Meridian: A Lightweight Network Location Service without Virtual Coordinates

Bernard Wong
Aleksandrs Slivkins
Emin Gün Sirer

Department of Computer Science
Cornell University

Network Location Service

- Select nodes based on a set of network properties

- Real-world problems:
  - **Locate closest game server**
  - **Distribute web-crawling to nearby hosts**
  - **Perform efficient application level multicast**
  - **Satisfy a Service Level Agreement**
  - **Provide inter-node latency bounds for clusters**

- Underlying abstract problems
  - **Finding closest node to target**
  - **Finding the closest node to the center of a set of targets**
  - **Finding a node that is $<r_i$ ms from target t_i for all targets**

Current State-of-the-Art: Virtual Coordinates

- Maps Internet latencies into low dimensional space
  - GNP, Vivaldi, Lighthouse, ICS, VL, BBS, PIC, NPS, etc.
- Reduces number of real-time measurements
- 3 practical problems:
  - Introduces inherent embedding error
  - A snapshot in time of the network location of a node
    - Coordinates become stale over time
    - Latency estimates based on coordinates computed at different times can lead to additional errors
  - Requires additional P2P substrate to solve network location problems without centralized servers or *O(N)* state

Meridian Approach

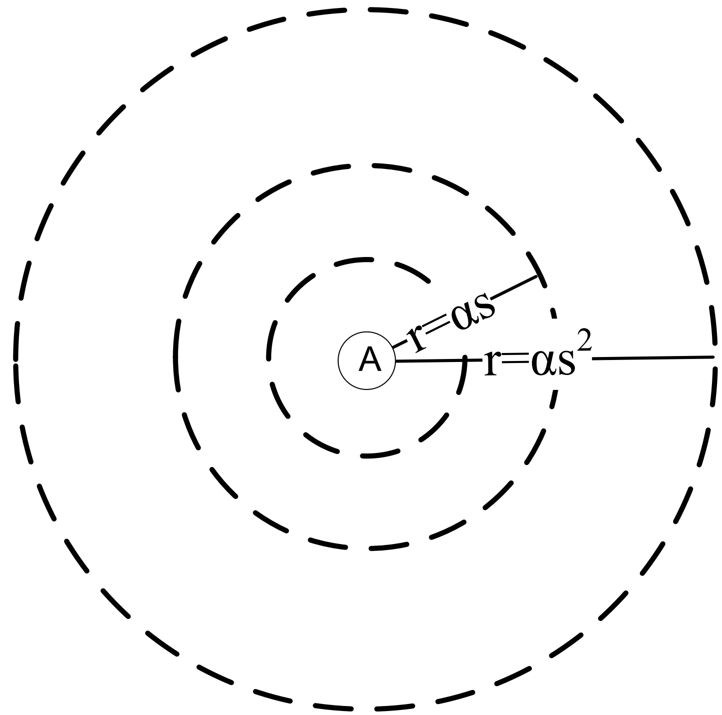
- Solve node selection directly without computing coordinates
  - Combine query routing with active measurements
- 3 Design Goals:
  - Accurate: Find satisfying nodes with high probability
  - General: Users can fully express their network location requirements
  - Scalable: O(log N) state per node, O(log D) hops per query
- Design tradeoffs:
  - Active measurements incur higher query latencies
  - Overhead more dependent on query load

Meridian Operation

- Framework:
  - Loosely structured overlay network
- Algorithms:
  - Solve network location problems in O(log D) hops
- Language:
  - General-purpose language for expressing network location requirements

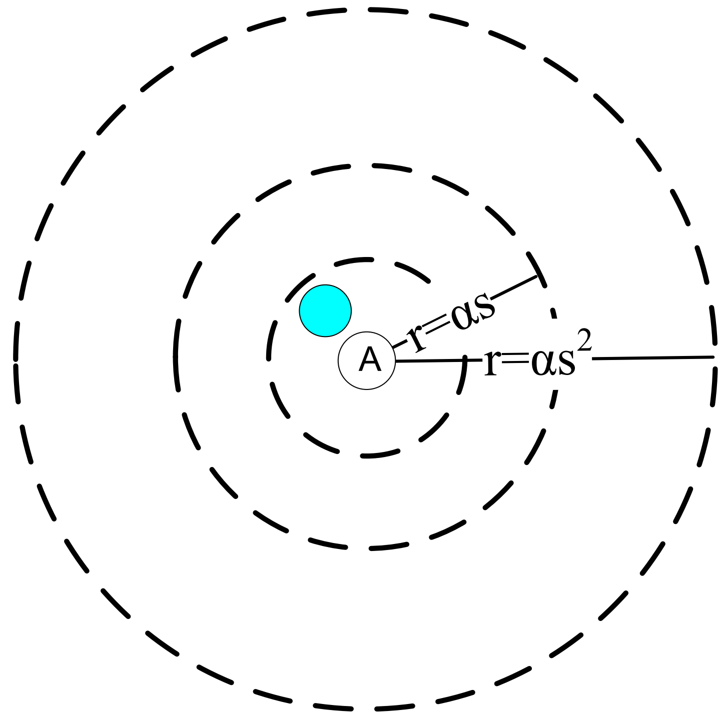
Multi-resolution Rings

- Organize peers into small fixed number of concentric rings
- Radii of rings grow outwards exponentially
  - Logarithmic # of peers per ring
  - Favors nearby neighbors
  - Retains a sufficient number of pointers to remote regions
- Gossip protocol used for peer discovery



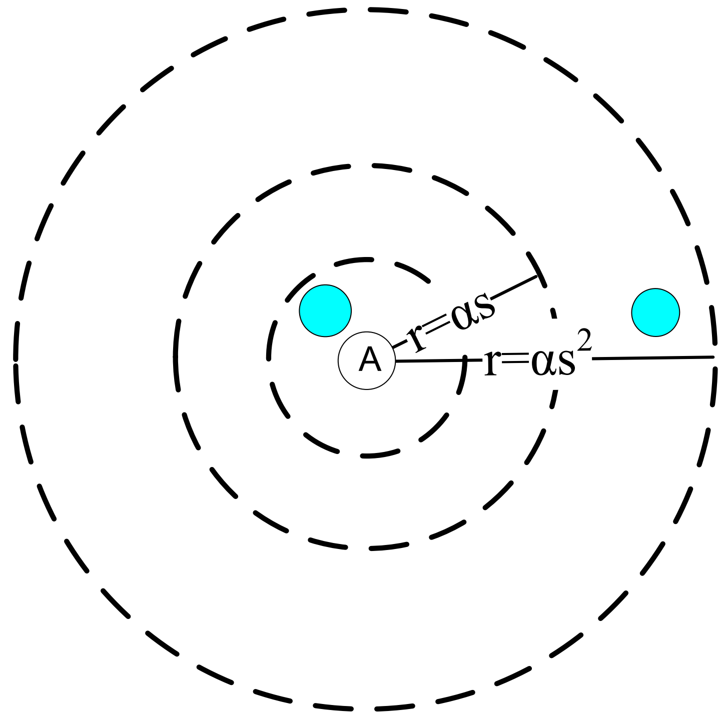
Multi-resolution Rings

- Organize peers into small fixed number of concentric rings
- Radii of rings grow outwards exponentially
  - Logarithmic # of peers per ring
  - Favors nearby neighbors
  - Retains a sufficient number of pointers to remote regions
- Gossip protocol used for peer discovery



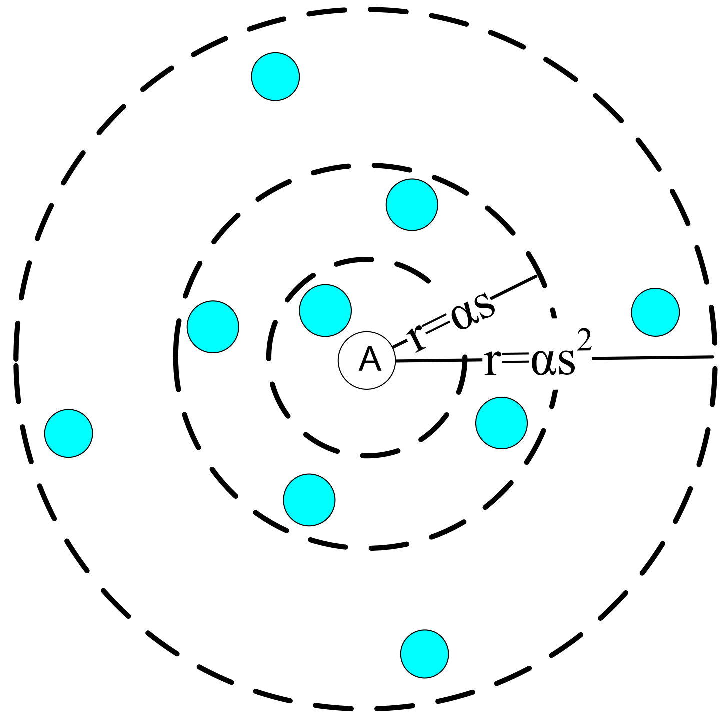
Multi-resolution Rings

- Organize peers into small fixed number of concentric rings
- Radii of rings grow outwards exponentially
  - Logarithmic # of peers per ring
  - Favors nearby neighbors
  - Retains a sufficient number of pointers to remote regions
- Gossip protocol used for peer discovery



Multi-resolution Rings

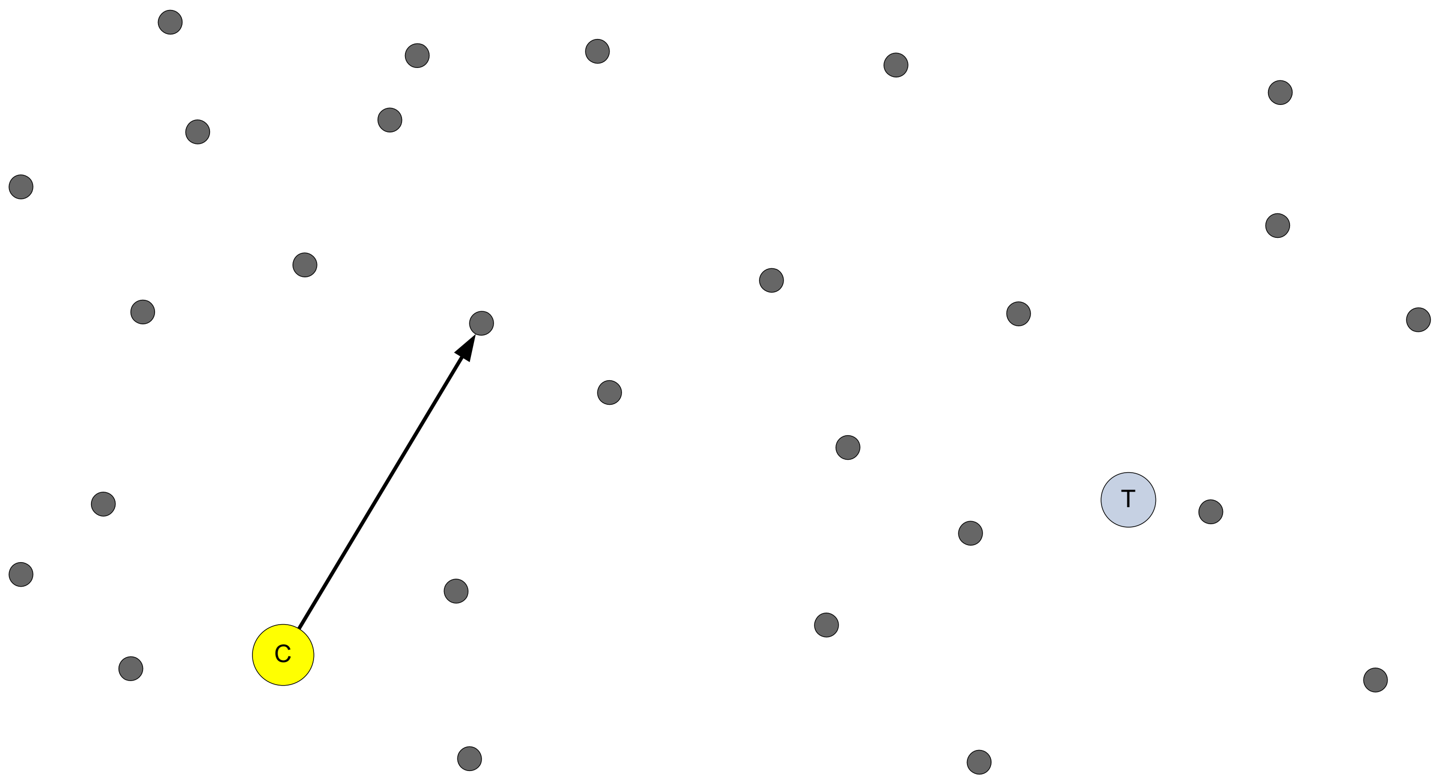
- Organize peers into small fixed number of concentric rings
- Radii of rings grow outwards exponentially
  - Logarithmic # of peers per ring
  - Favors nearby neighbors
  - Retains a sufficient number of pointers to remote regions
- Gossip protocol used for peer discovery



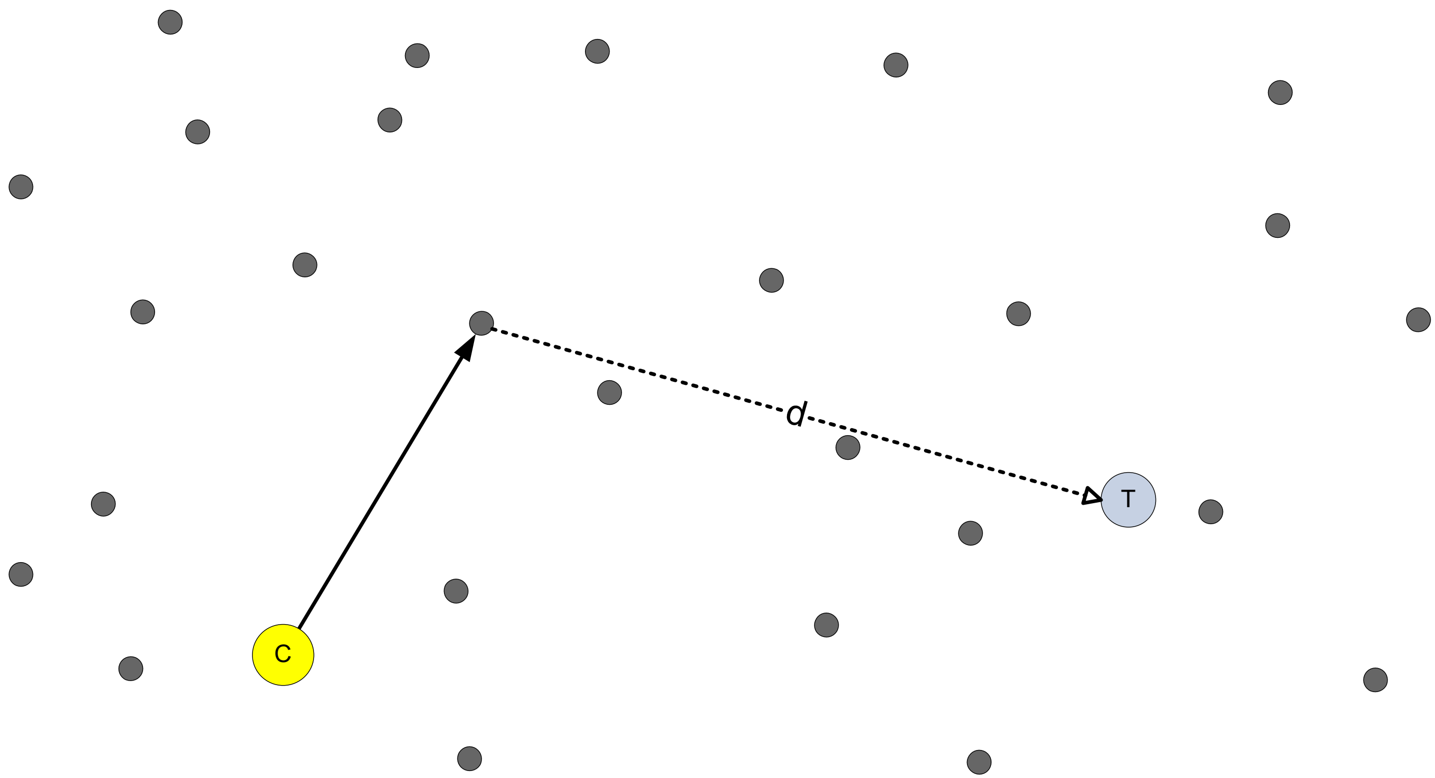
Closest Node Discovery

- Multi-hop search
  - Similar to finding the closest identifier in DHTs
    - Replaces virtual identifiers with physical latencies
  - Each hop exponentially reduces the distance to the target
  - Reduction threshold β for $0 \leq \beta < 1$
    - Only take another hop if a peer node is β times closer
  - Limits # of probed peers through triangle inequality

