Source Snooping Cache Coherence Protocols

James Goodman
Computer Science Dept.
University of Auckland

Source Snooping Cache Coherence Protocols

The gap between point-to-point network speeds and buses has grown dramatically in the last few years, leaving the dominant, bus-based snoopy cache coherence methods disadvantaged. Directory-based schemes use point-to-point networks and scale to large numbers of processors, but generally require at least three hops for most cache misses, making them slow for small- or medium-sized systems. Point-to-point networks can be used to broadcast, but the global ordering and synchronization provided by a bus are missing. Intel recently introduced a new cache coherence protocol as part of the QuickPath Interface (QPI), replacing the Front Side Bus (FSB). QPI includes the first example of a "source snooping" protocol to be introduced into a commercial product.

We will discuss source snooping protocols, showing how they can combine both the scalability of directories with the two-hop access delay of snooping caches. We will describe some of the challenges and trade-offs by means of two examples: QPI and MESIF, an ancestral protocol developed in 2001.

Disclaimer: I Don't Speak for Intel…

~~**intel Confidential**~~

Acknowledgement

This talk focuses on joint work with Herbert Hum at Intel in 2001.

QPI Reference

R.A. Maddox, G. Singh and R.J. Safranek, *Weaving High Performance Multiprocessor Fabric—Architectural insights into the Intel QuickPath Interconnect,"* Intel Press, 2009.

QPI

The Quick Path Interconnect (QPI) was recently introduced by Intel to replace the Front Side Bus

- Link Layer: QPI Link Pairs
  - "Pre-emphasis clocking"
  - Intel: "Transmitter Equalization"

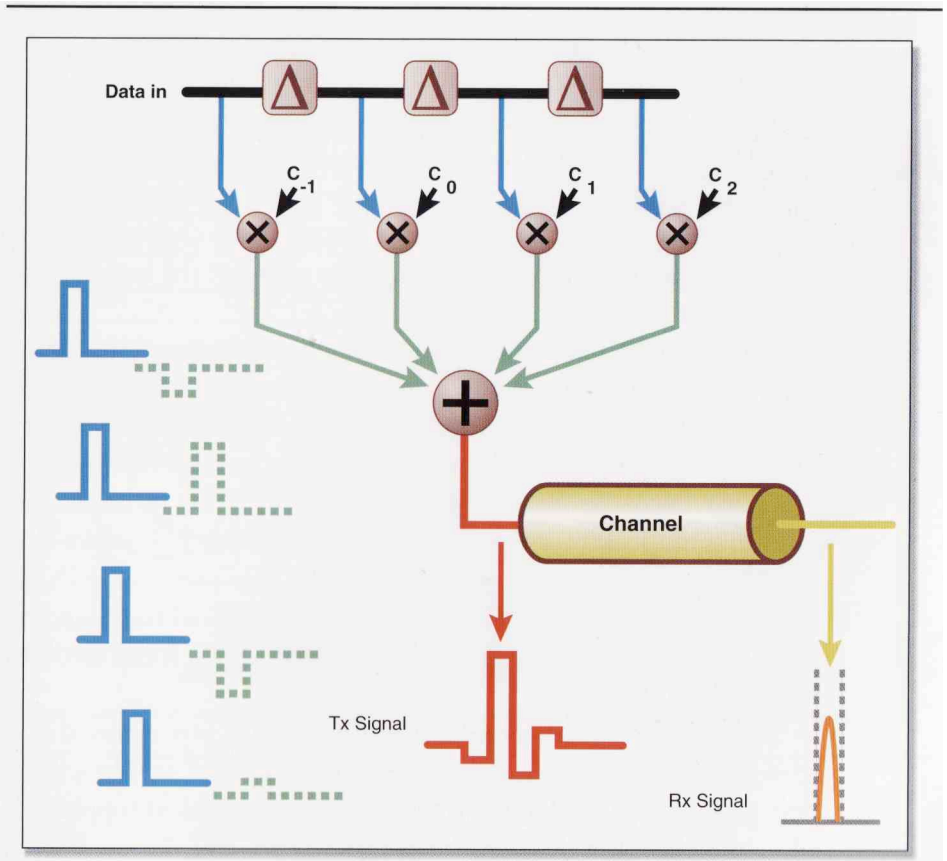
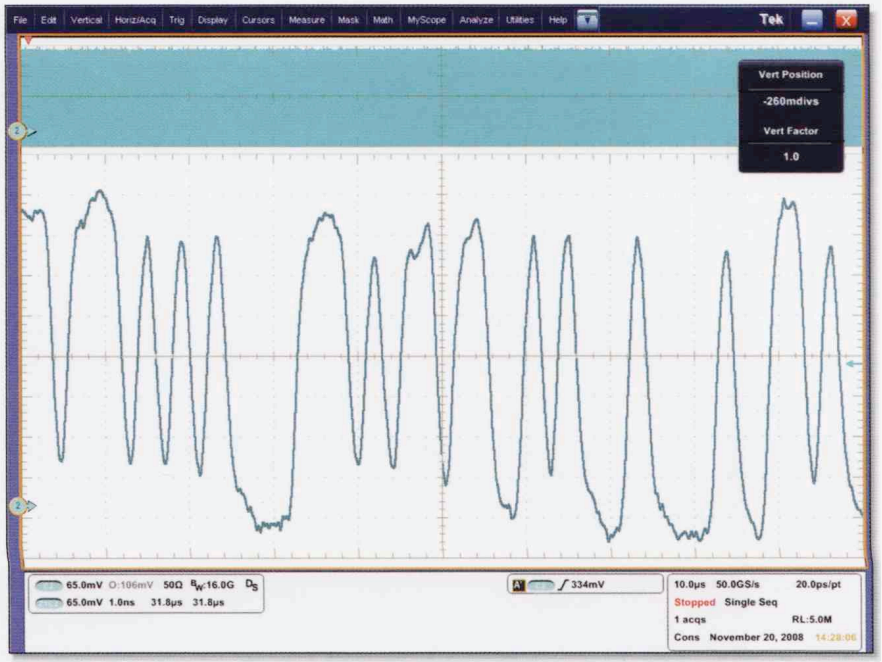


Figure 3.6 Conceptual View of Transmitter Equalization



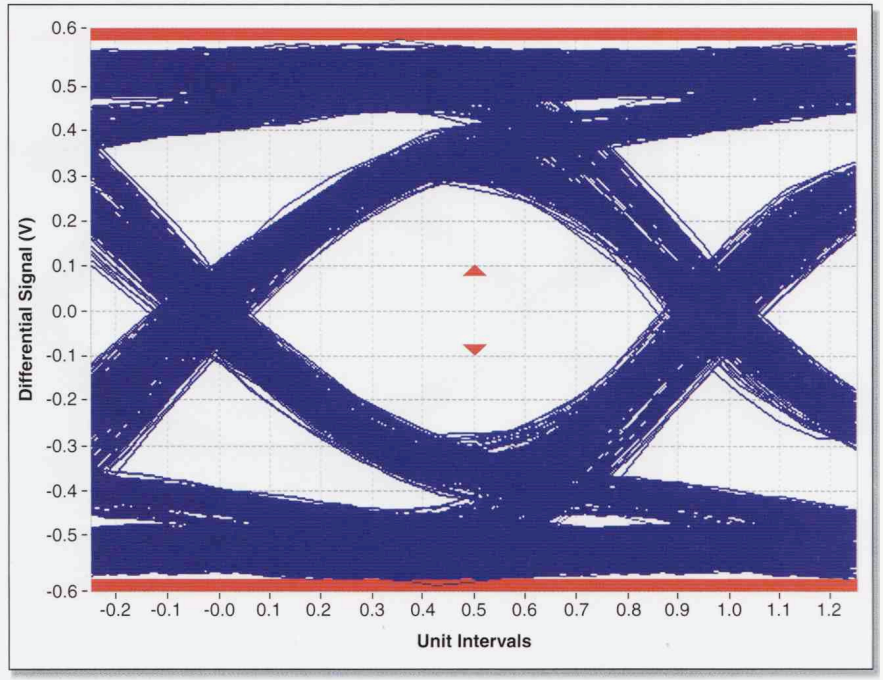
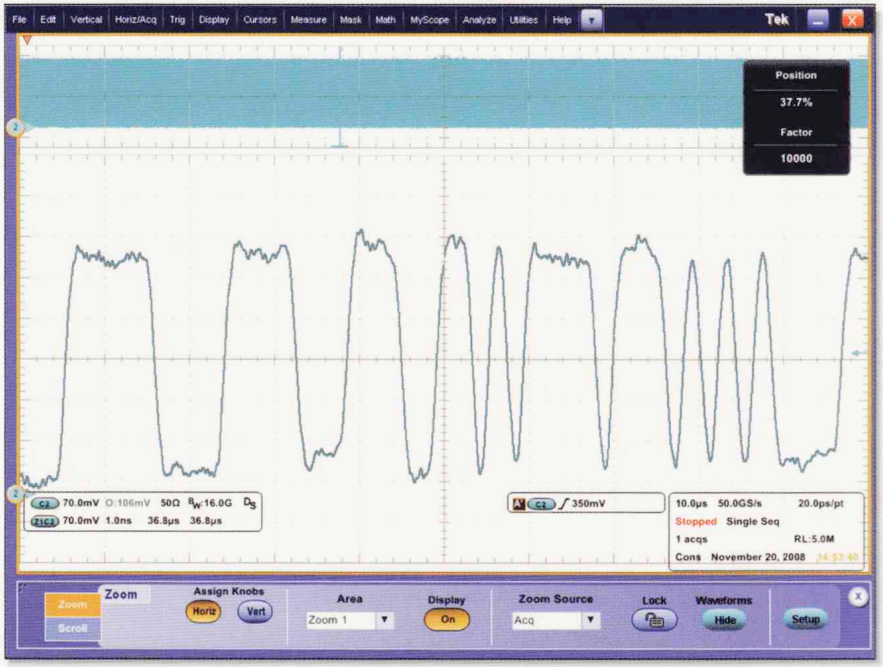


Figure 3.8 No Equalization Applied



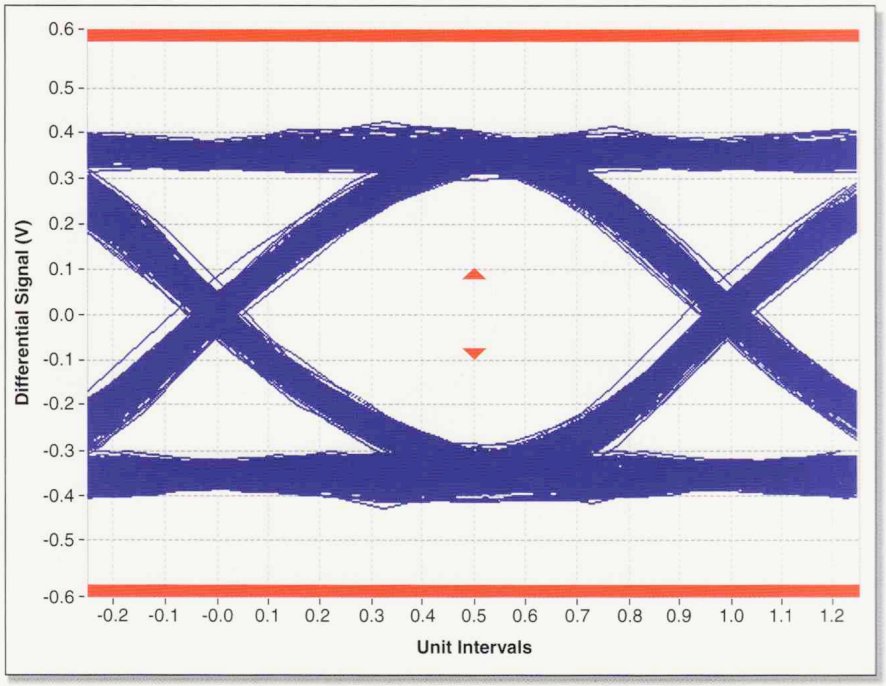


Figure 3.9 Properly Tuned Equalization

Our 2001 Goals

Develop a point-to-point cache coherency protocol that:

- Is efficient for a small number of nodes
- Has benefits of a bus protocol in terms of hops
- Does not require Backoff/Retry
- Maximizes throughput by handling concurrent requests efficiently
- Is not dependent on the switching fabric (i.e., no ordering required)
- Can scale in a hierarchical fashion
  - Does not rely on a directory scheme
  - Is directory-compatible in large systems
- Supports all memory models, including sequential consistency

QPI Protocol Layer

- Directories for large Systems (?)
- Home-Snoop Option
  - Appears similar to Archibald/Baer's "2-bit solution" (1984)
- Source-Snoop Option

QPI Source Snoop Protocol

Characteristics

- Is efficient for a small number of nodes
- Has benefits of a bus protocol in terms of hops
- Does not require Backoff/Retry
- Maximizes throughput by handling concurrent requests efficiently
- Is dependent on the switching fabric (i.e., no ordering required)
- Can scale in a hierarchical fashion
  - Does not rely on a directory scheme
  - Is directory-compatible in large systems

Our 2001 Goals

Develop a point-to-point cache coherency protocol that:

- Is efficient for a small number of nodes
- Has benefits of a bus protocol in terms of hops
- Does not require Backoff/Retry
- Maximizes throughput by handling concurrent requests efficiently
- Not dependent on the switching fabric (i.e., no ordering required)
- Can scale in a hierarchical fashion
  - Does not rely on a directory scheme
  - Is directory-compatible in large systems
- Supports all memory models, including sequential consistency

QPI Source Snoop Protocol

Characteristics

- Is efficient for a small number of nodes
- Has benefits of a bus protocol in terms of hops
- Does not require Backoff/Retry
- Maximizes throughput by handling concurrent requests efficiently
- ~~(Not)~~ Dependent on the switching fabric (i.e., no ordering required)
- Can scale in a hierarchical fashion
  - Does not rely on a directory scheme
  - Is directory-compatible in large systems
- Supports all memory models, including sequential consistency (???)

