Transactional Memory Schedulers for Diverse Distributed Computing Environments

Costas Busch

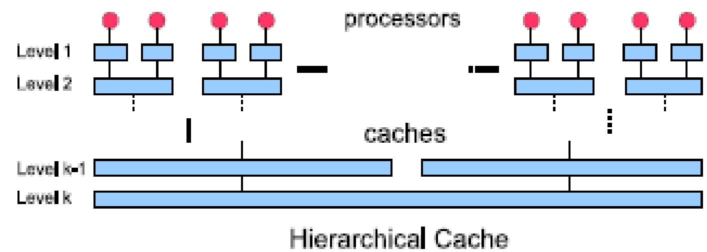
Louisiana State University

(Joint work with Gokarna Sharma)

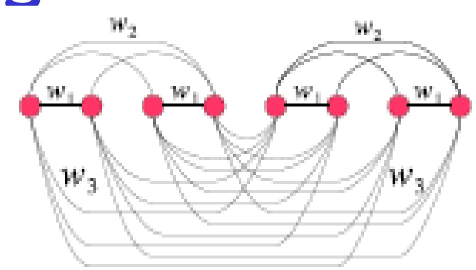
WTTM 2013

Multiprocessor Systems

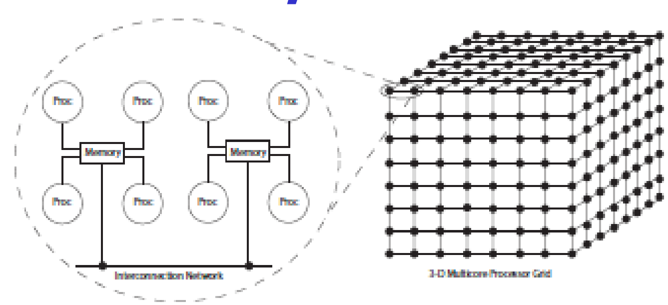
- Tightly-Coupled Systems
  - Multicore processors
  - Multilevel Cache



- Distributed Network Systems
  - Interconnection Network
  - Asymmetric communication



- Non-Uniform Memory Access Systems (NUMA)
  - Partially symmetric Communication



Scheduling Transactions

Contention Management

Determines:

- when to start a transaction
- when to retry after abort
- how to avoid conflicts

Efficiency Metrics

- Makespan
    - Time to complete all transactions
- Abort per commit ratio
    - Energy
- Communication cost
    - Time and Energy
    - Networked systems
- Load Balancing
    - Time and Energy
    - NUMA and networked systems

Inspiration from Network Problems

Packet scheduling techniques

- Helps to schedule transactions in multicores

Mobile object tracking in sensor networks

- Helps to schedule transactions in networked systems

Oblivious routing in networks

- Helps to load balance transaction schedules in NUMA

Presentation Outline

- <u>1. Tightly-Coupled Systems</u>

2. Distributed Networked Systems

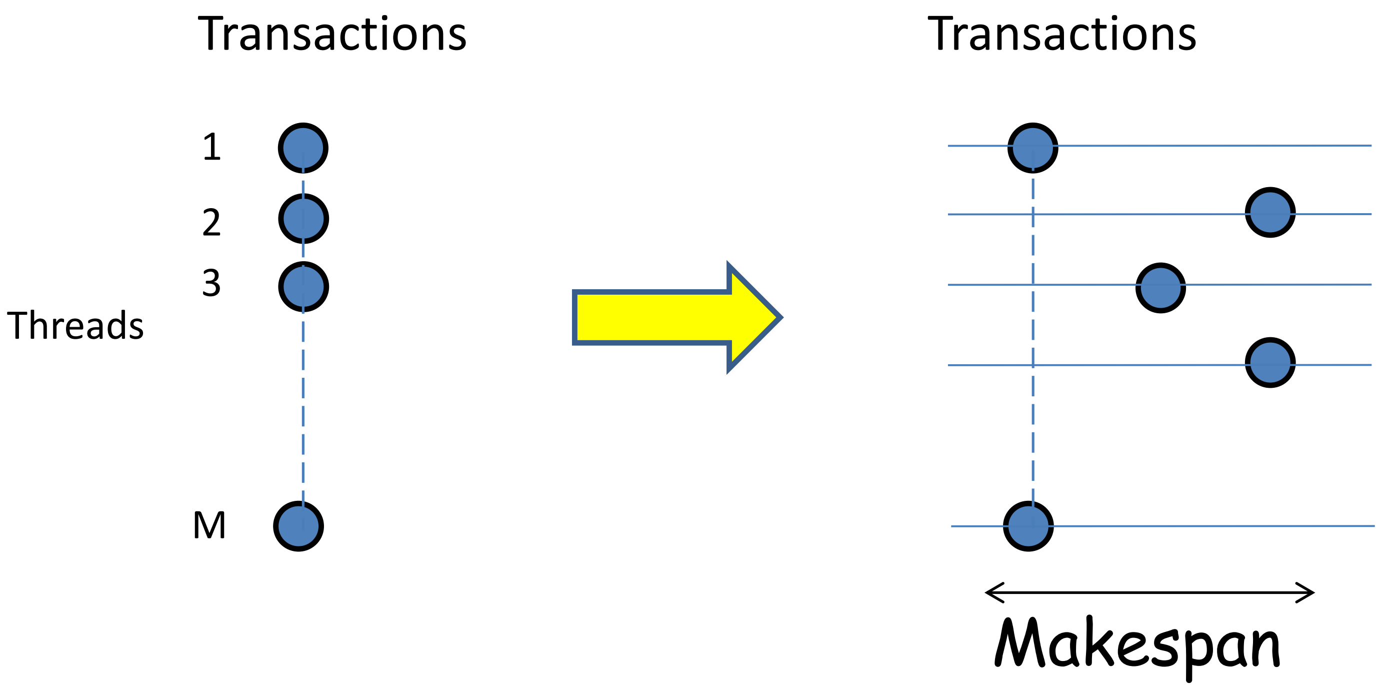
3. NUMA

4. Future Directions

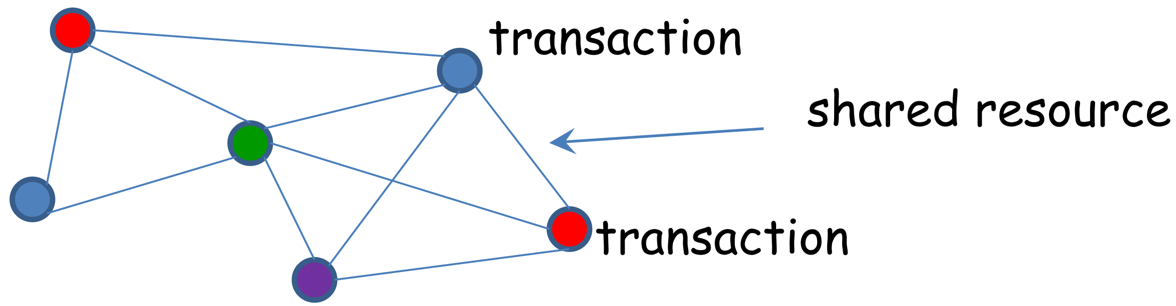
Scheduling in Tightly-Coupled Systems

One-shot scheduling problem

- *M* transactions, a single transaction per thread
- *s* shared resources
- Best bound proven to be achievable is *O(s)*



- Problem Complexity: directly related to vertex coloring



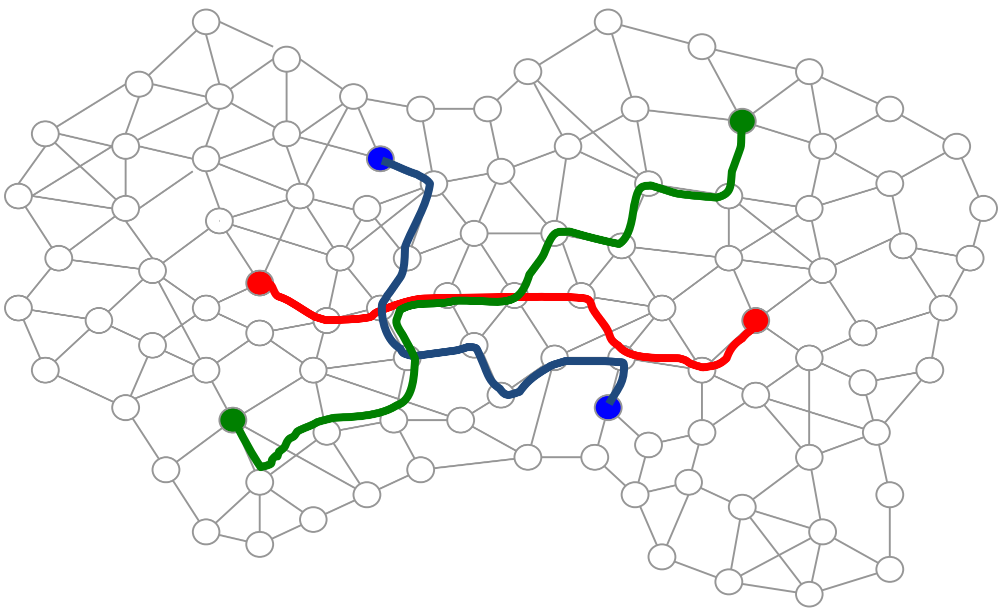
- NP-Hard to approximate an optimal vertex coloring

- ***Can we do better under the limitations of coloring reduction?***

Inspiration

Packet routing and job-shop scheduling in O(congestion+dilation) steps (1994)

F. T. Leighton , Bruce M. Maggs , Satish B. Rao

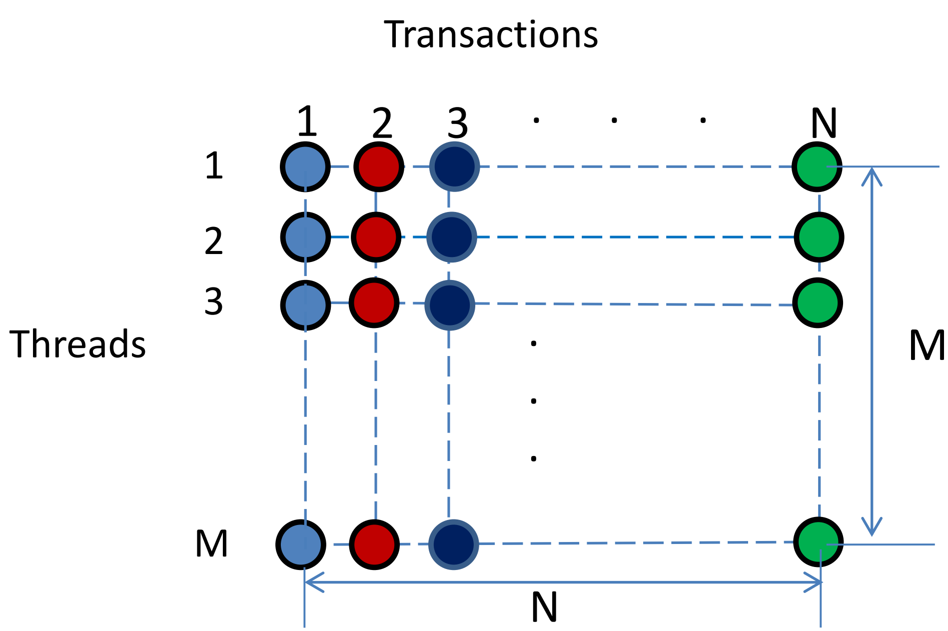


Congestion (C) = max edge utilization

Dilation (N) = max path length

Execution Window Model

- *A M × N window W*
  - *M* threads with a sequence of *N* transactions per thread
  - collection of *N* one-shot transaction sets



Makespan

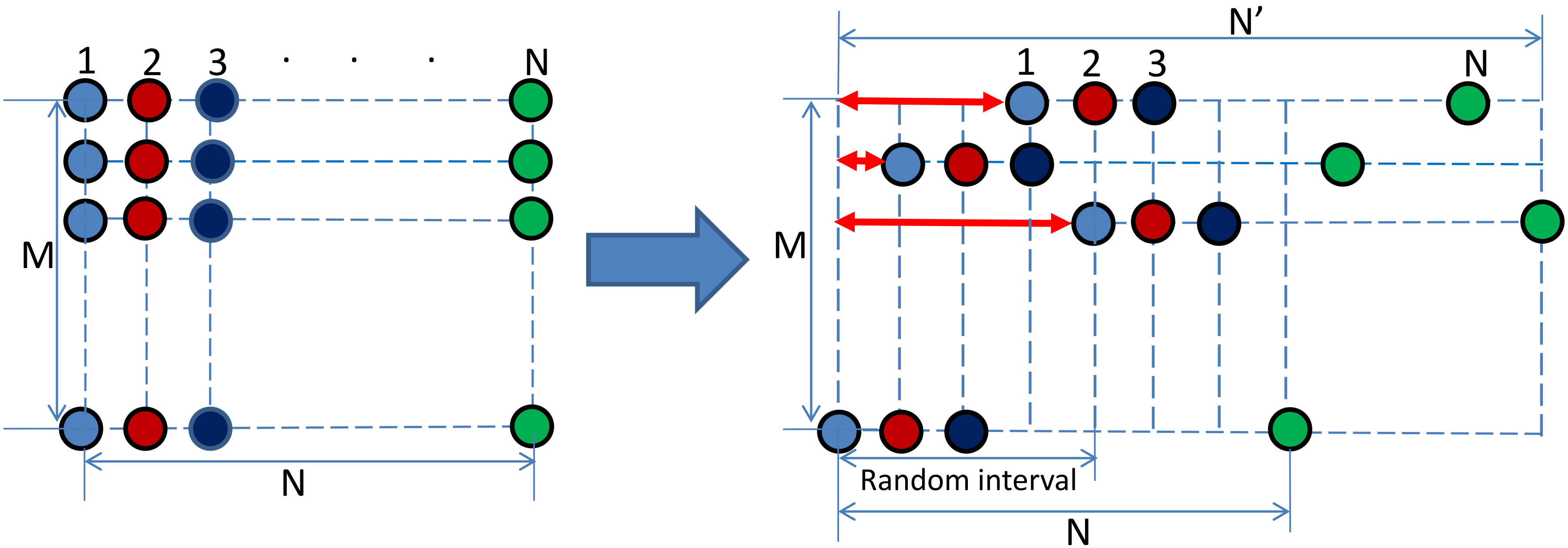
$O(C + N \log(MN))$

Analogy: Packet = thread

Path Length (N) = sequence of thread's transactions

Congestion (C)= conflicts of thread's transactions

Intuition

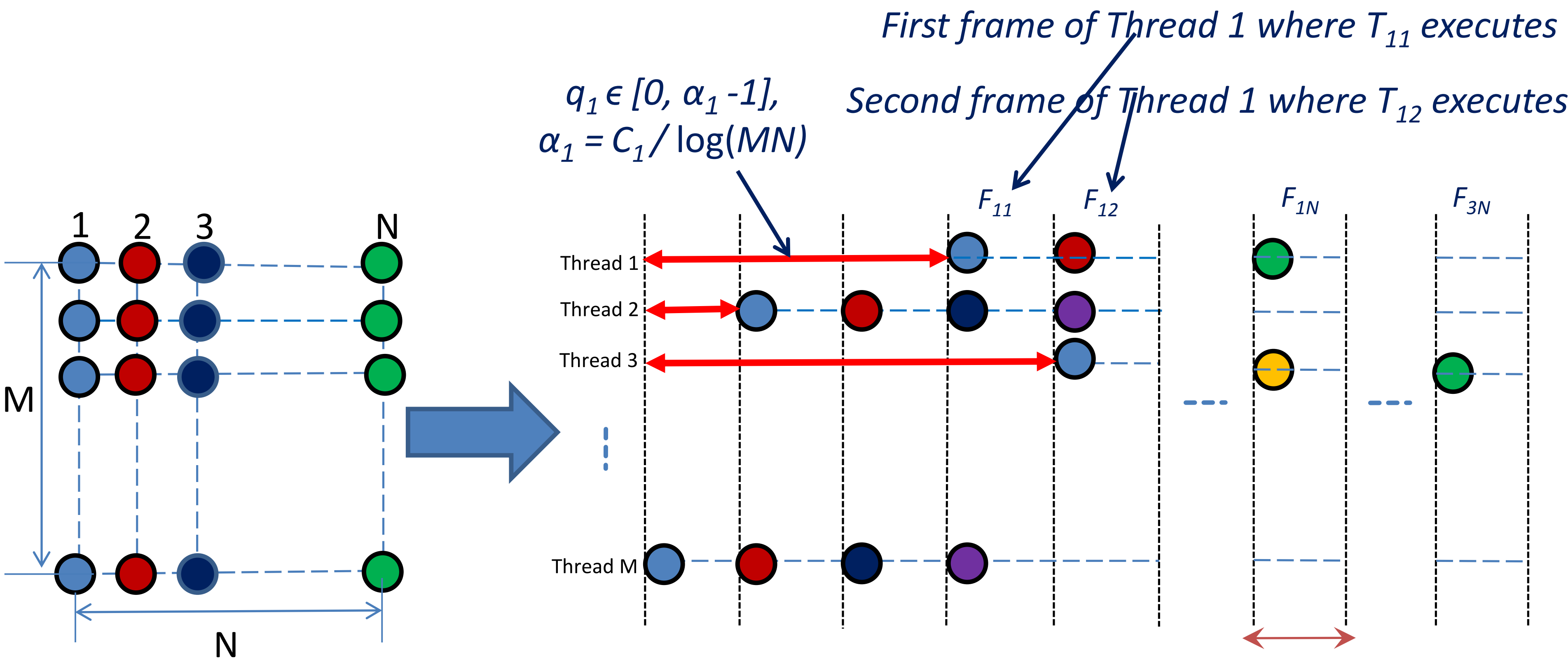


Random delays help conflicting transactions shift inside the window

Initially each thread is low priority

After random delay expires a thread becomes high priority

How it works: Frames



$C = \max_i C_i,\ 1 \le i \le M$

$$Makespan = (C / \log(MN) + Number\ of\ frames) \times Frame\ Size$$
$$= (C / \log(MN) + N) \times Frame\ Size$$
$$= O(C + N \log(MN))$$

Challenges

- Unit length Transactions
- C: may not be known
  - Try to guess it for each transaction
  - Use random priorities within frame
- N: what window size is good?
  - Dynamically try different window sizes

Presentation Outline

1. Tightly-Coupled Systems

➢ <u>2. Distributed Networked Systems</u>

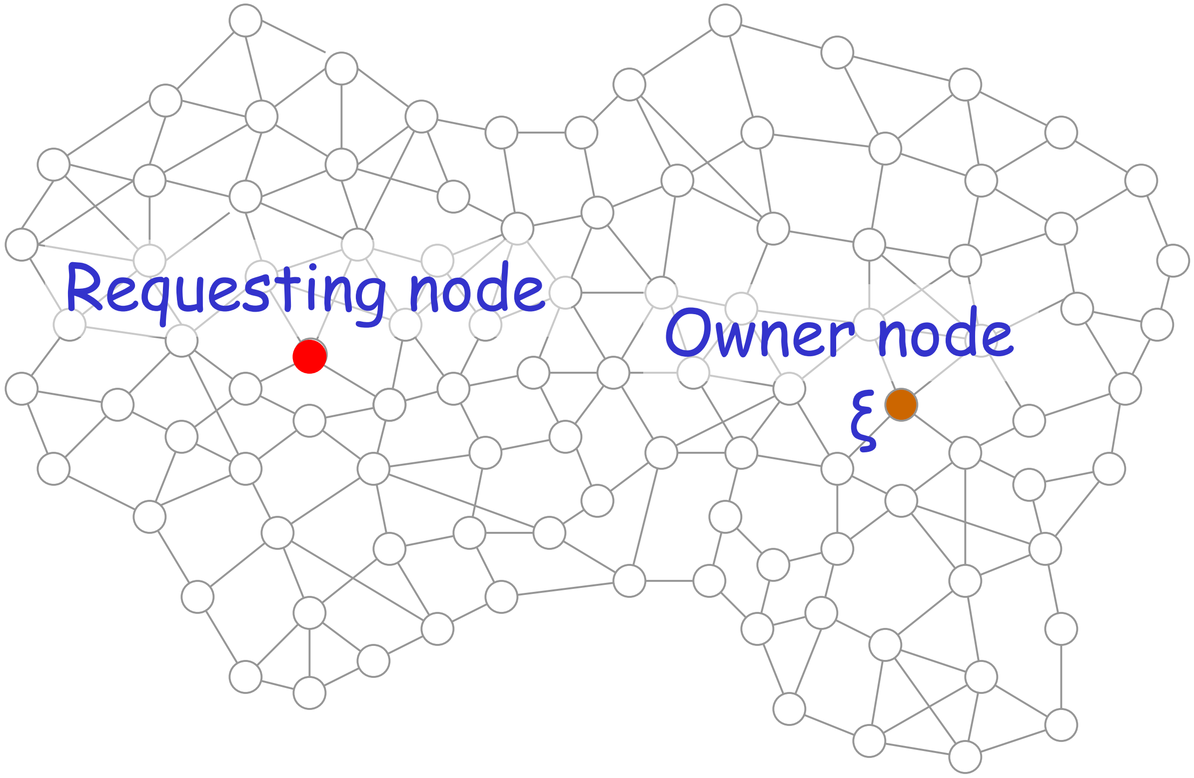
3. NUMA

4. Future Directions

Distributed Transactional Memory

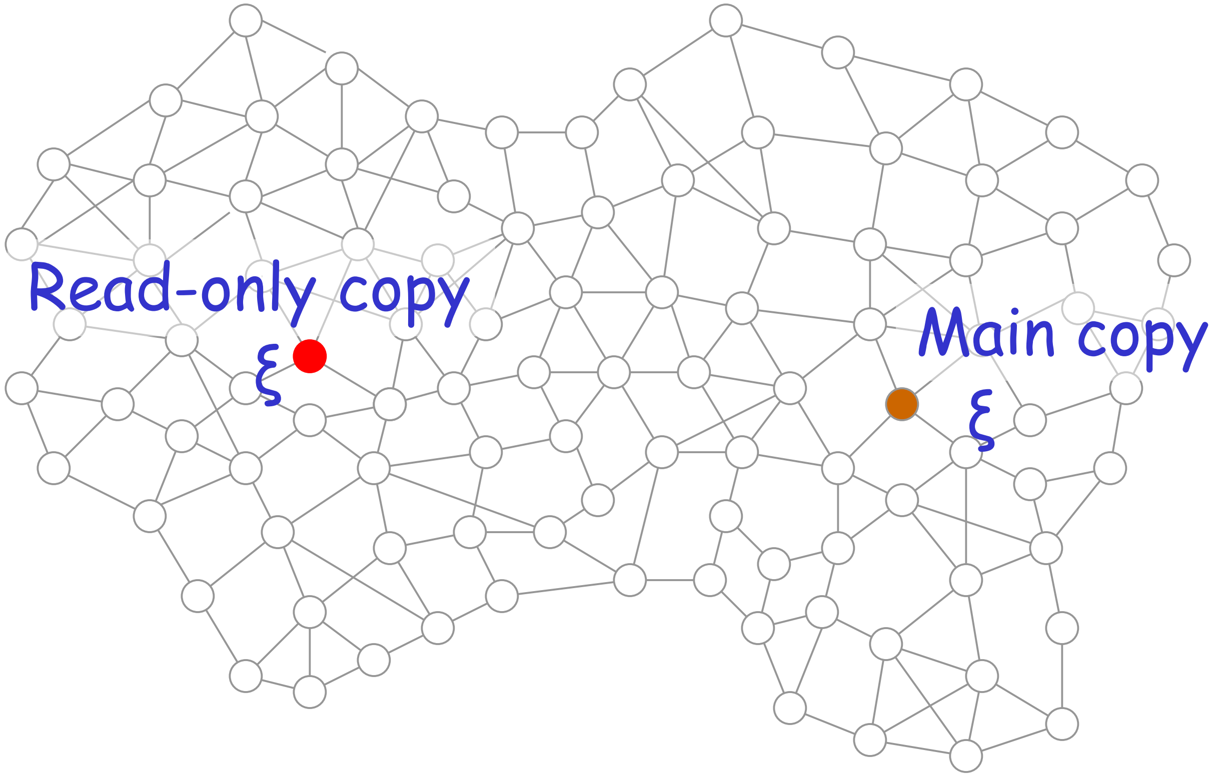
- Transactions run on network nodes
- They ask for shared objects distributed over the network for either read or write
- They appear to execute *atomically*
- The reads and writes on shared objects are supported through three operations:
  - Publish
  - Lookup
  - Move