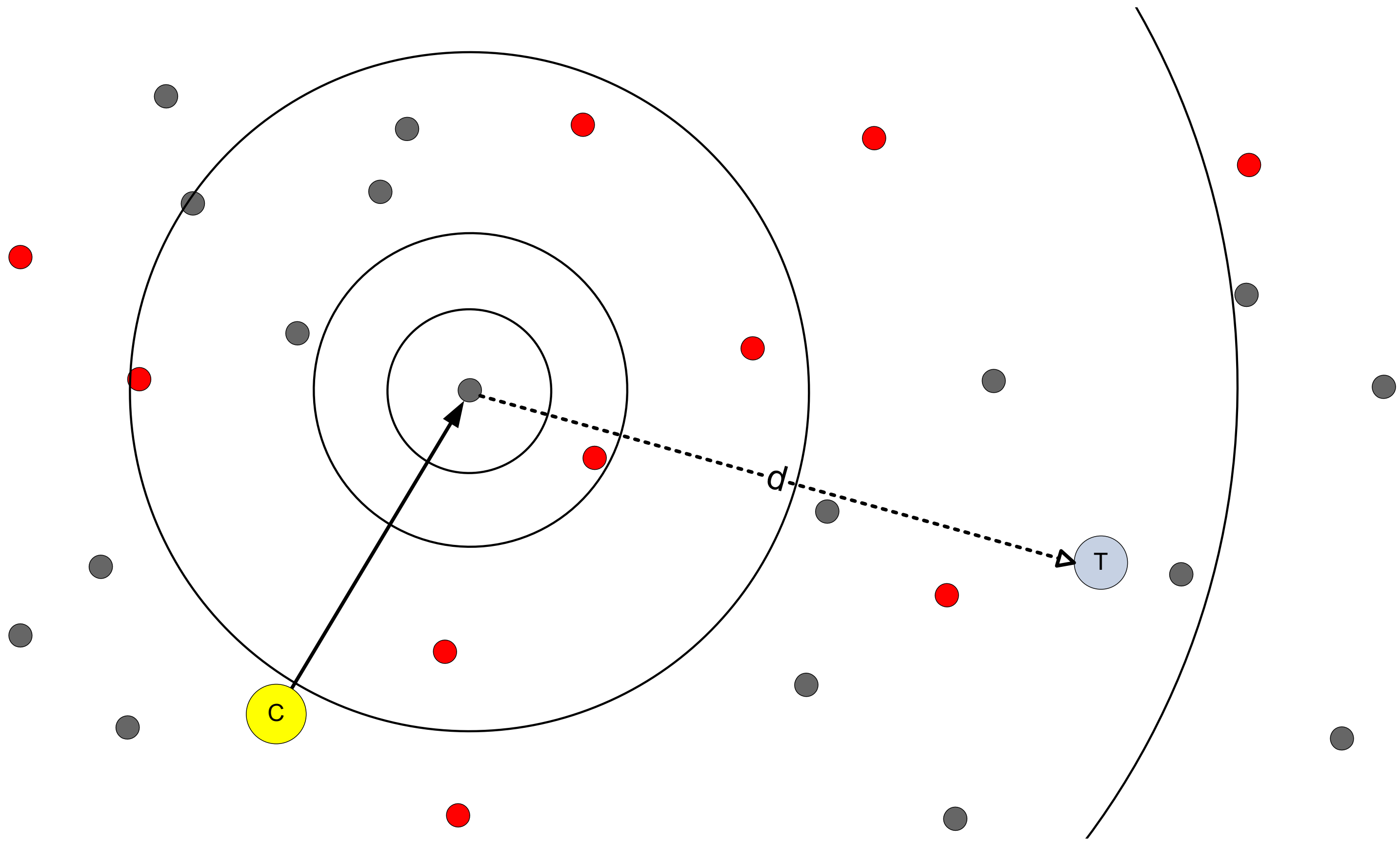
Closest Node Discovery



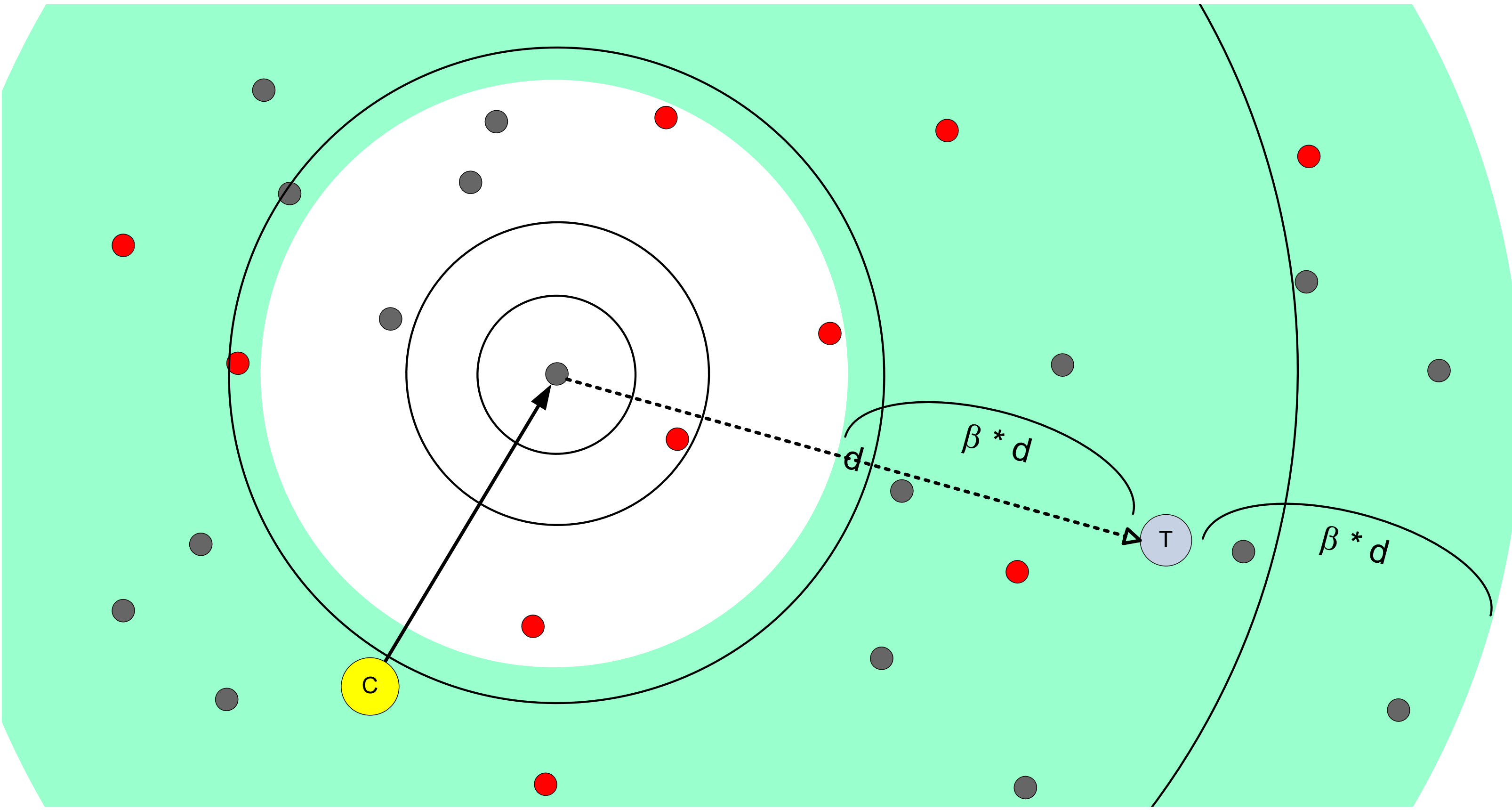
Closest Node Discovery



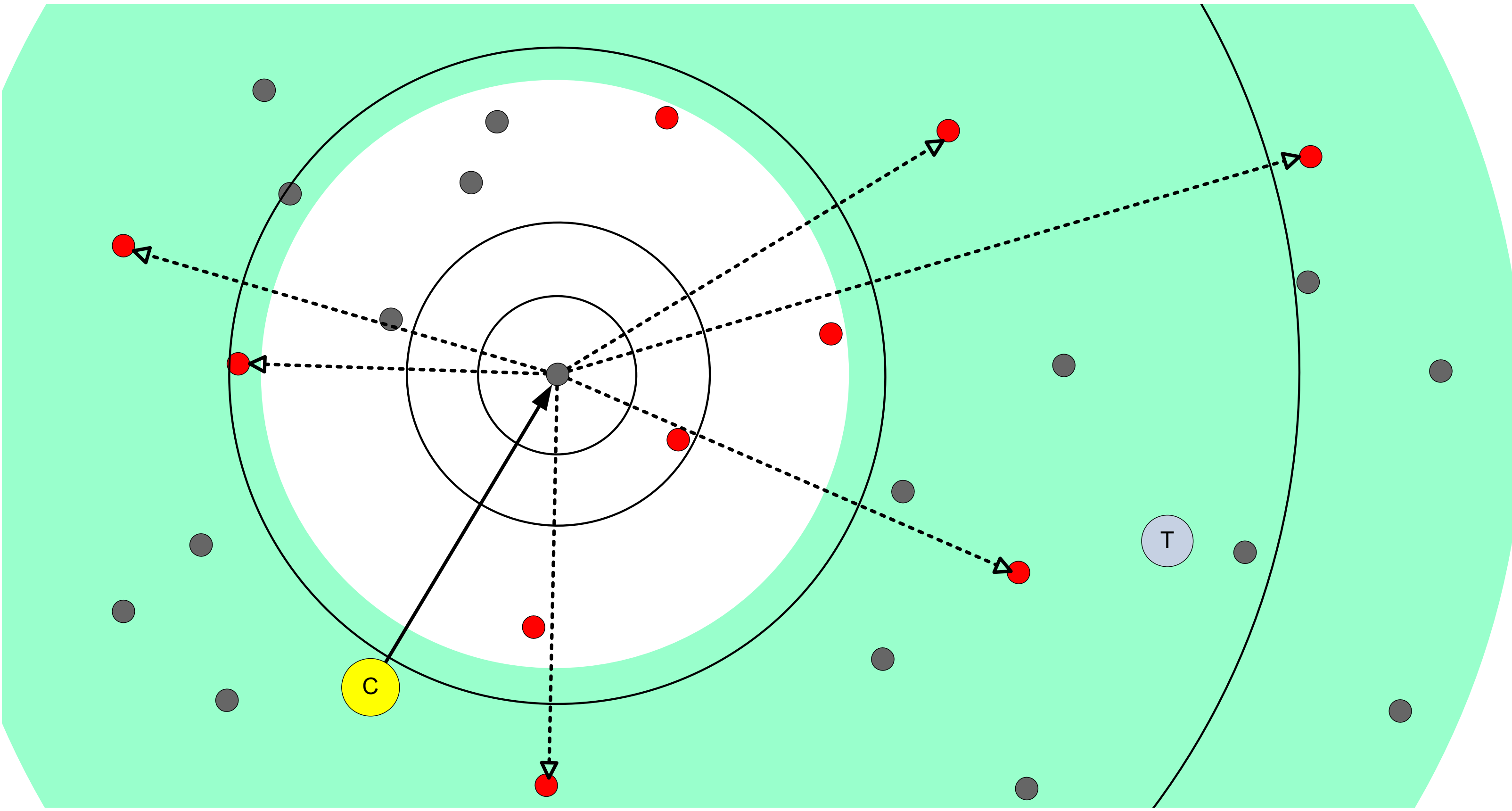
Closest Node Discovery



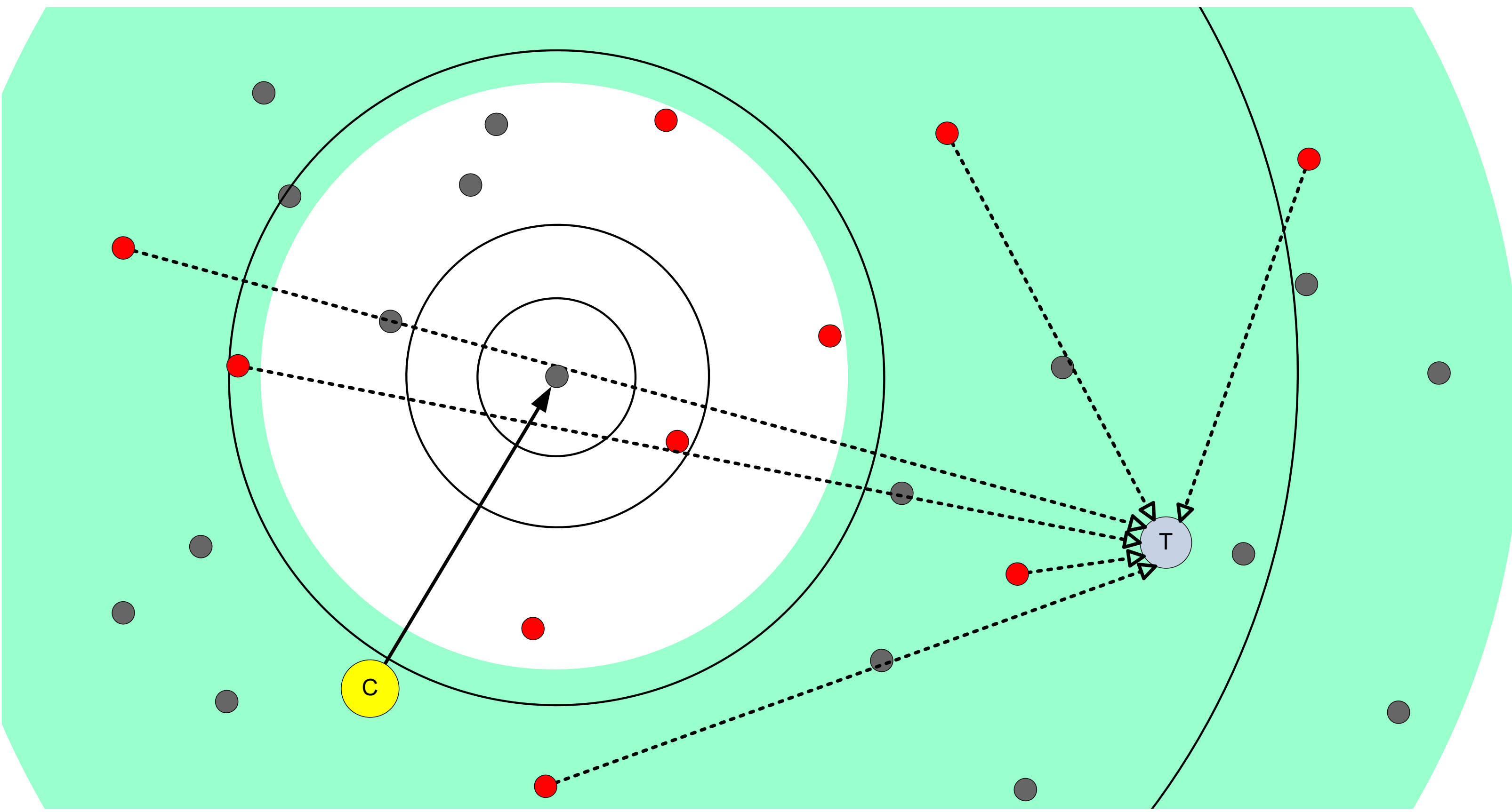
Closest Node Discovery



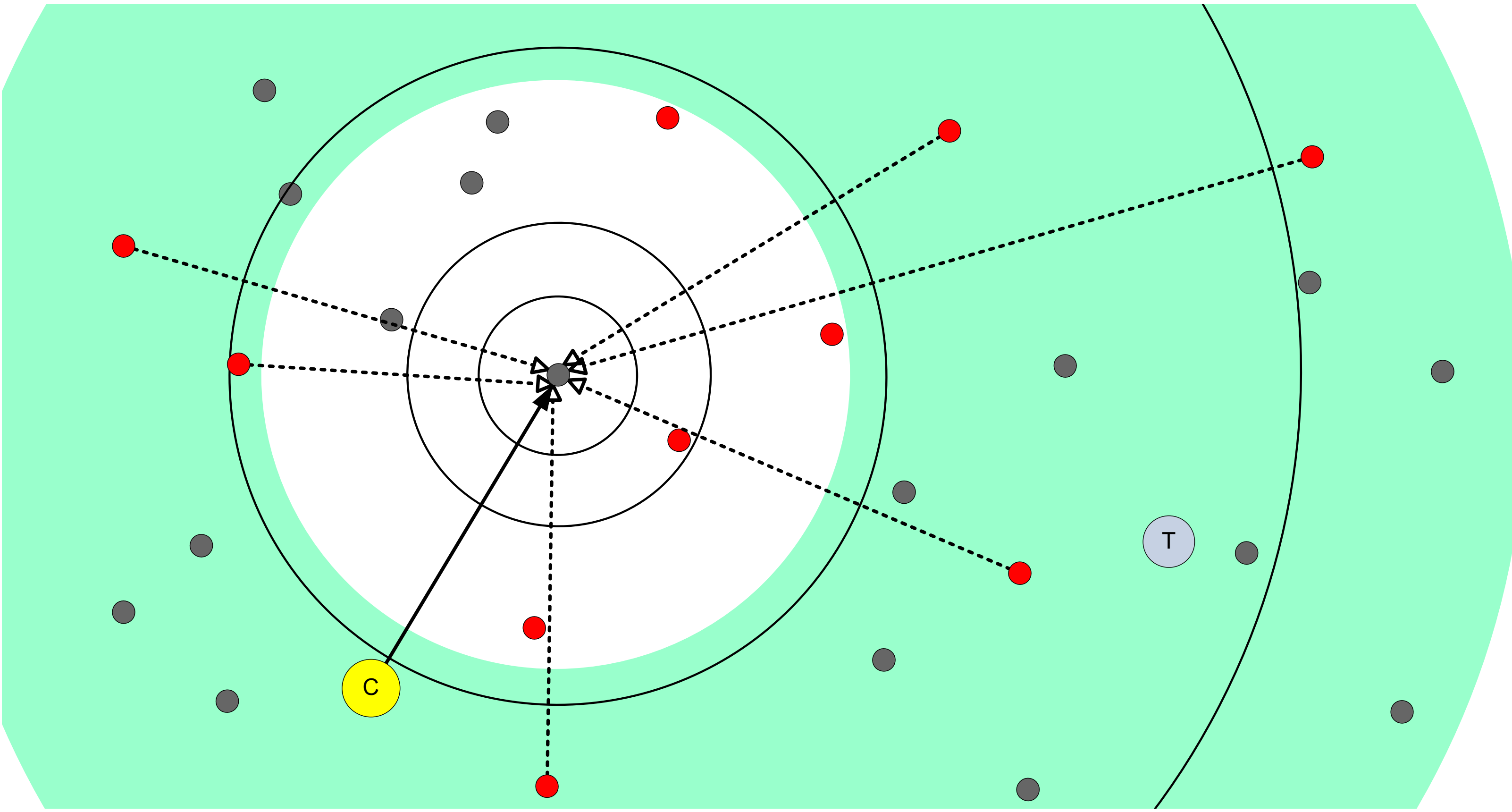
Closest Node Discovery



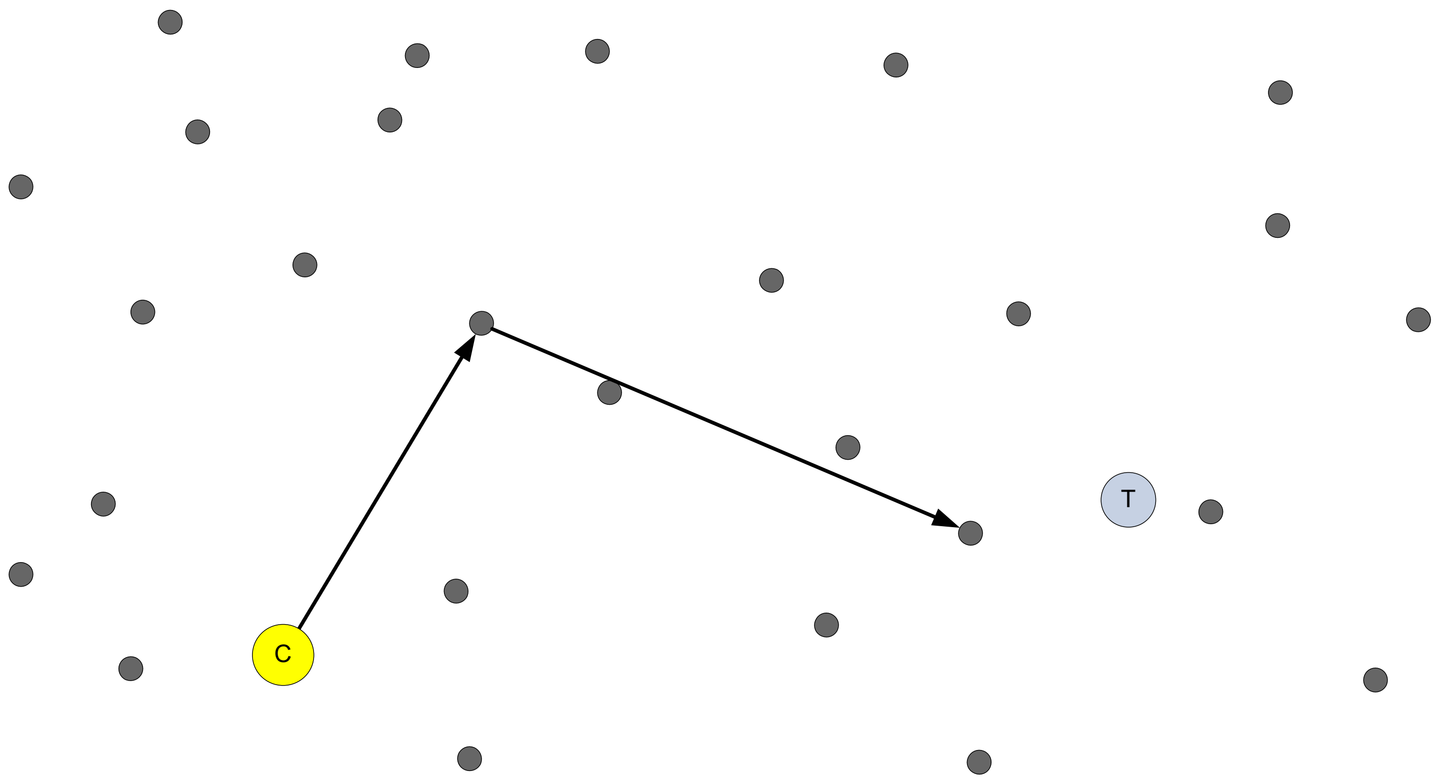
Closest Node Discovery



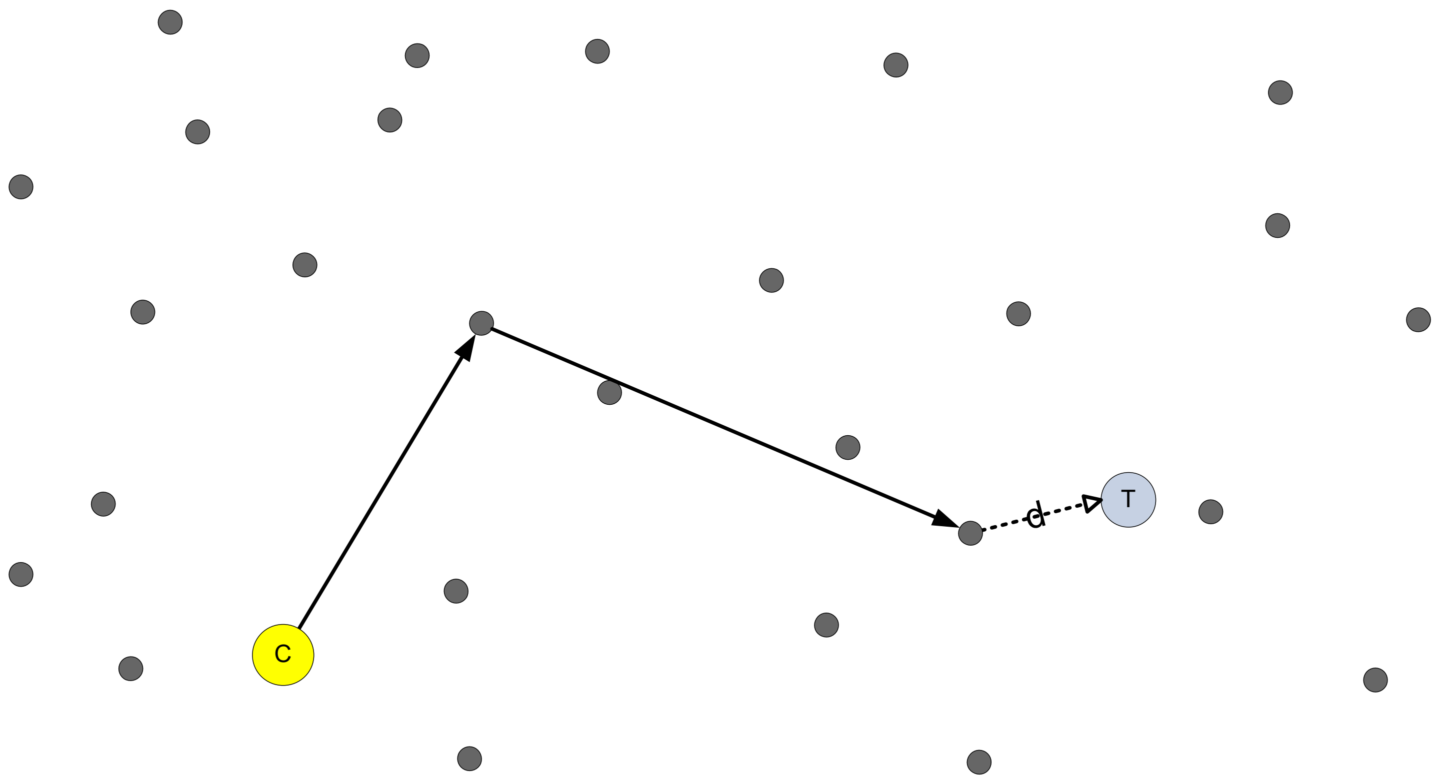
Closest Node Discovery



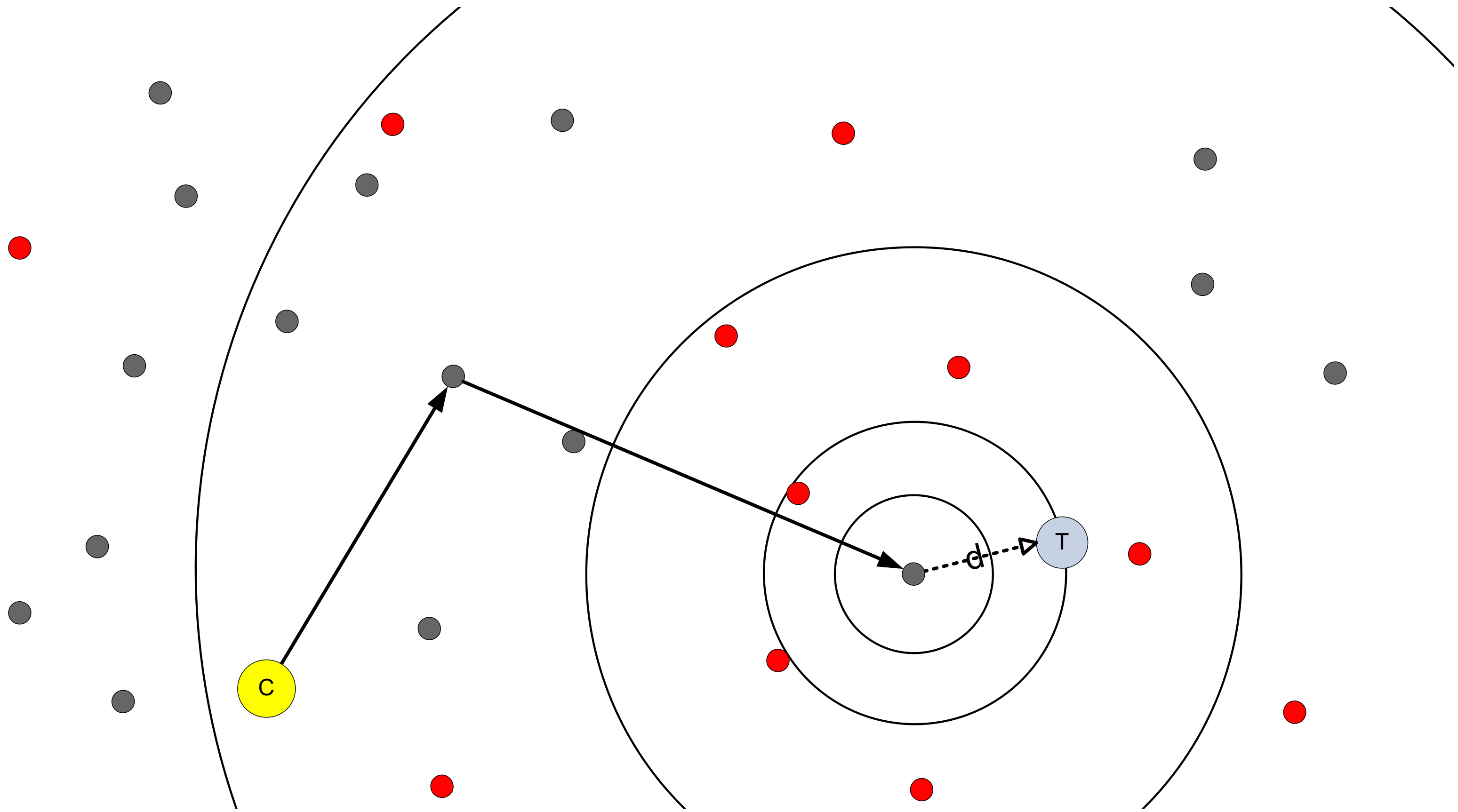
