

Load Balancing in Structured P2P Systems

Ananth Rao, **Karthik Lakshminarayanan**,
Sonesh Surana, Richard Karp, Ion Stoica
UC Berkeley

Goals and Assumptions

- Goal: To maintain the system in a state in which *load* on a node is less than its *target*.
- Load: Depends on the particular P2P system.
 - E.g. Storage, Network bandwidth.
- Target: Maximum load a node is willing to hold.
- Assumptions:
 - Nodes of the P2P system are co-operative.
 - Only one bottlenecked resource, e.g. storage, network bandwidth.

Outline

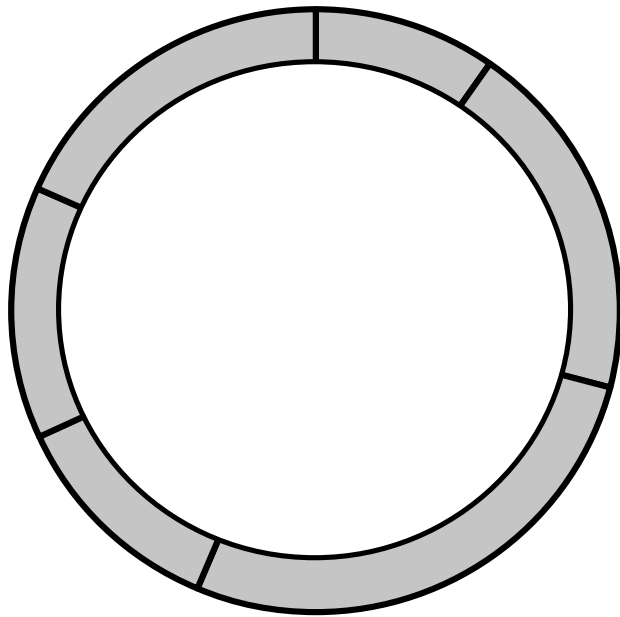
- **Motivation and Preliminaries**
- Load balancing algorithms
- Evaluation

Need for Load-Balancing

- Choice of identifiers determines the node the objects are mapped to.
 - Objects and nodes are assigned IDs at random.
 - Leads to $O(\log N)$ imbalance in the load of a node.
- Problems:
 - Sizes of objects might not be the same.
 - Object IDs might not be chosen at random.
 - Heterogeneity in the capabilities of nodes.

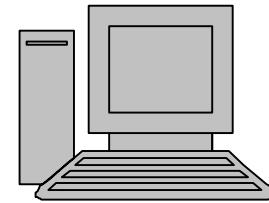
Virtual Servers

- Contiguous region of the ID space.
- Each node can be responsible for many virtual servers.
- Used in Chord/CFS.