Suppose the object ξ is at node ● and ● is a requesting node



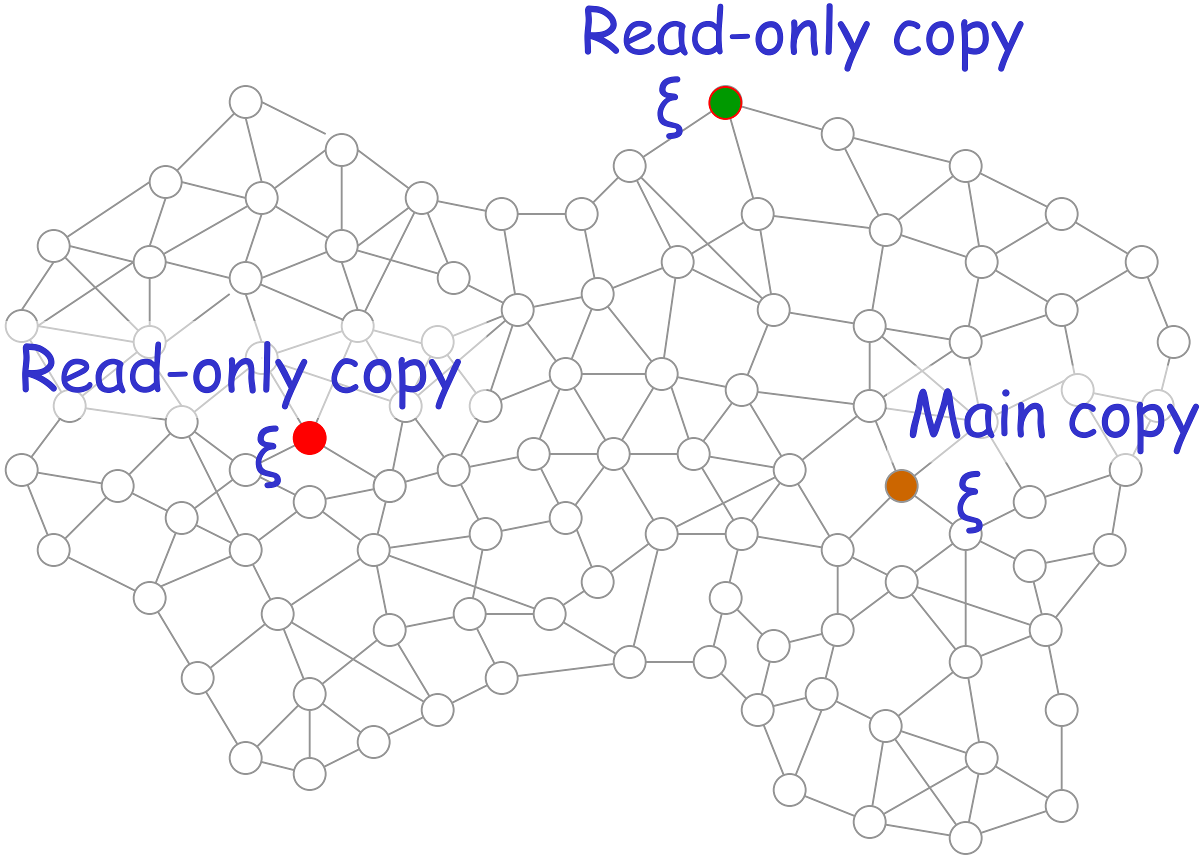
Suppose transactions are immobile and the objects are mobile

Lookup operation



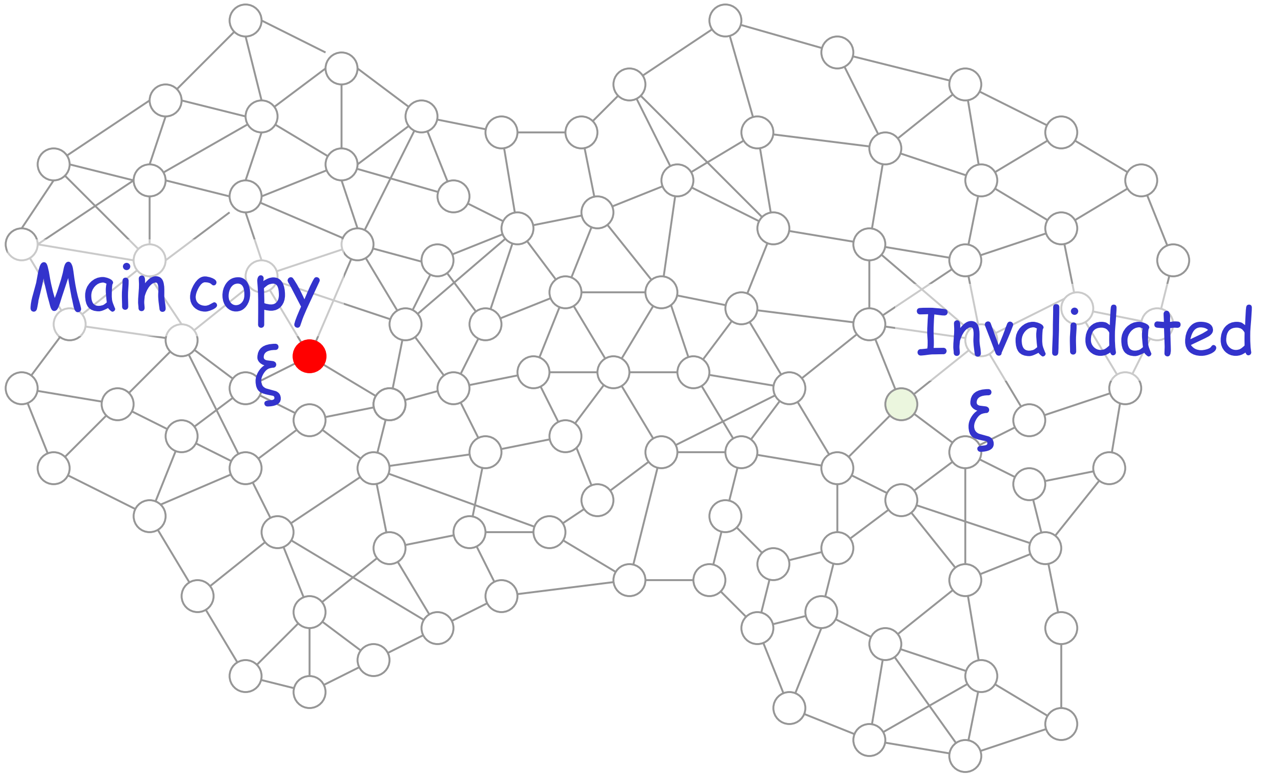
Replicates the object to the requesting node

Lookup operation



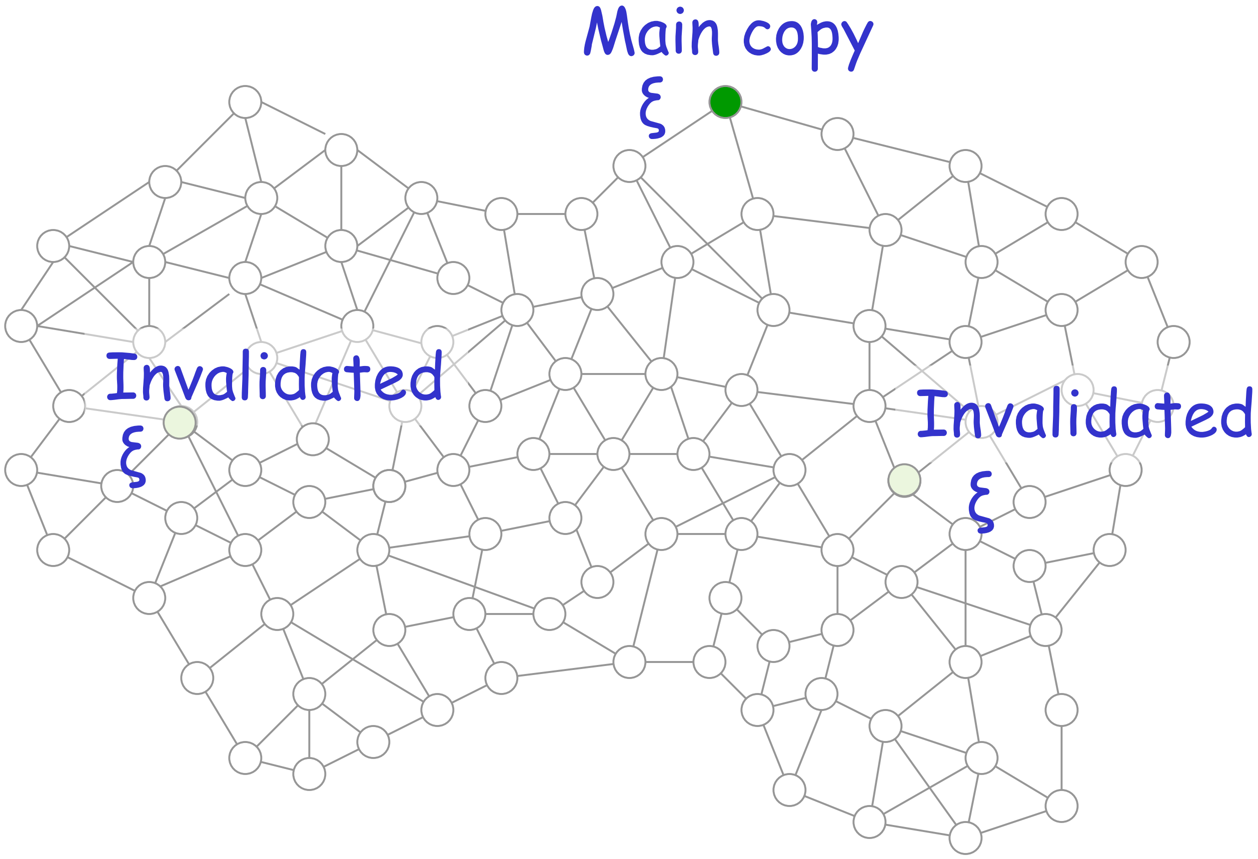
Replicates the object to the requesting nodes

Move operation



Relocates the object explicitly to the requesting node

Move operation



Relocates the object explicitly to the requesting node

Related Work

Protocol	Stretch	Network Kind	Runs on
Arrow [DISC'98]	$O(S_{ST})=O(D)$	General	Spanning tree
Relay [OPODIS'09]	$O(S_{ST})=O(D)$	General	Spanning tree
Combine [SSS'10]	$O(S_{OT})=O(D)$	General	Overlay tree
Ballistic [DISC'05]	$O(\log D)$	Constant-doubling dimension	Hierarchical directory with independent sets
Spiral [IPDPS'12]	$O(\log^2 n \log D)$	General	Hierarchical directory with sparse covers

- D is the diameter of the network kind
- S_* is the stretch of the tree used

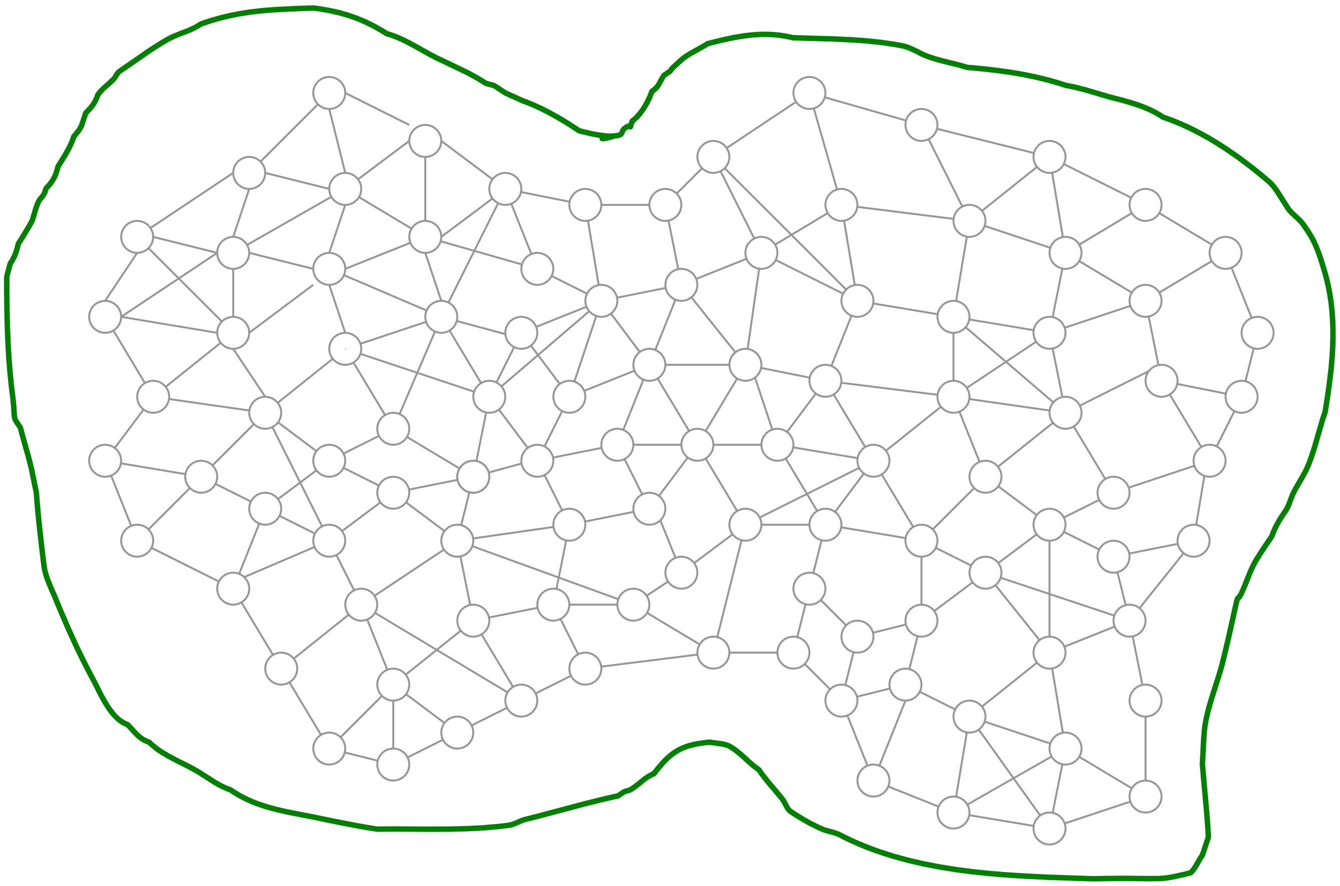
Inspiration

Concurrent online tracking of mobile users (1991)

Awerbuch, B., Peleg, D.

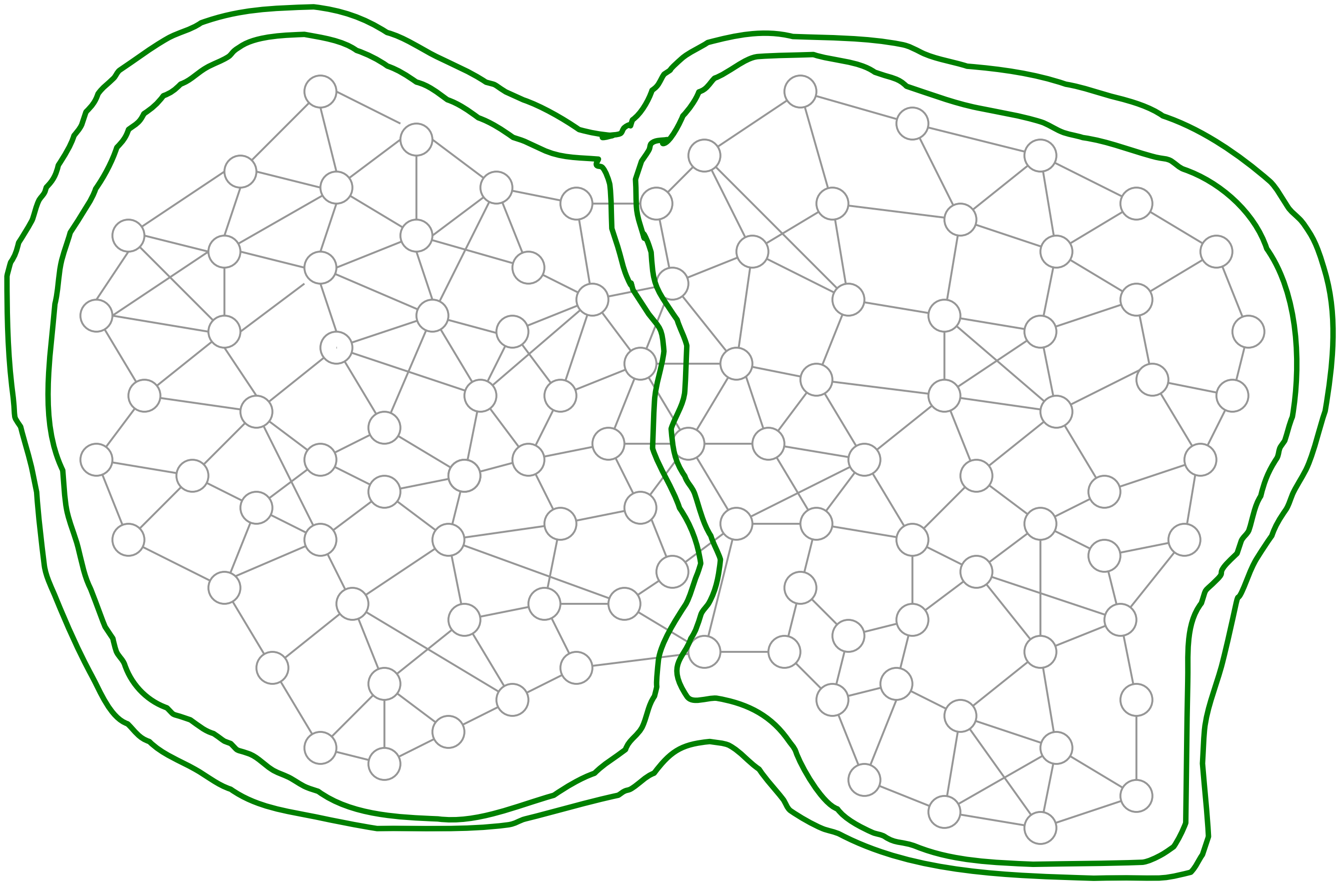
- A distributed directory scheme to minimize cost of moving objects
  - Total communication cost is proportional to the distances of positions of moving objects
- Uses a hierarchical clustering of the network
  - sparse partitions

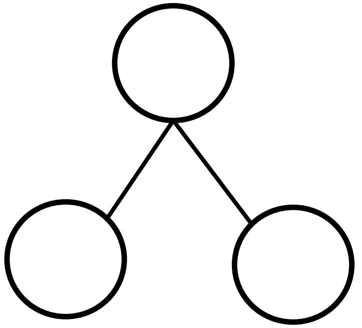
Spiral Approach: Hierarchical clustering