Closest Node Discovery



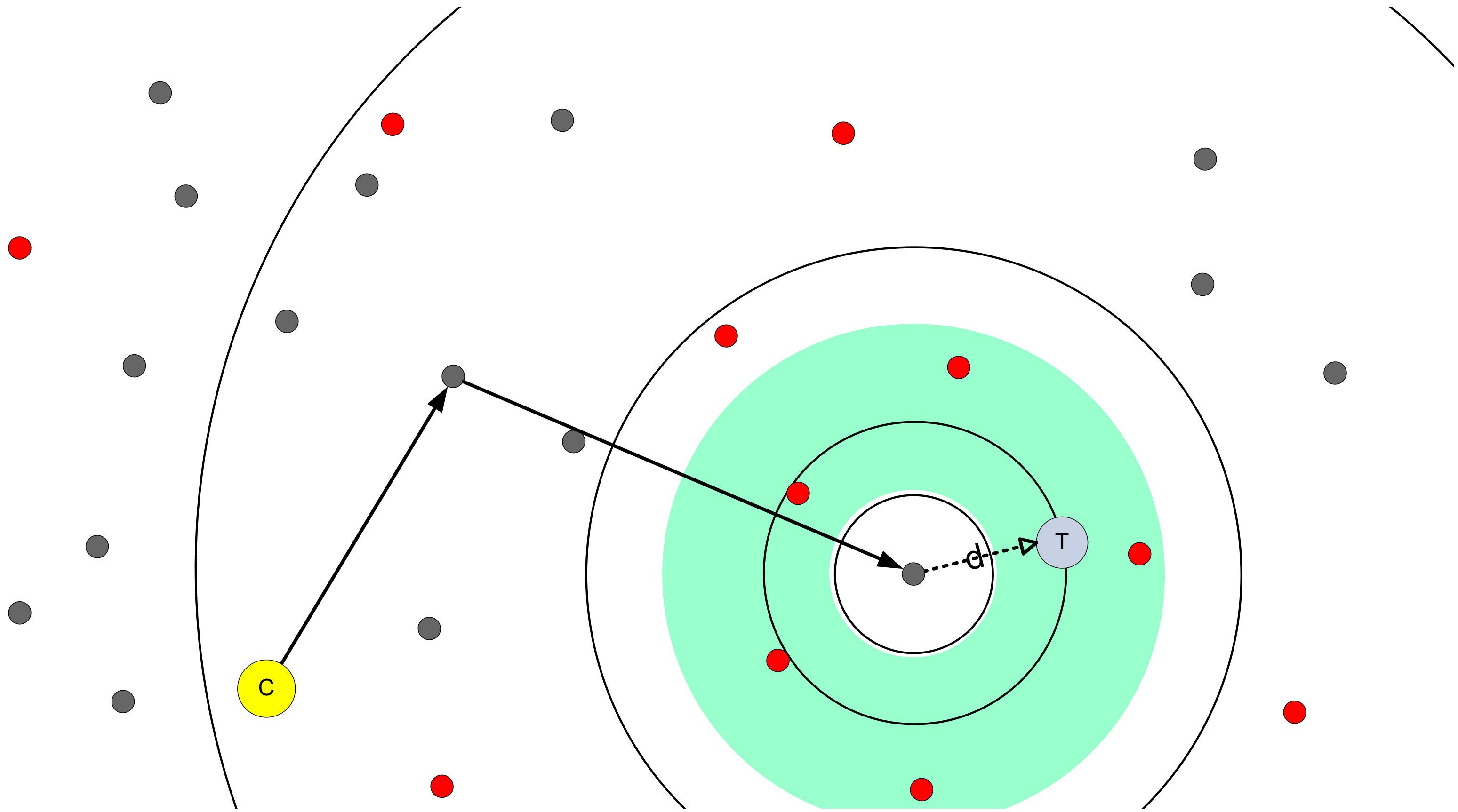
Closest Node Discovery



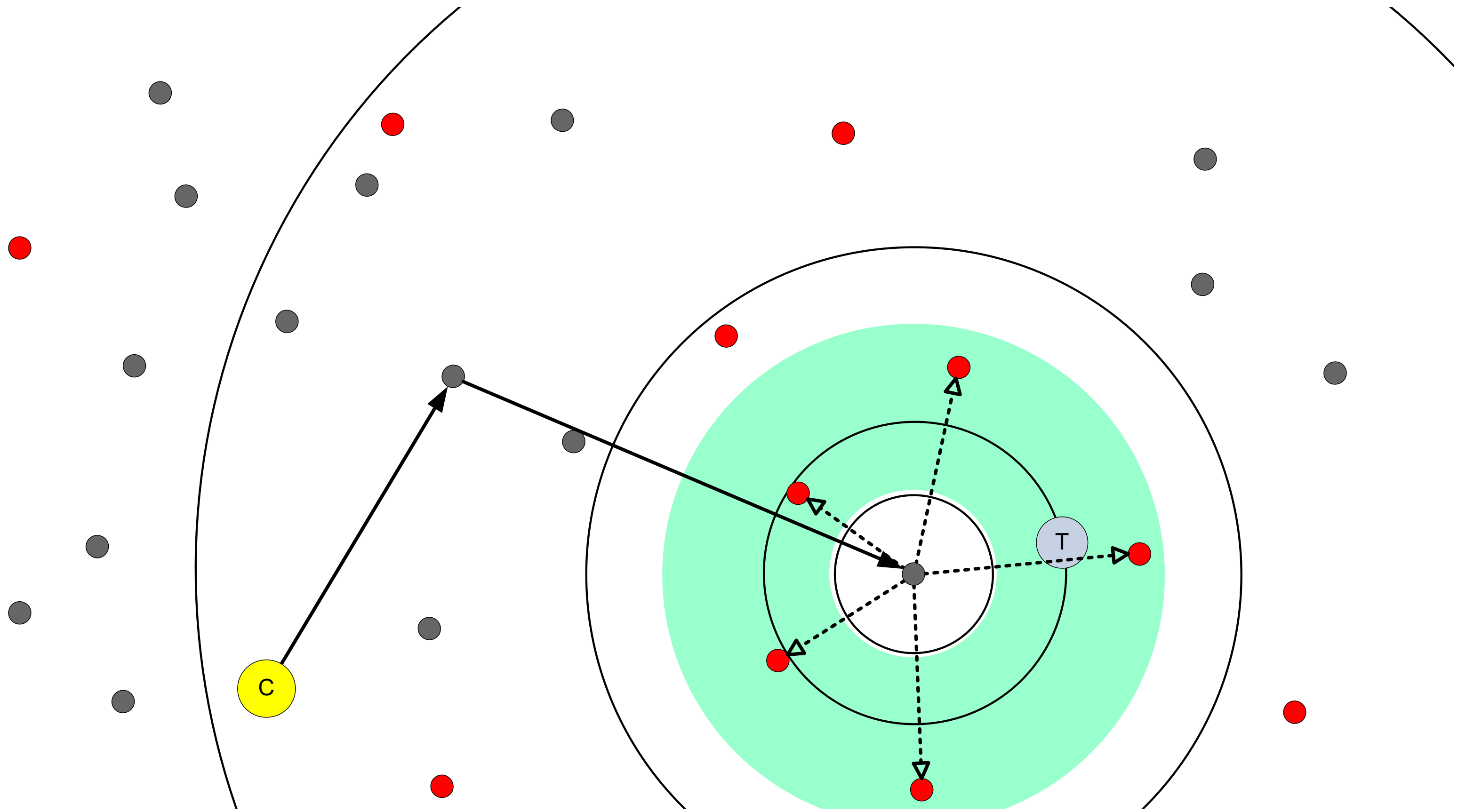
Closest Node Discovery



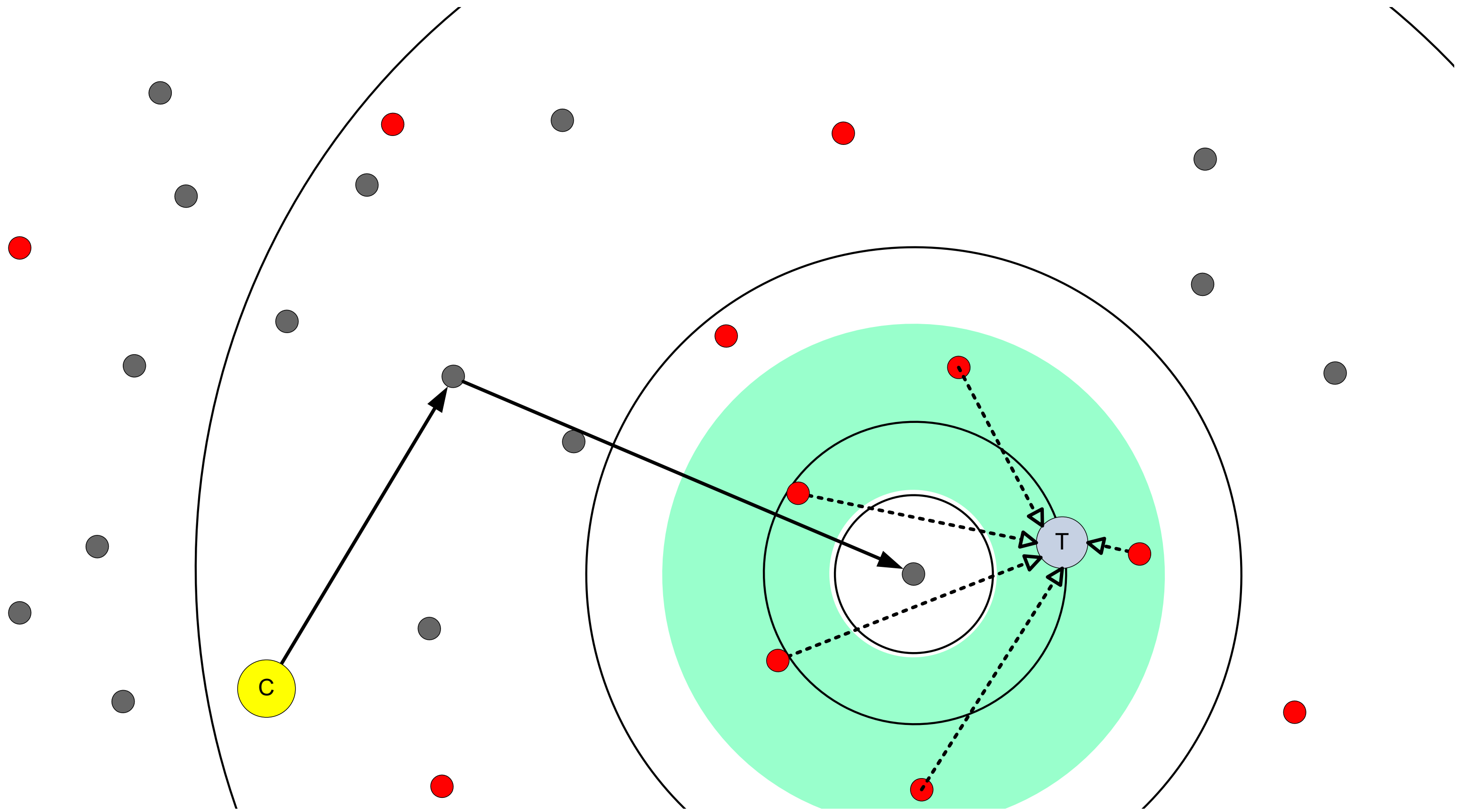
Closest Node Discovery



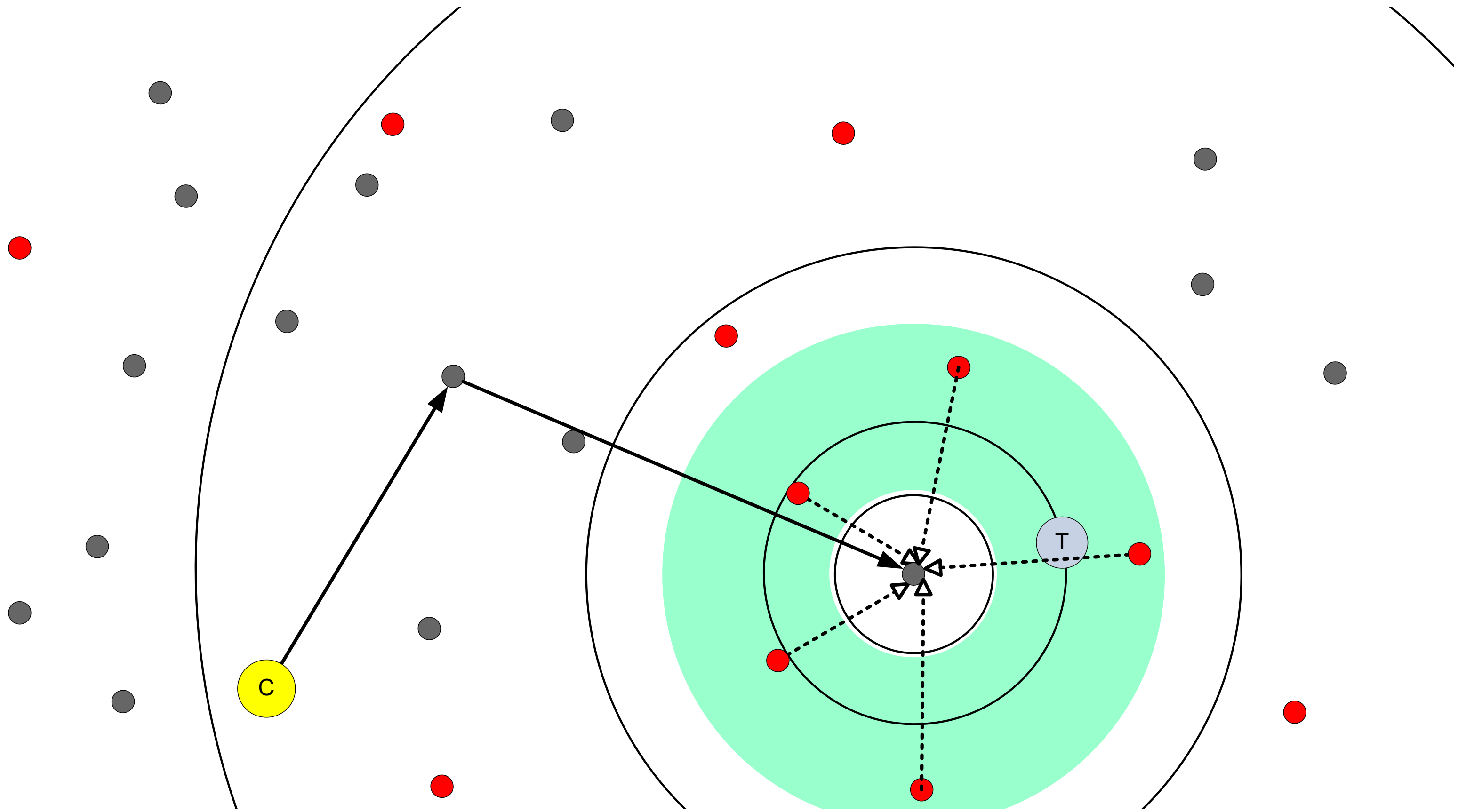
Closest Node Discovery



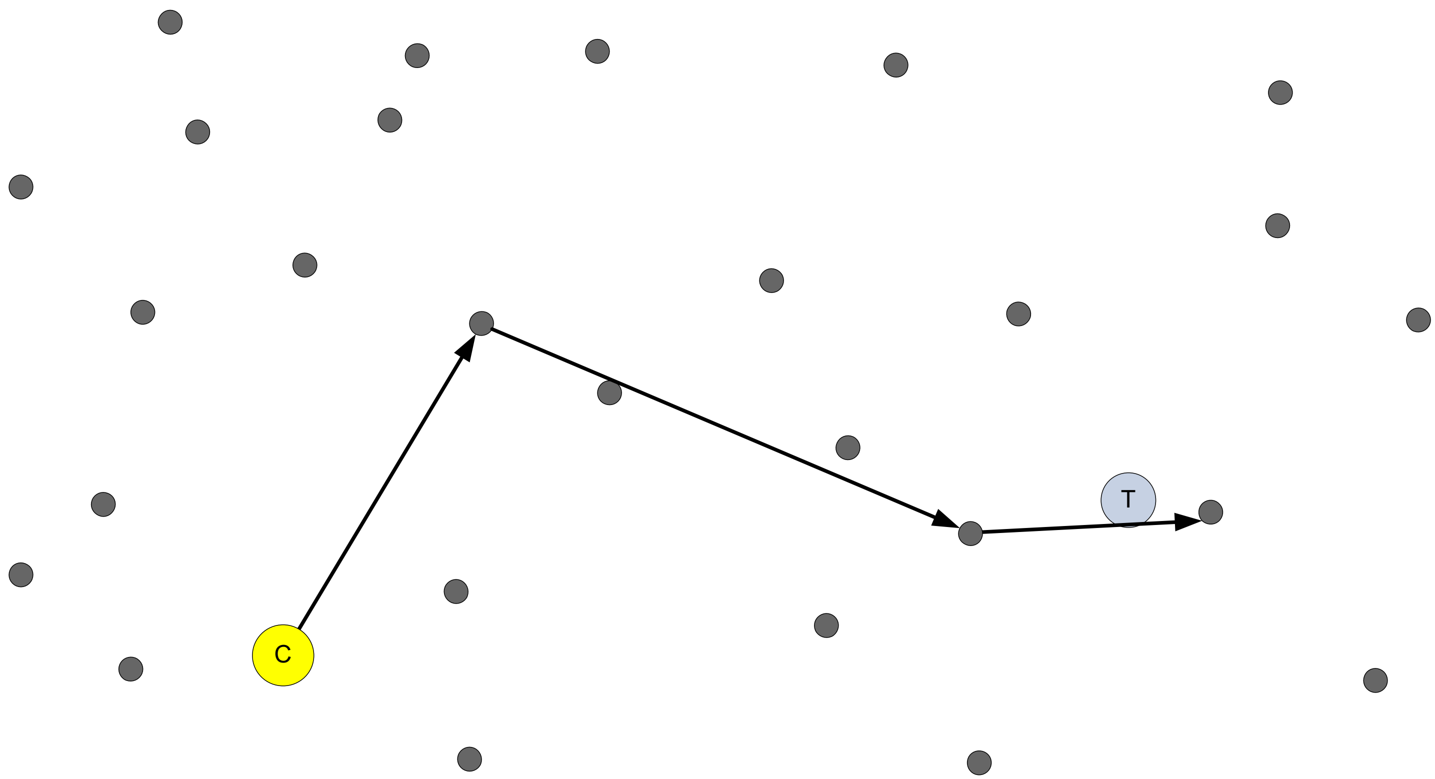
Closest Node Discovery



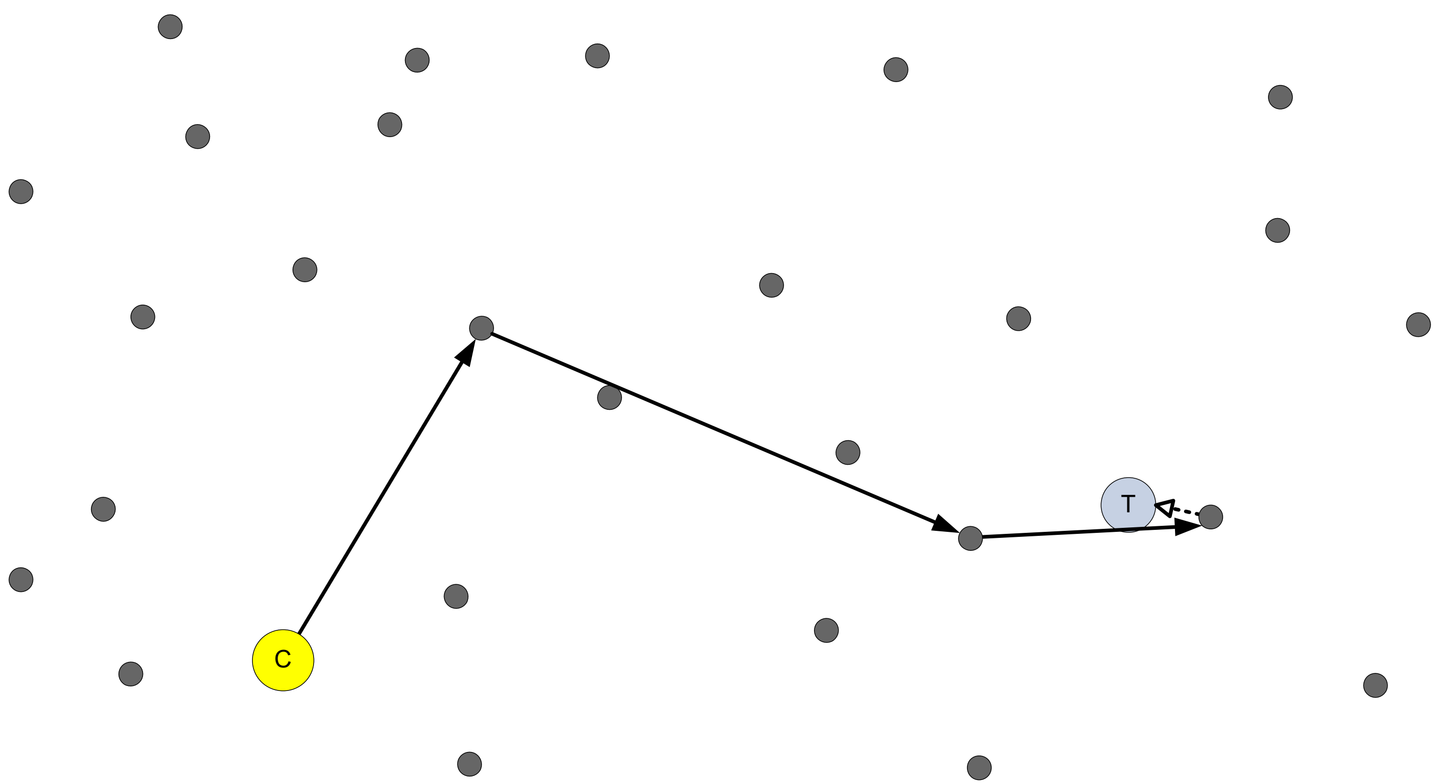
Closest Node Discovery



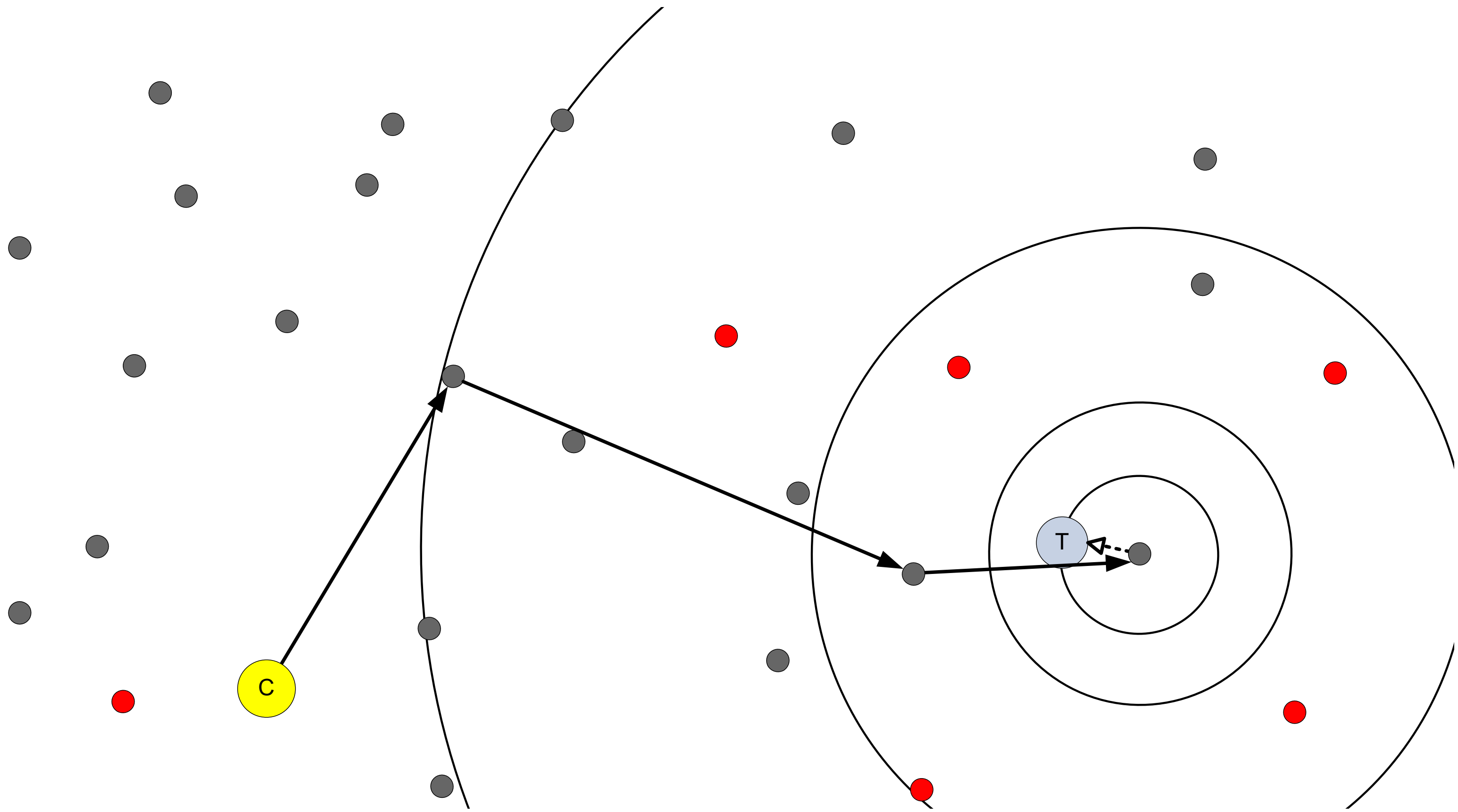
Closest Node Discovery



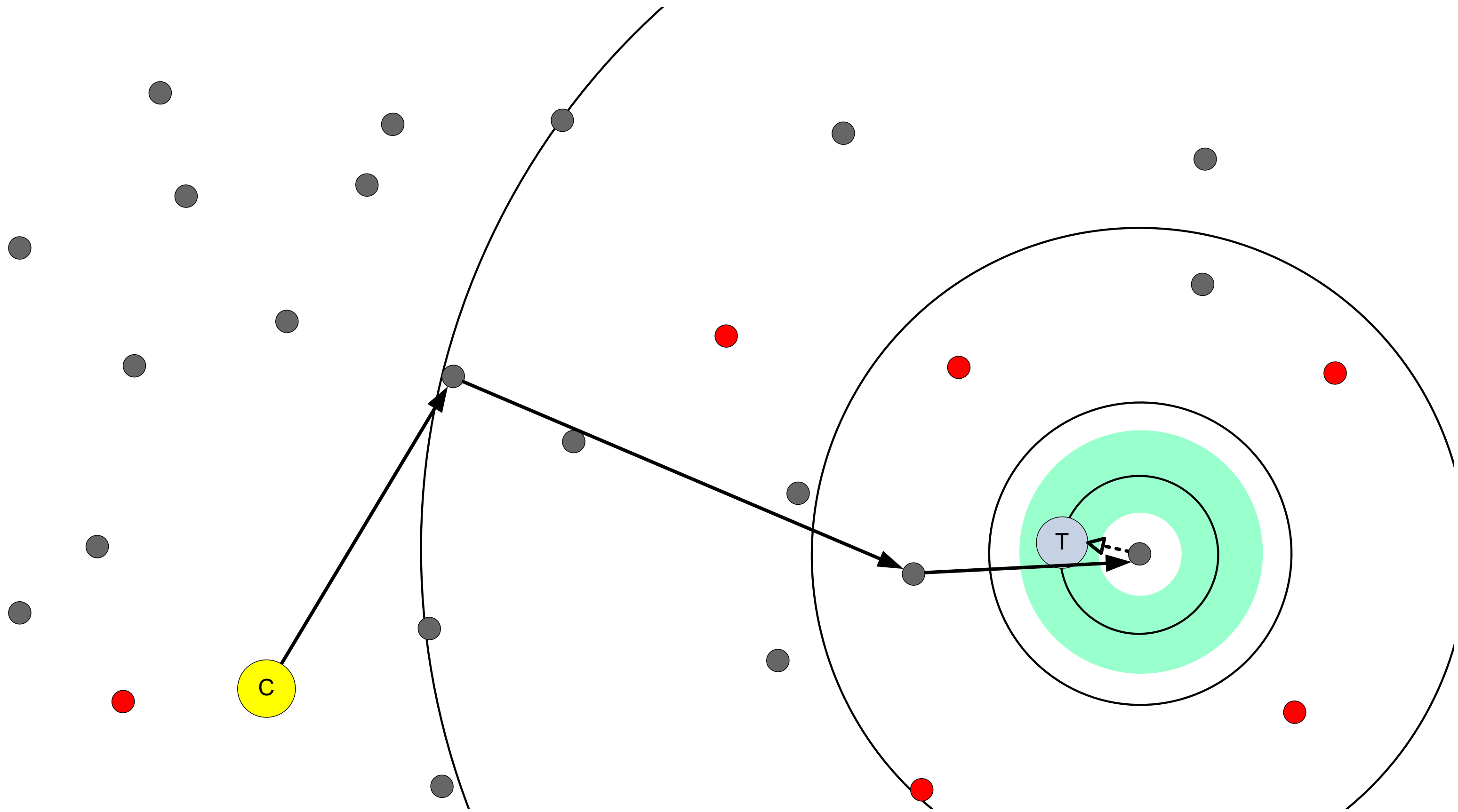
