Interprocessor Communication seen as load/store instruction generalization

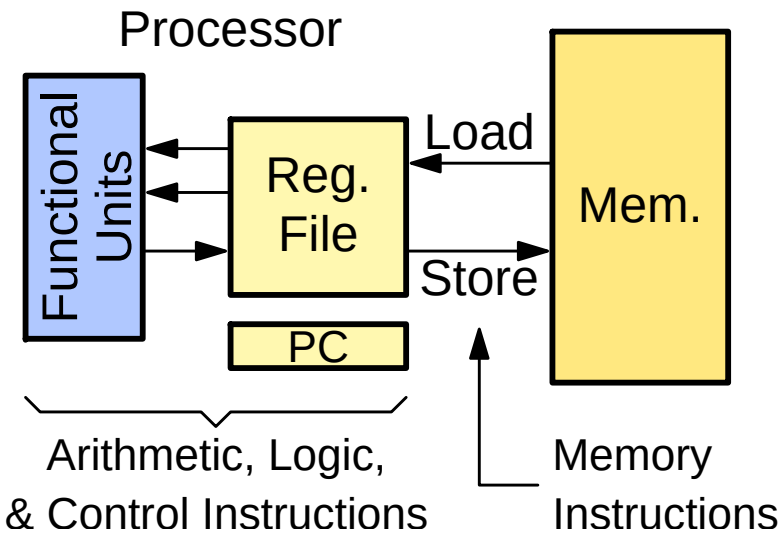
Manolis Katevenis

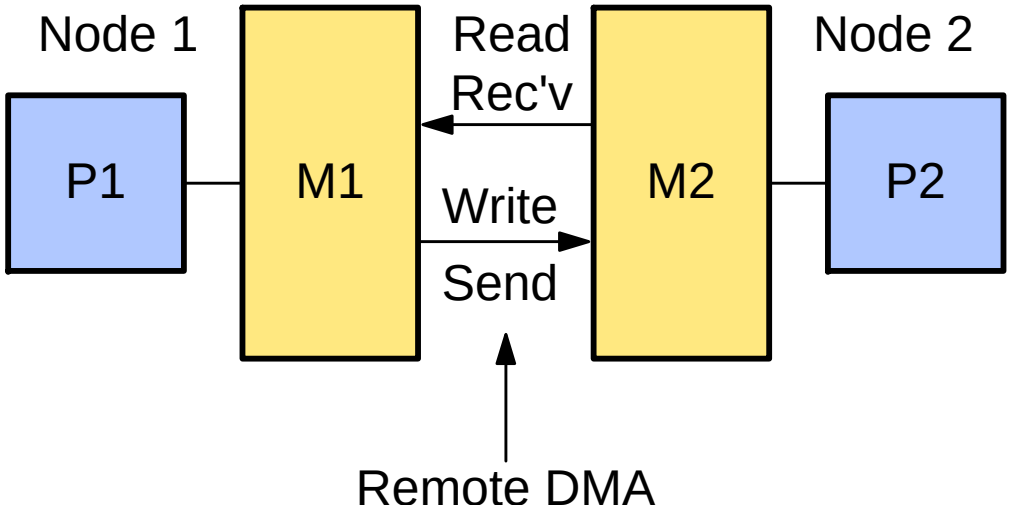
FORTH and University of Crete, Greece

Summary

- Central importance of Interprocessor Communication
- Must go from low-speed I/O to high-speed p2p commun.
- Data Transfer Primitives: Remote DMA, Remote Queues
- Cache operations on top of Network Interface primitives?
- Combined Translation/Routing Tables – Data Migration

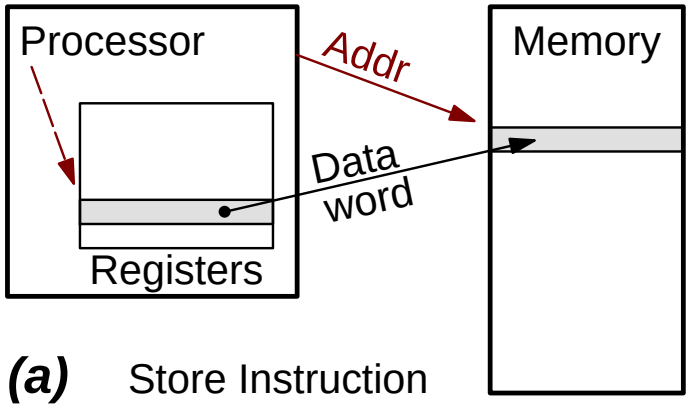
Communication Primitives: Intra-Node, Inter-Node



