Application-Transparent Near-Memory Processing Architecture with Memory Channel Network

Mohammad Alian[1], Seung Won Min[1], Hadi Asgharimoghaddam[1], Ashutosh Dhar[1], Dong Kai Wang[1], Thomas Roewer[2], Adam McPadden[2], Oliver O'Halloran[2], Deming Chen[1], Jinjun Xiong[2], Daehoon Kim[1], Wen-mei Hwu[1], and Nam Sung Kim[1,3]

[1]University of Illinois Urbana-Champaign
[2]IBM Research and Systems
[3]Samsung Electronics

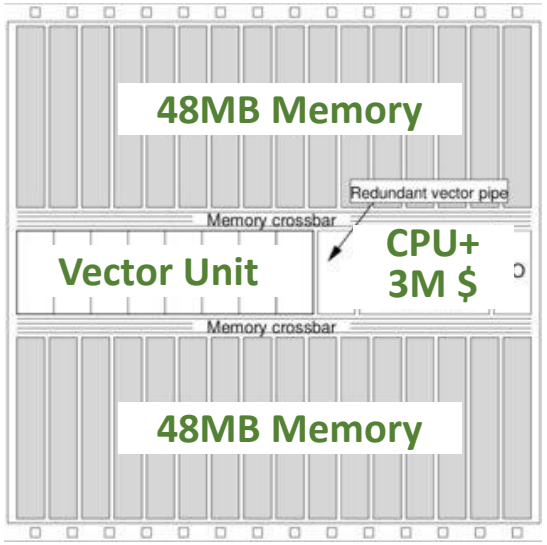
ILLINOIS
Electrical & Computer Engineering
COLLEGE OF ENGINEERING

C3SR center for cognitive computing systems research

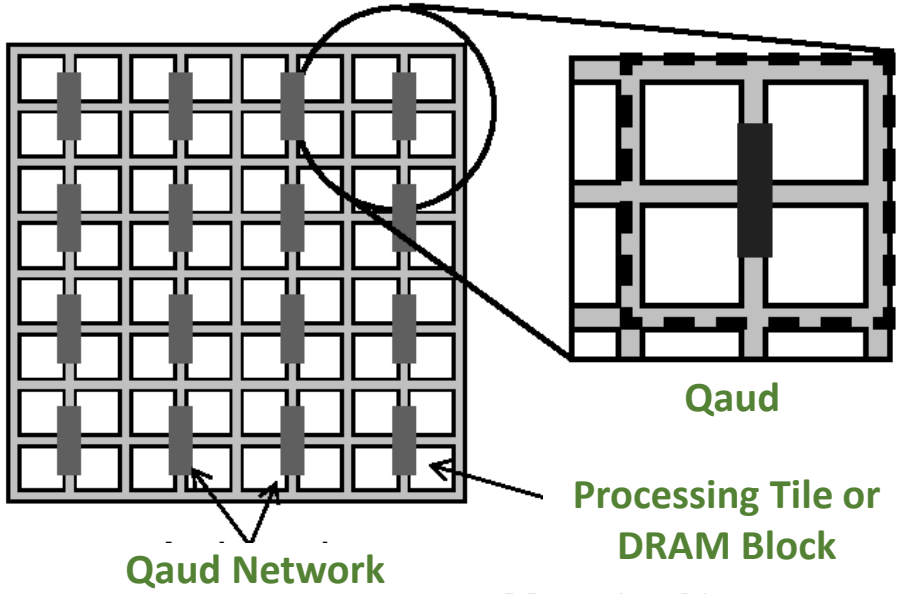
IBM | ILLINOIS

Executive Summary

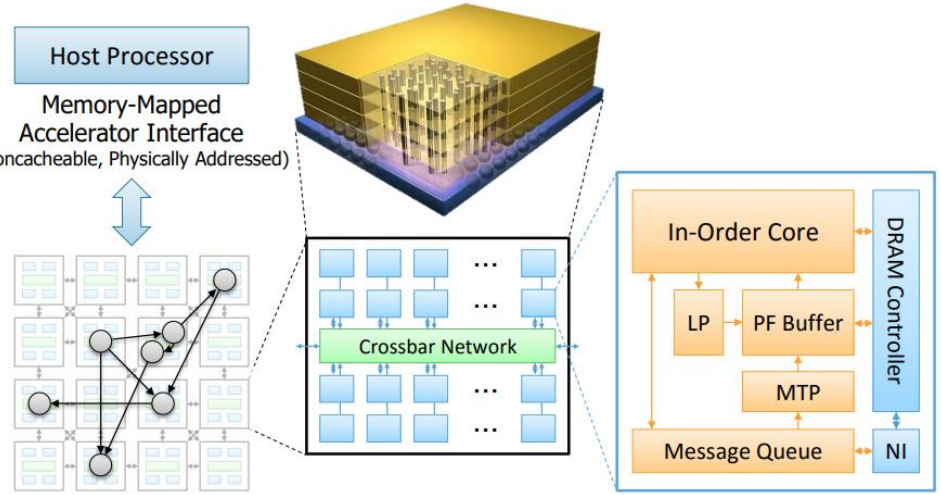
- Processing In Memory (PIM), Near Memory Processing (NMP), …
  - ✓EXECUBE'94, IRAM'97, ActivePages'98, FlexRAM'99, DIVA'99, SmartMemories'00, …
- **Question**: why haven't they been commercialized yet?
  - ✓Demand changes in application code and/or memory subsystem of host processor



IRAM'97



SmartMemories'00



ISCA'15

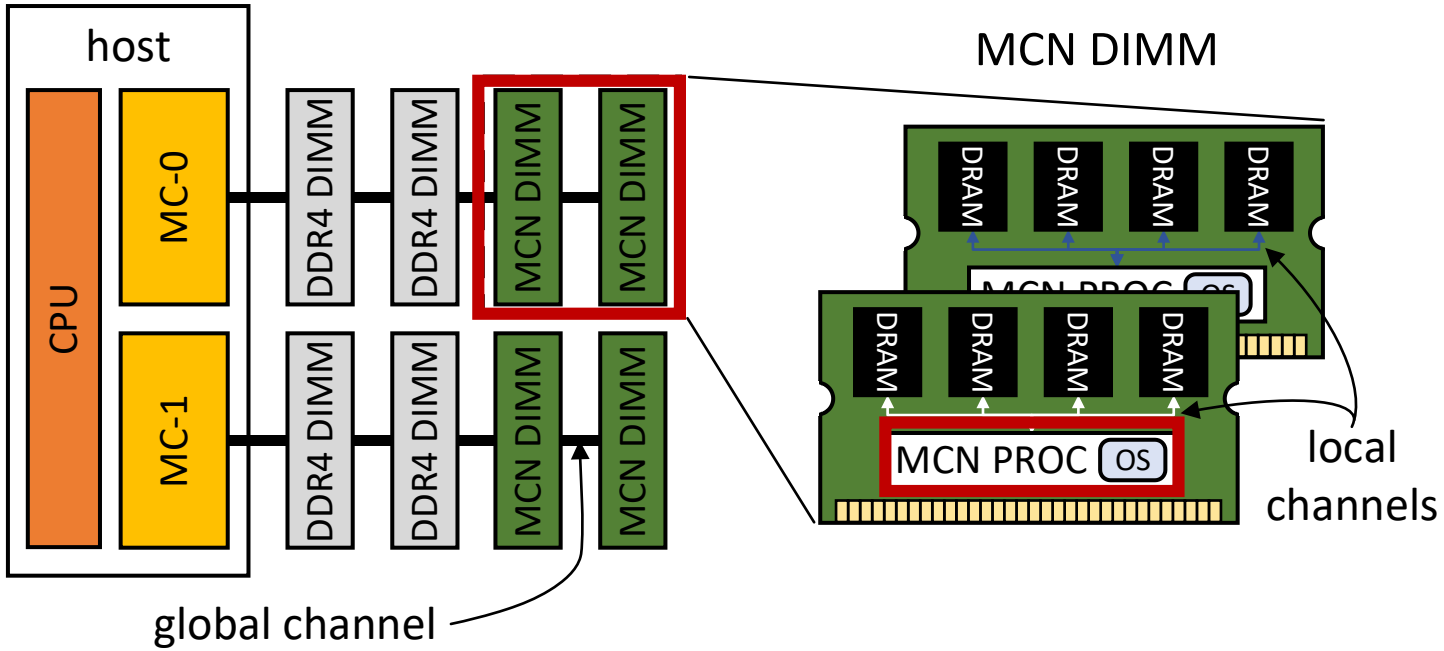
Executive Summary

- Processing In Memory (PIM), Near Memory Processing (NMP), …
  - ✓EXECUBE'94, IRAM'97, ActivePages'98, FlexRAM'99, DIVA'99, SmartMemories'00, …
- **Question**: why haven't they been commercialized yet?
  - ✓Demand changes in application code and/or memory subsystem of host processor
- **Solution**: memory module based NMP + Memory Channel Network (MCN)
  - ✓Recognize NMP memory modules as distributed computing nodes over **Ethernet** → no change in application code or memory subsystem of host processors
  - ✓Seamlessly integrate NMP w/ distributed computing frameworks for better scalability

Executive Summary

- Processing In Memory (PIM), Near Memory Processing (NMP), …
  - ✓EXECUBE'94, IRAM'97, ActivePages'98, FlexRAM'99, DIVA'99, SmartMemories'00, …
- **Question**: why haven't they been commercialized yet?
  - ✓Demand changes in application code and/or memory subsystem of host processor
- **Solution**: memory module based NMP + Memory Channel Network (MCN)
  - ✓Recognize NMP memory modules as distributed computing nodes over **Ethernet** → no change in application code or memory subsystem of host processors
  - ✓Seamlessly integrate NMP w/ distributed computing frameworks for better scalability
- **Feasibility & Performance:**
  - ✓Demonstrate the feasibility w/ an **IBM POWER8 + experimental memory module**
  - ✓Improve the performance and processing bandwidth by **43%** and **4×**, respectively

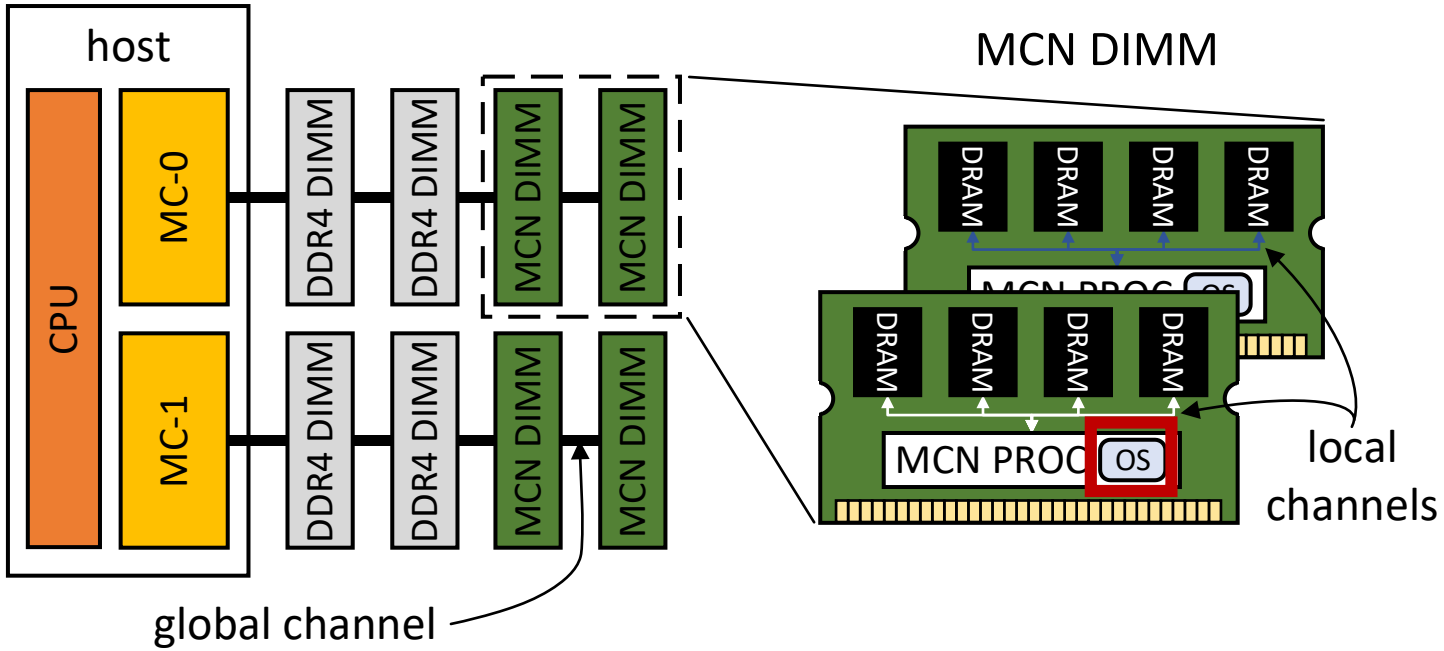
Overview of MCN-based NMP



*Application Processor

- Buffered DIMM w/ a low-power but powerful AP* in a buffer device

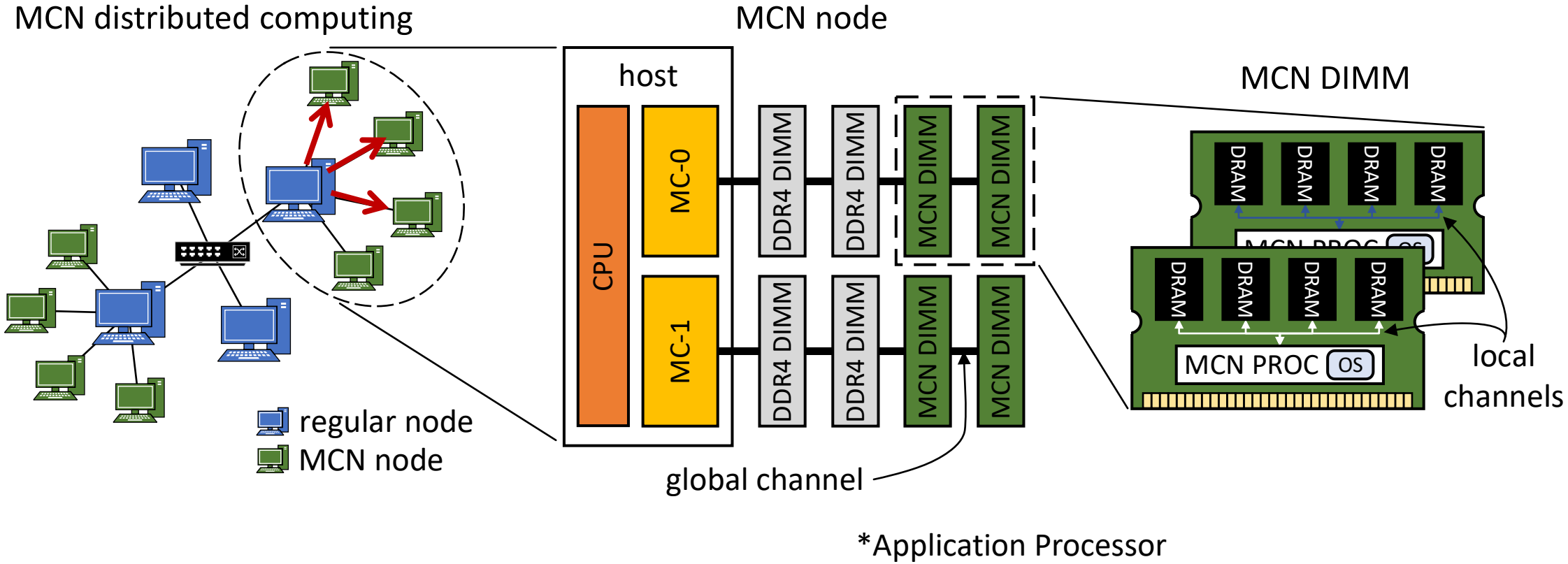
Overview of MCN-based NMP



*Application Processor

- Buffered DIMM w/ a low-power but powerful AP* in a buffer device
  - ✓An MCN processor runs its own lightweight OS including the minimum network stack

Overview of MCN-based NMP

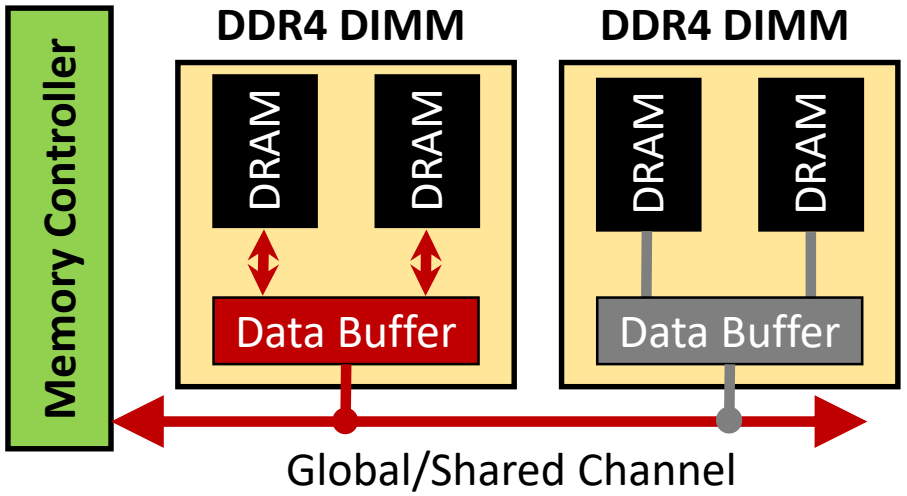


- Buffered DIMM w/ a low-power but powerful AP* in a buffer device
- Special driver faking memory channels as Ethernet connections