Closest Node Discovery



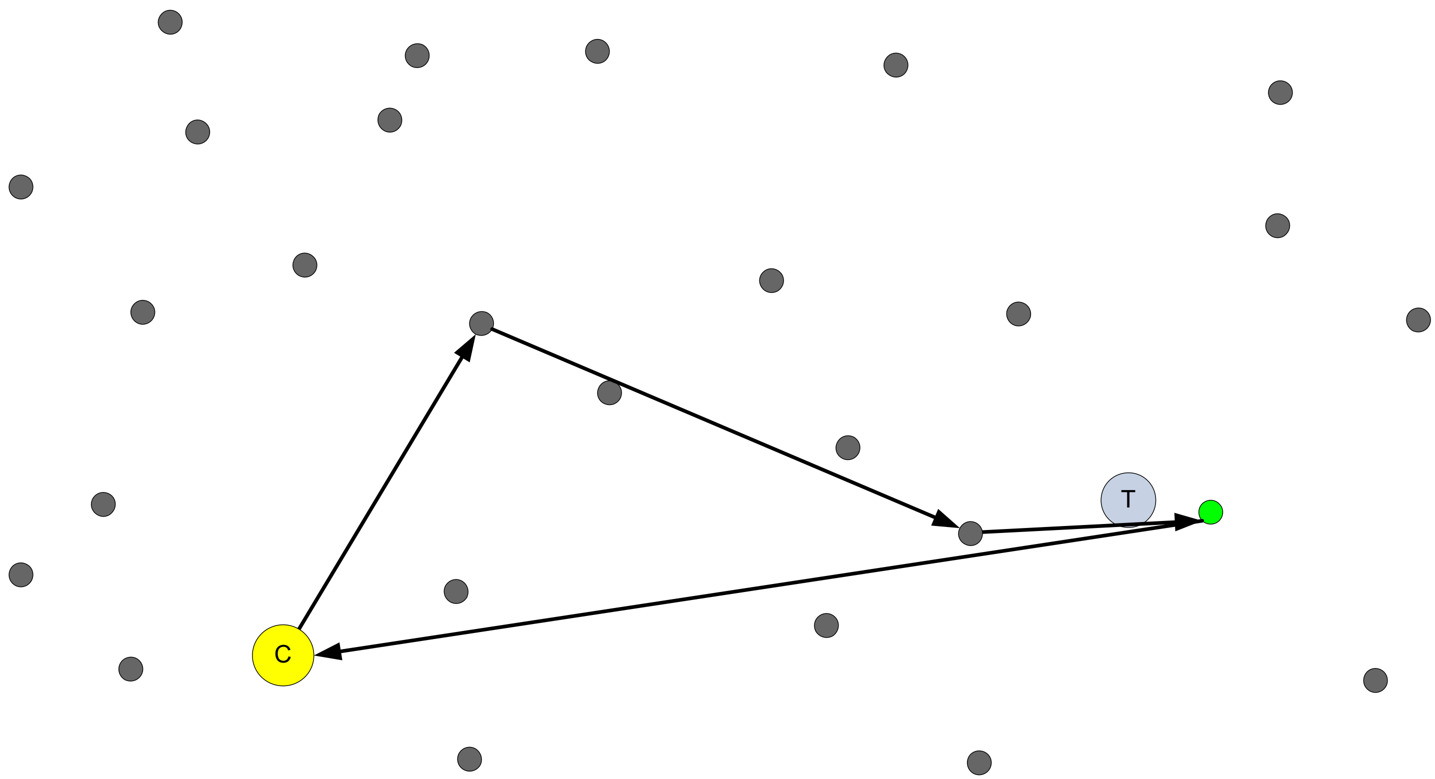
Closest Node Discovery



Closest Node Discovery



Closest Node Discovery



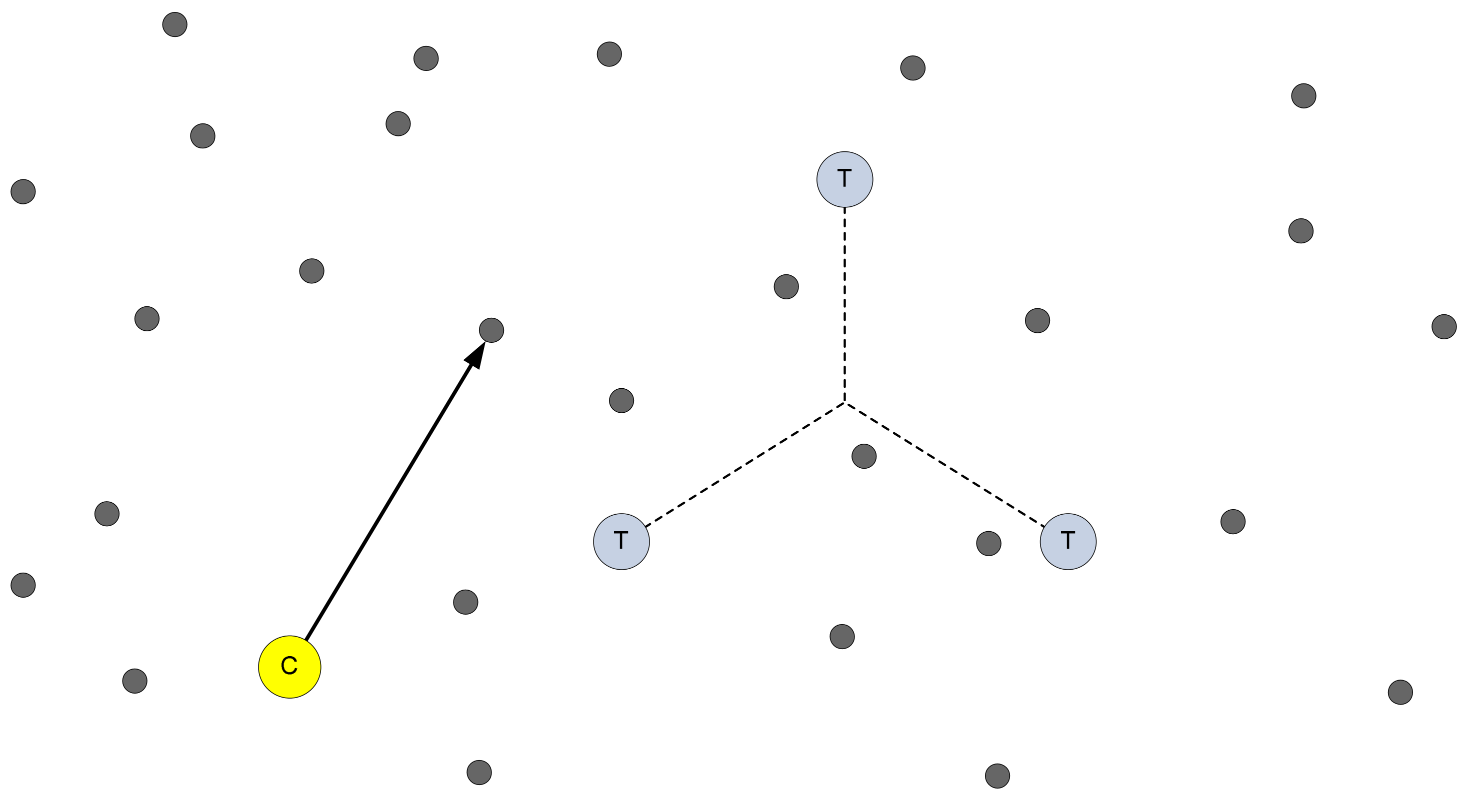
Meridian Theoretical Analysis

- Analytical guarantees for closest node discovery
- Meridian can find the closest node with high probability
  - Given nodes in a space with a *doubling* metric
  - As well as a *growth constrained* metric
- Scales well with increasing system size
- Does not lead to hot spots