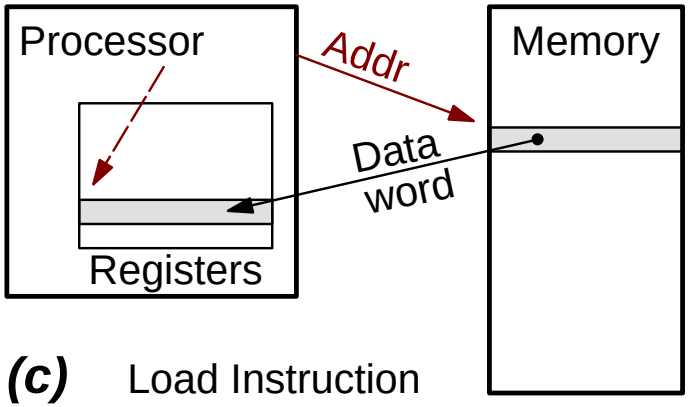


- Processor to Memory communication:
  - Load/Store primitives
- Interprocessor communication:
  - Read/Write (send/receive) data transfer primitives

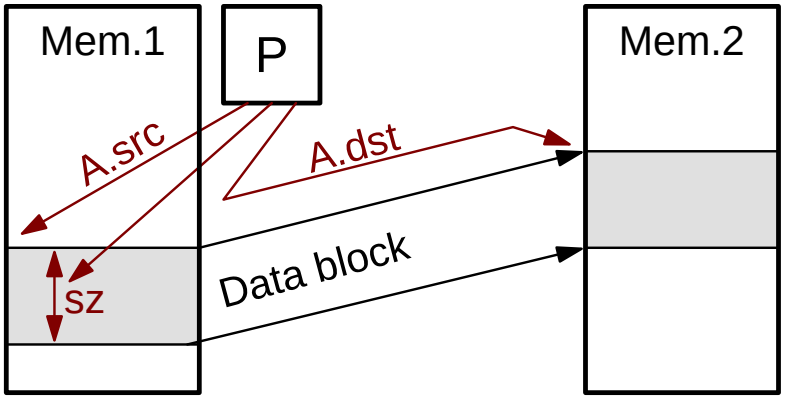
Remote DMA as generalization of single-word Instr'ns