Higher Processing BW* w/ Commodity DRAM

*bandwidth

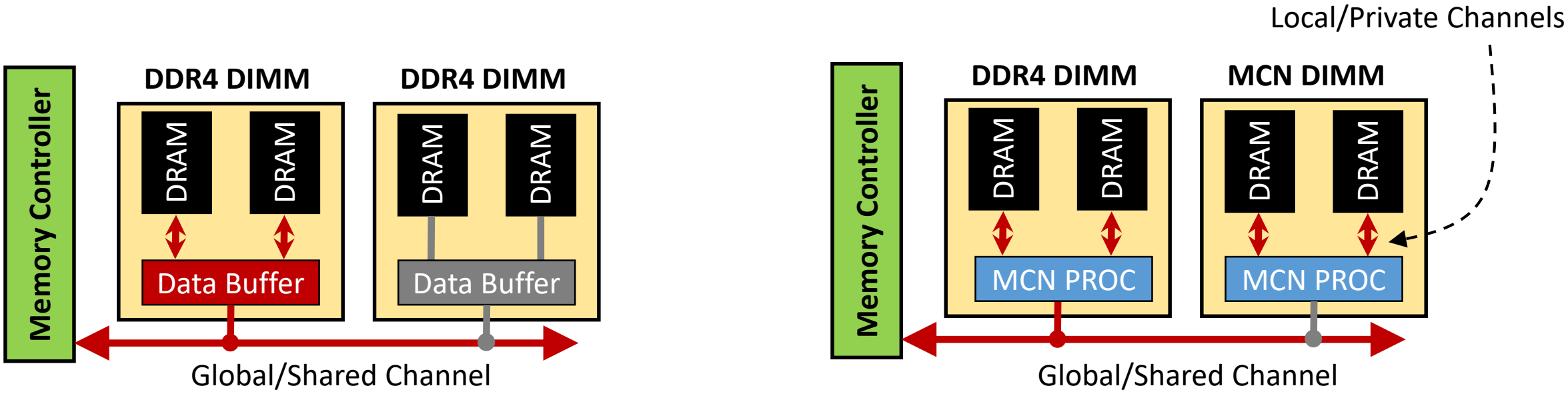
- Conventional memory system
  - ✓More DIMMs → larger capacity but the same bandwidth



Higher Processing BW* w/ Commodity DRAM

*bandwidth

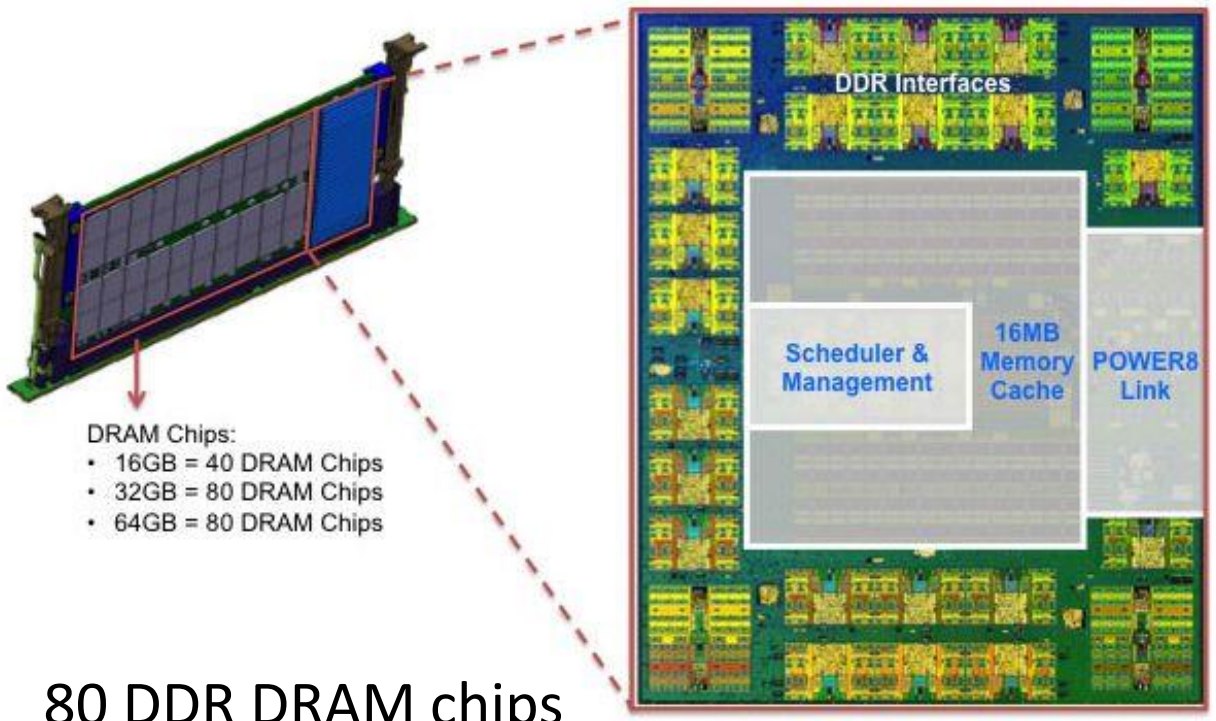
- Conventional memory system w/ near memory processing DIMMs
  - ✓an MCN processor w/ local DRAM devices through private channels → scaling aggregate processing memory bandwidth w/ # of MCN DIMMs



MCN DIMM Architecture

IBM Centaur DIMM

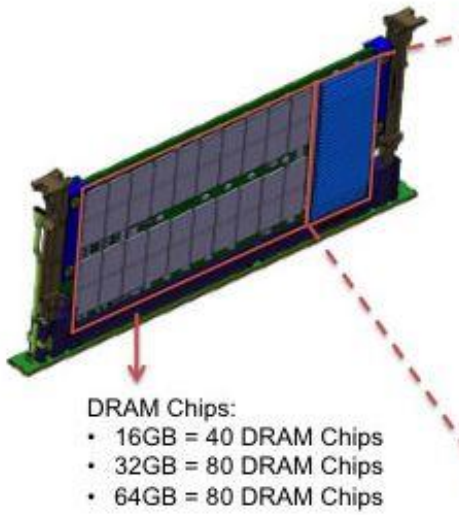
Buffer Device



80 DDR DRAM chips
+ buffer chip
w/ a tall form factor

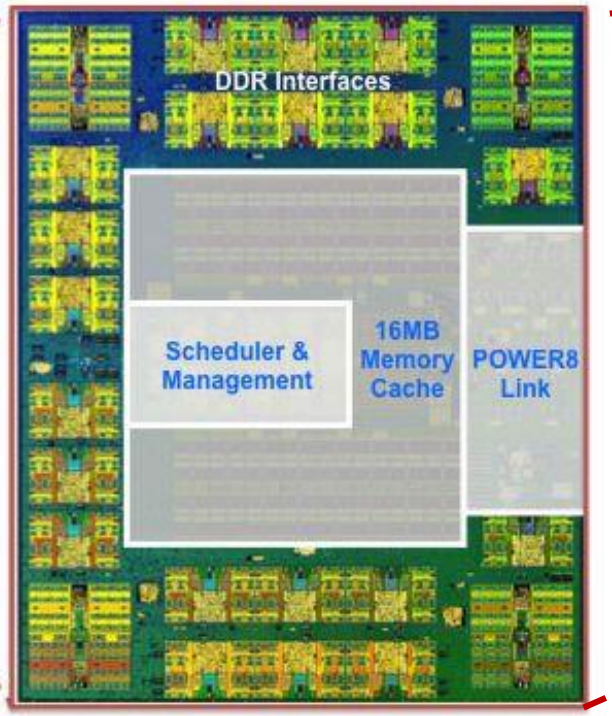
MCN DIMM Architecture

IBM Centaur DIMM



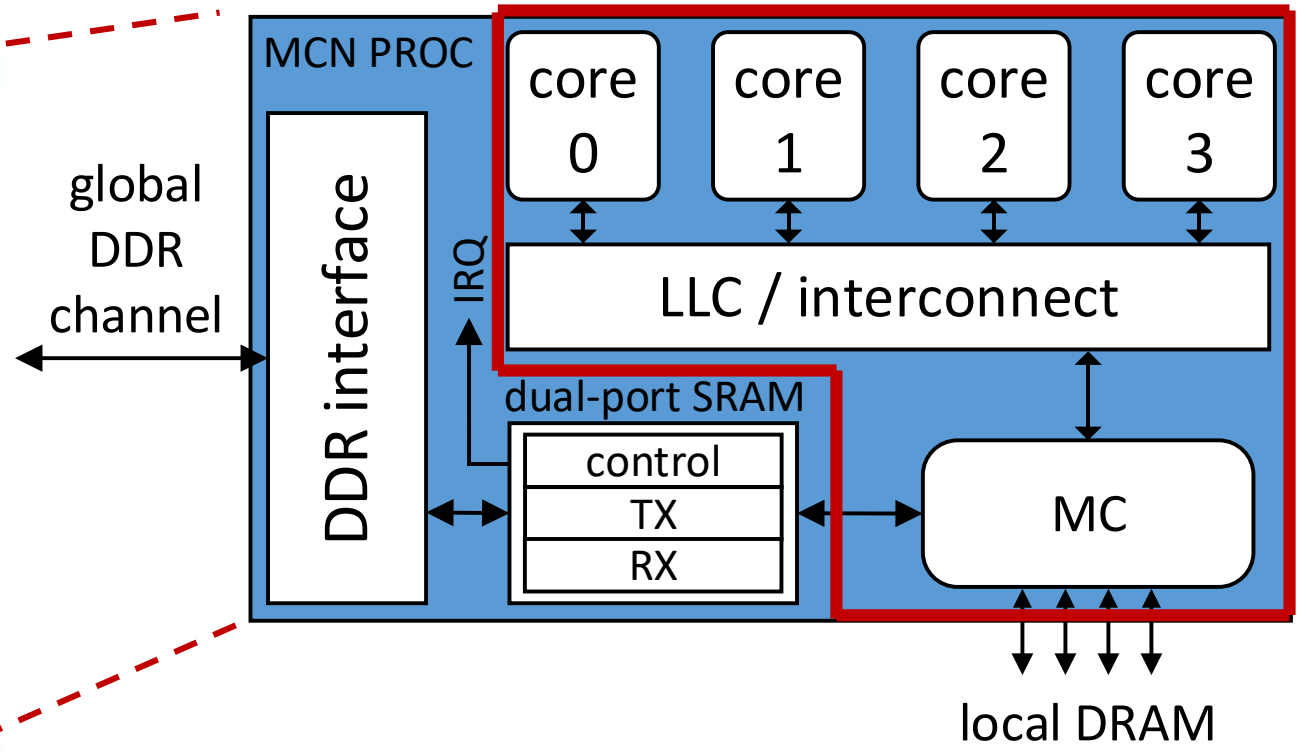
80 DDR DRAM chips
+ buffer chip
w/ a tall form factor

Buffer Device



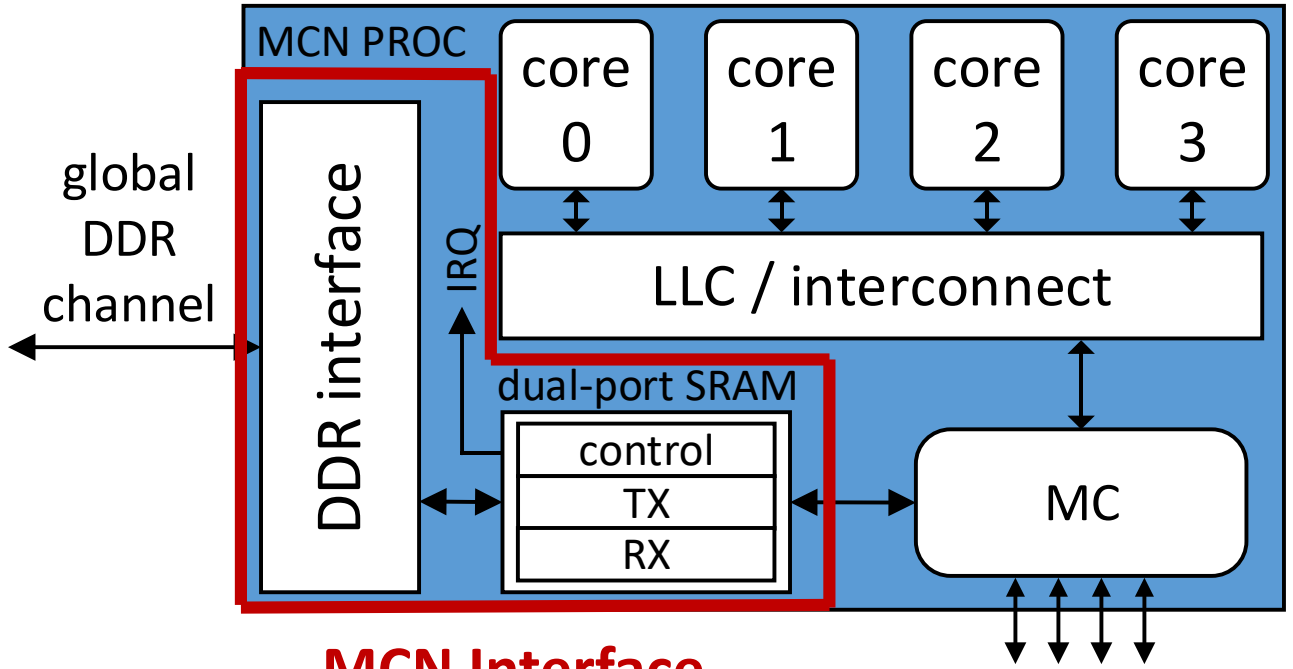
~20W TDP &
~10*mm*×10*mm*

Near-Memory Processor



Snapdragon AP w/ 4 A57 ARM cores + 2MB LLC, GPU, 2 MCs, etc. @
~5W & ~8×8mm^2 (1.8W & ~2×2mm^2)

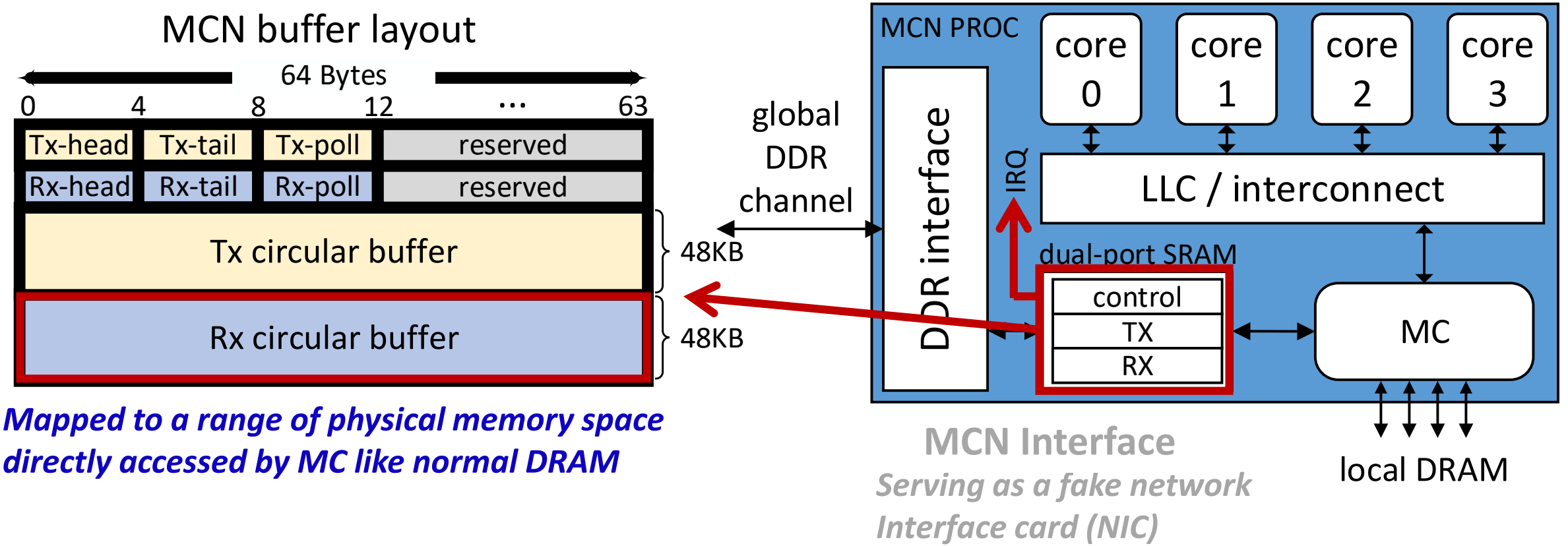
MCN DIMM Architecture: Interface Logic



MCN Interface

Serving as a fake network Interface card (NIC)

MCN DIMM Architecture: Interface Logic

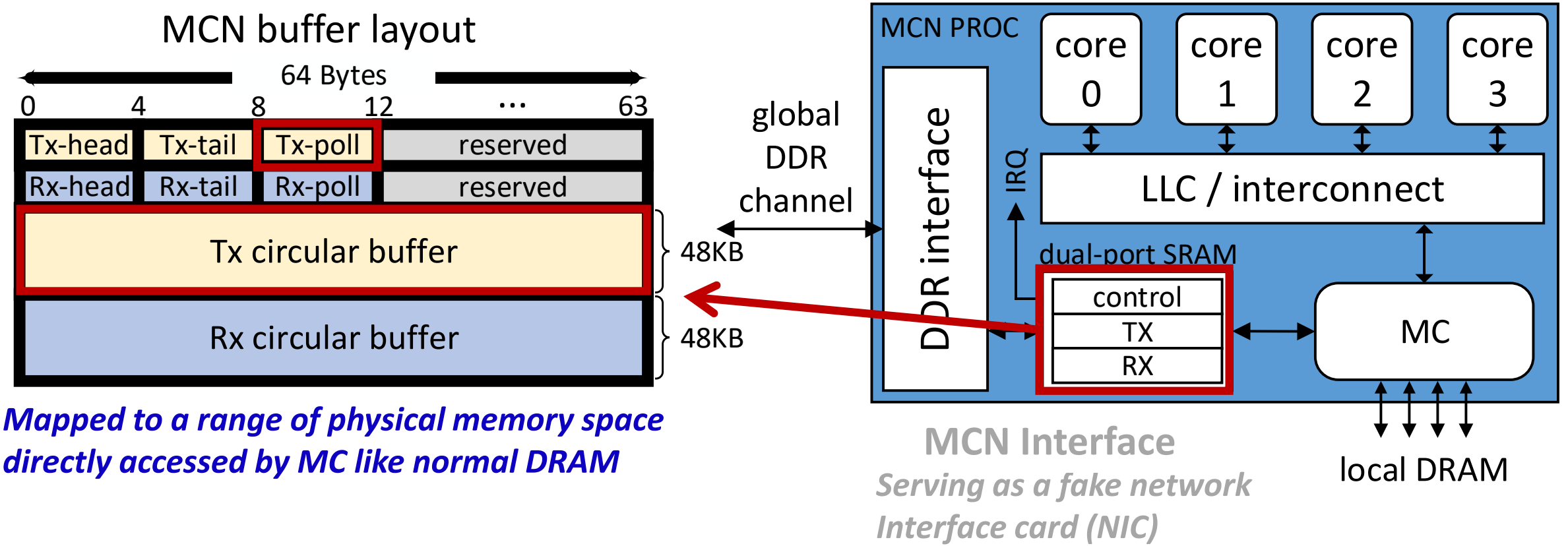


Mapped to a range of physical memory space directly accessed by MC like normal DRAM

