Creating a Robust Desktop Grid using Peer-to-Peer Services

CMSC 818S: Prof. Alan Sussman

Presented by Jik-Soo Kim

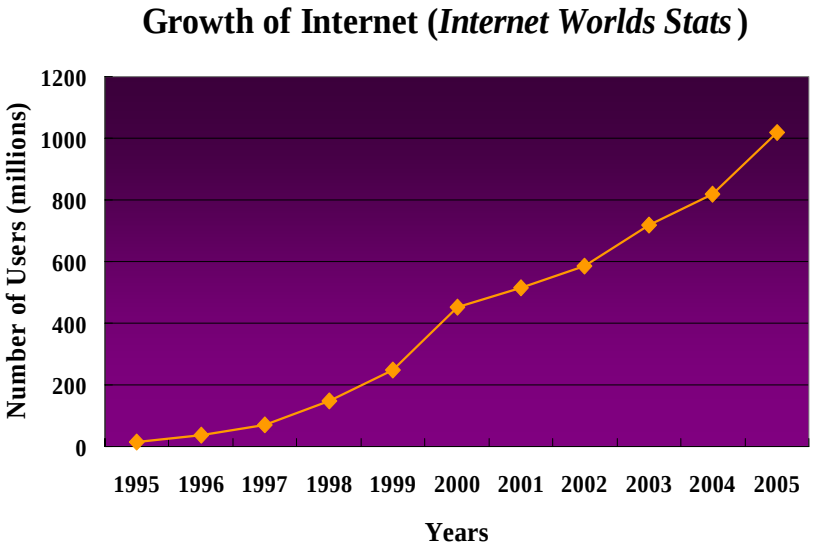
Department of Computer Science

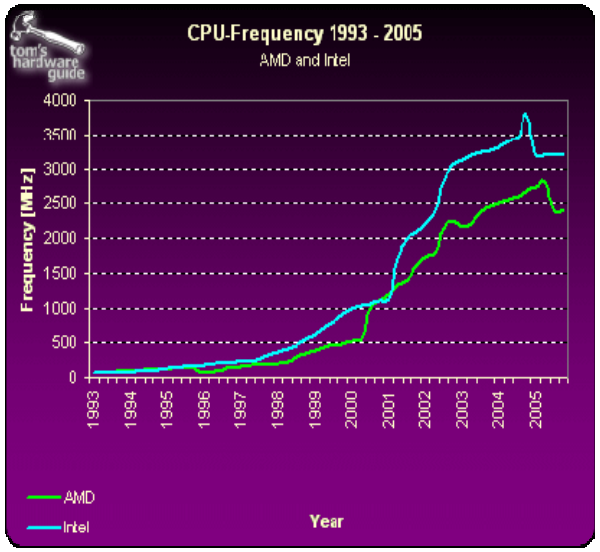
UNIVERSITY OF MARYLAND

Desktop Grid Computing

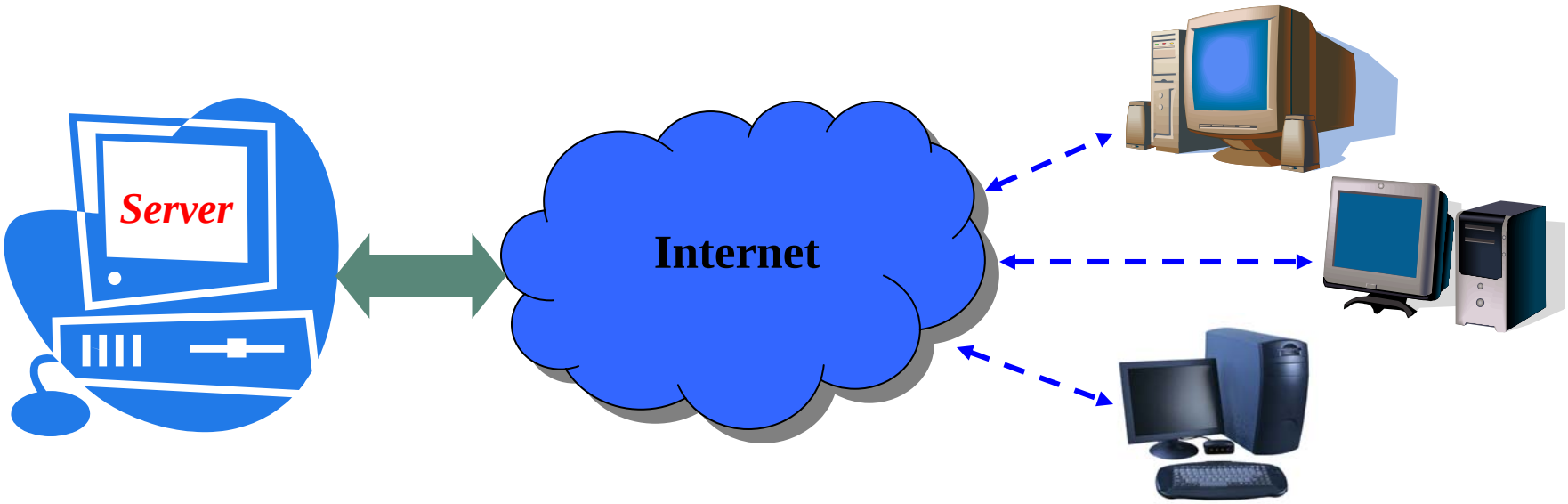
to Compute !

