CompuP2P: An Architecture for Sharing of Computing Resources In Peer-to-Peer Networks With Selfish Nodes

Rohit Gupta and Arun K. Somani

(firstname@iastate.edu)

Dependable Computing and Networking Laboratory

Iowa State University

Iowa State University

Outline

- CompuP2P overview
- Prototype implementation for compute power sharing
  - Comparison with SETI@Home, Condor, and POPCORN
- Open Issues

CompuP2P: An Overview

- CompuP2P is a peer-to-peer (P2P) utility infrastructure designed to span WANs
- Dynamically build markets for a computing resource
- Uses game theoretic ideas to govern pricing of computing resources
- Usage
  - Provide computation capabilities to processing-intensive user applications, like network simulations, graphics
  - Support storage intensive applications such as data-bases and file systems

System Model

- Assumes a P2P configuration that uses Chord for addressing and peer connectivity
- Nodes are selfish, earn profit by selling their computing resources
  - Sellers incur a cost, referred to as marginal costs
- Resource Units
  - Compute power: cycles/sec for T time units
  - Memory storage: giga(mega) bytes for T time units

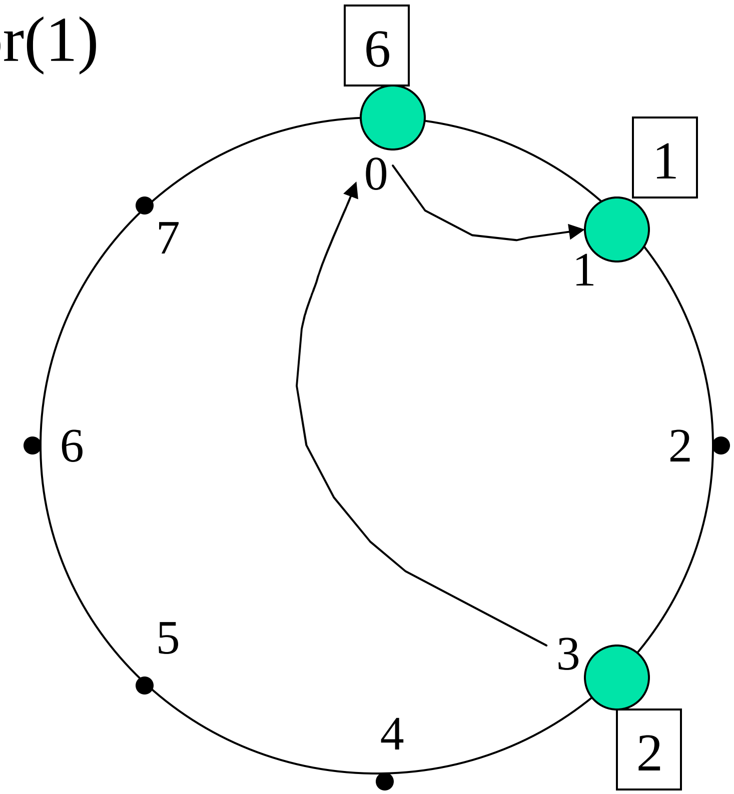
Iowa State University

Overview of Chord

- Chord provides fast distributed hash function that maps keys to nodes
- Each node and key is assigned an m-bit identifier
- Identifiers are ordered on an identifier circle modulo 2^m
- Key k is assigned to the first node (called the successor node) whose identifier is equal to or follows (the identifier of) k in the identifier space

