# ON LID ALLOCATION AND ASSIGNMENT IN INFINIBAND NETWORKS

Wickus Nienaber, Xin Yuan, Zhenhai Duan

Department of Computer Science

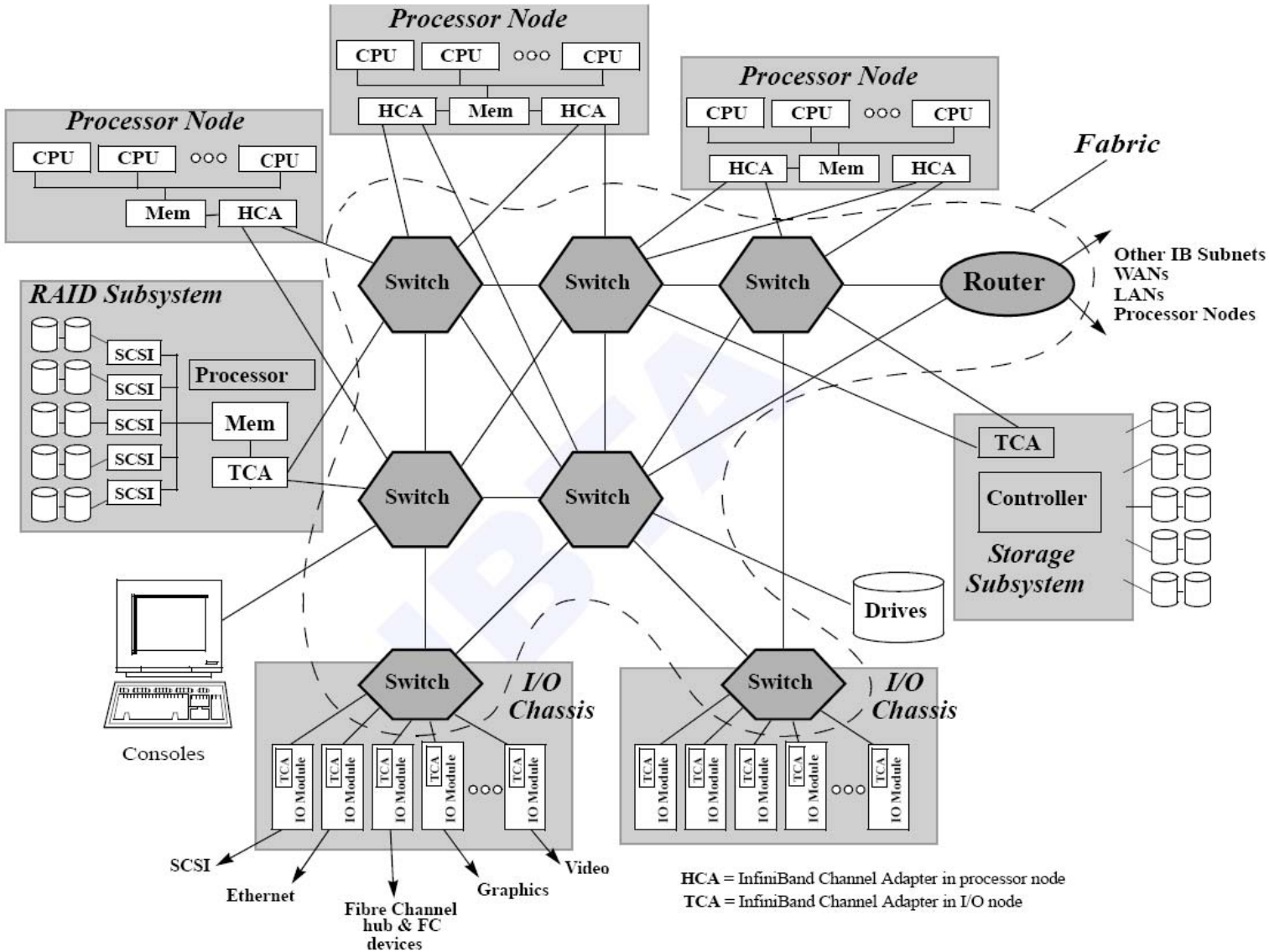
Florida State University

Tallahassee, Florida

# Introduction

- InfiniBand:
  - High Bandwidth/Low latency.
  - Widely adopted in HPC clusters:
    - Used in many of the Top 500 clusters. [Top 500 Supercomputing Site]