Cache Memory—A Review

- Coherence States (MESI)
- Directories
- Snooping

Coherence States

- MESI
  - Invalid
    - Not present or stale
  - Shared
    - Readable
    - May be other cached copies
  - Modified
    - Writable
    - Only valid copy
    - Main Memory is stale
  - Exclusive
    - Writable
    - Memory is consistent
- Extensions
  - MOESI also includes Owned
  - MESIF: Forward (optimization)

Directories [1976]

+ Scalable
  + "The Future" since the 1970s
- Multihop
  - Require 3+ sequential hops for most common operations
- Optimized for large systems
  - Slow even for small systems
- No large-scale commercial success

Snoopy Caches [1982]

+ Quicker access to shared data
+ Exploit broadcast capability
+ Bus defines order
– Many limits on scalability

Observed Trends

- Buses are out of gas - nasty transmission line
- Point-to-point, one-directional wires can be very fast
- Systems are hierarchical
- Small number of nodes is often a sweet spot
- Caches can provide data faster than main memory
  - Sharing is increasingly important in cache misses

Point-to-Point Links

- Good properties
  - Fast
  - Robust
  - Can be indirect
- Observation: point-to-point links can be used to "broadcast"
  - Need complete interconnect?
- May not scale, but works for small numbers

Idea: Replace a bus with point-to-point links and snoop

Good idea, but buses support (and snooping protocols exploit)

- Intervention
- Conflicts
- Ordering

State of the Art, 2001

Focus on ordering requests entering network:

- P.F. Reynolds, Jr., C. Williams, and R.R. Wagner, Jr., "Isotach networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 4, pp. 337-348, April 1997.
- J. Regehr, "An isotach implementation for Myrinet," Technical Report CS-97-12, Dept. of Computer Science, University of Virginia, May 1997.
- A.E. Condon, et al., "Using Lamport clocks to reason about relaxed memory models," in *Proceedings of the 5th International Symposium on High Performance Computer Architecture*, Orlando, Florida, January 1999.
- E.E. Bilir, R.M. Dickson, Y. Hu, M. Plakal, D.J. Sorin, M.D. Hill, and D.A. Wood, "Multicast snooping: a new coherence method using a multicast address network," in *Proceedings of the 26th International Symposium on Computer Architecture*, Atlanta, Georgia, May 1999.
- M.K.M. Martin, D.J. Sorin, A. Ailamaki, A.R. Alameldeen, R.M. Dickson, C.J. Mauer, K.E. Moore, M. Plakal, M.D. Hill, D.A. Wood, "Timestamp Snooping: An Approach for Extending SMPs," in *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, Cambridge, Massachusetts, November 13-15, 2000.

State of the Art, 2009

Focus on ordering requests entering network:

- N. Agarwal, L-S. Peh and N.K. Jha, "In-Network Snoop Ordering (INSO): Snoopy Coherence on Unordered Interconnects," *15th International Conference on High-Performance Computer Architecture (HPCA-15)*, February 2009, Raleigh, North Carolina, USA 2009

"There are many different interpretations of snoopy protocols. We use snoopy to imply a broadcast protocol in which requests are sent directly to other nodes in the system, without having to go to an ordering point. Other nodes in the system "snoop" to determine whether the request is meant for them and act accordingly."

"We propose In-Network Snoop Ordering (INSO), in which coherence requests from a snoop-based protocol are inserted into the interconnect fabric and the network orders the requests in a distributed manner, creating a global ordering among requests."

Insight

The network does not need to provide ordering

- The network must inform a node when it is safe to expose modifications
- Global ordering can be achieved if each processor orders its own memory operations, assuring changes in values are observable only after the instruction has committed.
- Nodes can collectively detect conflicts and prevent instructions from committing until they are resolved.

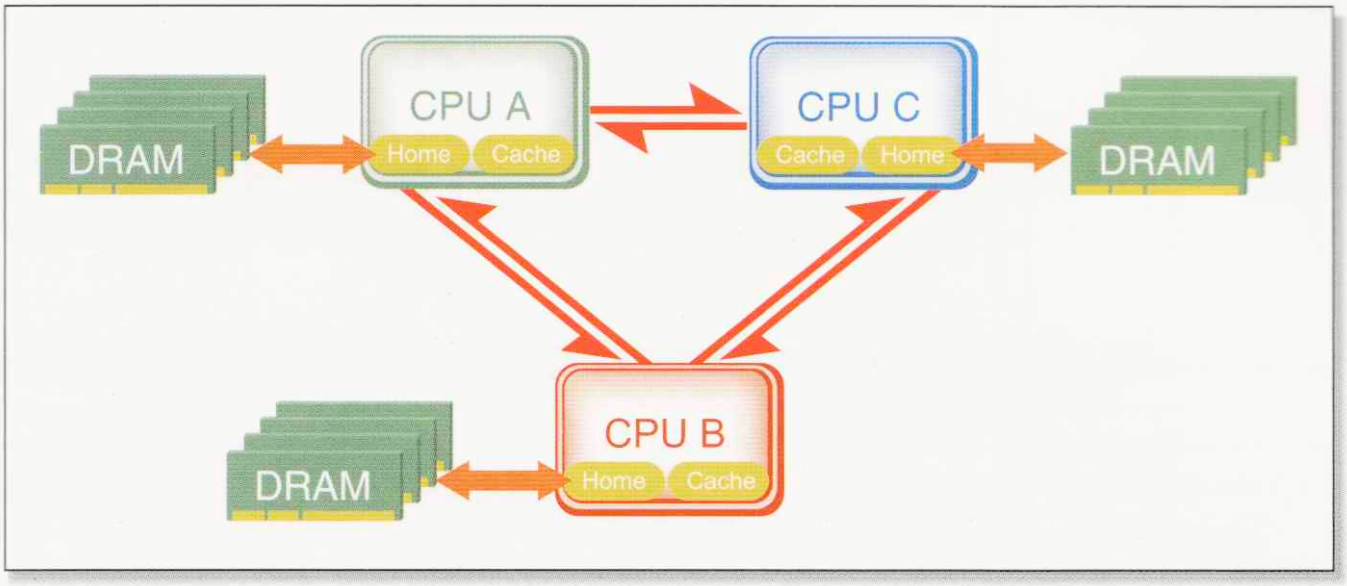


Figure 2.5 Three-Processor System Based on Intel® QuickPath Interconnect Technology

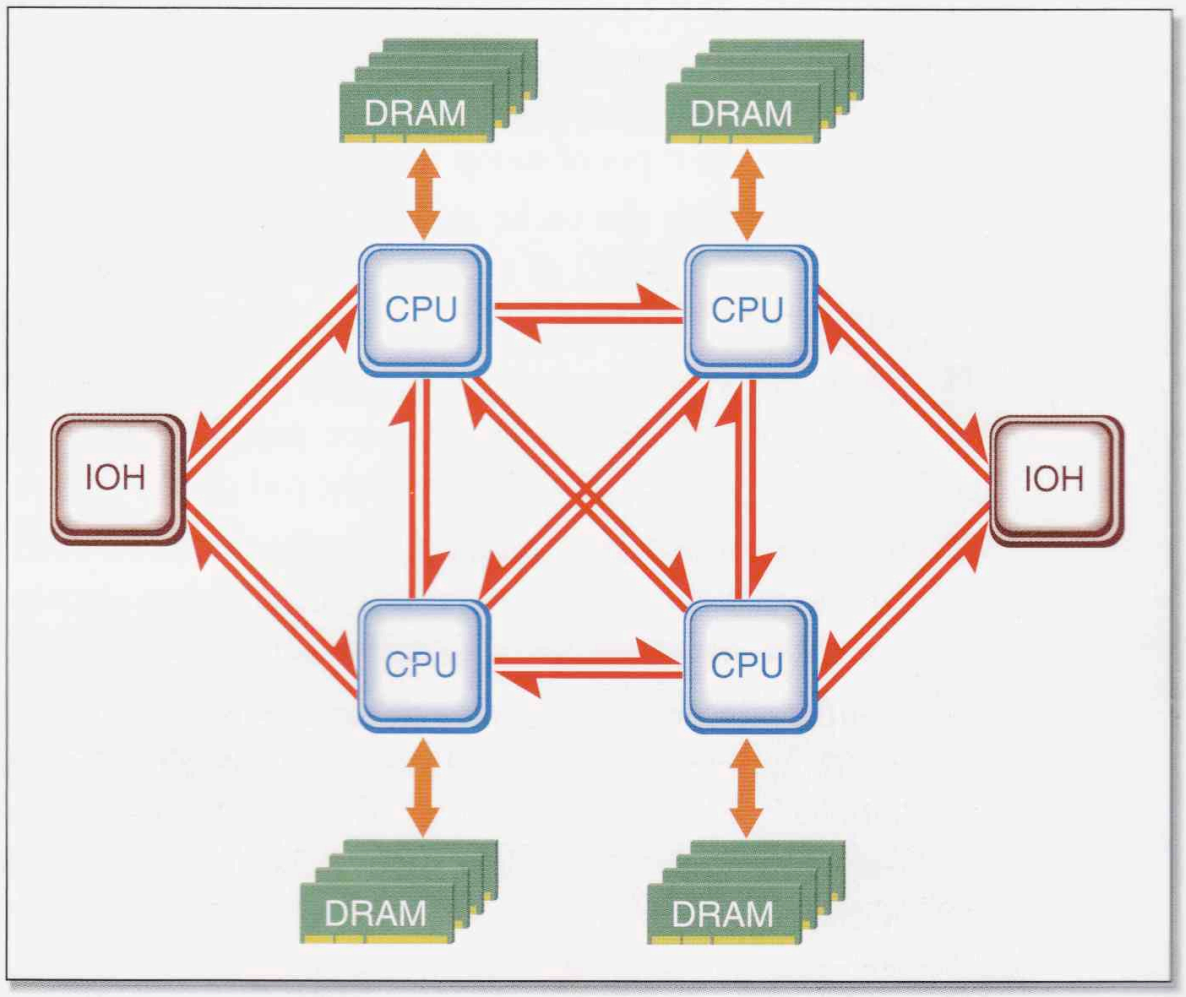


Figure 2.2 Fully Connected Four-Processor System Based on Intel® QuickPath Interconnect Technology

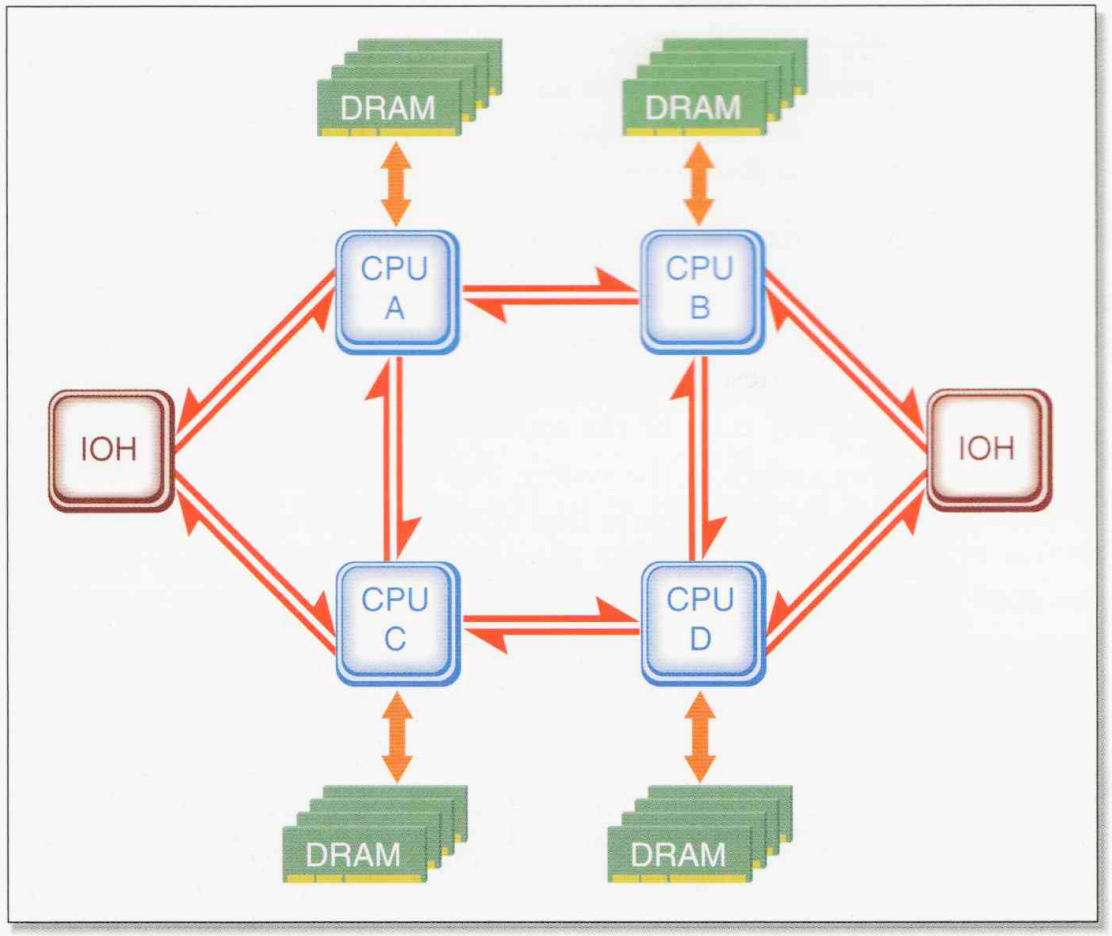


Figure 2.3 Partially Connected Four-Processor System Based on Intel® QuickPath Interconnect Technology