Network graph

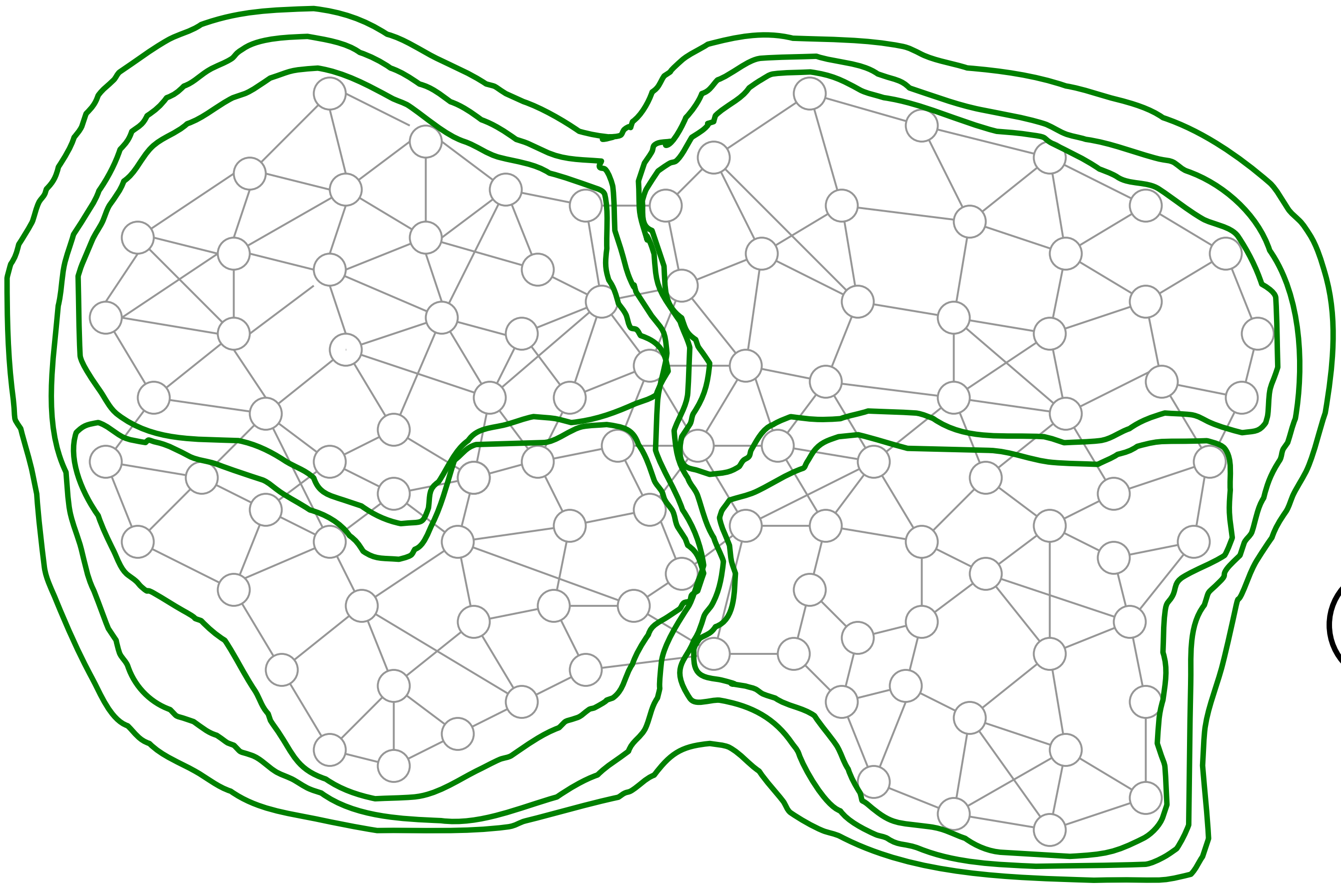
Spiral Approach: Hierarchical clustering

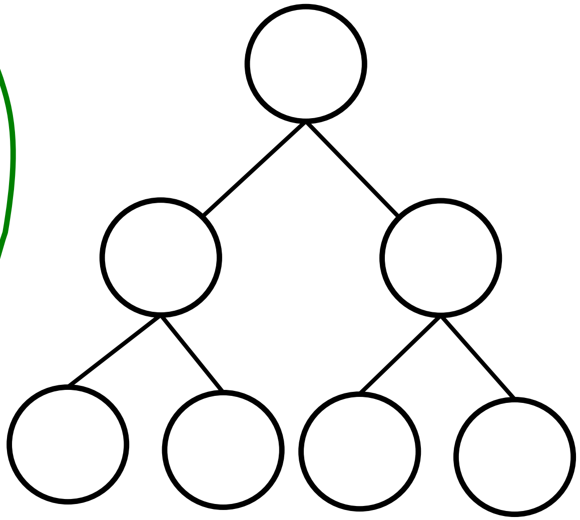




Alternative representation as a hierarchy tree with leader nodes

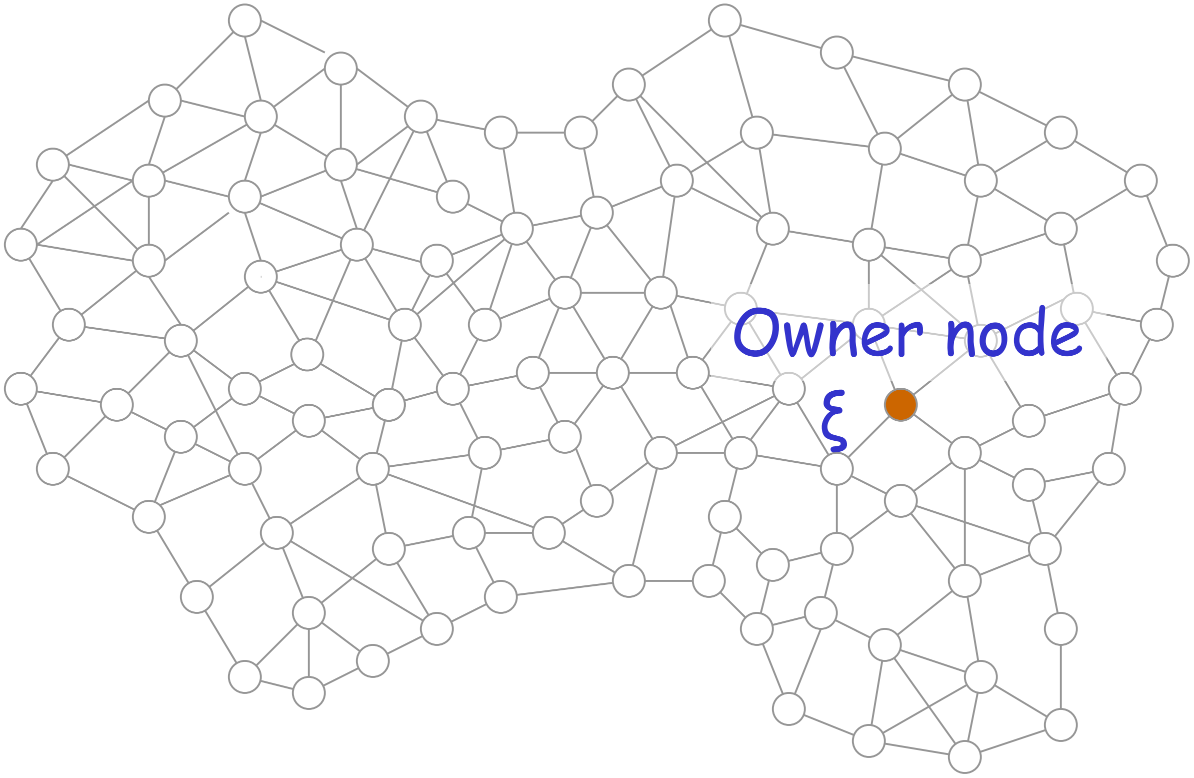
At the lowest level (level 0) every node is a cluster

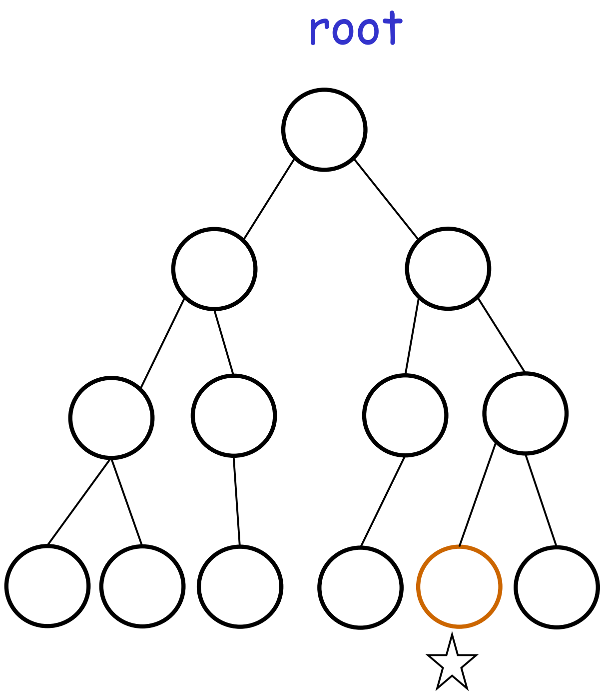




Directories at each level cluster, downward pointer if object locality known

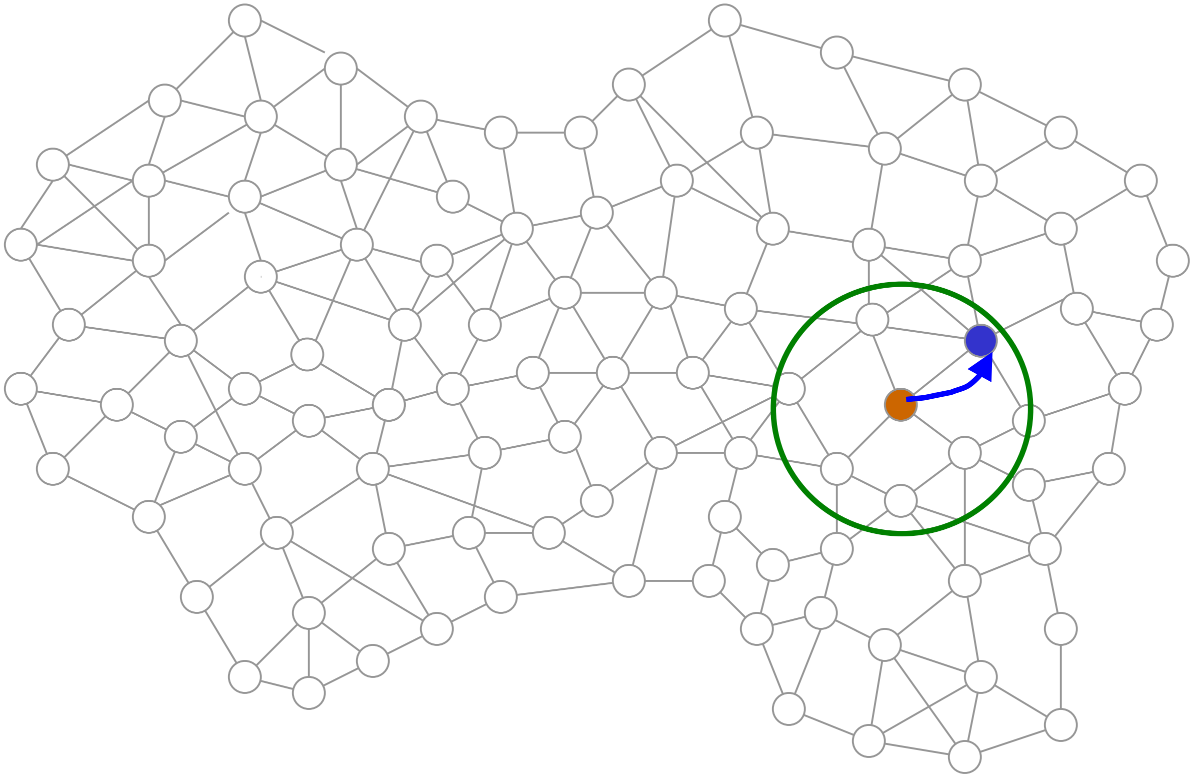
A Publish operation

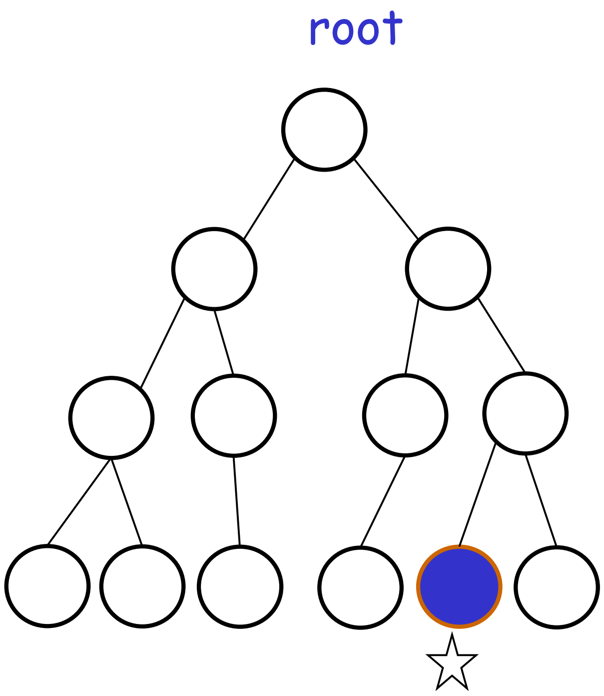




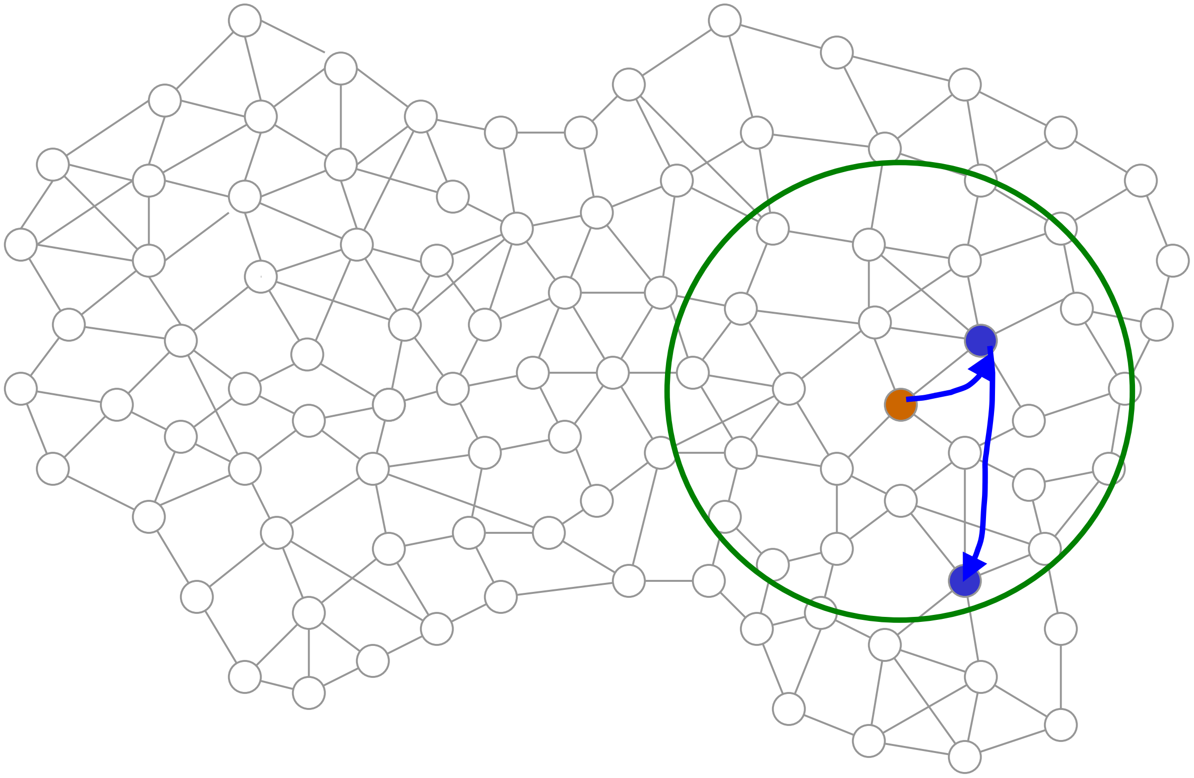
- Assume that ● is the creator of ξ which invokes the Publish operation
- Nodes know their parent in the hierarchy