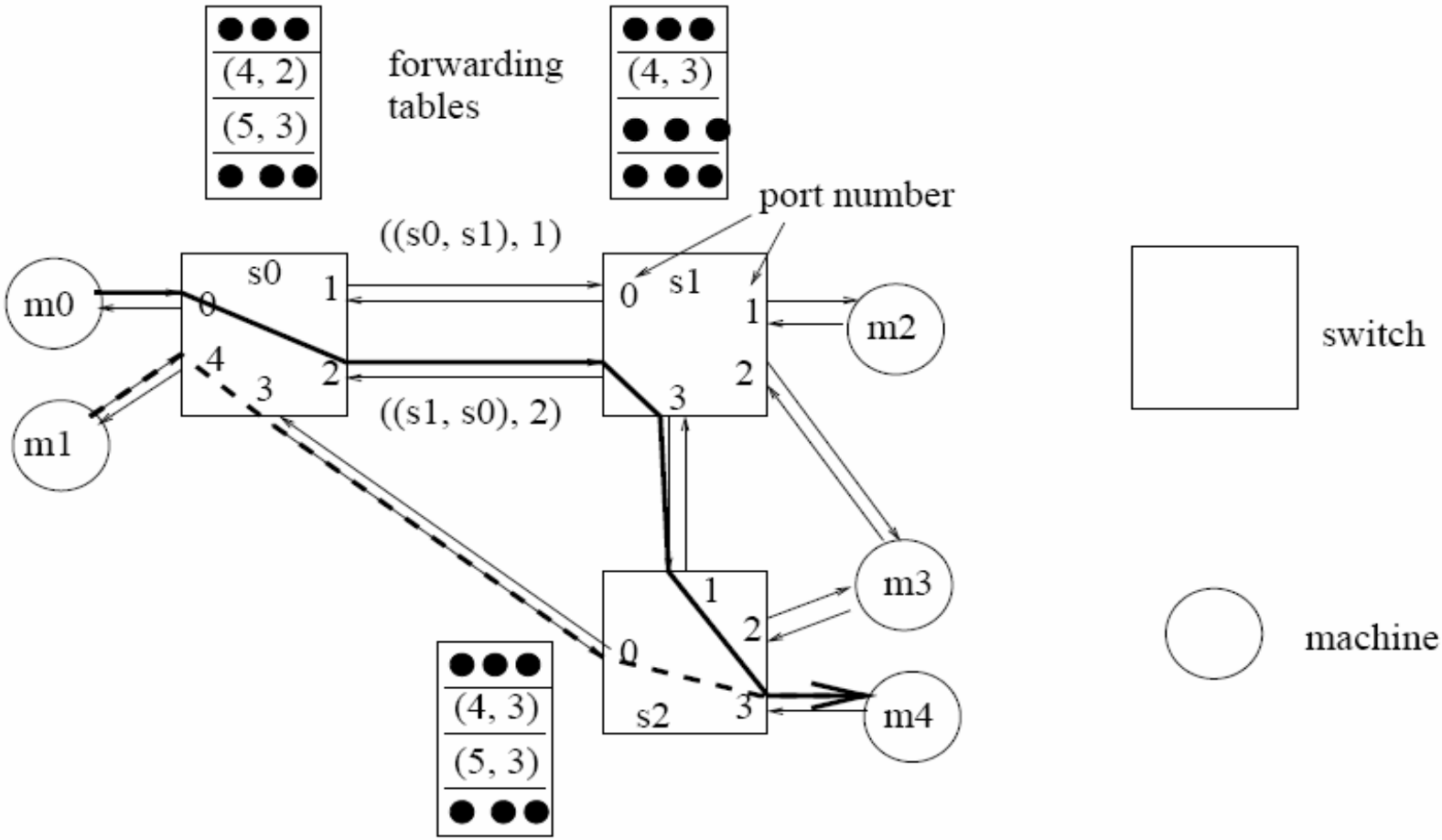
# InfiniBand Fabric



# InfiniBand Routing

- Subnet Manager:
  - Topology discovery, Routing, and network configuration.
- Destination Based routing.
  - Local Identifiers (LID) are used to identify end nodes.
  - Destination LID is used by the switch to determine how a packet would be forwarded.
    - Each Destination LID is mapped to one output port of the switch.

# Example




An InfinBand network Topology (LIDs 4 and 5 area assigned to m4).

- Routing in InfiniBand has two Components:
  - Path Computation.
  - LID Allocation and Assignment.
- InfiniBand provides a 16bit header.
  - Limits number of LID's to 64k.
  - Each node is limited to 128 LIDs.
    - Local Mask Control (LMC).
  - LIDs are a limited resource.
- Existing InfiniBand Routing:
  - Combine routing and LID assignment
    - It is unclear whether the routing has the best performance.
