Dragonfly Interconnect Topology

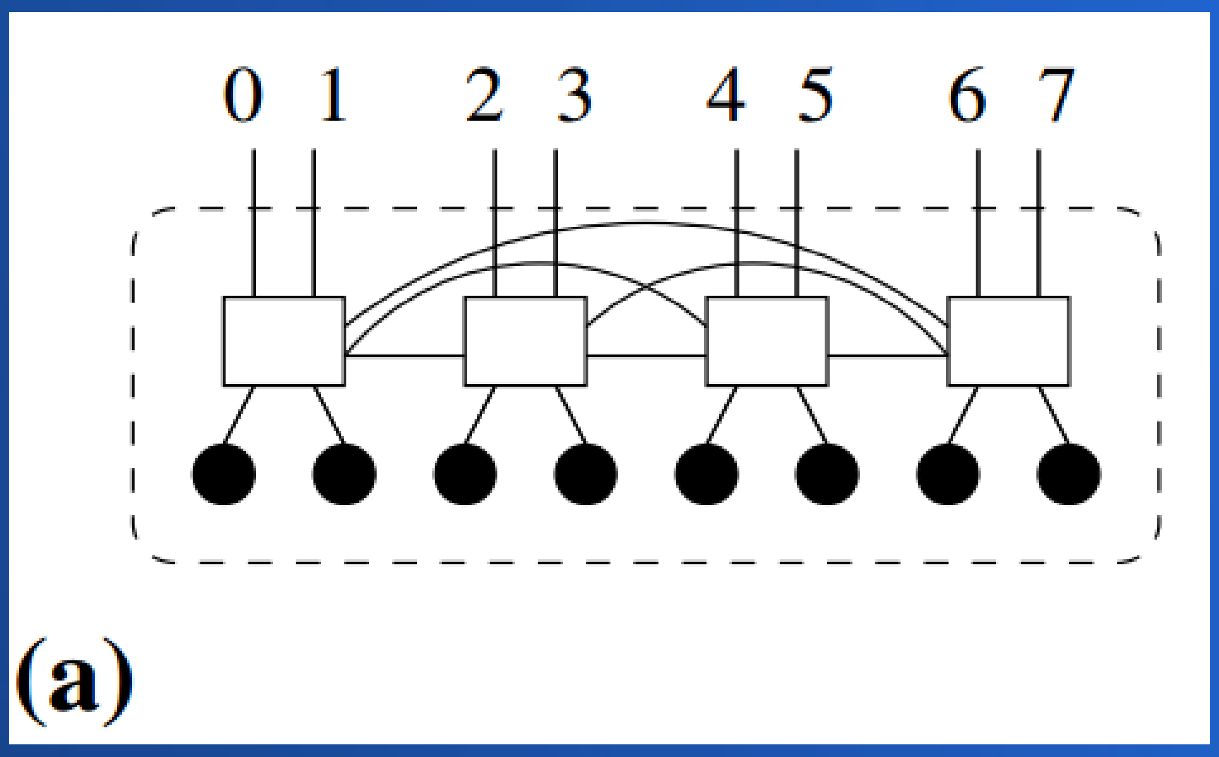
Emily Hastings — Anda Xu

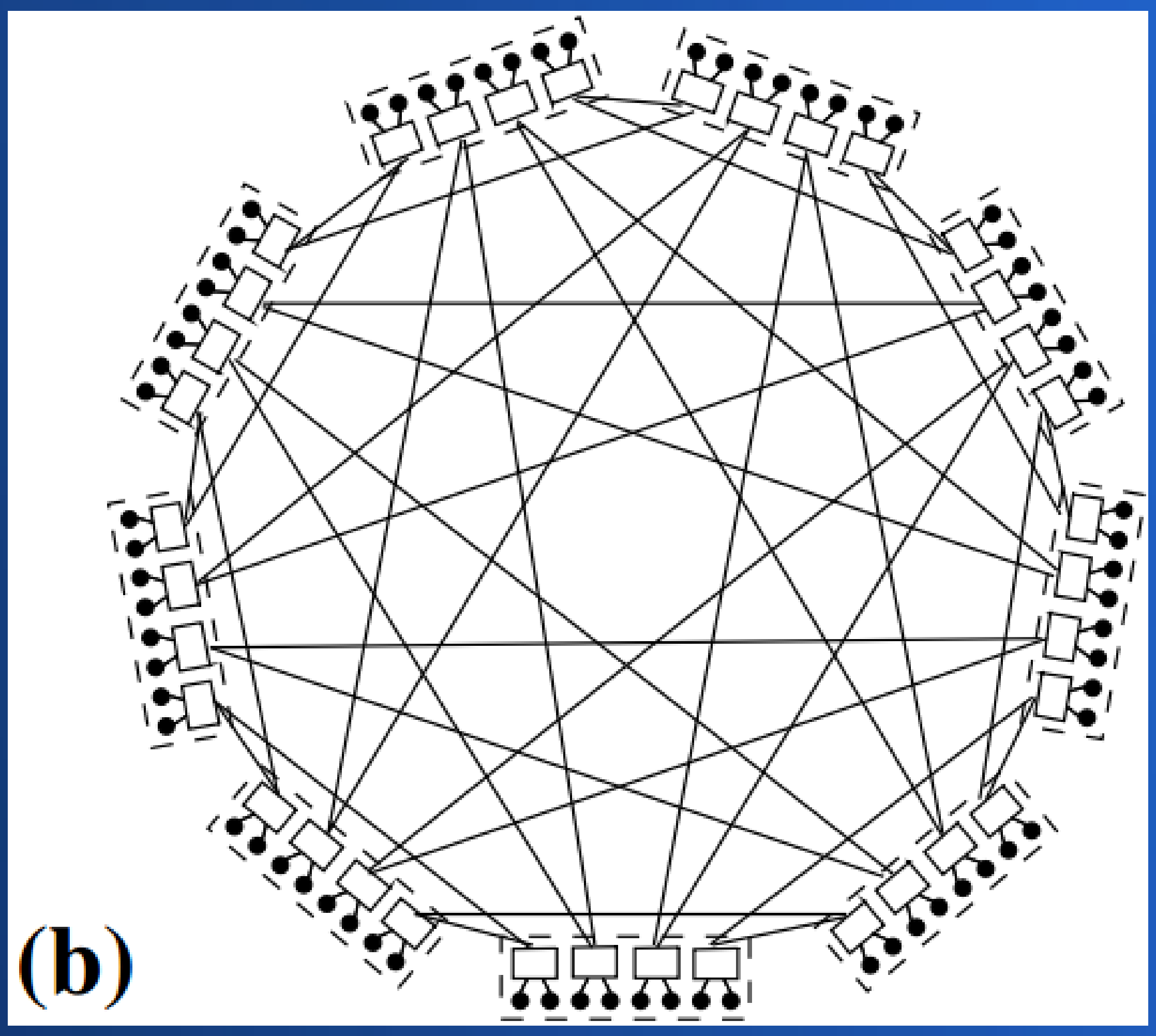
What Is Dragonfly?

Dragonfly is an interconnect topology for supercomputers introduced in 2008 by scientists at Stanford University and Cray, Inc.

In a Dragonfly machine, nodes (processors) are connected to routers, and routers are connected in groups of equal size. Within each group, all routers are connected to each other, and as such can act as a single "virtual router" that is much larger than its components (a). These virtual routers are then connected to each other, with one edge between each pair of groups (b).

With this configuration, direct communication between any pair of nodes can occur in no more than three hops (locally to router connected to destination group, globally across inter-group edge, and locally within destination group).



