# Multicast Snooping: A New Coherence Method Using A Multicast Address Network

**Ender Bilir, Ross Dickson, Ying Hu, Manoj Plakal, Daniel Sorin, Mark Hill & David Wood**

**Computer Sciences Department**
**University of Wisconsin**
**http://www.cs.wisc.edu/multifacet**

With sponsorship and/or participation from

| | |
|---|---|
| National Science Foundation | Compaq Computer |
| Wisconsin Romnes Fellowships | Intel |
| IBM Graduate Fellowship | Sun Microsystems |

Also Acknowledge: Anne Condon, Charles Fischer, & Milo Martin

# Summary

- Want Shared-Memory Multiprocessors
  - Find data directly (like snooping)
  - Scale larger than an SMP (like directories)
  - Use prediction between processors to do both
- Multicast Snooping
  - For each coherence transaction, predict multicast “mask”
  - Deliver multicasts on “ordered” network
  - Processors just “snoop” transactions
  - Simplified “directory” audits masks to handle incorrect ones
- Preliminary numbers for SPLASH-2 on 32 processors
  - Limit multicasts to 2-6 processors (<< 32)
  - Find data directly on 84-95% of multicasts
  - But ... <many buts>

# Outline

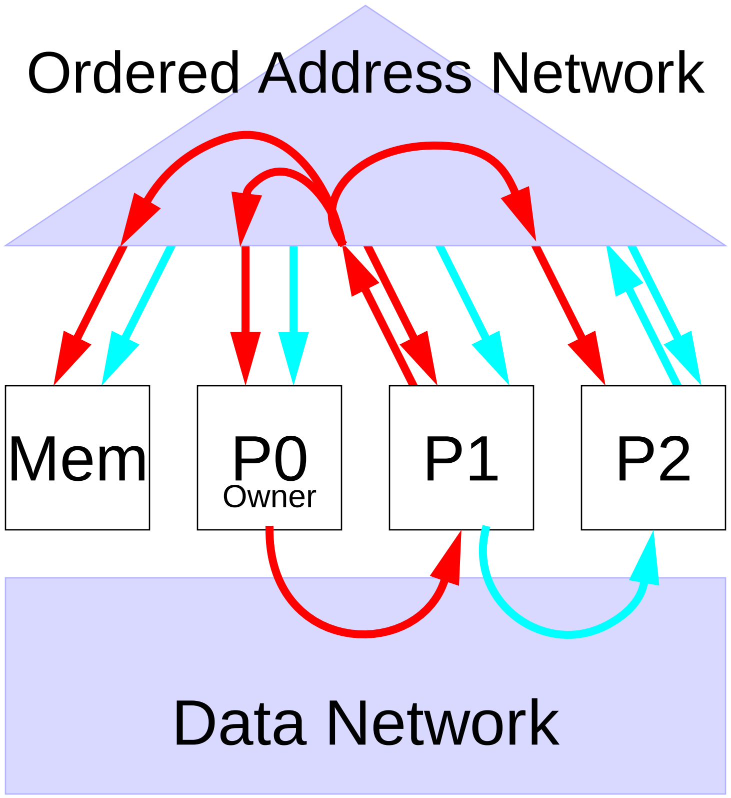
- Background
  - Snooping
  - Directories
- Multicast Snooping
- Some Experiments & Summary

# Terminology

- Write-Invalidate MSI Protocol States
  - Modified - read/write, exclusive, & (potentially) dirty ("Owner")
  - Shared - read-only, (potentially) shared, & clean
  - Invalid - no access, not valid or not present

- Processors Issue
  - GETS B (Get Shared) to go from Invalid to Shared
  - GETX B (Get eXclusive) to go from Invalid/Shared to Modified
  - PUTX B (Put eXclusive) to go from Modified to Invalid