SETI HOME

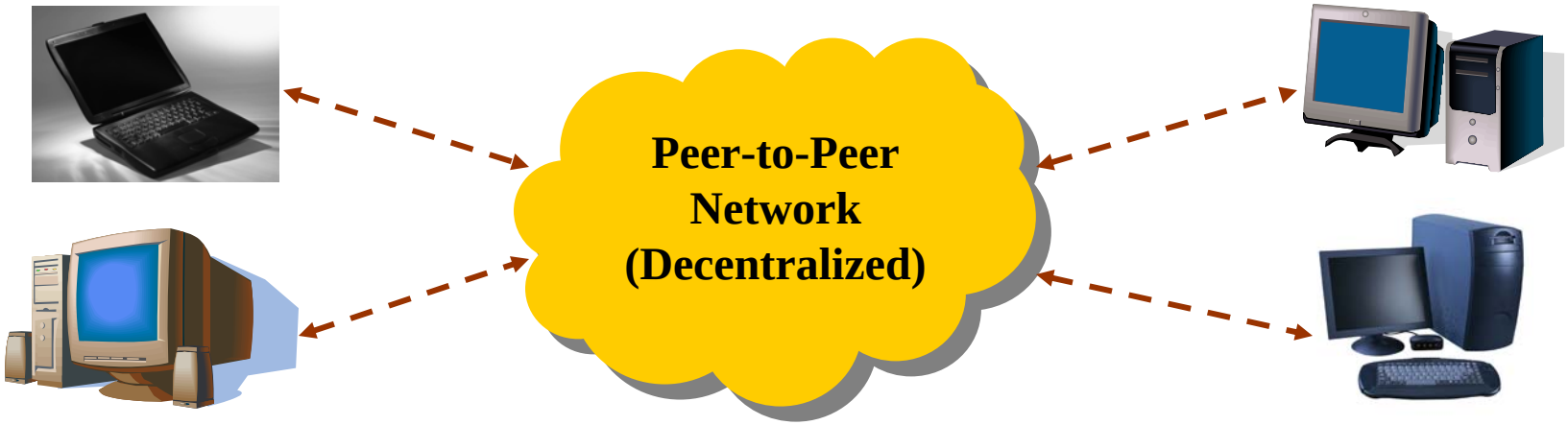




Confluence of P2P and Grid



Robustness, Reliability and Scalability?