Our System Model

- A *node* may include
  - A processor, which generates memory requests
  - A cache, which may contain redundant copies (or the only copy) of parts of the address space
  - Main memory for some part of the address space
- A *point-to-point communication network*
  - Provides pair-wise communication paths that connect all nodes, possibly indirectly
  - Reliably delivers messages
  - Is unordered
- Each address resides at a "Home" node, and may be cached in some or all nodes
- A node may consist of multiple nodes connected by a communication network and behaving as if it were a single node

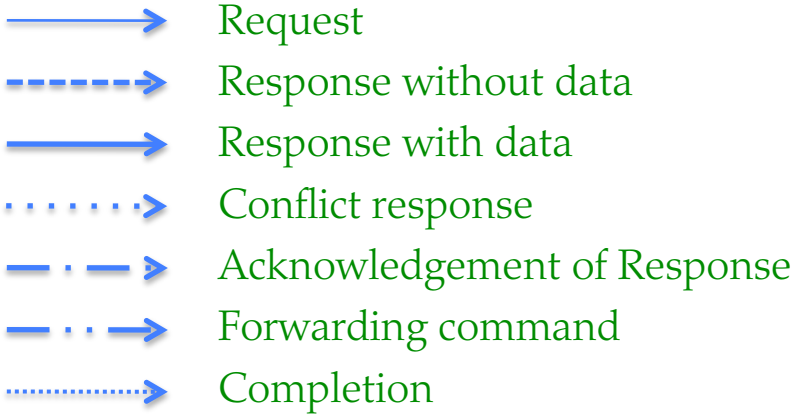
MESIF (măs′ĭv) Protocol

Message Sequence Diagrams

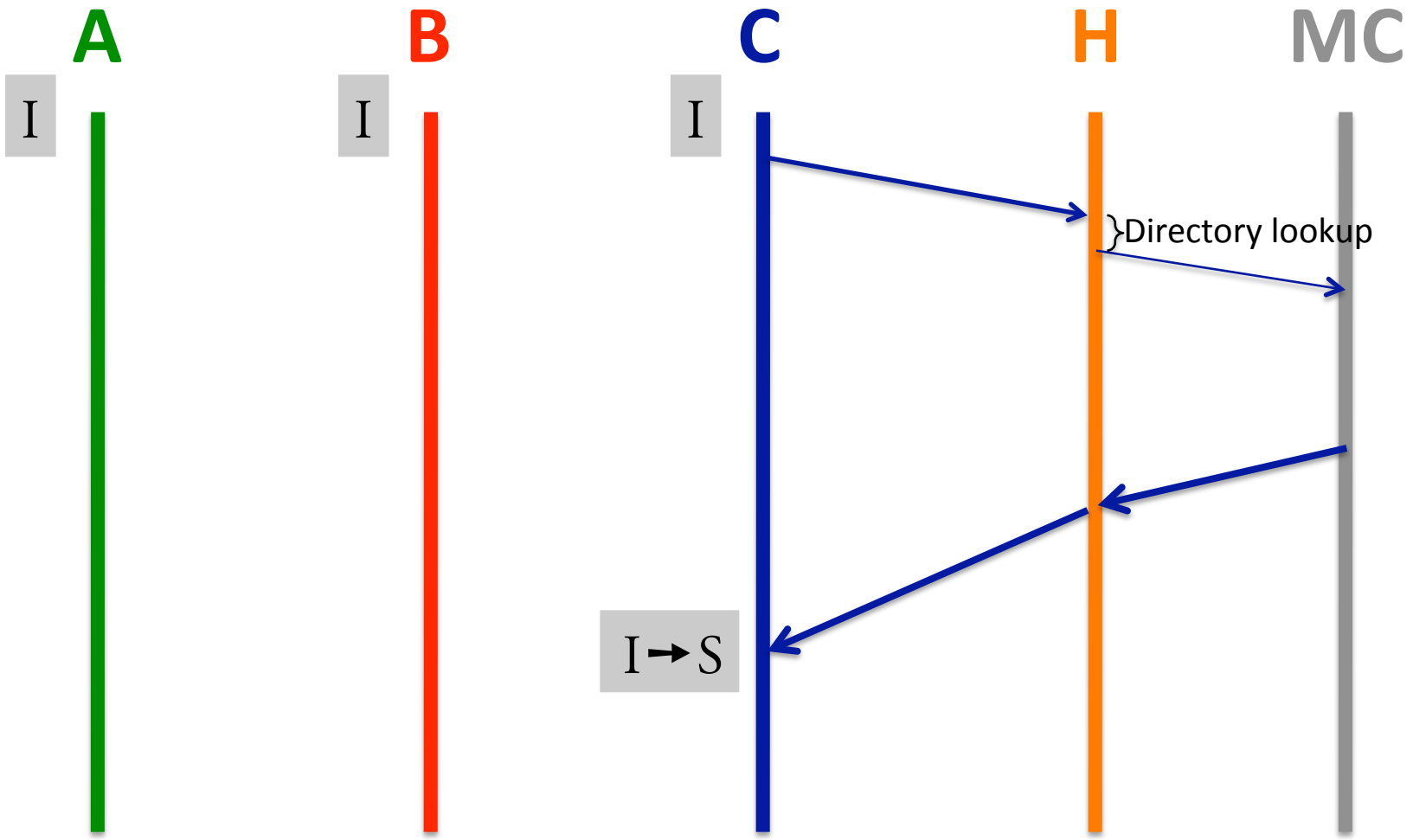
ABC Caching Agents

H Home node (possibly including cache)

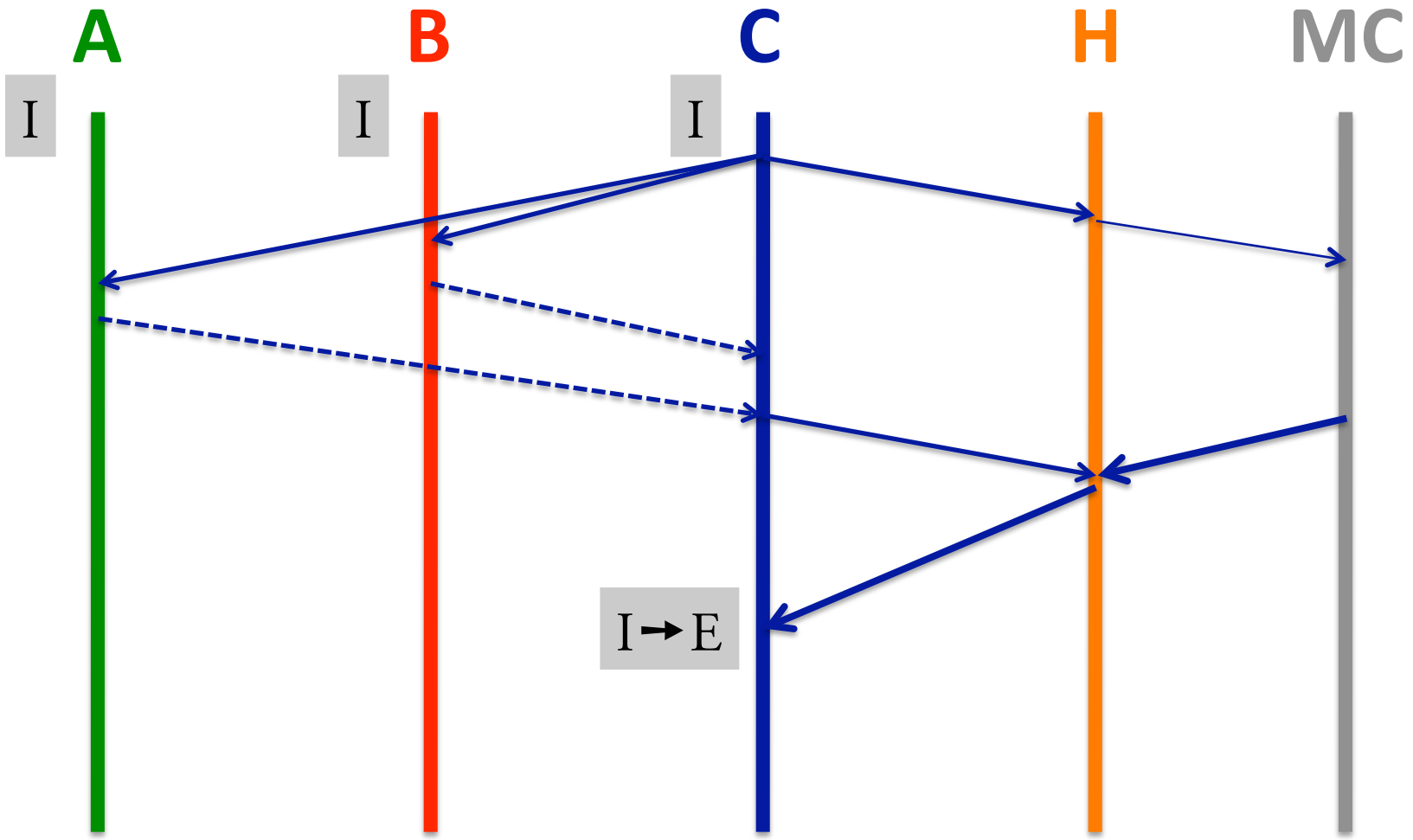
MC Memory Controller



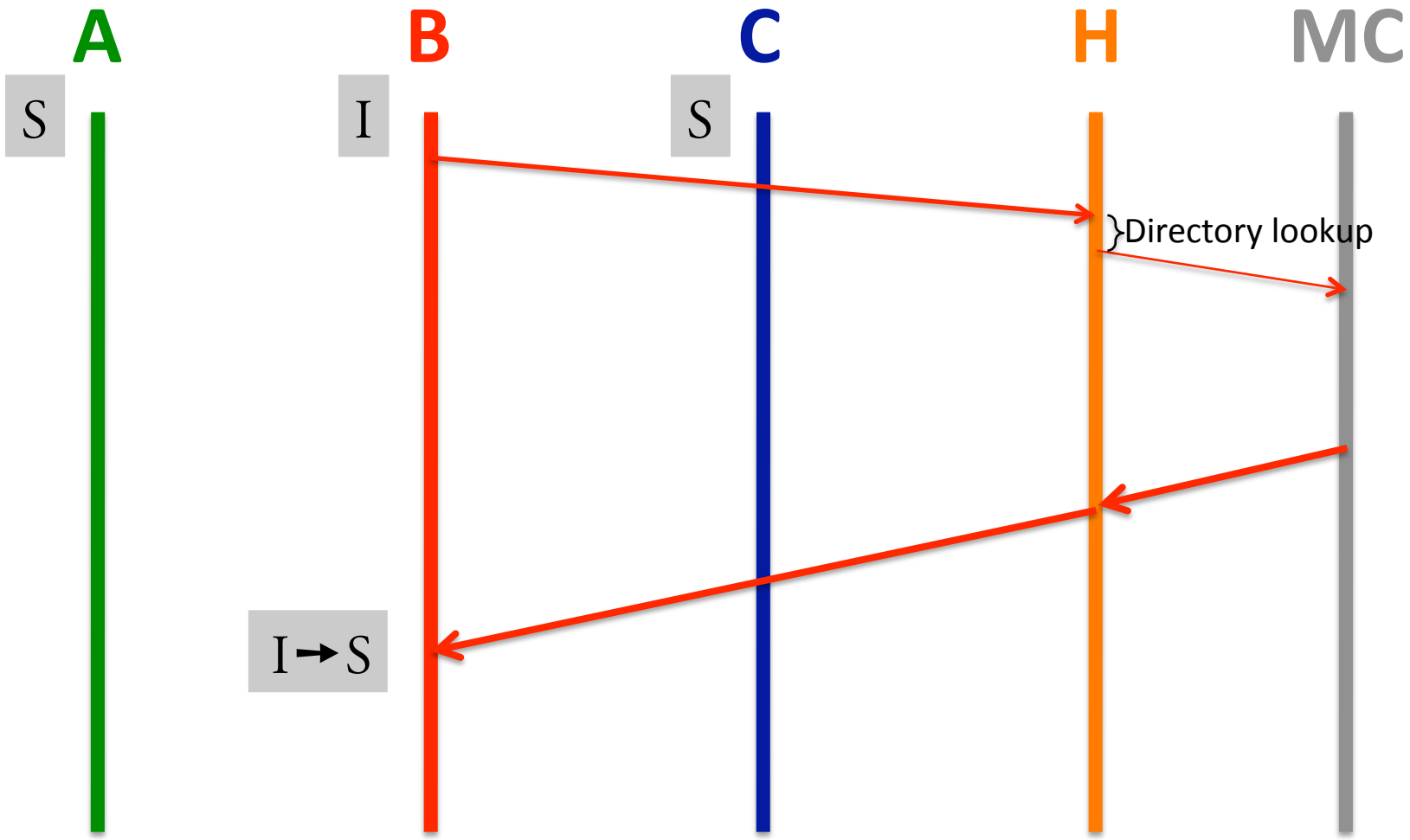
Read Uncached Line (Directory)