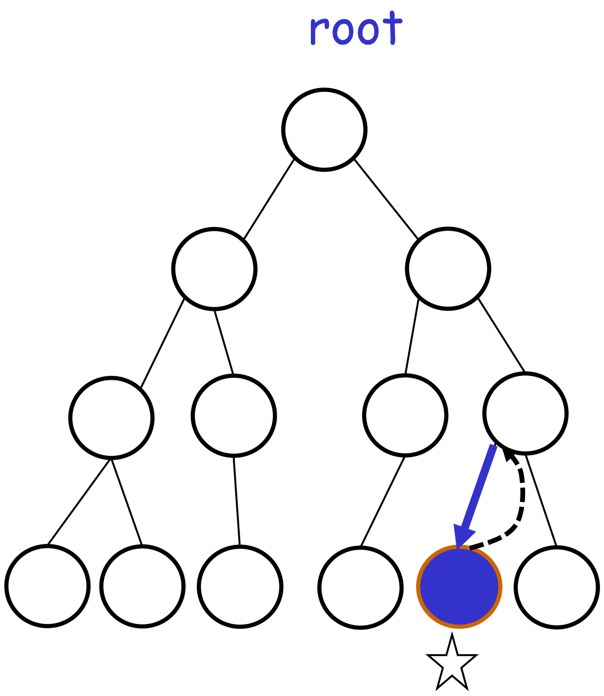
Send request to the leader





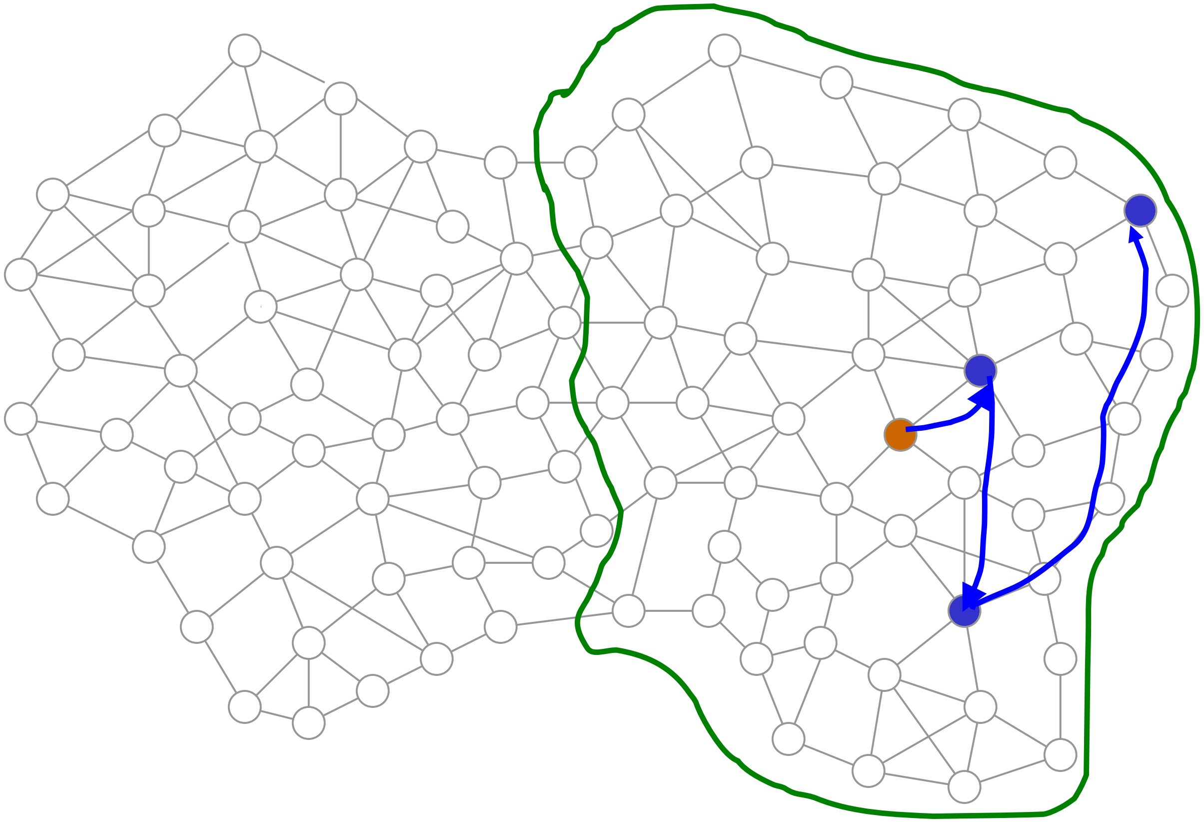
Continue up phase

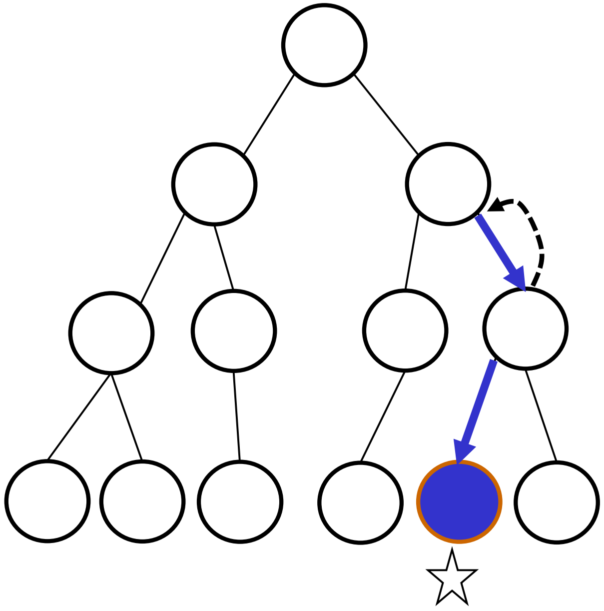




Sets downward pointer while going up

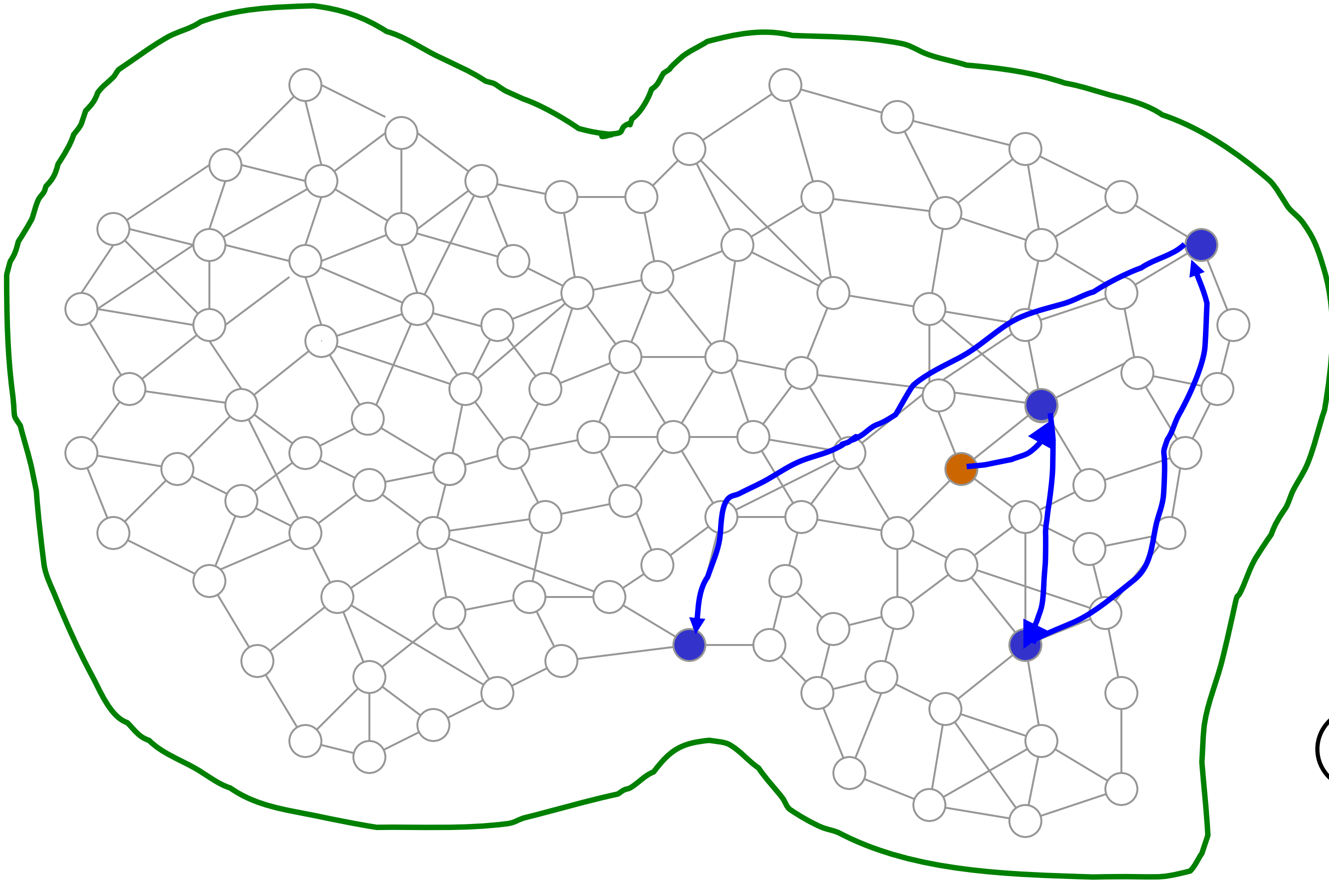
Continue up phase

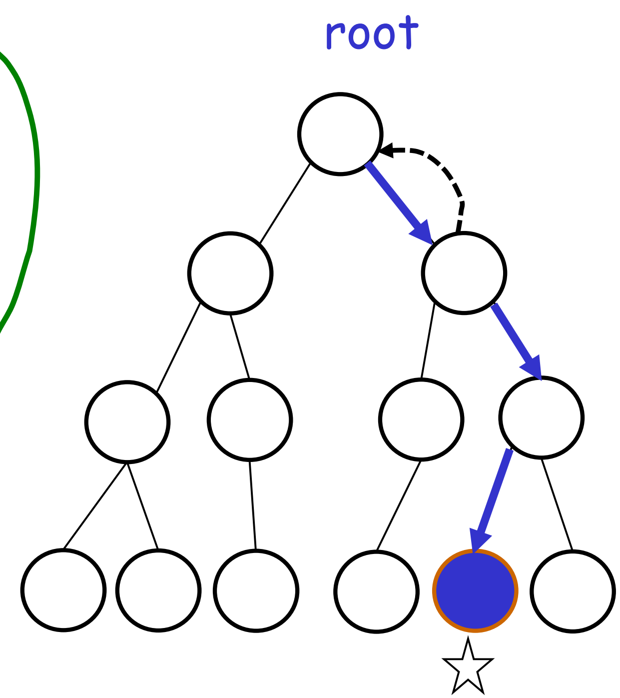


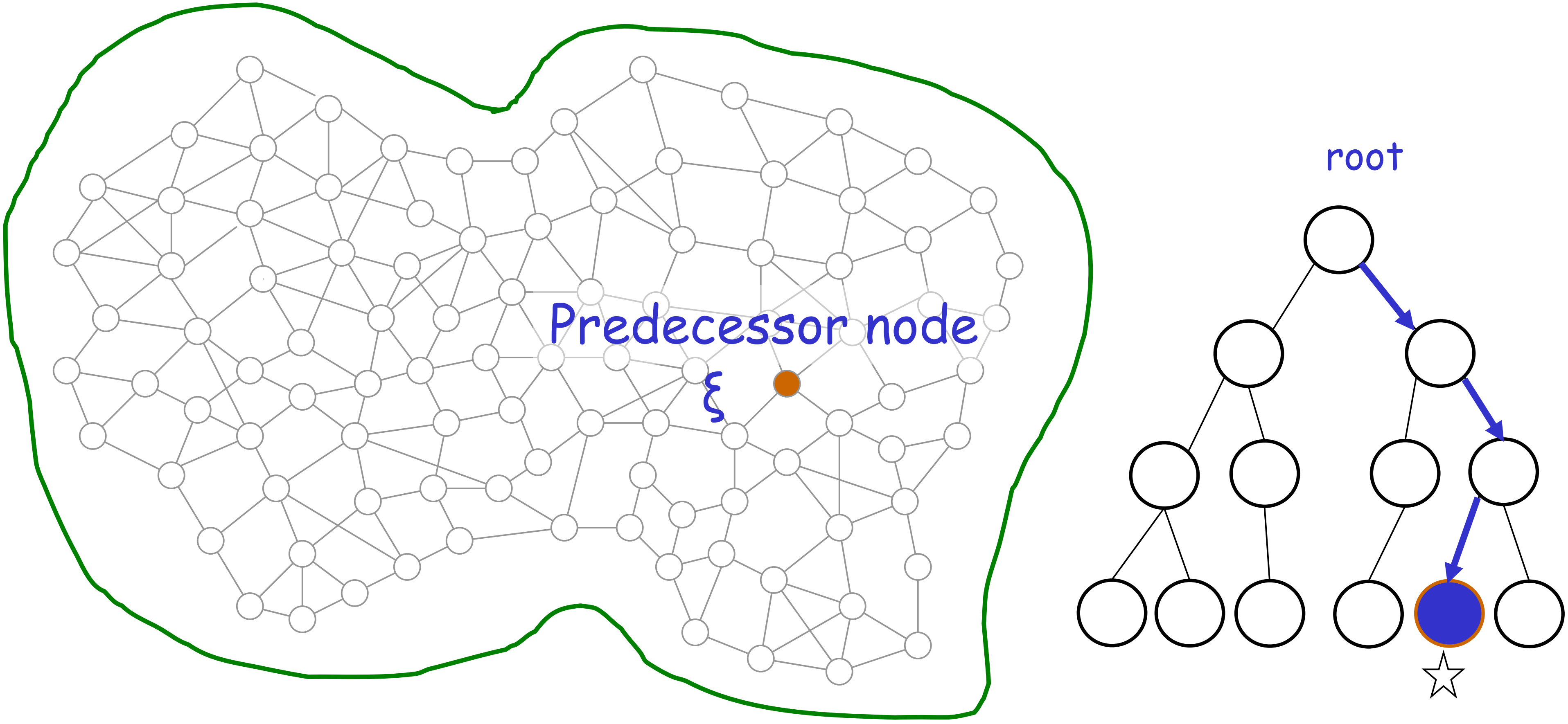


Sets downward pointer while going up

Root node found, stop up phase

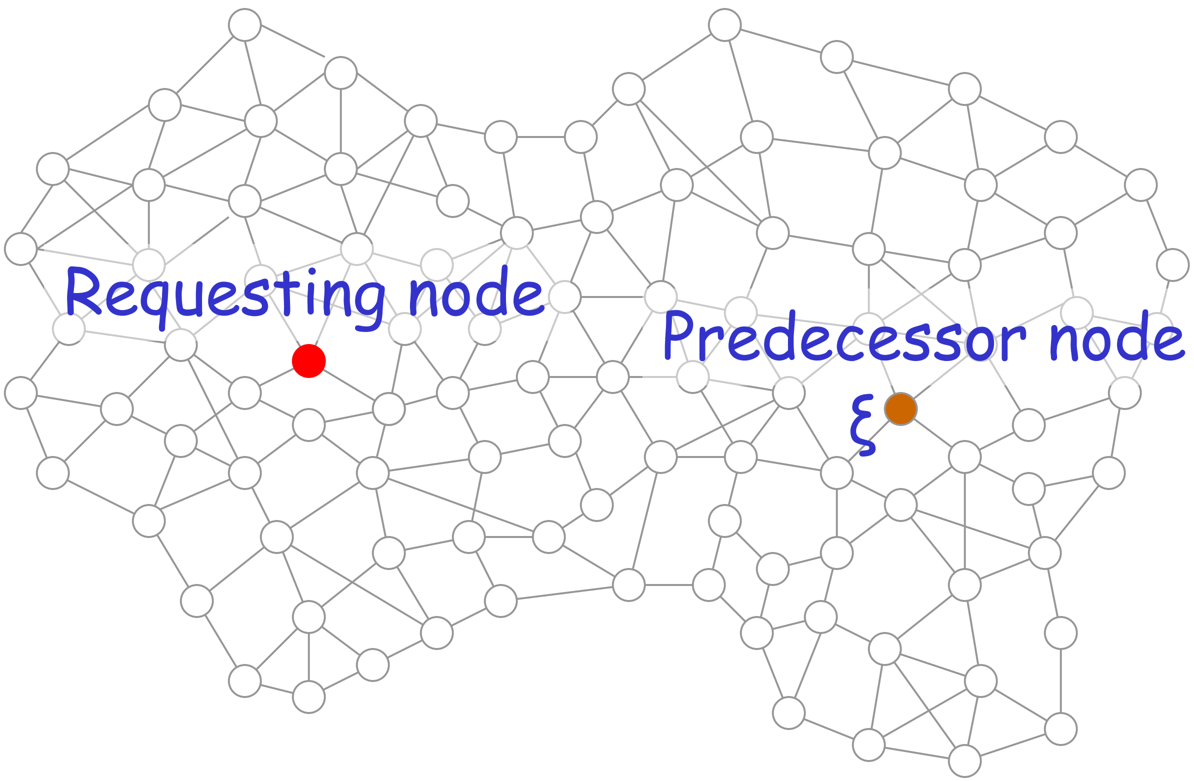


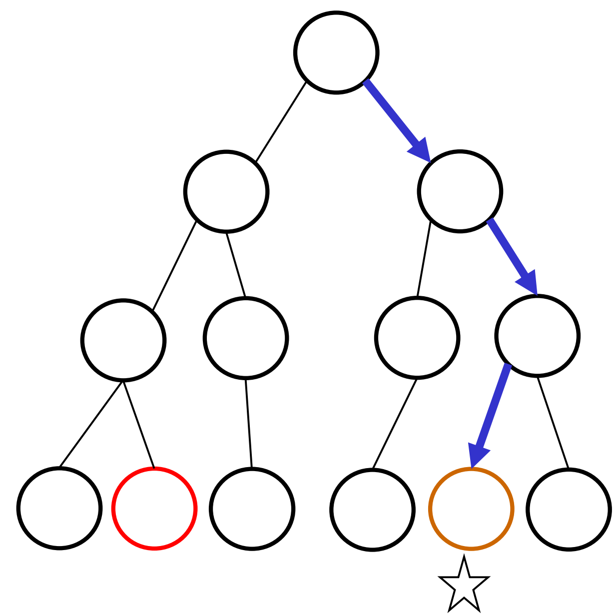




A successful Publish operation

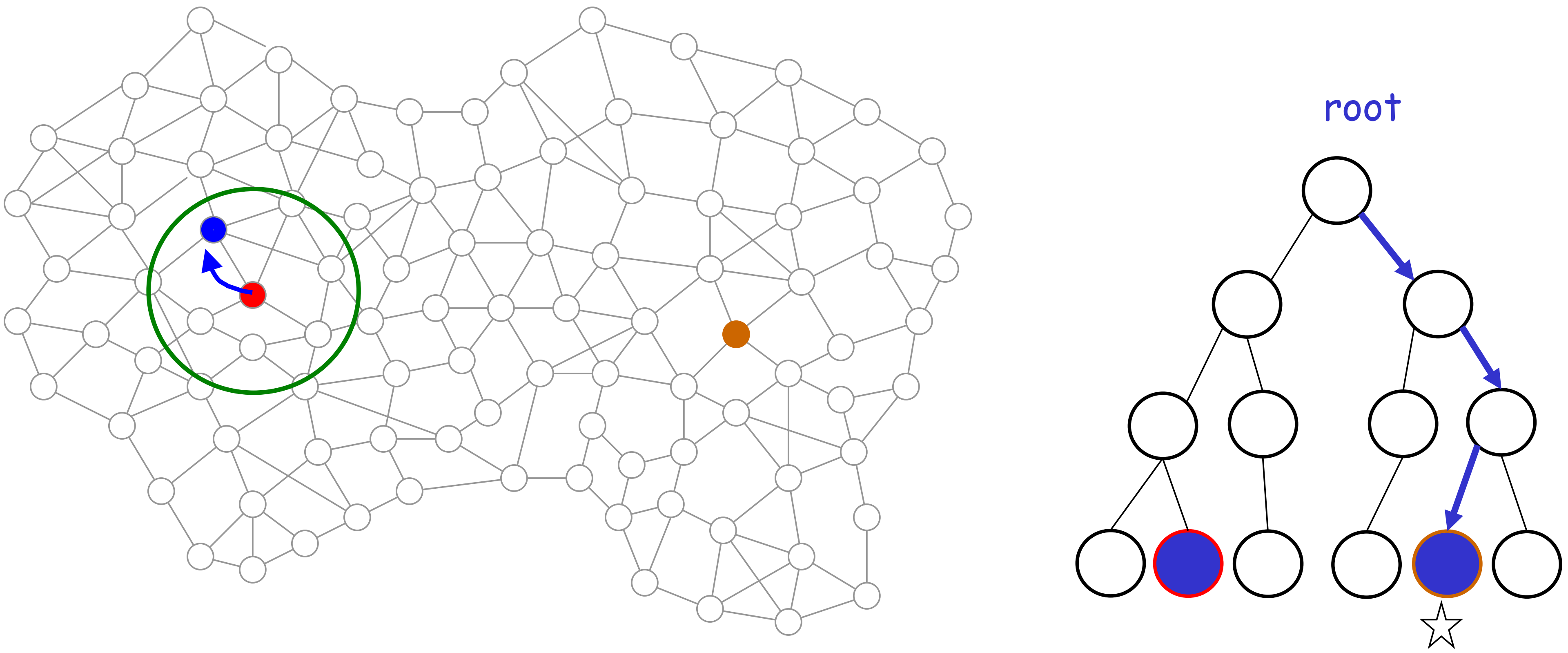
Supporting a Move operation



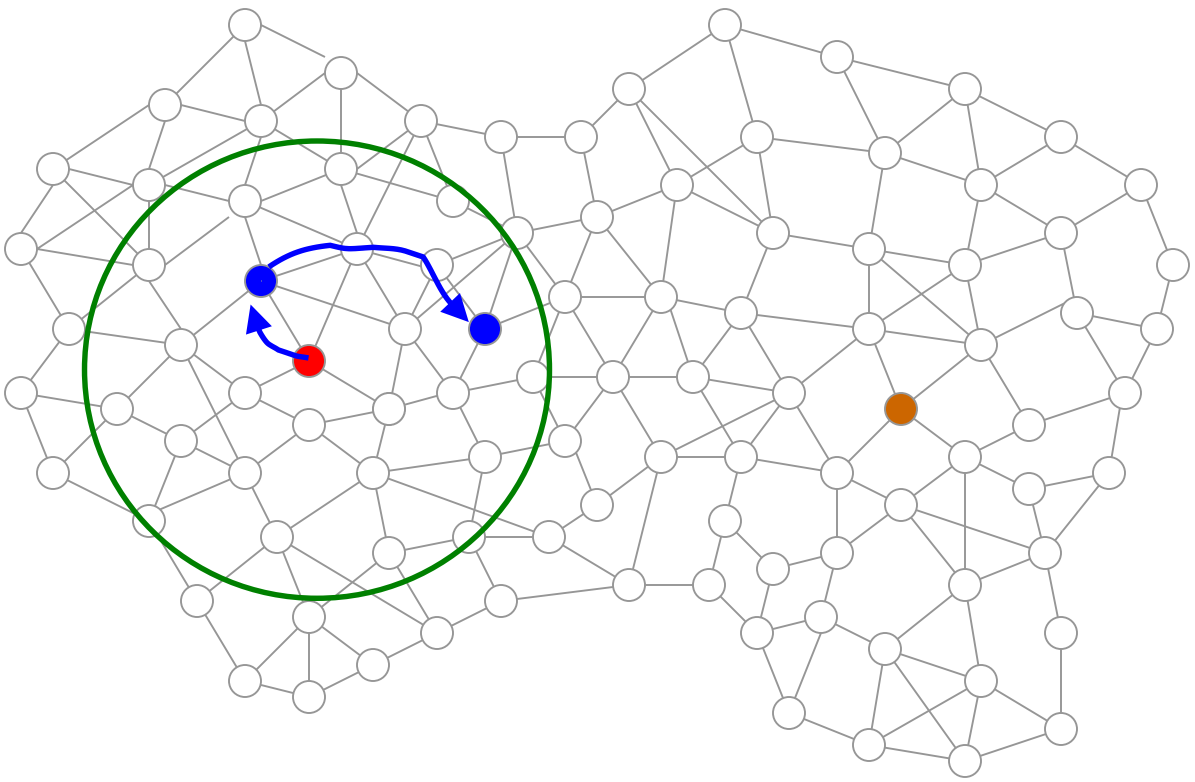


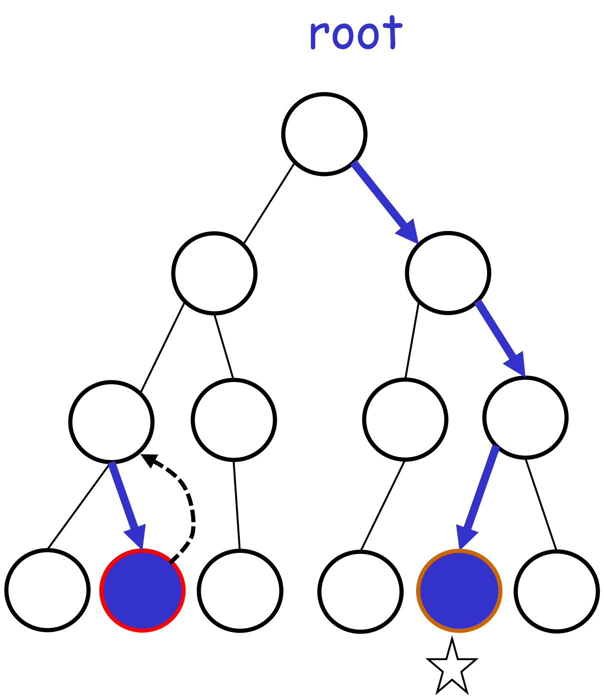
- Initially, nodes point downward to object owner (predecessor node) due to Publish operation
- Nodes know their parent in the hierarchy