MCN Interface

Serving as a fake network Interface card (NIC)

MCN DIMM Architecture: Interface Logic

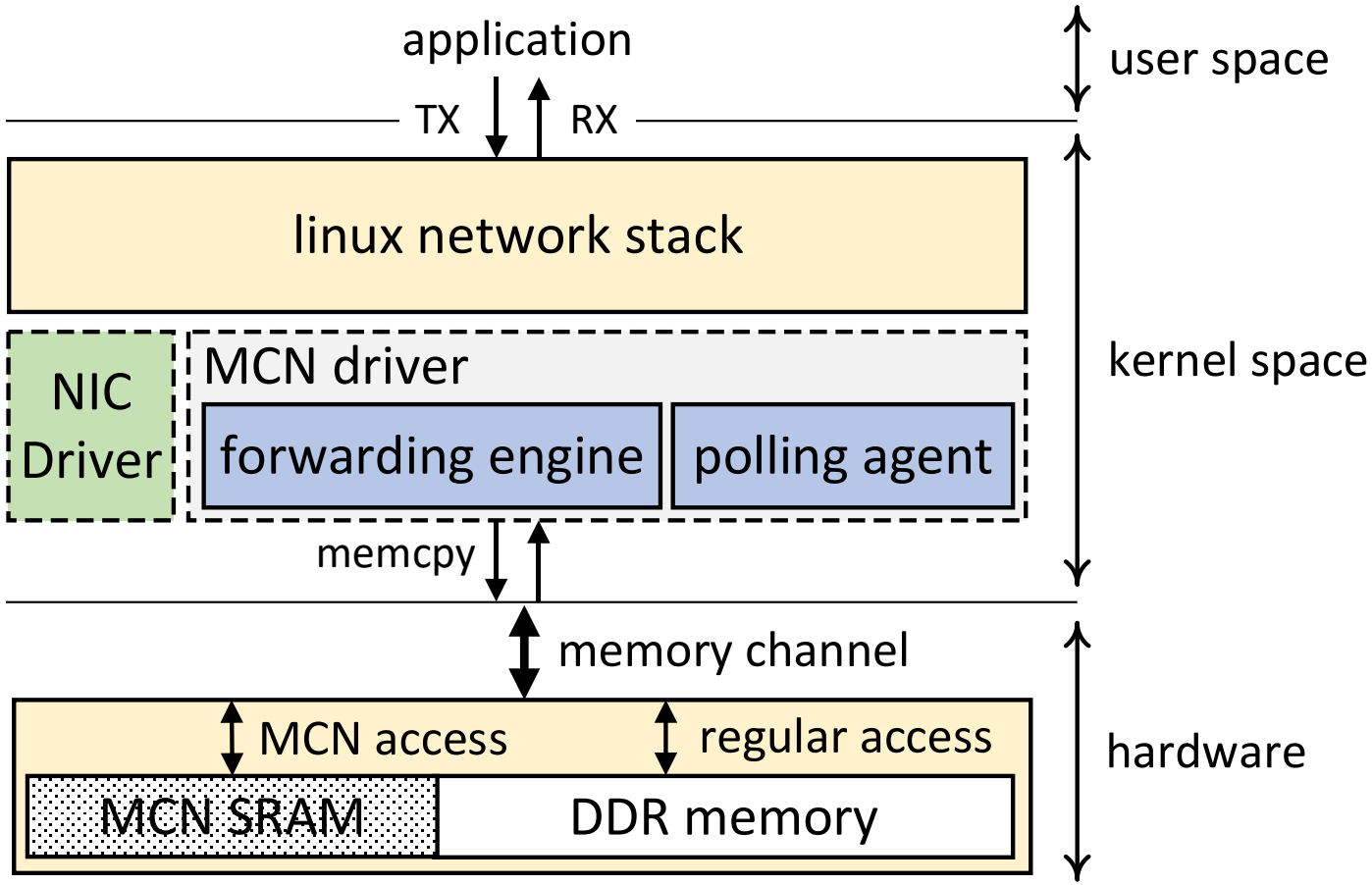


Mapped to a range of physical memory space directly accessed by MC like normal DRAM

MCN Interface

Serving as a fake network Interface card (NIC)

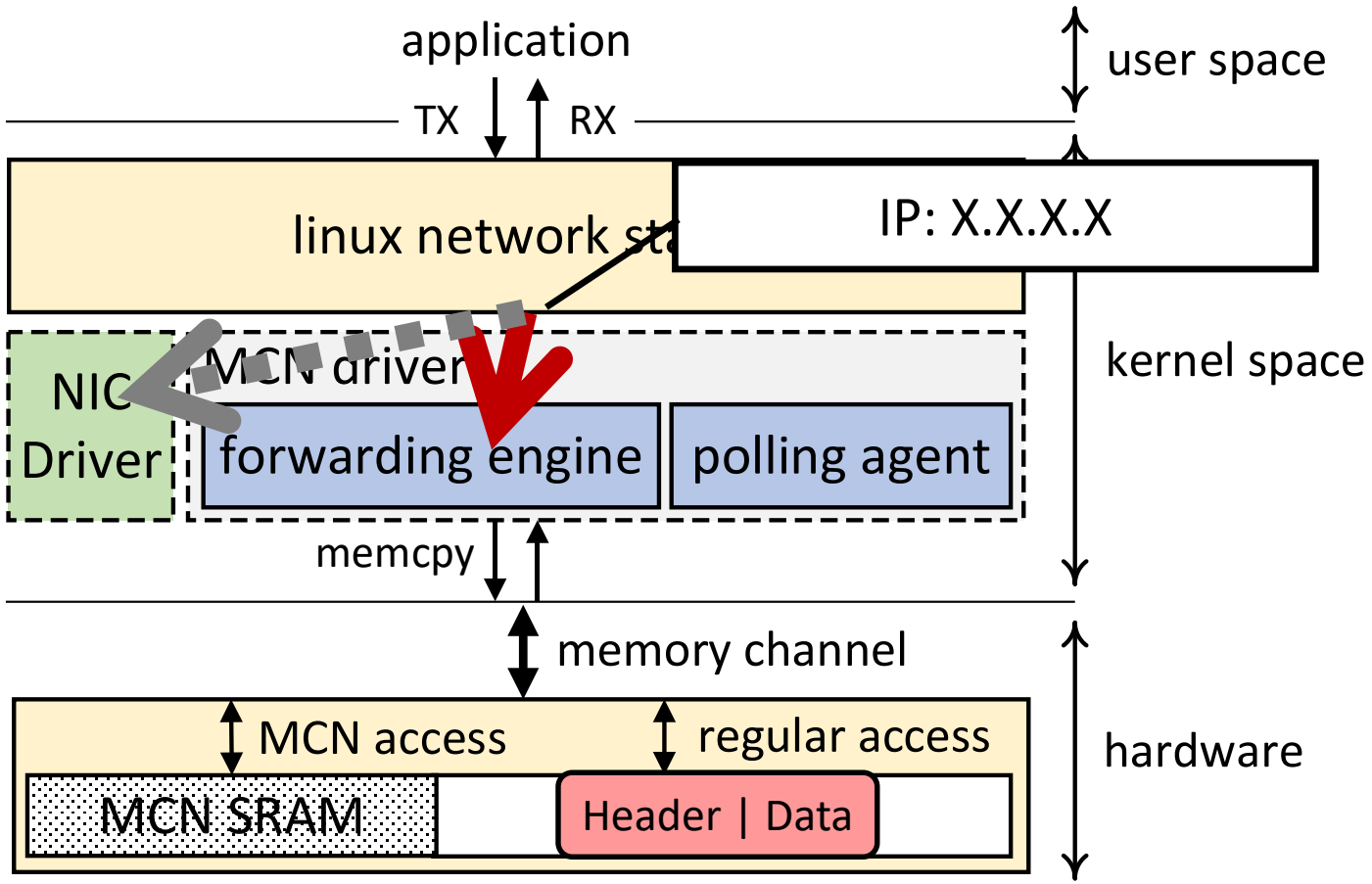
MCN Driver

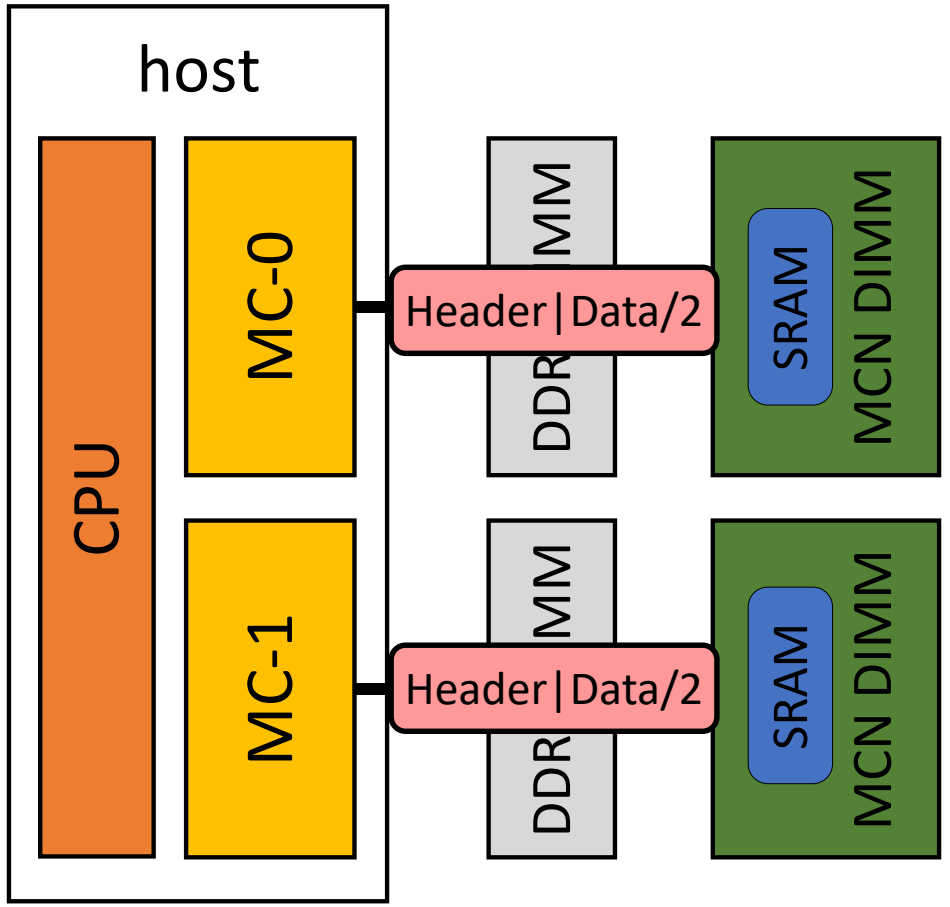

application
TX
RX
user space
linux network stack
NIC Driver
MCN driver
forwarding engine
polling agent
kernel space
memcpy
memory channel
MCN access
regular access
MCN SRAM
DDR memory
hardware

MCN Packet Routing

- Host → MCN

1. A packet is passed from the host network stack and the packet goes to the corresponding MCN DIMM or NIC

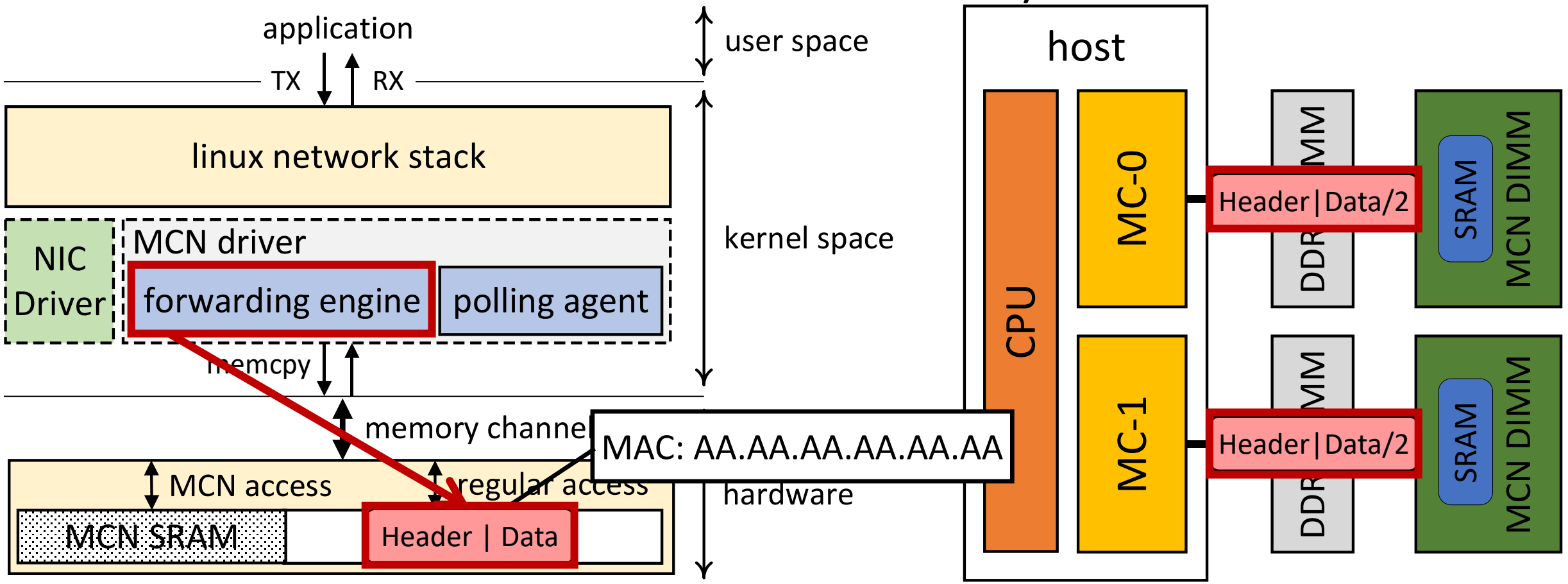




MCN Packet Routing

- Host → MCN

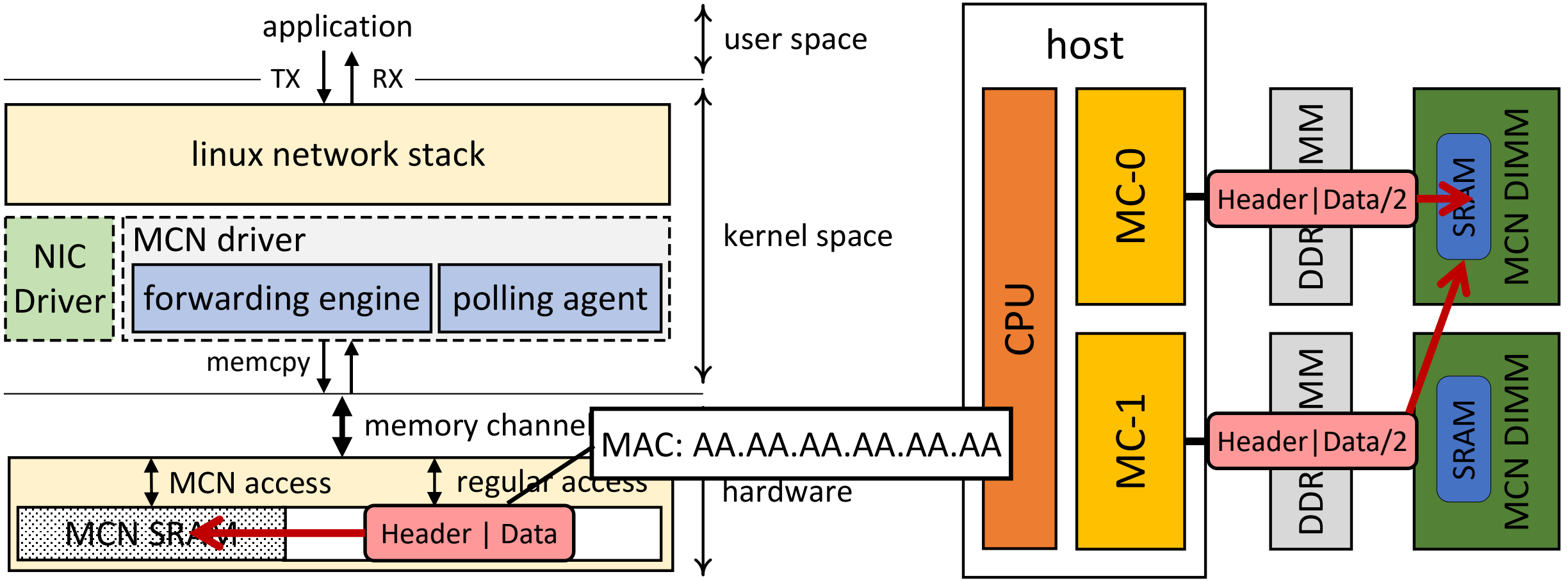
2. If the packet needs to be sent to an MCN DIMM, the forwarding engine checks the MAC of the packet stored in main memory