Chord Ring

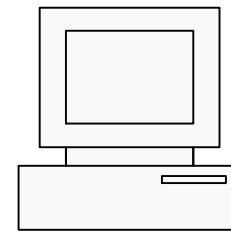
Node A



Node B

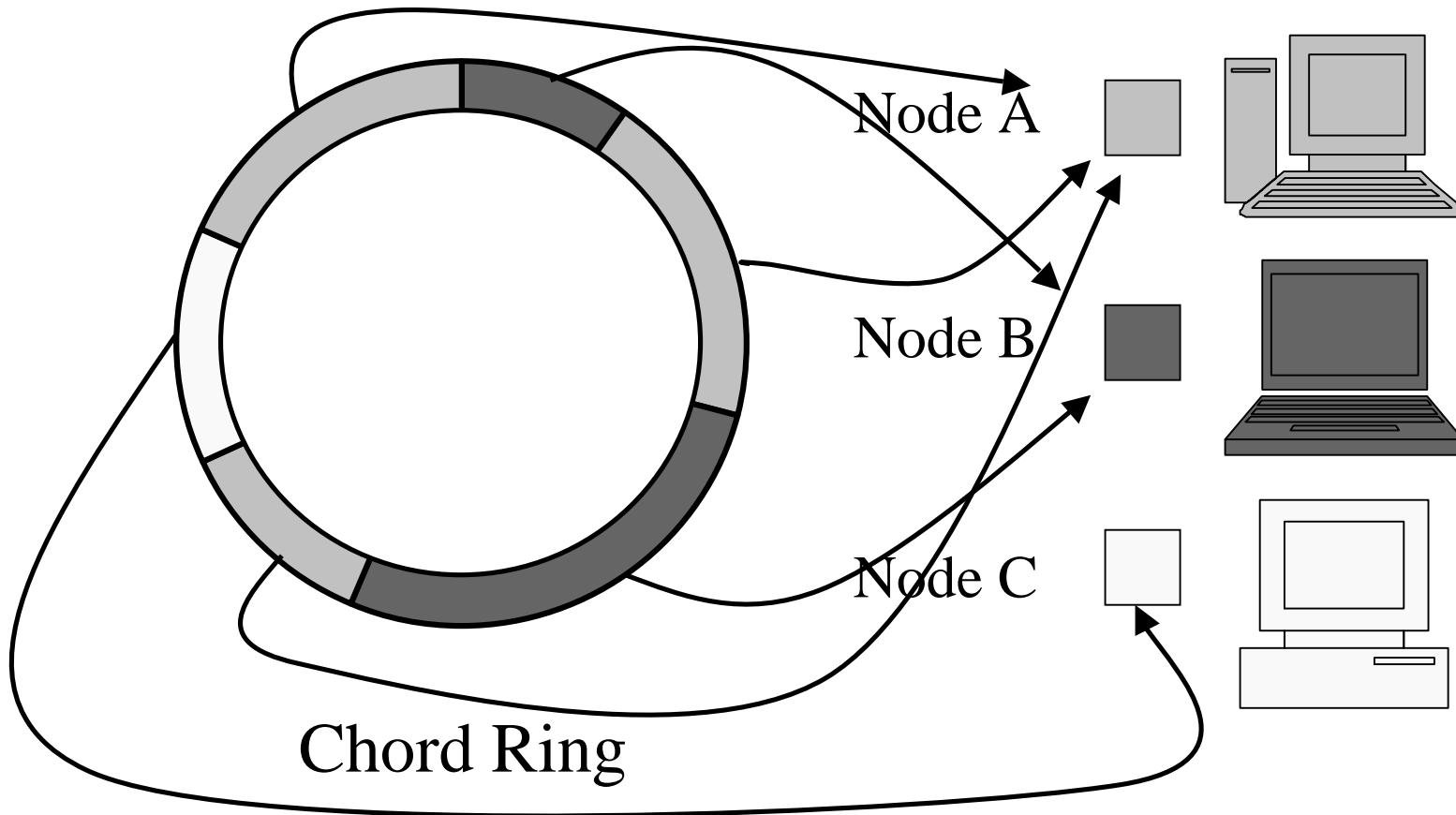


Node C



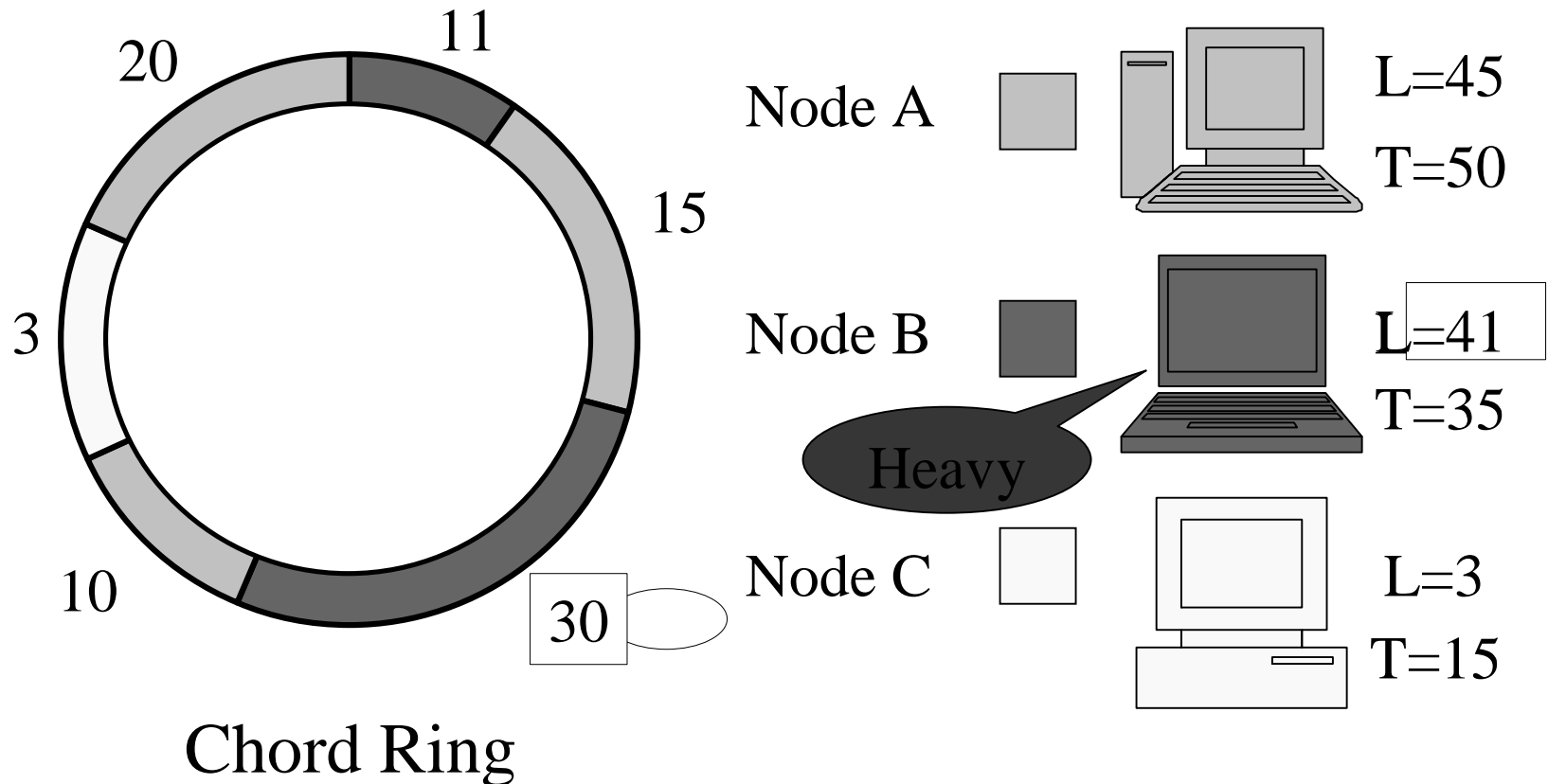
Virtual Servers

- Contiguous region of the ID space.
- Each node can be responsible for many virtual servers.
- Used in Chord/CFS.



Static Mapping of Virtual Servers