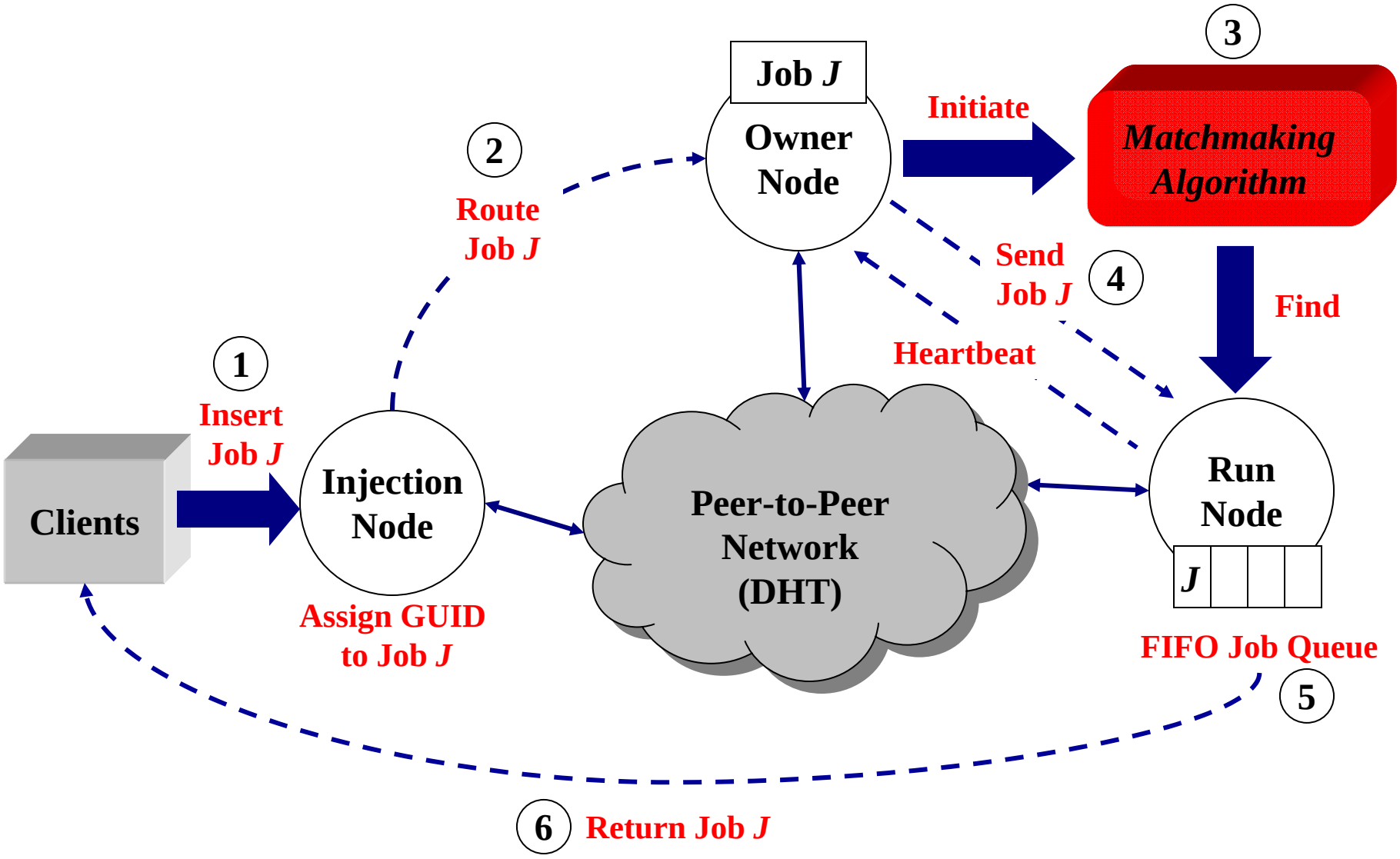
P2P Desktop Grid Computing System

- Executing Grid applications on a widely distributed set of resources
- *Decentralized*, *Robust*, *Highly available* and *Scalable*
- Applications
  - Large computational requirements and relatively low I/O requirements
  - Physical simulations and data analysis

Hard Problems / Issues

- Submitting jobs
- Matchmaking
- Load balancing
- Secure job execution
- Resilience to failure

System Architecture



Roadmap

- Basic Assumptions and Overall Goals
- Rendezvous Node Tree
- Content-Addressable Network
- Experimental Results from GRID 2006
- Conclusion

Workload Assumptions

- *Must* accommodate heterogeneous clusters of nodes running heterogeneous batches of jobs
- *Clustering* in nodes (resource capabilities) and jobs (requirements)
  - A small number of equivalent classes of nodes
  - Parameter sweeps, e.g., N-body or weather simulations

Nodes \ Jobs	**Clustered**	**Mixed**
Clustered		Condor
Mixed	BOINC/ SETI@Home	

Goals of Matchmaking Algorithms

- *Low overhead*
  - Routing must not add significant overhead
- *Completeness*
  - A valid assignment of a job to a node must be found if such an assignment exists
  - TTL-based mechanisms are not feasible
- *Precision*
  - Resources should not be wasted
- *Load balance*
  - Distribute load across multiple candidates