MCN Packet Routing

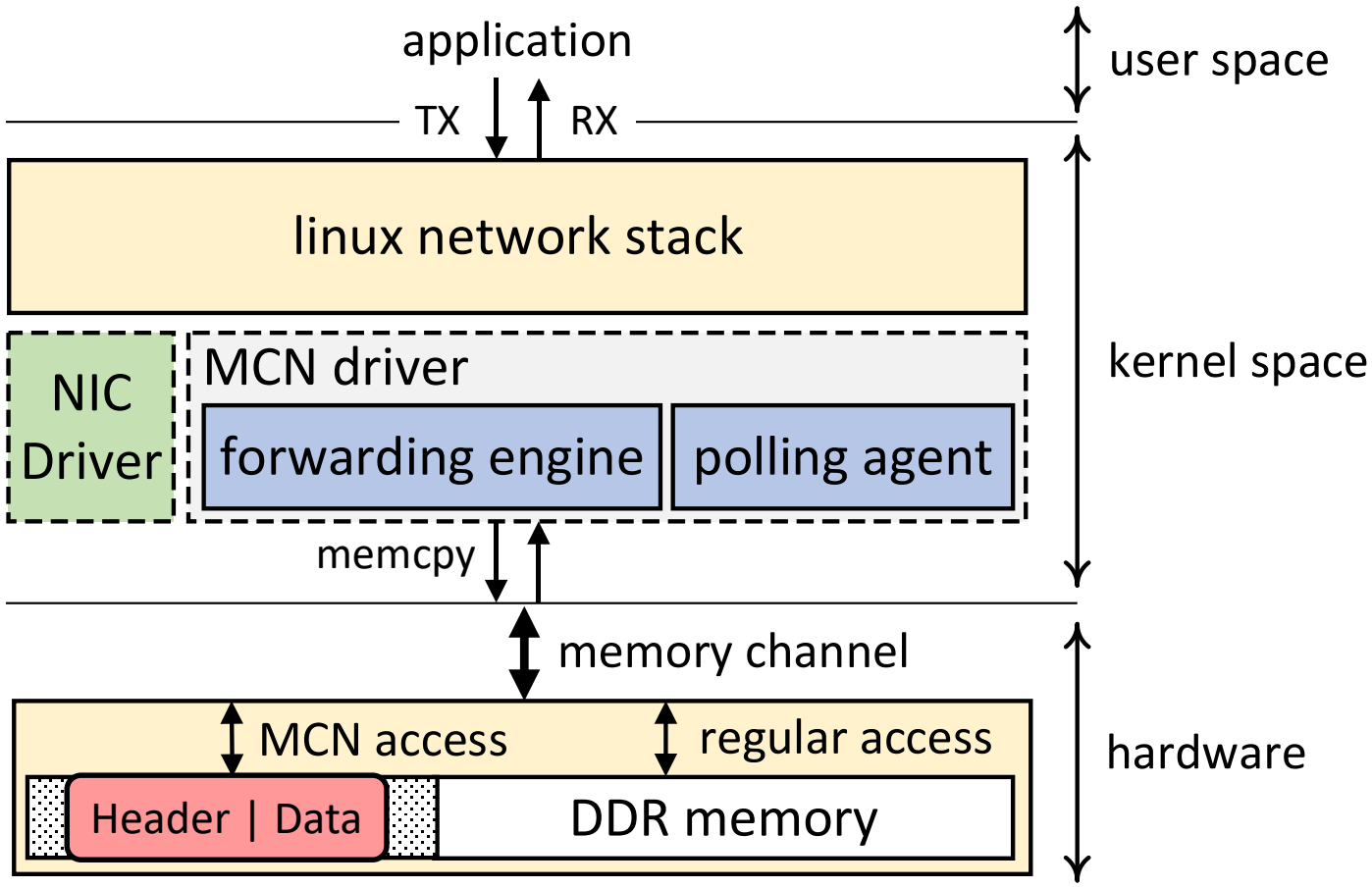
- Host → MCN

3. If the MAC matches w/ that of an MCN DIMM, the packet is copied to the SRAM buffer of the MCN DIMM

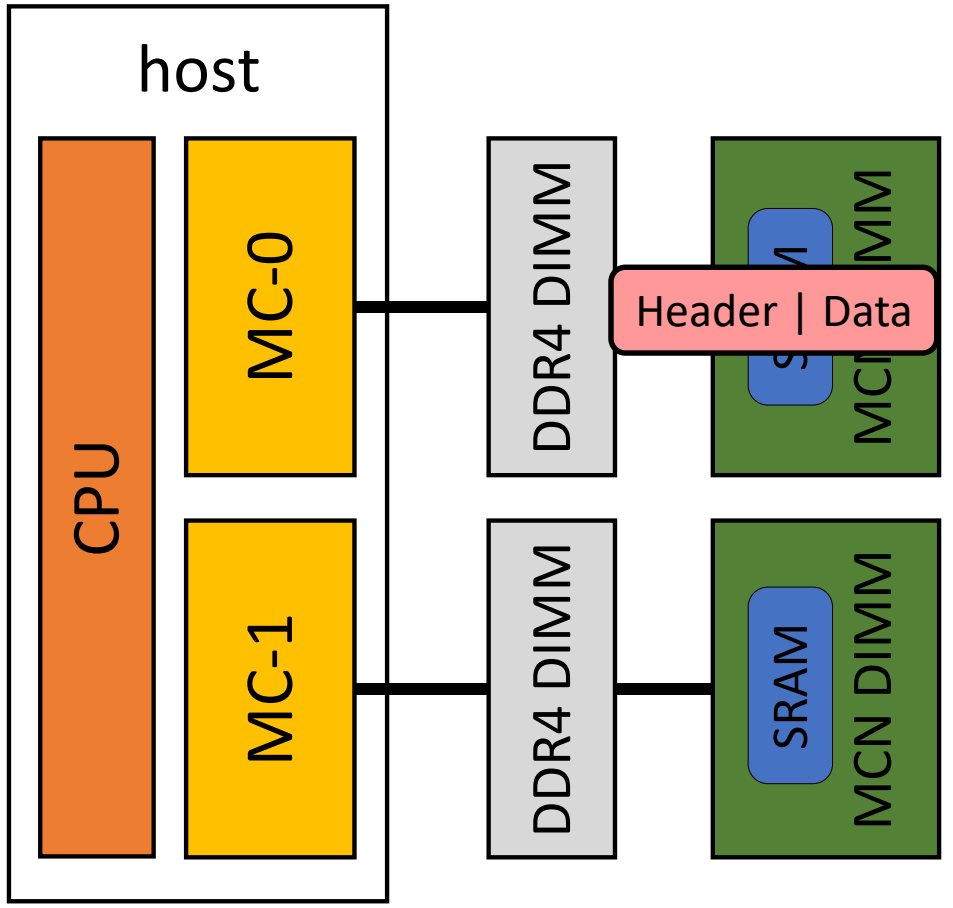


MCN Packet Routing

- Host → MCN



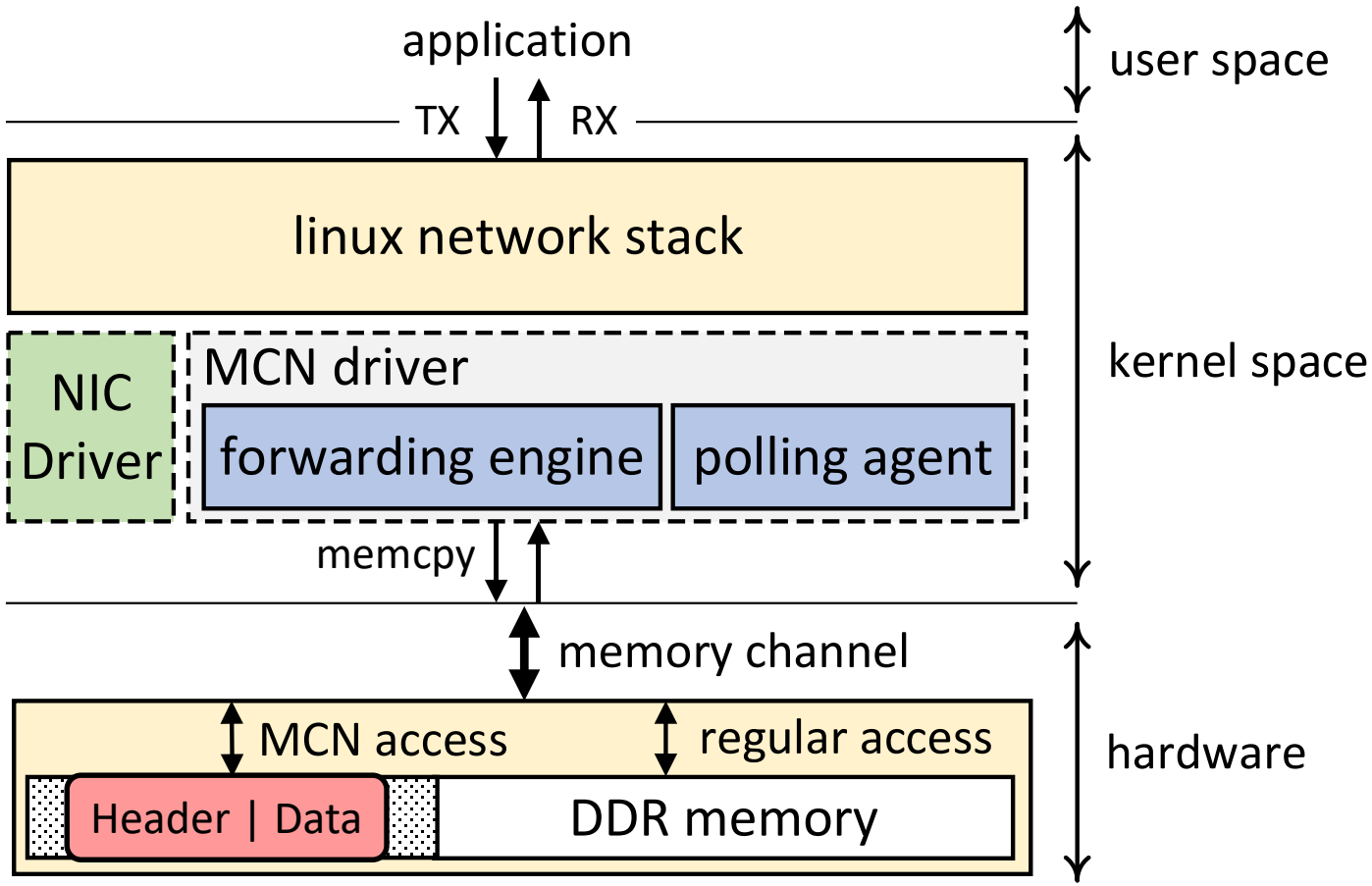
4. This data copy triggers IRQ from the MCN DIMM and MCN processor knows there is an arrived packet

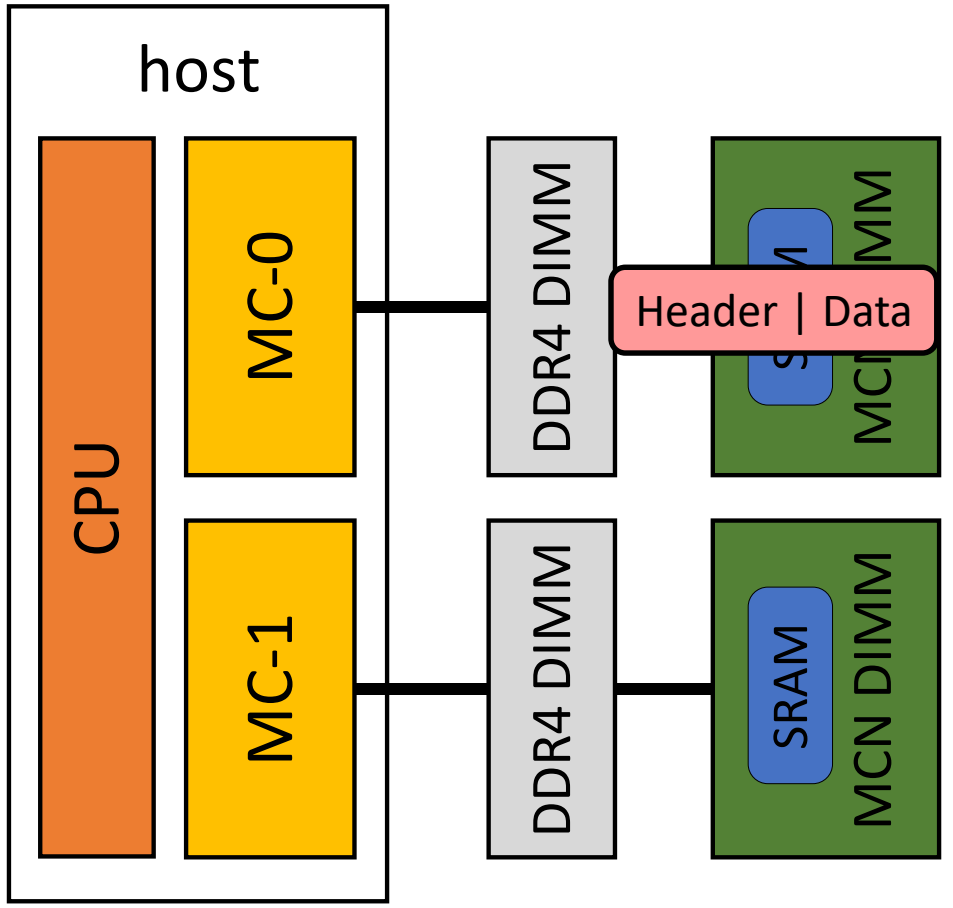


MCN Packet Routing

- MCN → Host

1. MCN DIMM writes a packet to its SRAM buffer and set the corresponding TX-poll bit

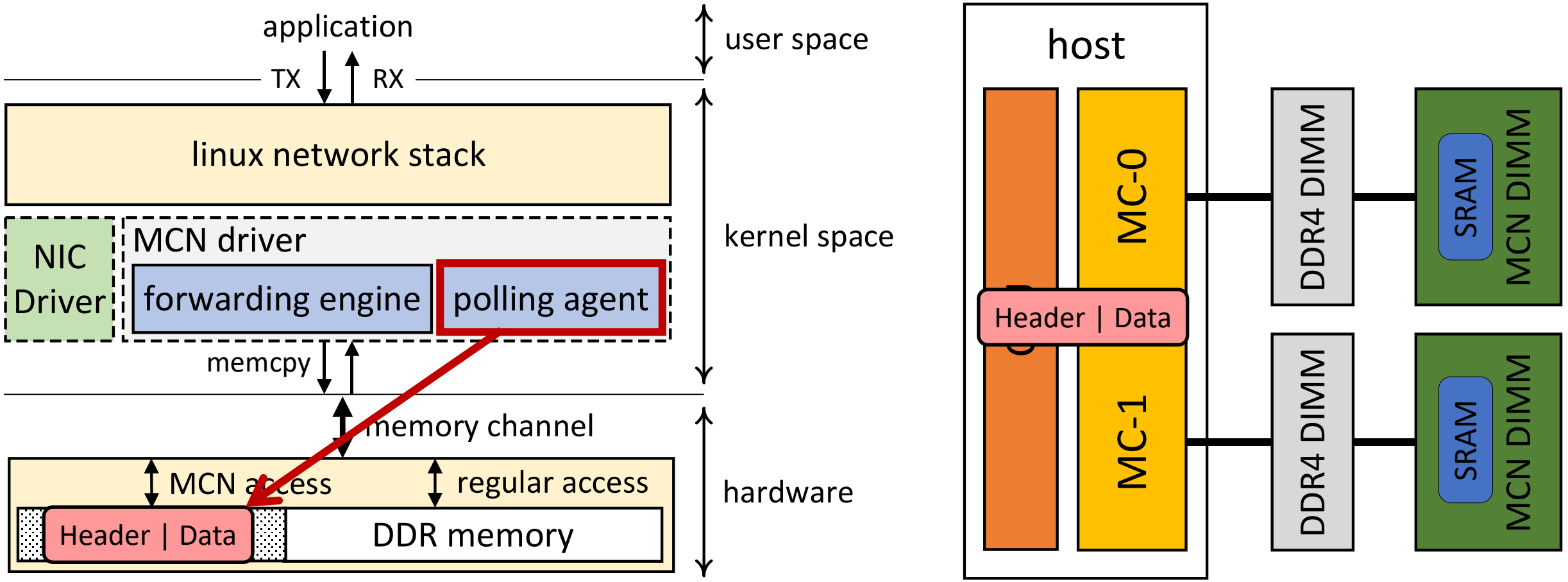




MCN Packet Routing

- MCN → Host

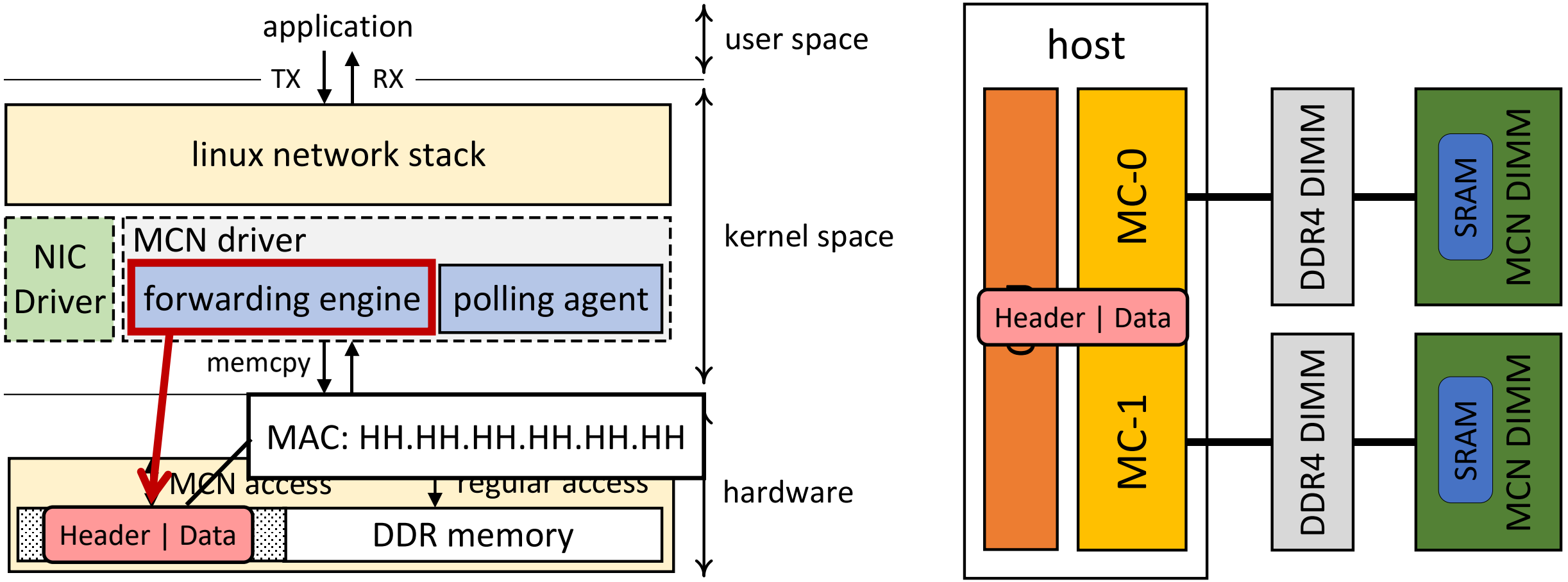
2. Polling agent from the host recognize the TX-poll bit is set and read the packet from the SRAM buffer



