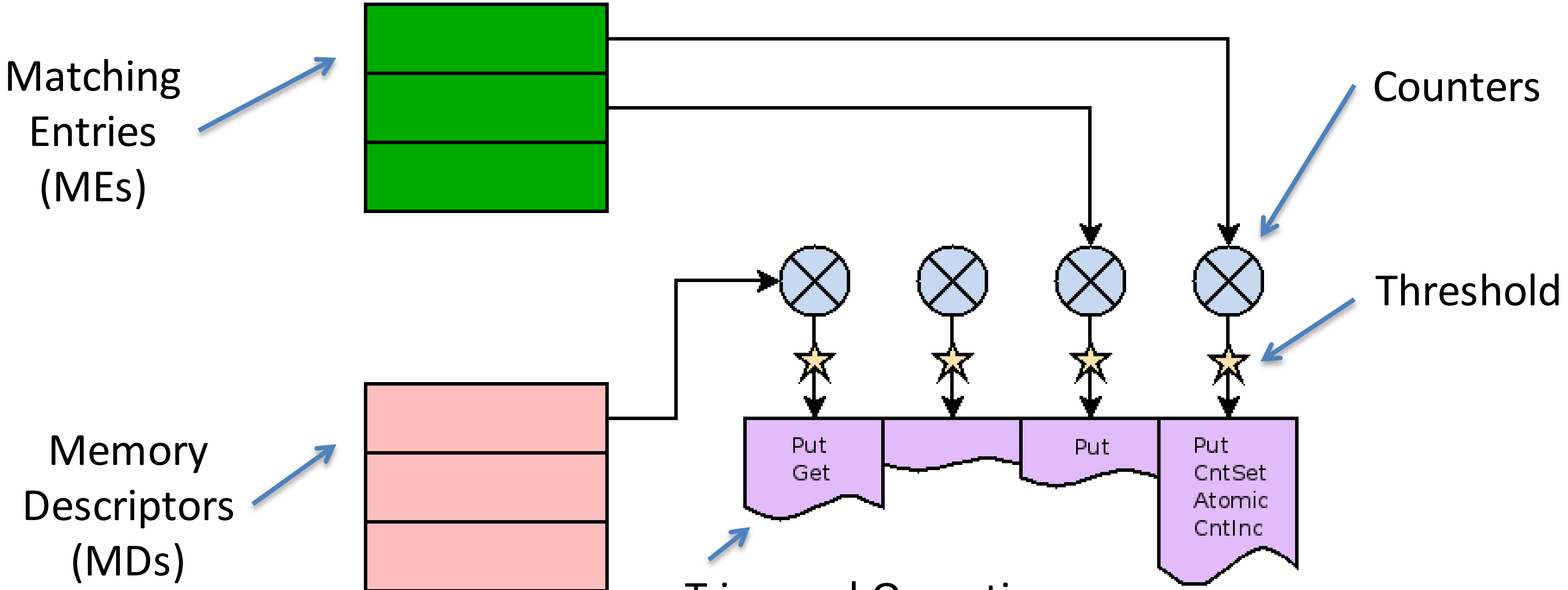

1 Put Request
NI
Matching NI
Matching Portals Table
EQ
Counter
ME
Counter
ME
Counter
ME
Memory Descriptor (MD)
MD
2 Data
Counter
EQ
3 (optional) Acknowledgment
Priority List

PORTALS 4


1 Put Request
NI
Matching NI
Matching Portals Table
Counter
ME
Counter
ME
Counter
ME
Memory Descriptor (MD)
MD
Counter
2 Data
Priority List

Portals 4 Triggered Operations

- During cDAG execution, the CPU must not be involved -> **full offload**



Matching Entries (MEs)