      - Load balancing property may not be the best.
    - It is unclear whether the LID assignment has the best performance.
      - We might be able to reduce the number of LIDs needed for the same routing.

# Our Proposal

- We propose to separate routing and LID assignment
  - Routing focuses on producing high quality paths
    - Many existing schemes can do this. (path selection, L-turn)
  - LID assignment focuses on minimizing LID usage.
    - Optimal LID assignment schemes have not been studied.
    - This is the focus of this work.

# Problem Statement

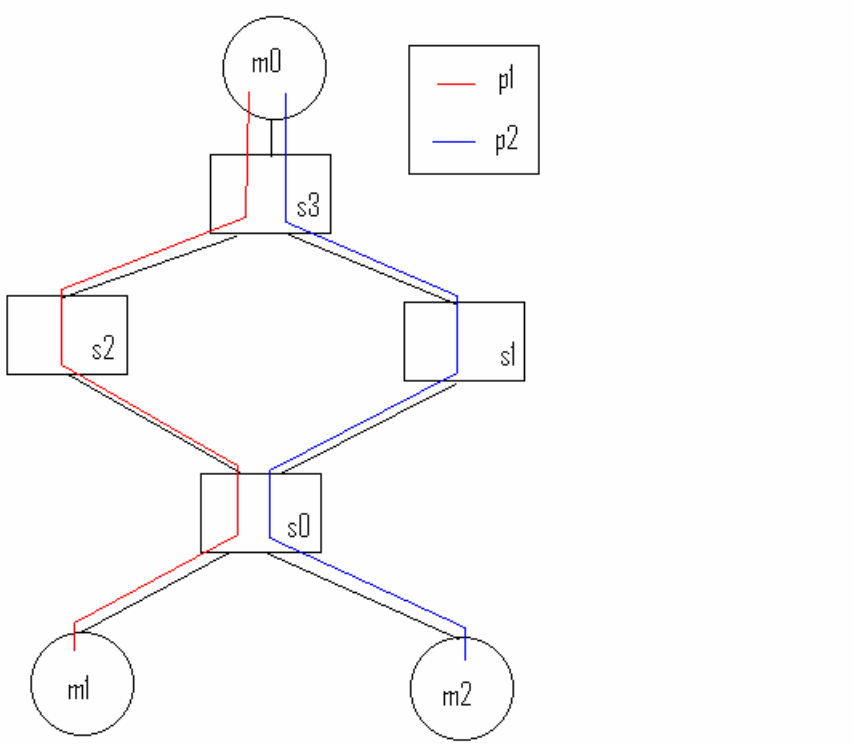
- LID allocation and assignment problem:
  - Given a routing how do we minimize the number of LIDs to realize the routing?
    - This paper shows that this problem is NP-Complete.
    - We propose three heuristics for this problem.
  - Is separating routing and LID assignment a good idea?