MCN Packet Routing

- MCN → Host

3. Forwarding engine checks the MAC address and determine the destination

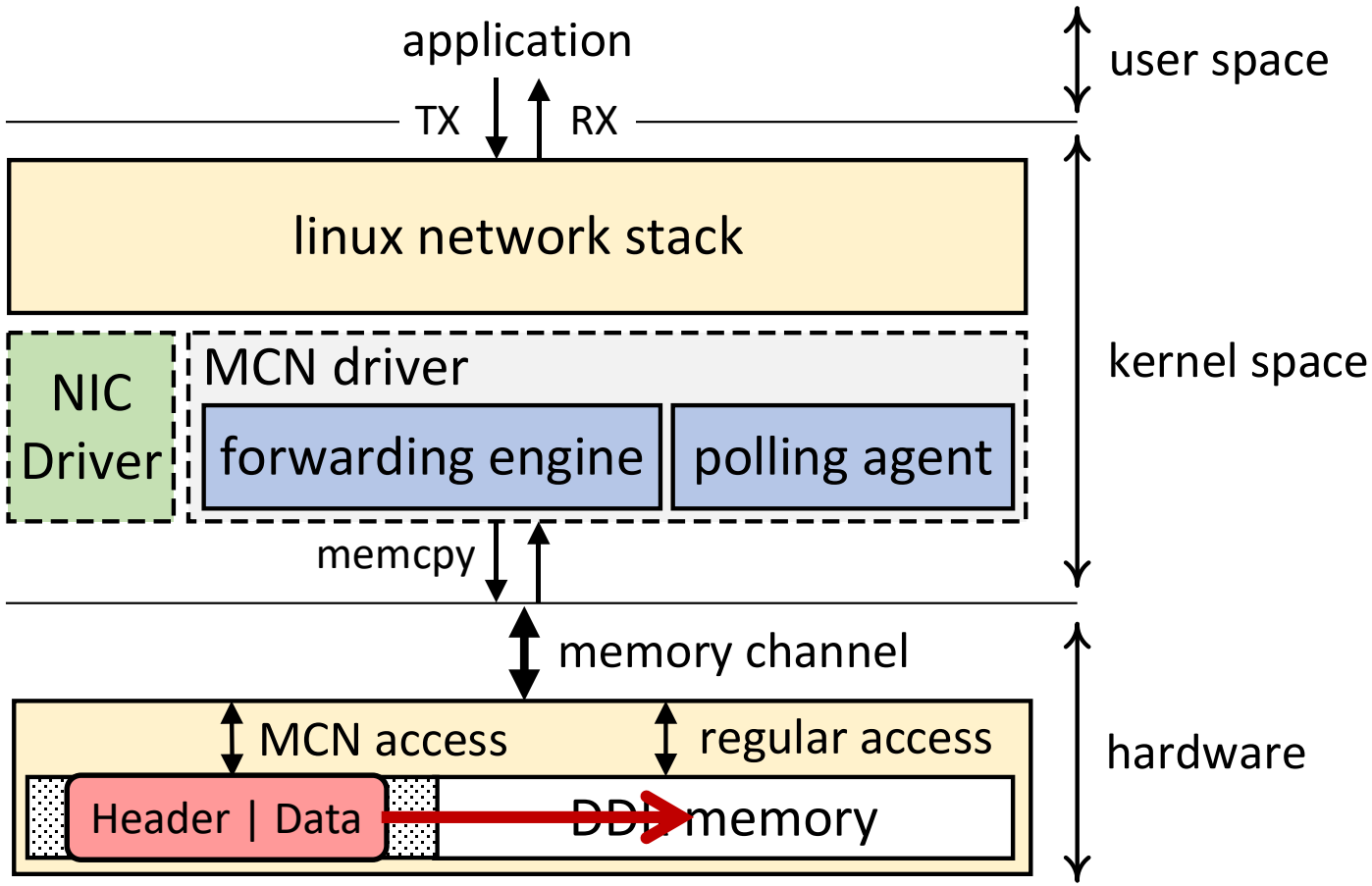


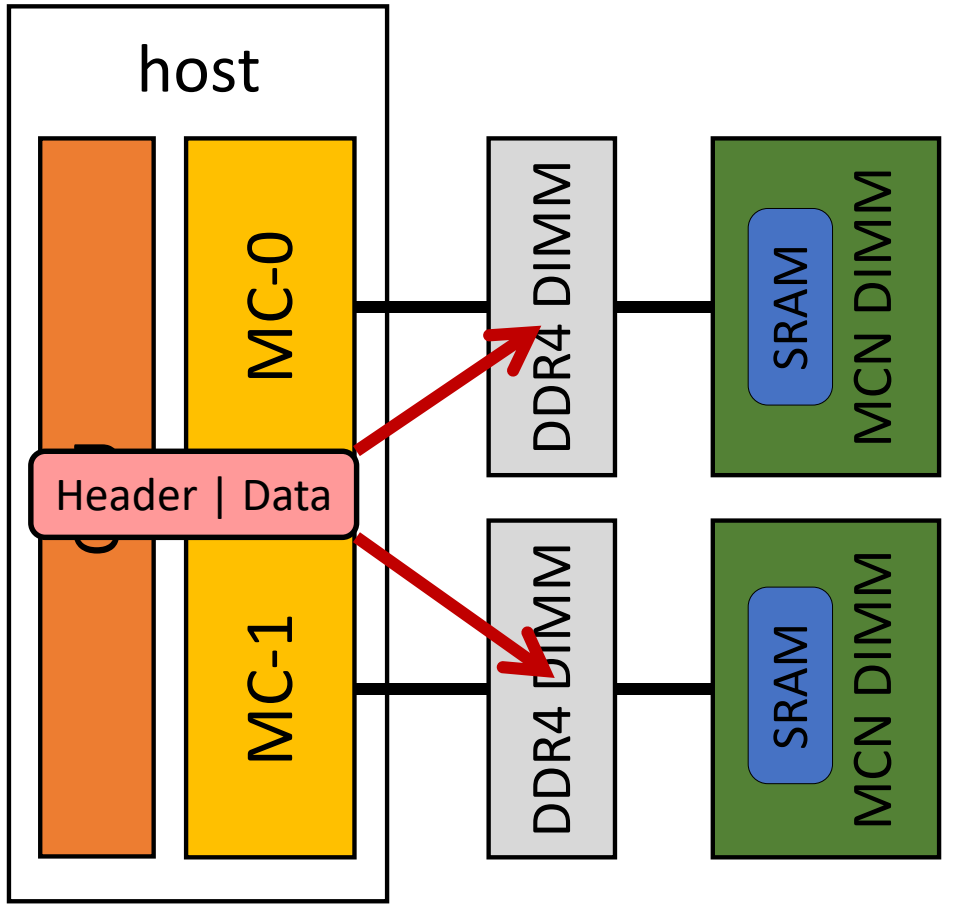
MCN Packet Routing

- MCN → Host

4. If this is for the host, it copies the packet to the host *skb* which in the host main memory

*network socket buffer



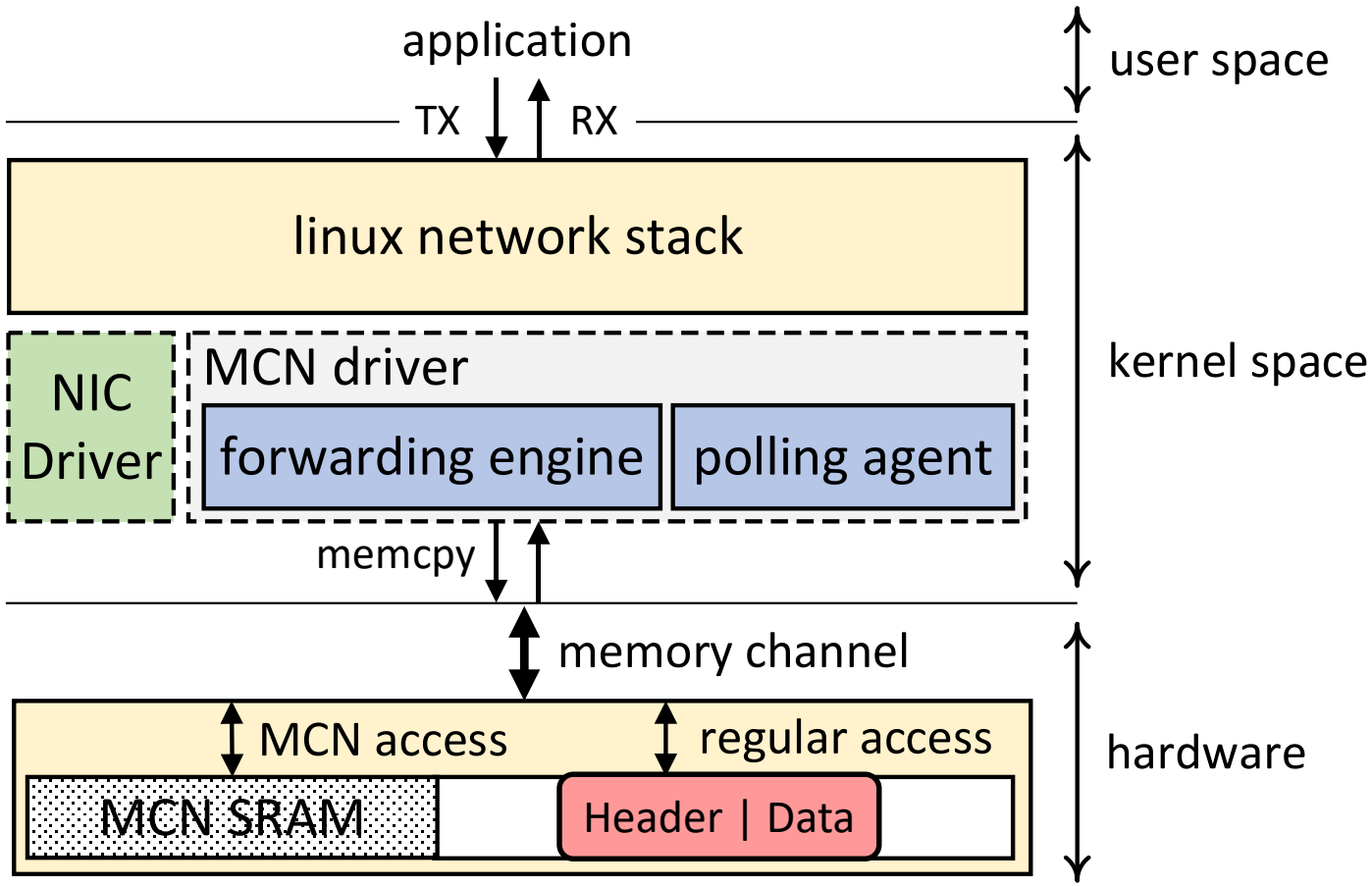


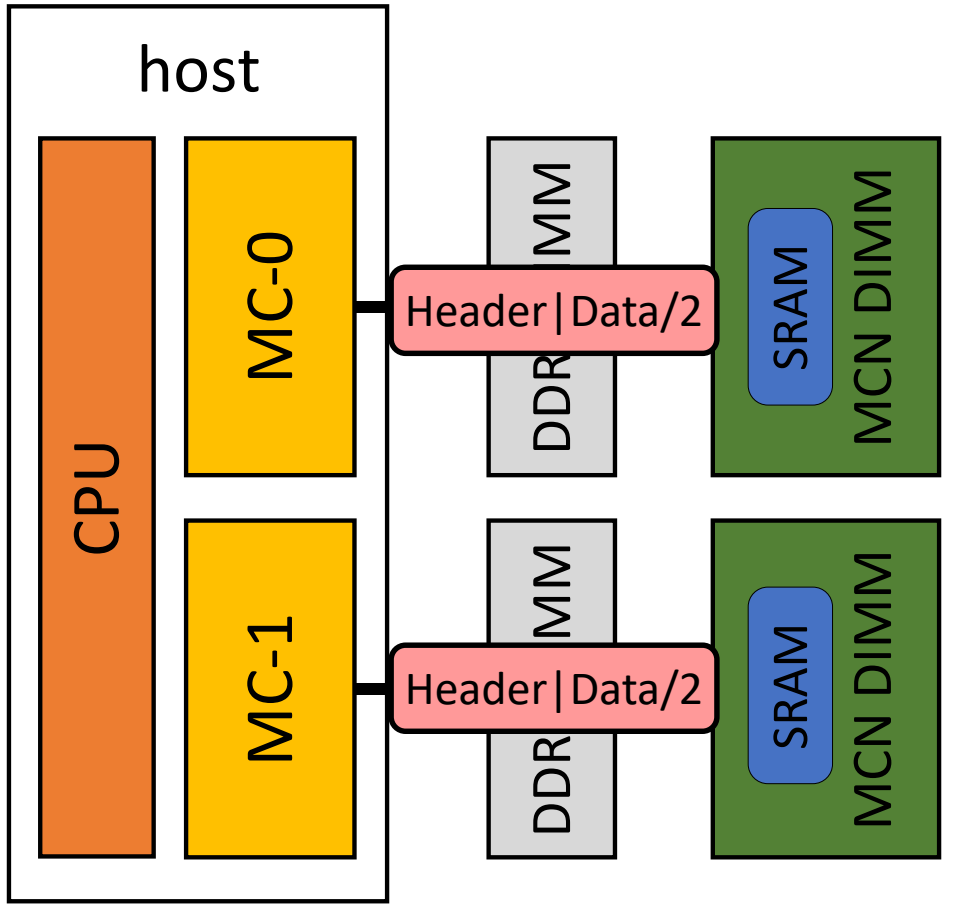
MCN Packet Routing

- MCN → Host

4. If this is for the host, it copies the packet to the host *skb* which in the host main memory

*network socket buffer

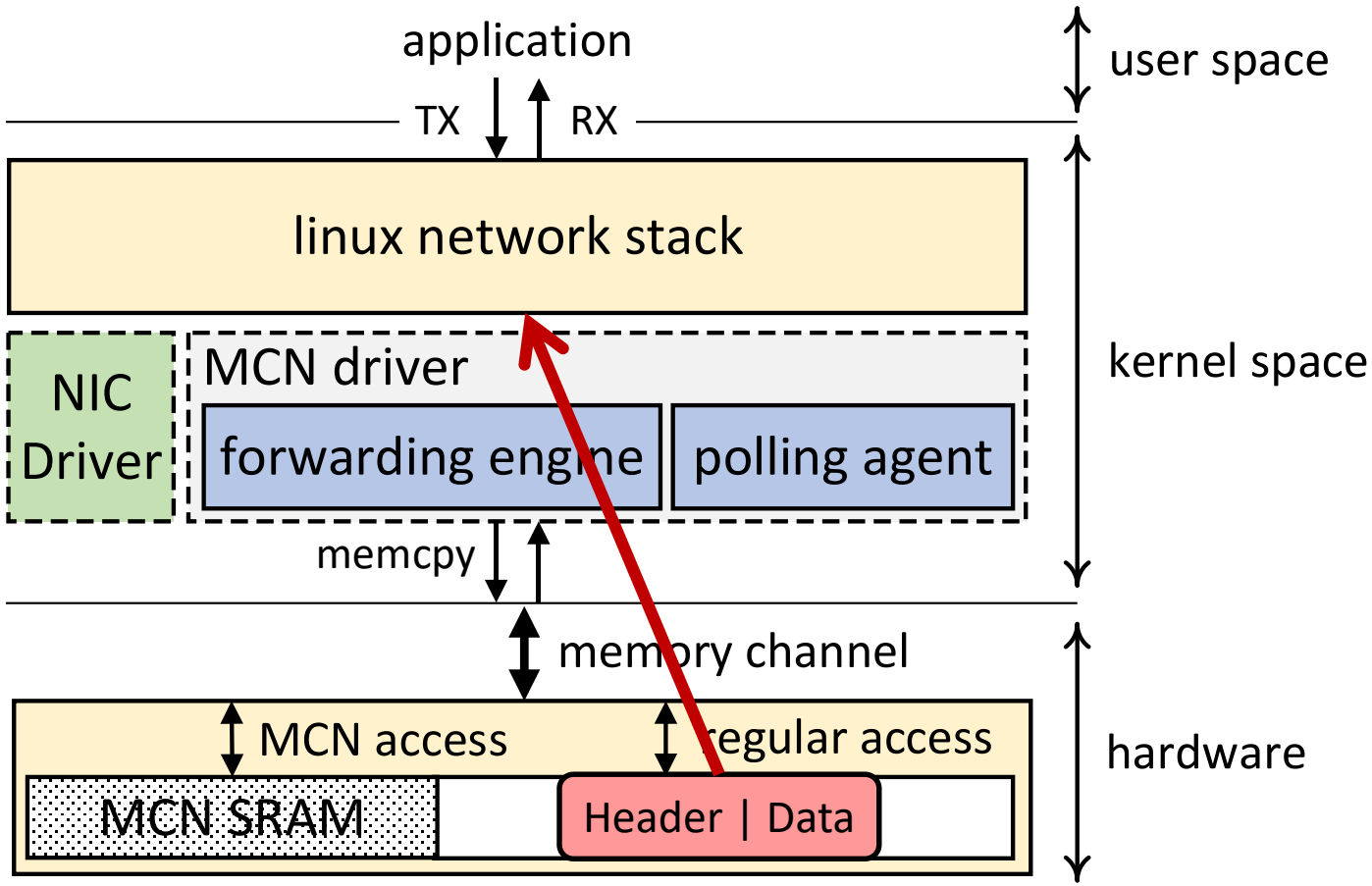


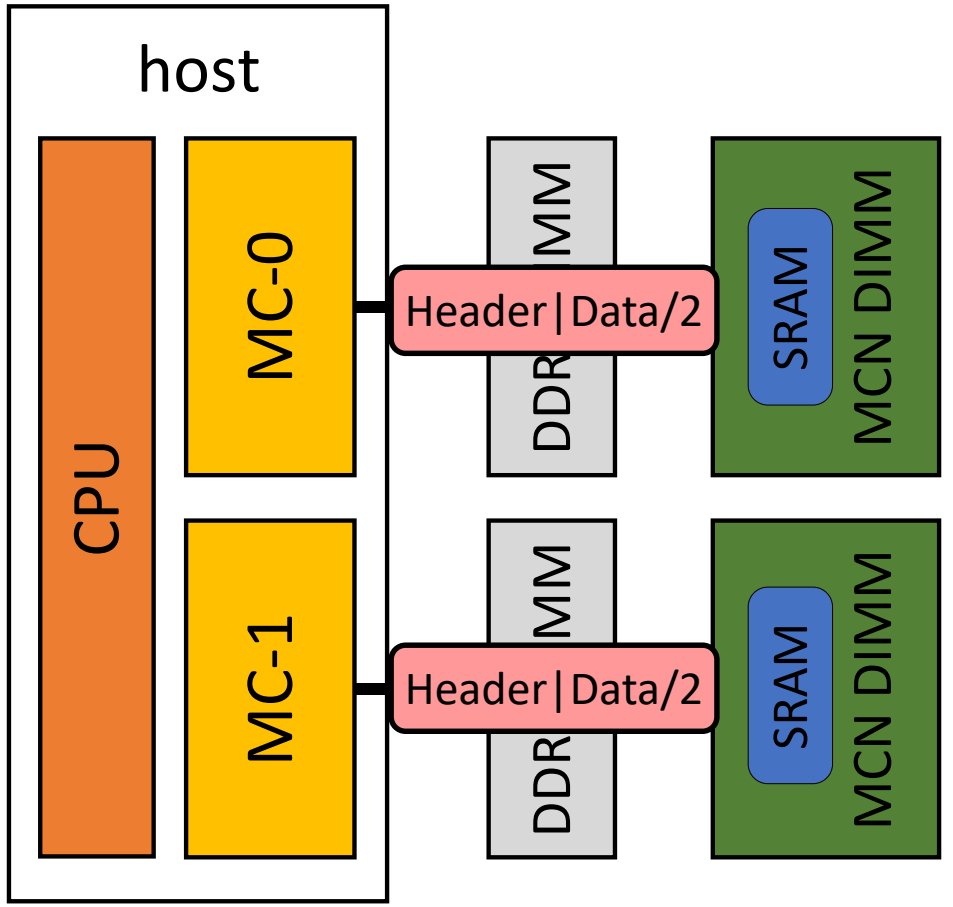


MCN Packet Routing

- MCN → Host

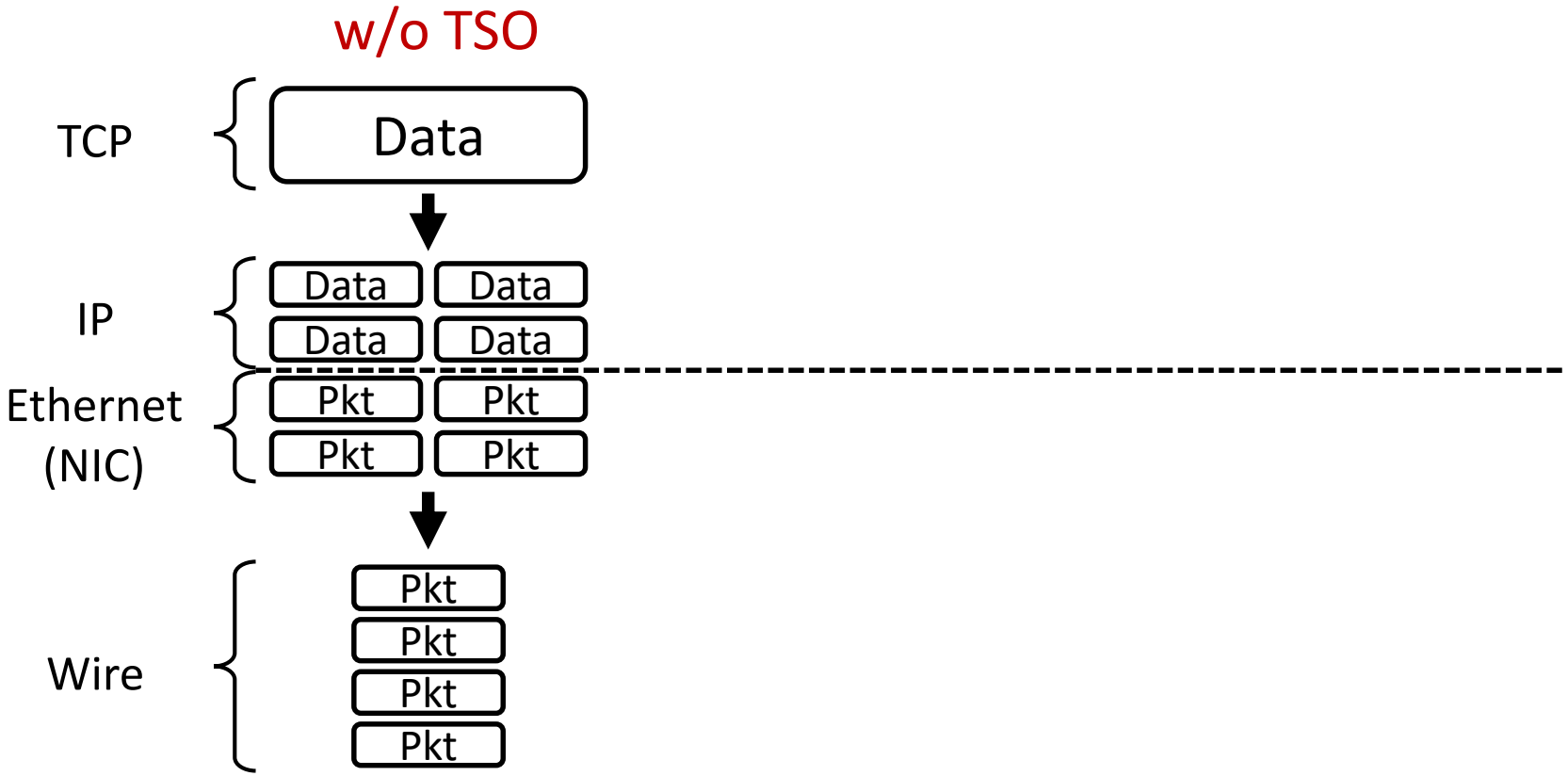
5. Finally the packet is passed to the host network stack (e.g., TCP/IP)





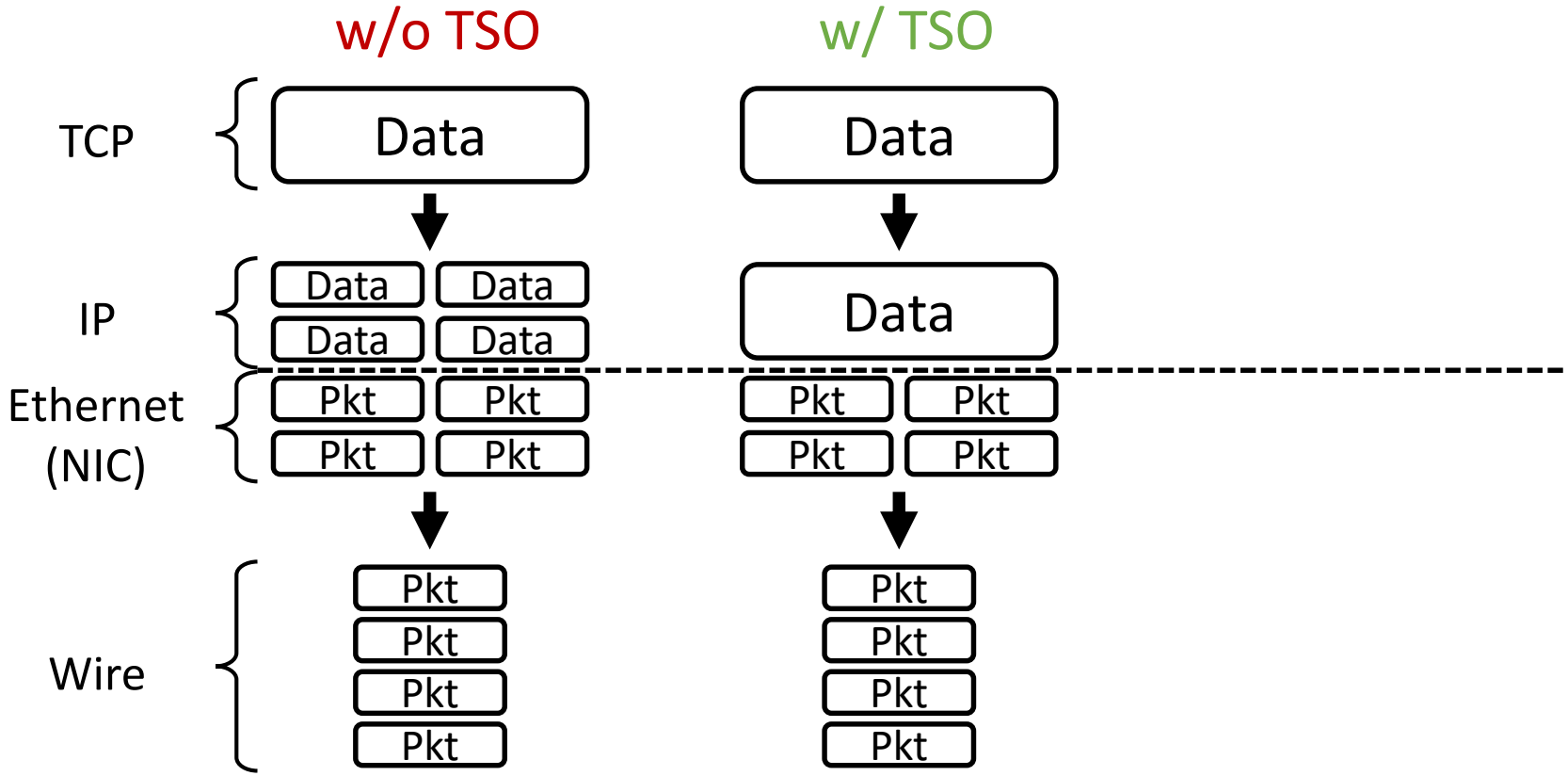
Optimizations