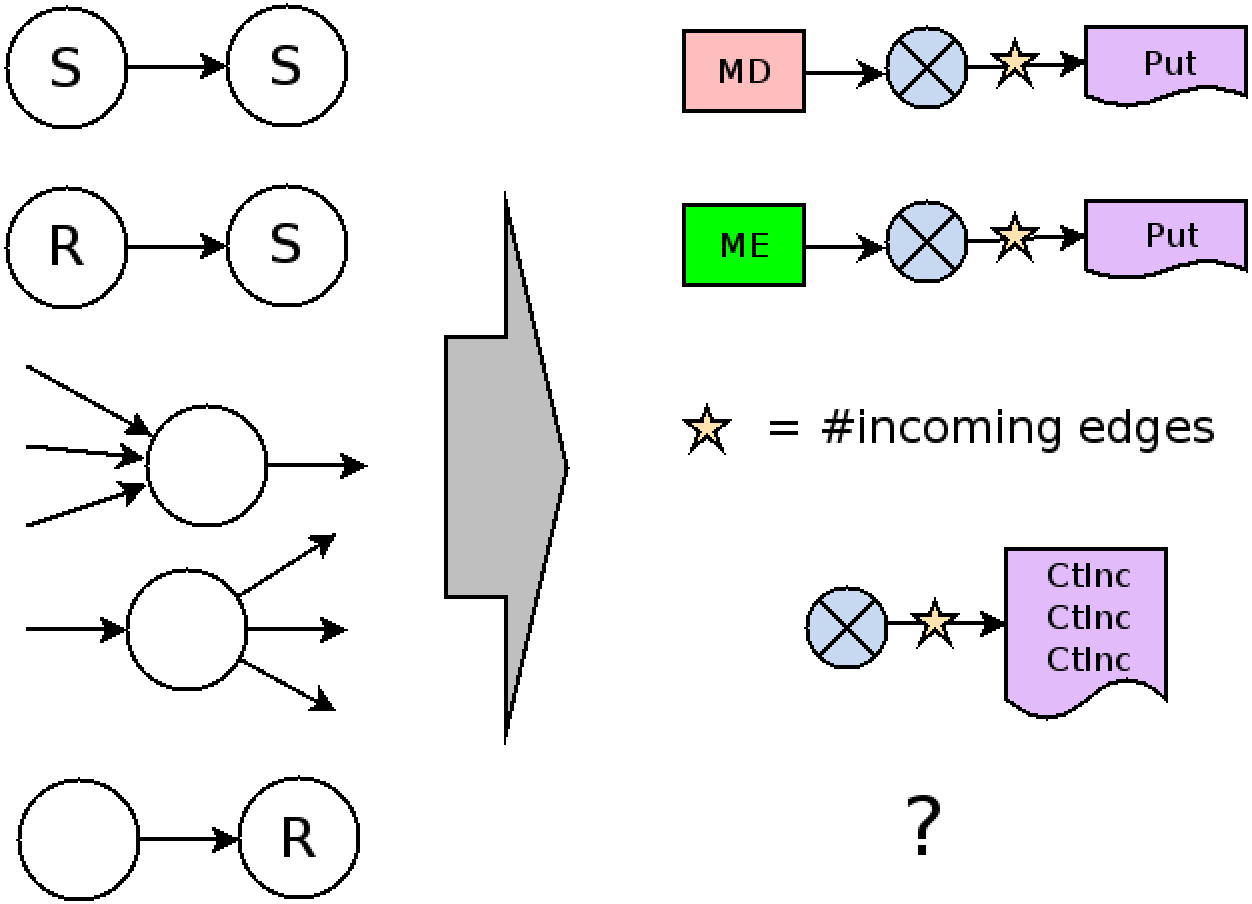
Counters

Threshold