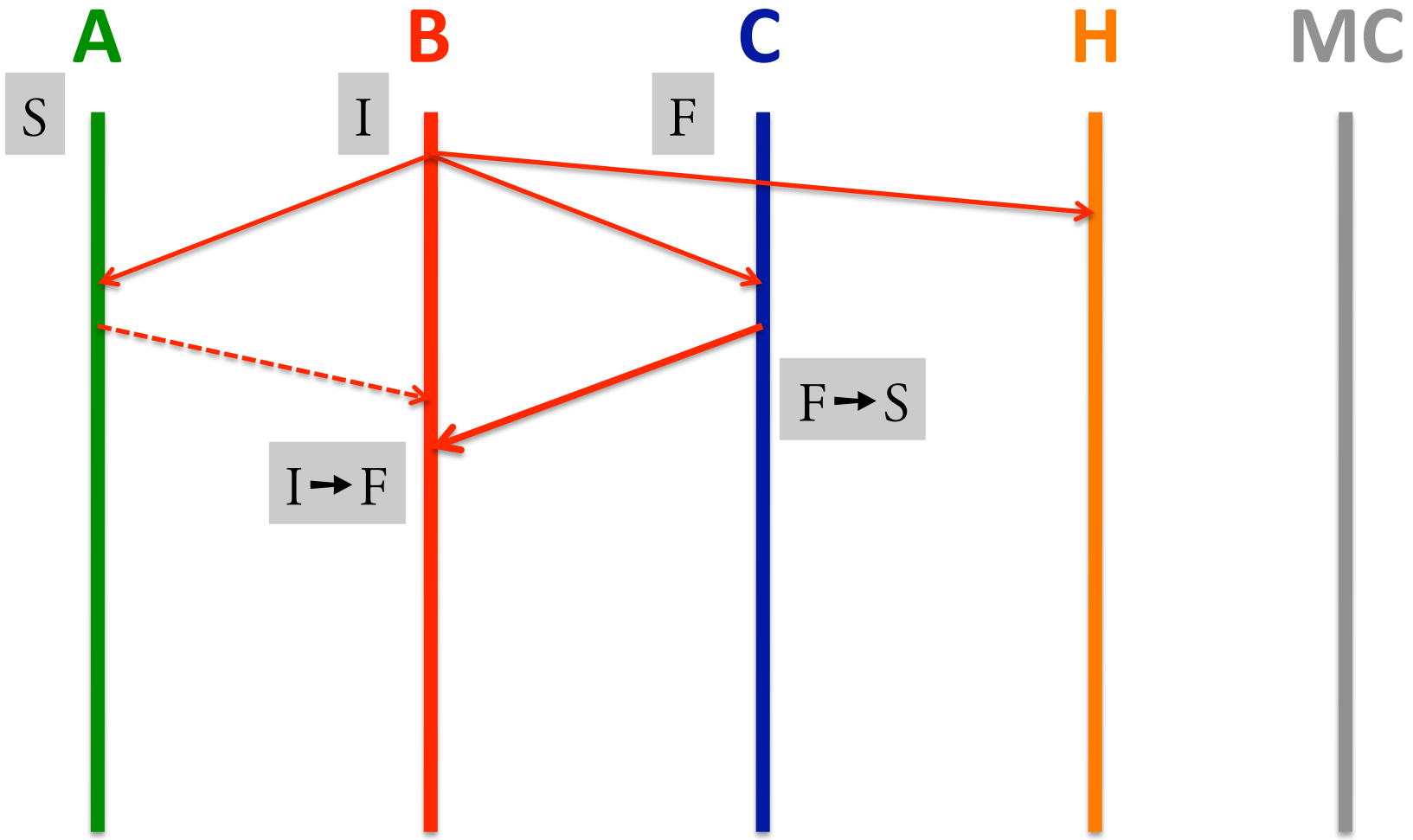
Read Uncached Line (MESI)



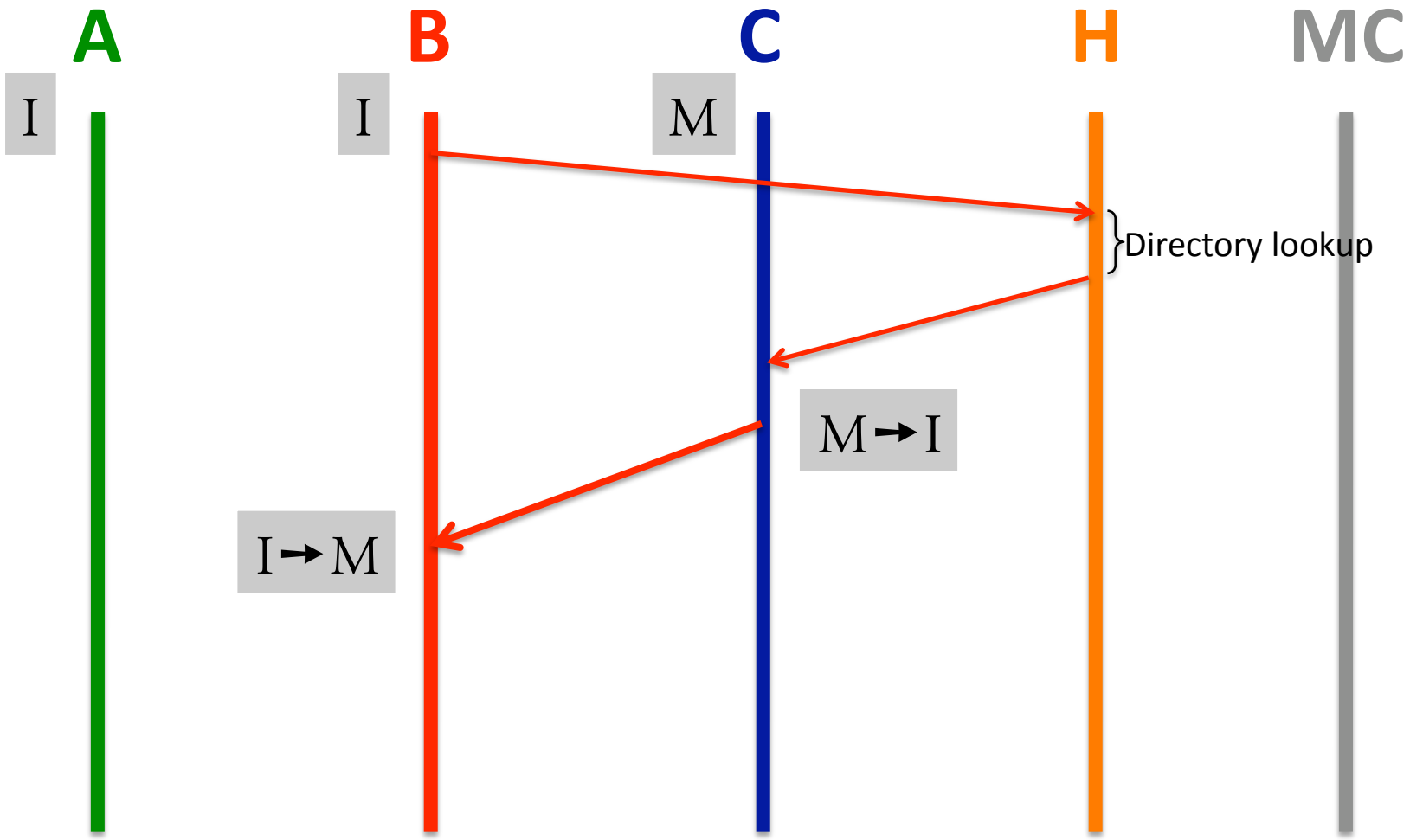
Read Shared (Directory)



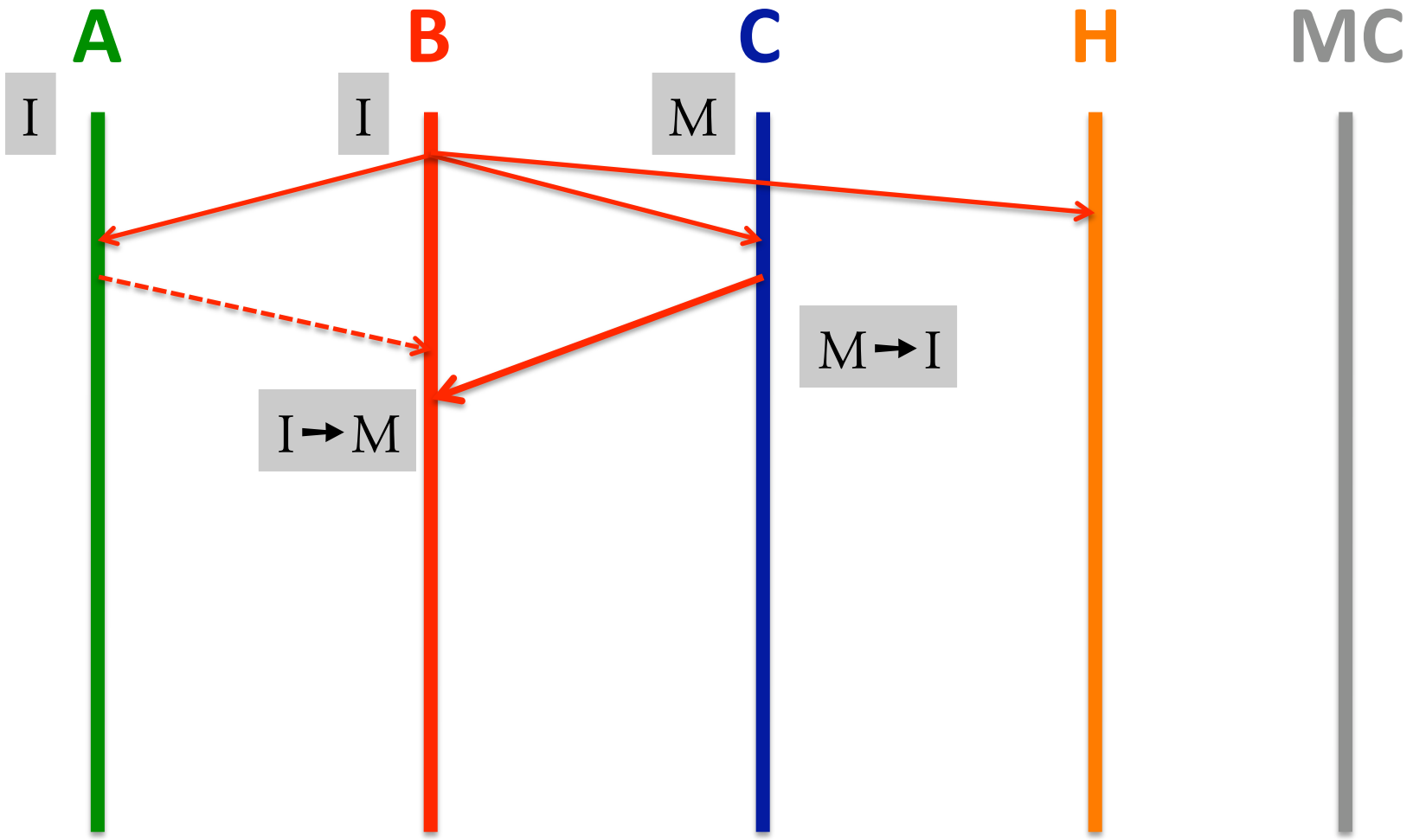
Read Shared (MESIF)