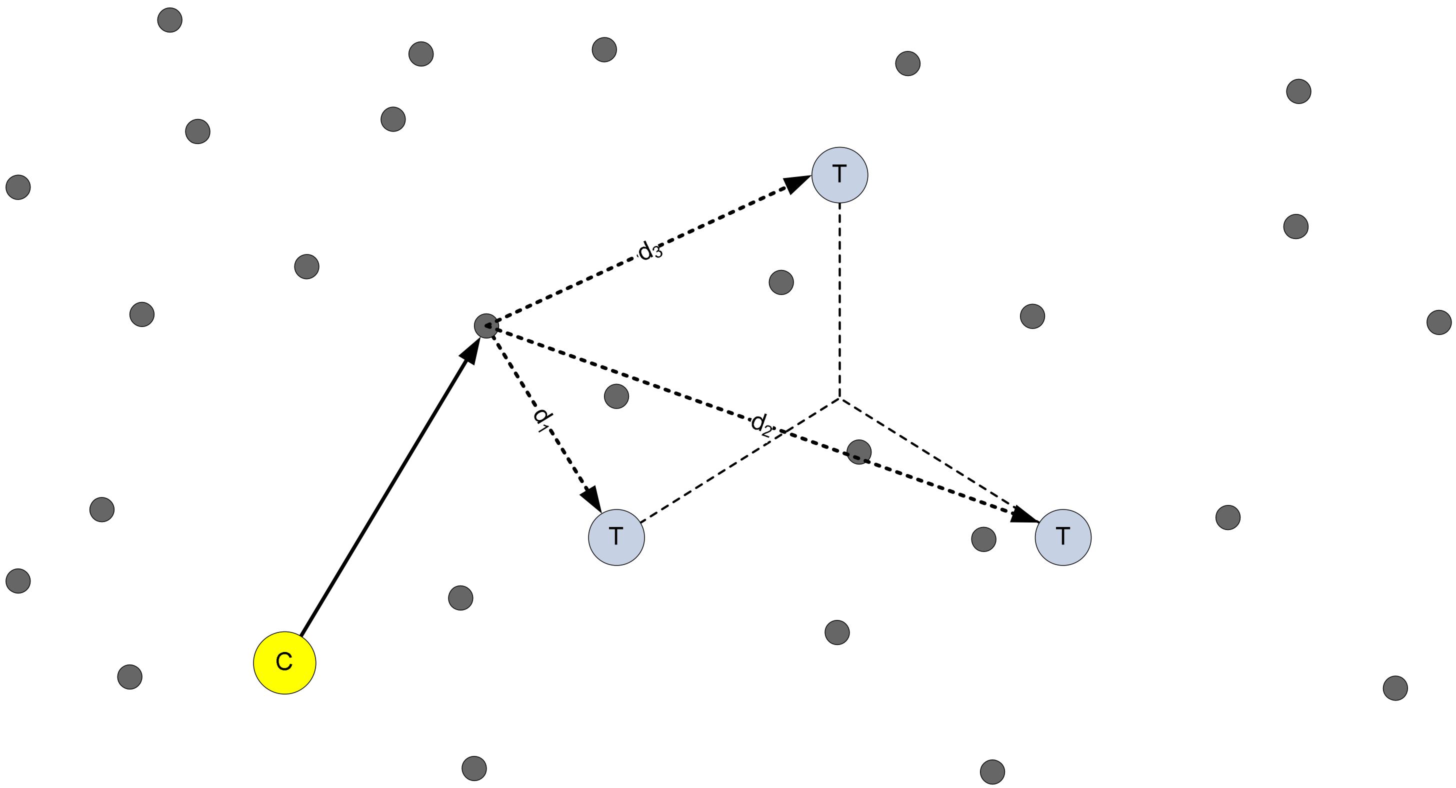
Central Leader Election

- Select the closest node to the center of a set of targets
  - Multi-cast trees can place central nodes higher in the hierarchy
- Algorithm similar to closest node discovery
- Minimizes avg. latency to a set of targets instead of one target
  - Uses distance metric d_{avg} instead of d
- Inter-node latencies of targets not known
  - Need to be conservative in pruning peers