- Adopt optimizations from conventional Ethernet interfaces
  - ✓Offload (or remove) packet fragmentation (e.g. TCP Segmentation Offload (TSO))



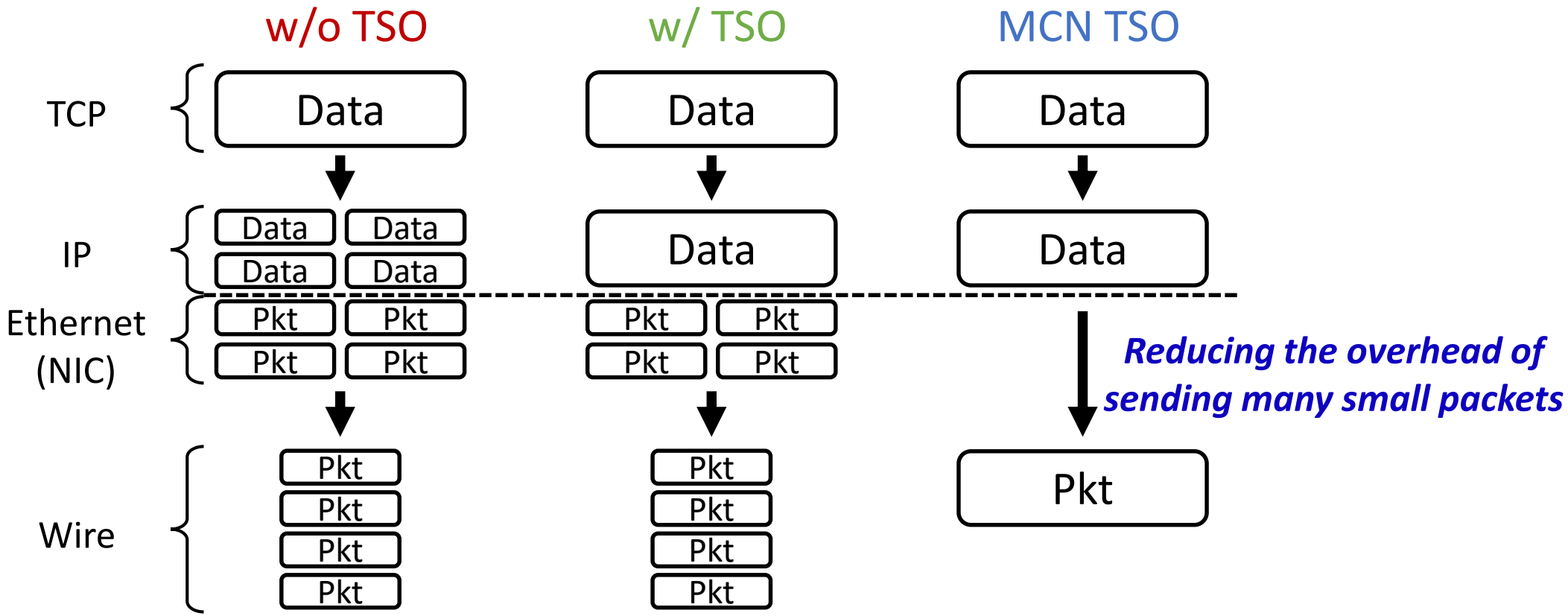
Optimizations

- Adopt optimizations from conventional Ethernet interfaces
  - ✓Offload (or remove) packet fragmentation (e.g. TCP Segmentation Offload (TSO))



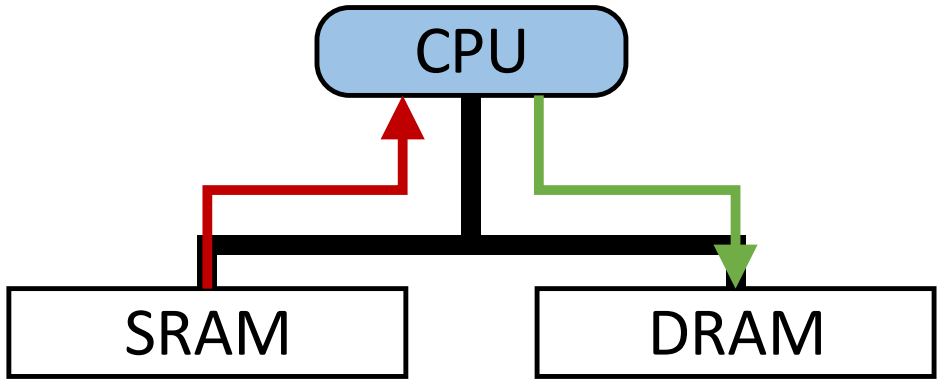
Optimizations

- Adopt optimizations from conventional Ethernet interfaces
  - ✓Offload (or remove) packet fragmentation (e.g. TCP Segmentation Offload (TSO))

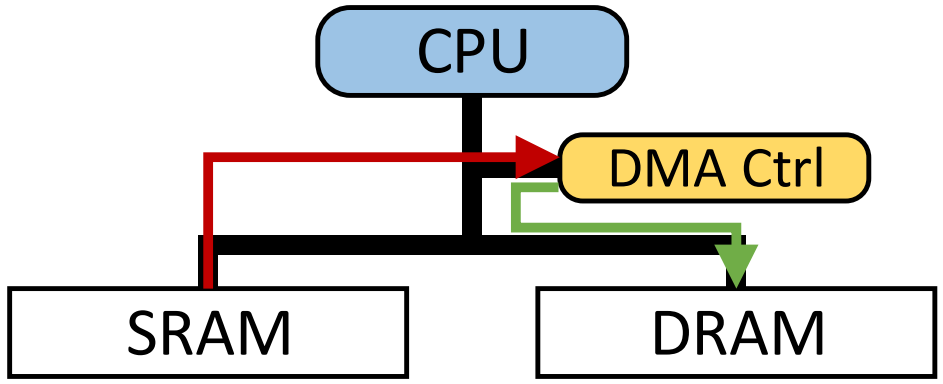


Optimizations

- Adopt optimizations from conventional Ethernet interfaces
  - ✓Offload (or remove) packet fragmentation (e.g. TCP Segmentation Offload (TSO))
- MCN-side DMA
  - ✓Baseline MCN processor manually copies data b/w SRAM and DRAM
  - ✓SRAM-to-DRAM DMA eliminates CPU *memcpy* overhead similar to NIC-to-DRAM DMA in conventional systems

