“Multicast Snooping: A New Coherence Method Using a Multicast Address Network”

E. Ender Bilir, Ross M. Dickson, Ying Hu, Manoj Plakal, Daniel J. Sorin, Mark D. Hill, David A. Wood
University of Wisconsin

Presented By: Daniel Gaultney
Motivation

- Broadcasting scales poorly
- Broadcasting to all cores is too conservative: only cores that share data need to see a broadcast.
- Directory protocols require indirect transactions, additional messages, and can have concurrency problems.
Multicast Snooping Coherence

• A “Mask” is used to predict which processors need a given broadcast.
  – A simplified directory responds with an ack, semiack, or a nack.
  – Memory states are explicit (like directory protocols)
• The “O” State is expanded
  – A block goes into the O state after a semiack
  – What are the advantages of this?
• Forward progress ensured by broadcasting after k retries.
Mask Prediction

• Speculates which processors will need a given message.

• The StickySpatial(k) predictor was used
  – Blocks are tagged with a “Multicast mask” and with their last invalidator
  – GETX->Logical OR of k surrounding masks
  – GETS->Last invalidator, requestor, & directory
Interconnect Network

• K-ary fat tree network
• Based on an Isotach-like network
• Importance to multicast snooping: Many multicasts can be transmitted per cycle
• Achieves roughly 50% of optimum network throughput.
Questions

• Is this added complexity compared to directory protocols a good trade-off?
  – Directory protocols are within 5-10% for all but two benchmarks
• Is the “Fat Tree” network an optimal interconnect network?
  – An Isotach-like network has more stringent requirements.
  – Could a more ideal network be created in absence of these requirements?
• Is the “Semi-ack” really necessary?
  – The “O” state was not even simulated
  – Performance increase if the mask predictor recognized a semi-ack?
• StickySpatial seems rather crude.
  – Possibly have some method of recognizing unrelated blocks?