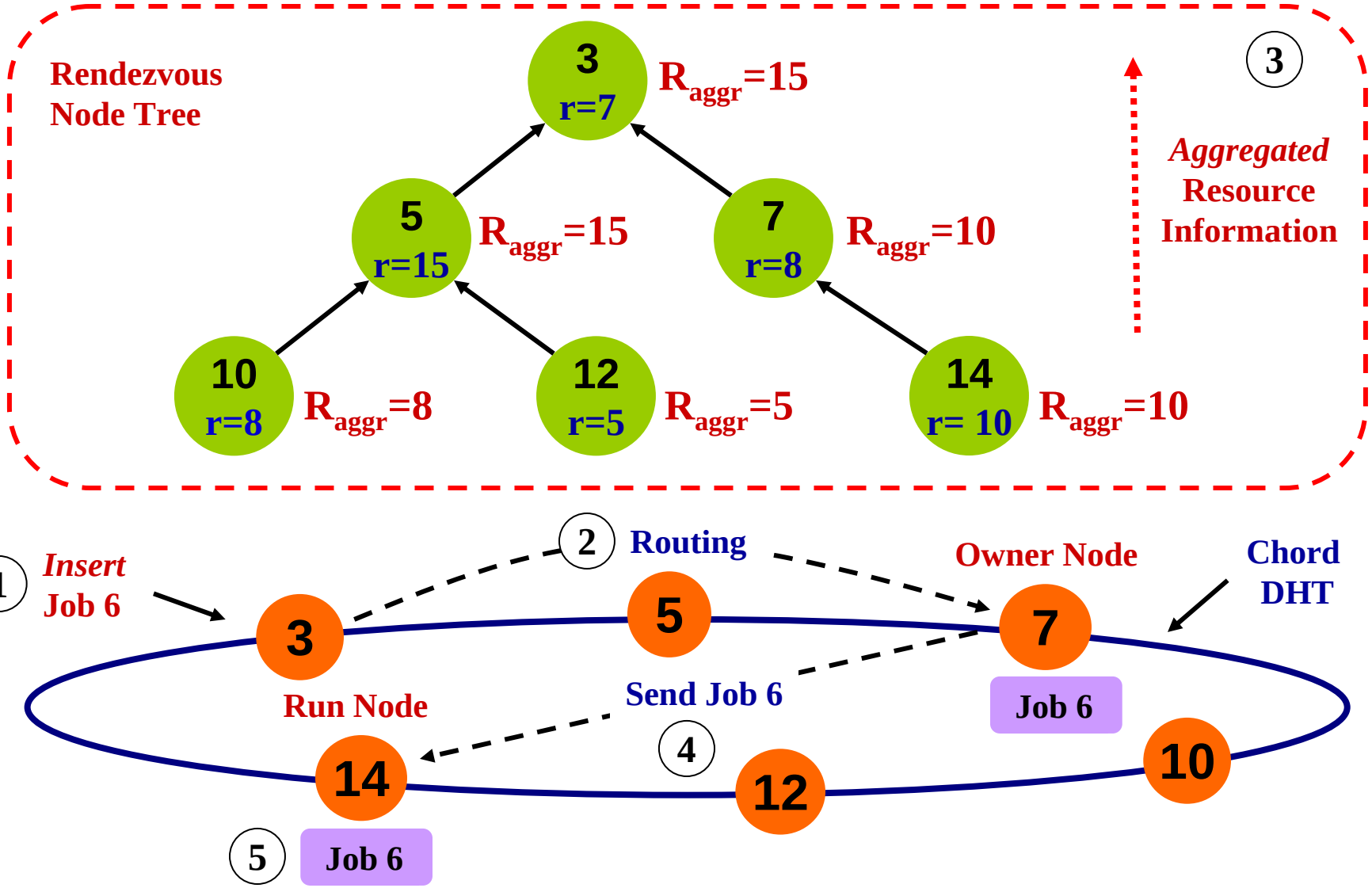
Basic Assumptions

- Underlying *Distributed Hash Table* (DHT)
  - Object location and routing in a P2P network
  - Reformulate the problem of matchmaking to one of routing
- *Job* in the system
  - Data and associated profile
  - All jobs are *independent*
- Optimization criterion
  - Minimize time to complete all jobs (combination of throughput and response time)

Rendezvous Node Tree (RNT)

- *Implicit* tree built on top of P2P network
  - 1-1 mapping from DHT (*Chord*) nodes to RNT nodes
- Why use a tree?
  - Need to *aggregate* current resource informatio n to perform matchmaking
  - Aggregated Resource Information
    - *Maximal* amount of each resource available at some node in the subtree rooted at a node

Rendezvous Node Tree (RNT)



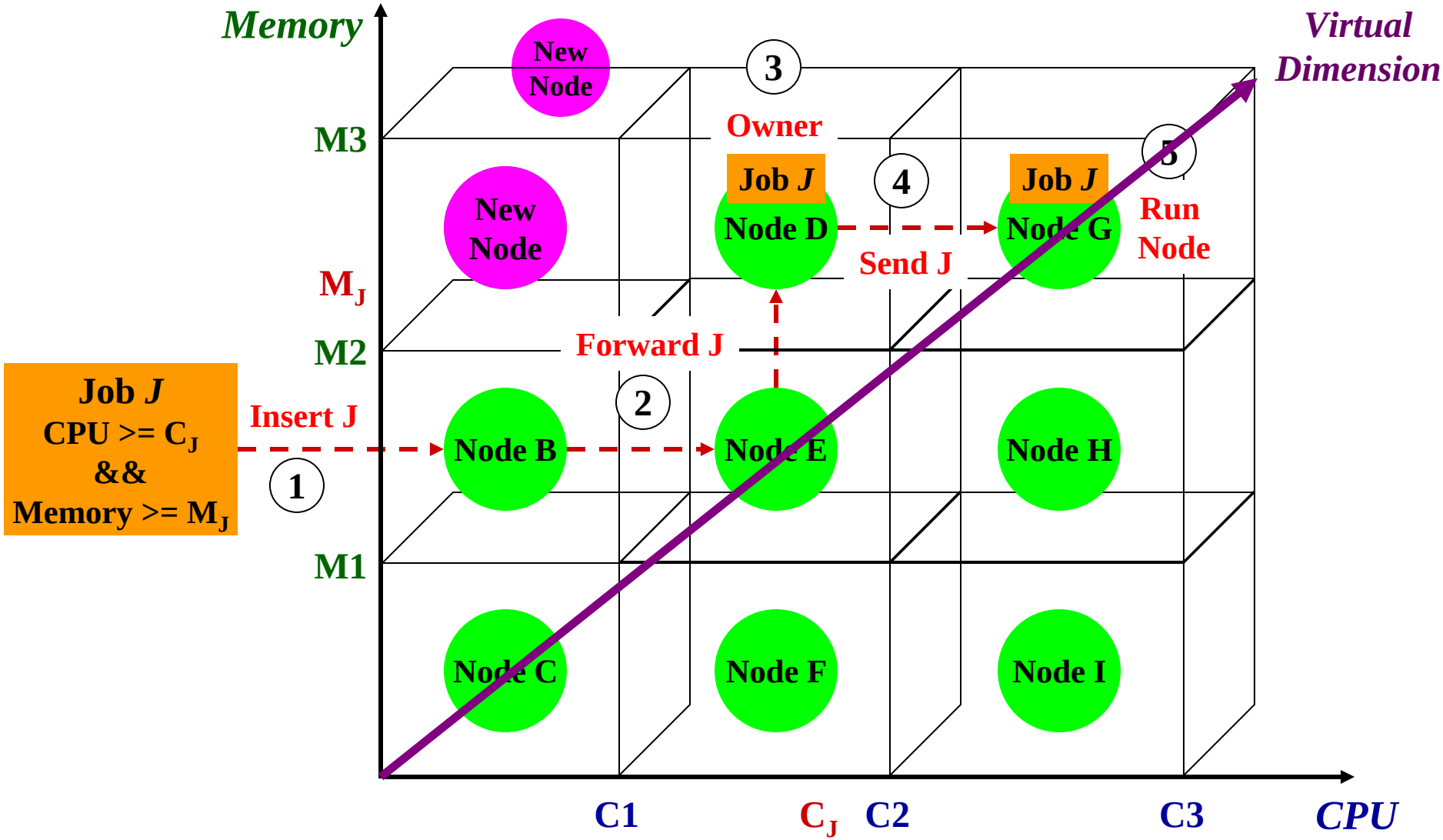
Modified Content-Addressable Network

- Basic CAN
  - Logical *d*-dimensional space
    - zone, neighbors, greedy forwarding
- *Formulate* the matchmaking problem as a routing problem in CAN space
  - Treat each *resource type* as a distinct CAN dimension
  - Map nodes and jobs into the CAN space
    - Resource *capabilities* and *requirements*, respectively