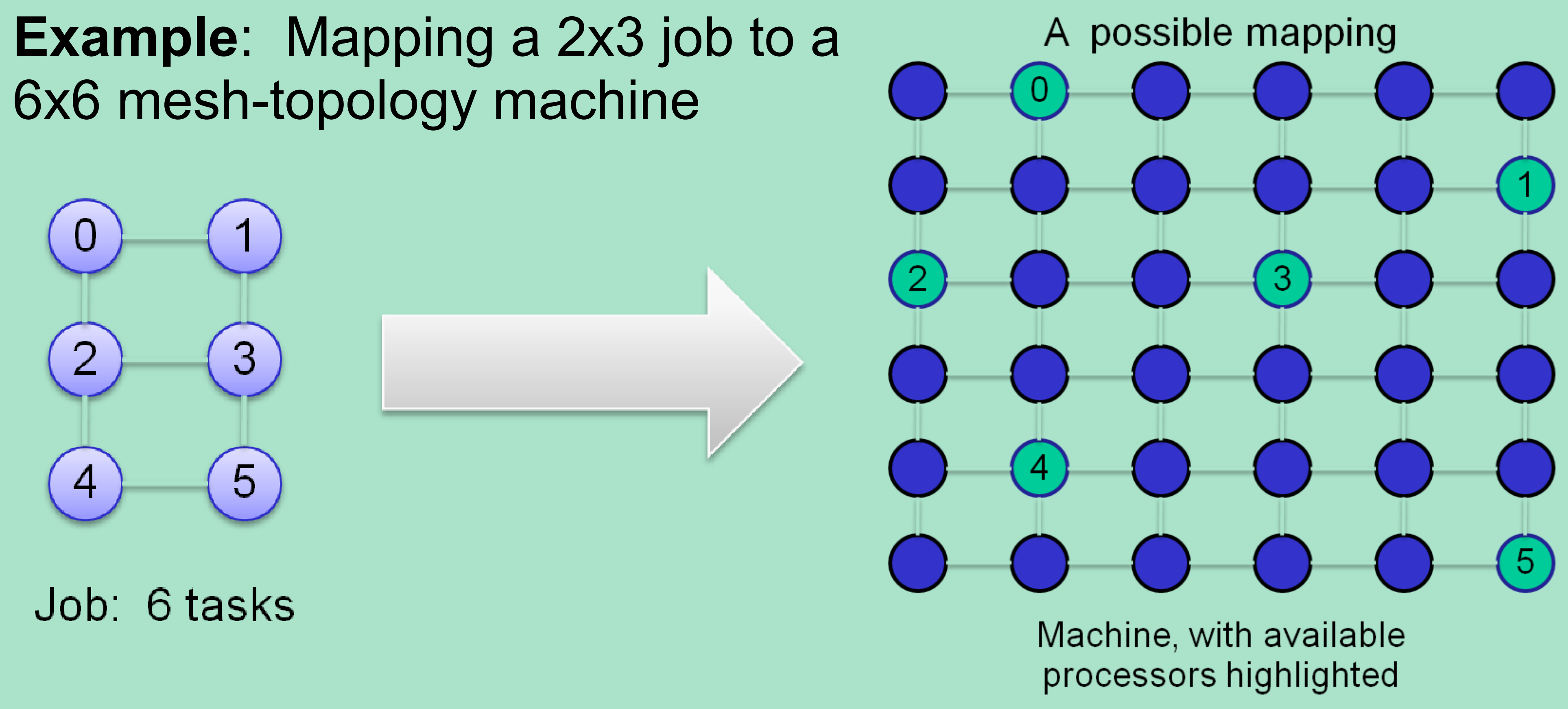


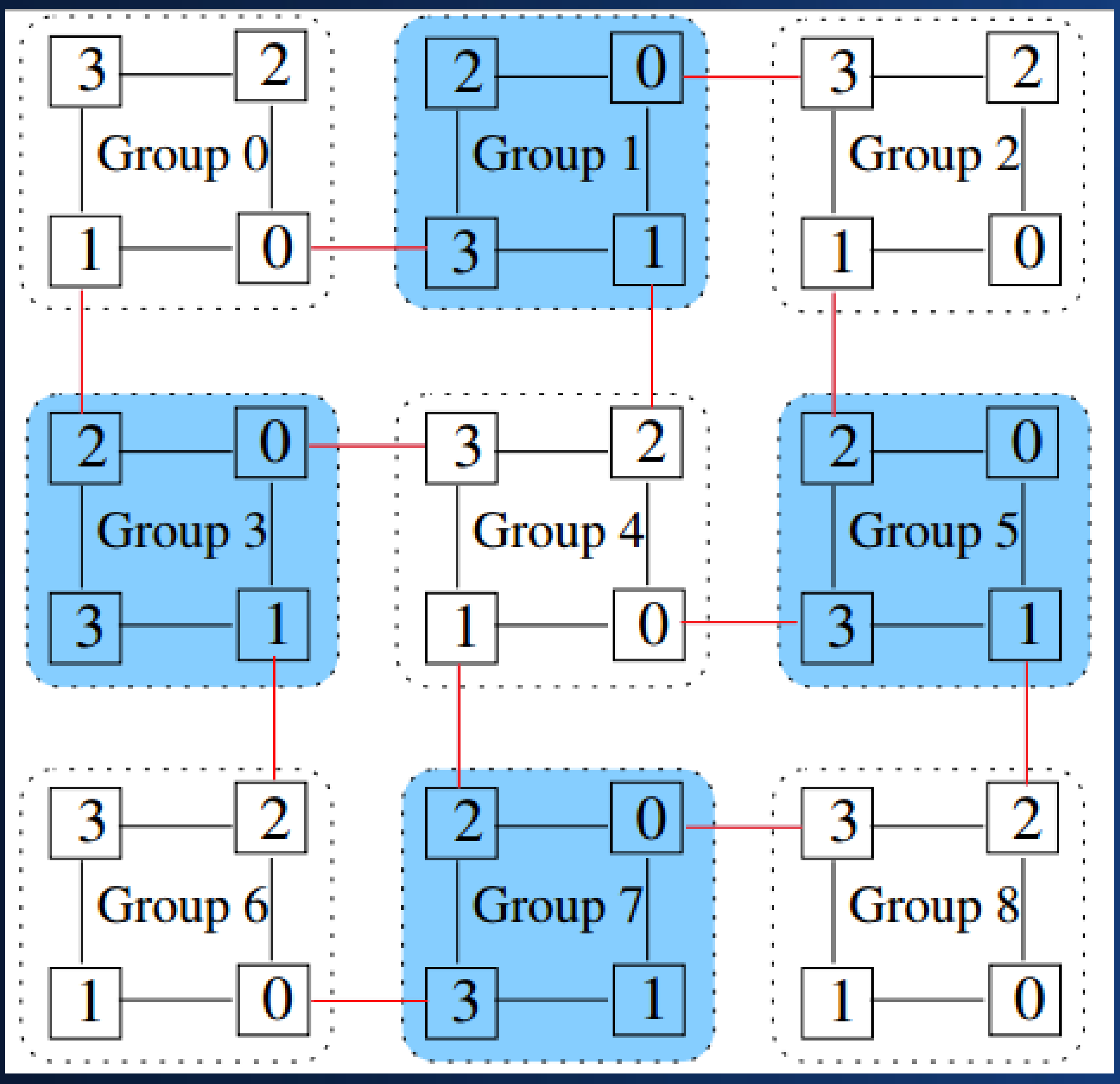
Advantages Over Other Topologies:

- Reduced latency (time to send a message) and hop counts
- Scalable
- Design minimizes necessary global channels, reducing cost significantly

Task Mapping

Example: Mapping a 2x3 job to a 6x6 mesh-topology machine





6x6 job, divided into 9 2x2 sub-jobs that correspond with Dragonfly groups. Tasks are numbered according to the Dragonfly router they have been mapped to, and inter-subgraph connections served by direct global connections on the machine are shown in red.

Task mapping is the process of assigning the tasks of a given job to available processors in the machine. The edges in the job graph represent tasks that need to communicate with each other, so in order to task-map efficiently, these nodes need to be close together on the machine. On Dragonfly, all nodes are close together, but mappings can be optimized so that many of these communication paths need only 1 hop on the machine.

Part of our research dealt with trying to find a general pattern for such mappings, by dividing jobs into sub-jobs that were the same size as the Dragonfly groups and assigning each router a task. This way, all edges within a sub-job are served by local connections, and many of the inter-group edges use direct global connections. For a relative-cabled Dragonfly as shown above, a 6x6 job can be mapped optimally using a checkerboard pattern of two router arrangements. We found that for, larger machines and jobs, this pattern can be generalized to a single repeated arrangement.

In the future, we hope to investigate applying this generalization to different kinds of jobs, including three-dimensional meshes.