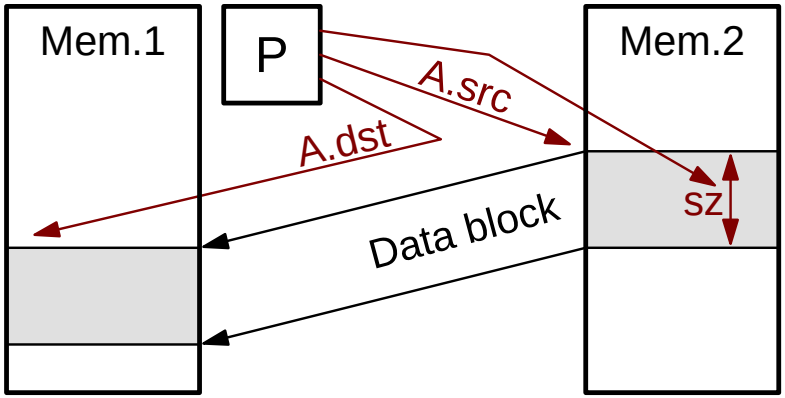
(a) Store Instruction



(c) Load Instruction



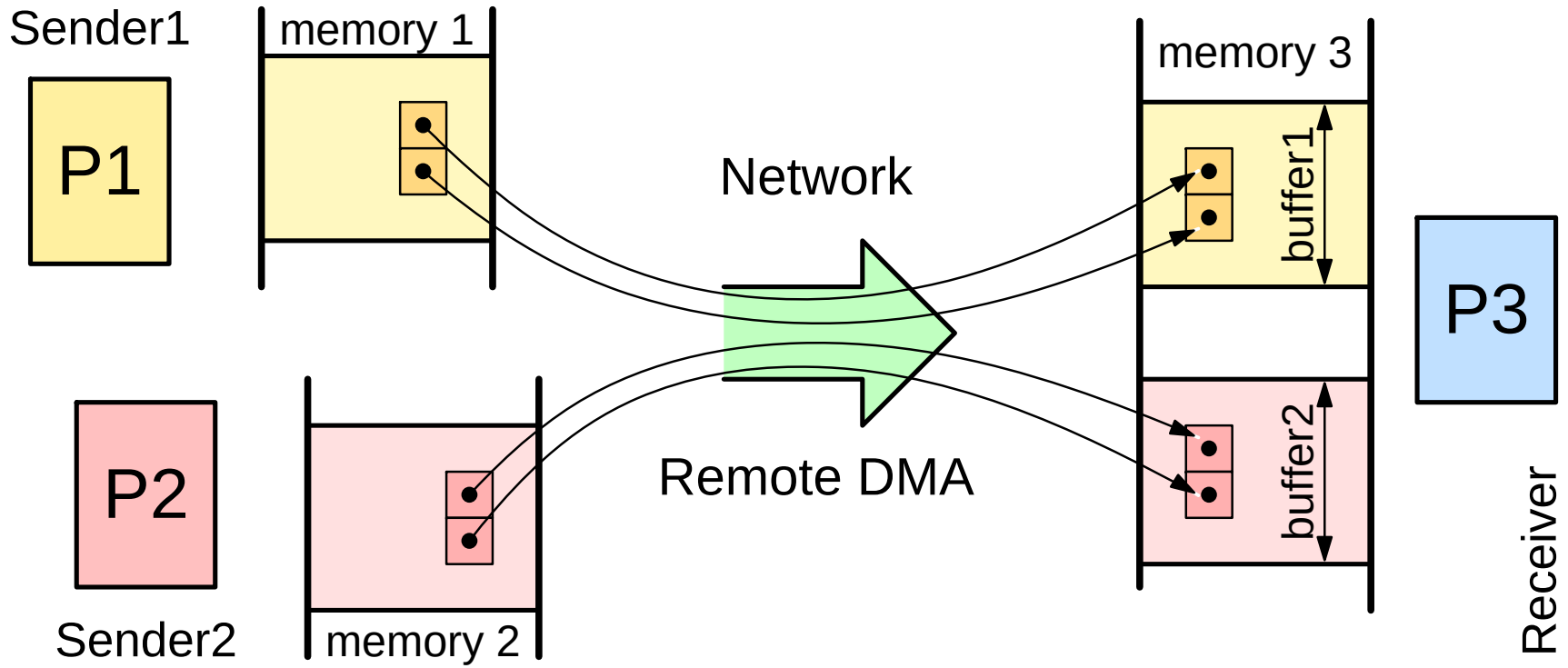
(b) Remote Write DMA



(d) Remote Read DMA

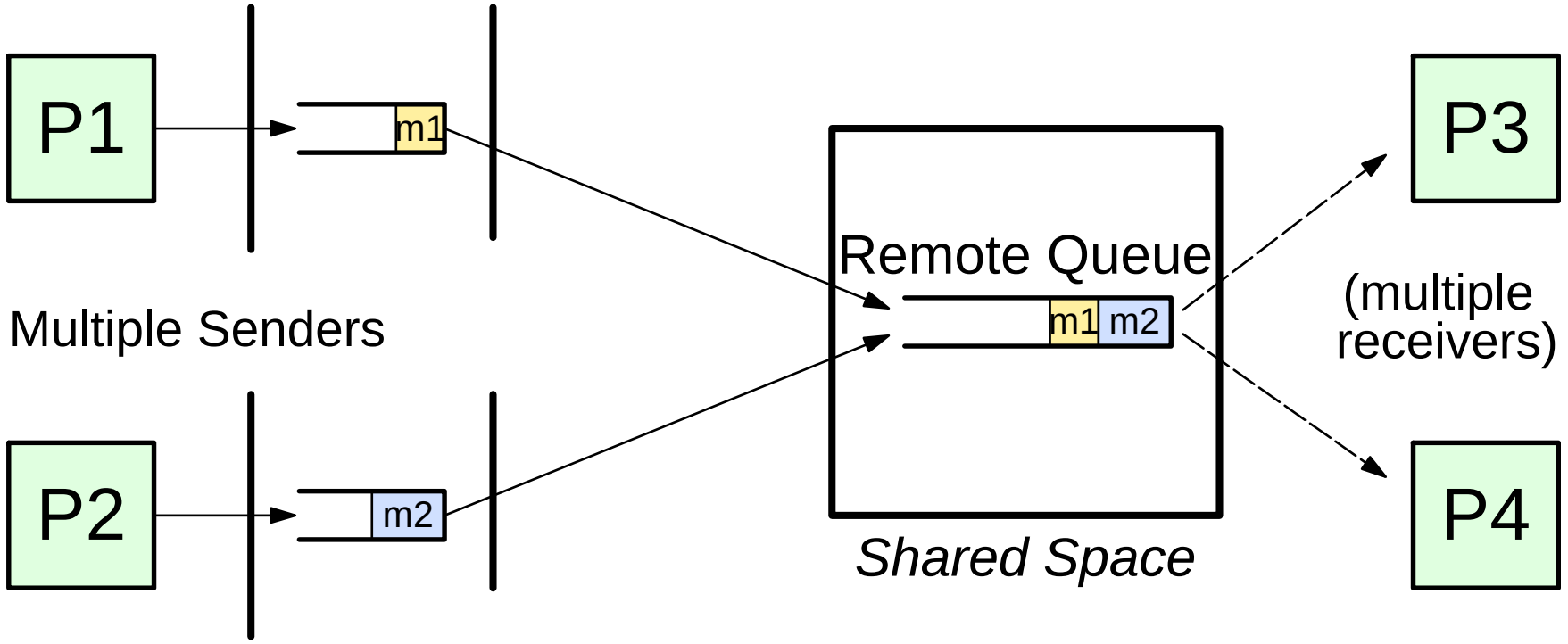
- block size is chosen so as to reduce overheads relative to data payload