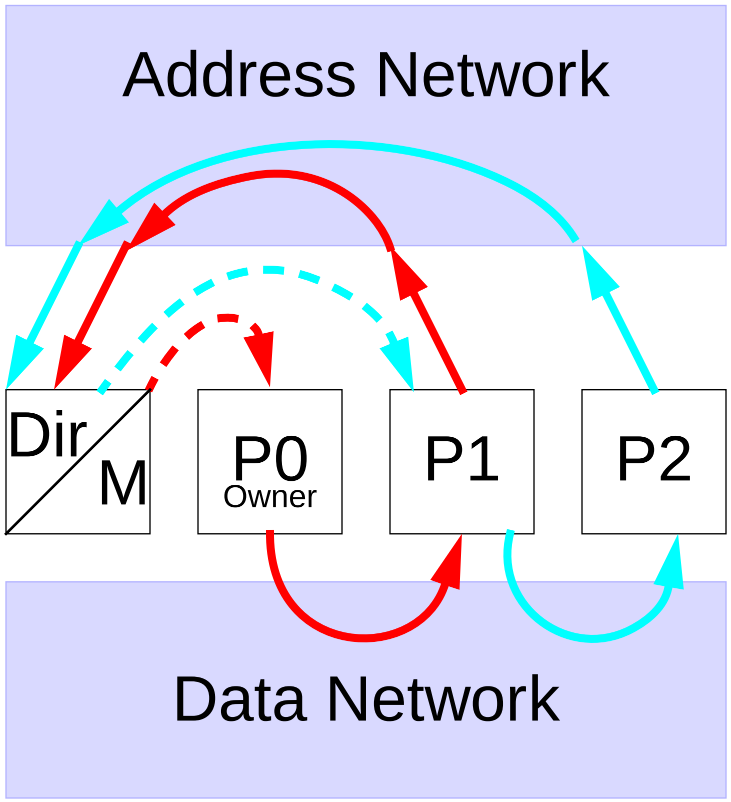
# Broadcast Snooping




Requires High Address Network and Snoop Bandwidth

- Initial State - Block B:
  - P0: Modified
  - P1&P2: Invalid
- Example
  - P1: GETX B & P2: GETX B
- Broadcast all coherence transactions
- Totally Ordered Address Network Serializes Transactions
- P1 “wins” in this example

# Directories




- Same initial state & example
- All coherence transactions to Directory
- Directory Serializes Transactions
- P1 “wins” in this example

Indirection Through Directory Can Increase Latency

# Comparison of Coherence Methods

| Coherence Attribute | Snooping | Directories | Multicast Snooping |
|---|---|---|---|
| Find previous owner directly? | Yes | Sometimes | Usually (good) |
| Always broadcast? | Yes | No | No (good) |
| Ordering without acknowledgements? | Yes | No | Yes (good) |
| Stateless (at memory)? | Yes | No | No but simpler |
| Ordered network? | Yes | No | Yes (challenge) |

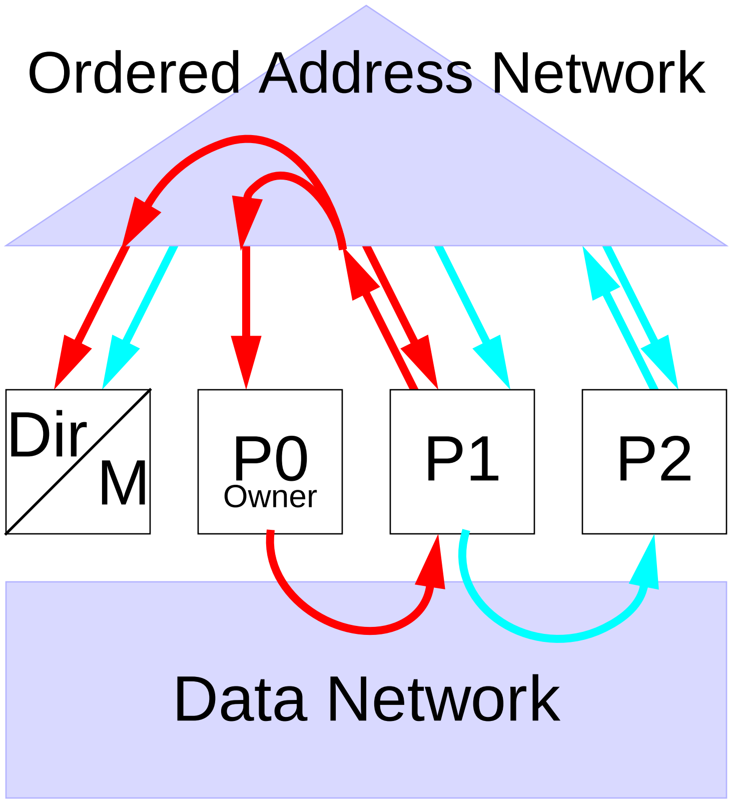
# Outline

- Background
- Multicast Snooping
  - Basic Operation
  - Mask Auditing & Prediction
  - Ordered Multicast Address Network
- Some Experiments & Summary