Dragonfly Interconnect Topology

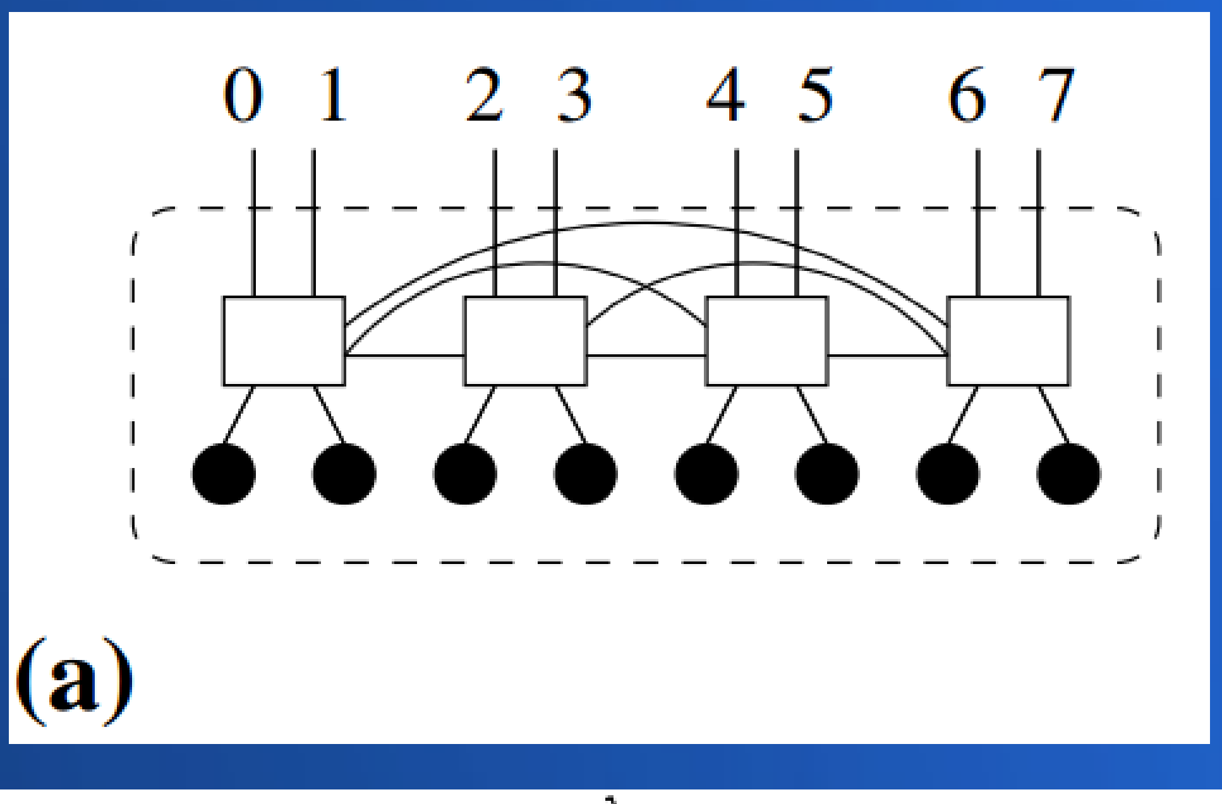
Emily Hastings — Anda Xu

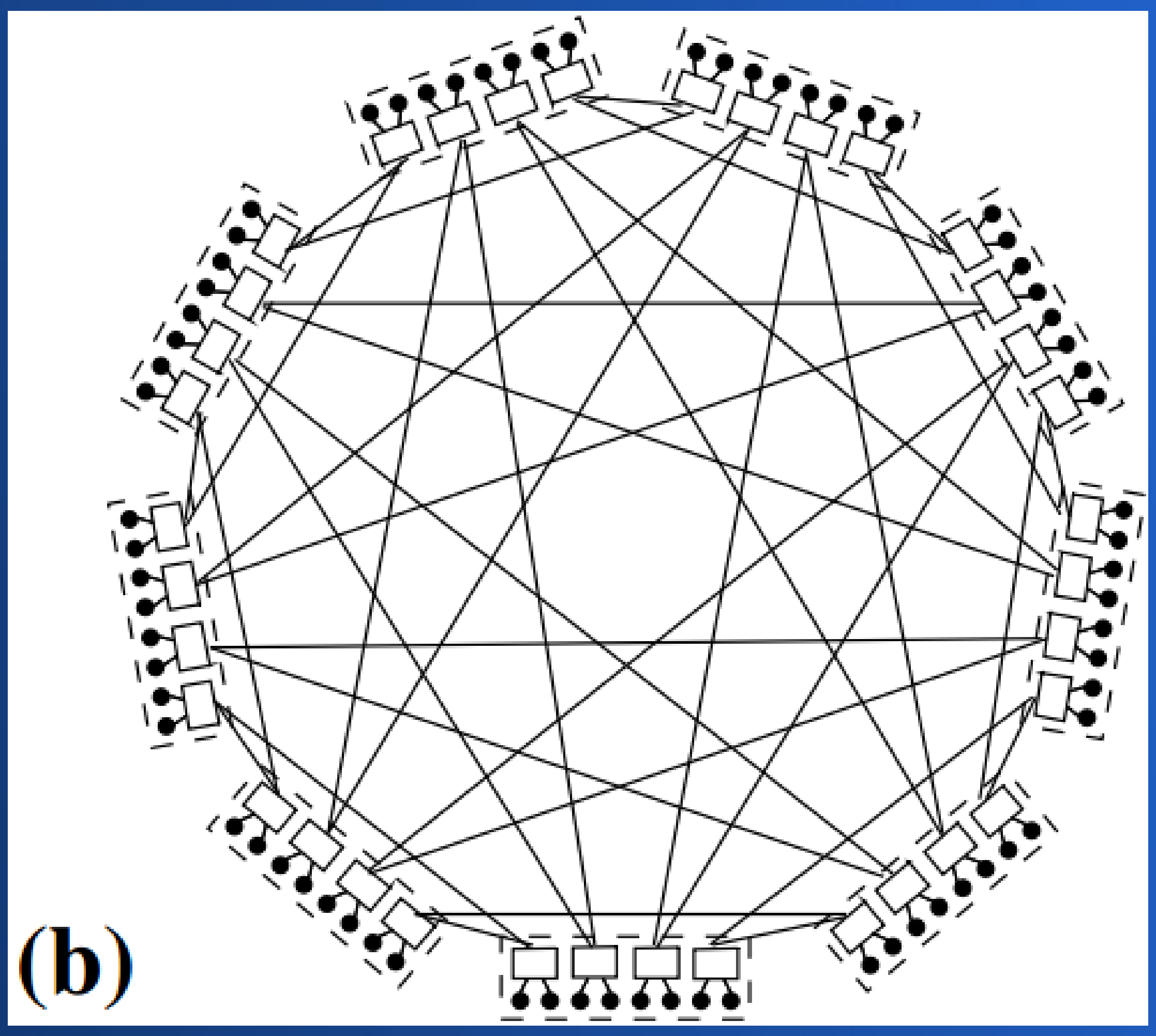
What Is Dragonfly?

Dragonfly is an interconnect topology for supercomputers introduced in 2008 by scientists at Stanford University and Cray, Inc.

In a Dragonfly machine, nodes (processors) are connected to routers, and routers are connected in groups of equal size. Within each group, all routers are connected to each other, and as such can act as a single "virtual router" that is much larger than its components (a). These virtual routers are then connected to each other, with one edge between each pair of groups (b).

With this configuration, direct communication between any pair of nodes can occur in no more than three hops (locally to router connected to destination group, globally across inter-group edge, and locally within destination group).