Modified Content-Addressable Network

- Matchmaking Process
  - Search for *the closest node whose coordinates in all dimensions meet or exceed the job's requirements*

Modified Content-Addressable Network



Modified Content-Addressable Network

- *Virtual* Dimension
  - Clustering of nodes and jobs
    - Resource capabilities and Requirements
    - Distribution of ownership of a zone and Load imbalance
  - Supplement the *real* dimensions
    - Corresponding to node capabilities
  - Coordinates for nodes and jobs for the virtual dimension generated *uniformly at random*

Multiple Candidate Run Nodes

- Search proceeds until at least *k* capable nodes are found for better load balancing
  - Choose the *least loaded* of *k* capable nodes to run the job
    - Measured by size of the job queue at the time the matchmaking is performed
  - *k* = 5 is enough from large set of experiments both for RN-Tree and CAN

RNT vs. CAN

- Underlying different rationales
  - RNT (*matchmaking after load balancing*)
    - Balance load by randomizing job assignments
    - Perform matchmaking by passing static capacity information around the tree
    - *First-fit* approach
  - CAN (*load balancing after matchmaking*)
    - First find a node whose capabilities approximately match the job's constraints
    - A short random walk among similar nodes to find one that is lightly loaded
    - *Best-fit* approach

Roadmap

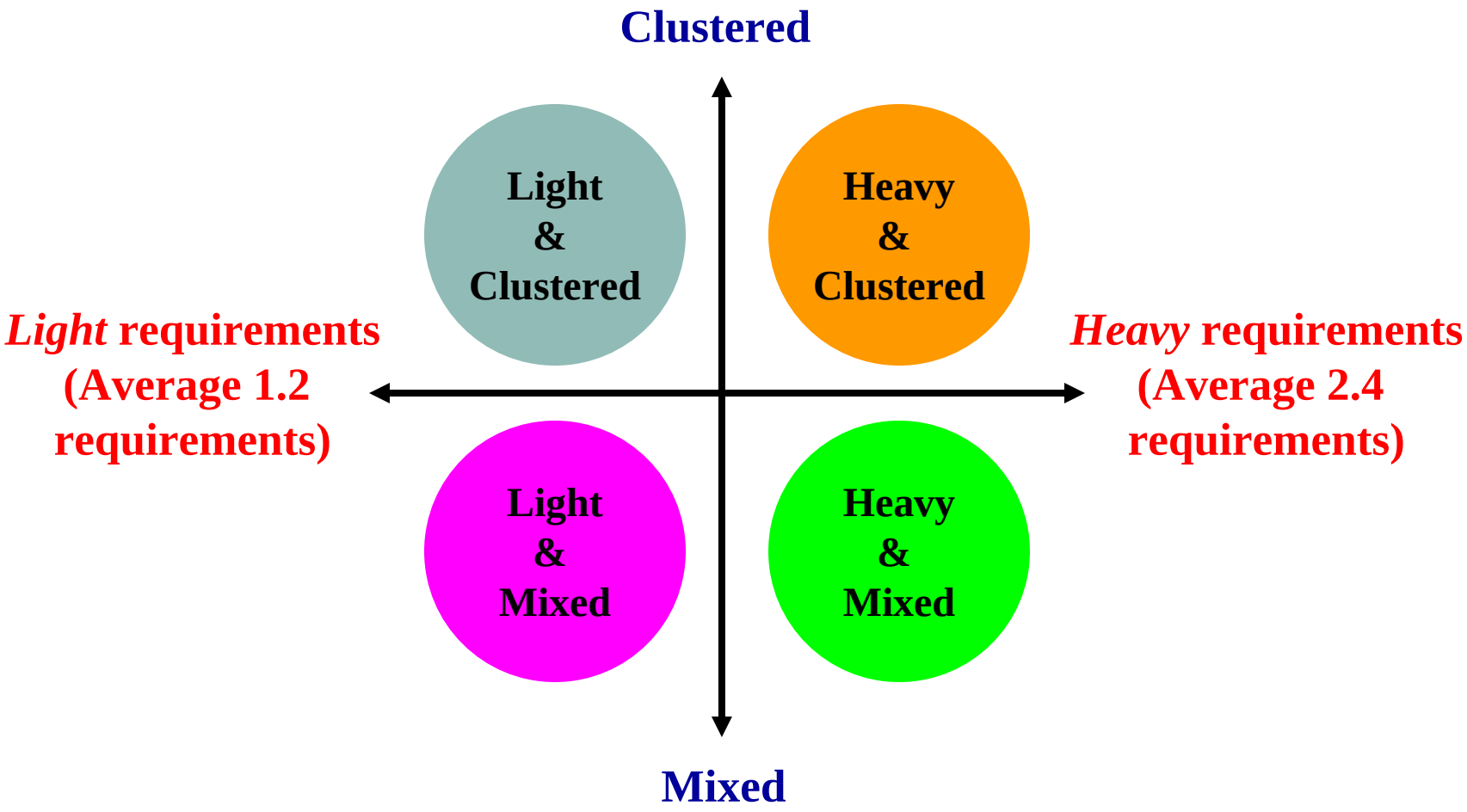
- Basic Assumptions and Overall Goals
- Content-Addressable Network
- Rendezvous Node Tree
- Experimental Results from GRID 2006
- Conclusion