# Single destination LID assignment problem

- A routing (a set of paths) typically have multiple destinations.
  - We need to know how to route to each destination.
- Different destinations need to be assigned different LID ranges in InfiniBand.
  - Routing to different destinations is independent of one another.
  - A general LID assignment problem (for multiple destinations) can be reduced to a single destination LID assignment problem.
  - We will focus on the single destination problem.
    - All paths have the same destination.

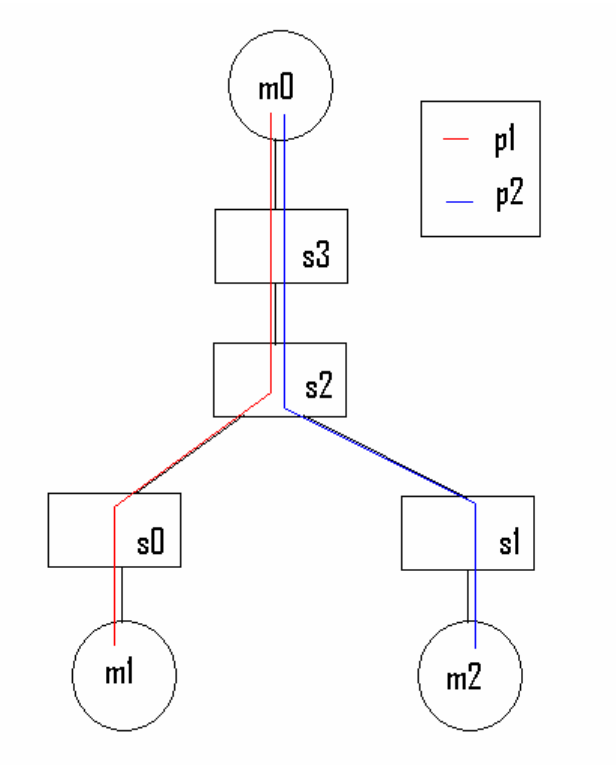
# Minimizing LIDs Used

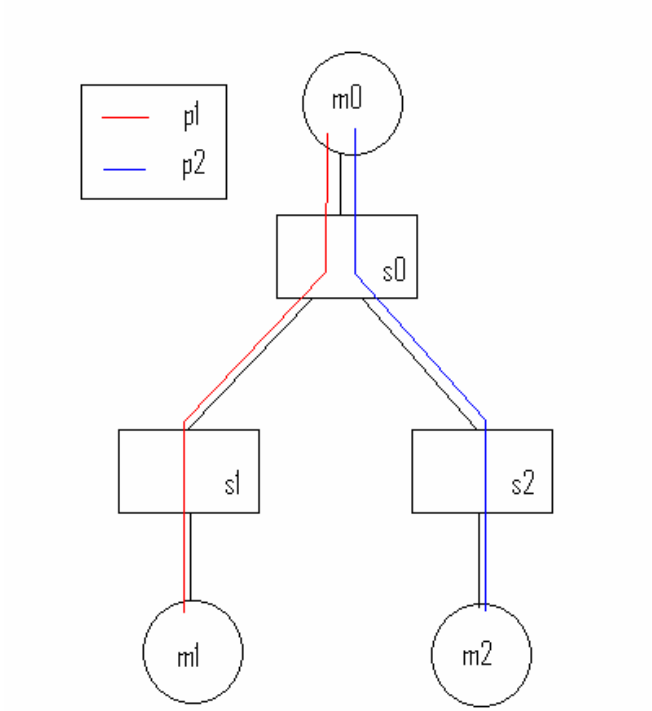
- Consider LID assignment for routes with the same destination.
- To minimize the number of LIDs paths have to share LIDs.
  - Some paths split and can not share LID's.
  - Different LIDs will have to be assigned to realize the routing.