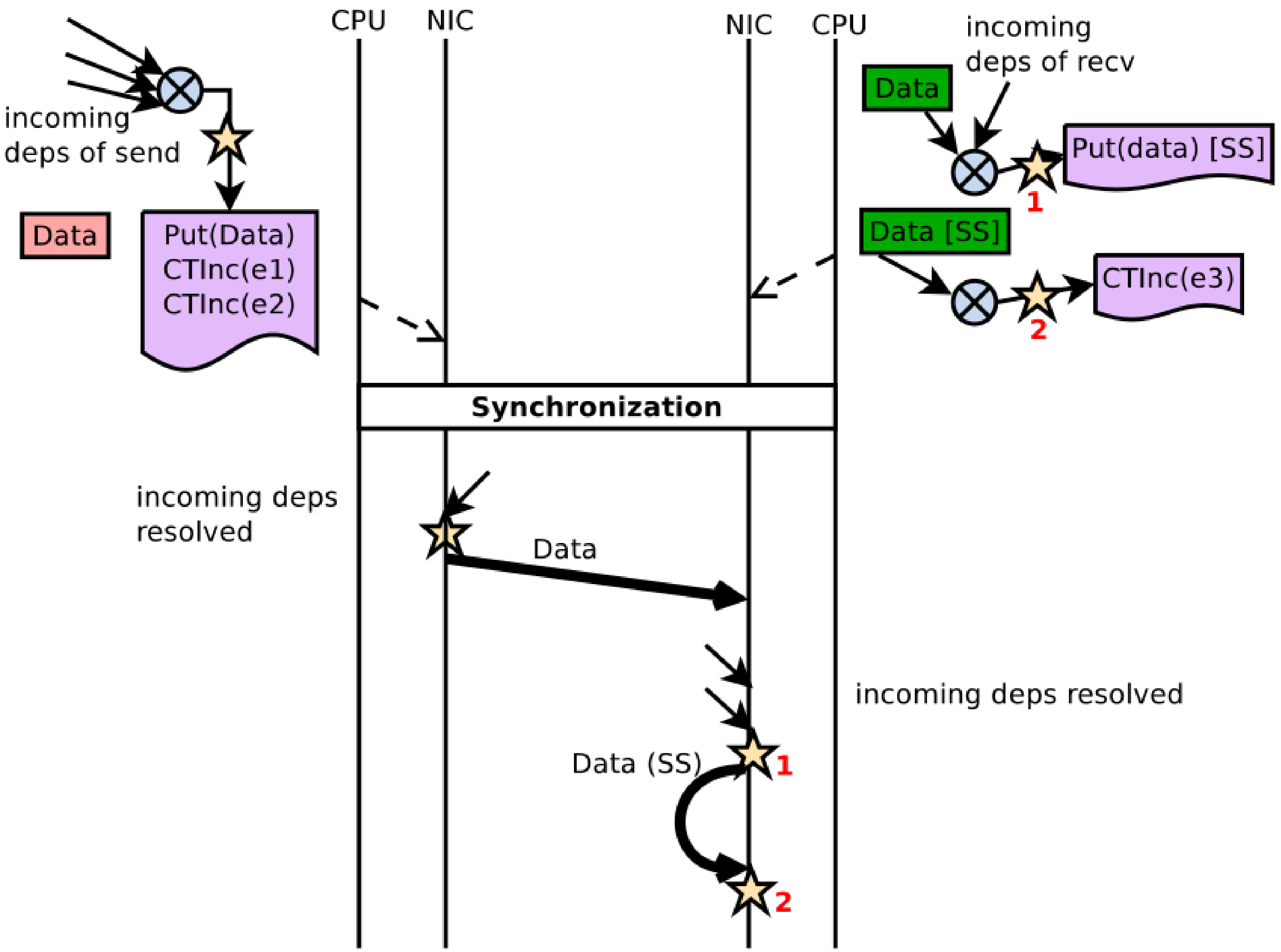
Memory Descriptors (MDs)