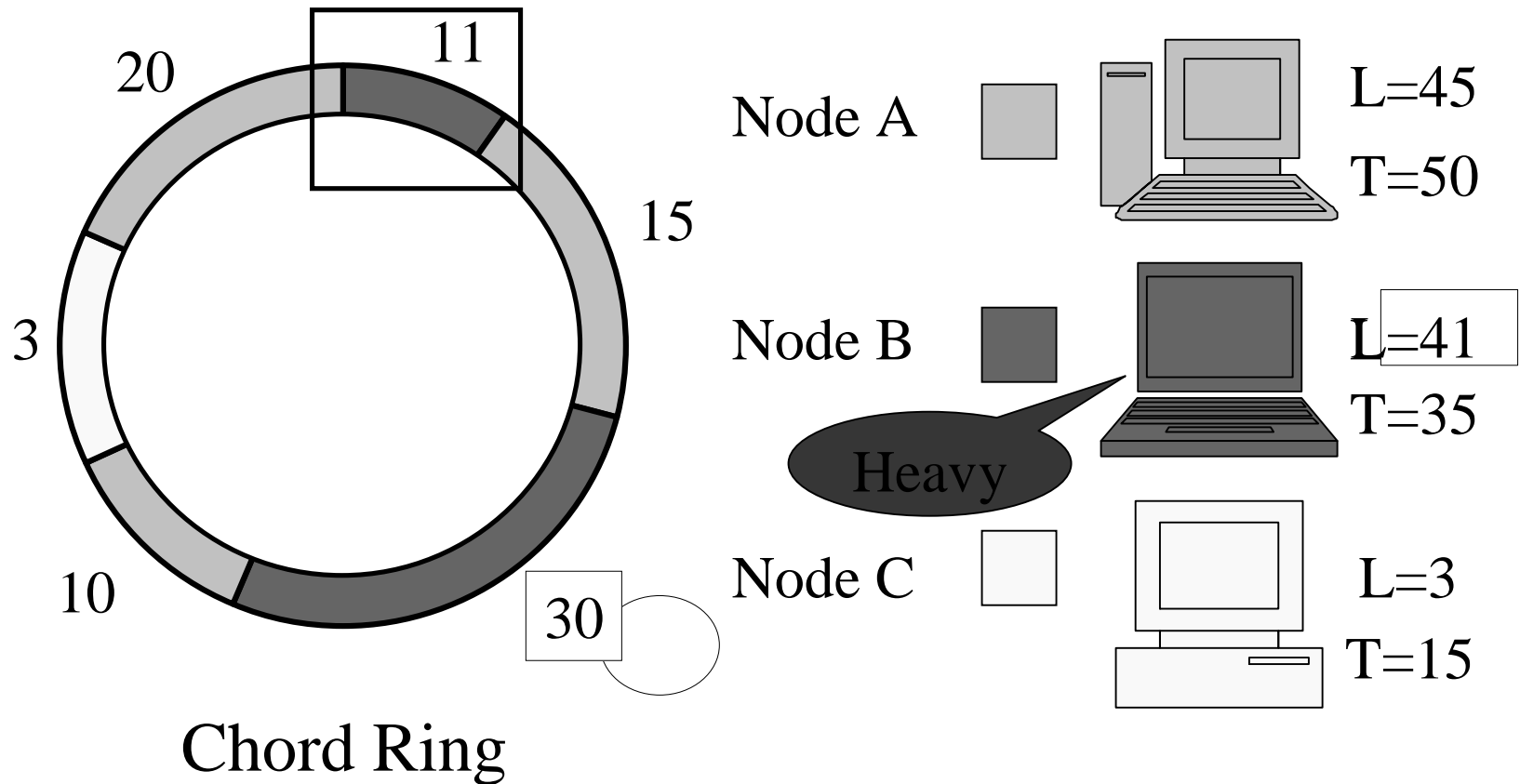
- Imbalance: Instantiate $\log N$ virtual servers at each node leads to constant factor imbalance.
- Heterogeneity: Number of virtual servers is proportional to node's target (e.g. CFS).



Dynamic Mapping of Virtual Servers

Allow dynamic re-mapping of load in a system.