# Minimizing LIDs Used

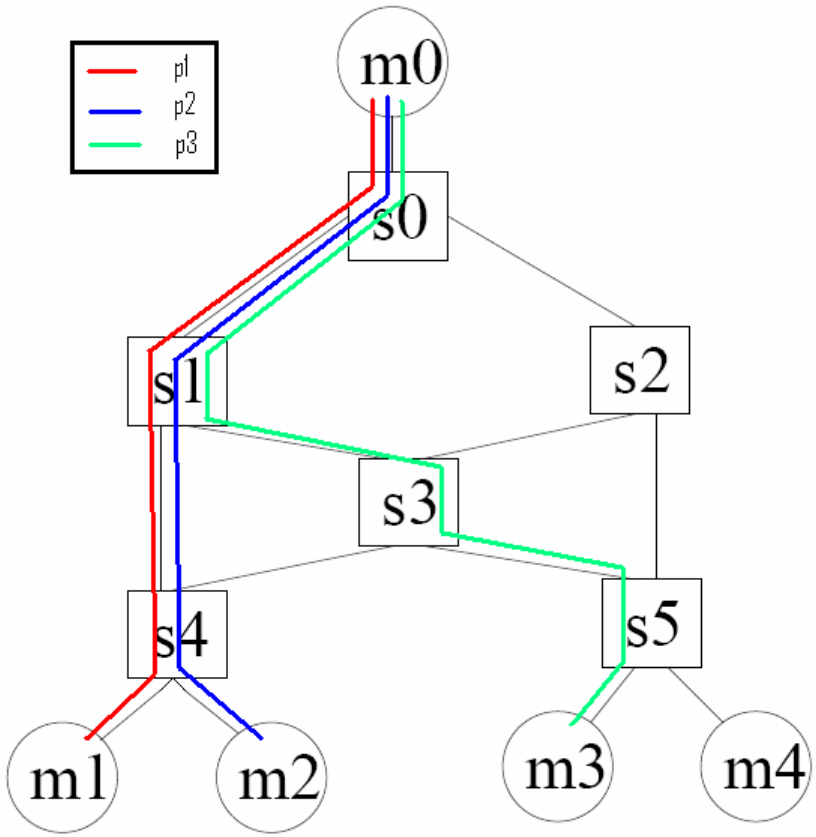
- Some paths can share LID's
  - Paths that never split can have the same LID in common.



# Configurations

- A set of paths that have no split.
- A configuration can be realized by one LID.
- Example set:
  - p1 = m1 → s4 → s1 → s0 →m0
  - p2 = m2 → s4 → s1 → s0 →m0
  - p3 = m3 → s5 → s3 → s1 → s0 →m0



# Formal Problem Definition

- If a set of paths can be separated into *k* configurations, then the set can be realized by k different LIDs.
- The LID assignment problem:
  - for a given routing (a set of paths), find the smallest k such that the routing can be separated into k configurations.