Triggered Operations:
i.e., Put, Set Counter, Increment Counter

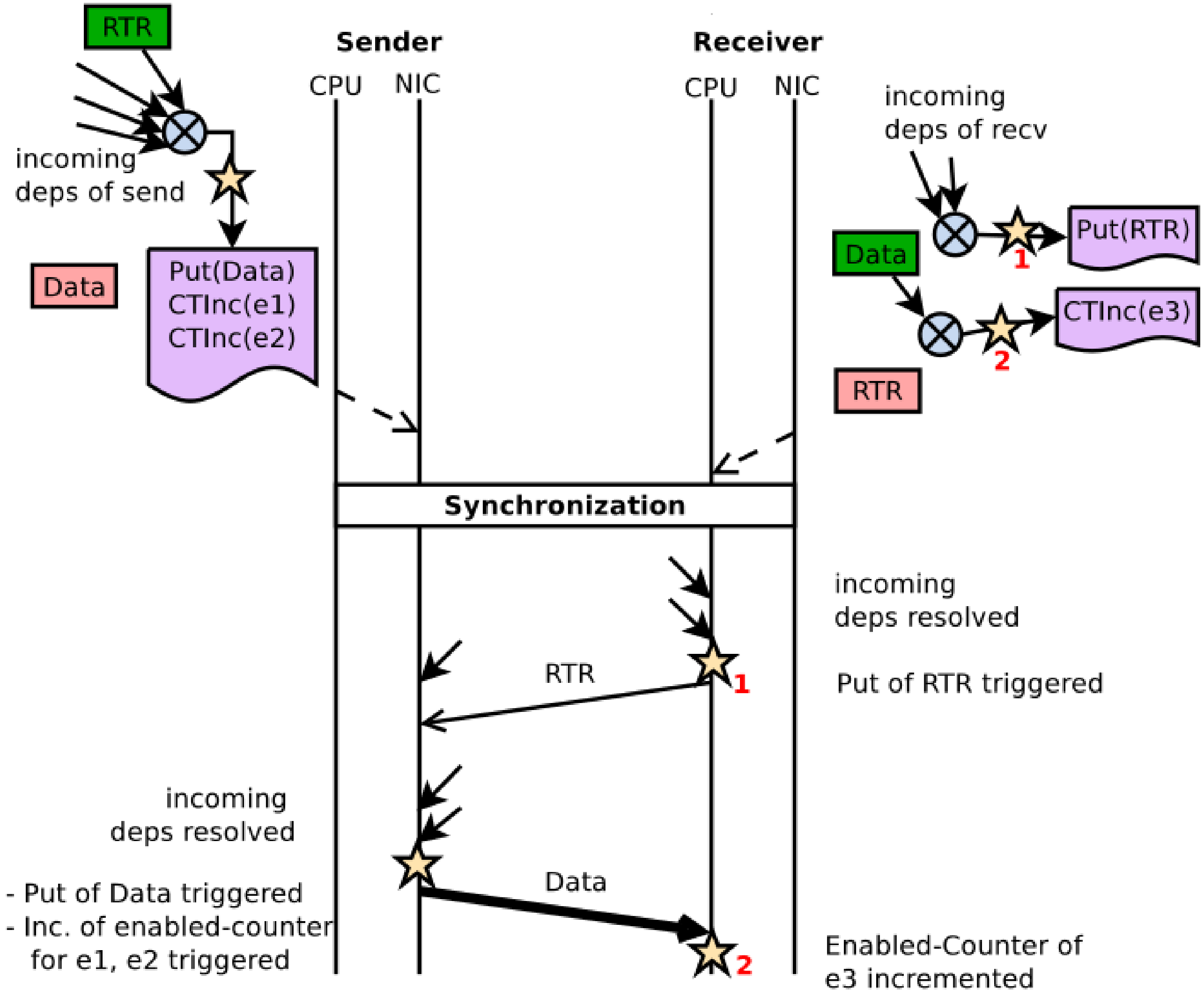
Translating cDAG to Portals