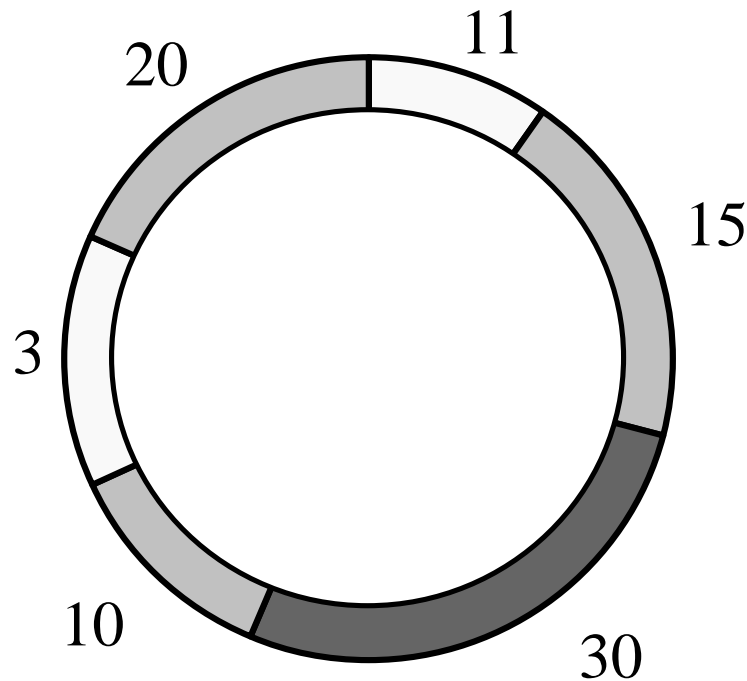
Virtual server is the basic unit of load movement.



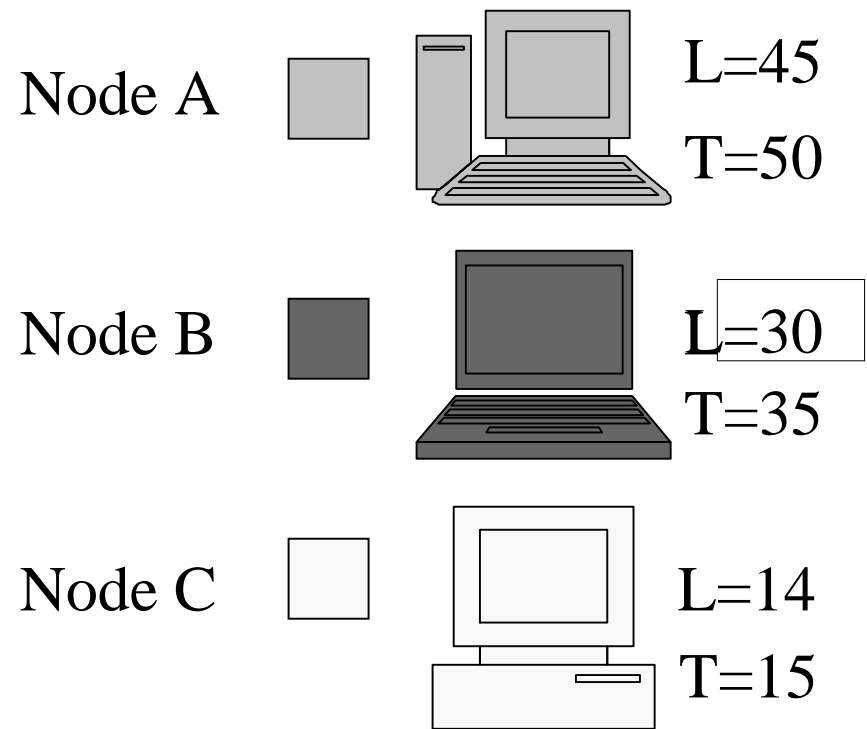
Dynamic Mapping of Virtual Servers

Allow dynamic re-mapping of load in a system.

Virtual server is the basic unit of load movement.