Experimental Setup

- Simulation based experiments
  - Event-driven simulator
  - Model a P2P desktop grid environment with a heterogeneous set of nodes and jobs
- Traffic workloads
  - A set of *nodes* and *events*
    - Node profiles are based on *clustering model*
  - Resource types: CPU, Memory, Disk
  - Heavily loaded environment
    - 1000 nodes and 10000 jobs
    - Very short average job inter-arrival time

Experimental Setup

- Four different static test workloads



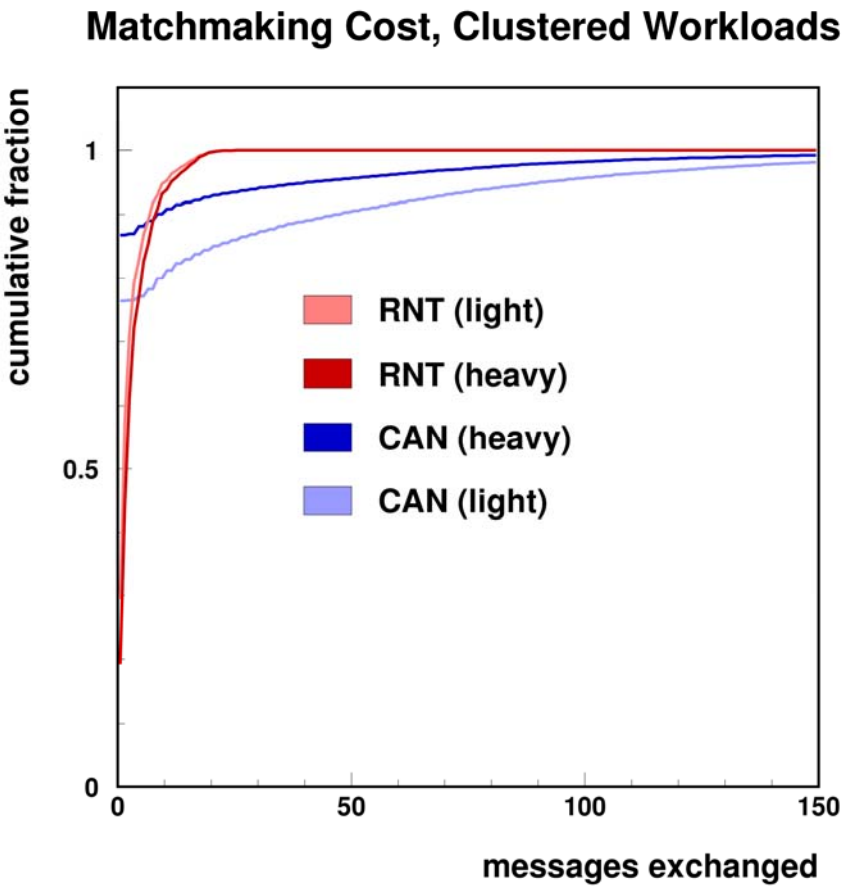
Experimental Setup

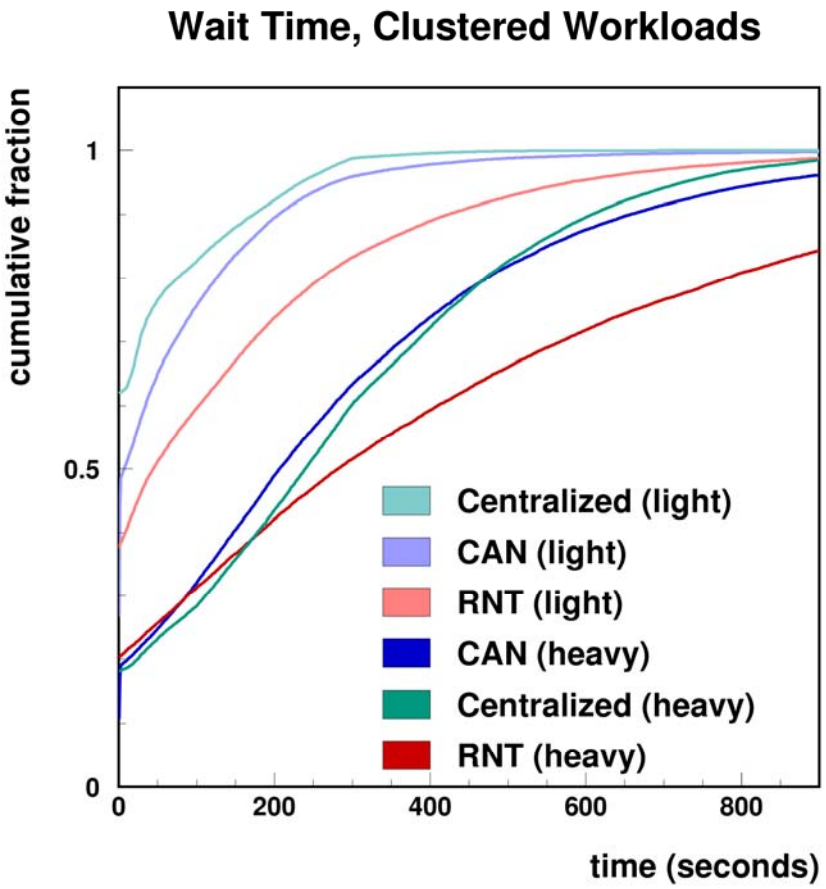
- Performance metrics
  - *Matchmaking Cost*
    - Number of messages required for finding candidate run nodes by the owner node
  - *Wait Time*
    - Amount of time between when a job is injected and when it actually starts running
  - *Queue Length*
    - Length of non-preemptive run queue seen by a job when it is finally assigned to a run node