Remote DMA is for One-to-One Communication



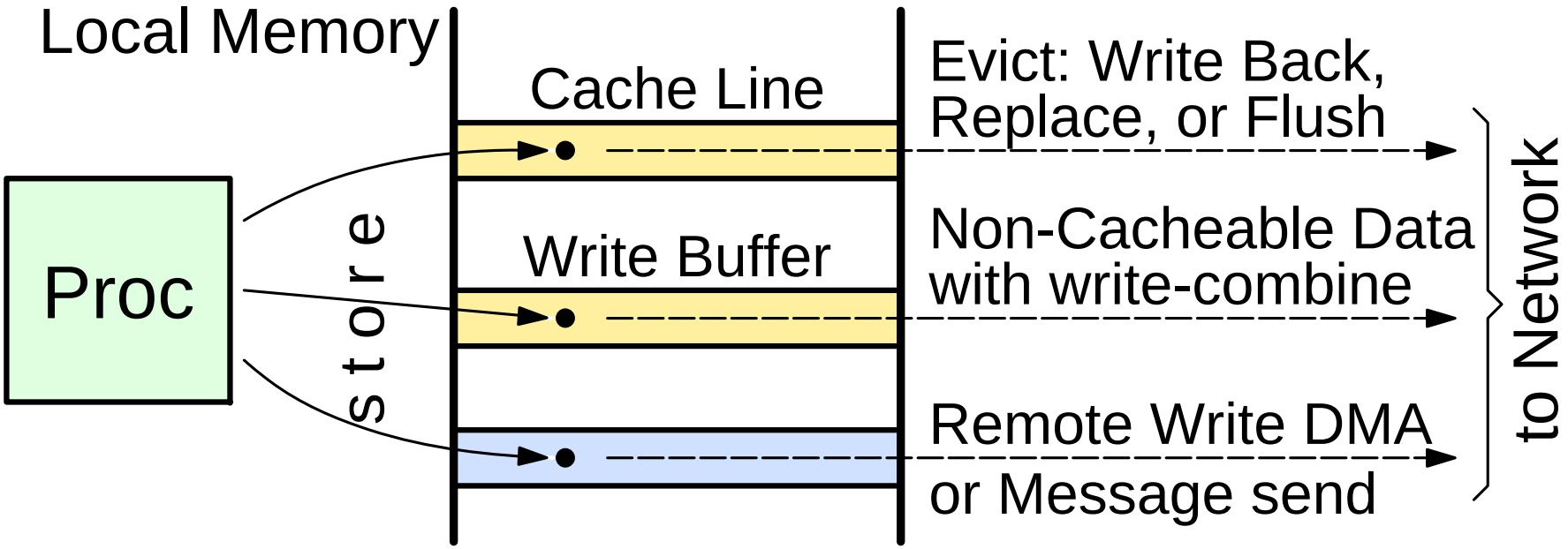
- Independent (unsynchronized) transfers have to occur into distinct memory regions
- Expensive when many potential senders but few actual ones
  - buffer reservation cost, polling-for-completion overhead