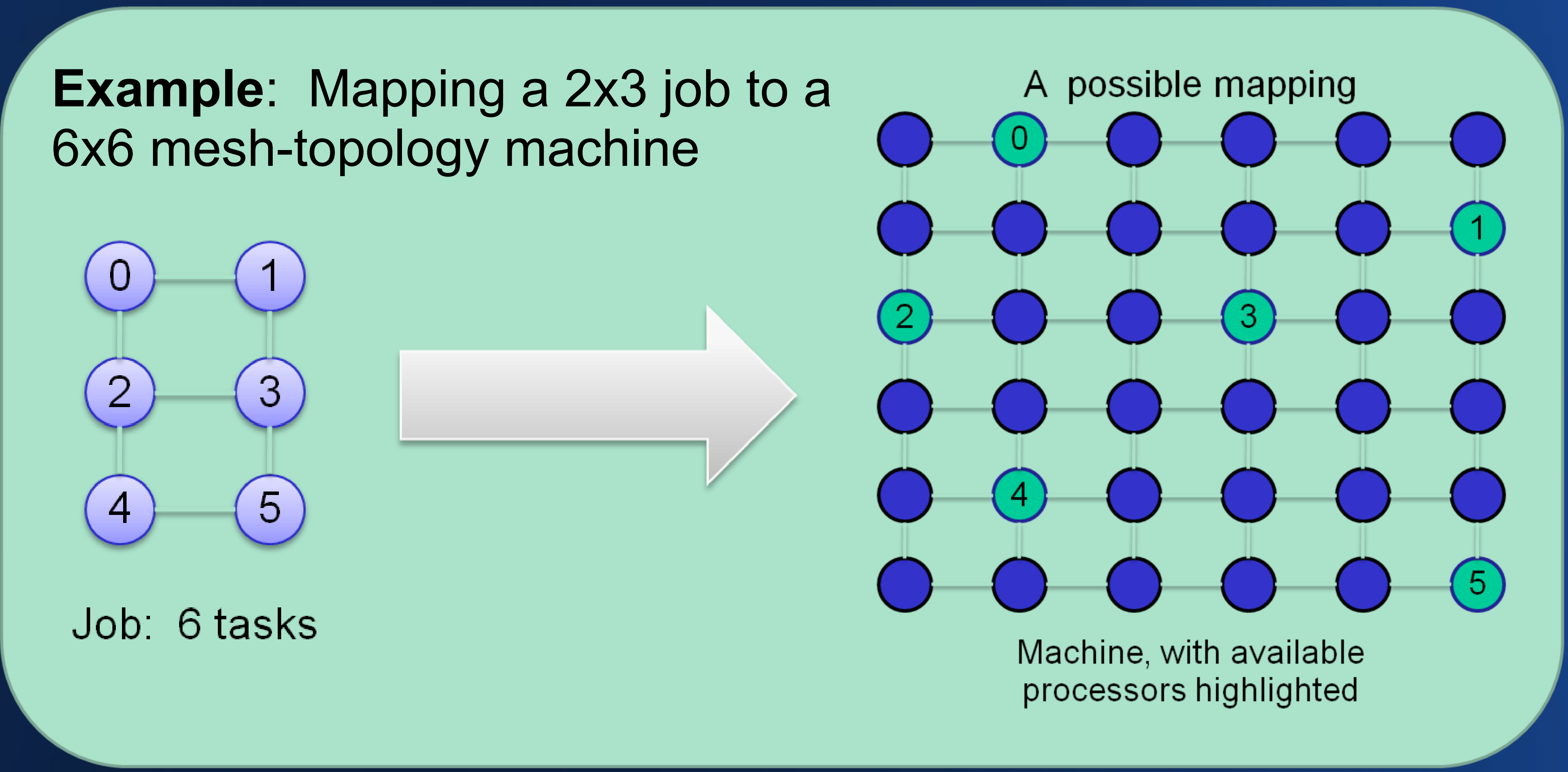


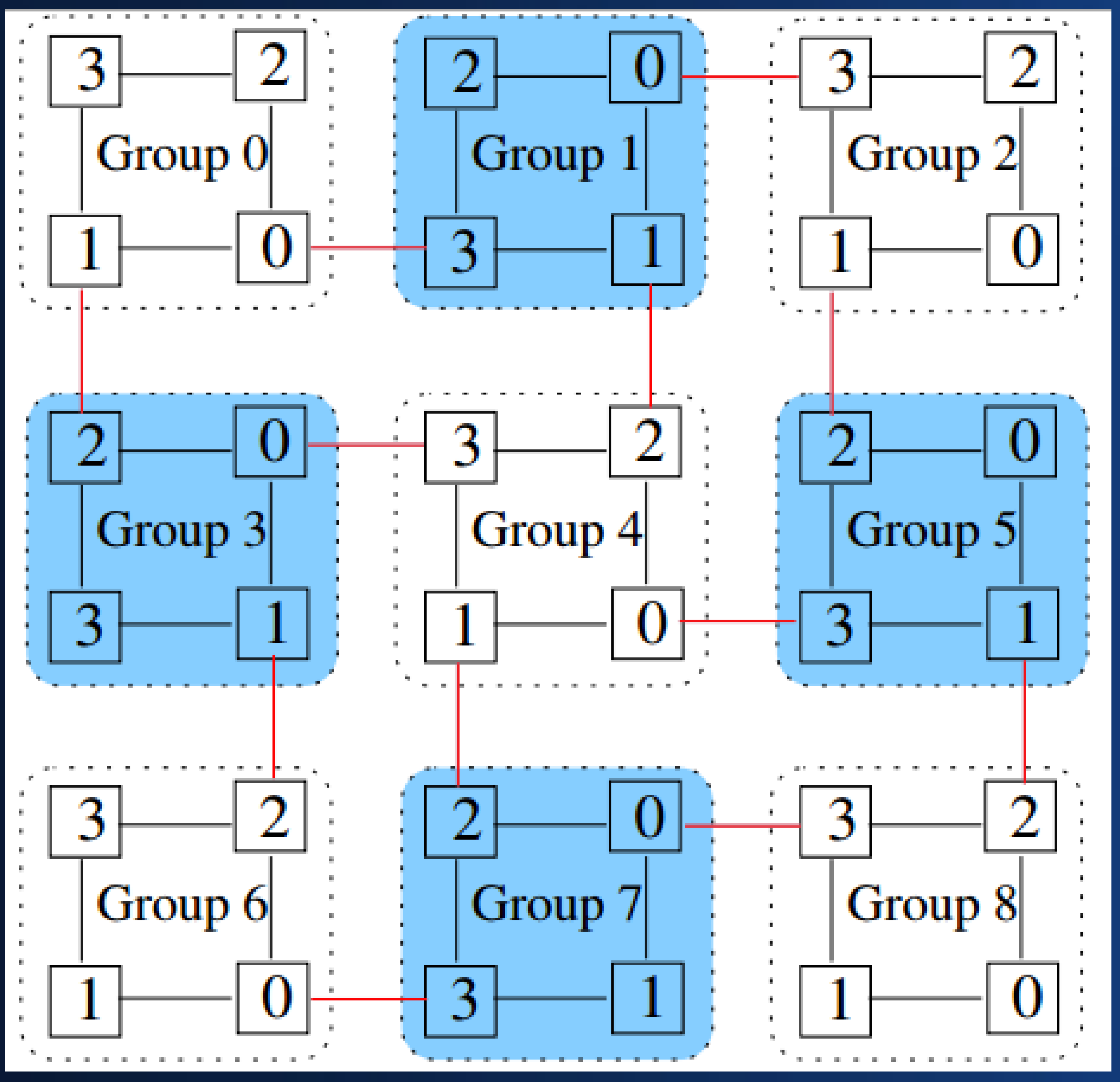


Advantages Over Other Topologies:

- Reduced latency (time to send a message) and hop counts
- Scalable
- Design minimizes necessary global channels, reducing cost significantly

Task Mapping


Example: Mapping a 2x3 job to a 6x6 mesh-topology machine
0
1
2
3
4
5
Job: 6 tasks
A possible mapping
Machine, with available processors highlighted



6x6 job, divided into 9 2x2 sub-jobs that correspond with Dragonfly groups. Tasks are numbered according to the Dragonfly router they have been mapped to, and inter-subgraph connections served by direct global connections on the machine are shown in red.

Task mapping is the process of assigning the tasks of a given job to available processors in the machine. The edges in the job graph represent tasks that need to communicate with each other, so in order to task-map efficiently, these nodes need to be close together on the machine. On Dragonfly, all nodes are close together, but mappings can be optimized so that many of these communication paths need only 1 hop on the machine.

Part of our research dealt with trying to find a general pattern for such mappings, by dividing jobs into sub-jobs that were the same size as the Dragonfly groups and assigning each router a task. This way, all edges within a sub-job are served by local connections, and many of the inter-group edges use direct global connections. For a relative-cabled Dragonfly as shown above, a 6x6 job can be mapped optimally using a checkerboard pattern of two router arrangements. We found that for, larger machines and jobs, this pattern can be generalized to a single repeated arrangement.

In the future, we hope to investigate applying this generalization to different kinds of jobs, including three-dimensional meshes.