Chord Lookup Protocol

Lookup successor(1)
from node 3



Keys 1, 2, 6

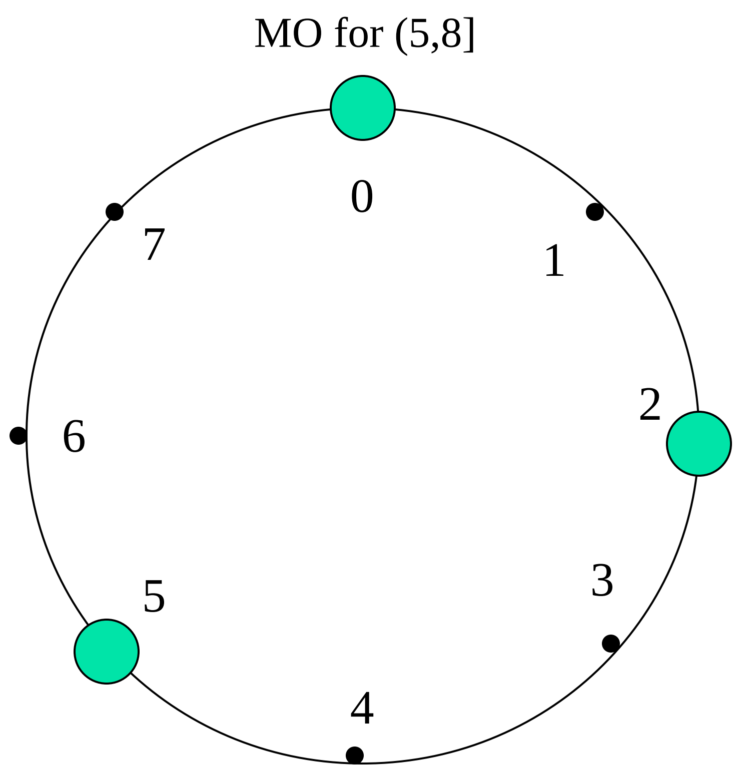
Successor(1) = 1
Successor(2) = 3
Successor(6) = 0

IOWA STATE UNIVERSITY

Construction of Compute Power Markets

- Markets for different amounts of compute power are created
- A market deals in only one type of commodity.
  - Commodity here refers to compute power in a certain well-defined range
- The same node can be responsible (i.e. be a market owner MO) for running multiple markets
- Two schemes
  - Single overlay
  - Processor overlay

Single Overlay Scheme



- The number of CPU cycles/sec gives the Chord ID of the market and the successor is the MO
  - MO = successor(C)
- Simple to implement
- Can lead to uneven assignment of markets among nodes and requires large number of hops

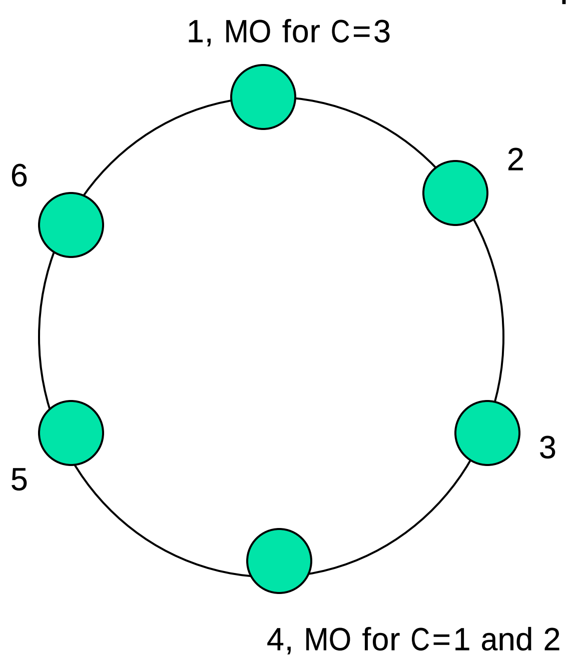
IOWA STATE UNIVERSITY

Processor Overlay Scheme

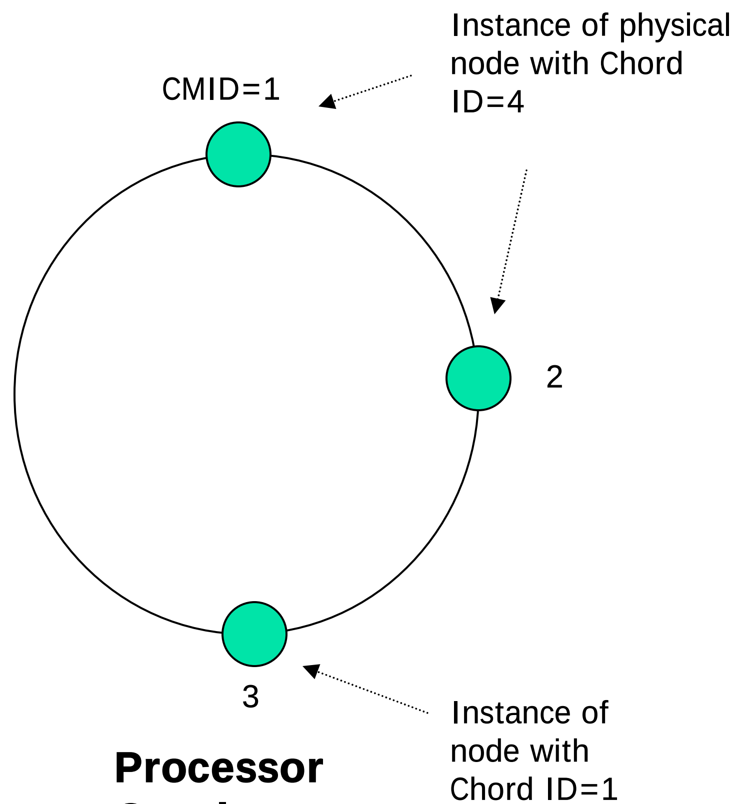
- More uniformly assign markets among nodes
  - MO = successor(hash(C))
- MOs form an additional overlay
  - IDs equal to the commodity values
- The lookup returns the IP address of the market trading in commodity equal to or greater in value than requested
  - Emulates the best-fit approach
  - Lookup is faster (O(log M) steps) in processor overlay
  - Requires extra overhead