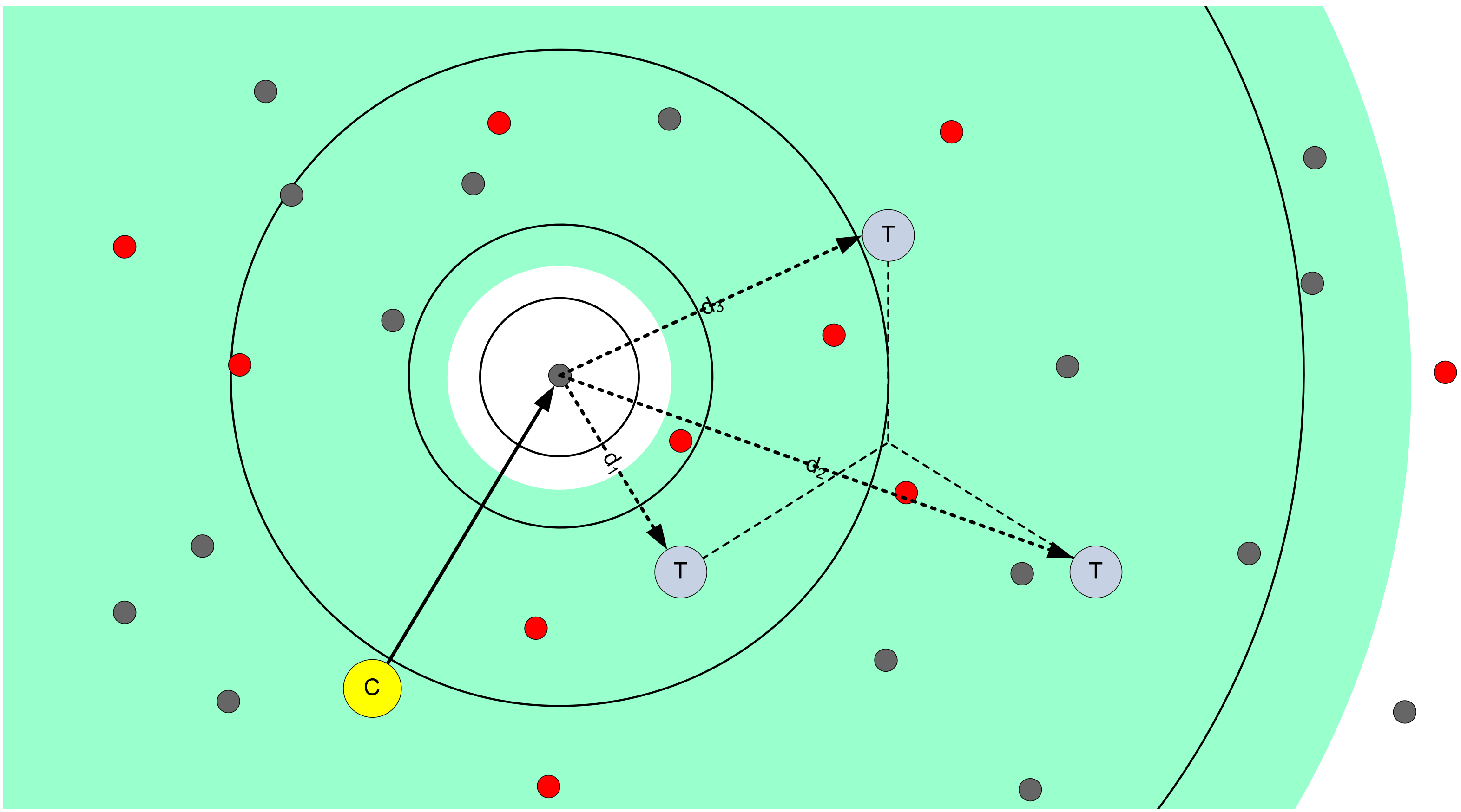
Central Leader Election



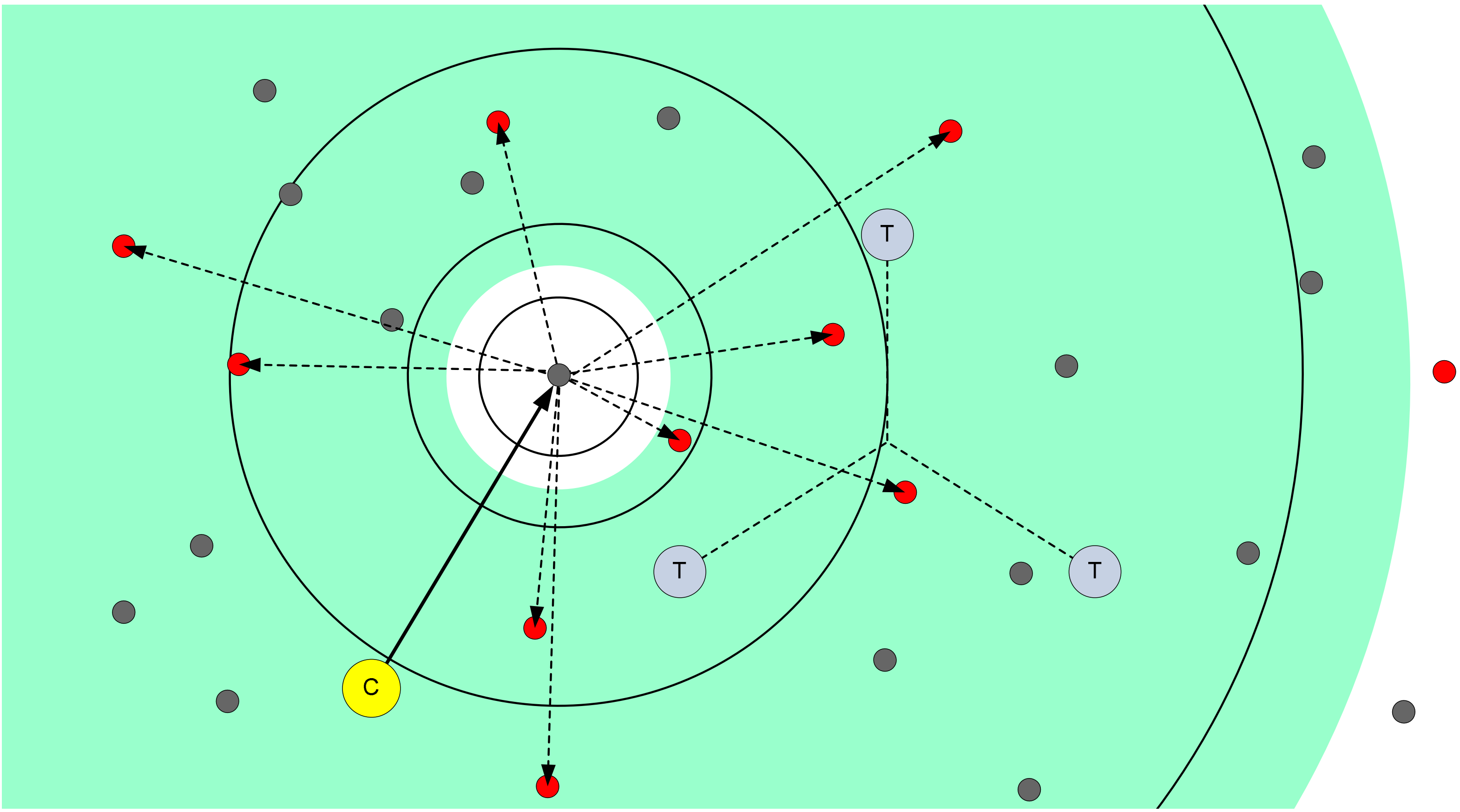
Central Leader Election



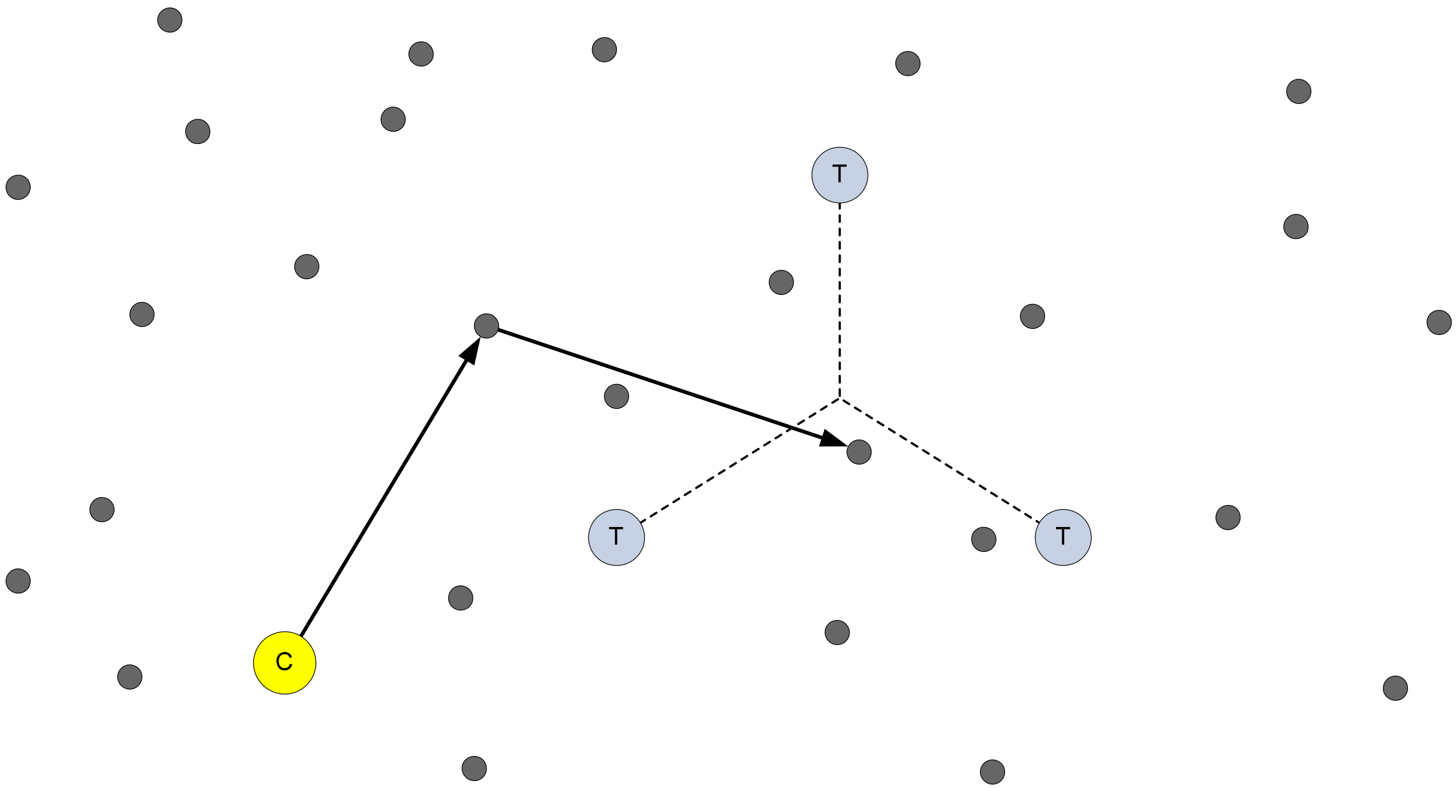
Central Leader Election



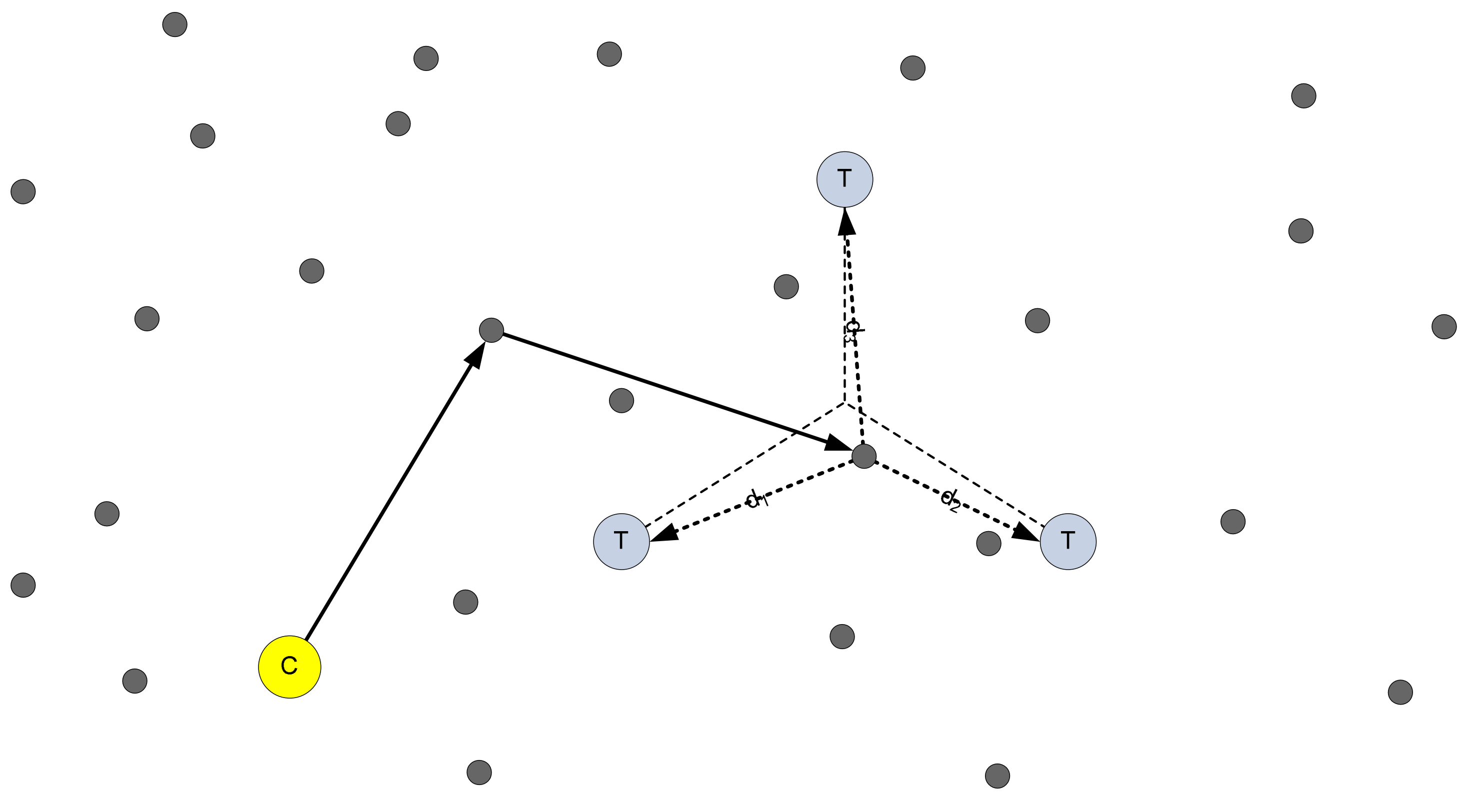
Central Leader Election



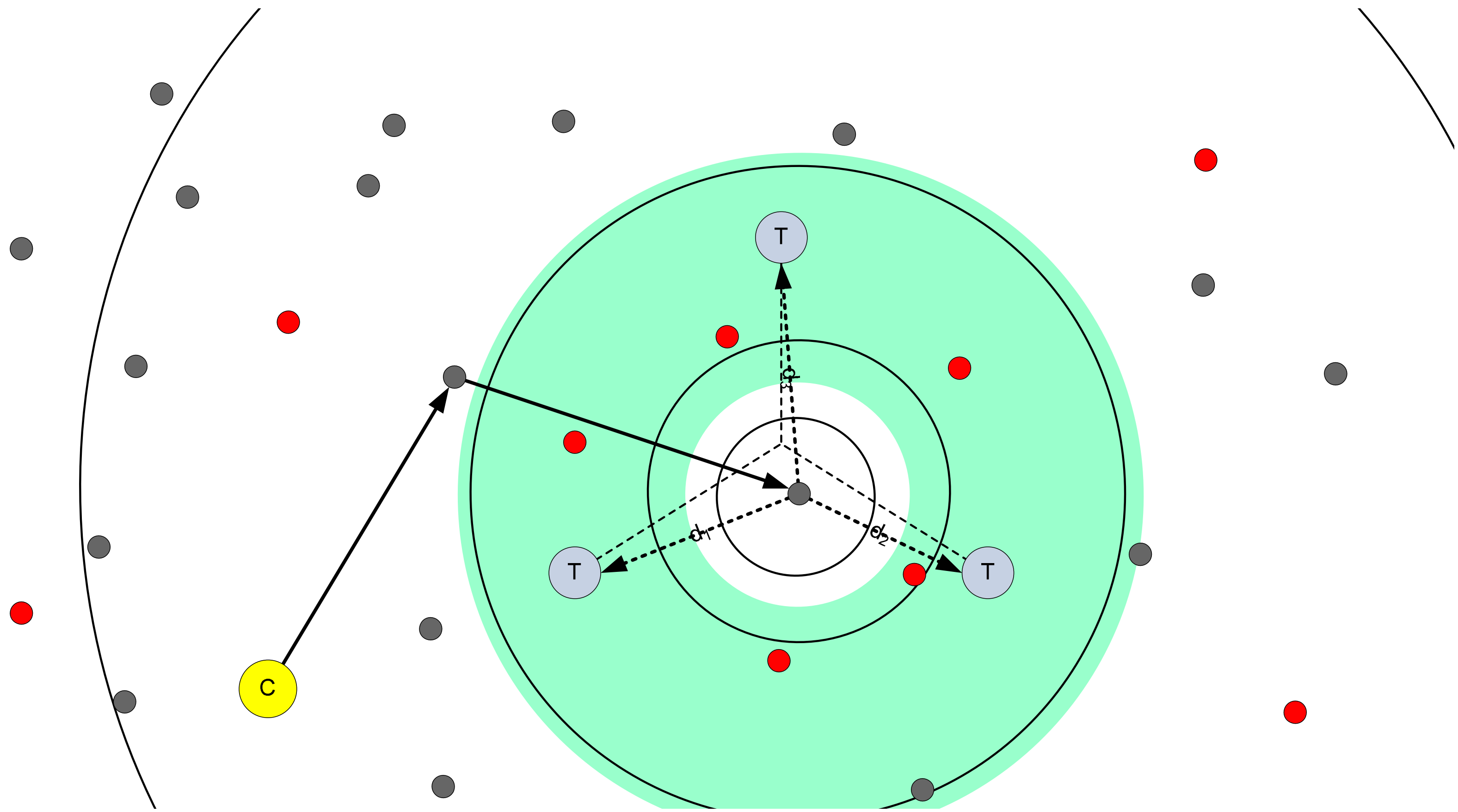
Central Leader Election



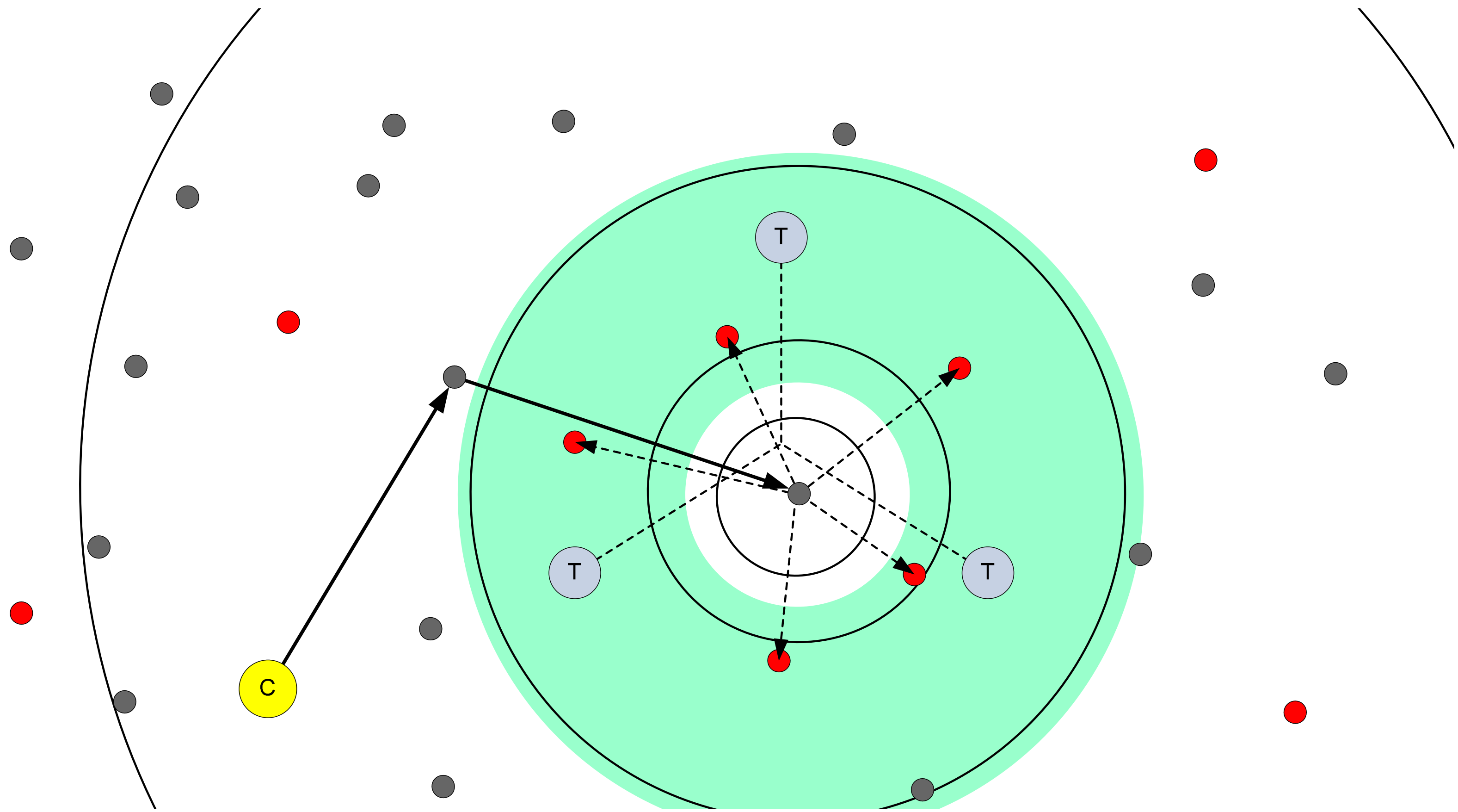
Central Leader Election



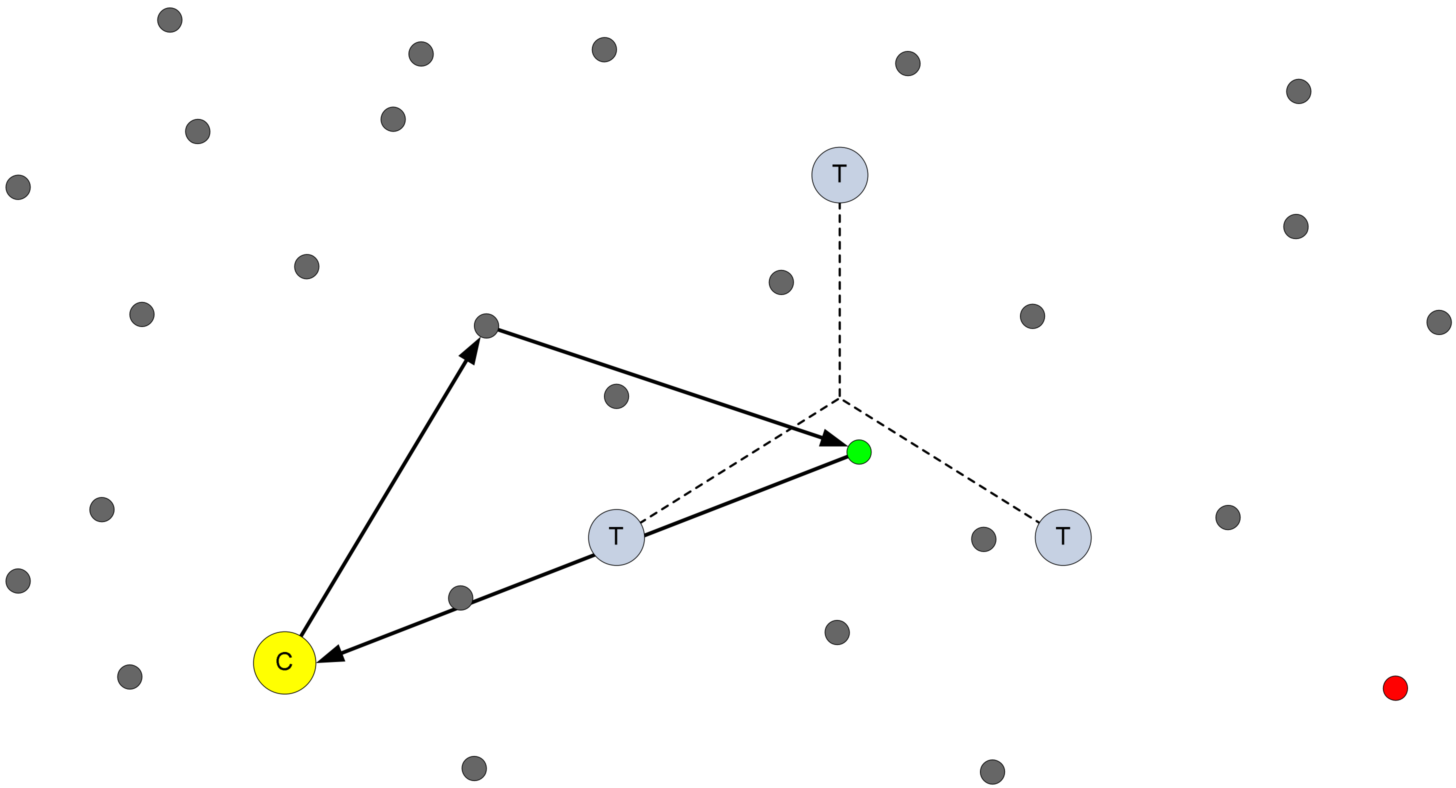
Central Leader Election



Central Leader Election



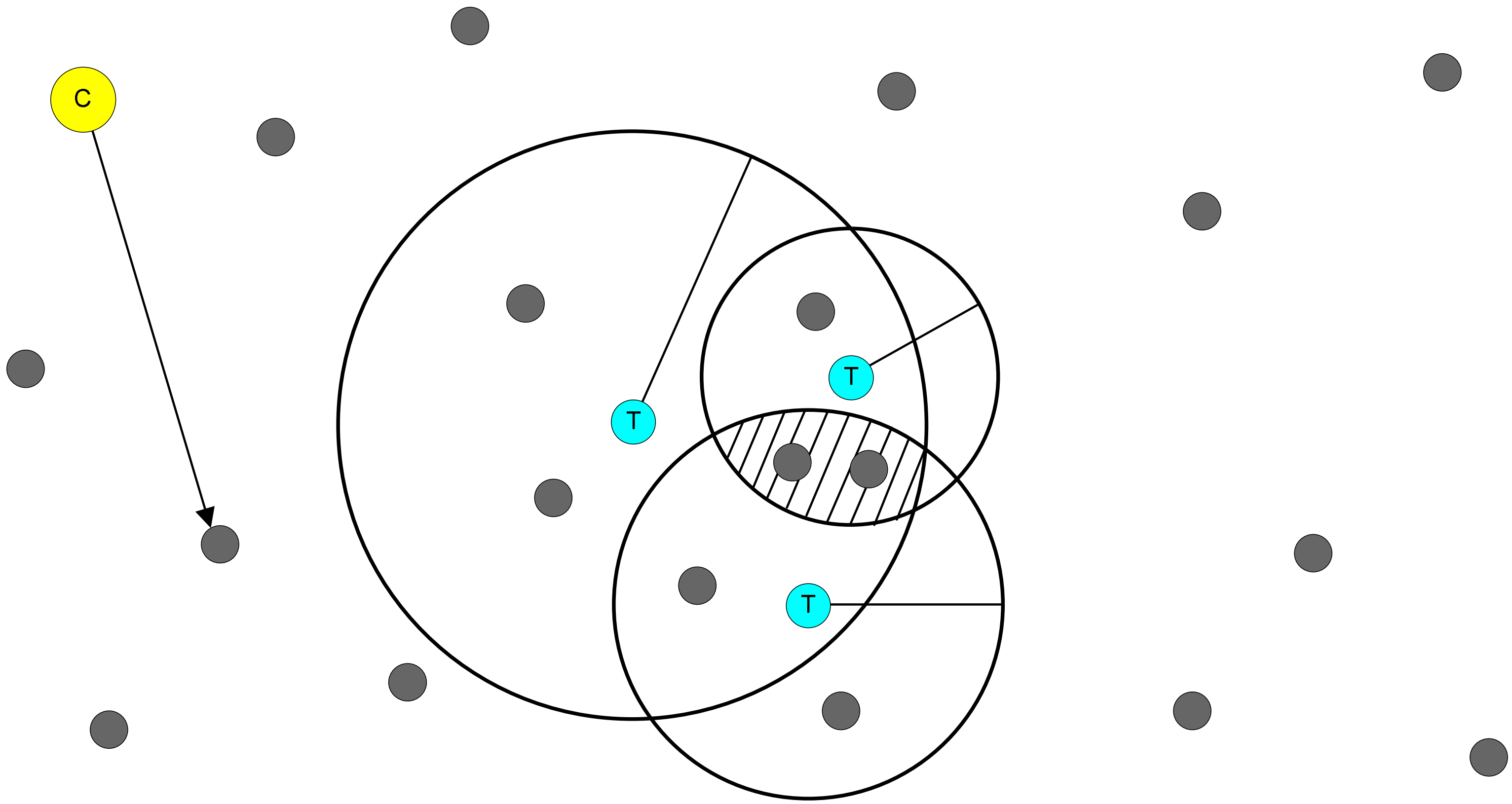
Central Leader Election



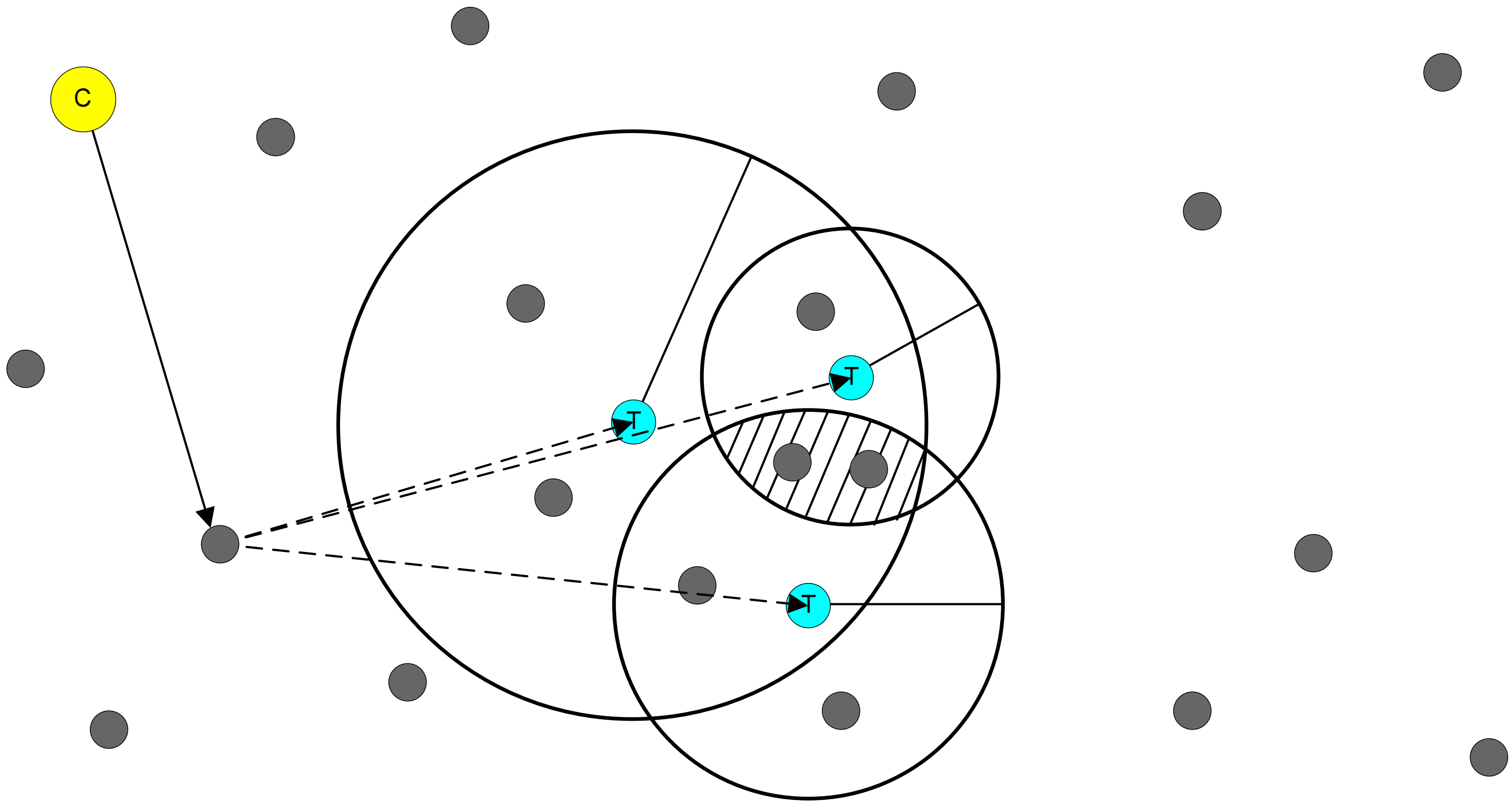
Multi-constraint System

- Find a node that satisfies a set of latency constraints
  - ISP can find a server that can satisfy a SLA with a client
  - Grid users can find a set of nodes with a bounded inter-node latency
- There exists a solution space, containing 0 or more nodes
  - Only a solution point in previous problems
- Requires a different distance metric *s* :
  - $s = \sum_{i=1}^{u} \max(0, d_i - range_i)^2$
  - s = 0 when all constraints are satisfied
  - Sum of squares places more weight on fringe constraints
    - Allows for faster convergence to solution space
  - Other metrics can be used, square works well in practice