Multi-Party Synchronization: Remote Queues



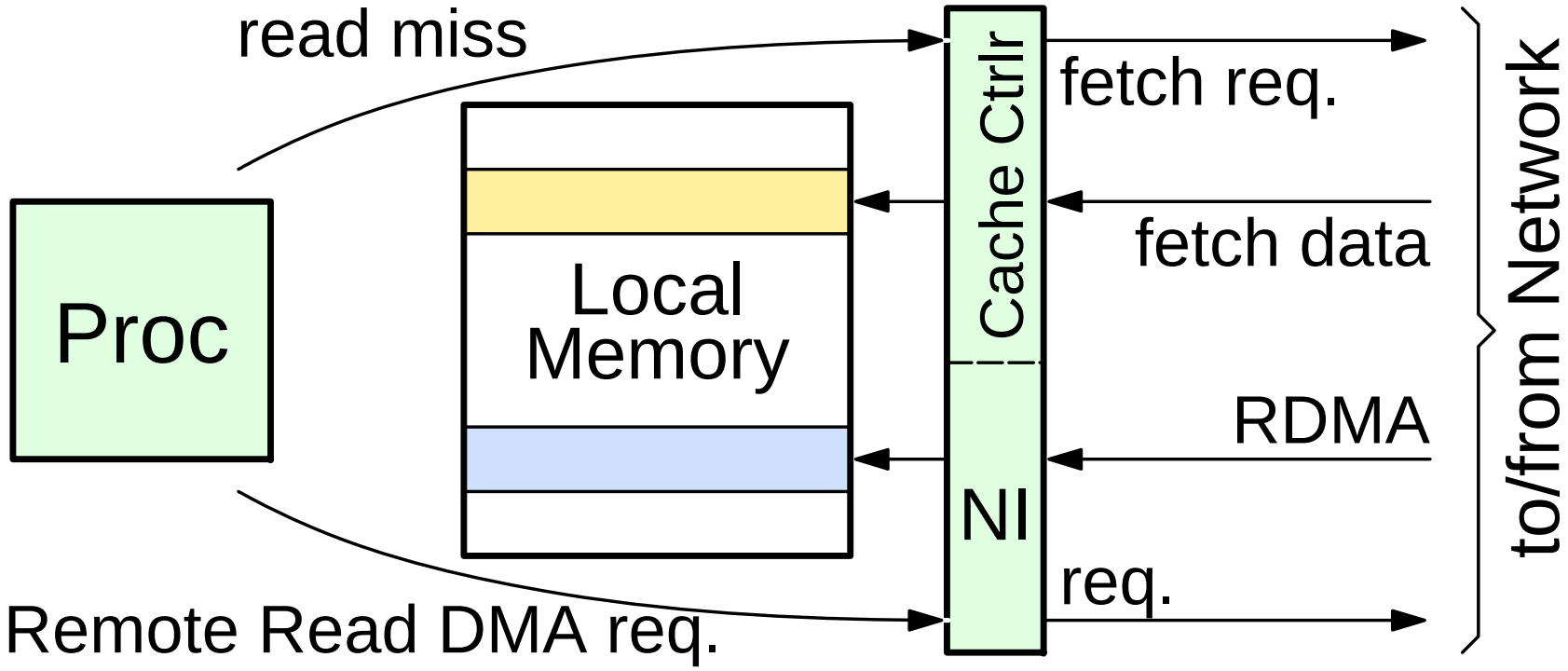
- Atomic enqueue into shared space, unlike dedicated sp. with RDMA
- Space reserved only for # of actual senders – not potential senders
- Speeds up polling / waiting on multiple receive channels
- Appropriate for synchronization (remote enqueue) & job dispatching (remote dequeue) – generalization of atomic operations