# Multicast Snooping

- On cache miss
  - Predict “multicast mask” (*e.g.*, bit vector of processors)
  - Issue transaction on multicast address network
- Networks
  - Address network that totally-orders address multicasts
  - Separate point-to-point data network
- Processors snoop all incoming transactions
  - If it’s your own, it “occurs” now
  - If another’s, then invalidate and/or respond
- Simplified directory (at memory)
  - Purpose: Allows masks to be wrong (explained later)

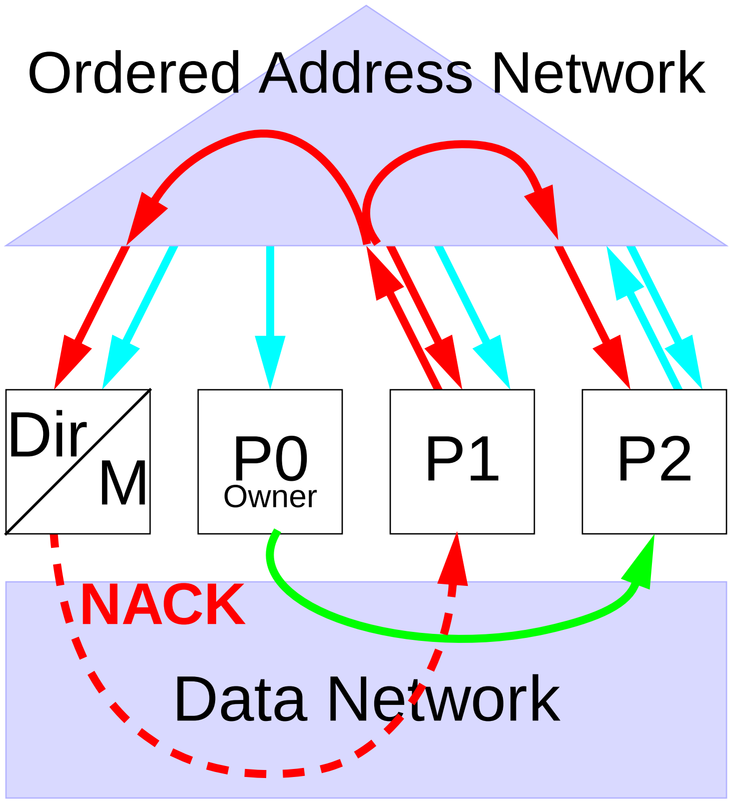
# Multicast Snooping Example




- Multicast coherence transactions
  - P1: GETX B <$Dir_B$, P0, P1>
  - P2: GETX B <$Dir_B$, P1, P2>
- Totally Ordered Multicast Network Serializes Transactions
- P1 “wins” in this example

No Indirection and Requires Less Address Network Bandwidth

# Multicast Snooping Example (Incorrect Mask)




- Multicast coherence transactions
  - P1: GETX B <Dir$_B$, P1, P2>
  - P2: GETX B <Dir$_B$, P0, P1, P2>
- Totally Ordered Multicast Network Serializes Transactions
- P1 "wins" in this example,