Send request to leader node of the cluster upward in hierarchy



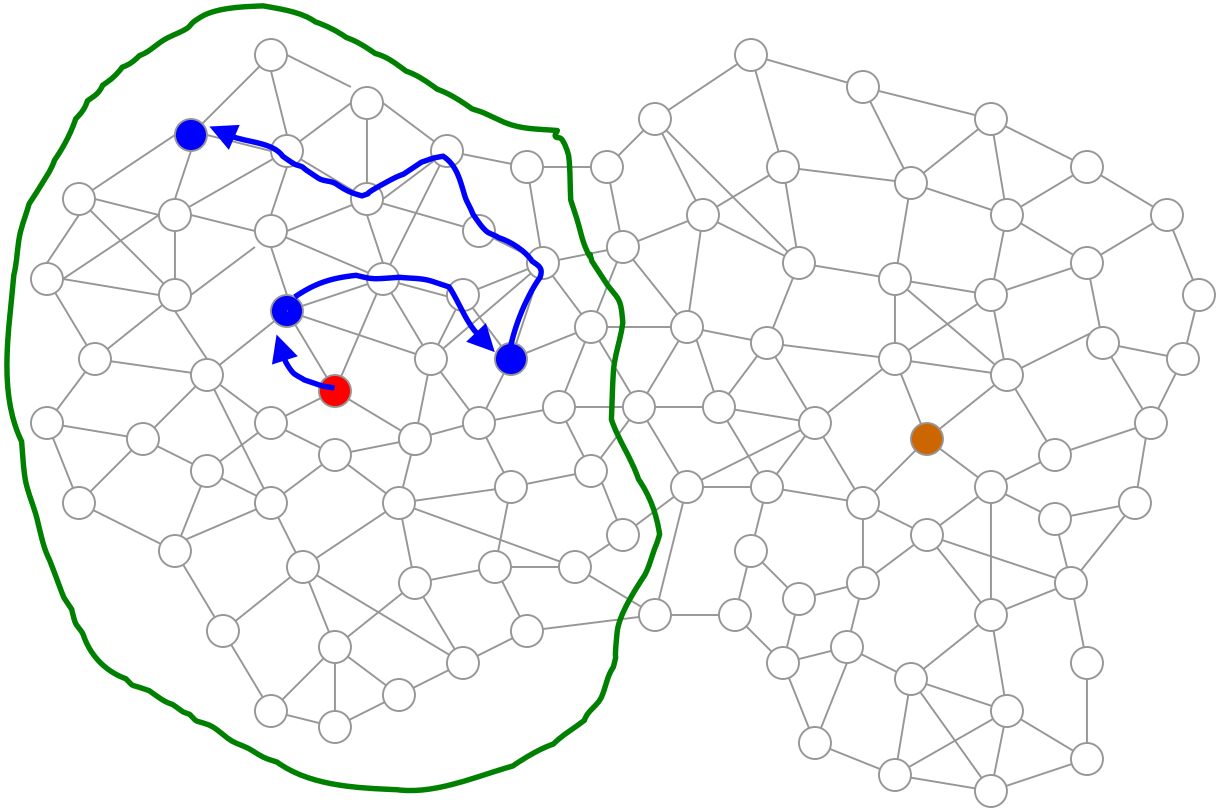
Continue up phase until downward pointer found

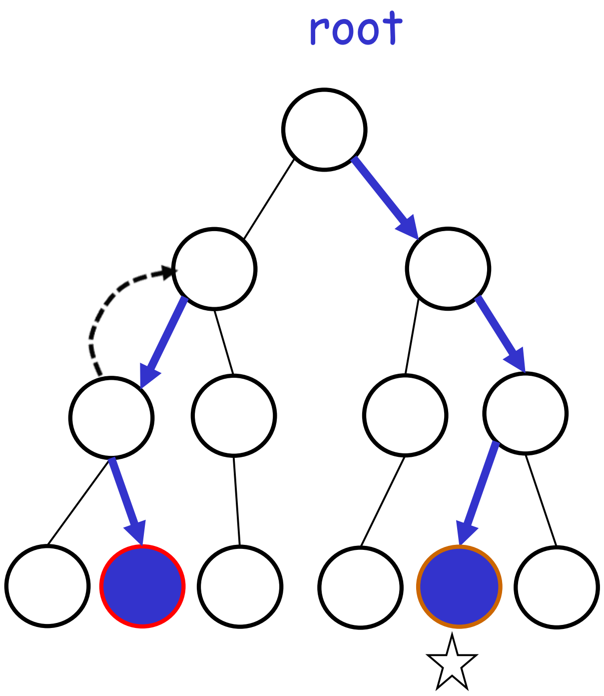




Sets downward path while going up

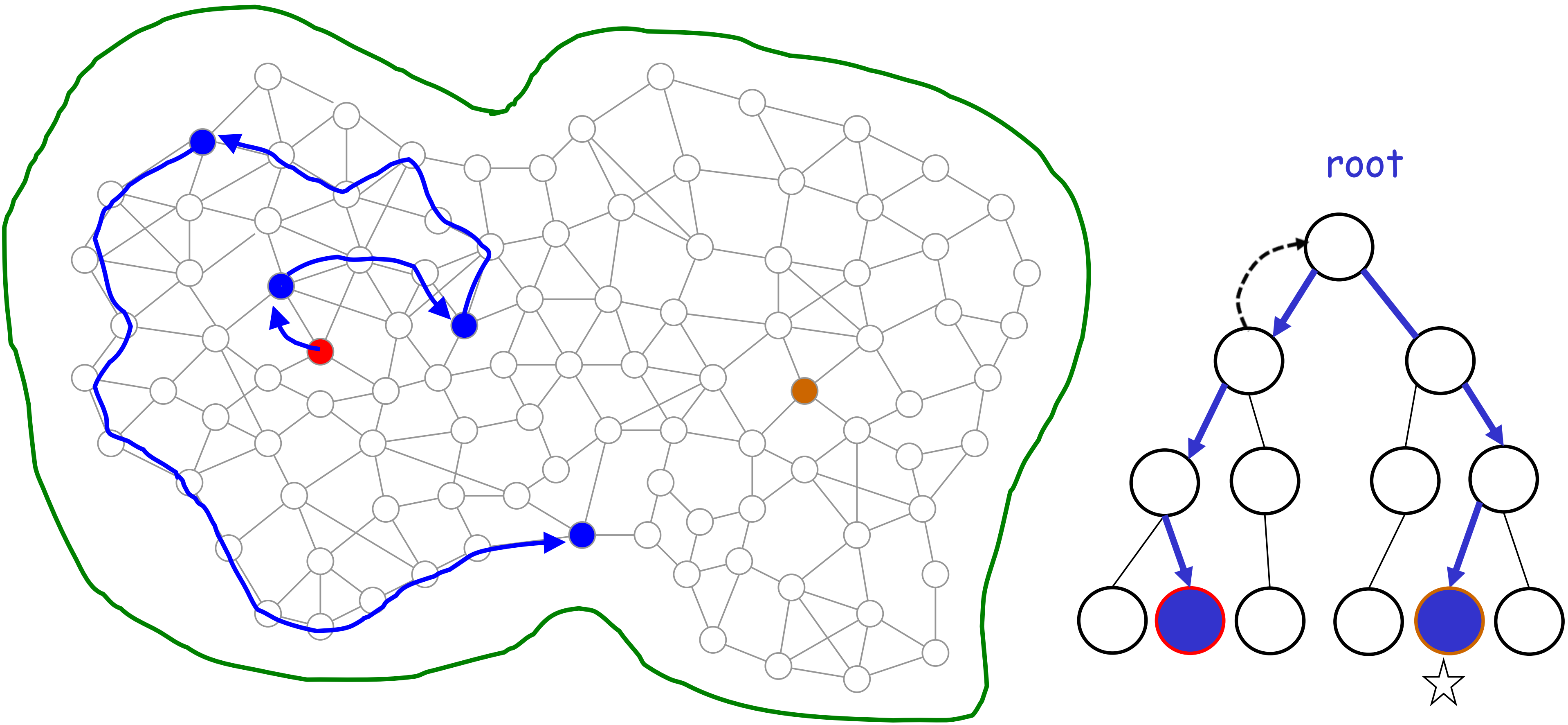
Continue up phase





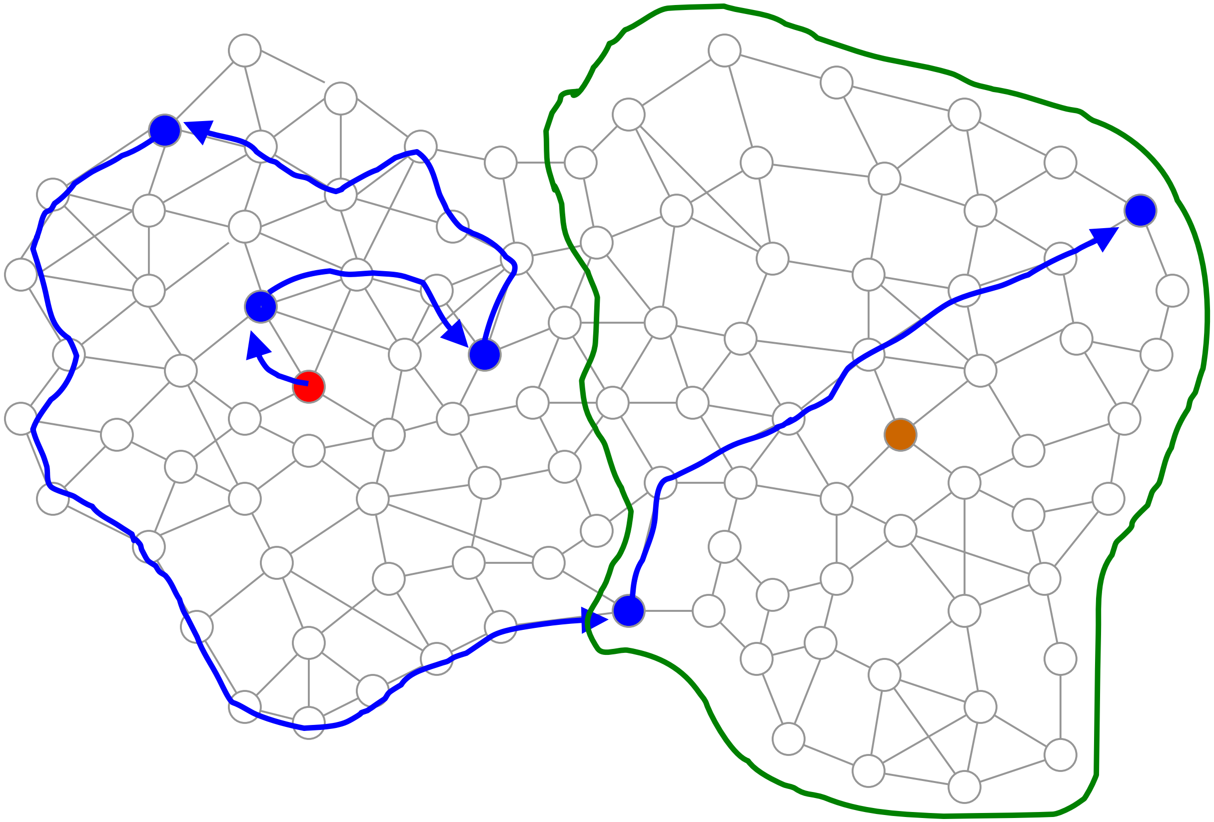
Sets downward path while going up

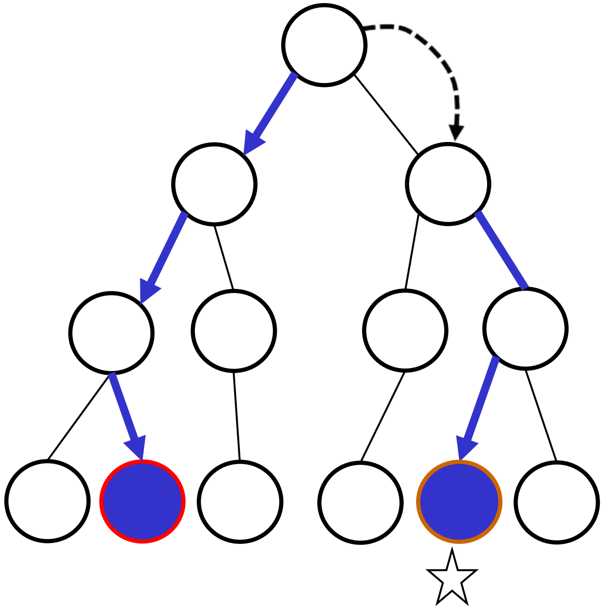
Continue up phase



Sets downward path while going up

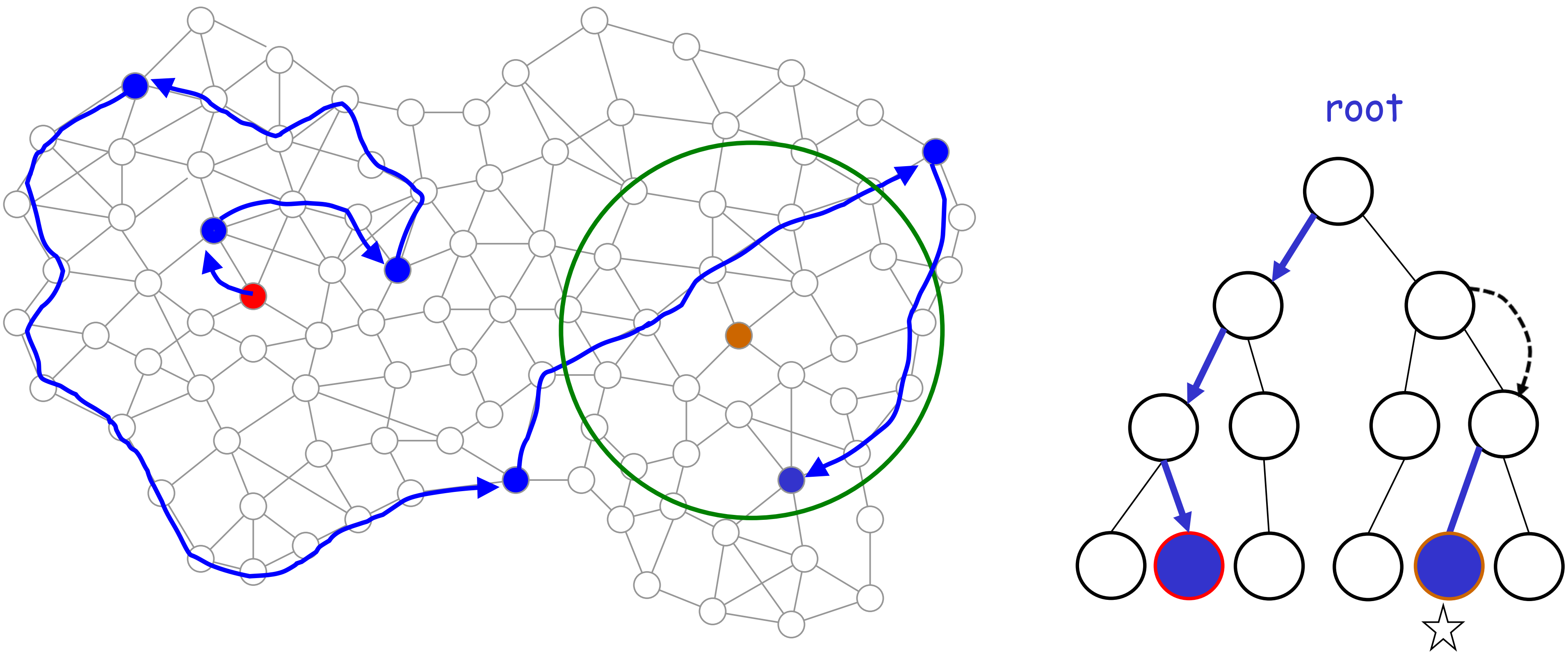
Downward pointer found, start down phase





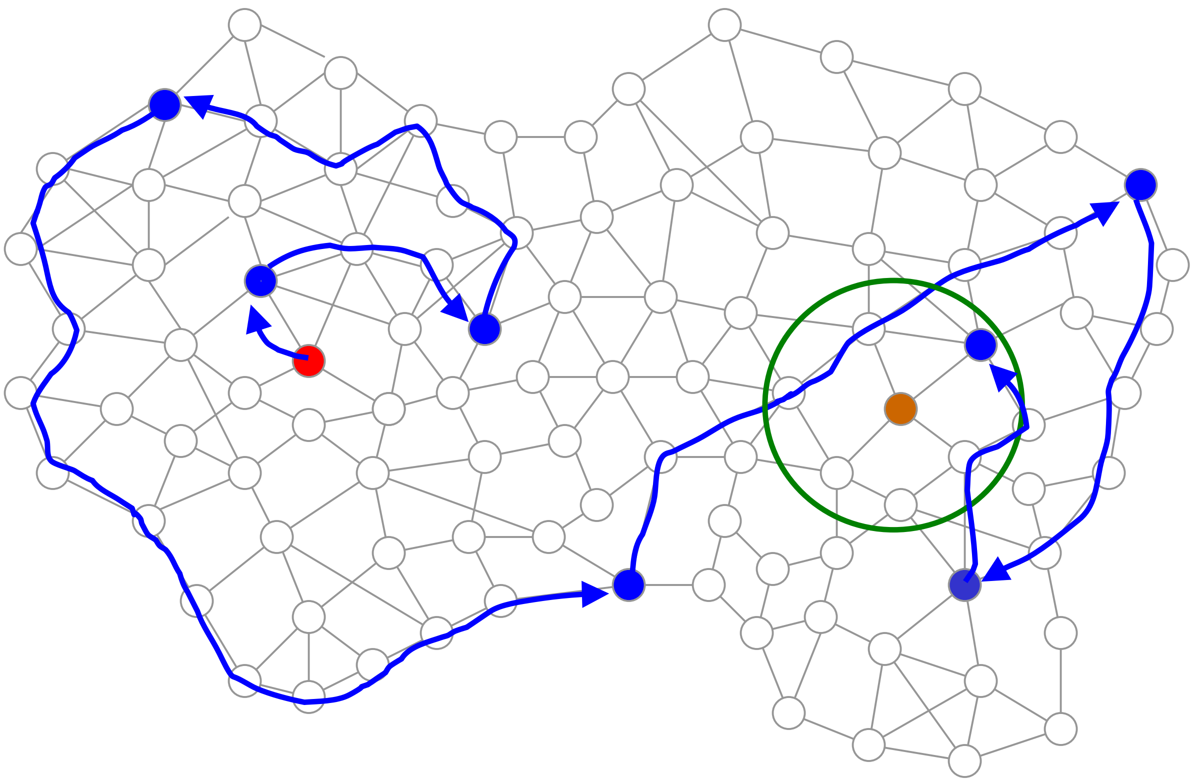
Discards path while going down

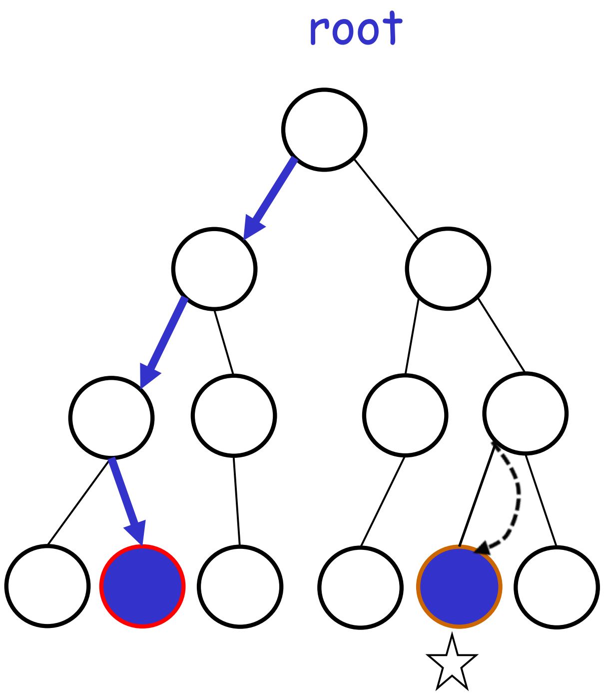
Continue down phase



Discards path while going down

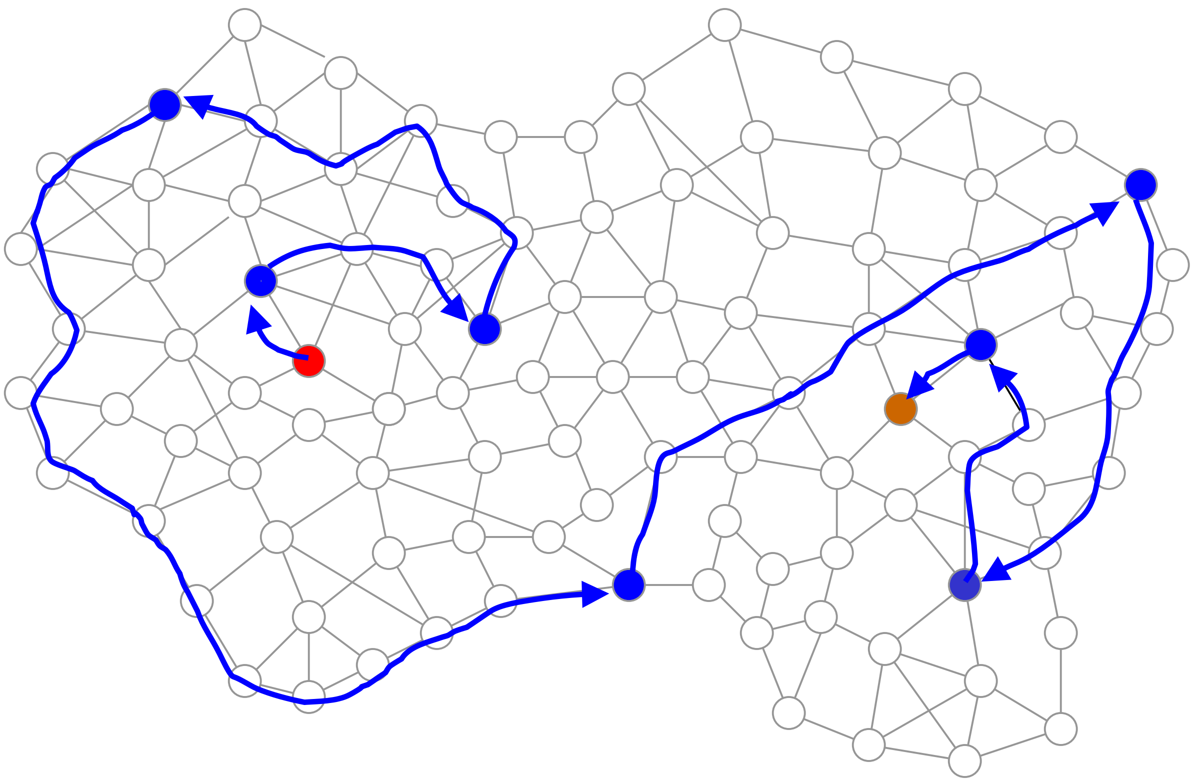
Continue down phase

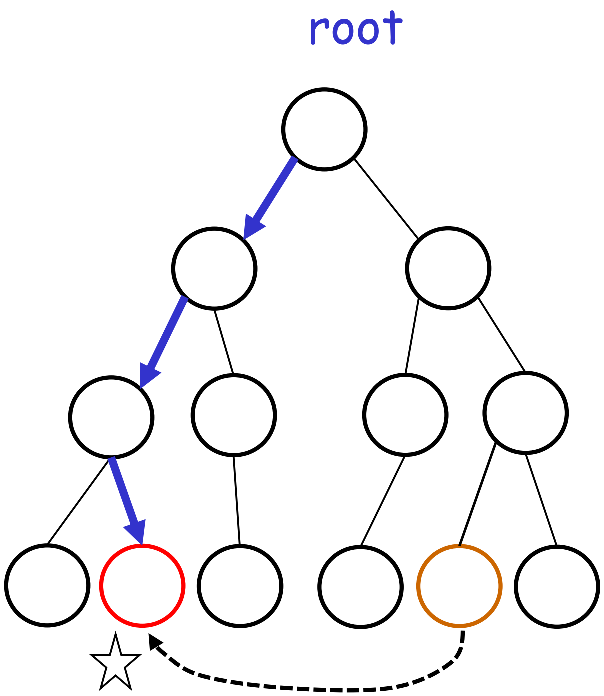




Discards path while going down

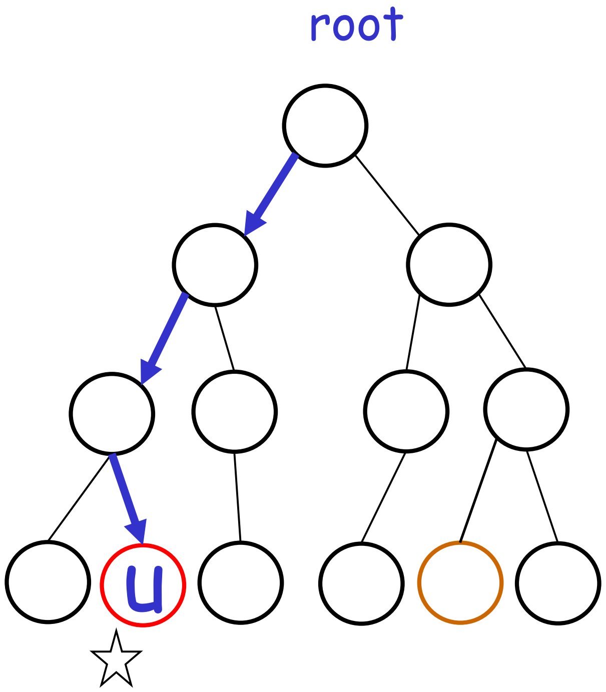
Predecessor reached, object is moved from node ● to node ●