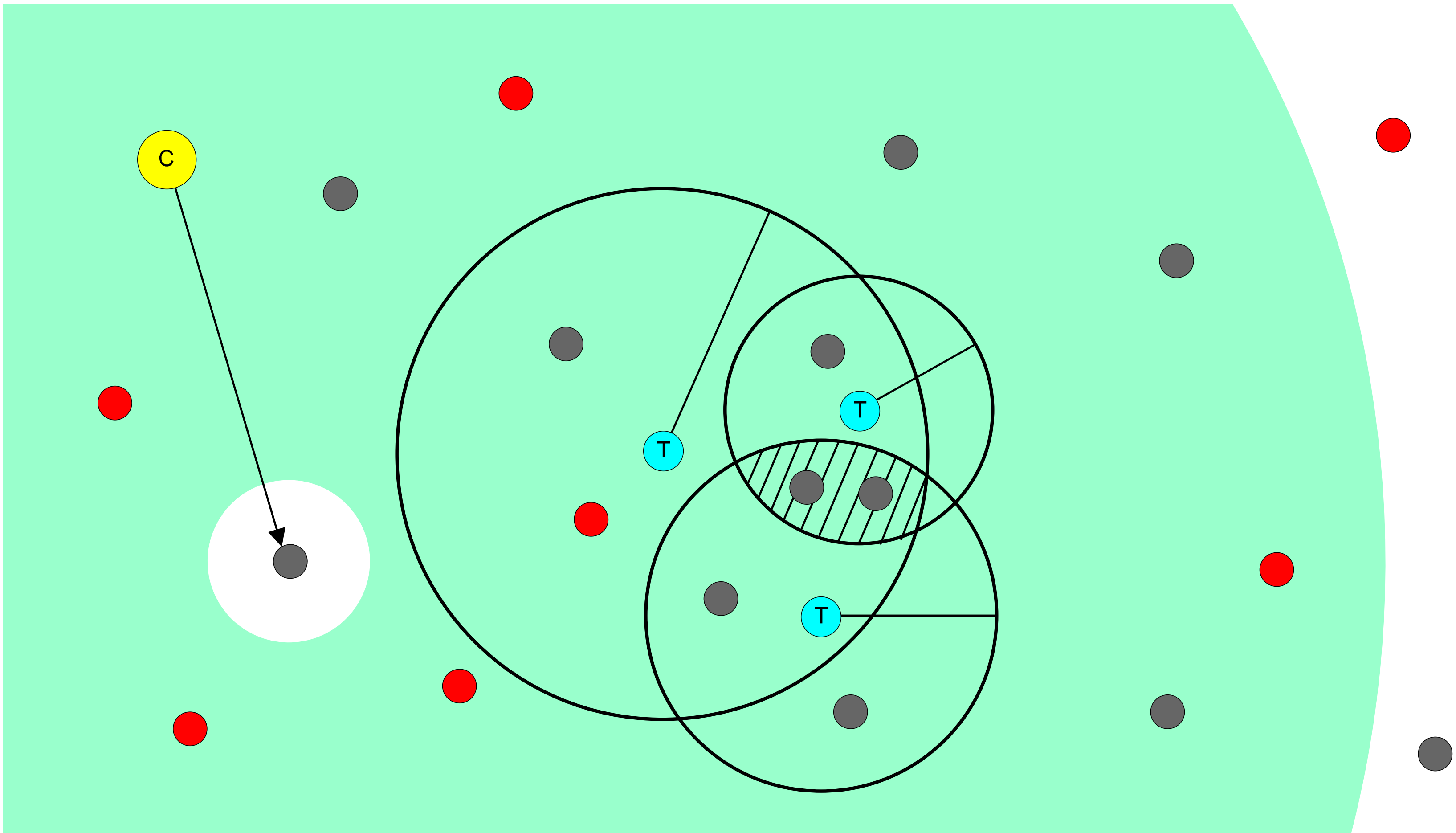
Multi-constraint System



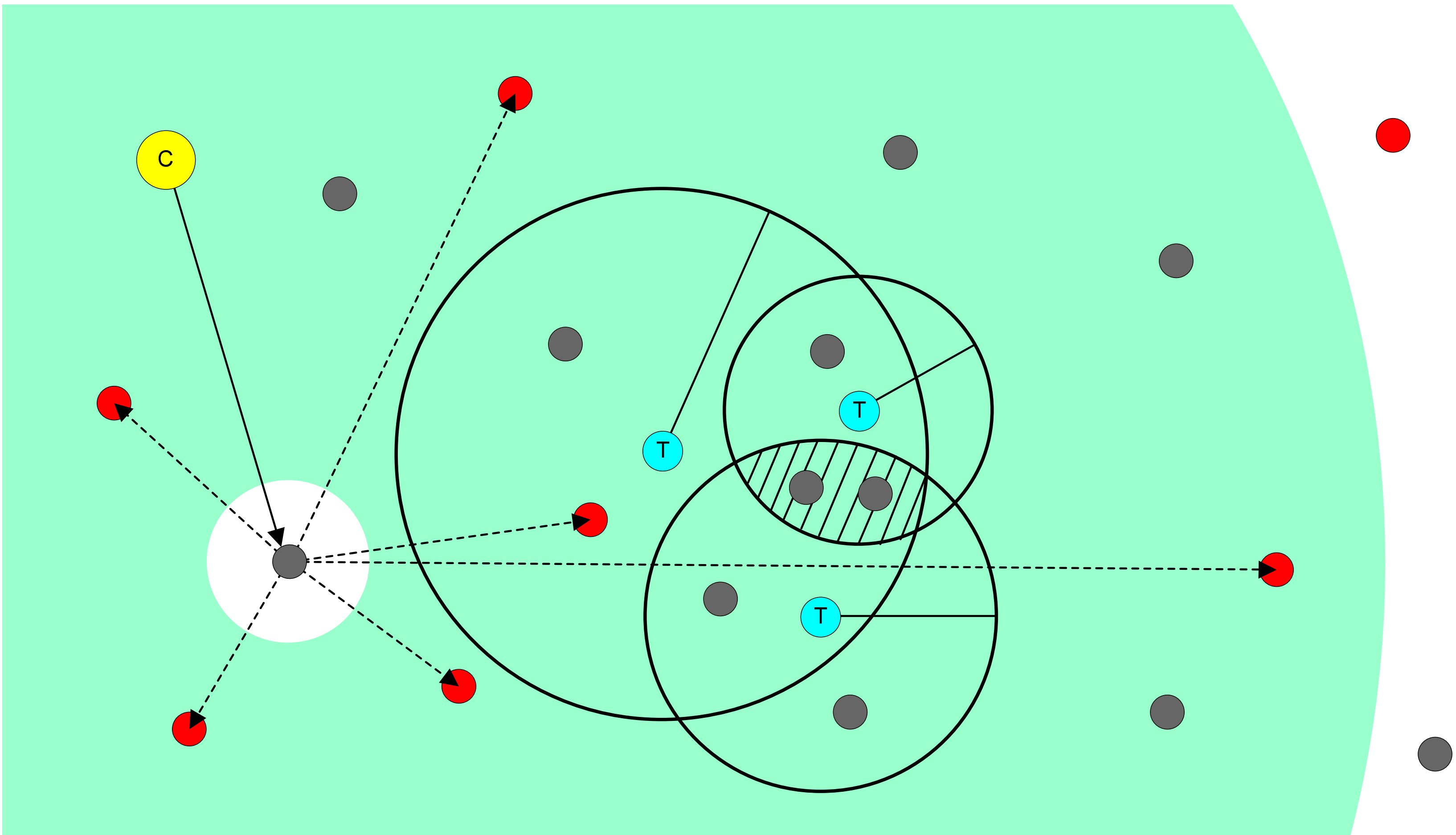
Multi-constraint System



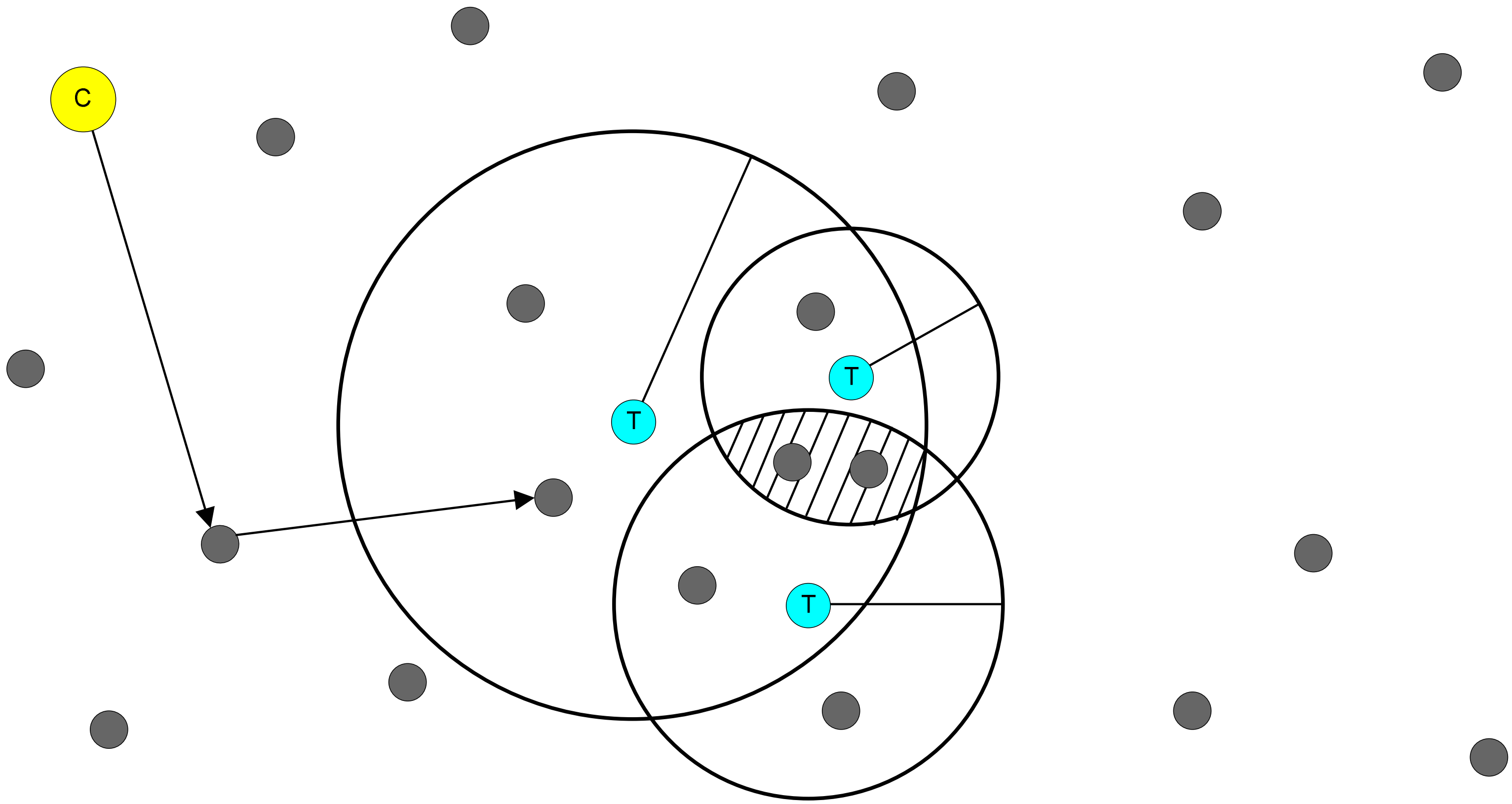
Multi-constraint System



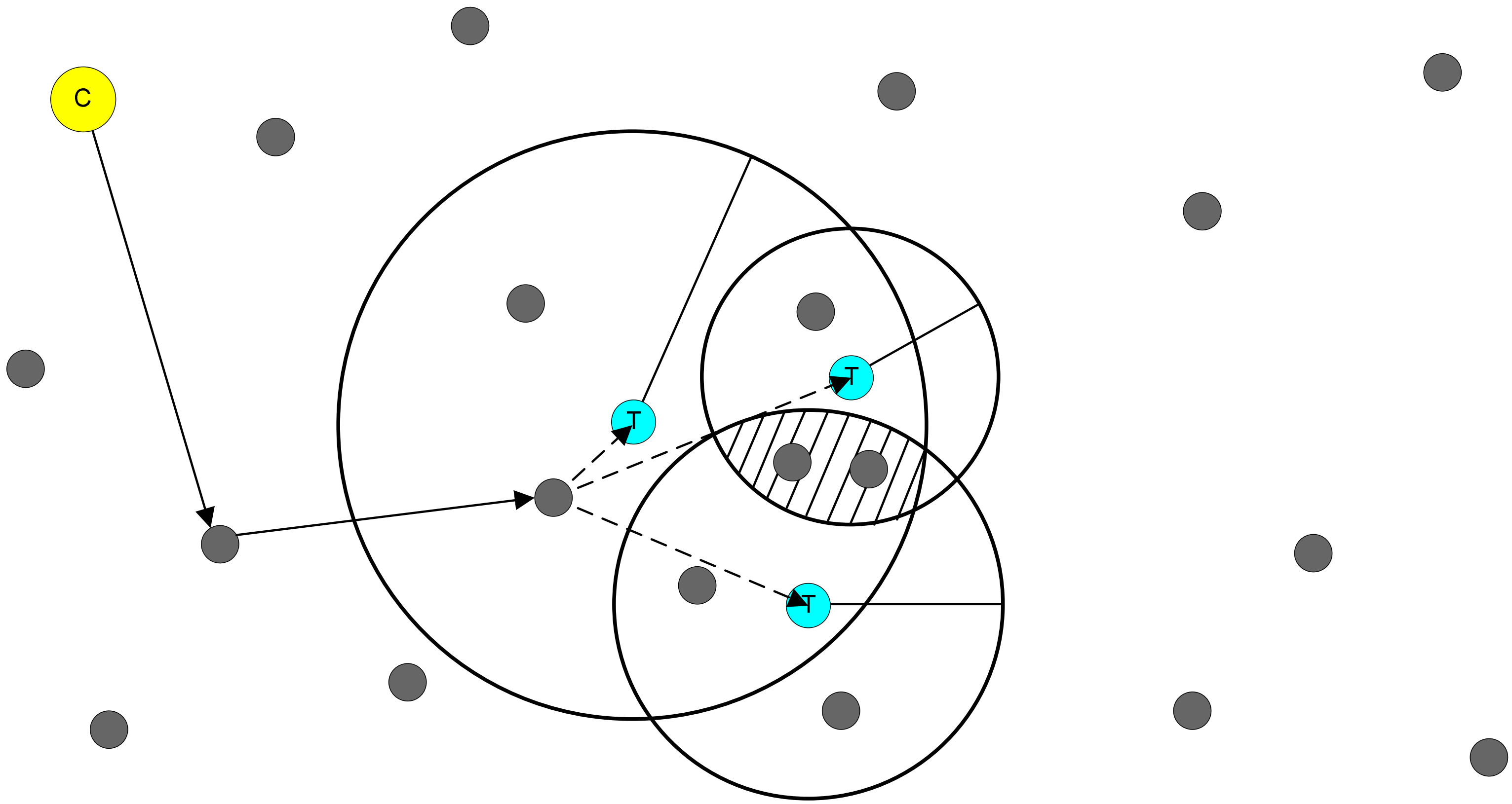
Multi-constraint System



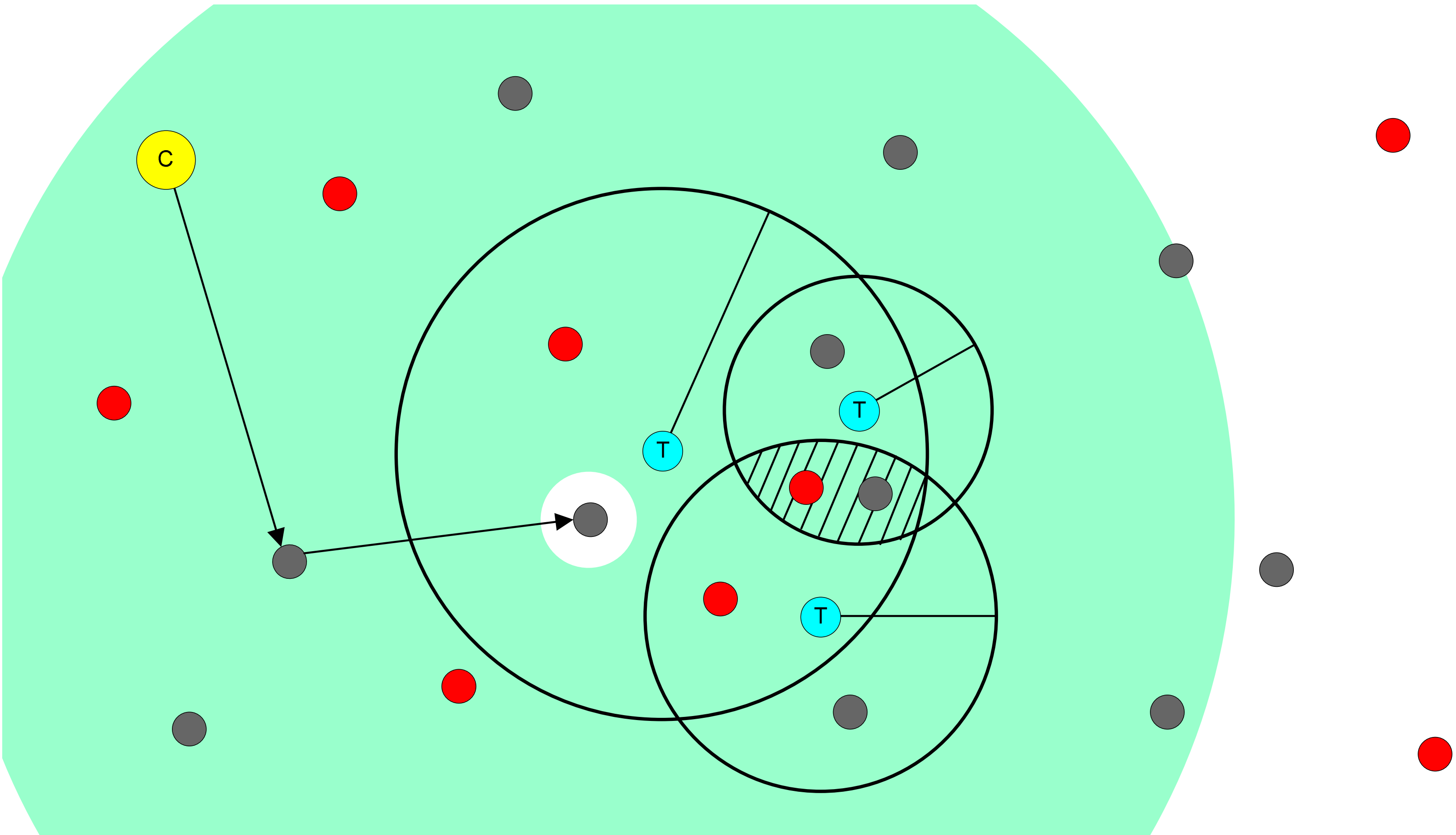
Multi-constraint System



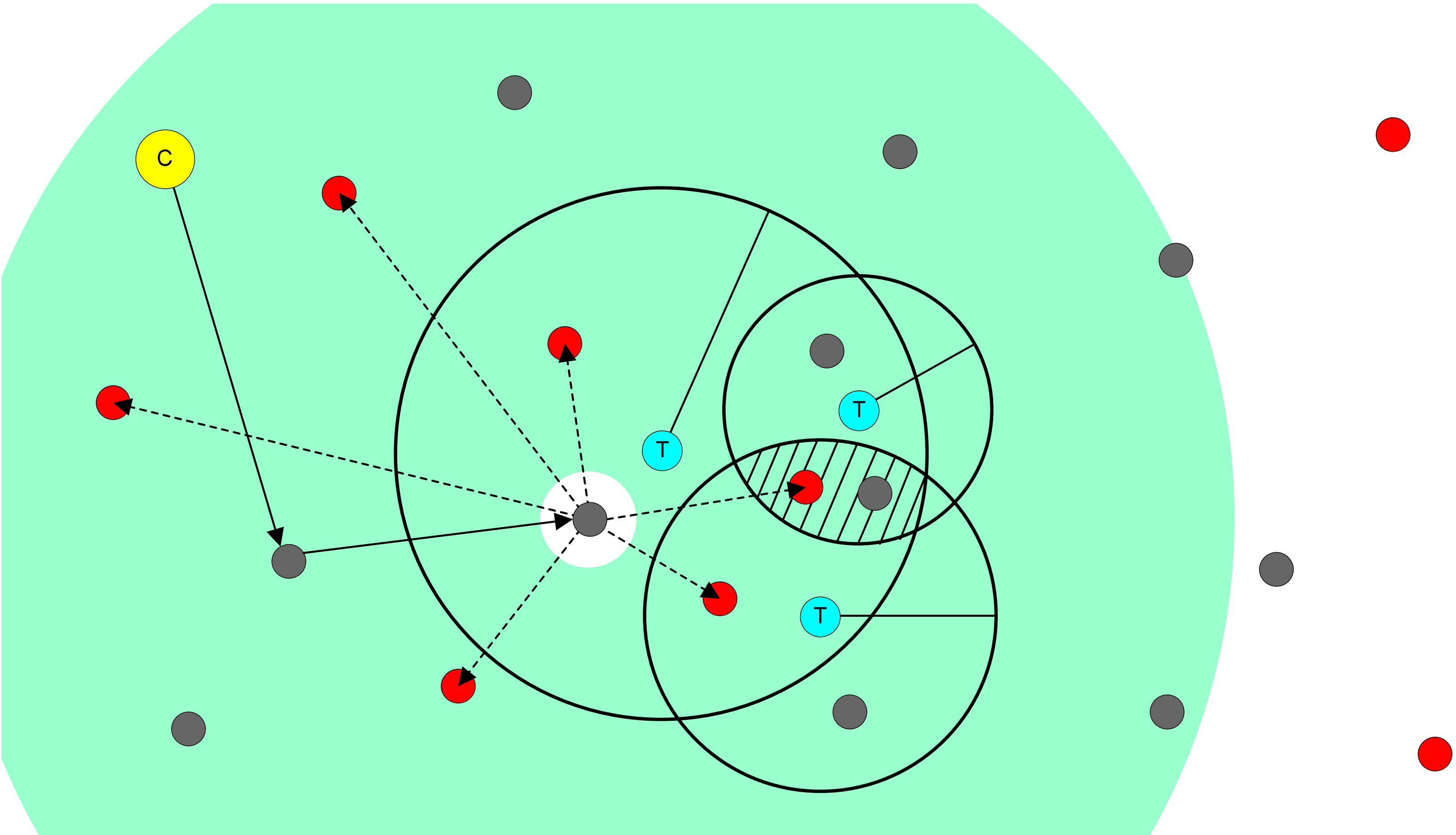
Multi-constraint System



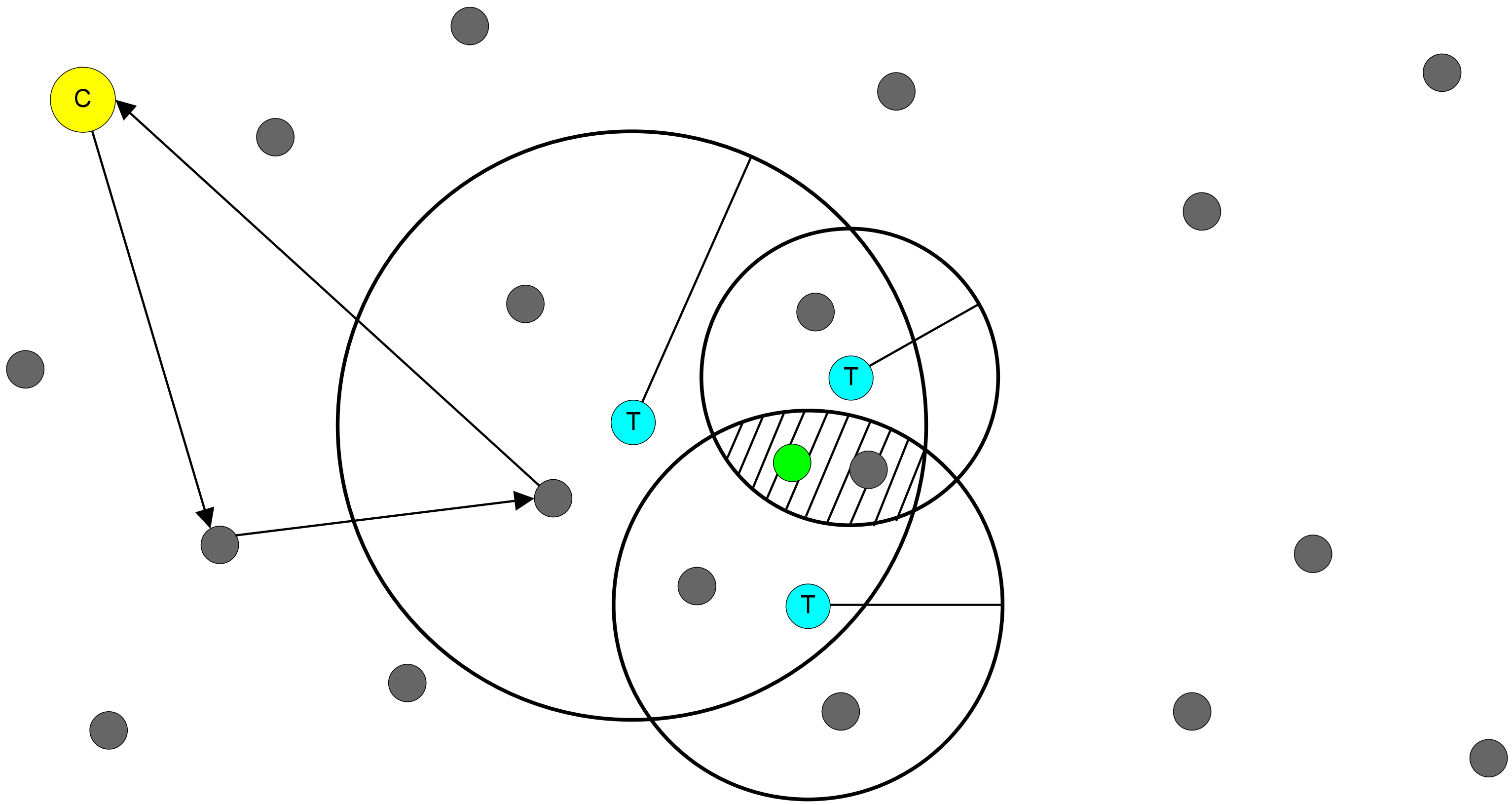
Multi-constraint System



Multi-constraint System



Multi-constraint System



Meridian Query Language

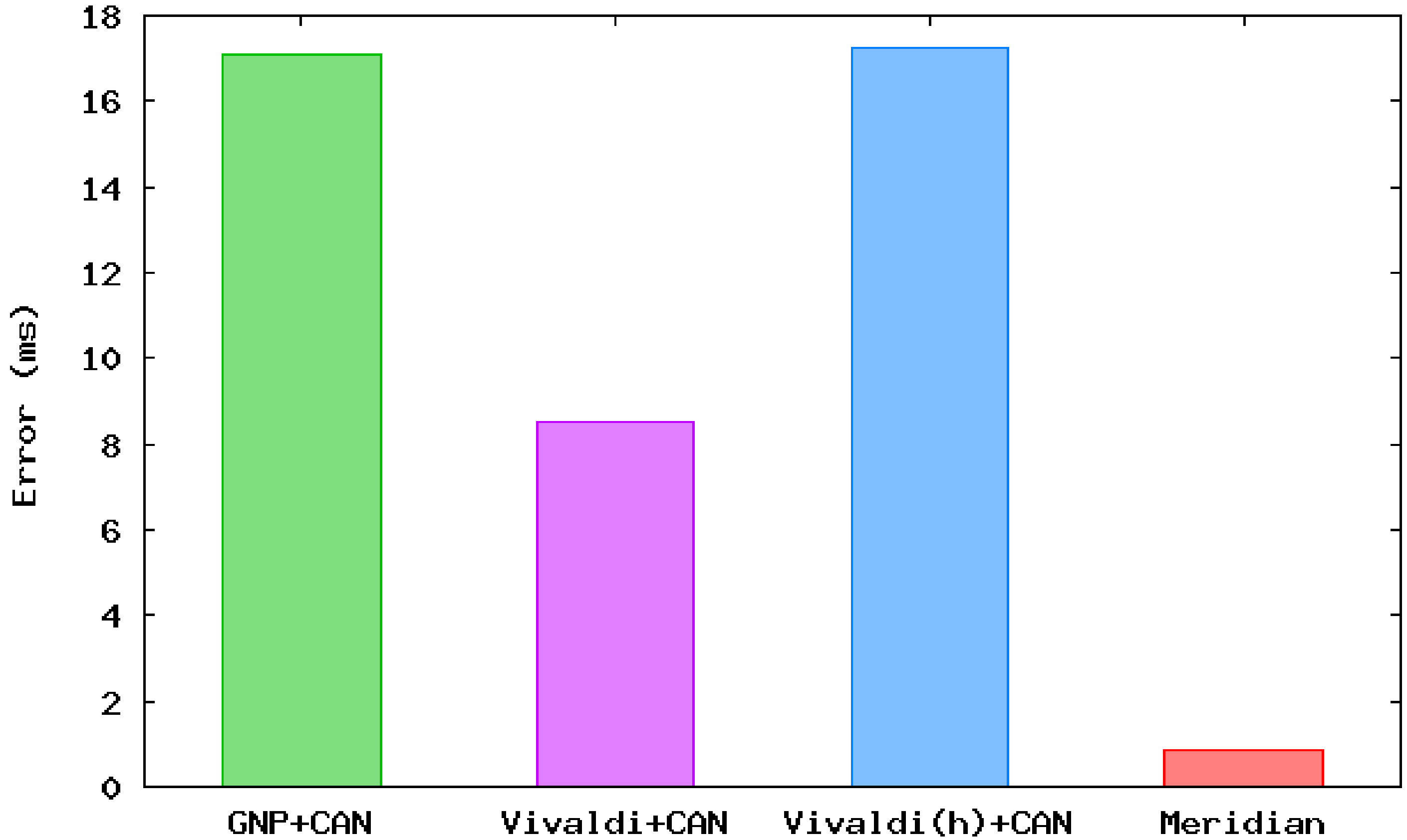
- Variant of C/Python
  - Safe, polymorphic, and dynamically-typed
  - Includes an extensive set of library functions
- Allows users to:
  - Access multi-resolution rings
  - Issue latency probes
  - Forward queries to peers
- Tight resource limits on:
  - Execution time of query
  - Number of hops
  - Amount of memory allocated

Evaluation

