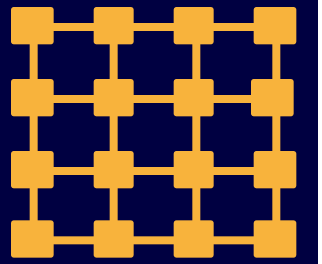
ECE 8823 A / CS 8803 - ICN
Interconnection Networks
Spring 2017



http://tusharkrishna.ece.gatech.edu/teaching/icn_s17/

Lecture 13: System Interface

Tushar Krishna

Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu

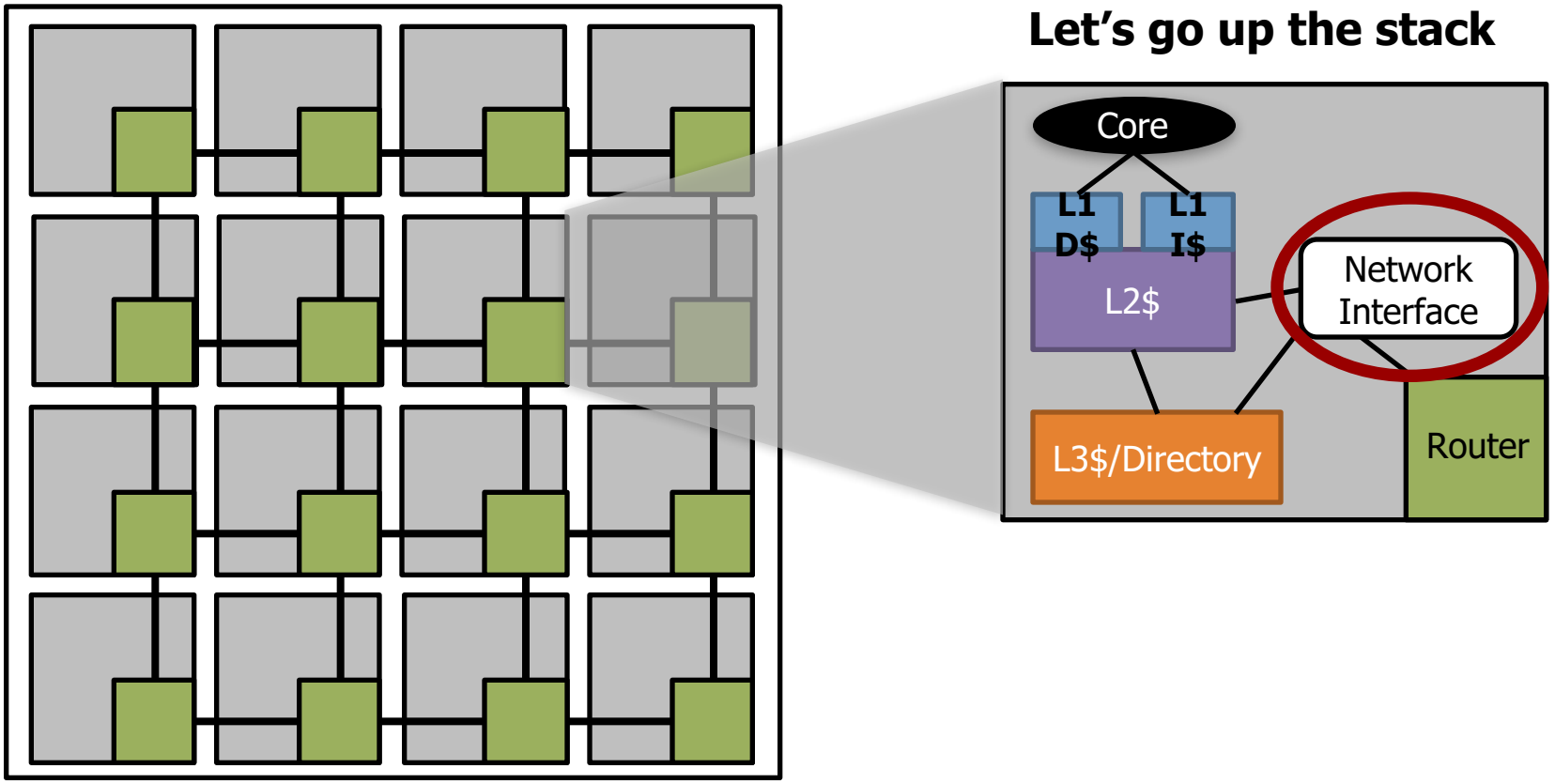
Network Architecture

- **Topology**
  - How to connect the nodes
  - ~Road Network
- **Routing**
  - Which path should a message take
  - ~Series of road segments from source to destination
- **Flow Control**
  - When does the message have to stop/proceed
  - ~Traffic signals at end of each road segment
- **Router Microarchitecture**
  - How to build the routers
  - ~Design of traffic intersection (number of lanes, algorithm for turning red/green)