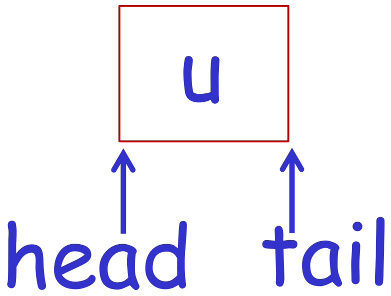




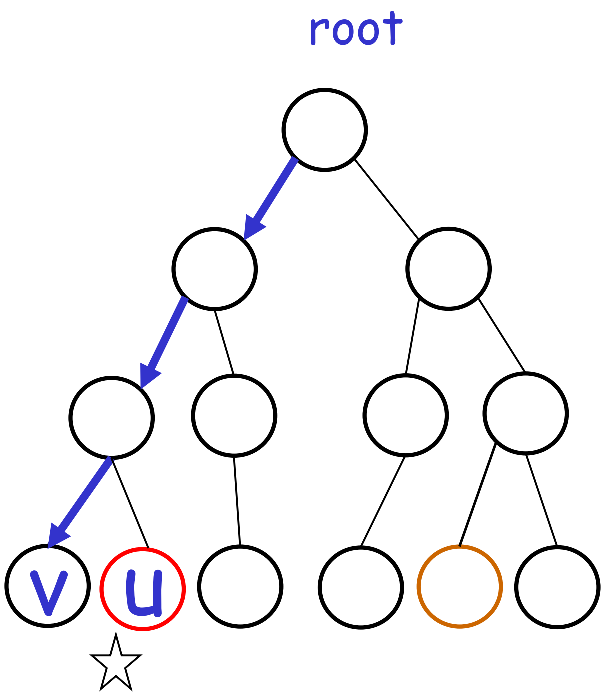
Lookup is similar without change in the directory structure and only a read-only copy of the object is sent

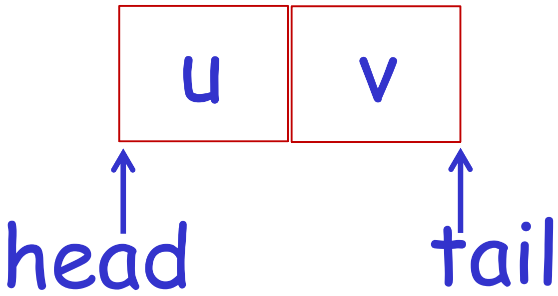
Distributed Queue



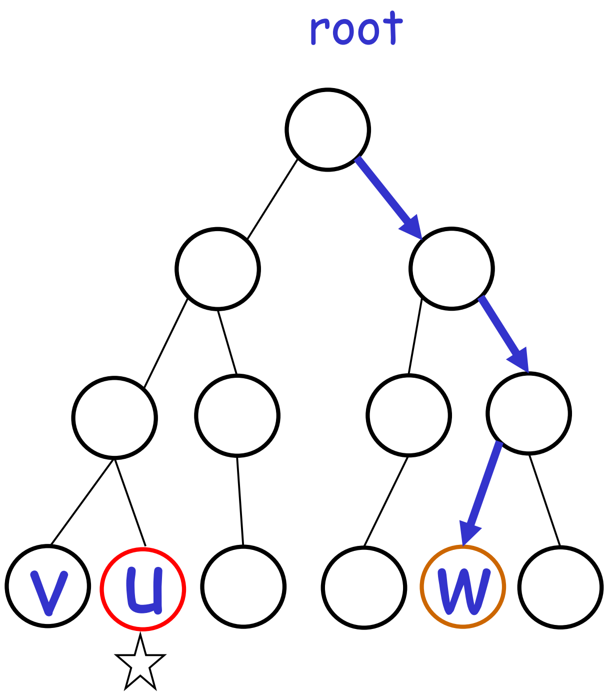


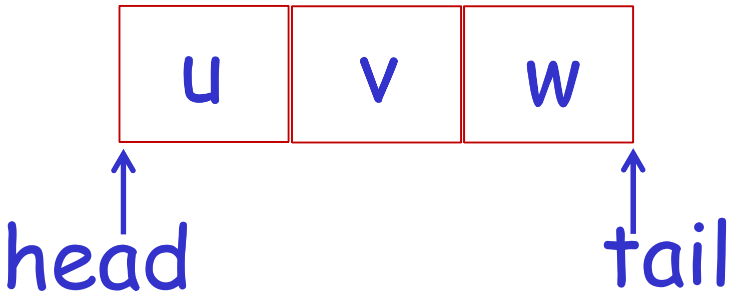
Distributed Queue



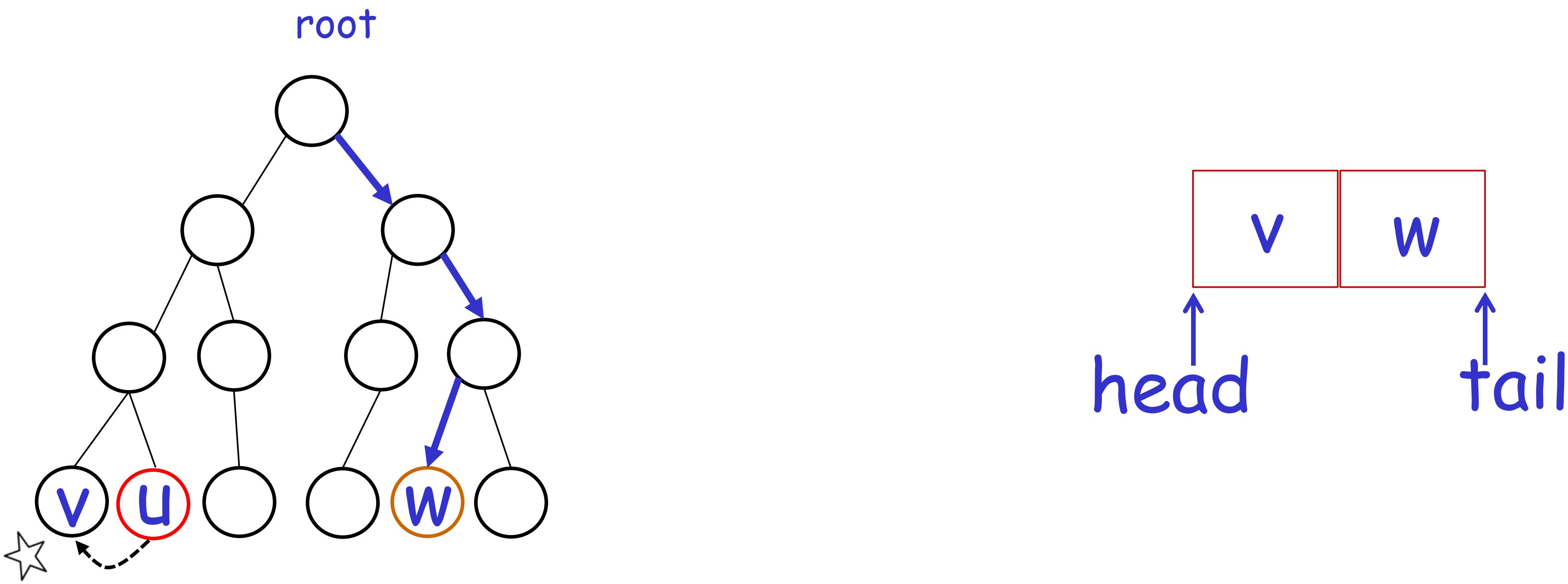


Distributed Queue

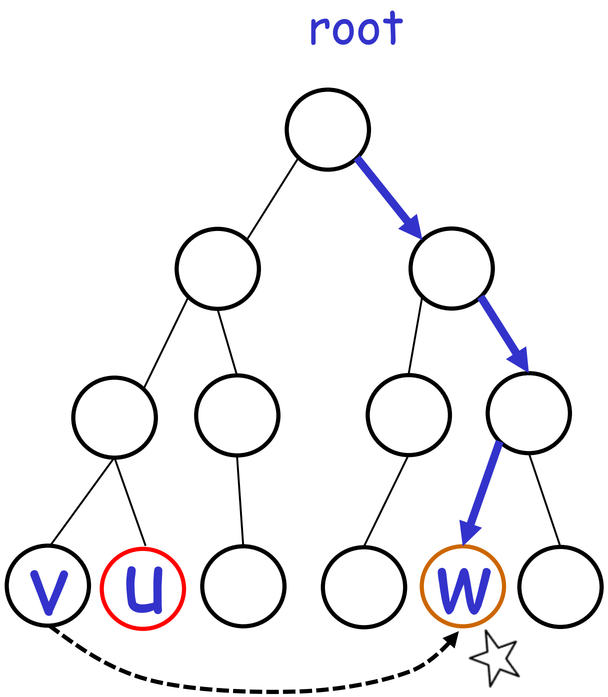


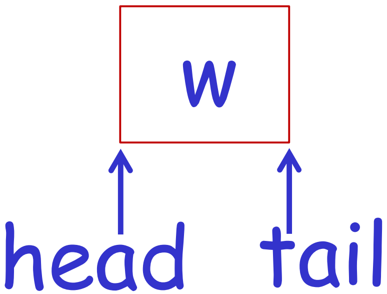


Distributed Queue



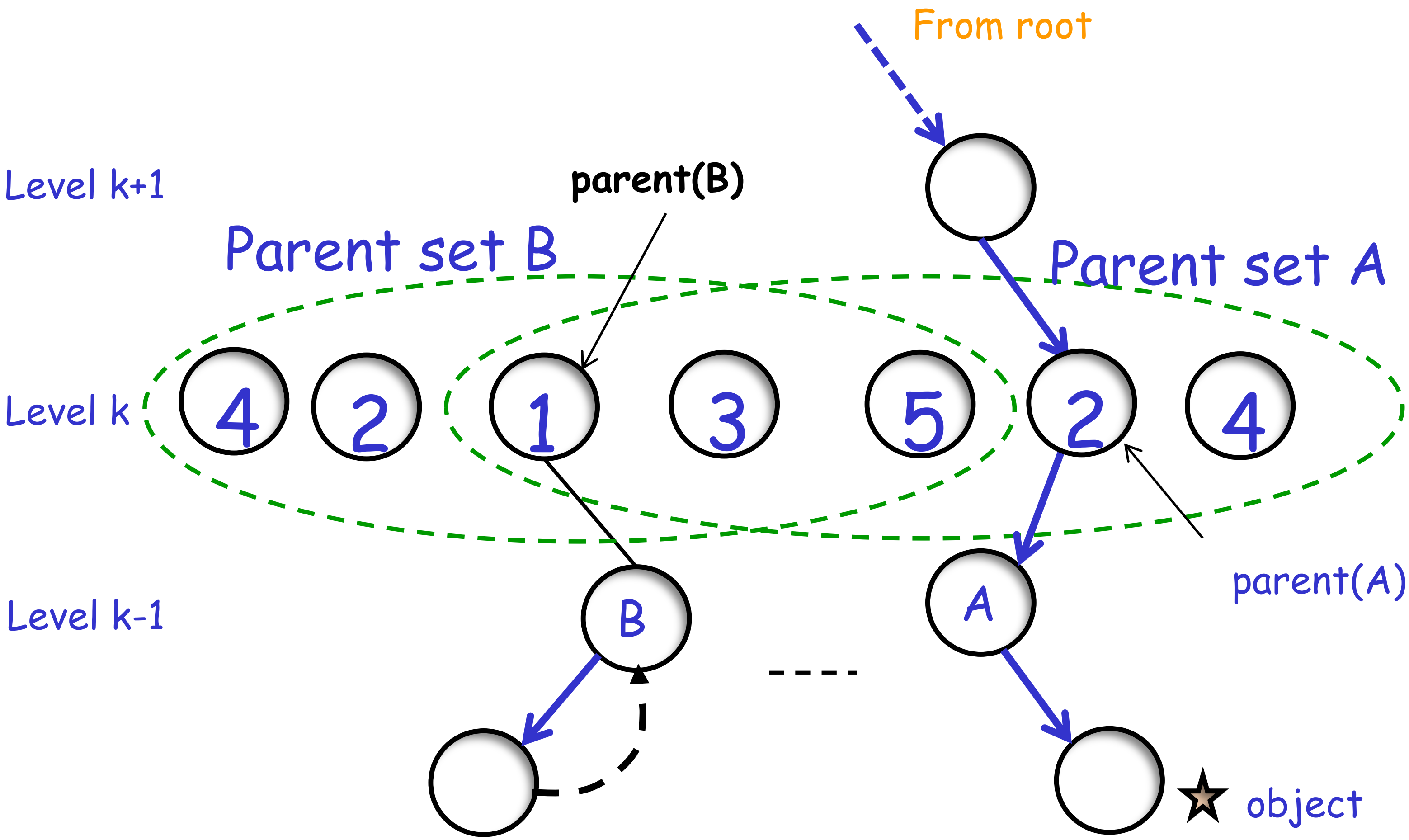
Distributed Queue





Spiral avoids deadlocks

Label all the parents in each level and visit them in the order of the labels.



Spiral Hierarchy

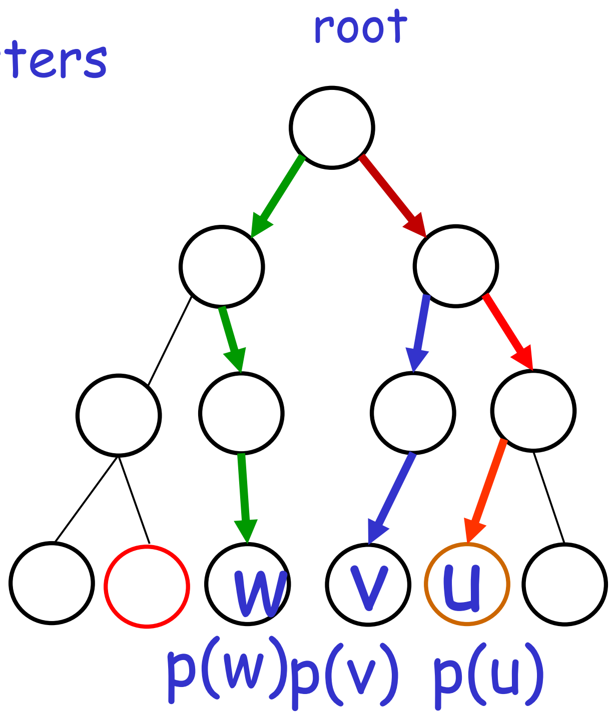
Cluster Diameter stretch

Cluster Overlaps

- (O(log n), O(log n))-sparse cover hierarchy constructed from O(log n) levels of hierarchical partitions
  - Level 0, each node belongs to exactly one cluster
  - Level h, all the nodes belong to one cluster with root r
  - Level 0 < i < h, each node belongs to exactly O(log n) clusters which are labeled different

Spiral Hierarchy

- How to find a predecessor node?
  - Via spiral paths for each leaf node u by visiting parent leaders of all the clusters that contain u from level 0 to the root level

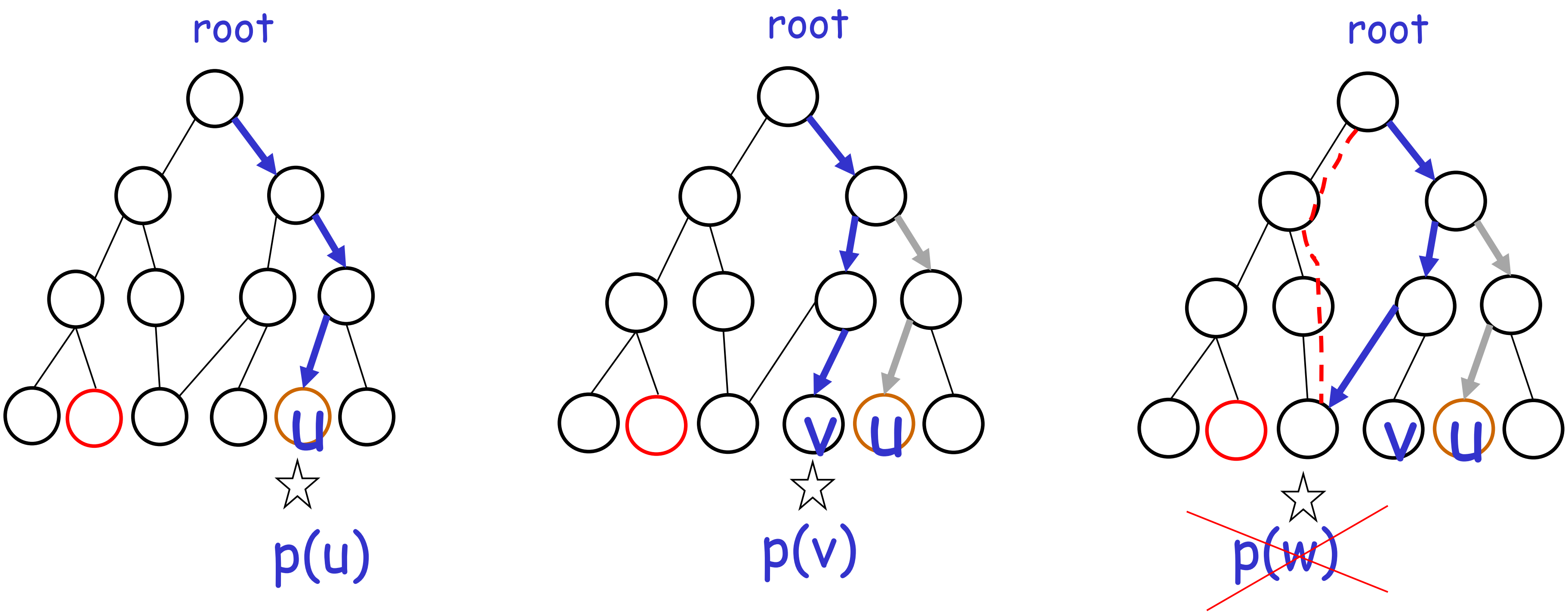


The hierarchy guarantees:

(1) For any two nodes u,v, their spiral paths p(u) and p(v) meet at level min{h, log(dist(u,v))+2}