  But only gets NACK & must retry

Everything is Fine, With a Cost

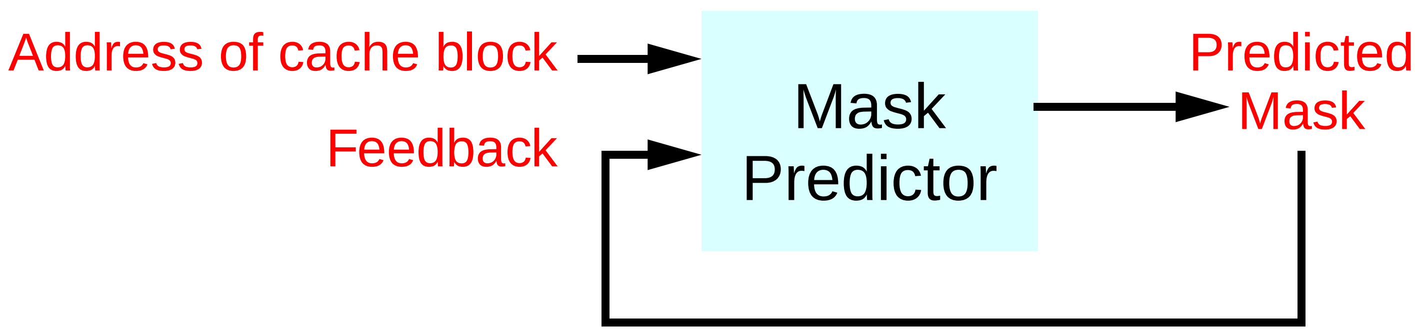
# Mask Auditing at Simplified Directory

- Simplified Directory
  - Always in block B's multicast mask
  - Tracks owner and sharers
  - Makes instantaneous transitions
- Audits Mask
  - Does GETS include owner?
  - Does GETX include owner & all sharers?
- Audit Result
  - Success: Update directory state & sometimes send block B
  - Failure: Reply with nack containing "better" mask

But how do processors predict masks?

## Predicting Masks

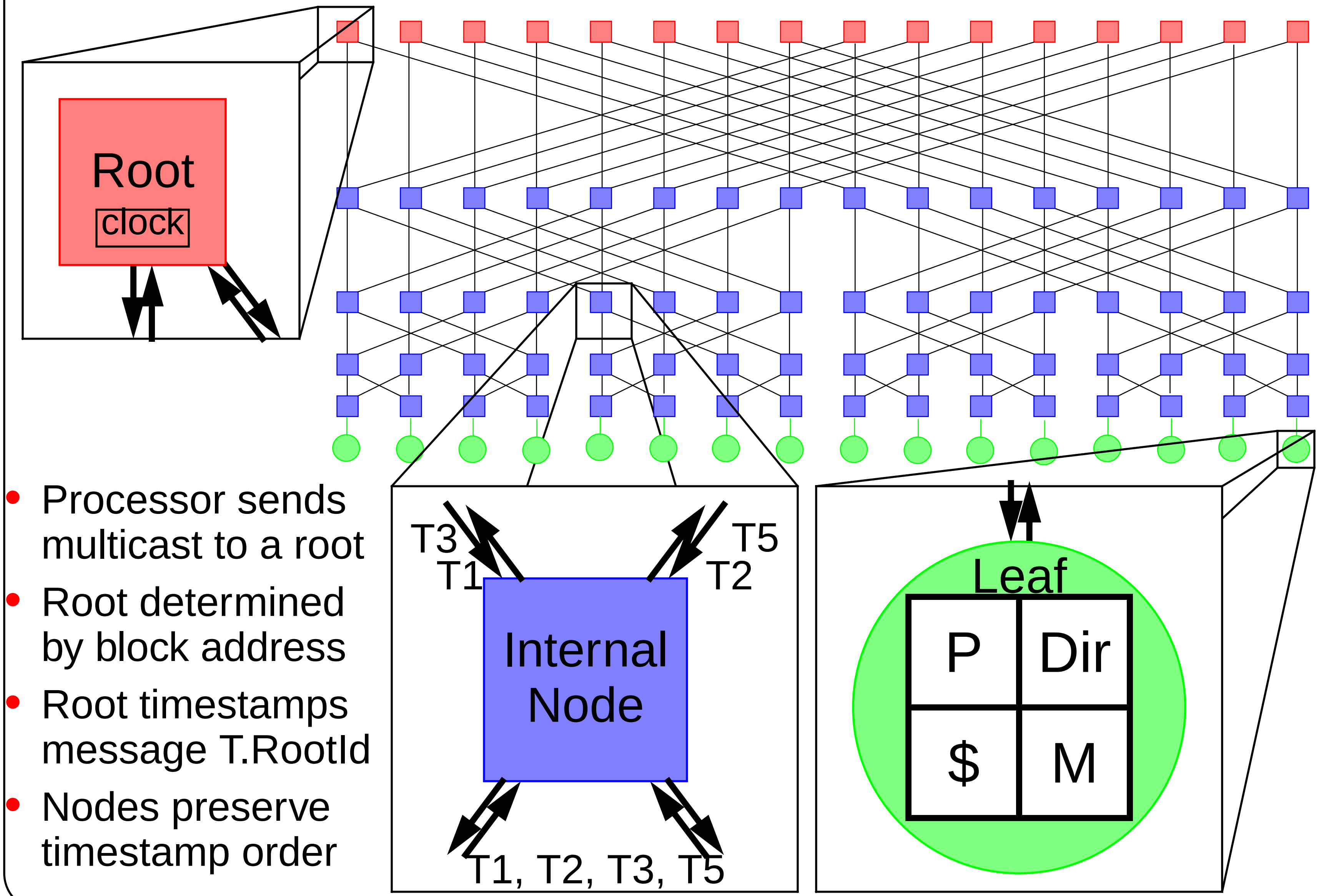
- Performed at Requesting Processor
  - Include owner (GETS/GETX) & all sharers (GETX only)
  - Exclude most other processors
- Many straightforward cases (e.g., stack, code, space-sharing)
- Many options (network load, PC, software, local/global, two-level)
- See paper for Sticky-Spatial(1) with a 4K-entry table