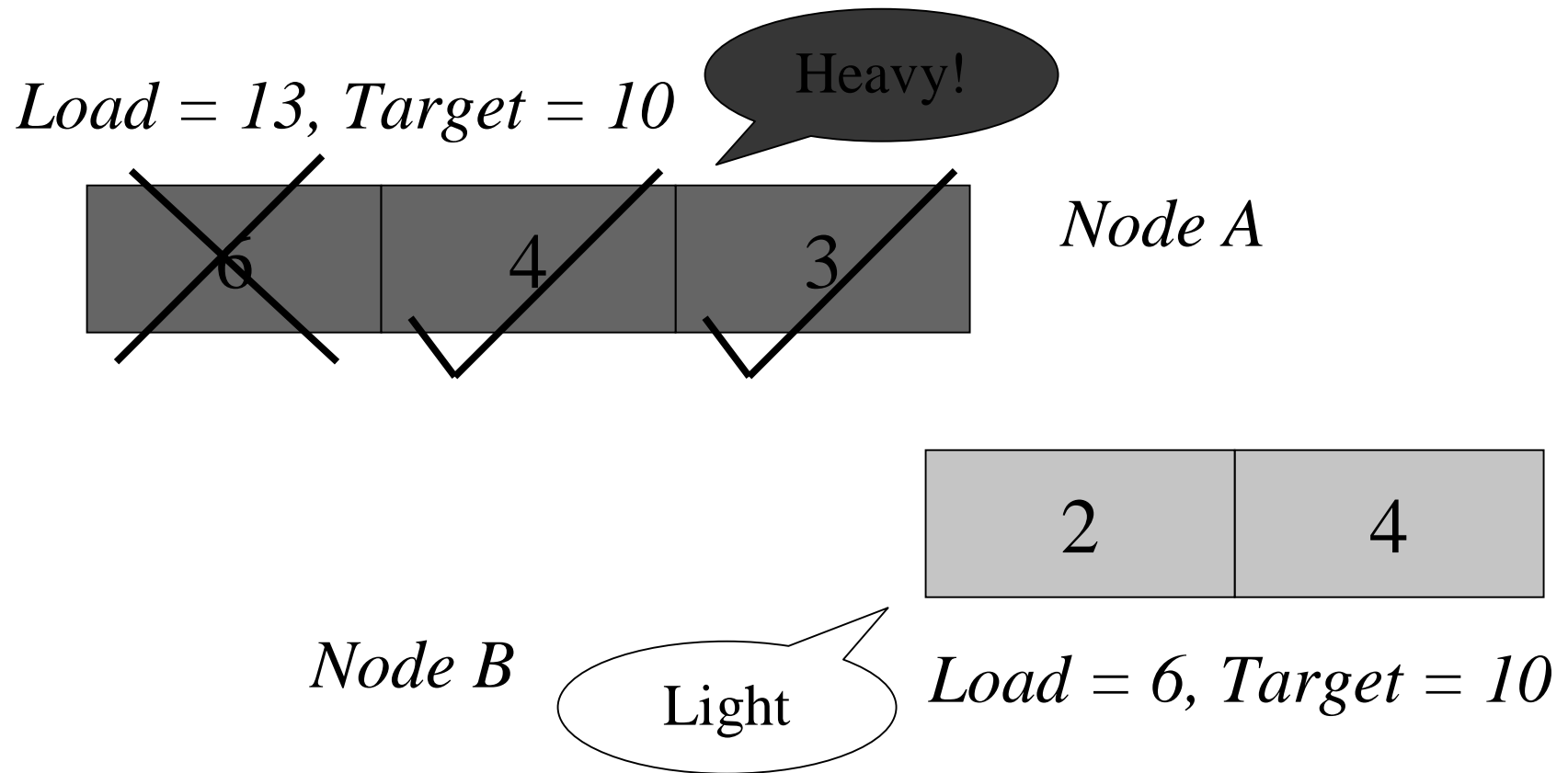
Chord Ring



Advantages

- Flexibility in being able to move load from any node to any other node in the DHT.
- Schemes proposed so far are restrictive in transferring load only to neighbors in the DHT.
- Movement of virtual servers appears as a *join* followed by a *leave* – supported by all DHTs.