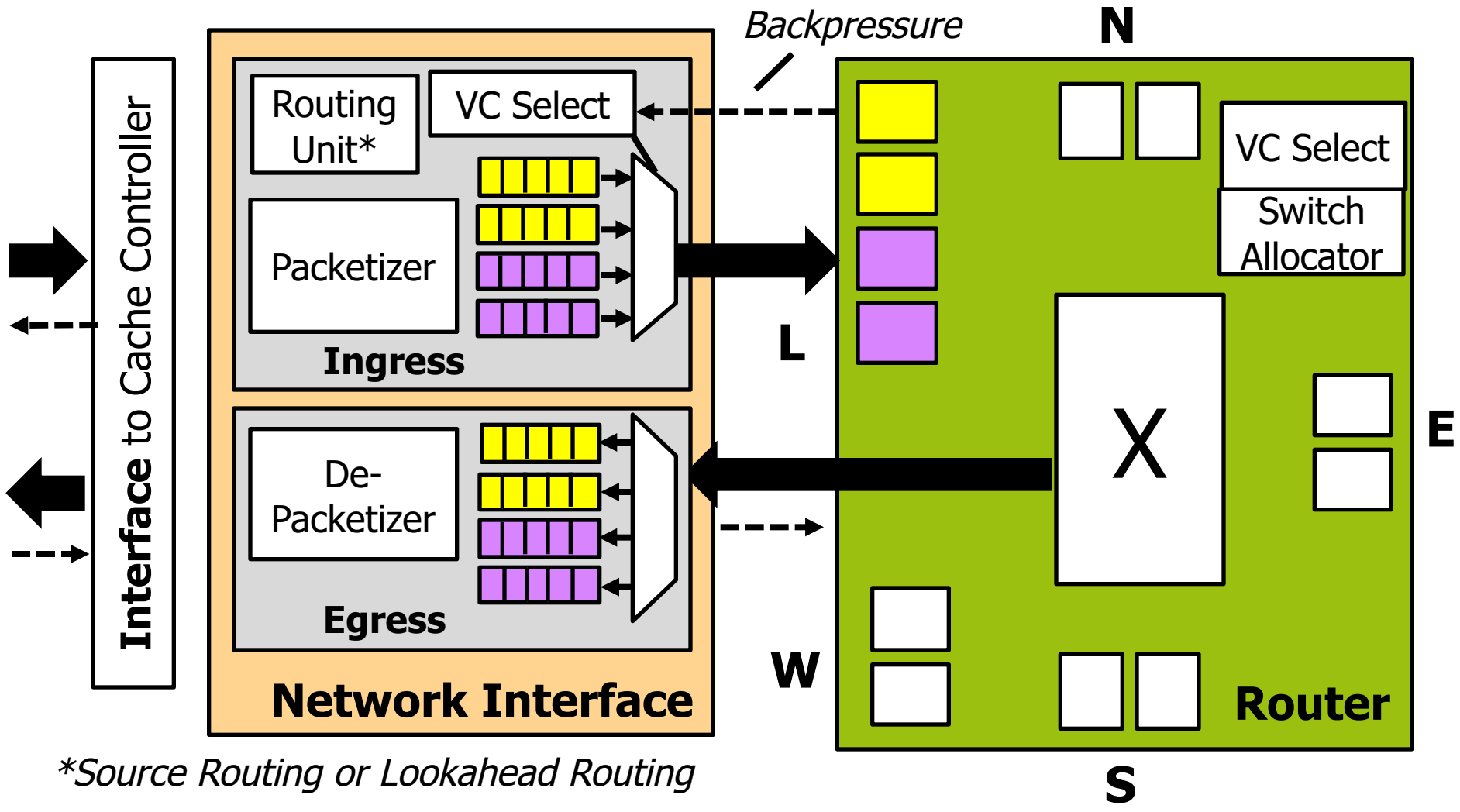
How does the NoC interface with the rest of the system?

Network Interface

So far we have focused inside the network: routers, their connections, and routing + flow-control protocols for communication between them



NIC Microarchitecture



Interface to Core/Cache Controllers

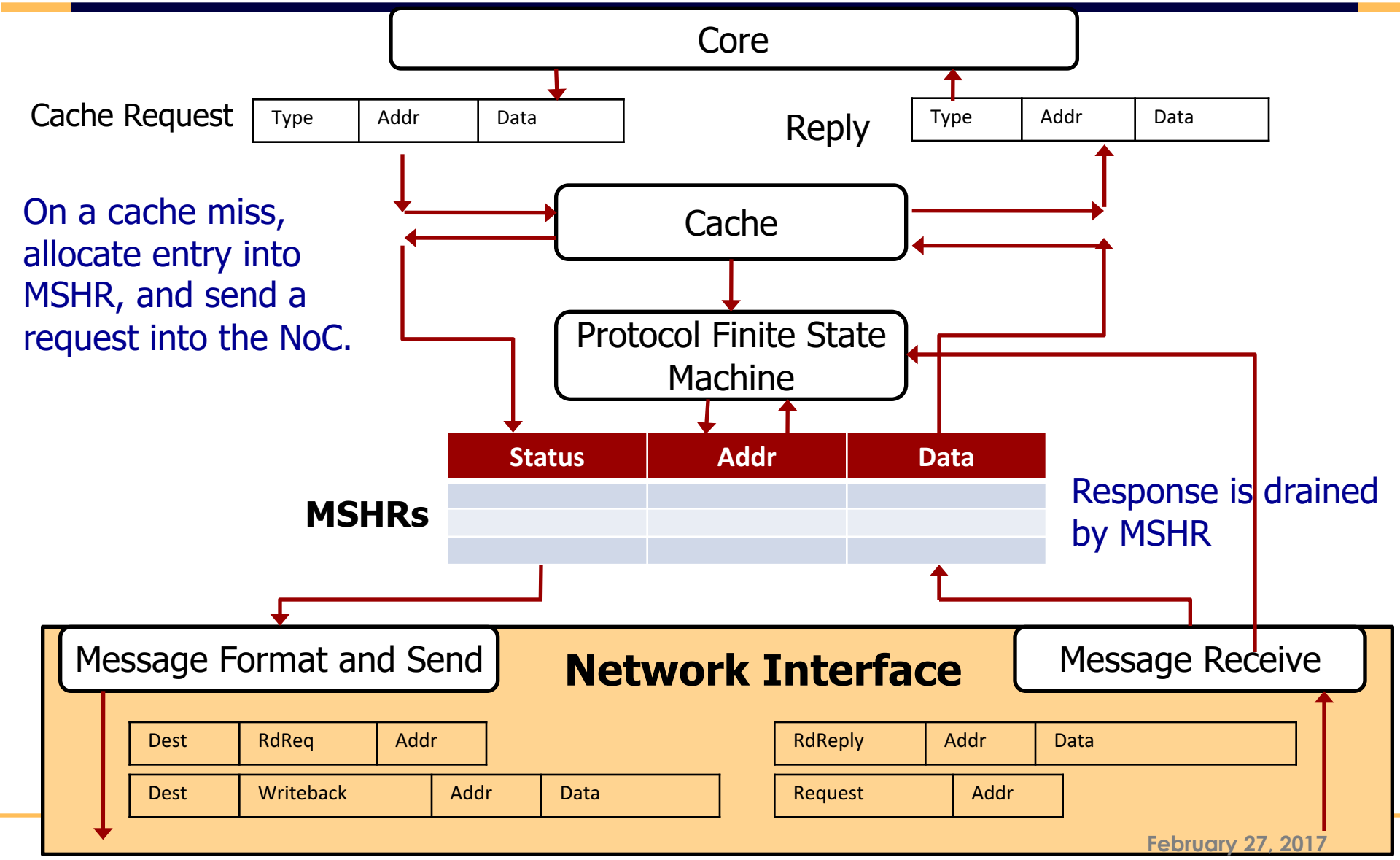
- Industry Standard Interfaces (in MPSoCs)
  - AMBA AXI (ARM)
    - AMBA 4 ACE (AXI Coherence Extensions)
    - AMBA 5 CHI (Coherent Hub Interface)
  - OCP (Sonics)
  - STBus (ST Microelectronics)
  - Wishbone (OpenCores)
- Custom Interfaces (in CMPs)
  - Intel
  - AMD
  - IBM