# Implementing an Ordered Multicast Address Network

- Address Network
  - Must create the illusion of total order of multicasts
  - Need NOT deliver a multicast to all destinations at the same time
- Wish List
  - High throughput for multicasts
  - No centralized bottlenecks
  - Low latency and cost (~ pipelined broadcast tree)
  - ...
- A Solution
  - Isotach Networks [Reynolds, Williams, & Wagner, IEEE TPDS 4/97]
  - Conceptually add logical timestamps to address messages

# Initial Proposal: Indirect Fat Tree




- Processor sends multicast to a root
- Root determined by block address
- Root timestamps message T.RootId
- Nodes preserve timestamp order

# Outline

- Background
- Multicast Snooping
- Some Experiments & Summary

# Hypothesis 1 of 2

- Hypothesis 1
  - Multicast Snooping uses significantly less address bandwidth than broadcast snooping.
- Experiment
  - Select mostly SPLASH-2 parallel benchmarks
  - Run 32-processor CC-NUMA on WWT2 execution-driven simulator
  - Run traces through Sticky-Spatial(1) mask predictor
  - Run traces through a 32-node binary fat tree simulator
- Result
  - Multicasts average 2-6 destinations (<< 32)
  - Allows initial network to deliver 2-5 multicasts per cycle (>> 1)

# Hypothesis 2 of 2

- Hypothesis 2
  - Multicast Snooping finds data directly significantly more often than directory protocols.
- Experiment: Same as for Hypothesis 1
- Result
  - Directories find data directly 45-92%
  - Multicast Snooping
    - Adds 11-50% (absolute) for fft, moldyn, ocean, & raytrace
    - Only 2-4% (absolute) for cholesky, lu, radix, & water-nq
    - Wider difference for “infinite” caches & preliminary TPC-B numbers
- Future: better benchmarks, larger systems, and timing results

## Summary

- Multicast Snooping
  - For each coherence transaction, predict multicast “mask”
  - Deliver multicasts on “ordered” network
  - Processors just “snoop” transactions
  - Simplified “directory” audits masks to handle incorrect ones
- Tentatively, multicast snooping allows a multiprocessor to:
  - Behave like snooping if address bandwidth adequate
  - Gracefully degrade toward directory-based solution
- More generally, Wisconsin Multifacet Project
  - Other opportunities for system-wide prediction & speculation
  - **`http://www.cs.wisc.edu/multifacet`**