(2) length(p_i(u)) is at most $O(2^i \log^2 n)$

Downward Paths

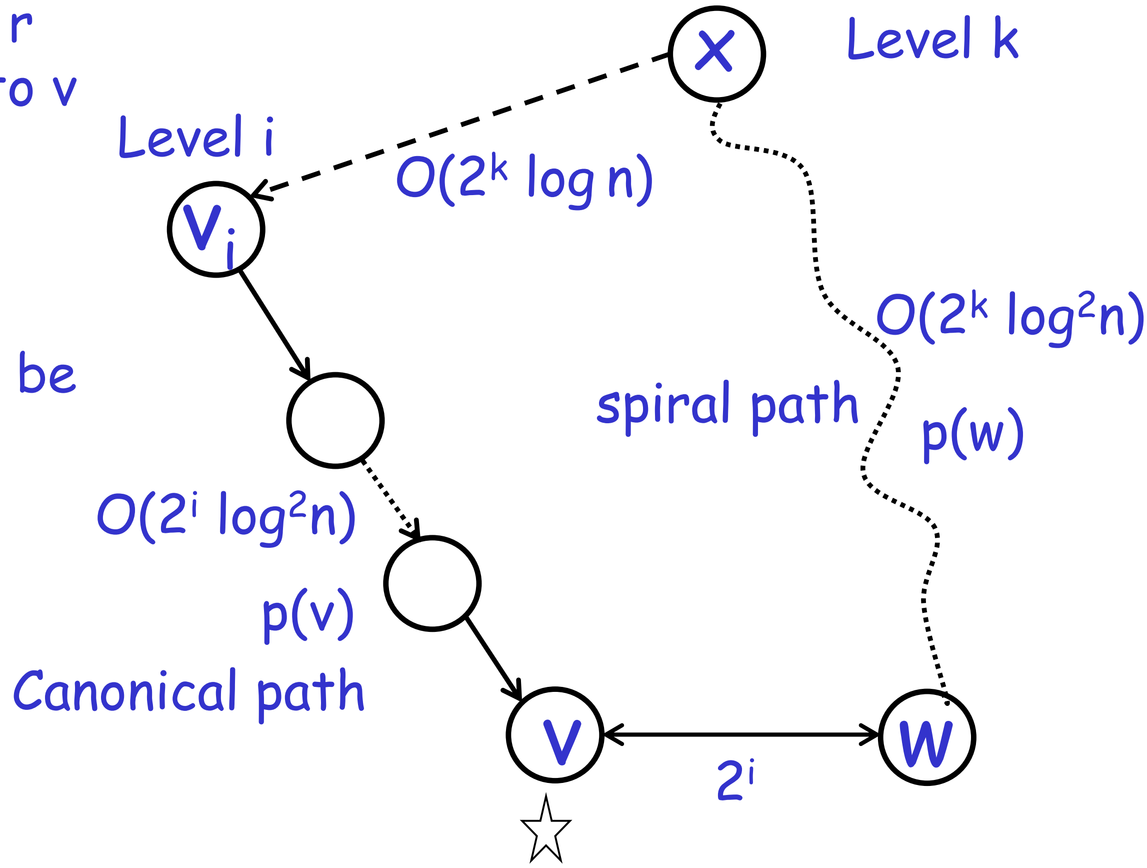


Deformation of spiral paths after moves

Analysis: lookup Stretch

If there is no Move, a Lookup r from w finds downward path to v in level log(dist(u,v))+2 = O(i)

When there are Moves, it can be shown that r finds downward path to v in level k = O(i + log $\log^2 n$)



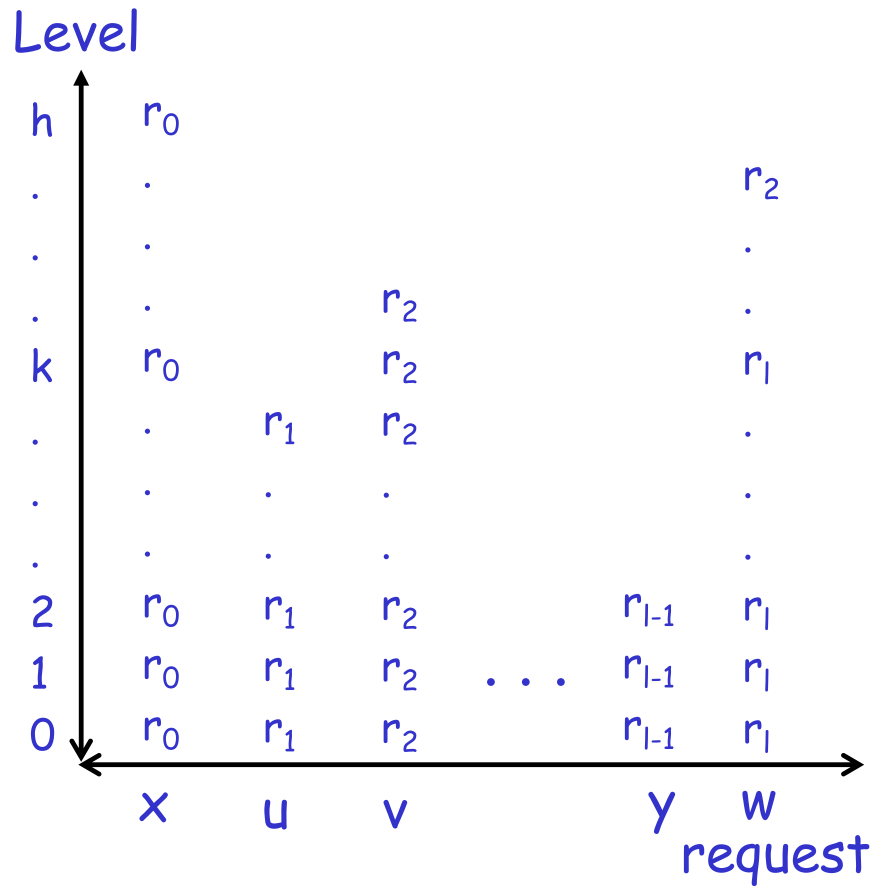
$$C(r)/C^*(r) = O(2^k \log^2 n) + O(2^k \log n) + O(2^i \log^2 n) / 2^{i-1}$$
$$= O(\log^4 n)$$

Analysis: move Stretch

Assume a sequential execution R of $l+1$ Move requests, where r_0 is an initial Publish request.

$C^*(R) \geq \max_{1 \leq k \leq h} (S_k - 1)\, 2^{k-1}$

$C(R) \geq \sum_{k=1}^{h} (S_k - 1)\, O(2^k \log^2 n)$



Thus,

$$C(R)/C^*(R) = \sum_{k=1}^{h} (S_k - 1)\, O(2^k \log^2 n) \,/\, \max_{1 \leq k \leq h} (S_k - 1)\, 2^{k-1}$$
$$= O(\log^2 n \cdot h) \max_{1 \leq k \leq h} (S_k - 1)\, 2^{k-1} \,/\, \max_{1 \leq k \leq h} (S_k - 1)\, 2^{k-1}$$
$$= O(\log^2 n \cdot \log D)$$

Presentation Outline

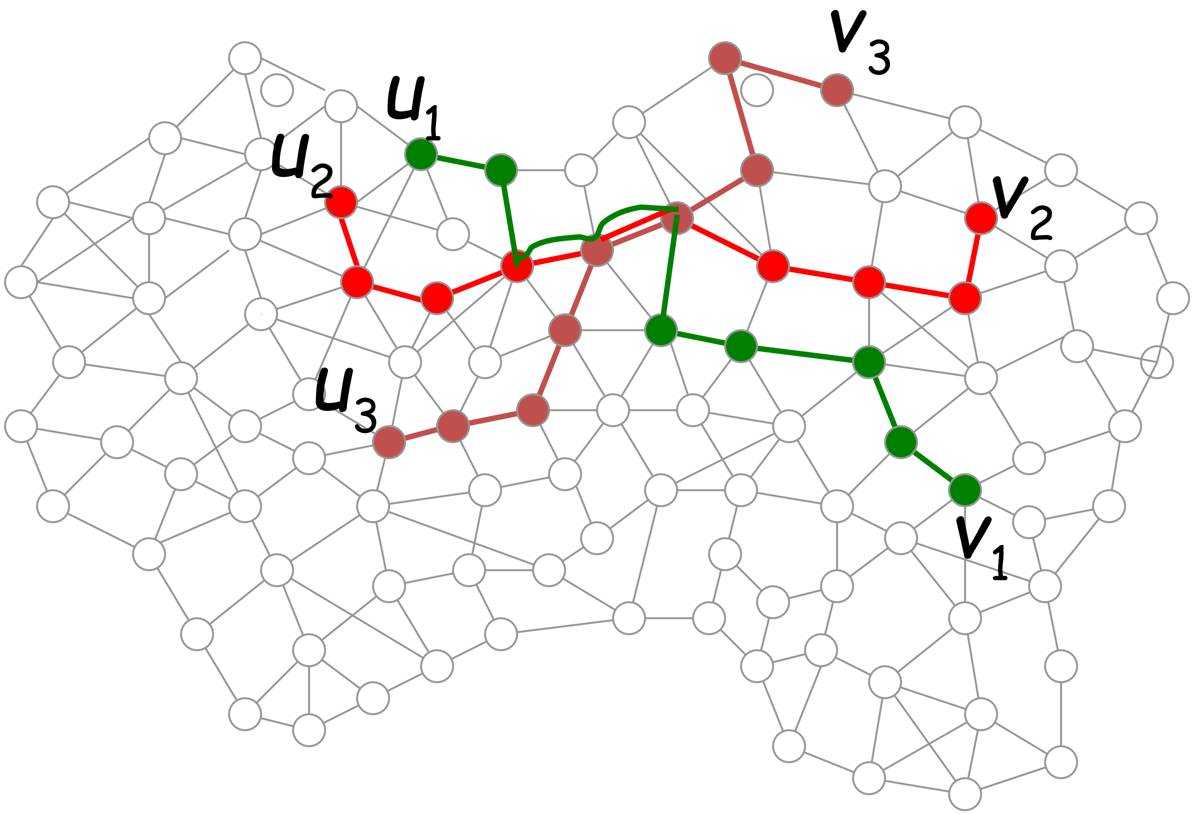
1. Tightly-Coupled Systems

2. Distributed Networked Systems

➢ <u>3. NUMA</u>

4. Future Directions

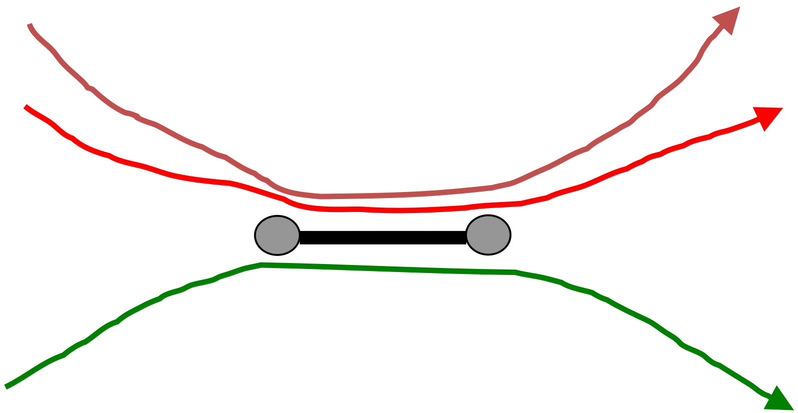
General routing: choose paths from sources to destinations



Routing in DTM: source node of the predecessor request in the total order is the destination of a successor request

Edge congestion

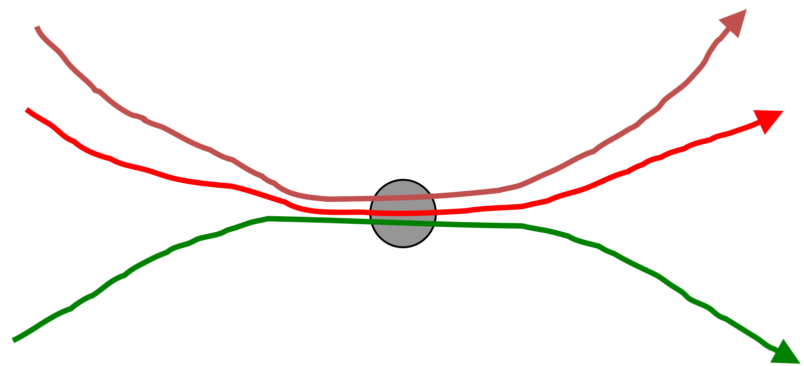
C_{edge}



maximum number of paths that use any edge

Node congestion

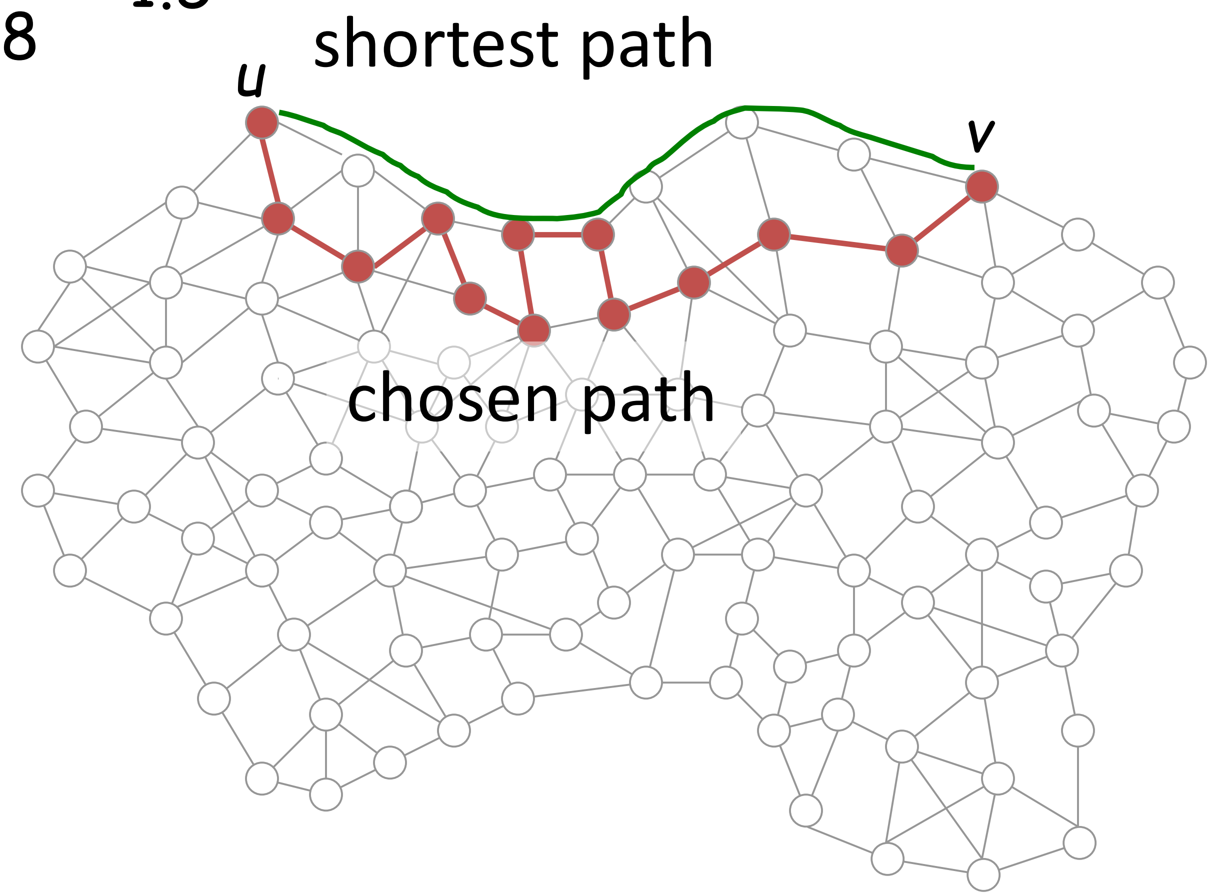
C_{node}



maximum number of paths that use any node

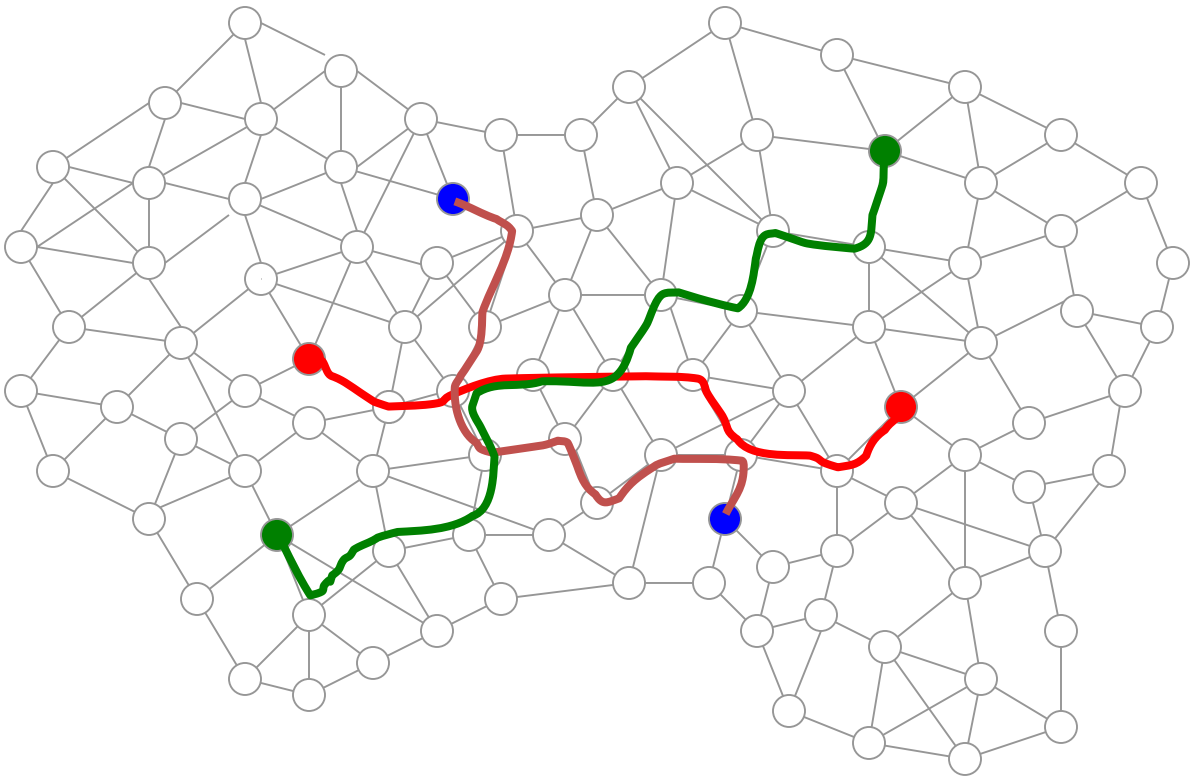
$$\text{Stretch} = \frac{\text{Length of chosen path}}{\text{Length of shortest path}}$$

$$stretch = \frac{12}{8} = 1.5$$



Inspiration: Oblivious Routing

Each request path choice is independent of other request path choices



Problem Statement

- Given a d-dimensional mesh and a finite set of operations $\boldsymbol{R} = \{r_0, r_1, \ldots, r_l\}$ on an object ξ
- Design a DTM algorithm that:
  - Minimizes congestion $C = max_e \, |\{i : p_i \ni e\}|$ on any edge e
  - Minimizes total communication cost $\mathrm{A}(\boldsymbol{R}) = \sum_{i=1}^{l} |p_i|$ for all the operations

<u>Limitation:</u> Congestion and stretch cannot be minimized simultaneously in arbitrary networks

Multibend DTM

- Focus on Mesh Neworks (general solution impossible)

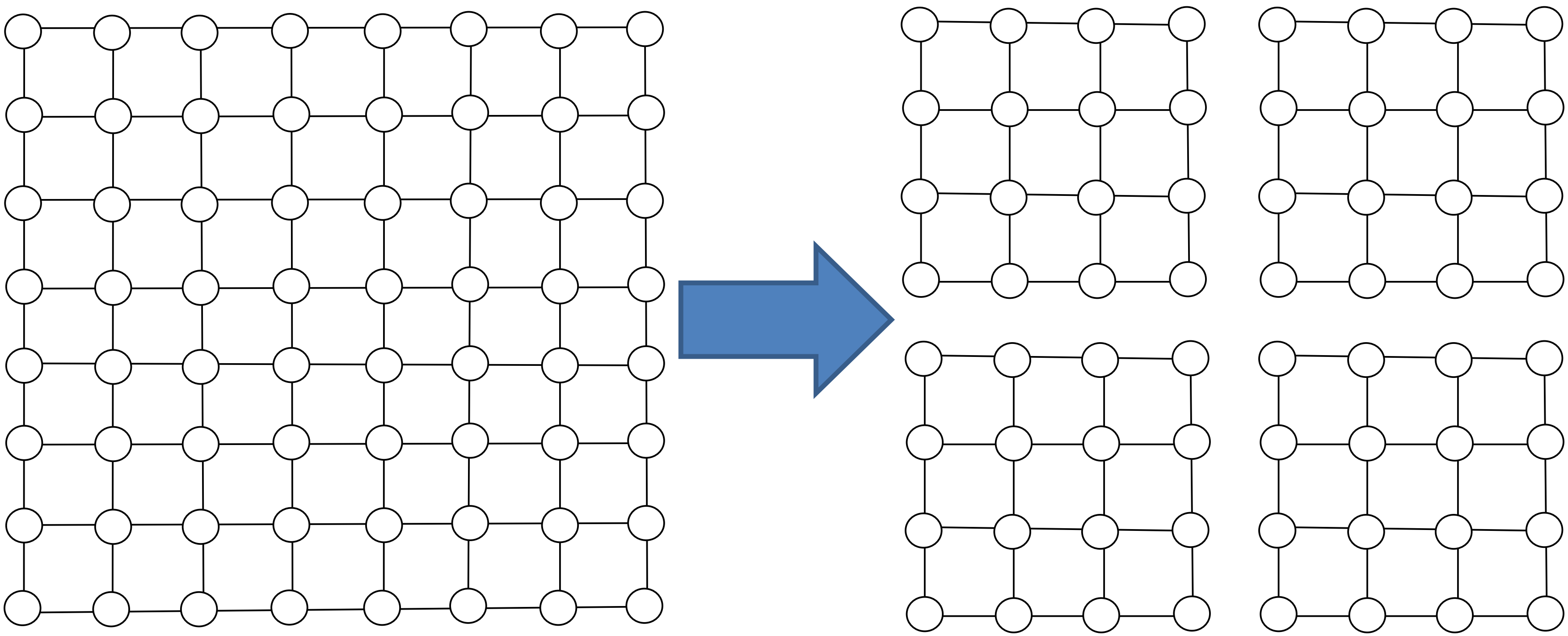
- For 2-dimensional mesh, **MultiBend** has both stretch and (edge) congestion $O(\log n)$