Communication Protocols

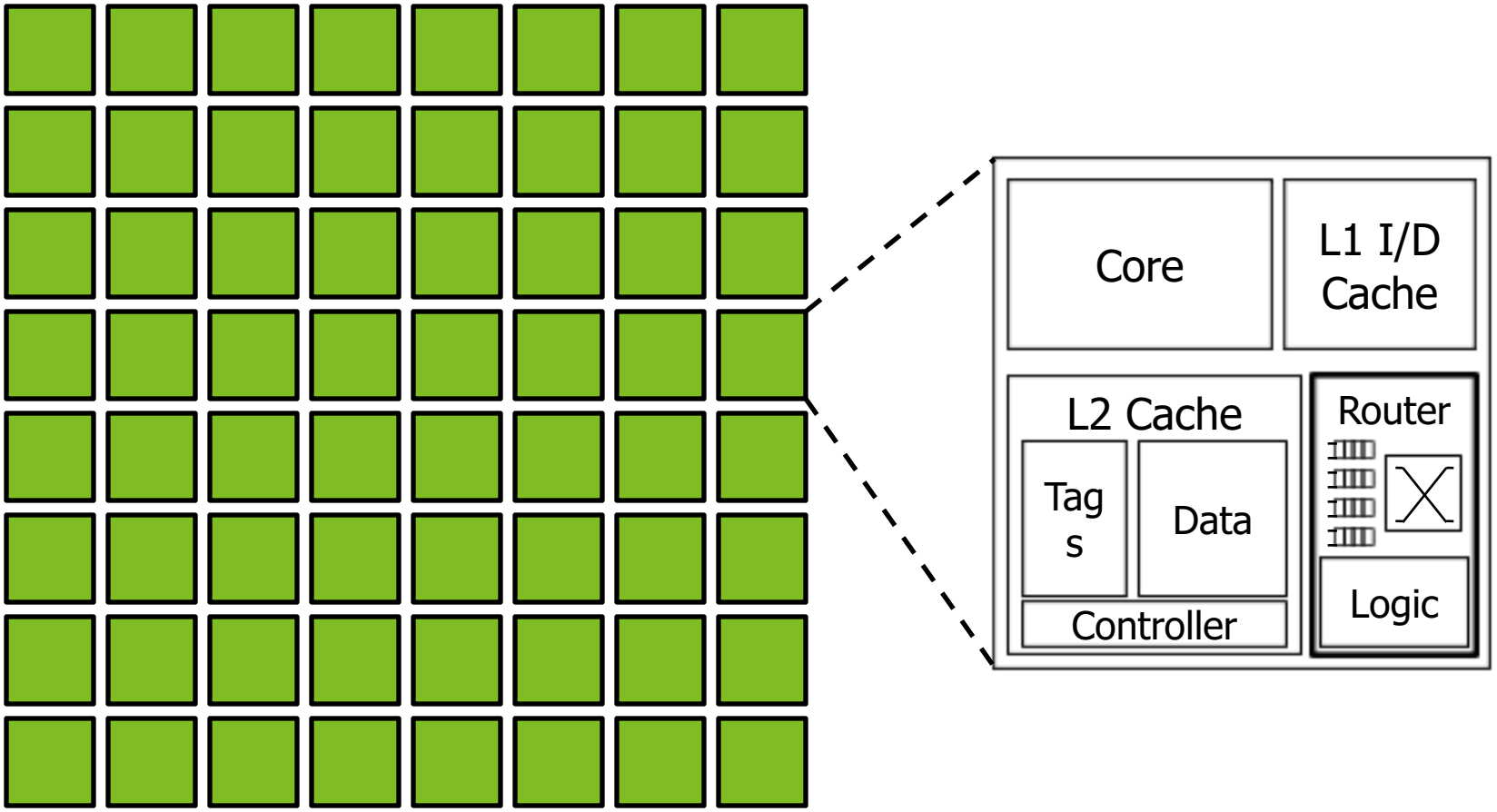
- Message Passing
  - **Explicit** movement of data between nodes and address spaces
  - Programmers manage communication

- Shared Memory
  - Communication occurs **implicitly** through loads/stores and accessing instructions
    - **Cache misses are serviced by the NoC**
    - We will focus on NoCs for shared memory systems

Cache Controller → NIC Interface: Miss Status Handling Register (MSHR)