Cache Operations related to RDMA



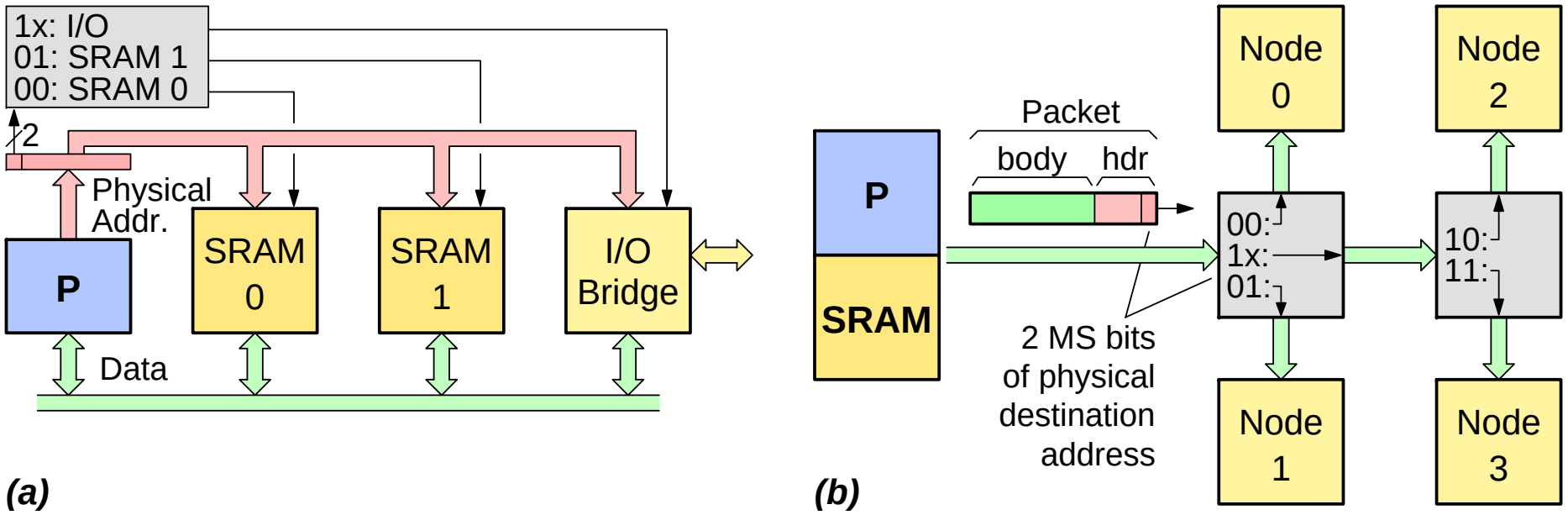
- Network interface as close to the processor as (L1) cache
- Messages or RDMA commands composed via *store* instr'ns
- Cache line eviction is a case of Remote Write DMA
- Is there potential for the cache controller to use the primitives supplied by the network interface?

Cache Read Misses are like Remote Read DMA's



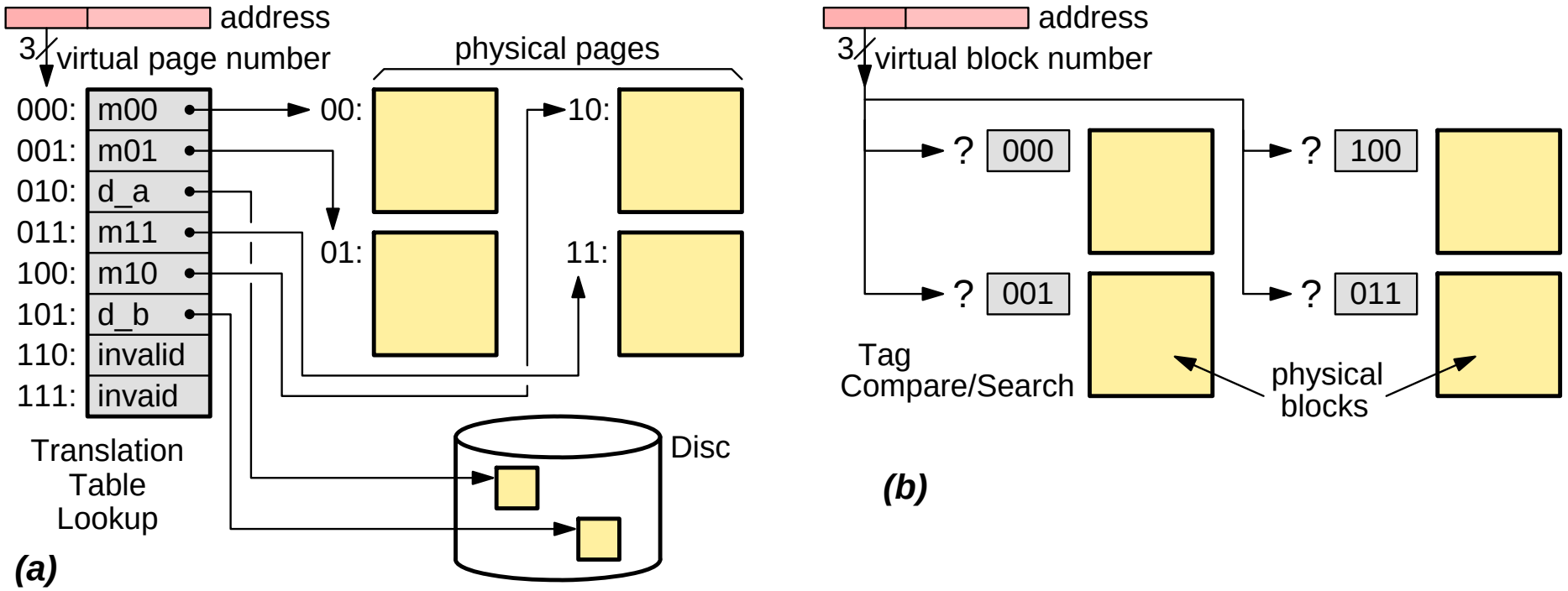
- Network Interface as close to the processor as (L1) cache
- Should the NI be combined with the cache controller?
- Should portions of cache coherence protocols be left to the software, with NI hardware assistance for the rest?