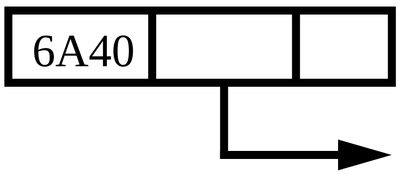
# BACKUP SLIDES

# MSI Multicast Snooping

| Requestor | | Memory | | | | | Other Processors in Mask | | | Requestor | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Trans-action** | **Old State** | **Old State** | **Owner in mask?** | **All in mask?** | **Send to requestor** | **New State** | **Old State** | **Send to requestor** | **New State** | **New State** | **Success?** |
| GETS | I | S, I | yes | x | data_ack | S | | | | S | yes |
| | | M(q) | yes | x | | S | M | data_ack, data to mem | S | S | yes |
| | | | no | no | nack | same | | | | I | no |
| GETX | S,I | S, I | yes | yes | data_ack | M(r) | S | | I | M | yes |
| | | S | x | no | nack | same | S | | I | S | no |
| | | M(q) | yes | yes | | M(r) | M | data_ack | I | M | yes |
| | | M(q) | no | no | nack | same | S | | I | same | no |
| PUTX | M | M(r) | yes | yes | | I | | | | I | yes |
| | I | I, S, M(q) | x | x | | same | | | | same | no |

Columns 1 and 2 give the requesting processor’s transaction and state when it sees its own transaction.Columns 3-5 give the states a transaction can encounter at memory, while Columns 6-7 give the memory’s response. Memory state M is augmented with “(r)” if the requestor is (was) the owner and with “(q)” otherwise. An “x” denotes “don’t care.” Column 8 gives the state that other processors may be in when they see a transaction, while Columns 9-10 give their response. Cases where these processors do nothing are omitted for brevity (observing a GETS in S, observing a PUTX in I, and when omitted from a multicast mask). Finally, Columns 11-12 give the requesting processor’s final state and whether the transaction was successful. All other cases are impossible.

# Predicting Masks: Sticky-Spatial(1)




| Tag | Multicast mask | Last invalidator |
|---|---|---|
| | | |
| 3A00 | 00000011 | 0 |
| 6A40 | 10010000 | 3 |
| 6A80 | 00100000 | 2 |
| | | |

OR together 1 spatial neighbor on each side

| | |
|---|---|
| Mask for block | = 10010000 |
| Mask for neighbor#1 | = 00000011 |
| Mask for neighbor#2 | = 00100000 |
| Mask for requester & directory | = 00000110 |
| Predicted multicast mask | = 10110111 |

Intuition: Accessing an array or record

# Implementing an Ordered Multicast Address Network

- On each network pulse T (pretend that the network in synchronous)
- Each Root J
  - Selects a queued multicast transaction (if any)
  - Assigns it logical timestamp T.J
  - Sends it down to the left, right, or both
  - And sends timestamp-only null messages down any idle down links
- Each Routing Node - Performs a “merge”
  - Selects a queued multicast transaction with oldest timestamp (if any)
  - Sends it down to the left, right, or both
  - And sends timestamp-only messages down any idle down links
- By induction, all links deliver transactions in pulse order
- Processors re-create multicast order using logical timestamp