Centralized Matchmaker

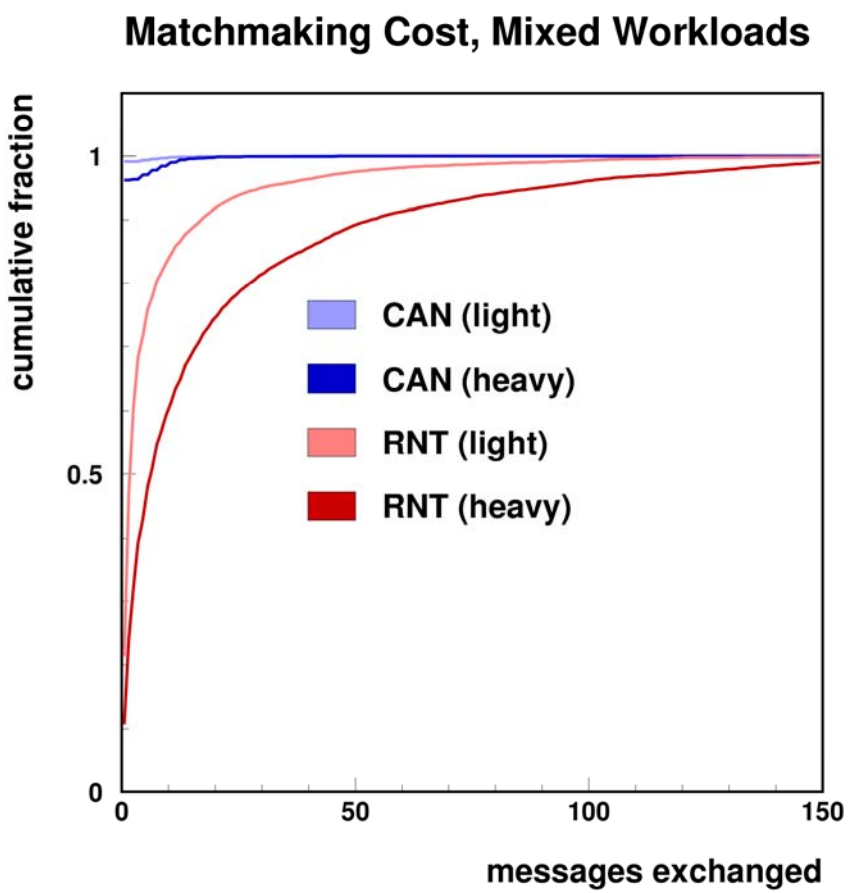
- *Online* and *global* scheduling mechanism
  - Maintains global information about the current state of the entire system (an *oracle*)
    - Instantaneous snapshot in time of the system
  - Assign a job to the node that both satisfies job requirements and has the shortest job queue
  - A greedy algorithm
  - Not feasible in a complete implementation of P2P system
    - Performance and Robustness