- For d-dimensional mesh, **MultiBend** has

  stretch $O(d \log n)$ and

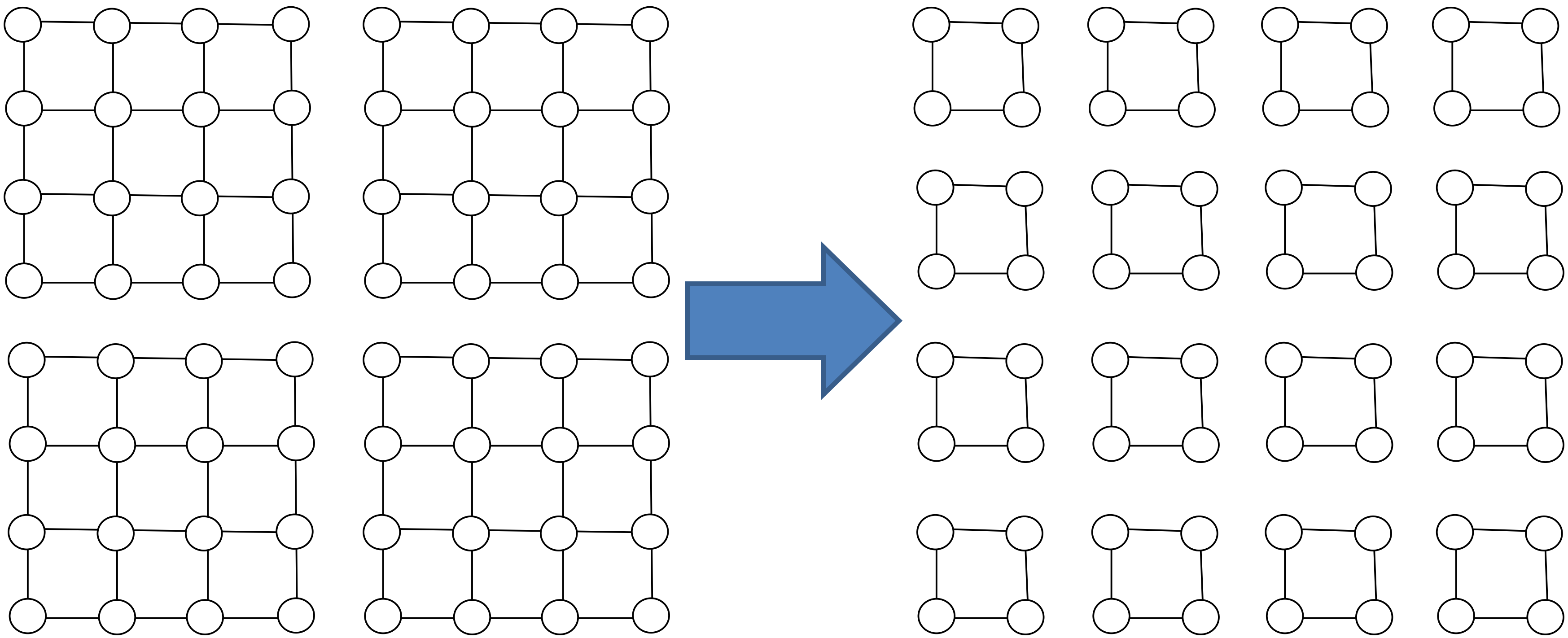
  congestion $O(d^2 \log n)$

Type-1 Mesh Decomposition

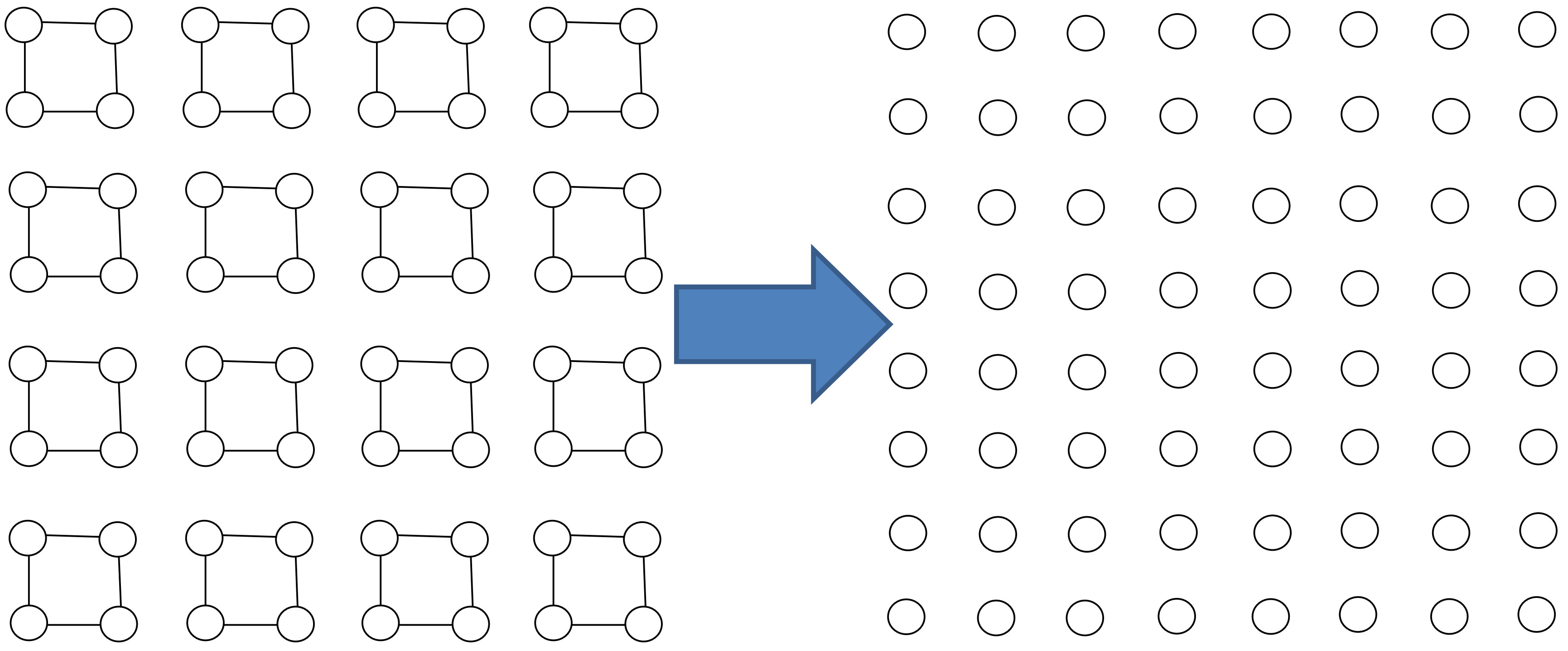


2-dimensional mesh

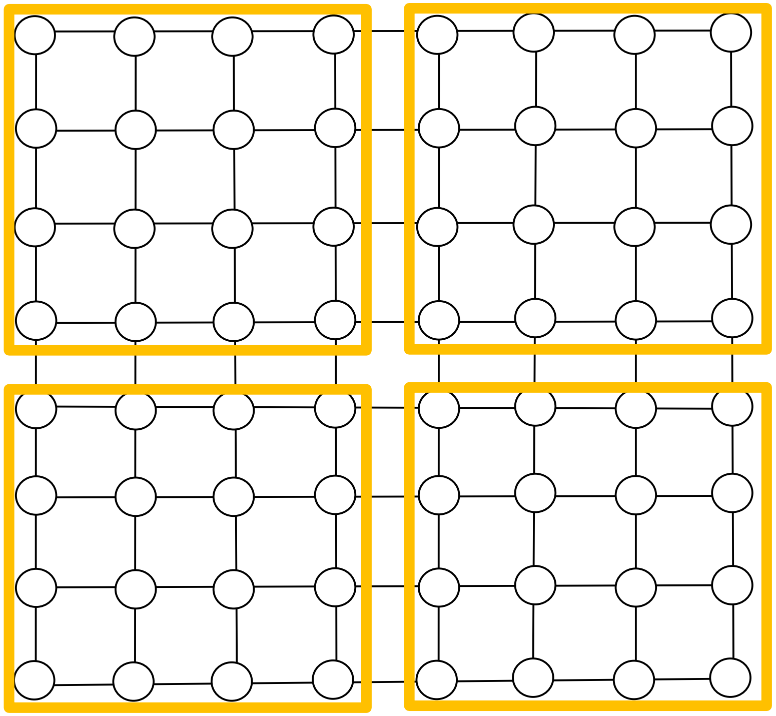
Type-1 Mesh Decomposition

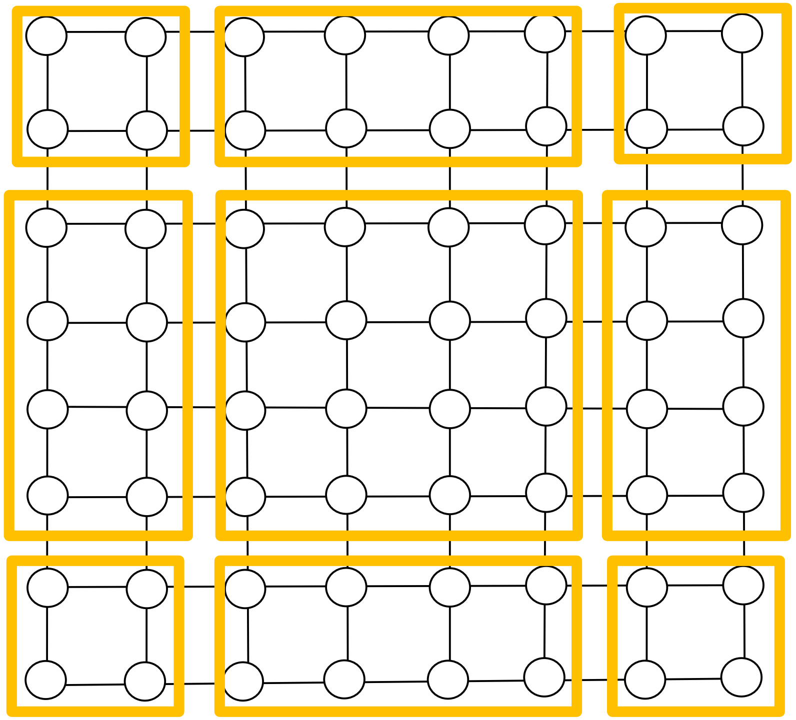


Type-1 Mesh Decomposition

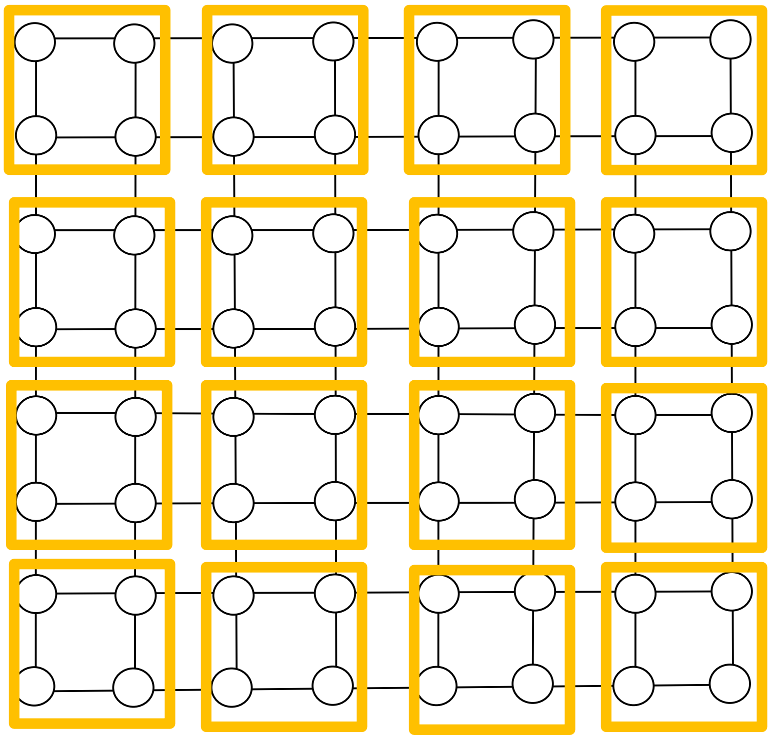


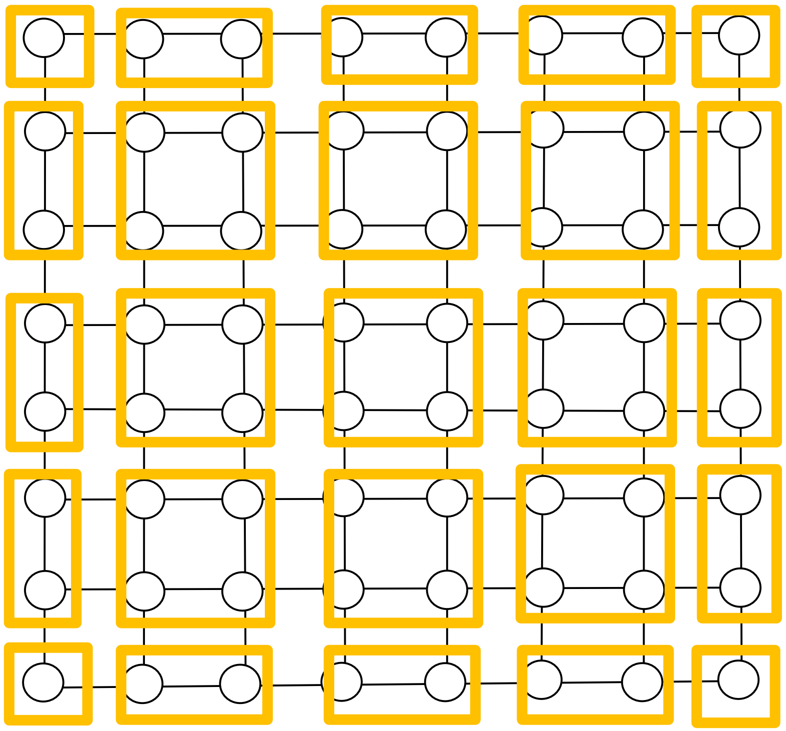
Type-2 Mesh Decomposition



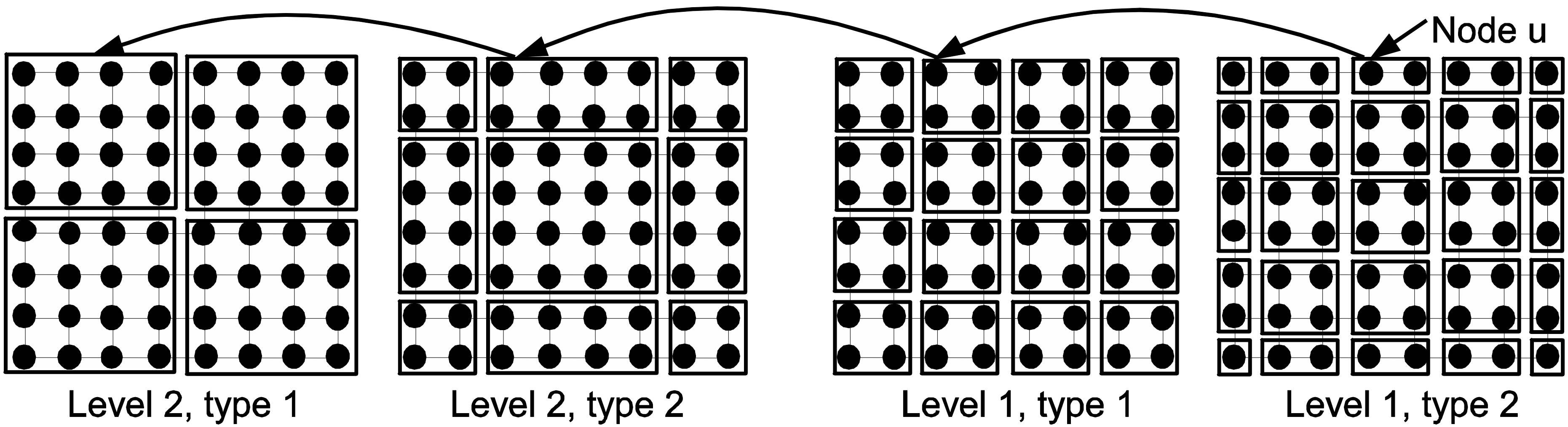


Type-2 Mesh Decomposition





Decomposition for 2^3x2^3 2-dimensional mesh

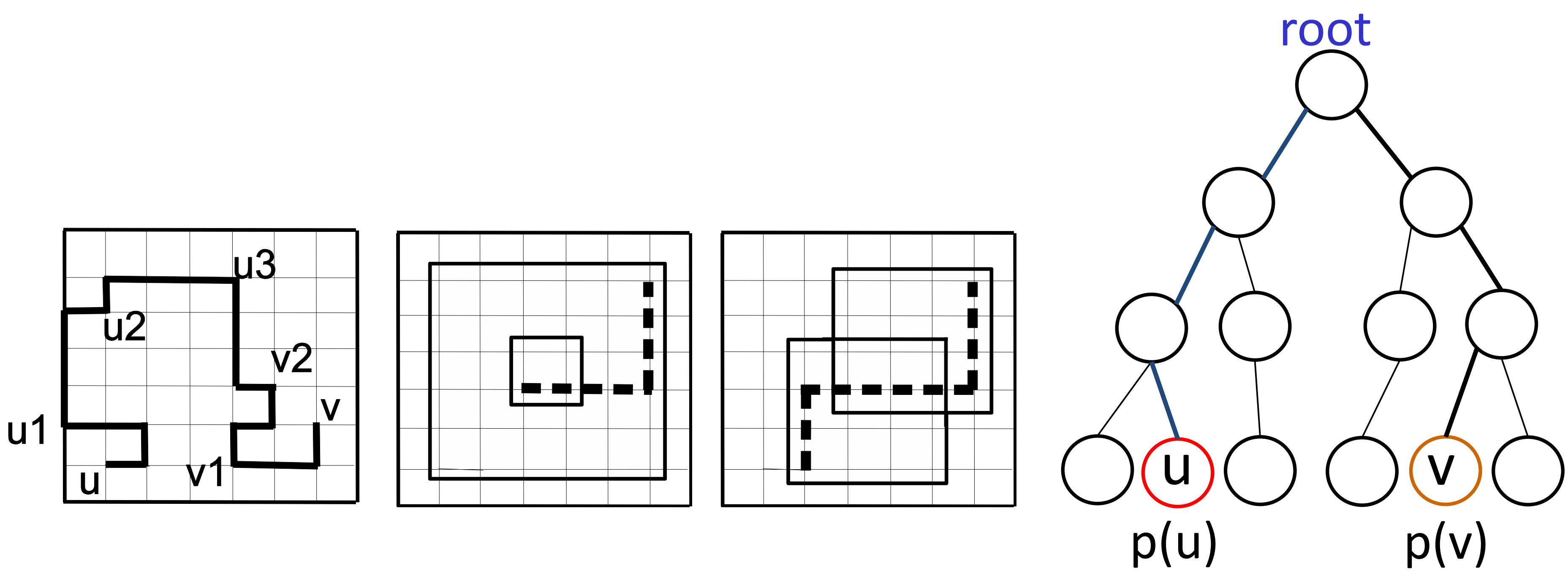


(i+1,2) (i+1,1) (i,2) (i,1)

Hierarchy levels

MultiBend Hierarchy

- Find a predecessor node via multi-bend paths for each leaf node u



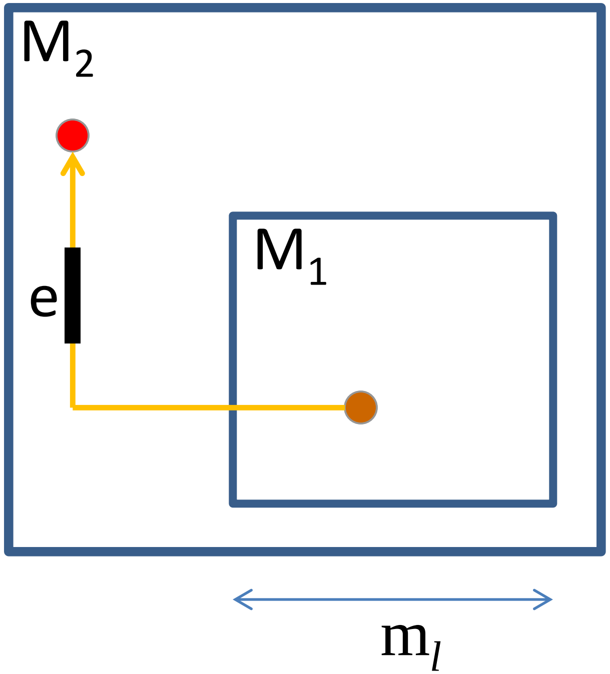
Load Balancing

- Through a leader election procedure
  - Every time we access the leader of a sub-mesh, we replace it with another leader chosen uniformly at random among its nodes
- The update cost is low in comparison to the cost of serving requests

Analysis on (Edge) Congestion

- A sub-path uses edge e with probability $2/m_l$
- P': set of paths from M_1 to M_2 or vice-versa
- $C'(e)$: Congestion caused by P' on e
- $E[C'(e)] \leq 2|P'|/m_l$
- $B \geq |P'|/out(M_1)$
- $out(M_1) \leq 4m_l$
- $C^* \geq B$

==> $E[C'(e)] \leq 8C^*$



Assume M_1 is a type-1 submesh

Presentation Outline

1. Tightly-Coupled Systems

2. Distributed Networked Systems

3. NUMA

➢ <u>4. Future Directions</u>

Future Directions

- Distributed Networked systems
  - Multiple objects
    - minimize time and communication cost
  - Fault tolerance
  - Dynamic networks

- NUMA
  - Study other network architectures