EAGER PROTOCOL



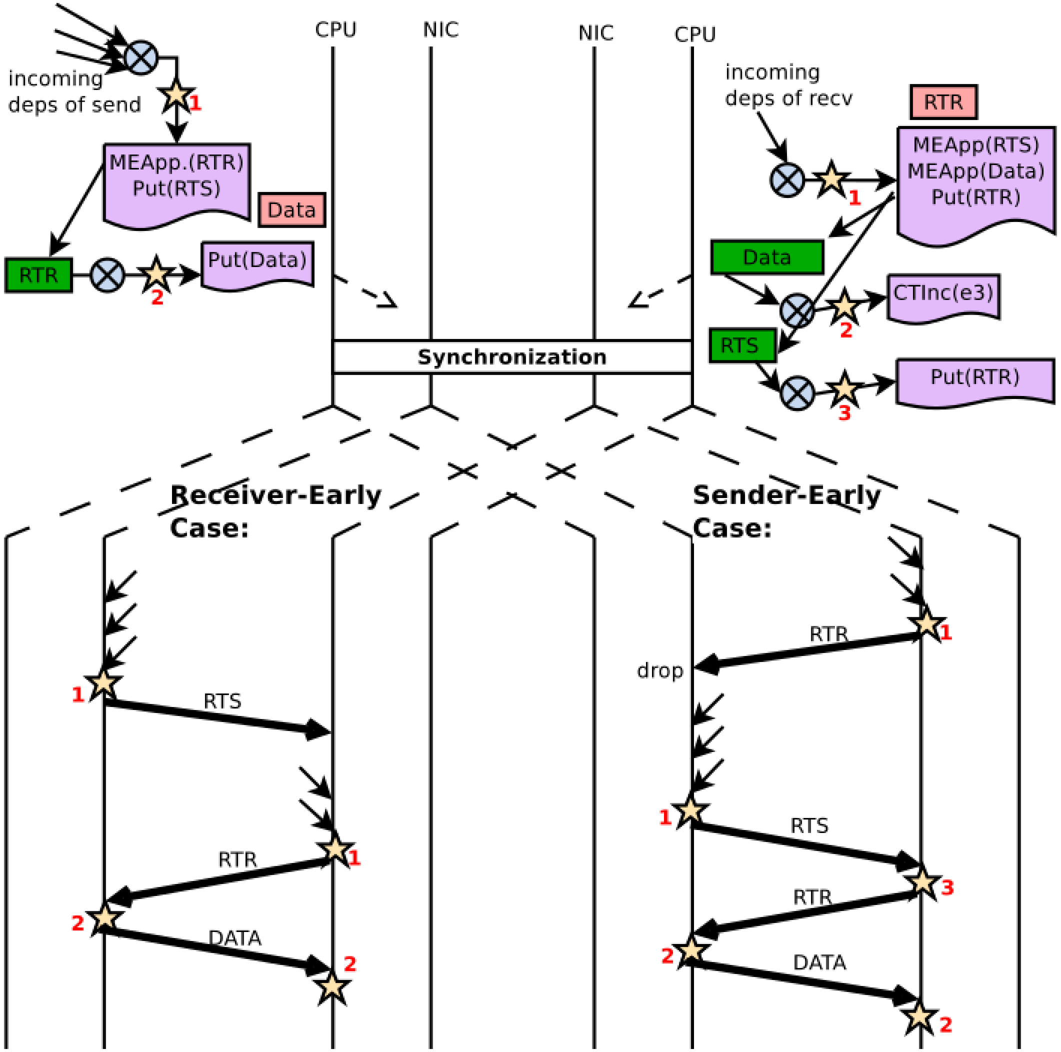
Pre-Matched Rendezvous



Missing Feature in Portals

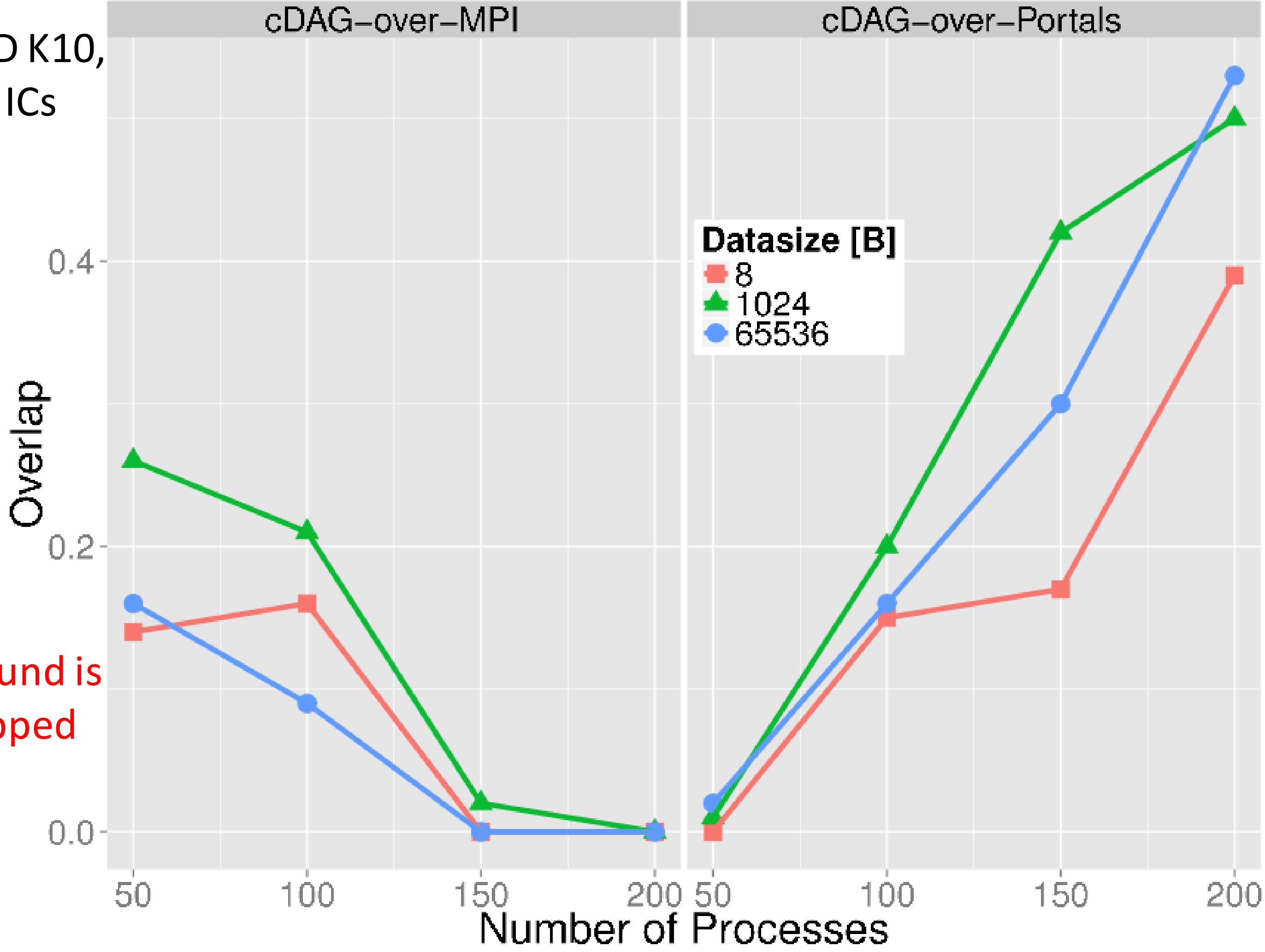
- Impossible to influence matching with Triggered Operations!
- We propose PtlTriggeredMEAppend()
- Adds (“activates”) a pre-defined Match Entry