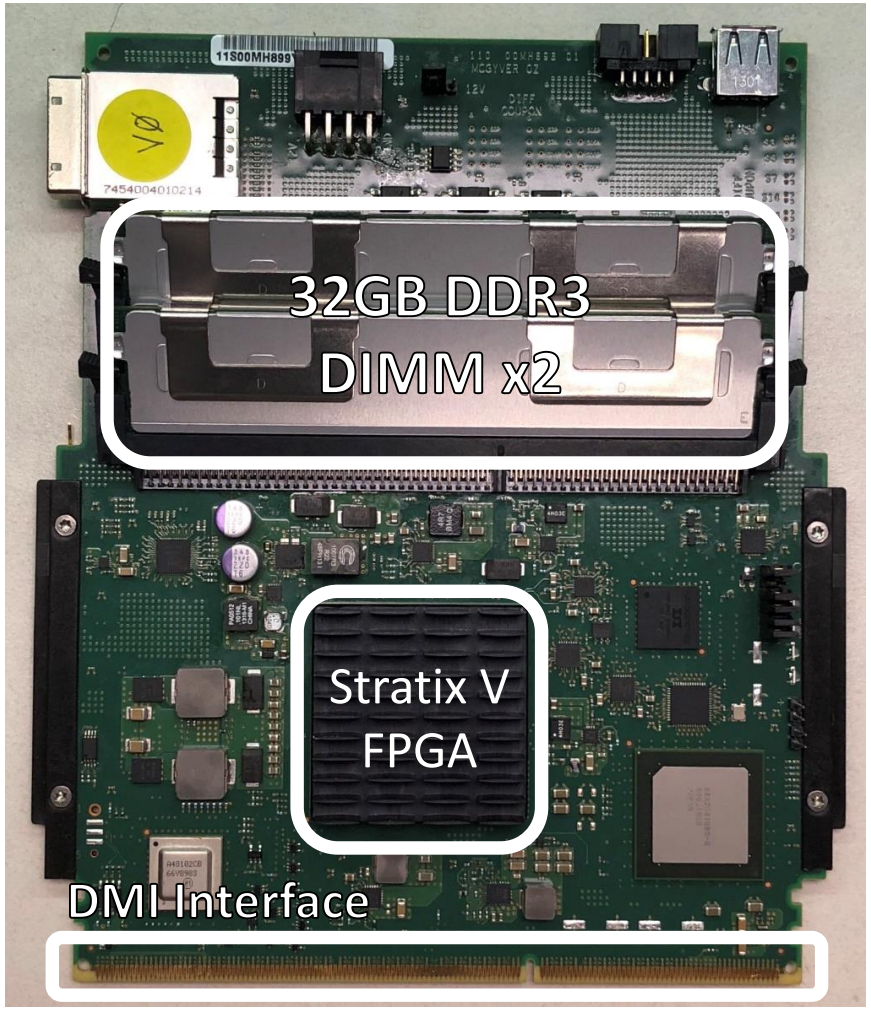


MCN Baseline

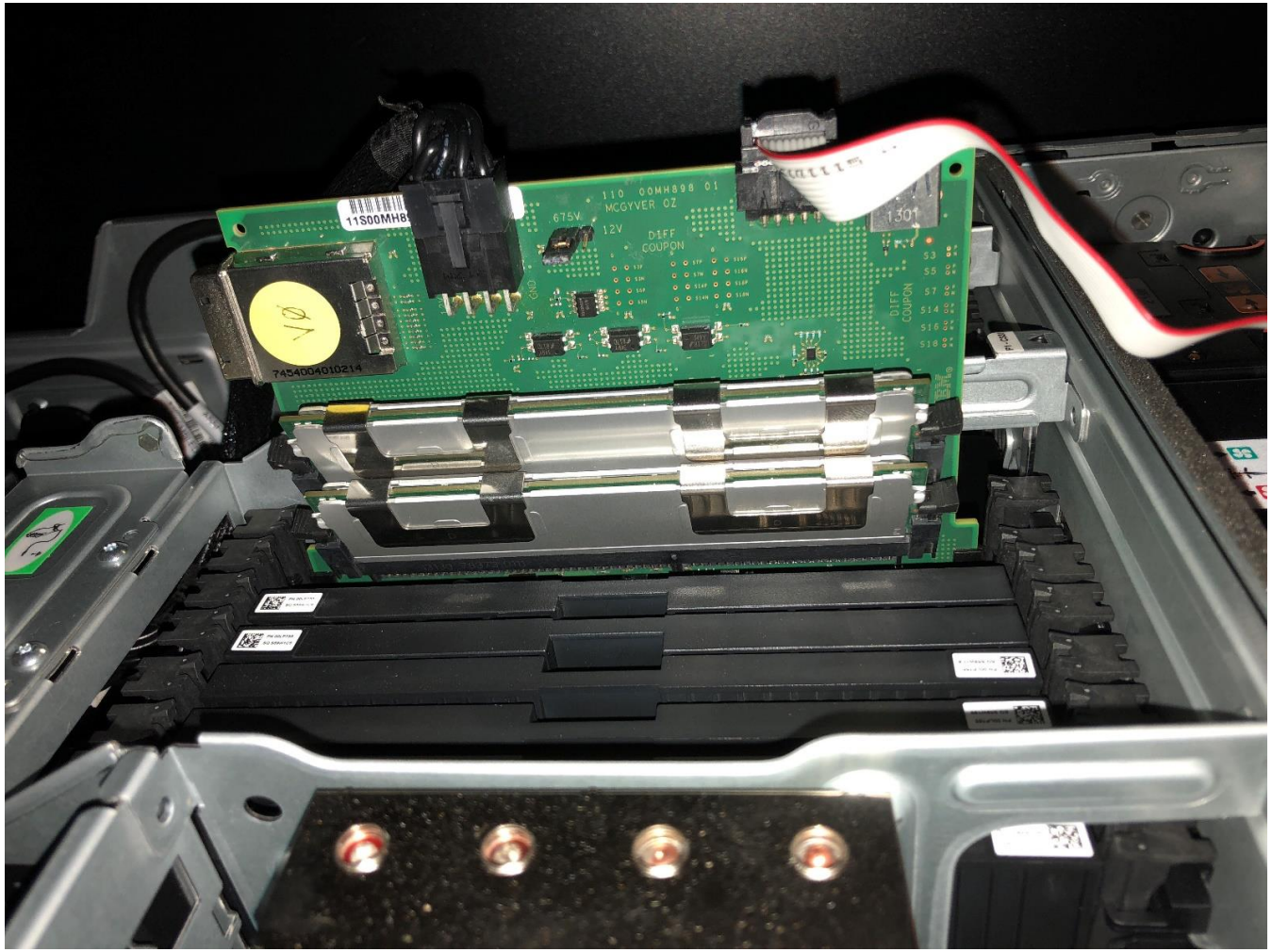


MCN w/ DMA

Proof of Concept HW/SW Demonstration



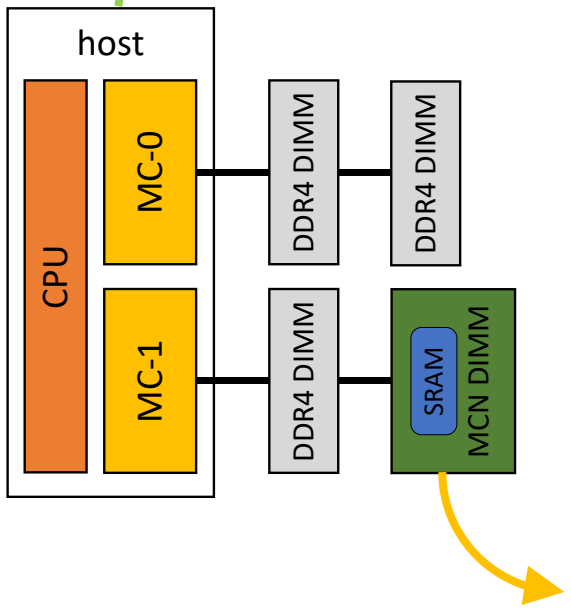
ConTutto top view



Contutto plugged in IBM POWER8 server

Proof of Concept HW/SW Demonstration

MPI case




```
power8@tuleta2-contutto:~$ mpirun --mca btl_base_warn_component
_unused 0 -np 2 -host power8@nios,tuleta /home/power8/mpihello_
300
Hello world from processor tuleta2-contutto, rank 0 out of 2 pr
ocessors
Hello world from processor nios2, rank 1 out of 2 processors
power8@tuleta2-contutto:~$ mpirun --mca btl_base_warn_component
_unused 0 -np 2 -host power8@nios,tuleta /home/power8/mpisendte
st_300
Machine tuleta2-contutto generated random number: 1804289383
Machine tuleta2-contutto sent 1804289383
processor nios2  got 1804289383
power8@tuleta2-contutto:~$
```

POWER8 (Host)

```
 length 0
00:07:21.005542 IP nios.ssh > y700.42860: Flags [.], ack 3239,
win 1246, options [nop,nop,TS val 141004 ecr 216486], length 0
00:07:21.288293 IP nios.ssh > y700.42860: Flags [FP.], seq 3173
, ack 3239, win 1246, options [nop,nop,TS val 141288 ecr 216486
], length 0
00:07:21.305009 IP y700.42860 > nios.ssh: Flags [.], ack 3174,
win 308, options [nop,nop,TS val 216607 ecr 141288], length 0
```

ConTutto (MCN DIMM)

MPI application running through MCN

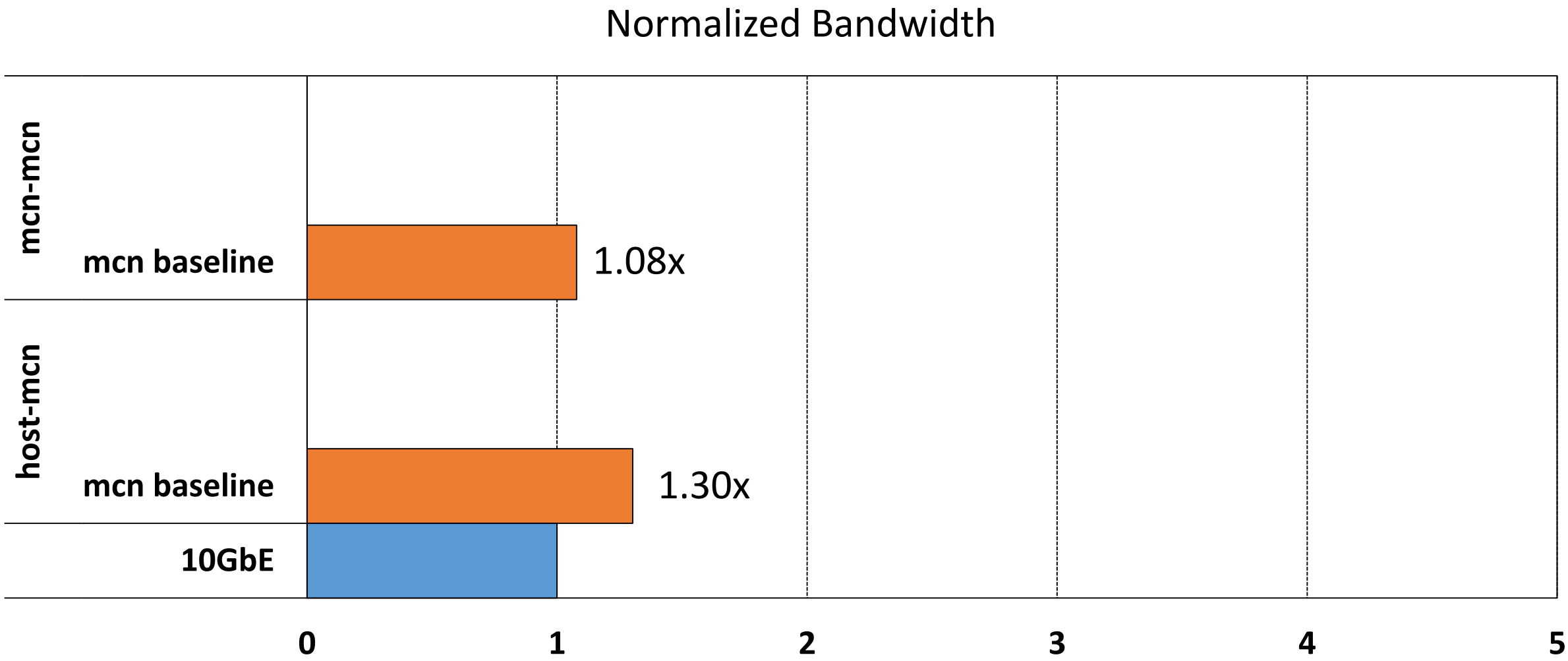
Evaluation Methodology

- Simulation → dist-gem5 (ISPASS'17)
- Network performance evaluation → iperf and ping
- Application performance evaluation → Coral, Bigdatabench and NPB

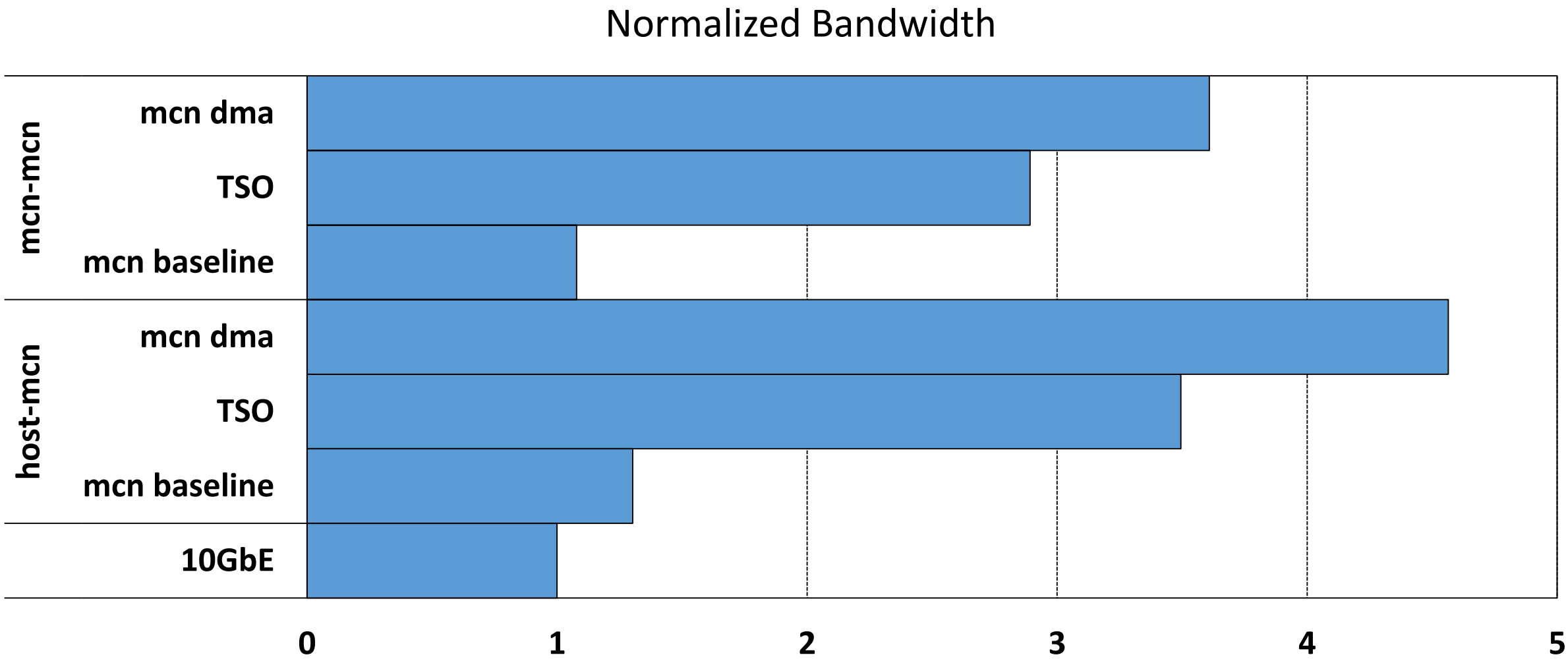
MCN System Configuration	
CPU	ARMv8 Quad Core running @ 2.45GHz
Caches	L1I: 32KB, L1D: 32KB, L2: 1MB
Memory	DDR4-3200
OS	Ubuntu 14.04

Host System Configuration	
CPU	ARMv8 Octa Core running @ 3.4GHz
Caches	L1I: 32KB, L1D: 32KB, L2: 256KB, L3: 8MB
Memory	DDR4-3200
NIC	10GbE/1μs link latency
OS	Ubuntu 14.04

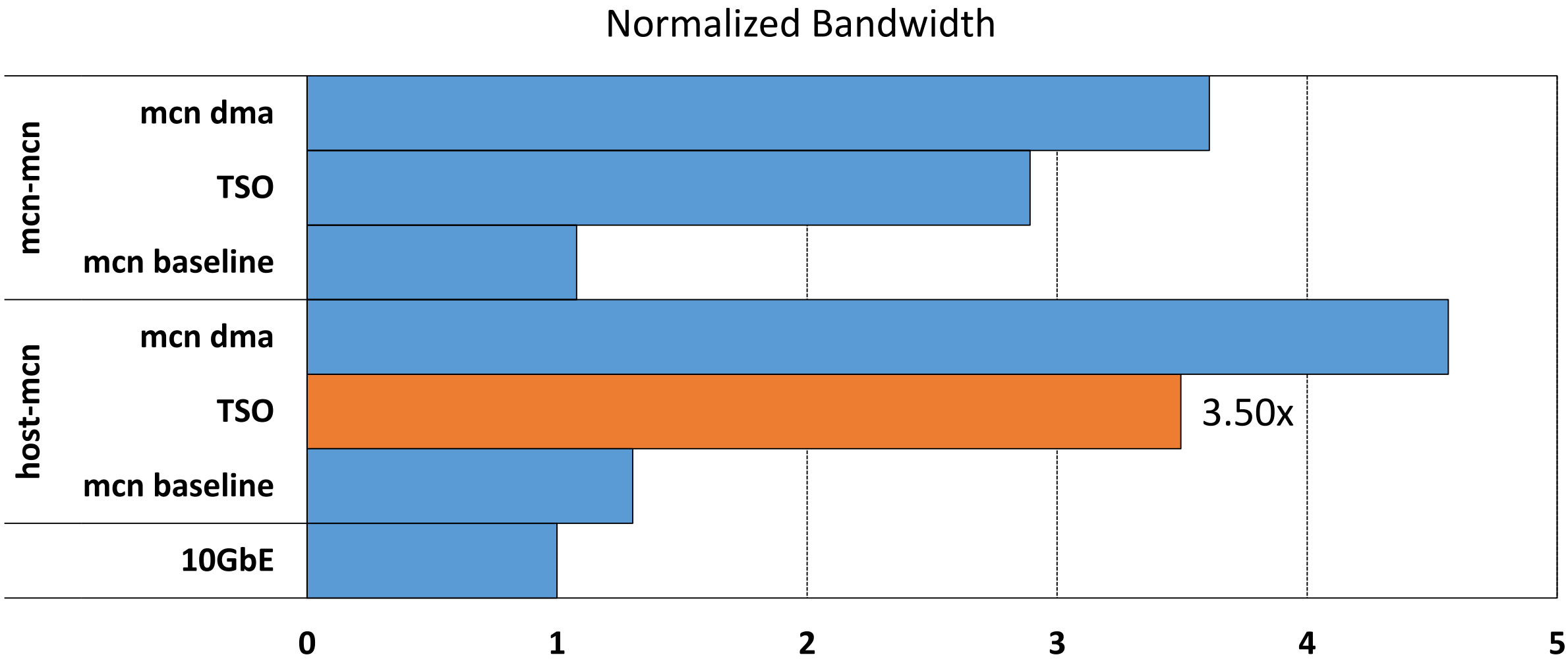
Evaluation – Network Bandwidth (iPerf)



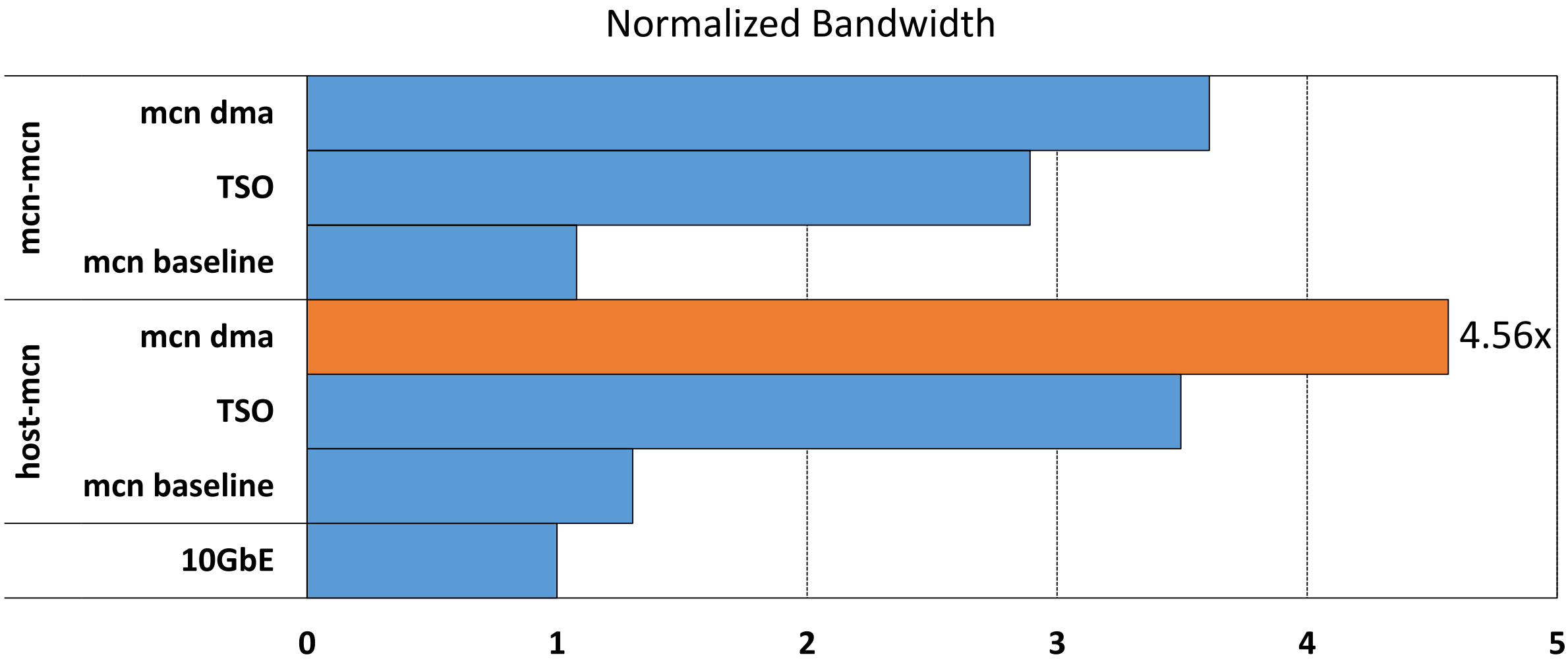
Evaluation – Network Bandwidth (iPerf)