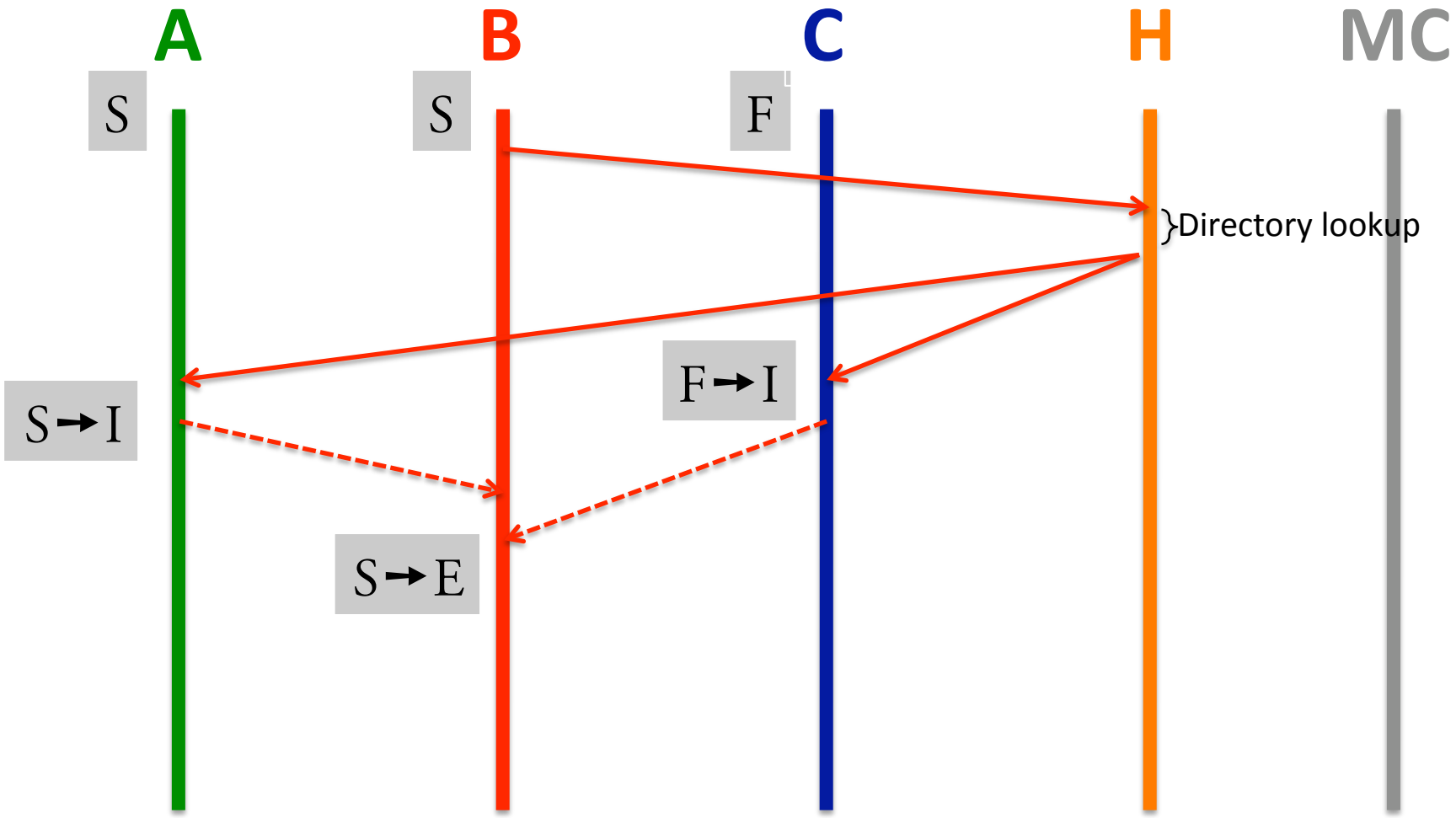
Read for Ownership (Directory)



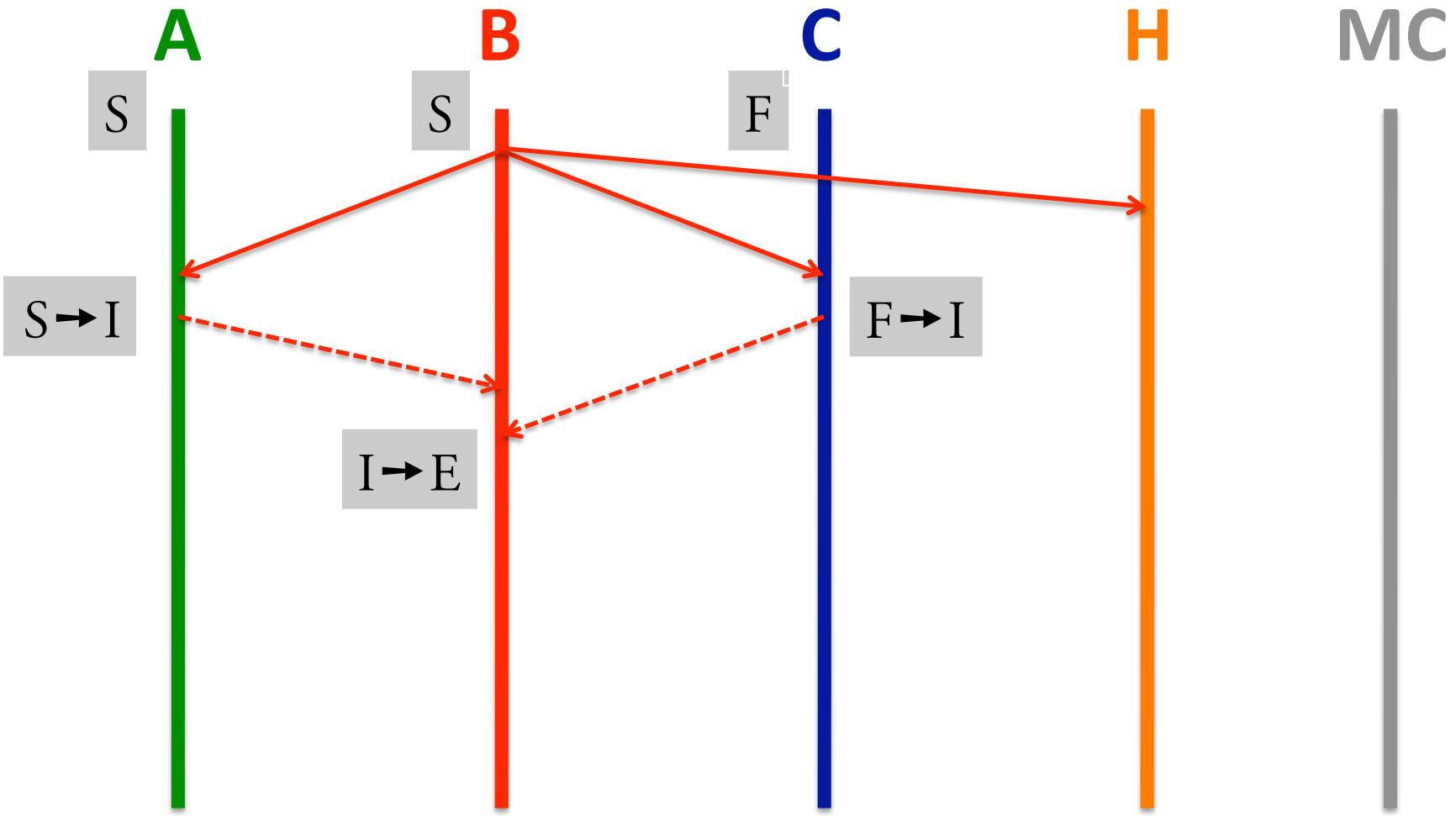
Read for Ownership (MESIF)



Upgrade/Invalidate (Directory)

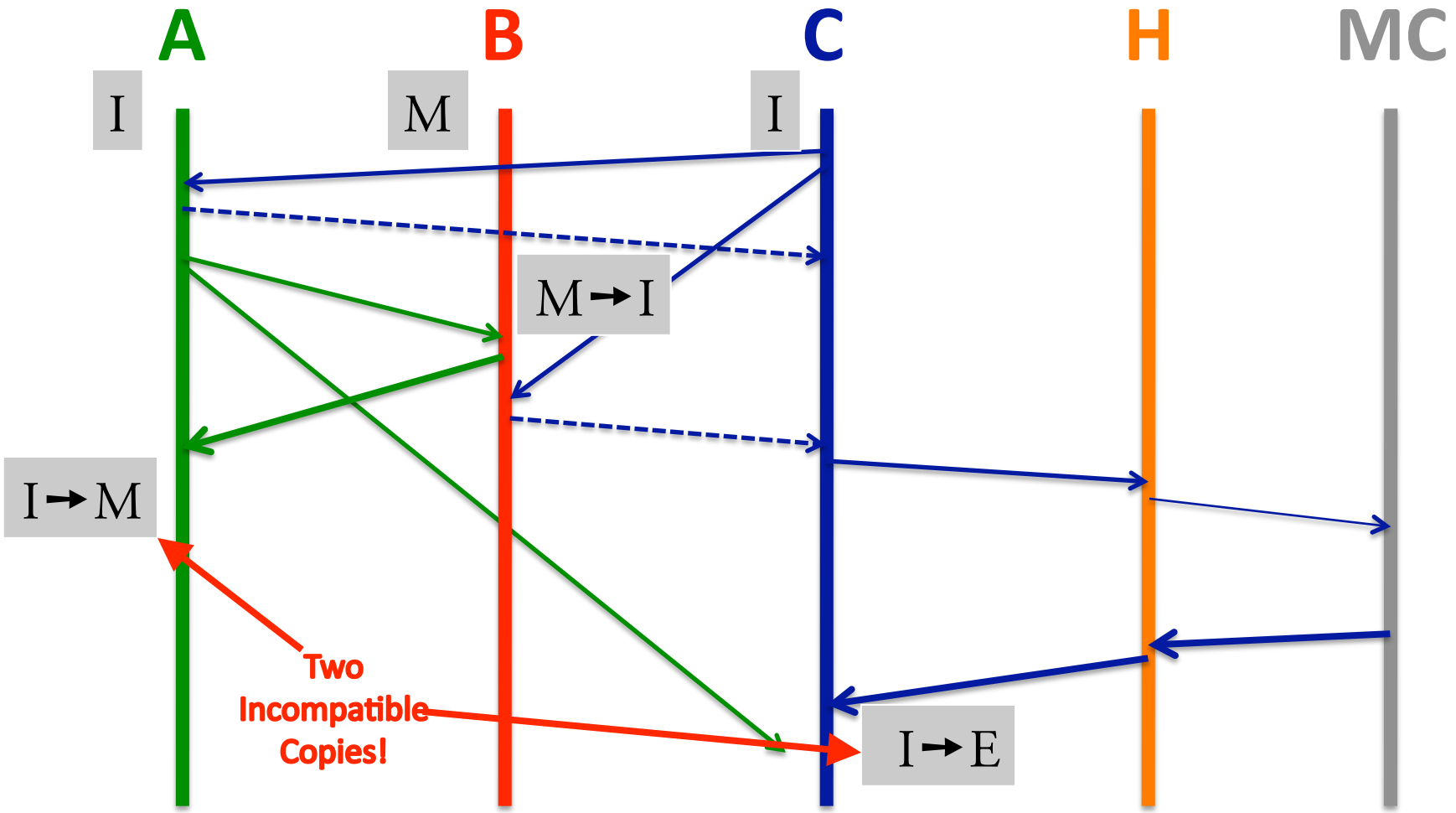


Upgrade/Invalidate (MESIF)



What about conflicts?