Shared Memory Systems

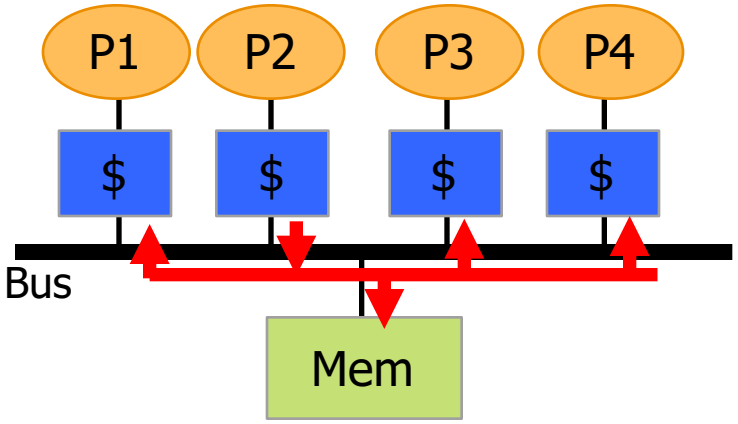


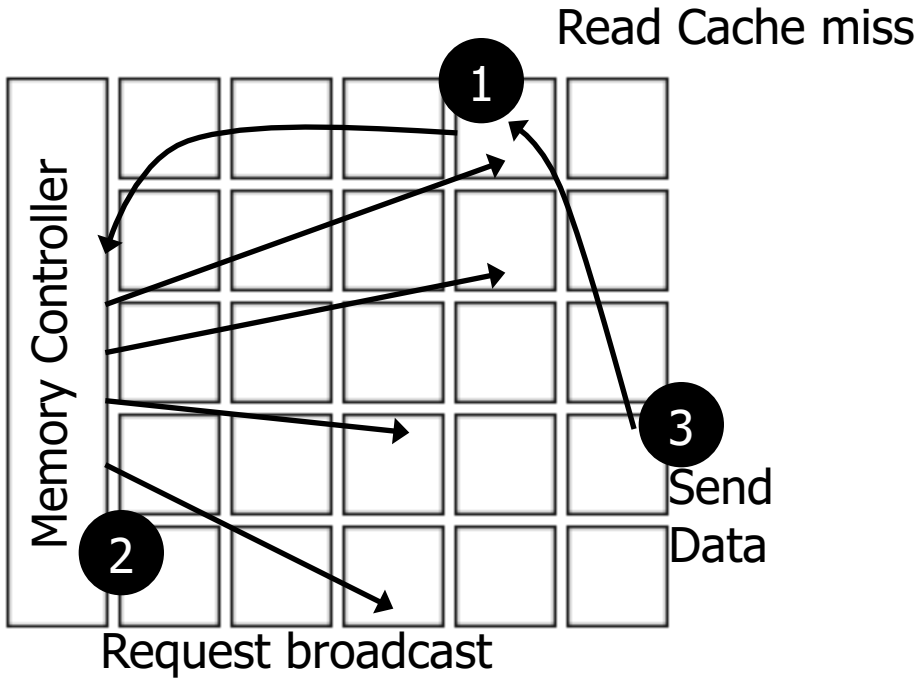
Slide Courtesy: N. Jerger, Univ of Toronto

Shared Memory Network for CMPs

- Logically…
  - all processors access same shared memory
- Practically…
  - cache hierarchies reduce access latency to improve performance
- Requires **cache coherence** protocol
  - to maintain **coherent** view in presence of **multiple** shared copies

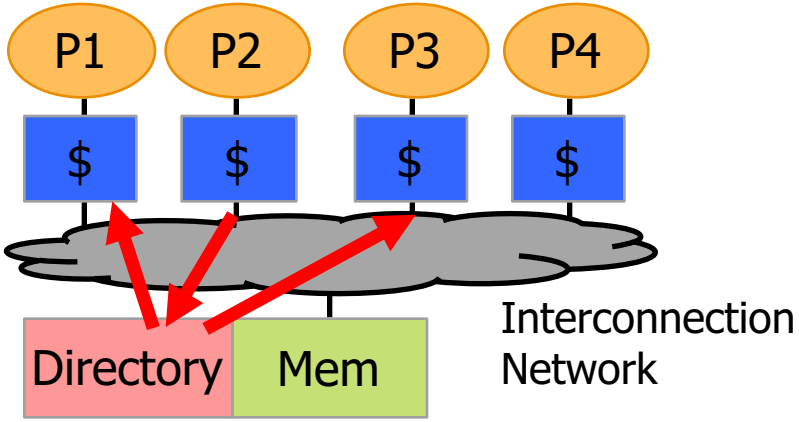
Hardware Cache Coherence

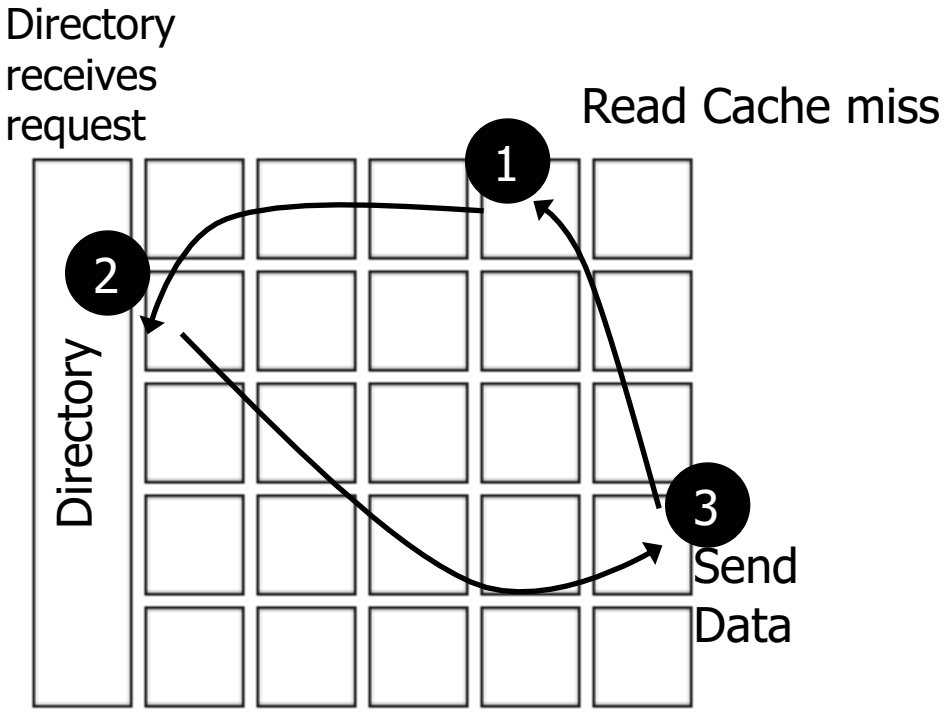




- **Snoopy Protocol**
  - Broadcast Rd/Wr request over a shared bus
  - Every cache snoops request
    - If some other cache is writing, invalidate self copy