# (A) Select Parallel Benchmarks

- Use (mostly) SPLASH-2 Benchmarks

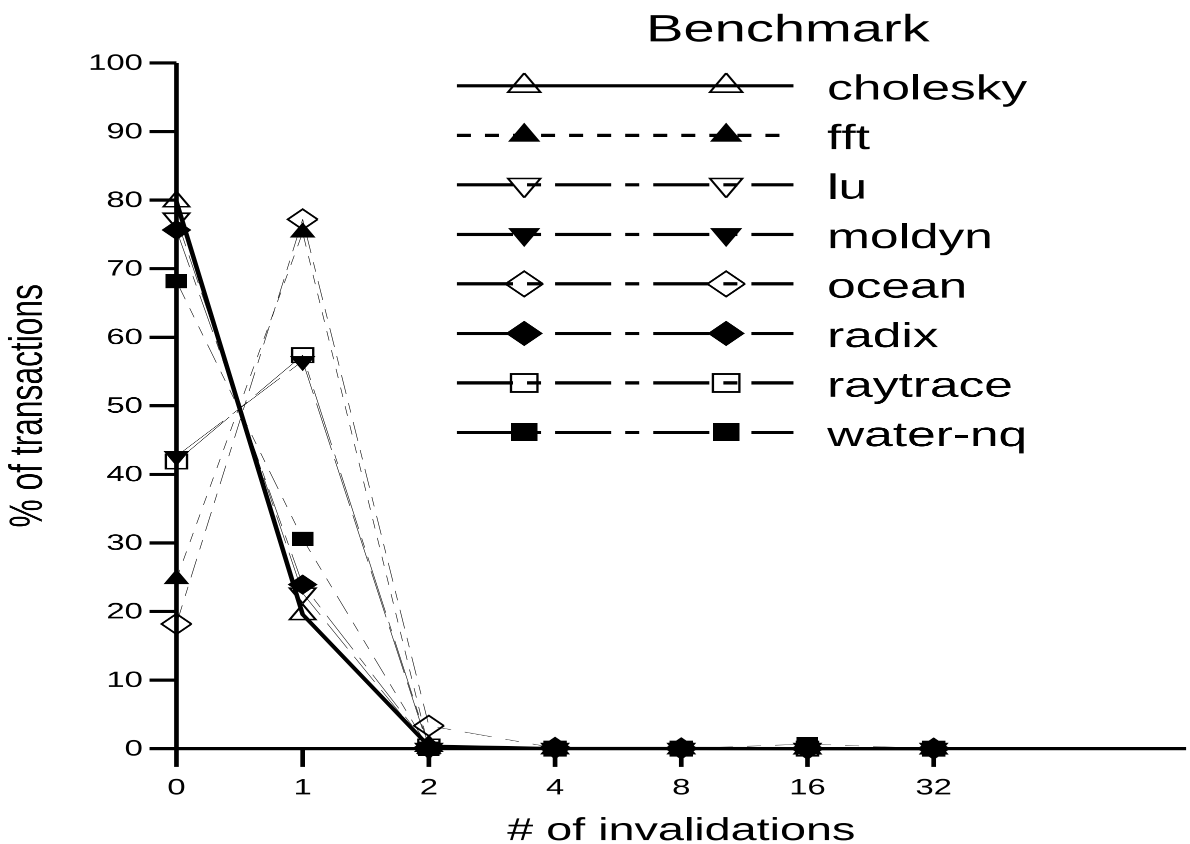
| Benchmark | **Description of Application** | **Input Data Set** |
|---|---|---|
| cholesky | Blocked sparse matrix Cholesky factorization | tk16.O from SPLASH-2 |
| fft | Complex 1-D radix-$\sqrt{n}$ 6-step FFT | 64K points |
| lu | Blocked dense matrix LU factorization | 512x512 matrices, 16x16 blocks |
| moldyn | Simulation of molecular dynamics | 2048 particles, 15 iterations |
| ocean | Simulates large-scale ocean movements | 130x130 ocean |
| radix | Integer radix sort | 1M integers, radix 1024 |
| raytrace | 3-D scene rendering using raytracing | teapot from SPLASH-2 |
| water-nq | Quadratic-time simulation of water molecules | 512 molecules |

# (B) Run 32-processor execution-driven simulator

- Use Wisconsin Wind Tunnel II (WWT2)

| **Parameter** | **Value** |
|---|---|
| # of processors | 32 |
| Type of system | CC-NUMA |
| Coherence mechanism | Directory protocol: full-map, write-invalidate, 3-state MSI |
| Data memory hierarchy | L1 cache, SPARC MBus, Local memory, Remote Block cache |
| L1 data cache | 128KB, direct-mapped, 32-byte blocks, write-back |
| Remote block cache | 512KB, direct-mapped, 32-byte blocks, writeback inclusion with L1 cache for read-write blocks |
| Local memory | 96MB |

# (1) Is the mean number of sharers encountered by a coherence transaction small?




- Yes, so multicasts can have far fewer destinations than broadcasts

## (2) Can plausible mask predictors usually include all necessary processors and limit multicasts to an average number of destinations much smaller than all processors?