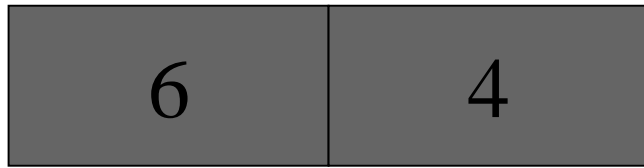
Virtual Server Transfer: An Example



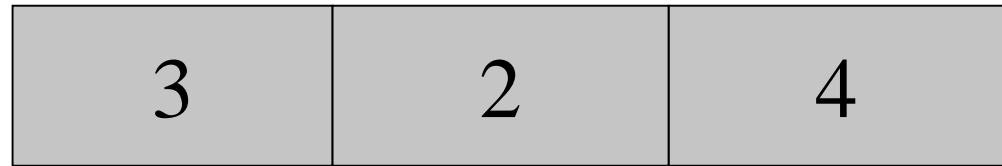
Virtual Server Transfer: An Example

Load = 10, Target = 10

Light



Node A



Node B

Light

Load = 9, Target = 10

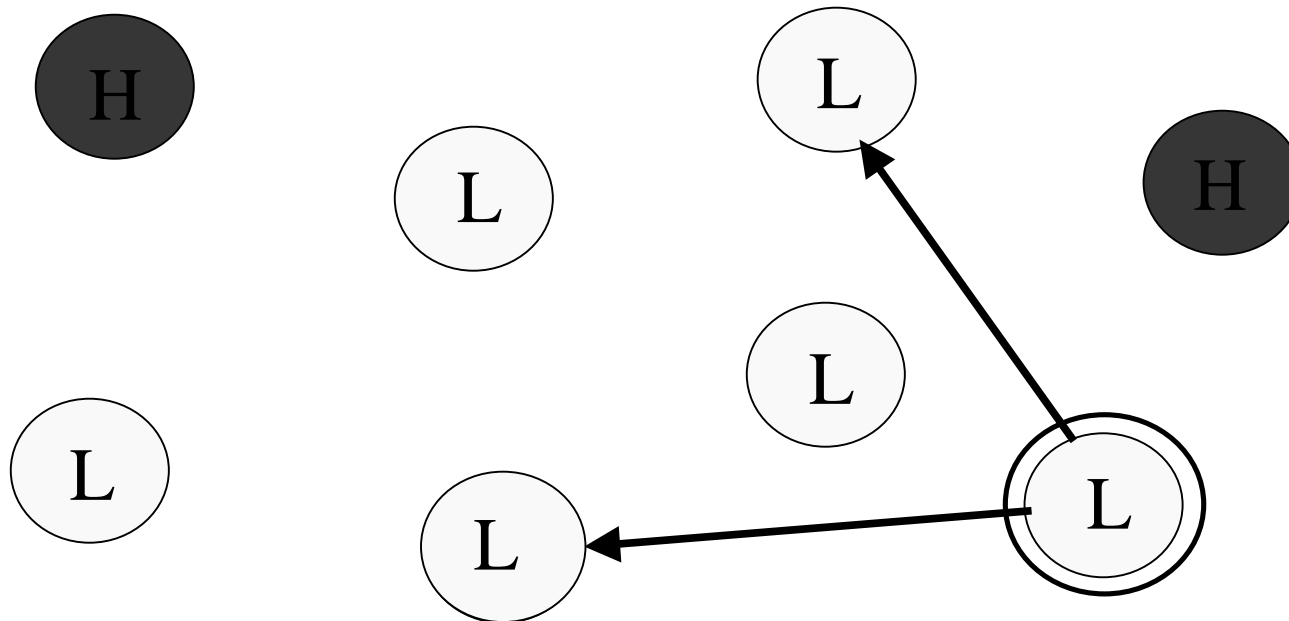
Splitting of virtual servers not done.

Outline

- Motivation
- **Load balancing schemes**
- Evaluation

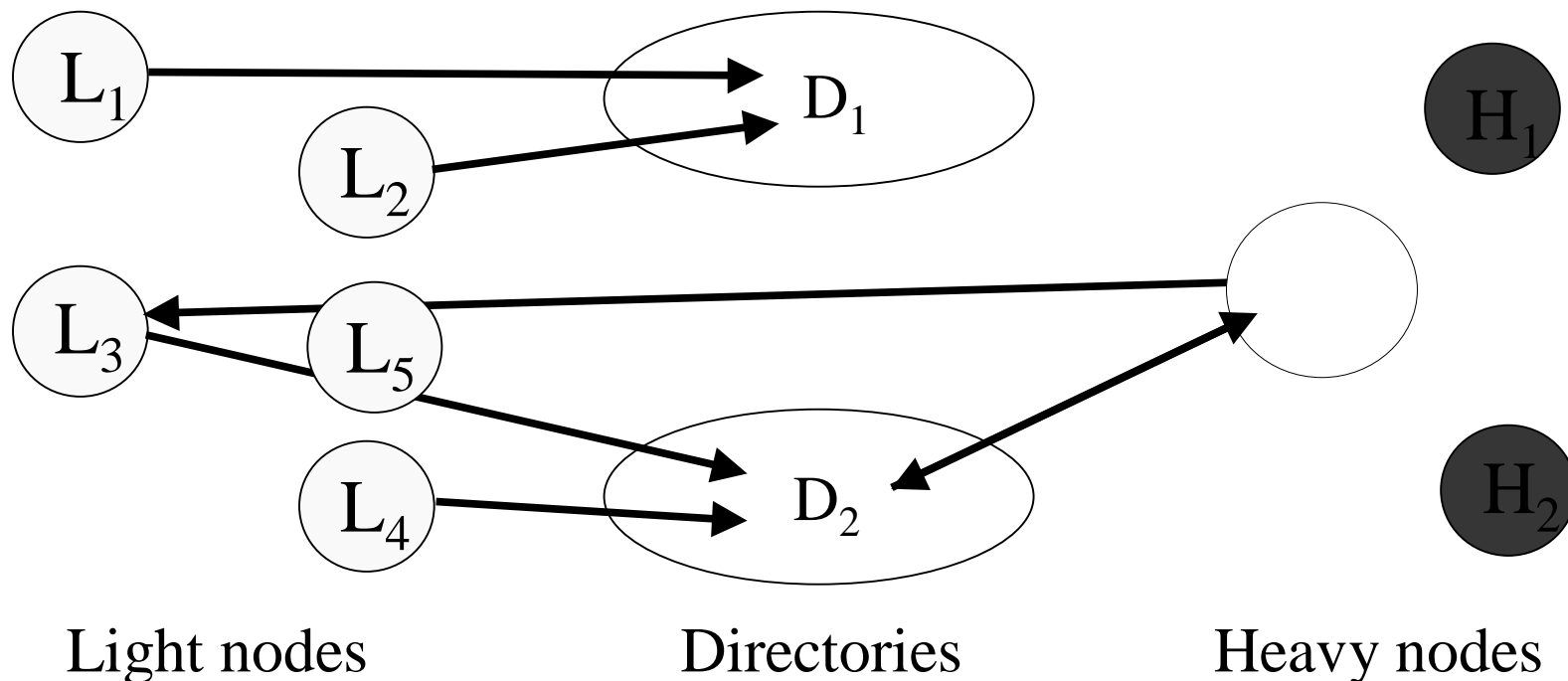
Scheme 1: One-to-One

Light node picks a random ID, contacts the node x responsible for it, and accepts load if x is heavy.