Processor Overlay Scheme

C = average idle capacity in cycles/sec



Chord overlay
(numbers next to the nodes are the Chord IDs)



Processor Overlay

Incentives to Market Owners (MO)

- MOs make profit by charging listing price
- Fixed listing pricing
  - Same price charged to all the sellers (buyers)
  - Simple but unfair and difficult to implement
- Variable listing pricing
  - Depends on the dynamics of the markets
  - Fairer but trickier due to selfish MOs

Incentives to Sellers

- Use of marginal costs is the optimal pricing strategy
  - Bertrand oligopoly
    - Sellers have control over prices
    - Prices equal to marginal costs

- ...means NO profits !!!

Pricing Compute Power

- Reverse Vickrey auction for fixed listing pricing
  - Select the lowest cost supplier at the price of the second lowest marginal cost
- Max-min payoff strategy for variable listing pricing
  - **Set the payoffs to the MO and seller opposite to each other**
    - Sellers 1,2,...N with costs MC_1, MC_2,...MC_N in increasing order of values
    - Buyer relies on the MO to get information about the sellers
      - Buyer looking to minimize its cost
    - Payoff functions used by buyers are well known

IOWA STATE UNIVERSITY

Max-Min Payoffs

$$Payoff_{MO} = \frac{(MC'_N - MC'_1)}{(MC'_N)^2} \Rightarrow \frac{1}{4 * MC_1}$$

$$Payoff_{seller} = MC'_1 + 1$$

Proposition

a) Collusion is avoided

b) The lowest cost supplier is always selected

c) The total cost to the buyer is bounded

d) Payoffs are market dynamics dependent

Prototype Implementation

- Implemented a Java-based prototype
  - Using it for running compute intensive simulations
  - Printing quota as a form of virtual currency
  - Users submit a task-specification file as input
    - Describe the inputs and precedence relation among the sub-tasks comprising a task
    - Class files can be downloaded from a well-defined code server
- Fault-tolerance
  - Handling node crashes
    - Dynamic checkpointing
    - Use PJama

Iowa State University

Comparison With Related Projects

- SETI@Home (UC Berkeley 1996)
  - Only one central server can allocate tasks to others
- Condor (University of Wisconsin-Madison 1985)
  - All machines under the control of a single cluster head
    - Task management, scheduling, and checkpointing is centralized
- POPCORN (Hebrew University 1997)
  - Uses a trusted centralized market

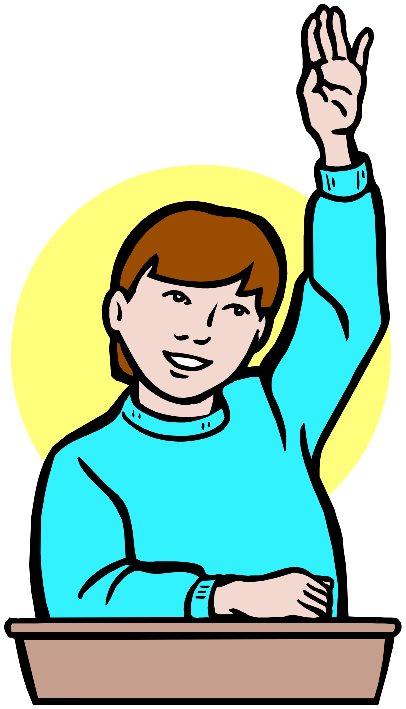
IOWA STATE UNIVERSITY

Open Issues

- CompuP2P relies on a monetary payment scheme
  - Using reputation as a substitute for currency
- Verifying computation results
  - Redundant computations
    - Can complicate pricing

Iowa State University

Questions



Iowa State University