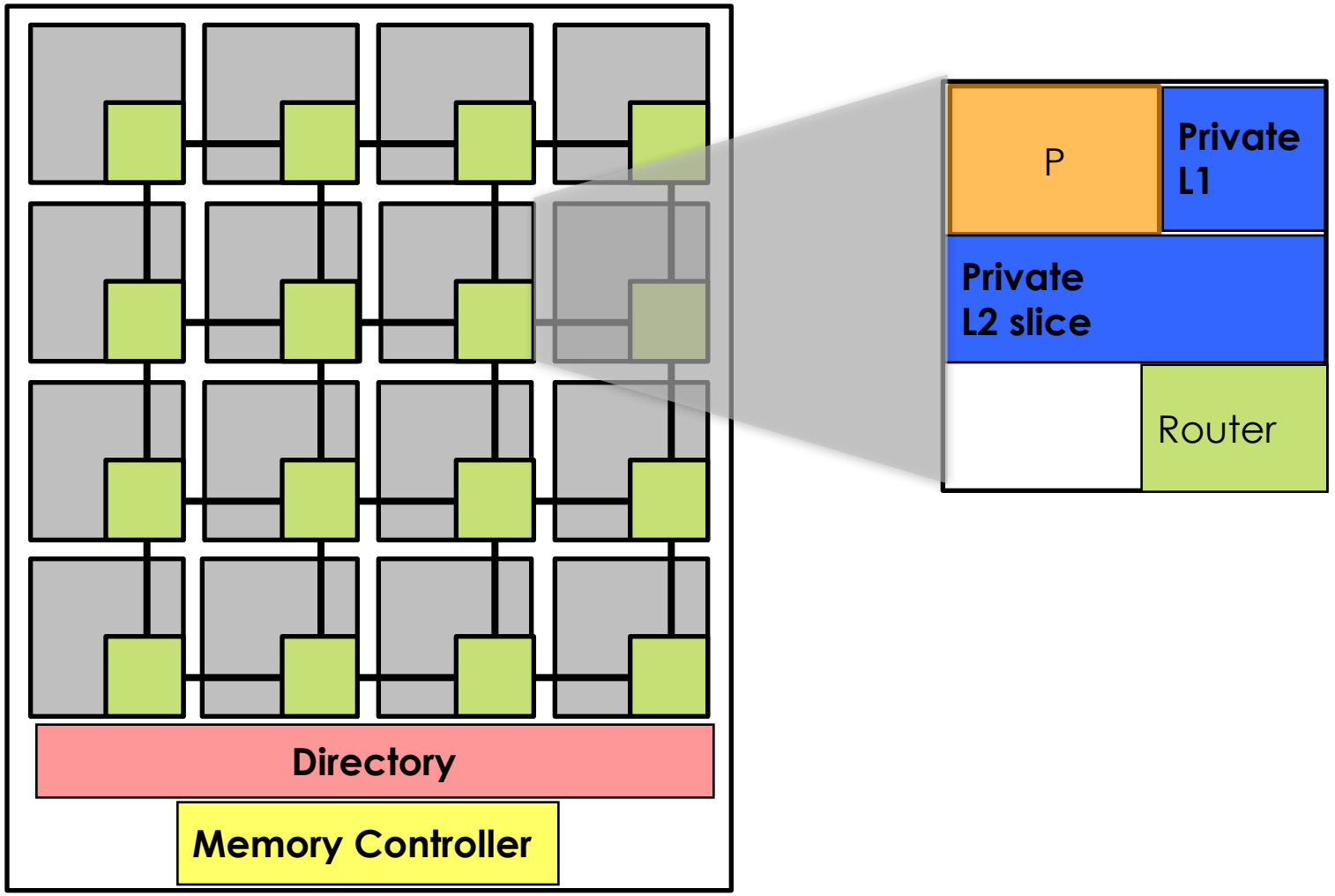
Hardware Cache Coherence





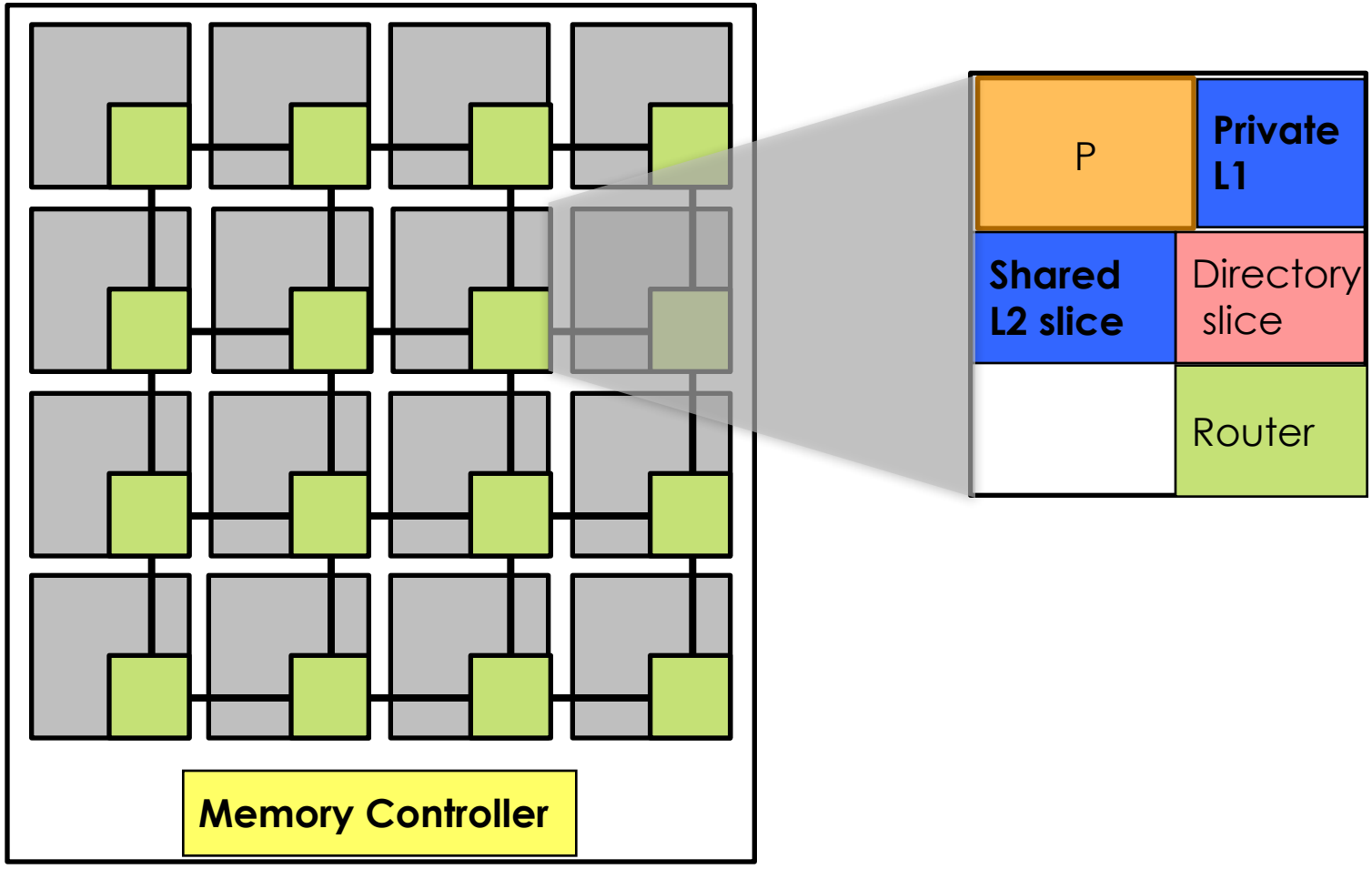
- **Directory Protocol**
  - Send a Rd/Wr request to a directory
  - Directory tracks dirty-copy and sharers and manages data response and invalidates