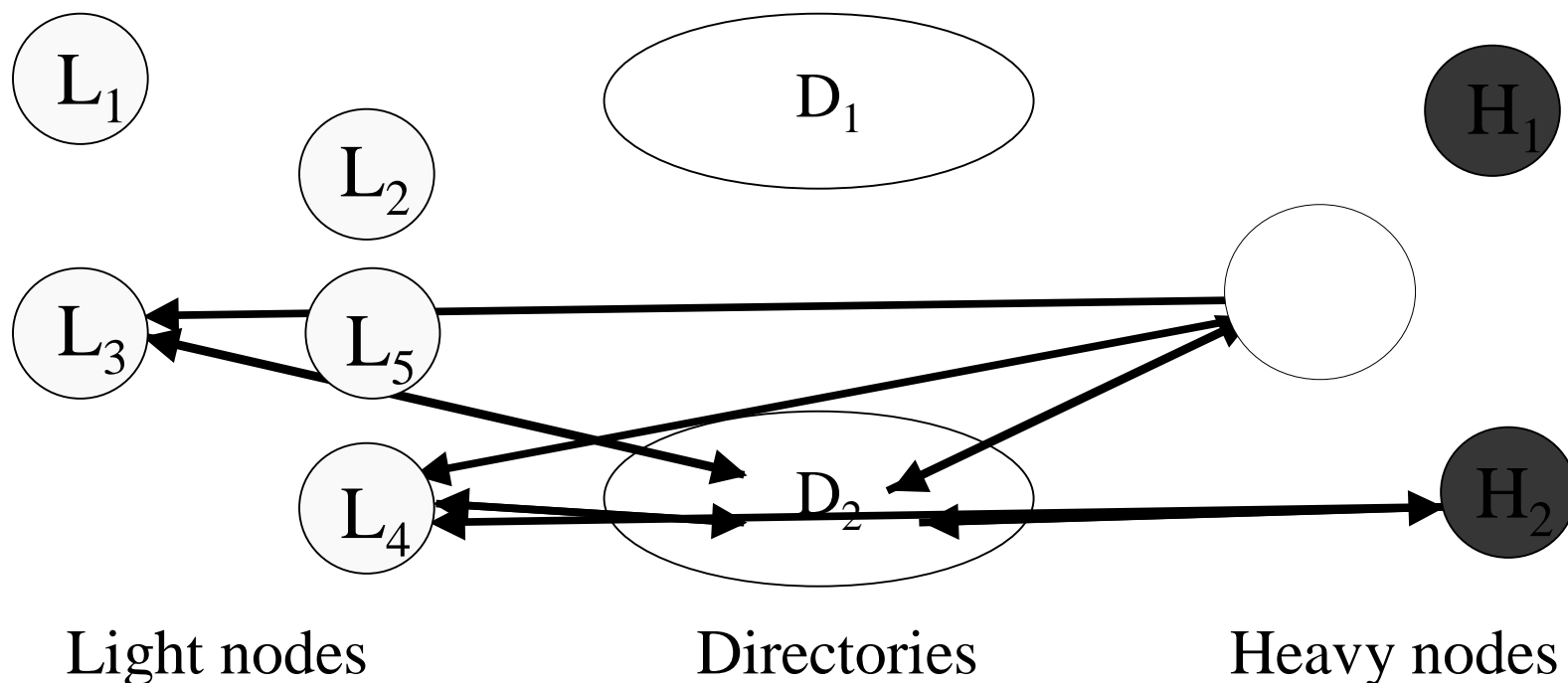
Scheme 2: One-to-Many

- Light nodes report their load information to *directories*.
- Heavy node H gets this information by contacting a directory.
- H contacts the light node which can accept the excess load.
- *Implementation:* Directories are stored in the nodes of the DHT.



Scheme 3: Many-to-Many

- Many heavy and light nodes rendezvous at each step.
- Directories periodically compute the transfer schedule and report it back to the nodes, which then do the actual transfer.
- Advantages: Best-fit heuristic reduces fragmentation of capacity.