# The Problem

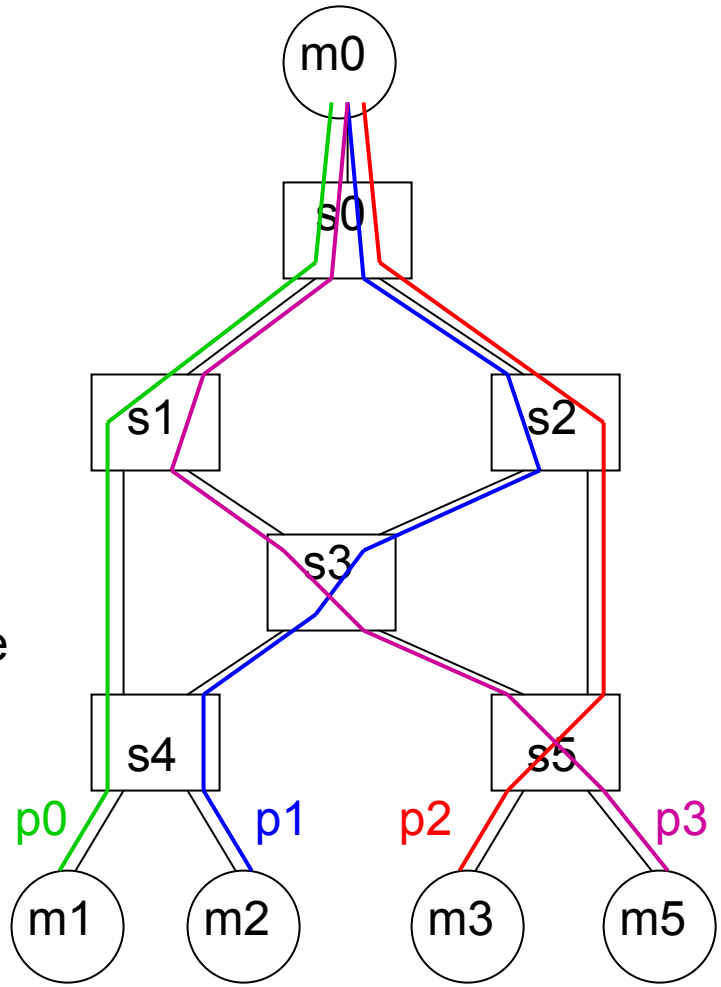
- This single destination LID assignment problem is NP-Complete.
  - Reduce graph coloring to this problem, if our problem can be solved in polynomial time then the graph coloring problem can be solved in polynomial time.
- We need heuristics for this problem.

# The Heuristics

- Our heuristics are based on the concept of minimal configuration set (MC):
  - A minimal configuration set for a set of paths is defined as:
    - Each element is a configuration.
    - Each path belongs to one element.
    - No two elements can be merged (and remain a configuration).
- Three Heuristics:
  - Greedy
  - Split-merge
  - Graph Coloring

# Greedy

- Iteratively pick paths.
  - Fit as many paths as possible into a configuration.
- Keep assembling configurations till all paths are assigned.
- Example paths:
  - p0 = m1 →s4 →s1 →s0 →m0
  - p1 = m2 →s4 →s3 →s2 →s0 →mo
  - p2 = m3 →s5 →s2 →s0 →m0
  - p3 =m3 →s5 →s3 → s1 →s0 →m0
- Using this heuristic the result would create the following configurations:
  - {p0,p2} , {p1}, {p3}
- Optimally this could be solved with two configurations:
  - {p0,p3},{p1,p2}
- Produces sub-optimal solutions