Cache Organization: Private L2



Cache Organization: Shared distributed L2

Non Uniform Cache Access (NUCA)

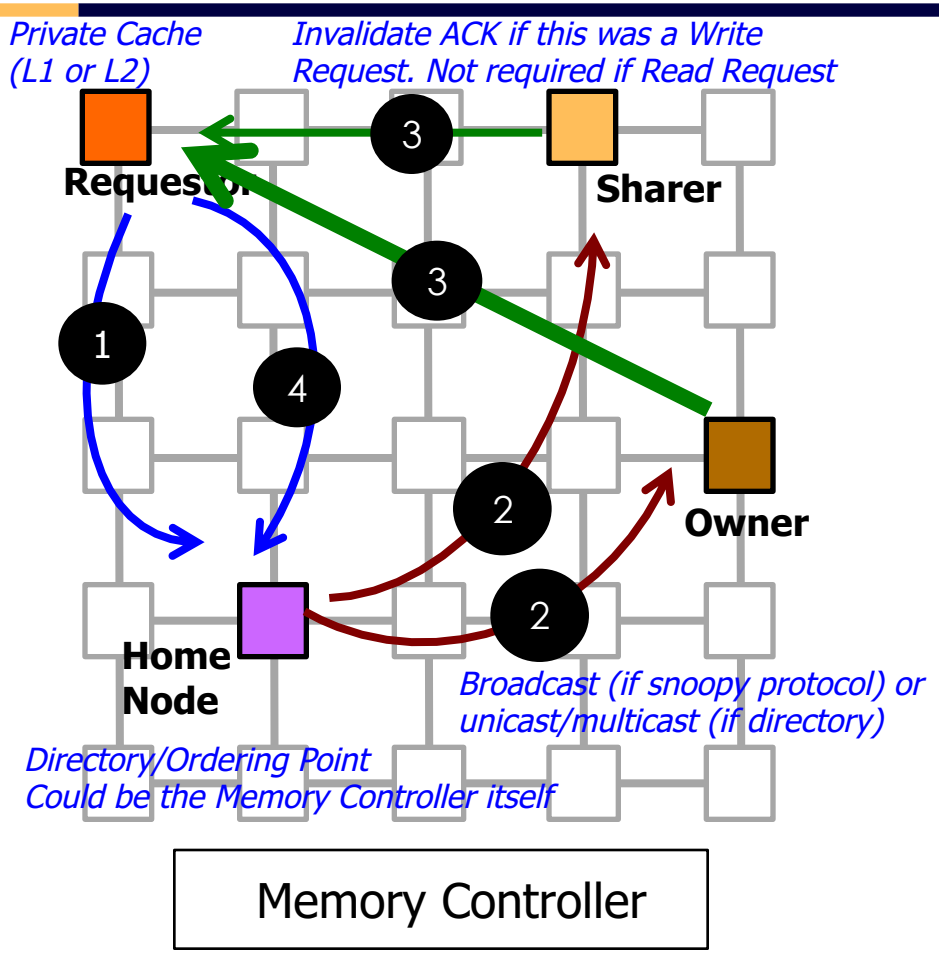


Cache Organization and Coherence Protocol Impacts Network Performance

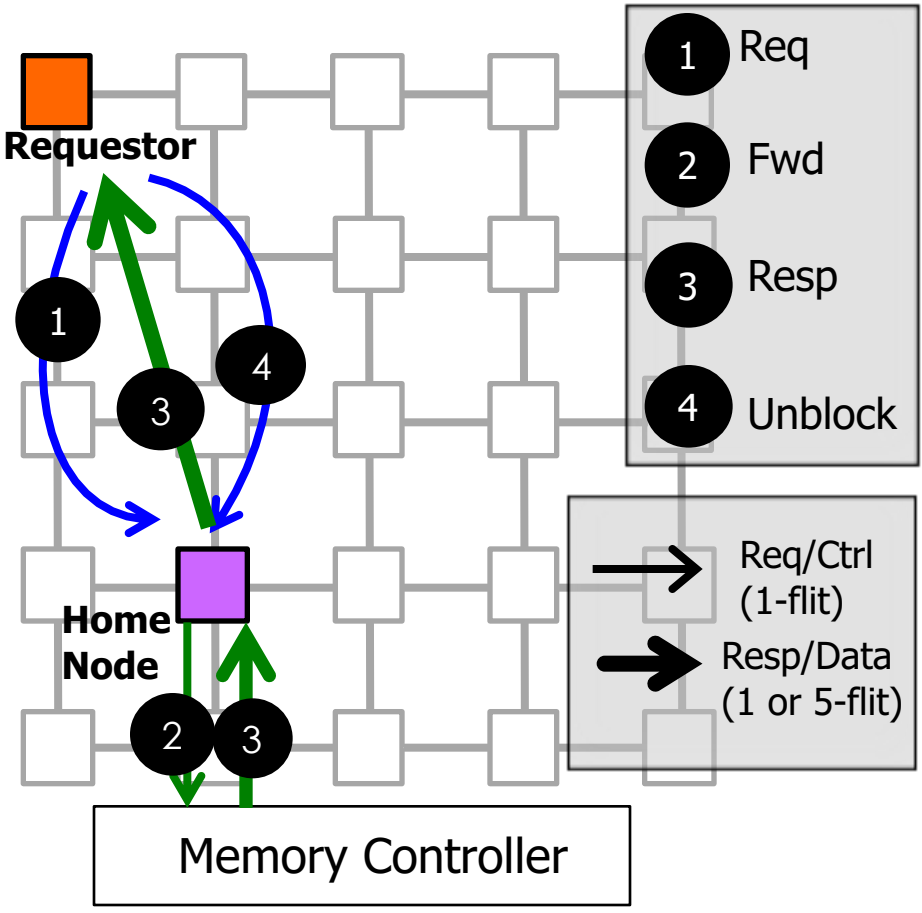
- Cache Organization shapes injection into the network
  - Private L2 caches
    - \+ **L1 Miss → L2 Miss → Traffic in the NoC [low miss penalty]**
    - – Data replication between L2s ➔ Overall lower cache capacity
  - Shared L2 caches
    - + Data can only exist in one L2 bank ➔ Higher cache capacity
    - – L1 Miss → Traffic in the NoC to go to L2 bank [increased miss penalty]