Read for Ownership (Conflict)

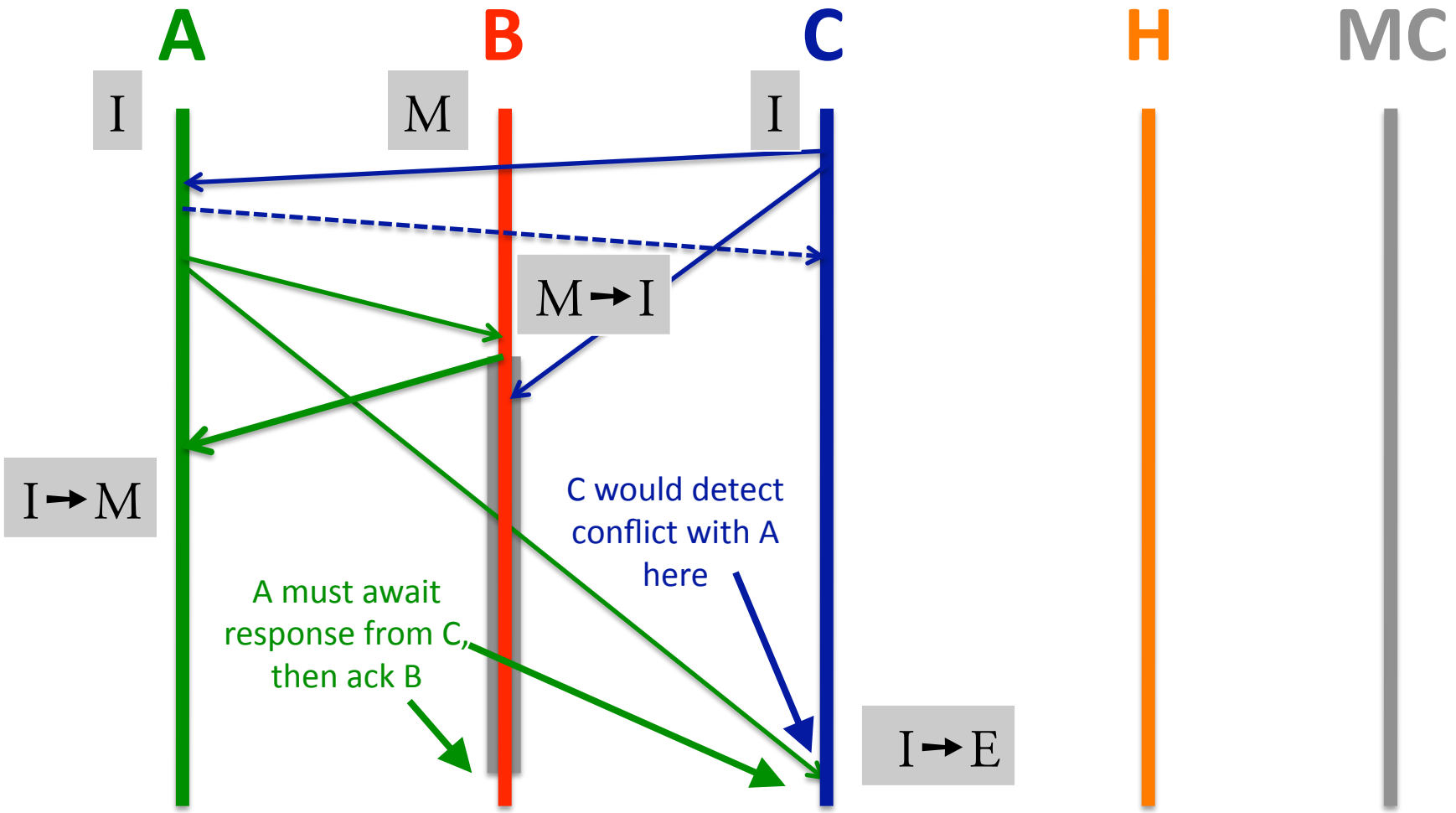


Conflicts

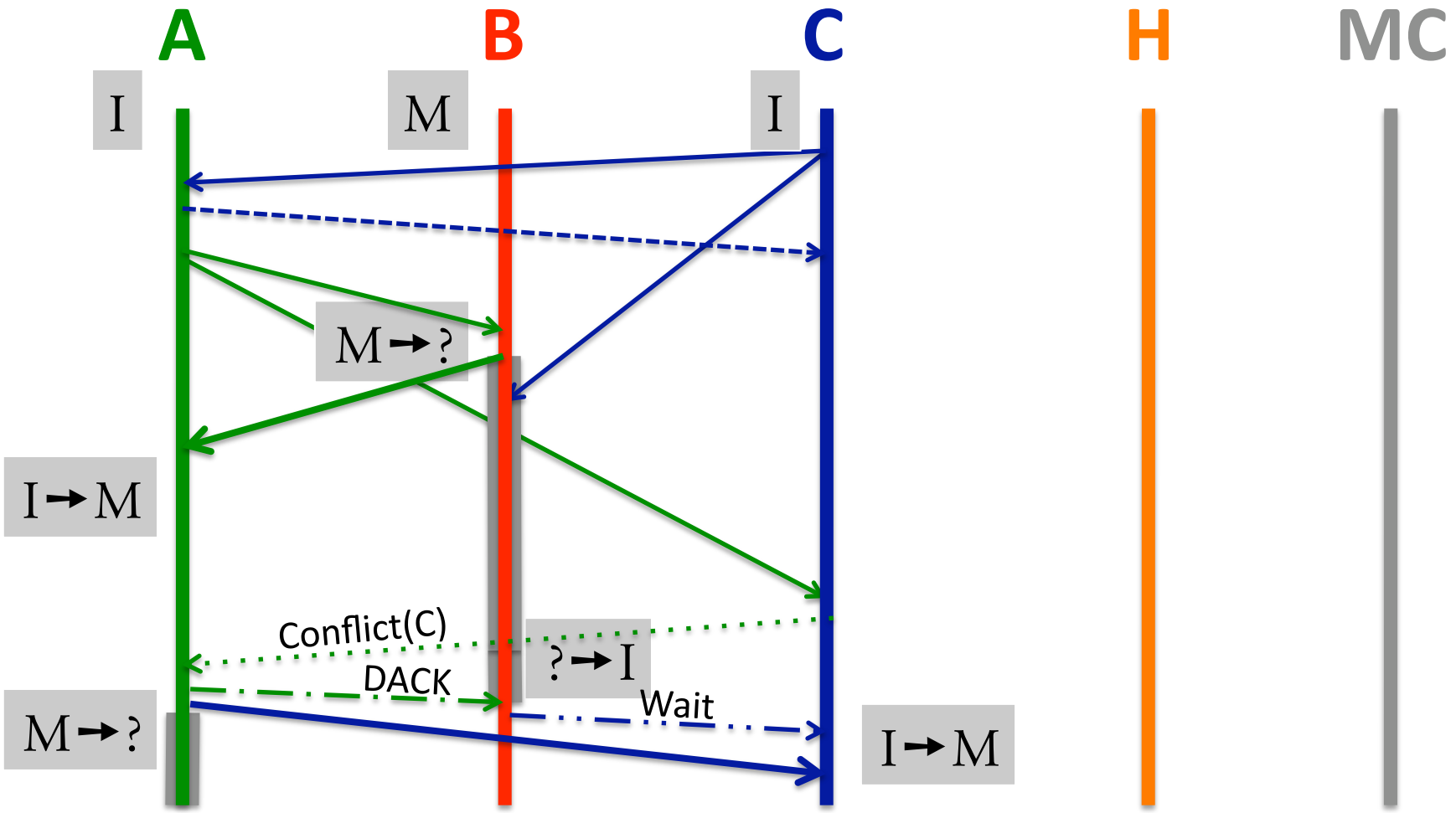
Conflicts between X and Y will always be detected if X sends cache request to Y and Y sends cache request to X

- Both may detect
- At least one will detect

Read for Ownership (Conflict)



Read for Ownership (Conflict)



Resolving Conflicts

- For more complex conflicts, a single point is required to sort out conflicts
- Home:
  - Does not determine the winner (FCFS)
  - Arranges for losers to be included

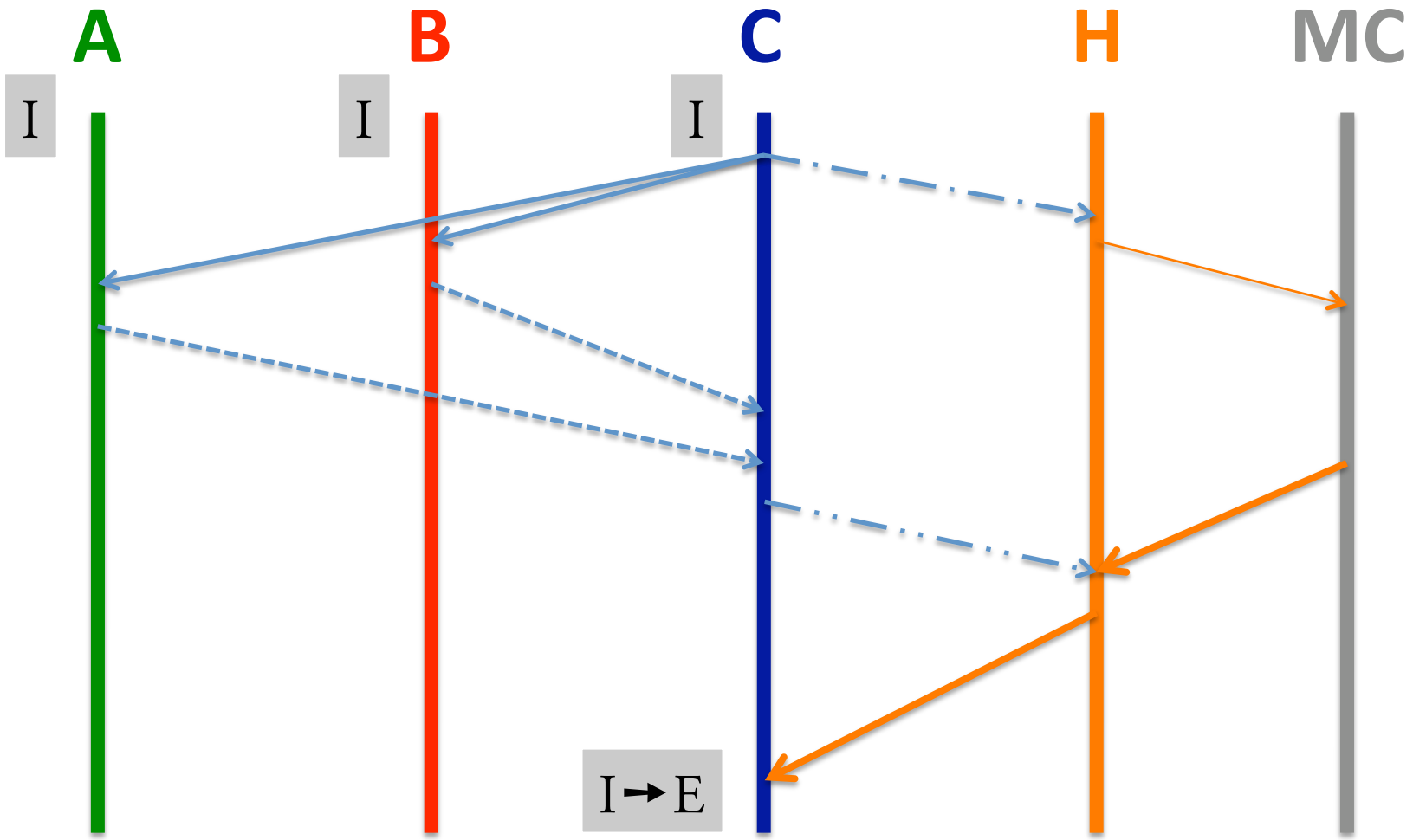
Global Ordering Point

- The *Global Ordering (G.O.)* point for a write instruction is the time when the system can guarantee no previous value can be observed; the instruction may now make visible a new value.
- A write instruction may finish executing an instruction and *retire* it, but cannot *commit* it until the G.O. point.

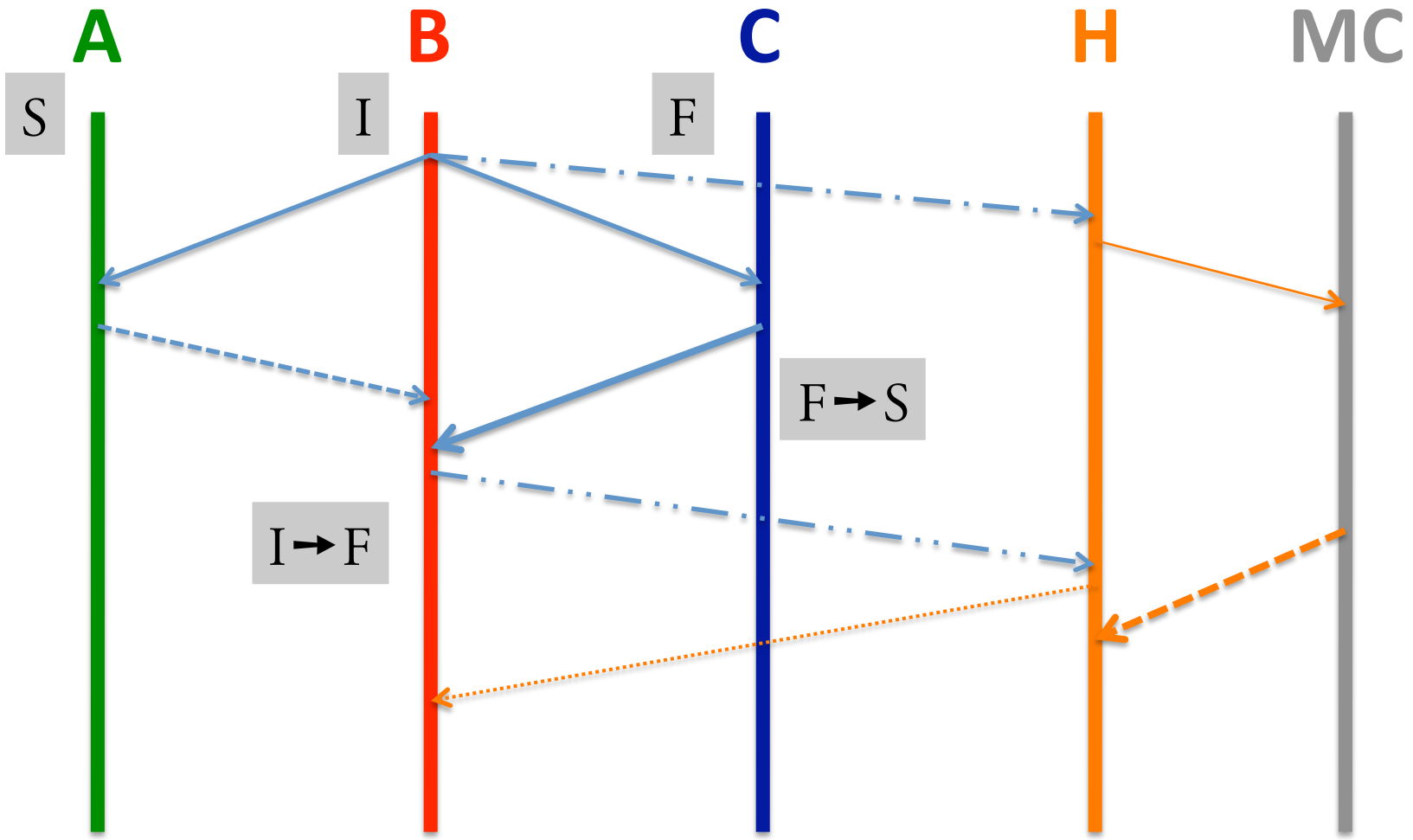
Making MESIF Safe

- In the MESIF protocol using HOME as the conflict resolution point requires
  - Home included with other nodes
    - Home does not respond
  - Cache-to-cache responses must report conflicts, if any
  - A second message to Home confirming or cancelling the original request
    - Message must indicate any conflicts observed
  - Memory must resolve conflicts and delay confirmation until resolved
  - Responses from memory to both requesters confirming the order
    - Winning processor is instructed to forward data to loser
    - Losing processor receives acknowledgement but no data
    - Losing processor receives data from winner
    - Multiple losers are queued
  - Processors may use data when it arrives, but may not make it visible until Home confirms.