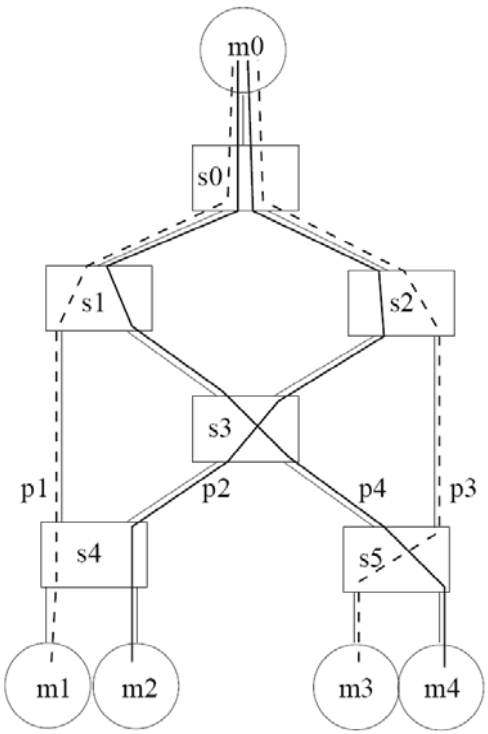


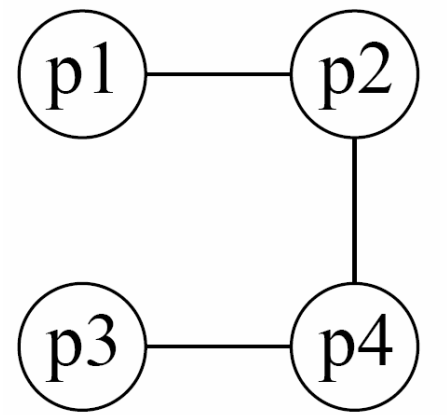
# Split-merge

- Greedy is a naive approach.
- Find a better starting point.
  - Initially create configurations based on splits at switches.
    - All paths are in the initial configuration.
    - Paths are separated into smaller sized configurations.
  - Merge configurations to create minimal configurations.
- Two versions are used
  - Split largest: sort switches from most split paths to least split paths.
  - Split smallest: sort switches from least split paths to most split paths.

# Graph Coloring

- Creating a split graph.
  - Each path is a node in the graph.
  - When paths $p_i$ and $p_j$ have a split an edge $e_{ij}$ exists in the graph.
  - The number of colors needed to color the split graph is equal to the number of LIDs needed to realize the routing.



# Graph Coloring

- Coloring heuristic:
  - Any graph coloring heuristic can be applied.
  - Our algorithm for coloring:
    1. Pick a node to color. (node to be placed in the configuration)
    2. Remove its adjacent nodes.
    3. If nodes remain in the graph, go to step 1 otherwise configuration is complete.
    - After a configuration is created, restore removed nodes and repeat till all nodes are used.
  - Picking a node to color:
    - Largest degree node first.
    - Smallest degree node first.

# Performance Study

- Evaluate the Performance of the LID assignment Heuristics.
  - Performance metrics: number of LIDs required for a given routing.
- Evaluate the Performance of Routing Schemes (path computation + LID Assignment).
  - Performance metrics: Load Balancing properties + Number of LIDs required.

# Performance of LID assignment Heuristics

- Topologies considered:
  - Random Irregular topologies: 16/32/64 switches with 64/128/256/512 nodes.
  - Nodal degree of 8.
  - Average of 32 random different topologies generated.
- Routing Considered:
  - Shortest Widest Routing.
  - Path Selection [Koibuchi et al, Parallel comput.,2005]
  - Our technique has no restrictions on routing, but routing affects the performance.

# Performance of LID assignment Heuristics

- Counting the LIDS needed.
  - Local Mask Control (LMC) requires the number of LIDs to be a power of two.
  - The heuristics returns the absolute number of LID required for a node.
    - The number counted is adjusted to fit to the smallest LMC.
      - For example: when LIDs required for a node is 5 it means the LMC = 3 and 8 LIDs are counted for that node.

| LMC | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| LIDs Available | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |

Table 5.2: LIDs available for each LMC.

# Performance of Heuristics (shortest widest case)

| Topologies (Nodes/switches) | greedy | s-m/S | s-m/L | color/S | color/L | |
|---|---|---|---|---|---|---|
| 128/16 | 478.7 | 478.9 | 477.3 | 479.3 | 476.4 | |
| 256/16 | 1044.3 | 1045.4 | 1041.5 | 1047.7 | 1039.2 | |
| 512/16 | 2218.3 | 2220.1 | 2211.8 | 2220.4 | 2208.5 | |
| 128/32 | 451.5 | 453.9 | 452.9 | 461.3 | 443 | |
| 256/32 | 1078.8 | 1084.7 | 1079 | 1100 | 1062.4 | |
| 512/32 | 2428.7 | 2440.2 | 2425.8 | 2461 | 2392.1 | |
| 128/64 | 422.8 | 427.7 | 427 | 441.5 | 407.4 | 8.4% |
| 256/64 | 1015.5 | 1022.2 | 1019.3 | 1044.6 | 990.6 | 5.5% |
| 512/64 | 2325.8 | 2338.4 | 2330.1 | 2385.1 | 2274.4 | 4.9% |