- Coherence protocol shapes NoC bandwidth requirement
  - Snoopy Protocol → More Messages
  - Directory Protocol → Fewer Messages
  - Messages Types
    - Data requests
    - Data responses
    - Coherence permissions

Cache Coherence (Private L1 + Private/Shared L2)



On-Chip Hit



On-Chip Miss

Implications of *Shared Memory* Traffic on NoC Design

- Virtual Networks
  - 3-4 Protocol Message Classes
    - request, forward, response, unblock
  - => 3-4 Virtual Networks in NoC
    - response and unblock guaranteed to drain (can share vnet)
  - There might be additional Message classes (=> Virtual Networks) for "non-cacheable requests"
    - DMA, synchronization, setup, …

Implications of *Shared Memory* Traffic on NoC Design

- Flit Size and VC depth
  - Control Packets: request, forward, response_ACK, and unblock
    - size links such that control packets fit in 1-flit
  - Data Packets: response_DATA
    - Suppose 64B cache line, 16B flits : data packets are 5-flit
      - 1-flit for control information (header etc)
      - 4-flits for cache line (64B cache line, 16B flits)

Example: MOESI_hammer (AMD Opteron) protocol in gem5

- Message Classes
  - src/mem/protocol/MOESI_hammer-cache.sm
  - src/mem/protocol/MOESI_hammer-dir.sm

```
 // Cache Controller
MessageBuffer * requestFromCache, network="To",
       virtual_network="2", vnet_type="request";

MessageBuffer * responseFromCache, network="To",
       virtual_network="4", vnet_type="response";

MessageBuffer * unblockFromCache, network="To",
       virtual_network="5", vnet_type="unblock";

MessageBuffer * forwardToCache, network="From",
       virtual_network="3", vnet_type="forward";

MessageBuffer * responseToCache, network="From",
       virtual_network="4", vnet_type="response";
```

```
// Directory Controller
MessageBuffer * forwardFromDir, network="To",
       virtual_network="3", vnet_type="forward";

MessageBuffer * responseFromDir, network="To",
       virtual_network="4", vnet_type="response";

MessageBuffer * dmaResponseFromDir, network="To",
       virtual_network="1", vnet_type="response";

MessageBuffer * unblockToDir, network="From",
       virtual_network="5", vnet_type="unblock";

MessageBuffer * responseToDir, network="From",
       virtual_network="4", vnet_type="response";

MessageBuffer * requestToDir, network="From",
       virtual_network="2", vnet_type="request";

MessageBuffer * dmaRequestToDir, network="From",
       virtual_network="0", vnet_type="request";
```

Example: MOESI_hammer (AMD Opteron) protocol in gem5

- Message Types
  - src/mem/protocol/MOESI_hammer-msg.sm

```
// CoherenceRequestType
enumeration(CoherenceRequestType, desc="...") {
  GETX,       desc="Get eXclusive";
  GETS,       desc="Get Shared";
  MERGED_GETS, desc="Get Shared";
  PUT,        desc="Put Ownership";
  WB_ACK,     desc="Writeback ack";
  WB_NACK,    desc="Writeback neg. ack";
  PUTF,       desc="PUT on a Flush";
  GETF,       desc="Issue exclusive for Flushing";
  BLOCK_ACK, desc="Dir Block ack";
  INV,        desc="Invalidate";
}
```

```
// CoherenceResponseType
enumeration(CoherenceResponseType, desc="...") {
  ACK,                desc="ACKnowledgment, responder does not have a
copy";
  ACK_SHARED,         desc="ACKnowledgment, responder has a shared
copy";
  DATA,               desc="Data, responder does not have a copy";
  DATA_SHARED,        desc="Data, responder has a shared copy";
  DATA_EXCLUSIVE,     desc="Data, responder was exclusive, gave us a
copy, and they went to invalid";
  WB_CLEAN,           desc="Clean writeback";
  WB_DIRTY,           desc="Dirty writeback";
  WB_EXCLUSIVE_CLEAN, desc="Clean writeback of exclusive data";
  WB_EXCLUSIVE_DIRTY, desc="Dirty writeback of exclusive data";
  UNBLOCK,            desc="Unblock for writeback";
  UNBLOCKS,            desc="Unblock now in S";
  UNBLOCKM,            desc="Unblock now in M/O/E";
  NULL,               desc="Null value";
}
```

Implications of Traffic on NoC Design

- **Design-time**
  - Placement of Cores/Caches/Memory Controllers
  - Homogeneous / "General Purpose CMP"
    - Typical Assumptions: each tile has 1 (or 2 cores), Private L1 Data + Instruction Cache, Private/Shared L2 slice, Directory
    - What about Memory Controllers?
      - If one or two memory controllers, usually on one end of the chip
      - What if there are more? (next)
  - Heterogeneous / "Application Specific SoC"
    - later in the course
- **Runtime (usually done by the OS)**
  - Mapping of threads/tasks to cores
  - Mapping of data across caches