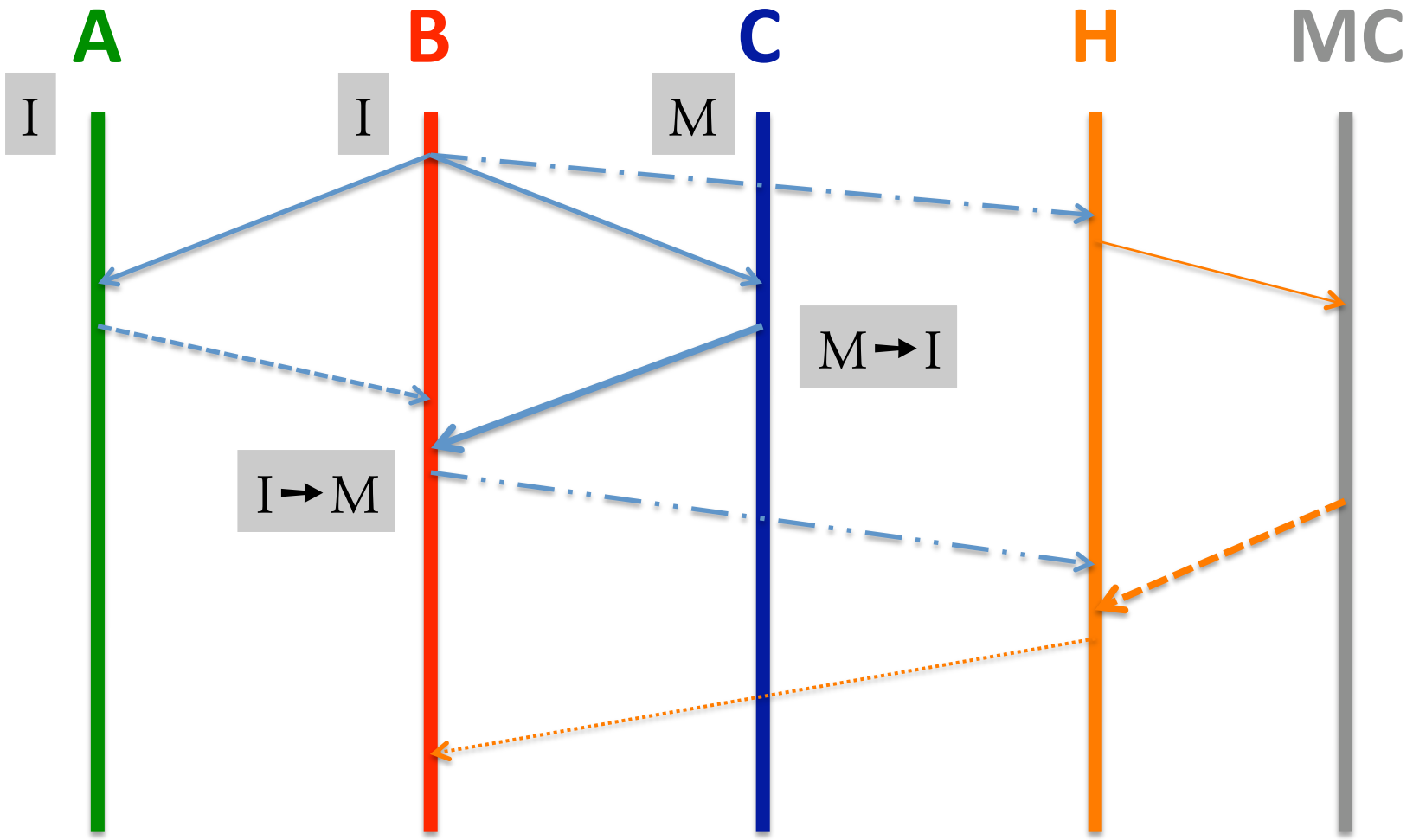
Read Uncached Line (MESIF)



Read Shared (MESIF)



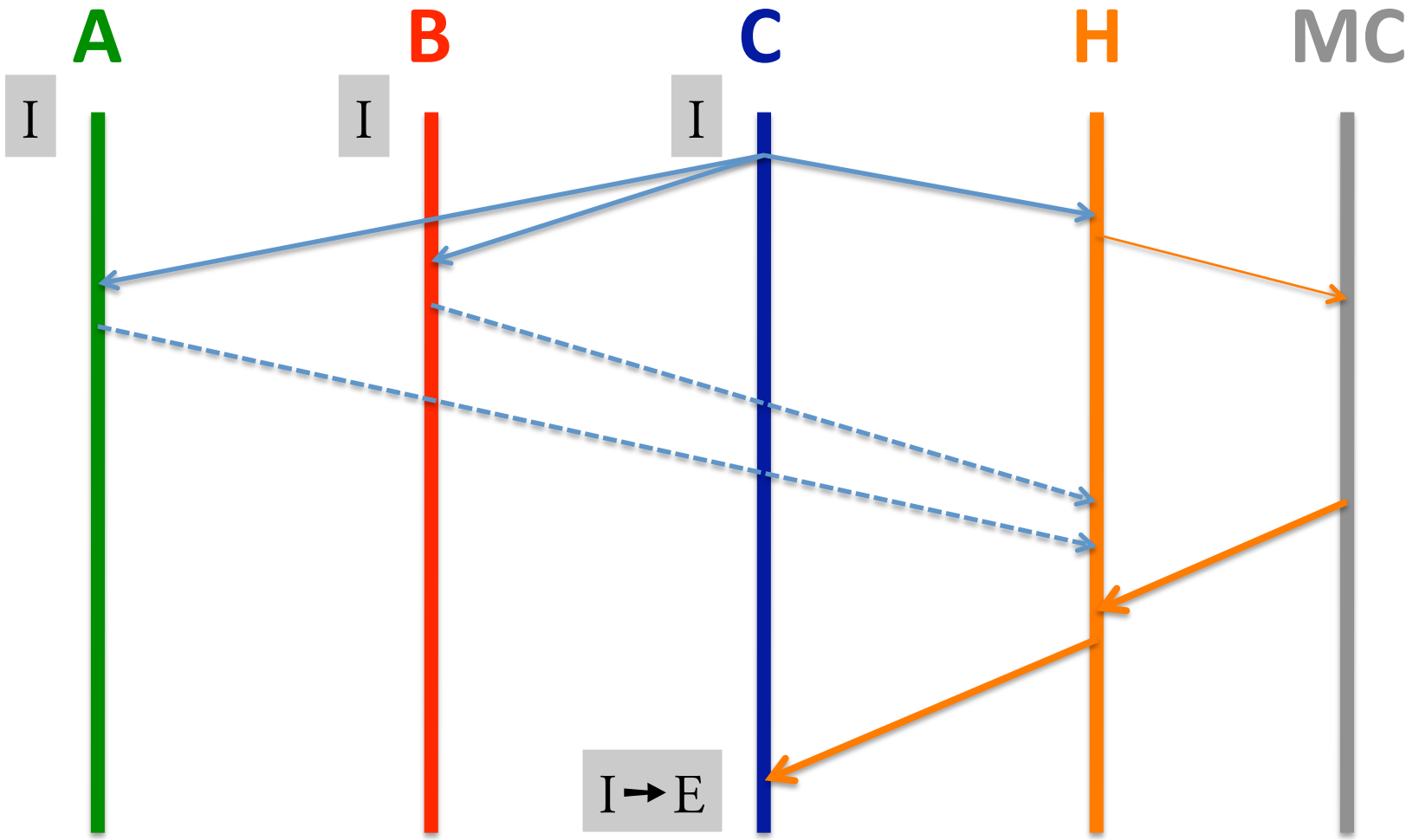
Read for Ownership (MESIF)



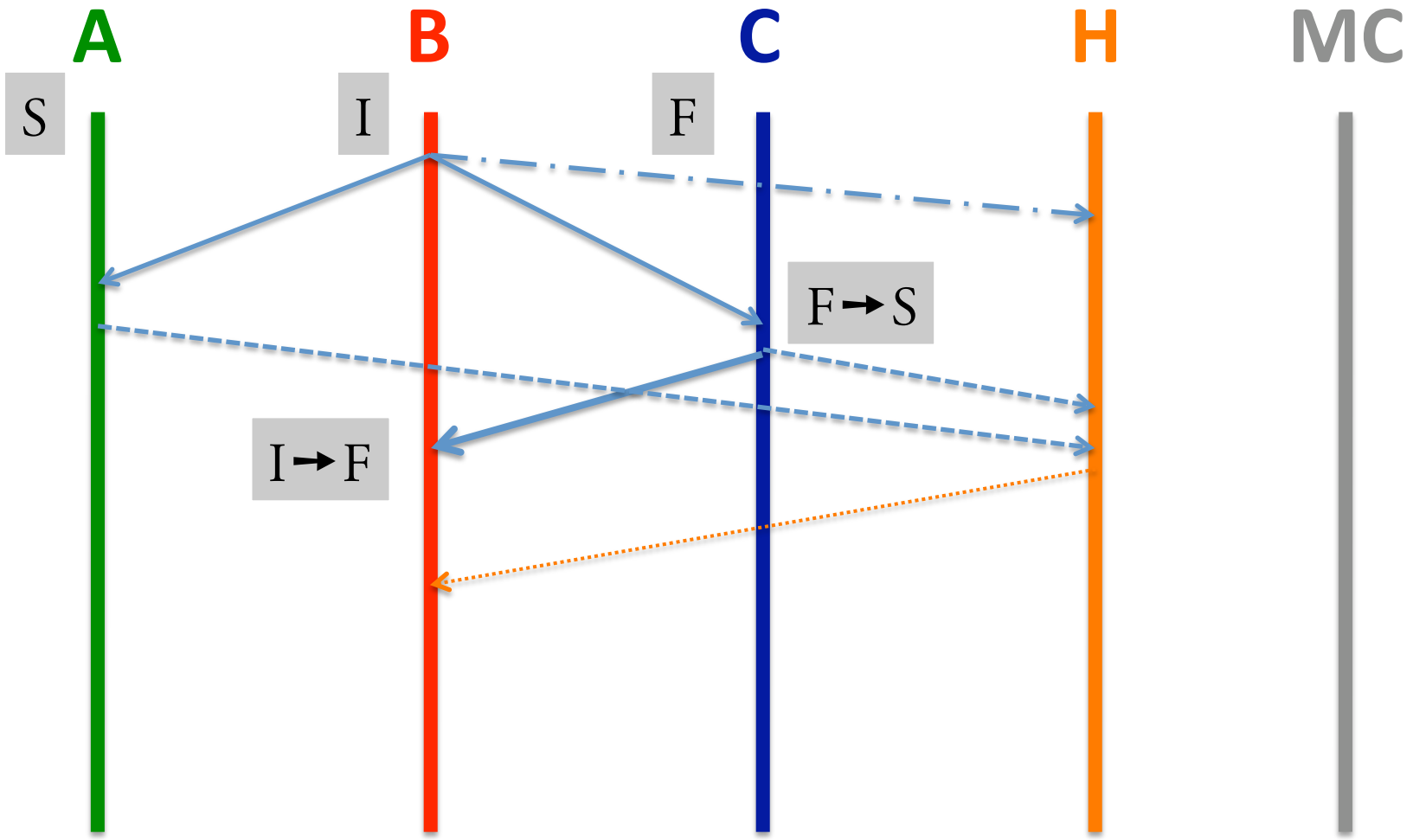
QPI Protocol

- Responses are sent to Home to collect rather than to requester
  - Data is sent directly to requester
- Home sends acknowledgement to requester
  - Data is included if not already delivered
- Home must be told of conflicts
  - Issues forwarding/wait directives