The Average of the total number of LIDs allocated (shortest widest)

# Performance of the Heuristics (Path Selection)

| Topologies (Nodes/switches) | greedy | s-m/S | s-m/L | color/S | color/L | |
|---|---|---|---|---|---|---|
| 128/16 | 520.9 | 524.2 | 514 | 581.2 | 466 | |
| 256/16 | 951.3 | 952.7 | 935 | 1062.6 | 851.2 | |
| 512/16 | 1829.2 | 1852.8 | 1823 | 2038.7 | 1653.2 | |
| 128/32 | 540.3 | 546.7 | 539.3 | 611.3 | 466 | |
| 256/32 | 1006.7 | 1018.2 | 1002.2 | 1130.8 | 887.2 | |
| 512/32 | 1904 | 1920.3 | 1895.7 | 2115.8 | 1688.7 | |
| 128/64 | 528 | 541.1 | 530.5 | 599.4 | 460.5 | 31% |
| 256/64 | 1054.9 | 1092.9 | 1068.1 | 1197.9 | 921.4 | 30% |
| 512/64 | 2019.9 | 2075.4 | 2043.4 | 2278.6 | 1786.6 | 27.5% |

The Average of the total number of LIDs allocated (path selection)

# Performance of the Routing Schemes

- Fully Explicit routing.
  - Modifies the routing such that only one LID is needed per destination.
  - Load balancing is sacrificed to simplify LID assignment.
- Destination Renaming.
  - Uses the same LID to assign to a path till it finds a conflicts.
  - Renames the LID and updates the routing tables with the new LID.
- Separate: Path Selection with graph-color/L.
  - Our best performing algorithm.
- Performance metrics:
  - (1) LIDs required for each routing algorithm.
  - (2) Load Balancing property: Maximum Link Load
    - Traffic between all nodes are the same.
      - The traffic volume between each pair of nodes is normalized to 1.

# Results

| Topologies (Node/switch) | Fully Explicit | | Renaming | | Separate | |
|---|---|---|---|---|---|---|
| | load | LIDs | load | LIDs | load | LIDs |
| 128/16 | 4.34 | 128 | 3.84 | 477.8 | 3.7 | 466 |
| 256/16 | 8.65 | 256 | 7.52 | 1044.9 | 7.35 | 851.2 |
| 512/32 | 14.71 | 512 | 13.24 | 2422.8 | 12.37 | 1688.7 |
| 512/64 | 11.36 | 512 | 10.55 | 2323.4 | 9.54 | 1786.6 |

- Fully explicit Routing used the least number of LIDs
  - Worse load balancing.
    - A Small 128/16 network is 17% worse than Separate
    - A Large 512/64 network is 19% worse than Separate
- With Separate and Destination Renaming:
  - Separate achieves less LIDs used.
  - Separate has better load balancing
    - Path Selection Routing has diverse paths.
  - We see a 10.6% better load balancing and 25.4% less LIDs (512/64).
- For the least number of LIDs Fully Explicit Routing is the best.
- For better Load Balancing, a separate scheme is more efficient.

# Conclusion

- Proposed to separate LID allocation and assignment.
  - LID allocation and assignment problem is NP-Complete.
  - Developed three heuristics:
    - Greedy
    - Split-merge
    - Graph Coloring
- LID assignment and routing can be separated.
  - Different routing schemes impact LID assignment.
  - Good routing schemes give good load balancing.
  - Different heuristics reduce the number LIDs used.
- LID assignment can be used on any InfiniBand routing scheme finding deadlock-free, deterministic paths.

Questions?