Logistics

- Lab 4 [5 points]: Due coming Sunday (March 4)
  - Full-System Simulations for PARSEC benchmarks
    - 2 coherence protocols
    - 2 NoC Configurations: 1-cycle and 5-cycle routers
  - Study Impact of Network Delay on Full-system Runtime

- Proposal Presentation [7 points] (March 15)
  - **Milestone I:** Motivation Graphs [5 points]
  - Proposed Plan + Timeline [2 points]

Paper Discussion

- "Achieving Predictable Performance through Better Memory Controller Placement in Many-Core CMPs"
  - Dennis Abts, Natalie Enright Jerger, John Kim, Dan Gibson, Mikko Lipasti, *ISCA 2009*

Discussion Points

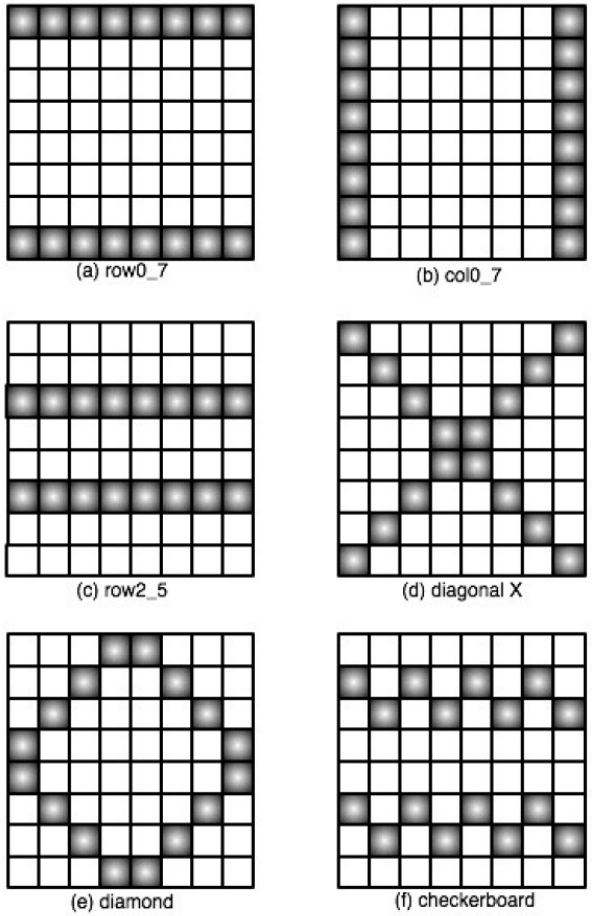


Figure 2. Different memory controller configurations. The shaded tiles represent a memory port co-located with a processor tile.

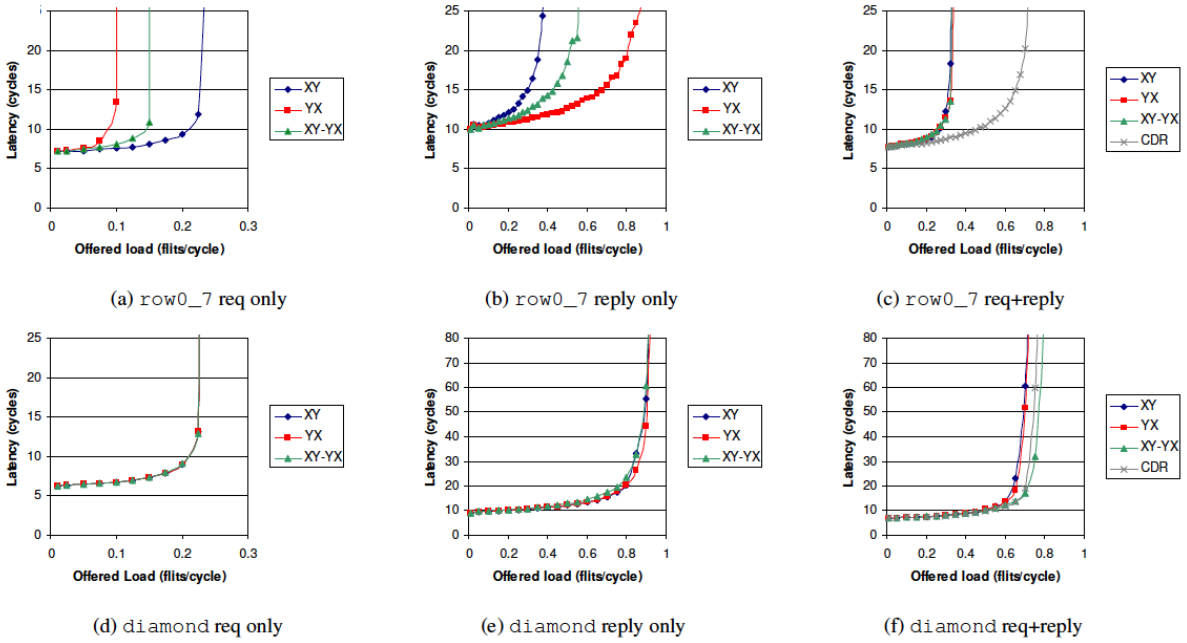
- Summary of paper
- Why do you think row0_7 and col0_7 popular?
  - What's the problem with this placement?
- Simulation Methodology?
  - Channel Load for Random Traffic
  - Why genetic algorithm?
- Routing Algorithms
  - XY, YX, XY+YX, CDR
  - Deadlock avoidance?
- Challenges with other placements?
- 2 strengths
- 2 weaknesses

Results

Memory Controller Configuration		Max. Channel Load	
		Mesh	Torus
row0_7	Figure 2a	13.50	9.25
col0_7	Figure 2b	13.50	9.25
row2_5	Figure 2c	13.49	9.22
diagonal X	Figure 2d	8.93	7.72
diamond	**Figure 2e**	**8.90**	**7.72**
checkerboard	Figure 2f	10.24	7.69

Table 1. Summary of link contention for memory configurations shown in Figure 2.

Traffic Types
Req-only, Rep-only, Req+Rep



(a) row0_7 req only (b) row0_7 reply only (c) row0_7 req+reply

(d) diamond req only (e) diamond reply only (f) diamond req+reply