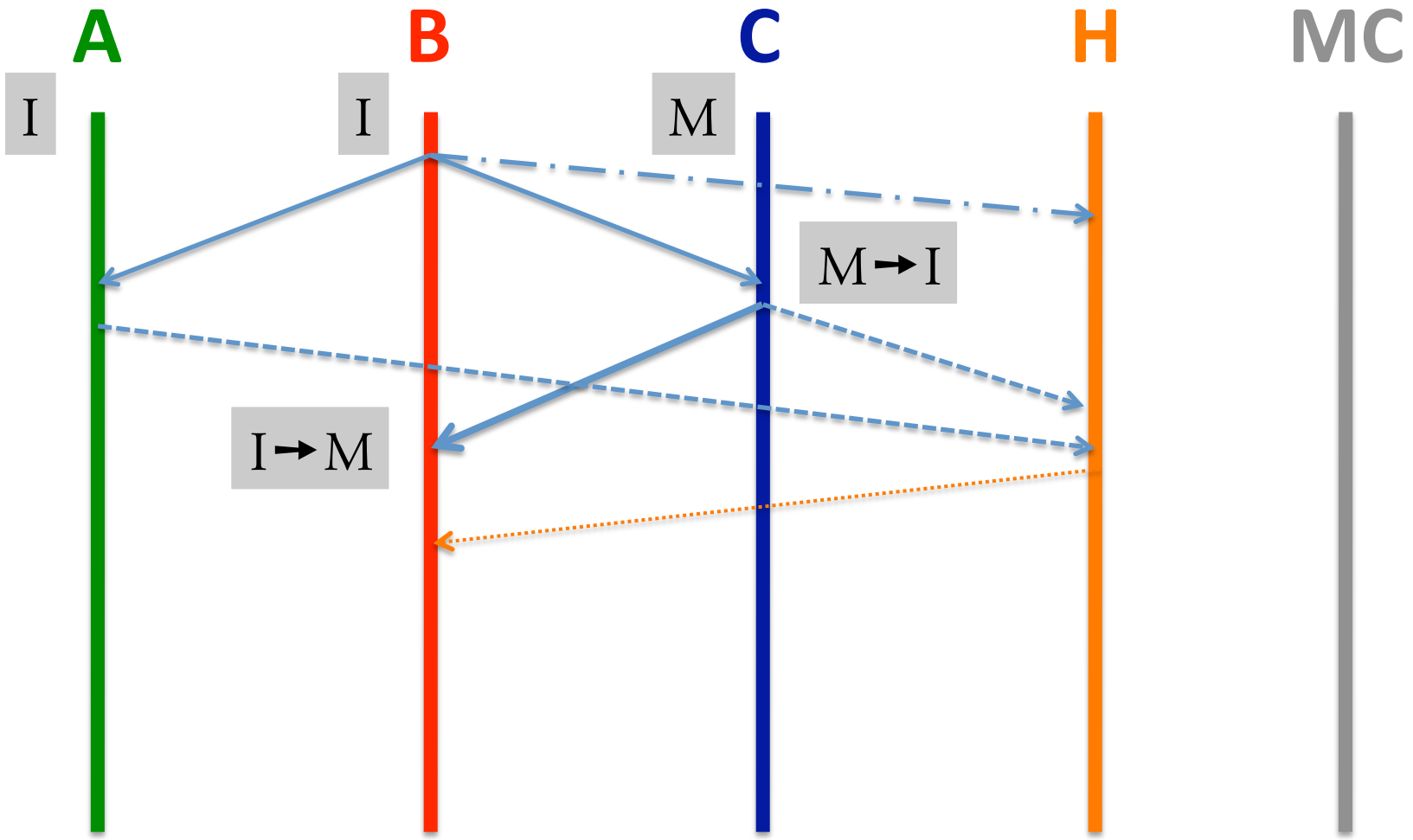
Read Uncached Line (QPI)



Read Shared (QPI?)



Read for Ownership (QPI)



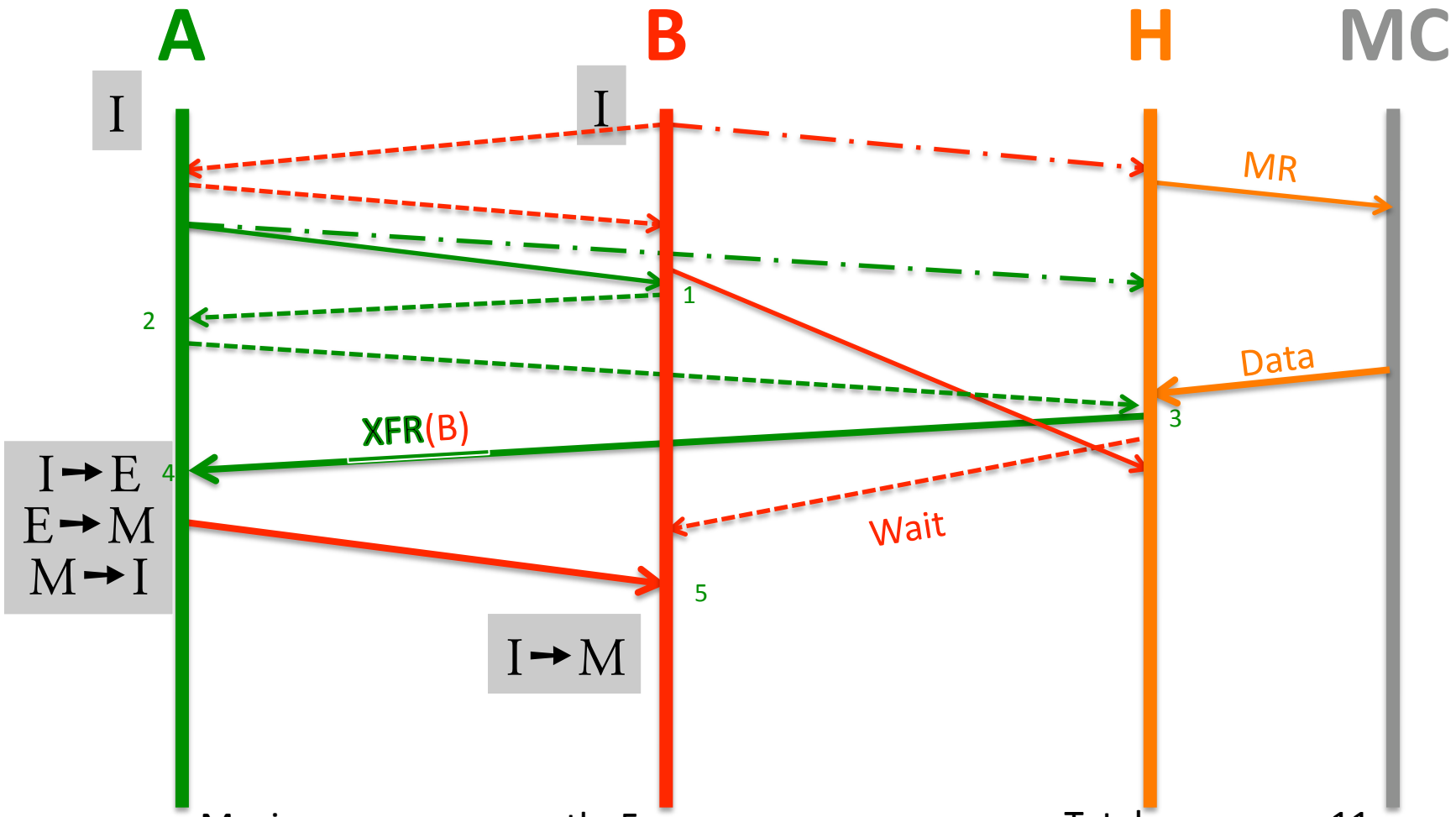
Comparison of QPI & MESIF

- QPI specification is not public
- Comparison is not straightforward

QPI & MESIF

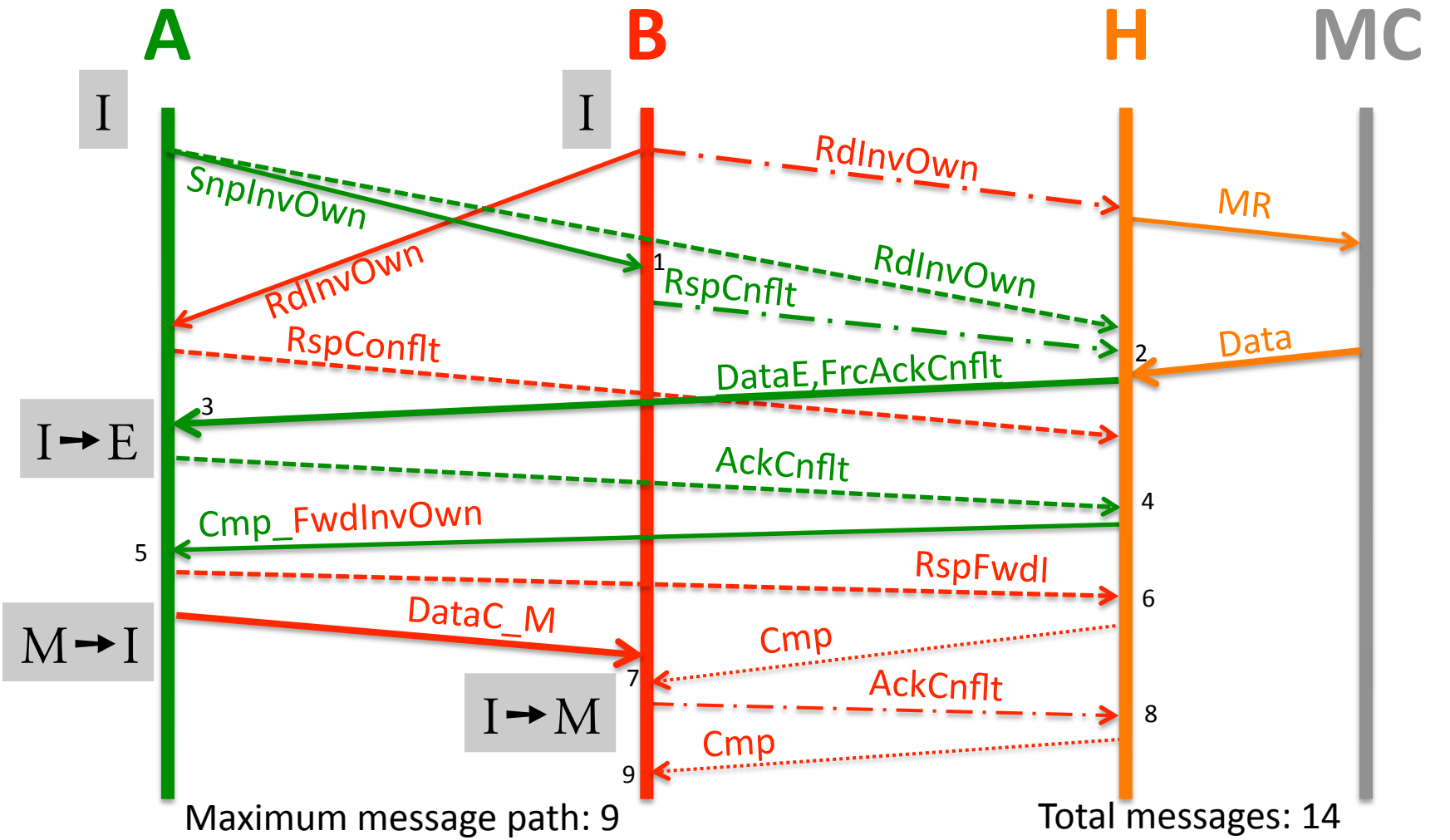
- QPI requires ordered messages to/from Home
- MESIF and QPI provide cache-to-cache data in same time
- In simple (common) cases, QPI requires:
  - one fewer message
  - shorter path to G.O. point
- MESIF requires fewer messages, has shorter path length in conflicts

Read for Ownership (MESIF)



Read for Ownership (QPI)

(QPI source: Maddox, Singh & Safranek)



Point-to-Point Networks

- Complete interconnect grows as N^2
- Build up hierarchically from small networks
- "Agent" appears as a single node (cache and/or memory), but represents the rest of the system

Open Questions

- Source snooping protocols have not been widely publicized or studied, but are now in products. What are the trade-offs?
- What are the benefits and costs of partial ordering of messages?
- Anticipating cache miss on unshared data trades off power for latency—can we do better?
  - Small systems: dynamic policies
  - Large systems: home snooping vs. hierarchical source-snoop
- Is this approach compatible with token coherence and other schemes to reduce broadcast?

Thank you!