Rendezvous Protocol



OVERLAP

Broadcast,
2.9 Ghz AMD K10,
Infiniband NICs

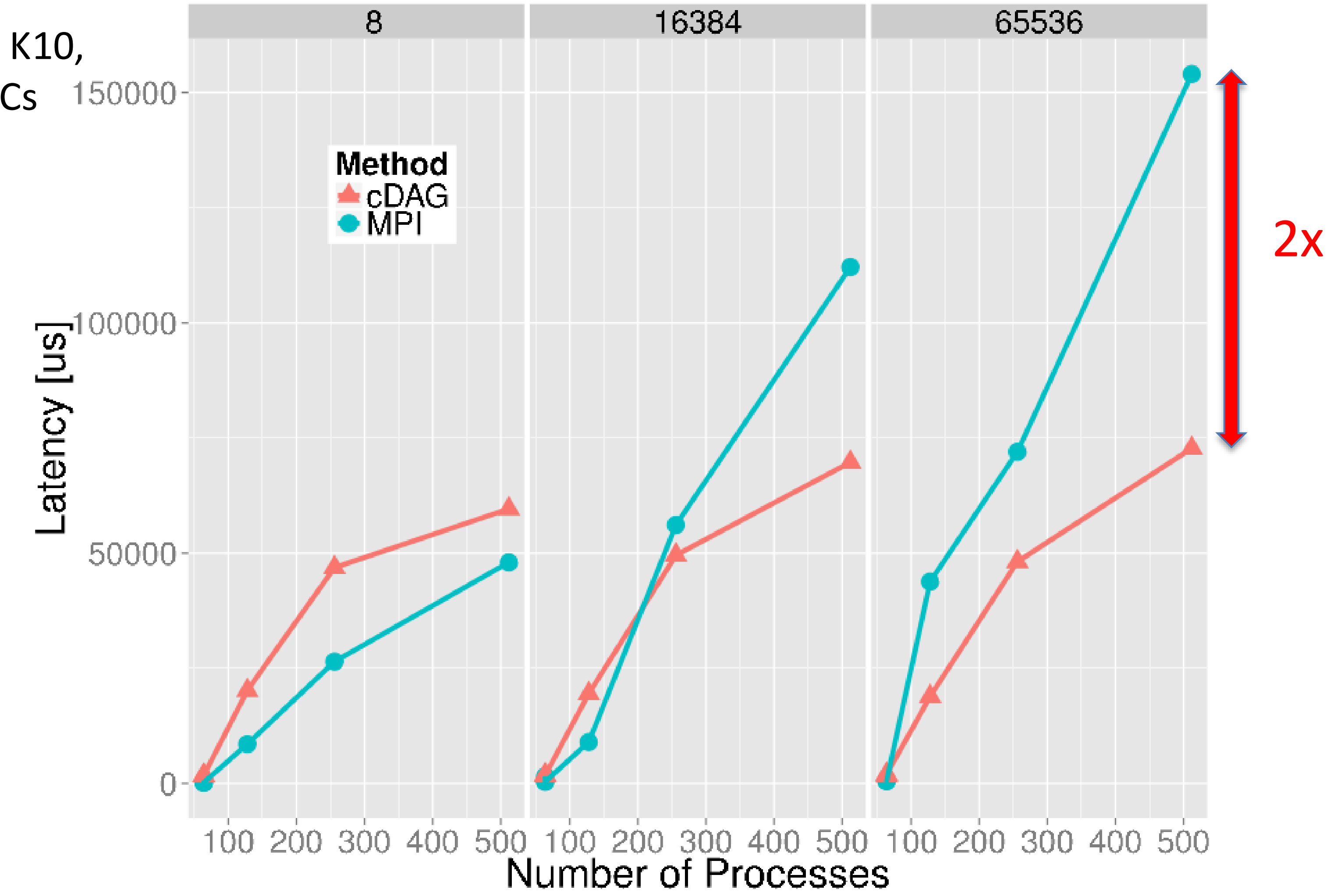


All rounds overlapped

First round is overlapped

Non-overlapped Latency

Broadcast,
2.9 Ghz AMD K10,
Infiniband NICs



Conclusions

- cDAG works well as a programming model for collective offload
- Translating cDAG to Portals 4 exposed "missing features" in current spec
- Good overlap and latency, even on software-emulated reference implementation
- cDAG backends for other offload engines (i.e., ConnectX) are future work

THANK YOU!

- Time for questions!
- Offline questions: timos@inf.ethz.ch
- Slides will be published, see Publication list at http://spcl.inf.ethz.ch