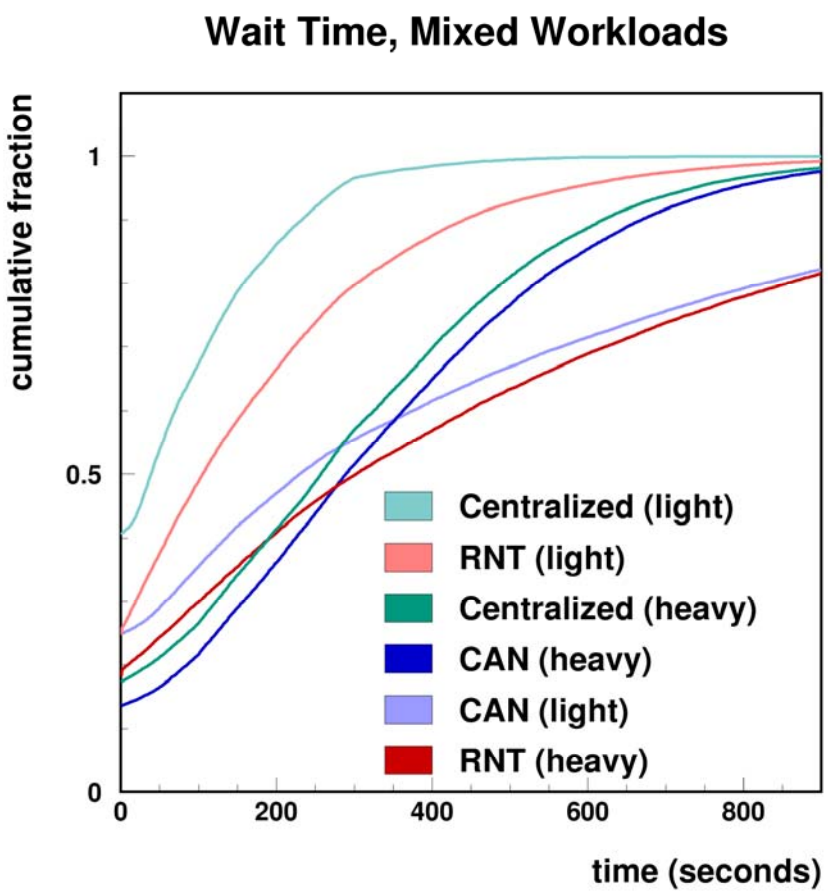
Results for Clustered Workloads





Results for Mixed Workloads



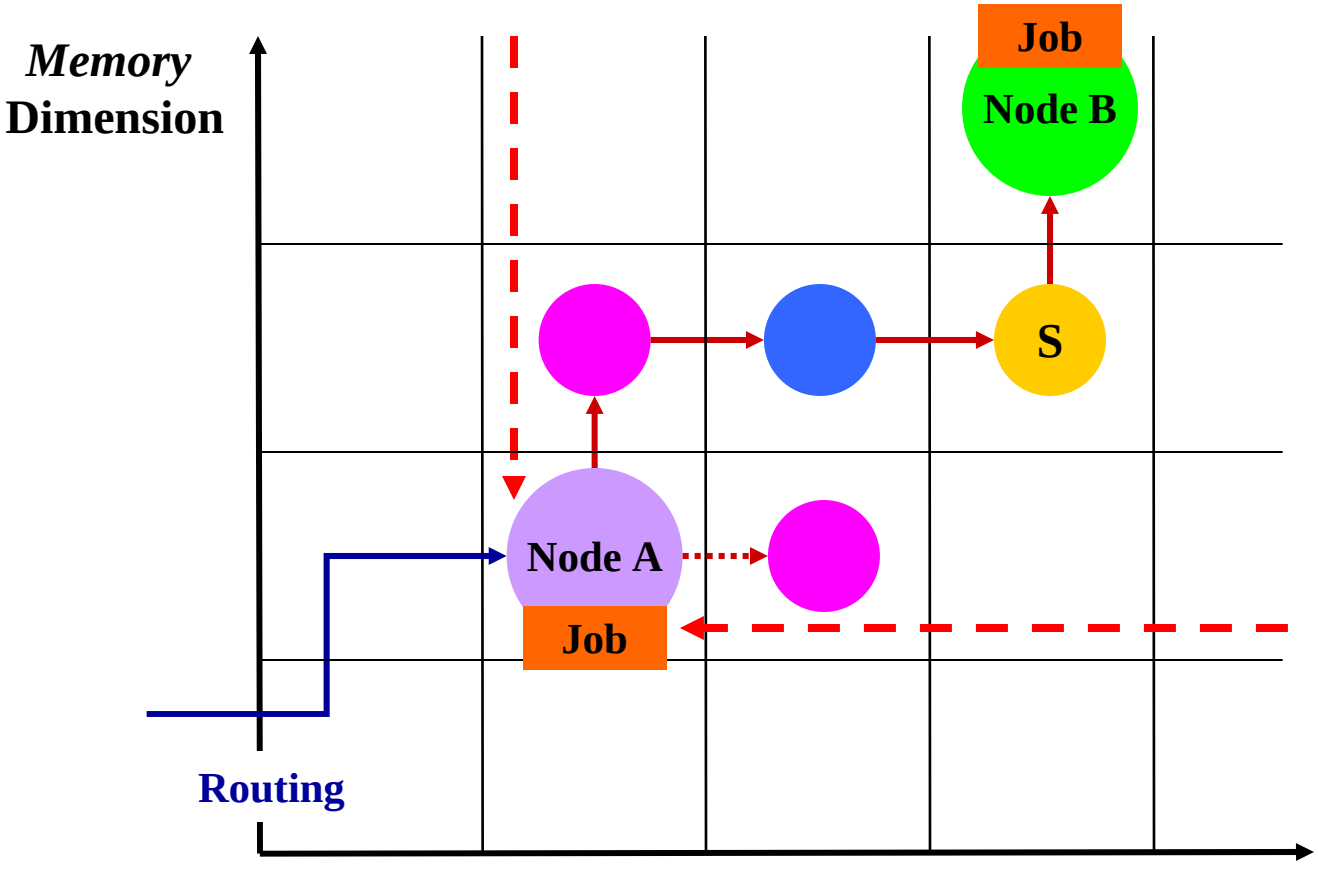


Wrap Up

- CAN and RNT algorithms balance load almost as well as the centralized algorithm
  - with low overhead (few messages)
- Overall, the CAN algorithm shows good performance for most workloads
- CAN's poor performance with the *light mixed* workloads
  - A broader problem in the robustness of load balancing

Improving CAN-based Algorithm

- Employing *Dynamic Aggregated Resource Information*



- Aggregate Resource Information
- Choose the least loaded direction
- Push a job into under-loaded region
- Stop pushing
- Choose the best run node

Ongoing Work

- Improving CAN-based matchmaking algorithm (to appear in HPDC 2007)
- Deploying the prototype system for real workloads and real machines
- Better characterization of real workloads
  - via consultation with Astronomy collaborators, and automated mining of Condor system logs