Evaluation – Network Bandwidth (iPerf)

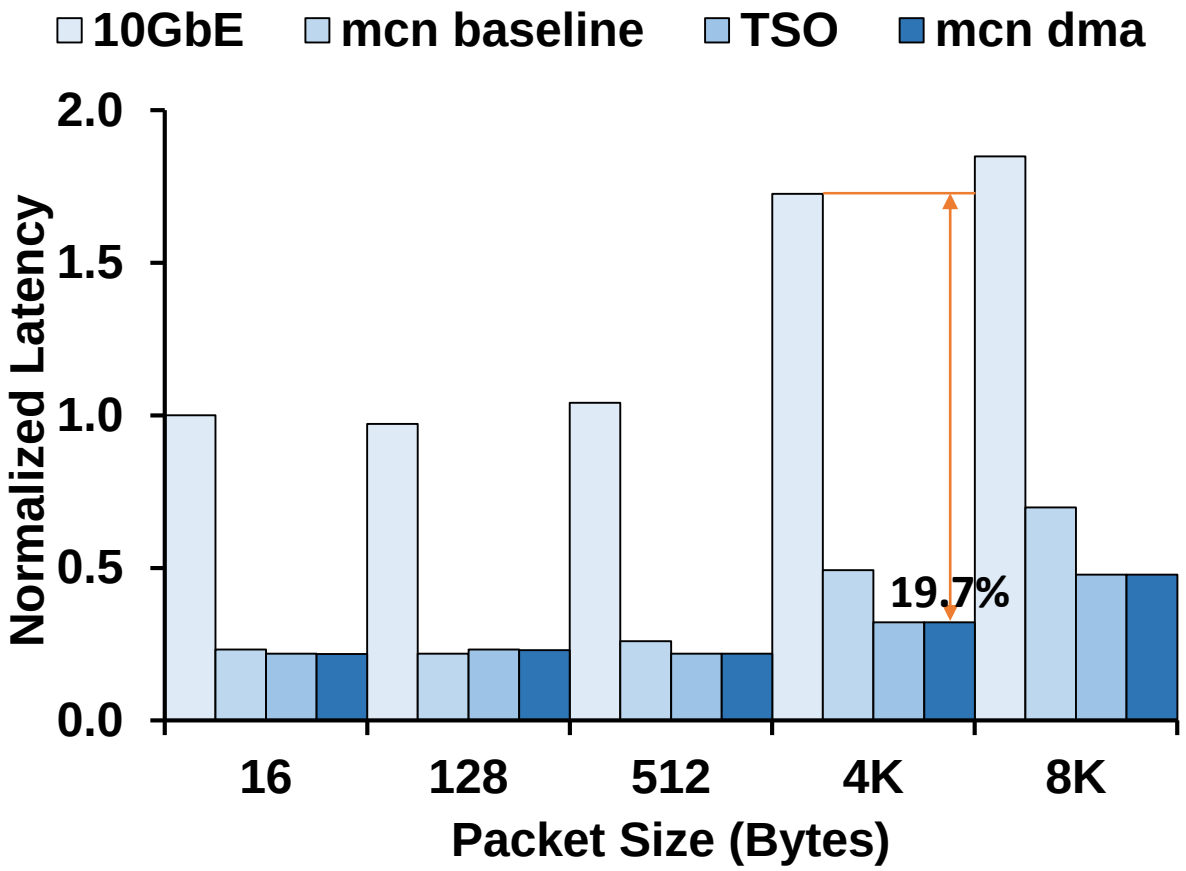


Evaluation – Network Bandwidth (iPerf)

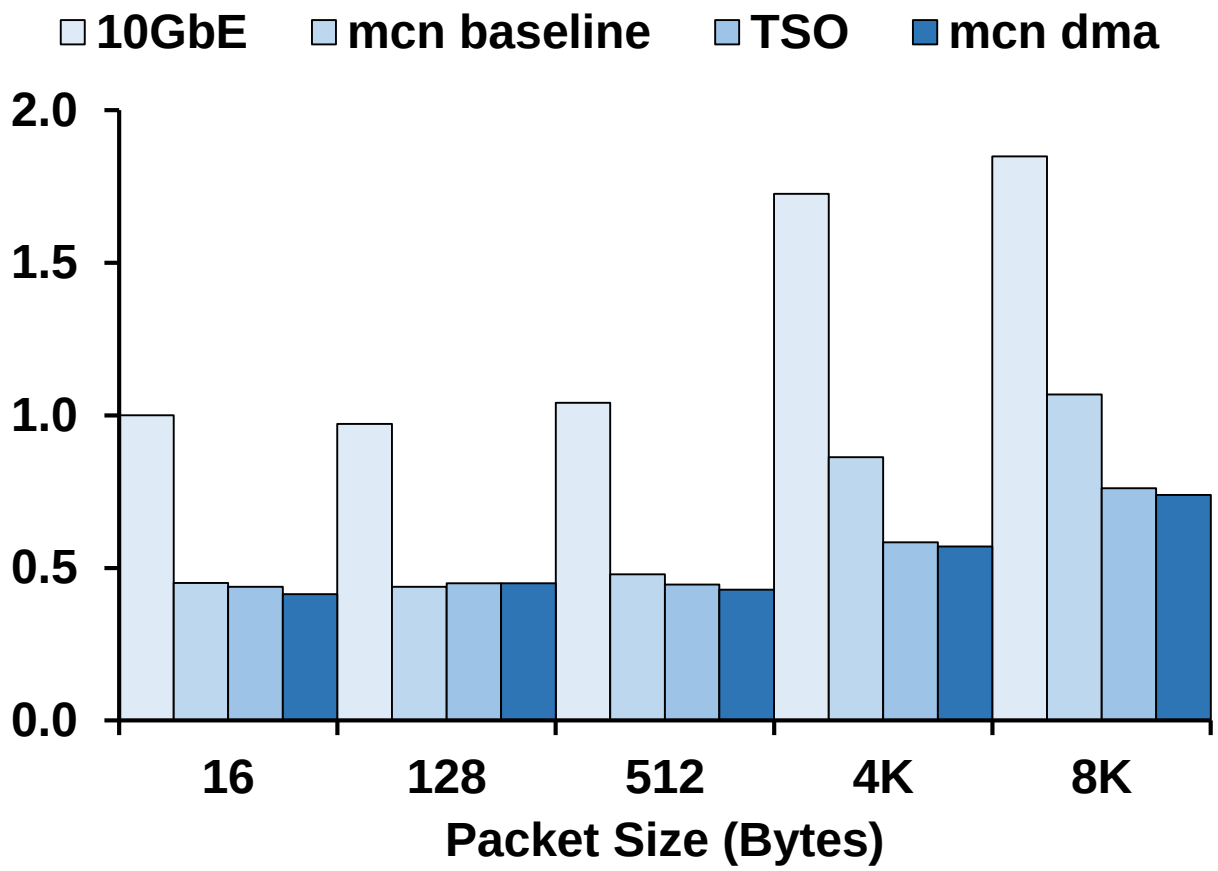


Evaluation – Network Latency (Ping)

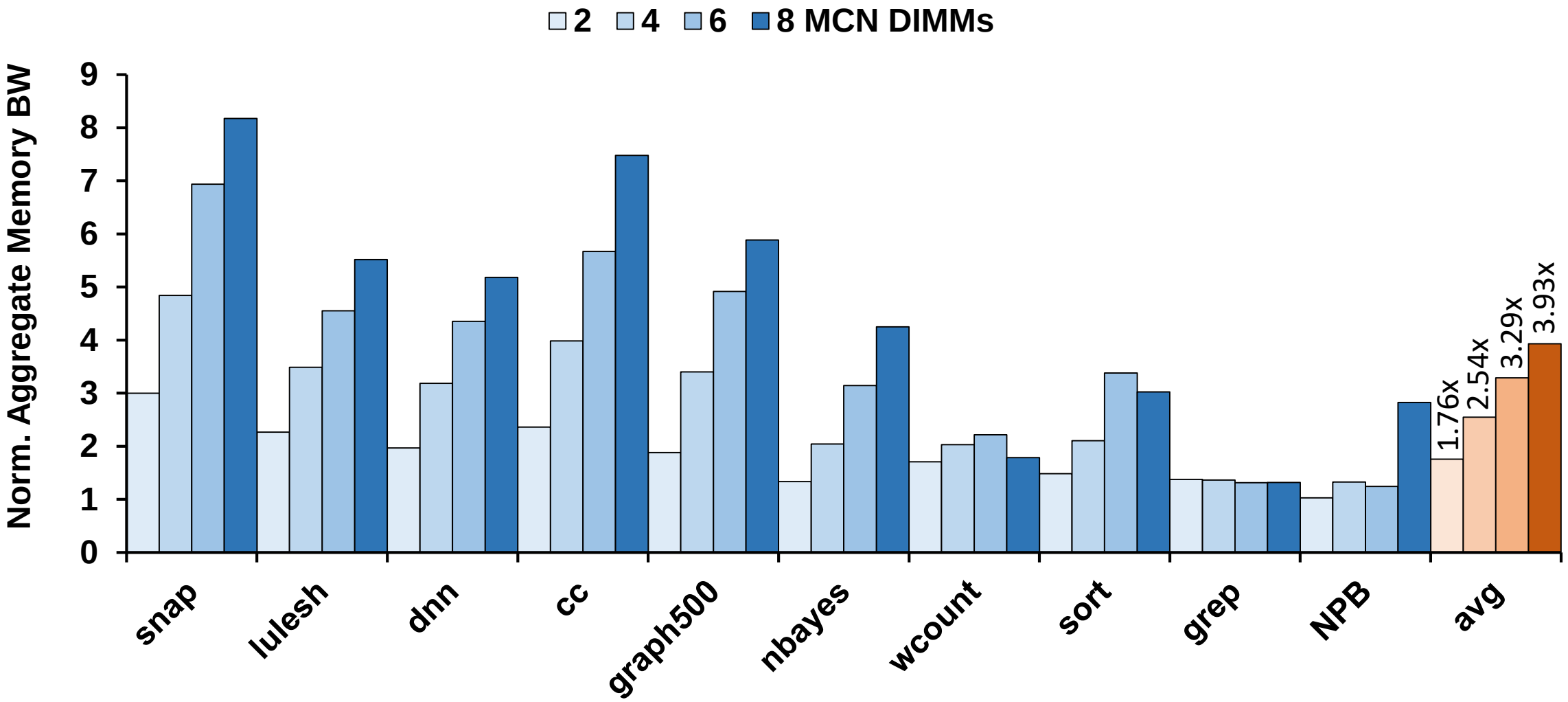
Host ↔ MCN



MCN ↔ MCN

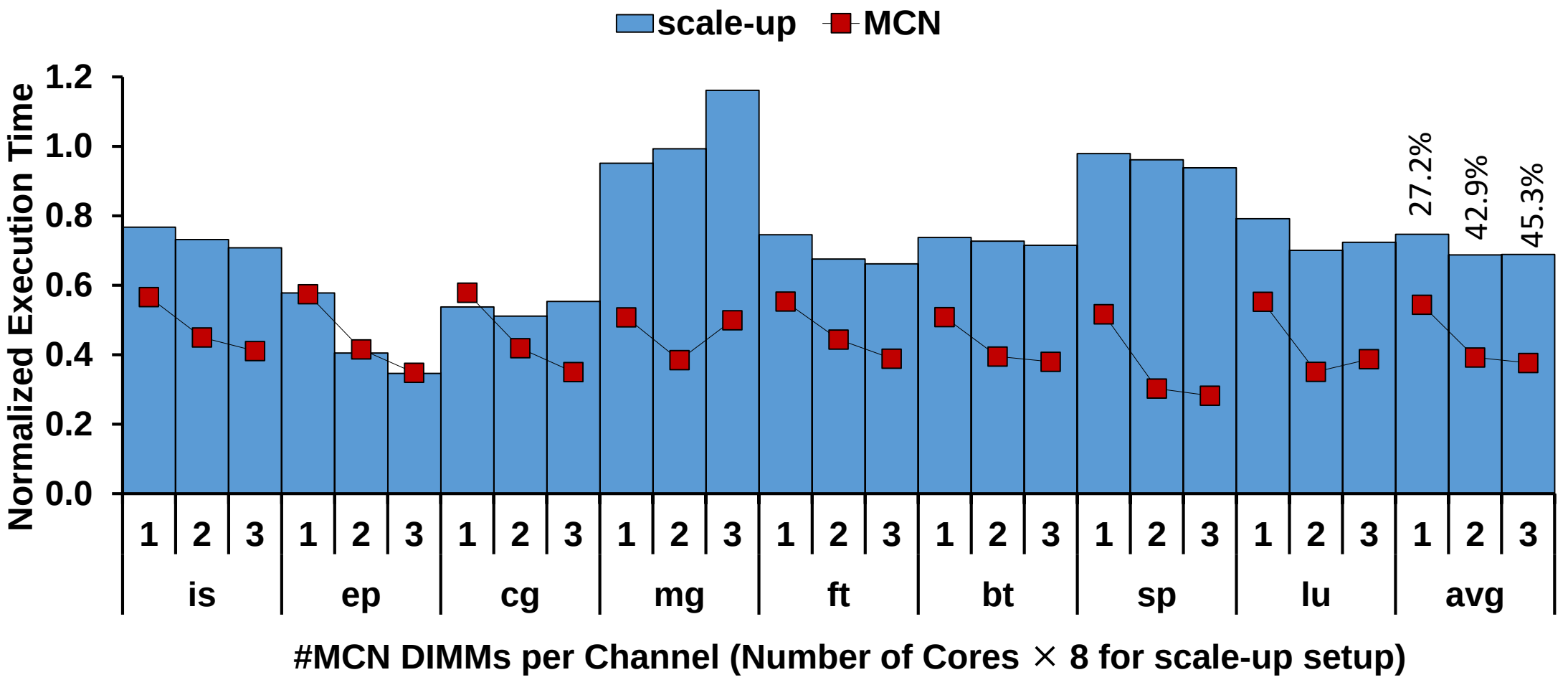


Evaluation – Aggregate Processing Bandwidth



Scale-up versus MCN: Application Performance

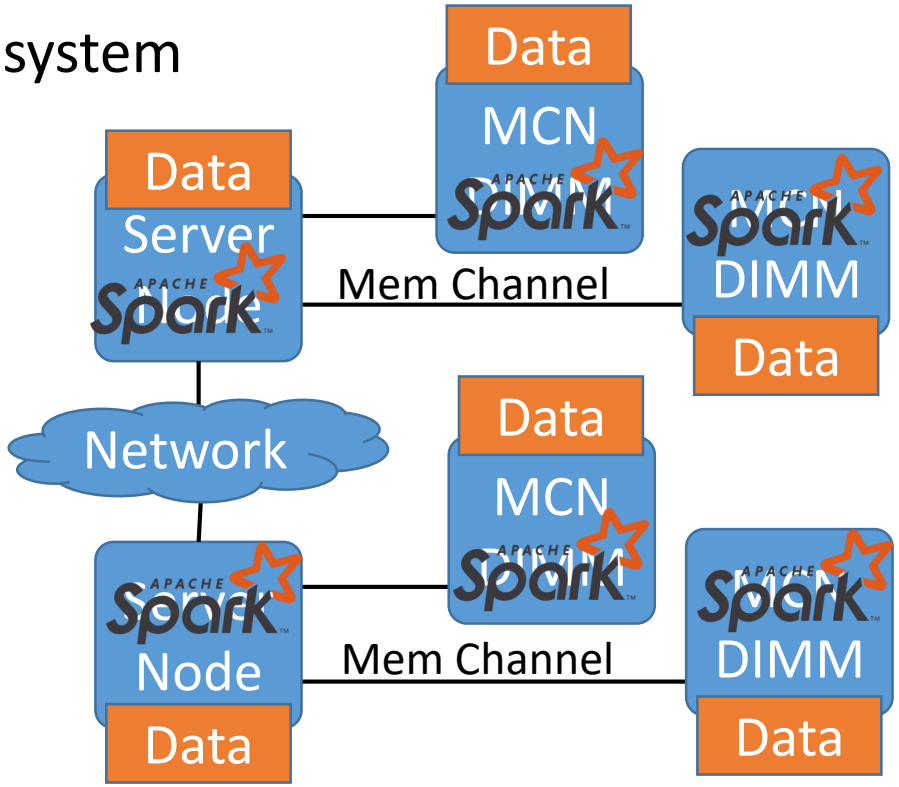
The same number of cores between scale-up server and a server w/ MCN-enabled near-memory processing modules.



Conclusion

- MCN is an innovative near-memory processing concept
  - ✓No change in host hardware, OS and user applications
  - ✓Seamless integration w/ traditional distributed computing and better scalability
  - ✓Feasibility proven w/ a commercial hardware system
- MCN can provide:
  - ✓4.6× higher network bandwidth
  - ✓5.3× lower network latency
  - ✓3.9× higher processing bandwidth
  - ✓45% higher performance

  than a conventional system



Application-Transparent Near-Memory Processing Architecture with Memory Channel Network

Mohammad Alian[1], Seung Won Min[1], Hadi Asgharimoghaddam[1], Ashutosh Dhar[1], Dong Kai Wang[1], Thomas Roewer[2], Adam McPadden[2], Oliver O'Halloran[2], Deming Chen[1], Jinjun Xiong[2], Daehoon Kim[1], Wen-mei Hwu[1], and Nam Sung Kim[1,3]

[1]University of Illinois Urbana-Champaign
[2]IBM Research and Systems
[3]Samsung Electronics

ILLINOIS
Electrical & Computer Engineering
COLLEGE OF ENGINEERING

C3SR center for cognitive computing systems research

IBM | ILLINOIS

Any Questions?