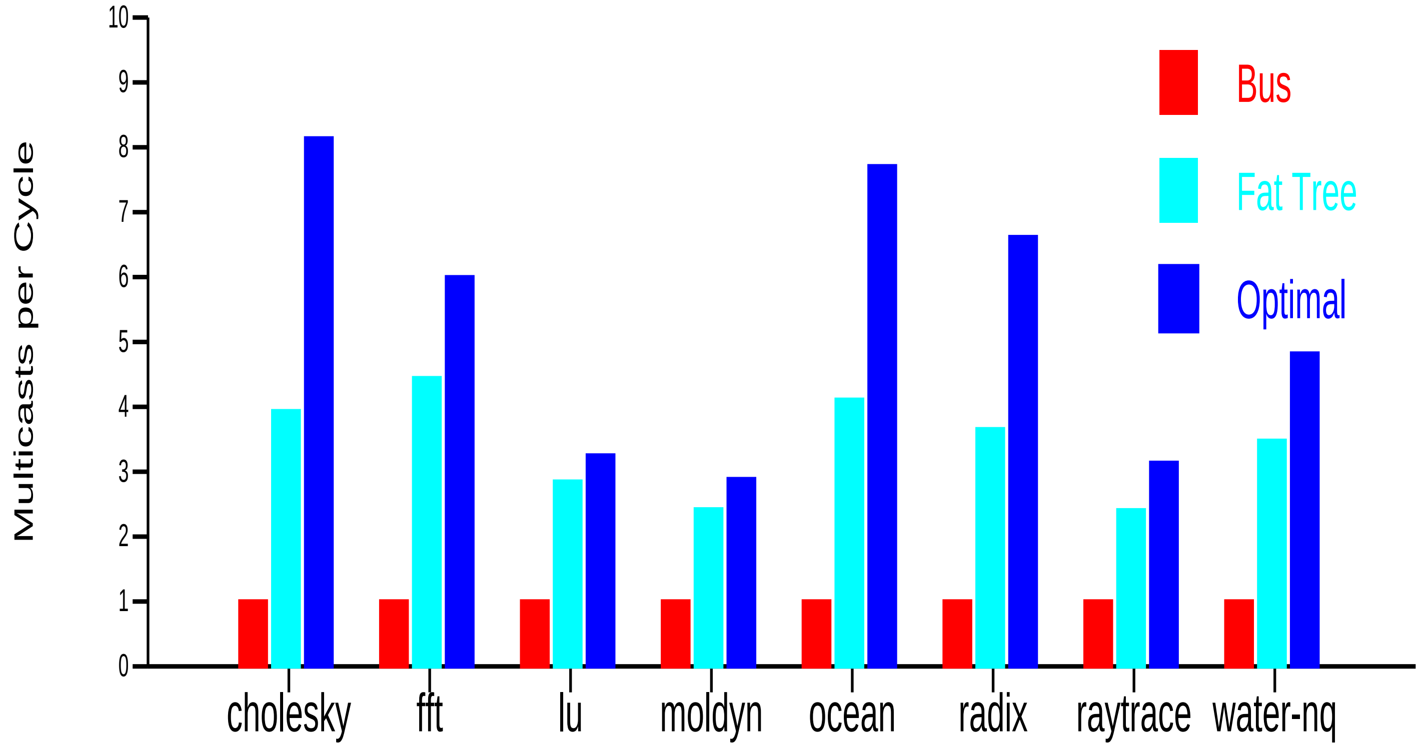
| Benchmark | Prediction Accuracy (%) | Blocks at Home (%) | Average Nodes in Multicast (of 32) |
|---|---|---|---|
| cholesky | 94 | 92 | 3.4 |
| fft | 73* | 57 | 3.2 |
| lu | 95 | 93 | 2.4 |
| moldyn | 88* | 56 | 5.4 |
| ocean | 95* | 45 | 3.4 |
| radix | 84 | 80 | 3.0 |
| raytrace | 86* | 75 | 5.6 |
| water-nq | 88 | 85 | 3.8 |

- Yes, so mask prediction can be effective

# (D) Run through Network Simulator

- Simple Java Program
- Takes traces of mask predictions from previous step
- Simulates multicast address network
  - Binary fat tree
  - 32 processors
  - 32 roots
- Computes network throughput
- Network latency not meaningful since input trace does not have meaningful time

# (3) Can our initial network deliver many multicasts per network cycle?




- Yes, much larger than broadcast snooping's one per cycle