Network Routing as Generalization of Address Decoding



- (a) Physical address decoding in a uniprocessor
- (b) geographical address routing in a multiprocessor

Address Translation for Transparent Data Migration

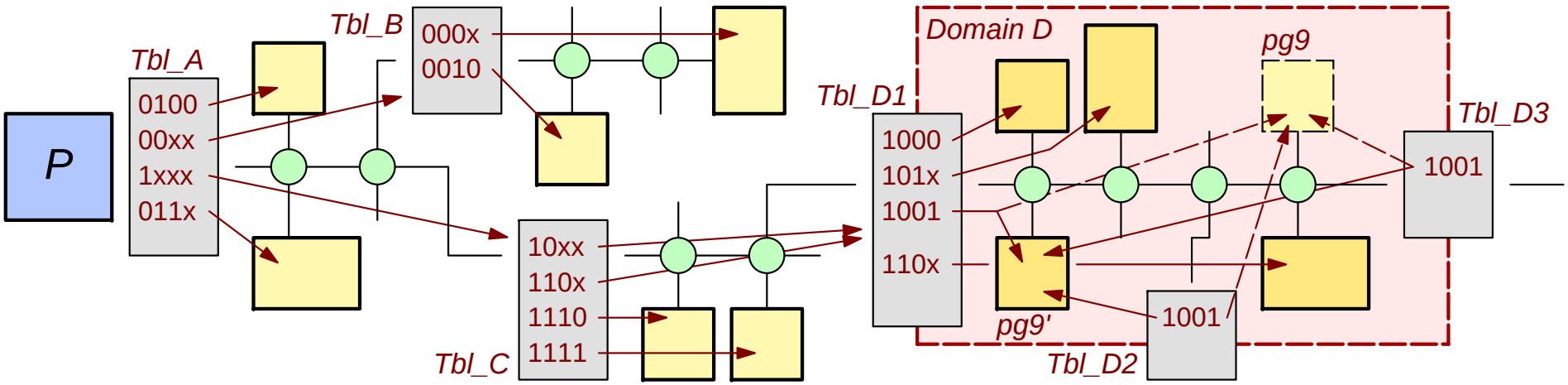


- Two methods to support data migration:

(a) Translation table: first consult an indirection/routing table/directory

(b) Cache style: search multiple places in parallel

Progressive Translation: Localize Migration Updates



- Packets carry global virtual addresses
- Tables provide physical route (address) for the next few steps
- When page 9 migrates within D, only tables in that domain need updating
- Variable-size-page translation tables look like internet routing tables (longest-prefix matches if we want small-page-within-big-region migration)
- Tables that partition the system, for protection against untrusted operating systems, look like internet firewalls

Conclusions

- Hardware should provide *"Primitives"* – not "solutions"
  - few, simple, general-purpose, flexibly combinable primitives
- Data transport primitive: Remote DMA
- Synchronization primitive: Remote Enqueue
- Cache operation on top of Network Interface primitives?
- Translation/Routing Tables for Data Migration support