Outline

- Motivation
- Load balancing algorithms
- **Evaluation**

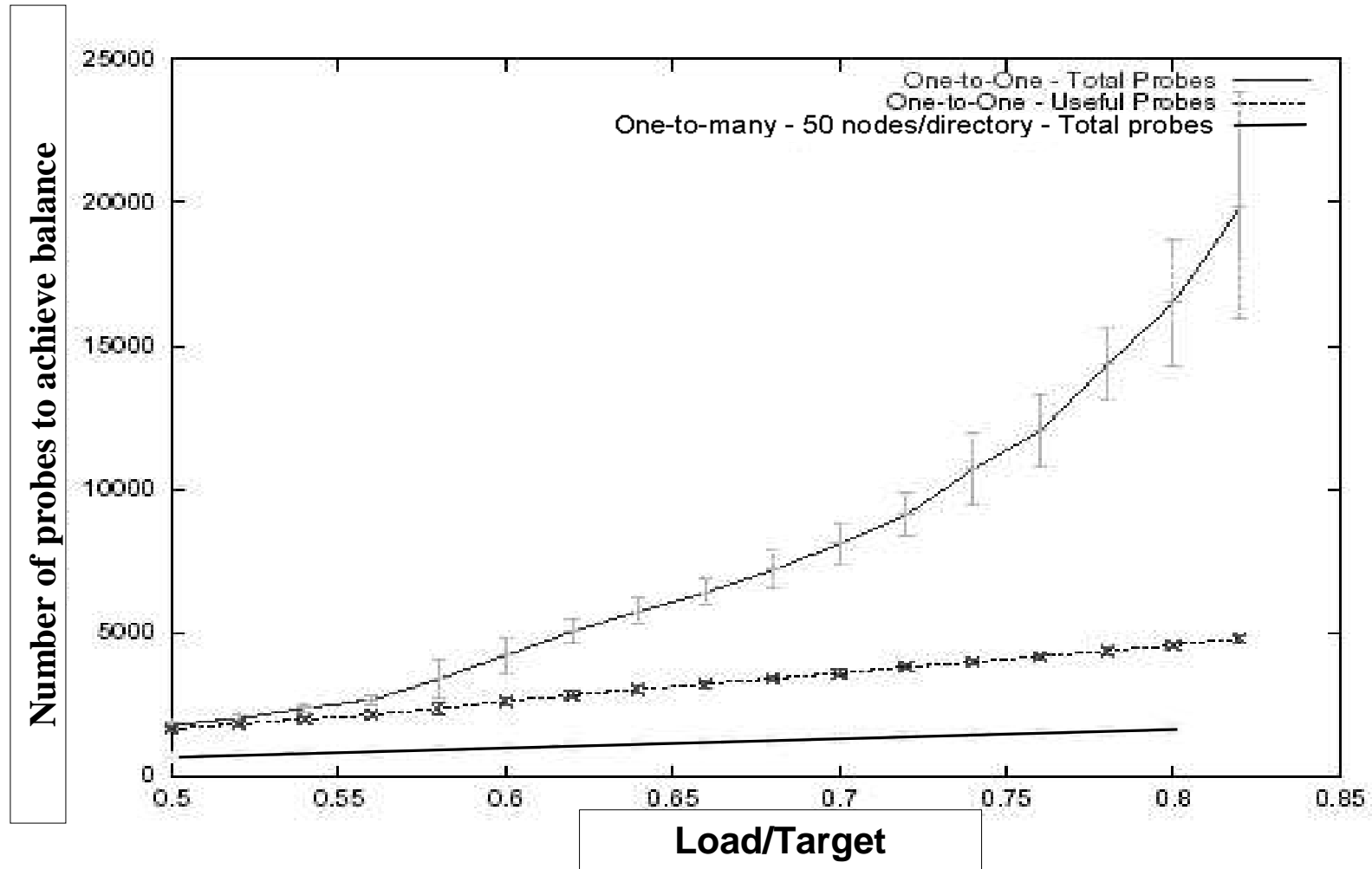
Simulation Setup

- Three schemes were simulated for up to 32,000 nodes, and compared along 2 metrics:
 - Load transferred to achieve balance.
 - Time to achieve balance.
- Distributions chosen:
 - Target of nodes: Pareto.
 - Loads on virtual servers: (i) Gaussian (ii) Pareto.

Metric 1: Total Load Transferred

- Total load transferred:
 - Load moved depends only on distribution of loads, and the target to be achieved and *not on the load balancing scheme*.
- Operational range:
 - Many-to-many scheme is able to produce balance even at very high system load (when load is up to a fraction of 0.94 of capacity).
 - Other two schemes work only up to a factor of about 0.8.

Metric 2: Time taken for Balance



One-to-one scheme may be sufficient if control traffic overhead does not affect the system adversely.

Other Results

- Size of the directory:
 - How many heavy and light nodes must come together to keep the total number of probes small?
 - Most heavy nodes shed their load by making only one probe, for number of nodes per directory as low as 16.
- Performing Swaps:
 - Move load out of a light node to accommodate a heavy virtual server.
 - Benefit only when allowed imbalance is extremely small.

Why Not Caching?

- Potential problems:
 - Does not work for certain resources, e.g. storage.
 - Need to maintain cache consistency.
 - Large number of active small objects.
 - E.g. DB applications.
 - Need to push out all of them for caching to be effective.
- Orthogonal and complementary to our schemes.

Questions?