- Evaluated our system through a large scale simulation and a PlanetLab deployment
- Simulation parameterized by real latency measurements
  - 2500 DNS servers, latency between 6.25 million node pairs
  - DNS servers are authorities name servers for domains found in the Yahoo! web directory
- We evaluated system sizes of up to 2000 nodes
  - 500 nodes reserved as targets

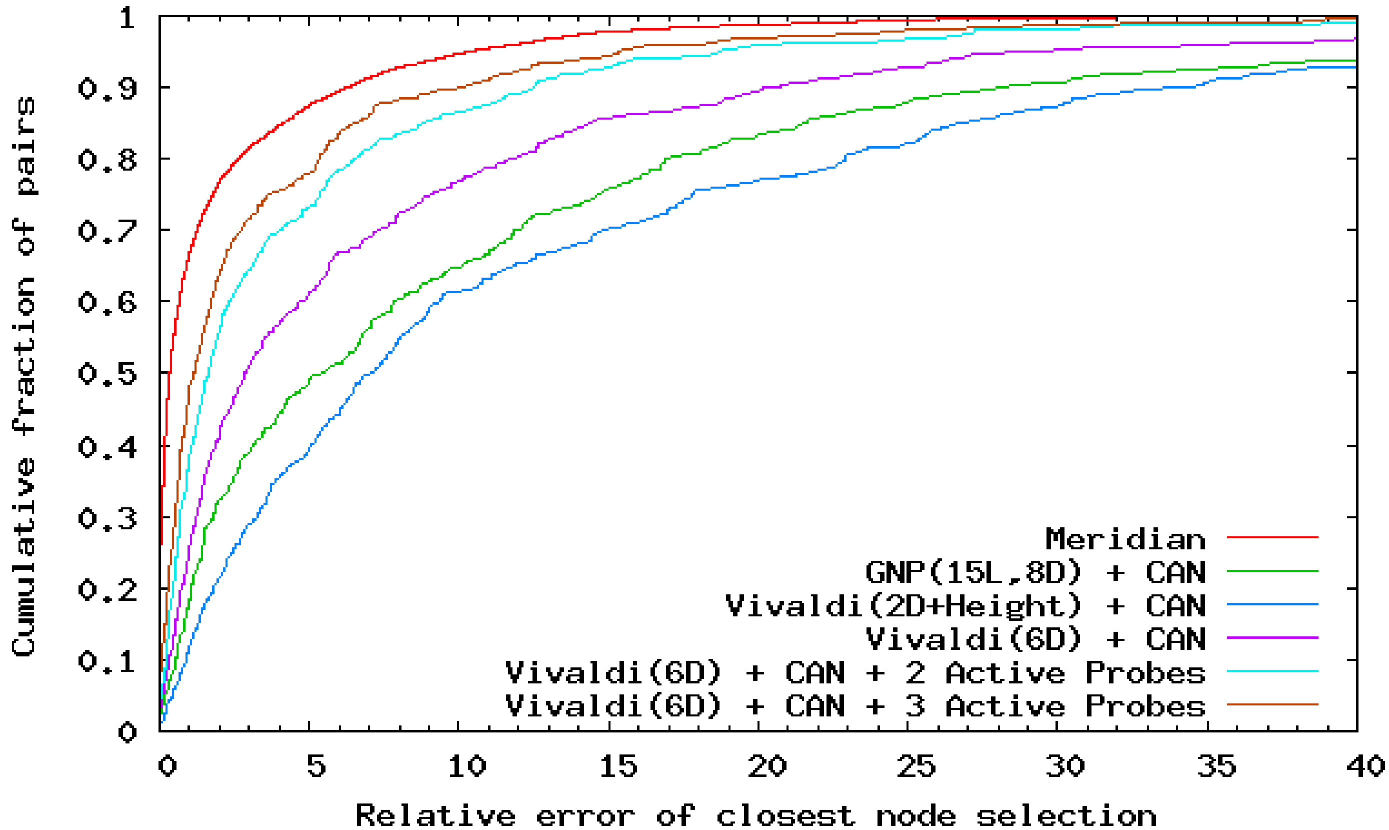
Evaluation: Closest Node Discovery

- Meridian has an order of magnitude less error than virtual coordinate schemes



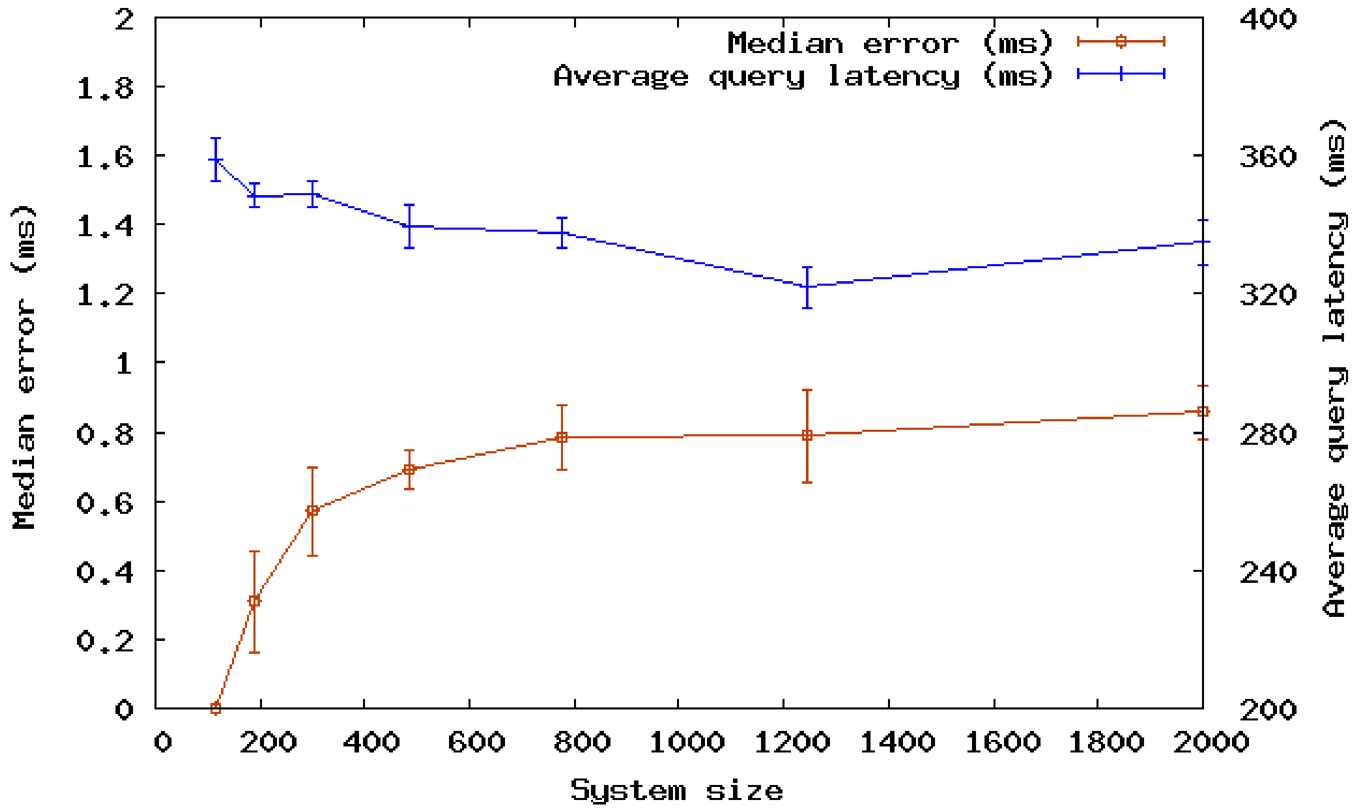
Evaluation: Closest Node Discovery

- CDF of relative error shows Meridian is more accurate for both typical nodes and fringe nodes



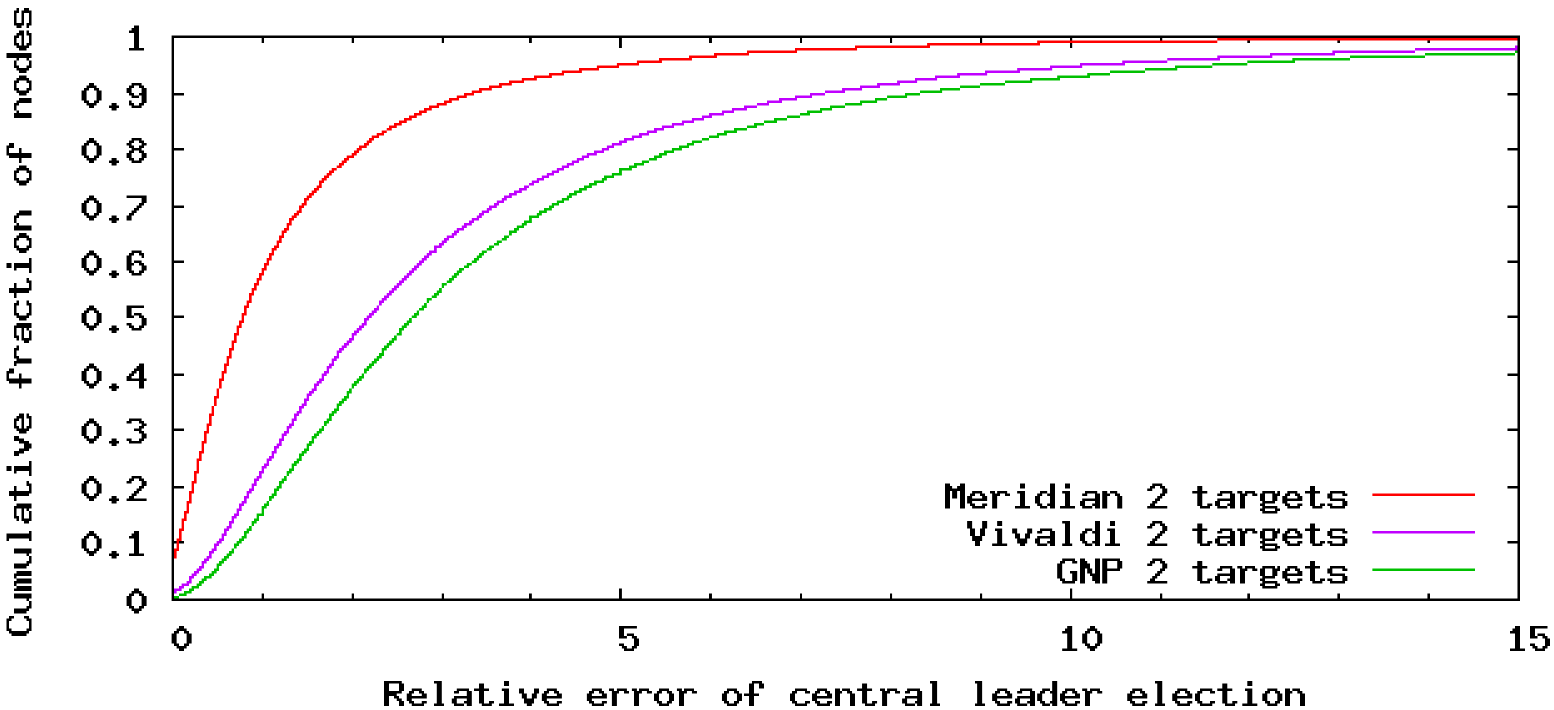
Evaluation: Closest Node Discovery

- With $k = \lfloor \log_{1.6} N \rfloor$, error and query latency remain constant as N increases
- Average query latency determined by slowest node in each ring



Evaluation: Central Leader Election

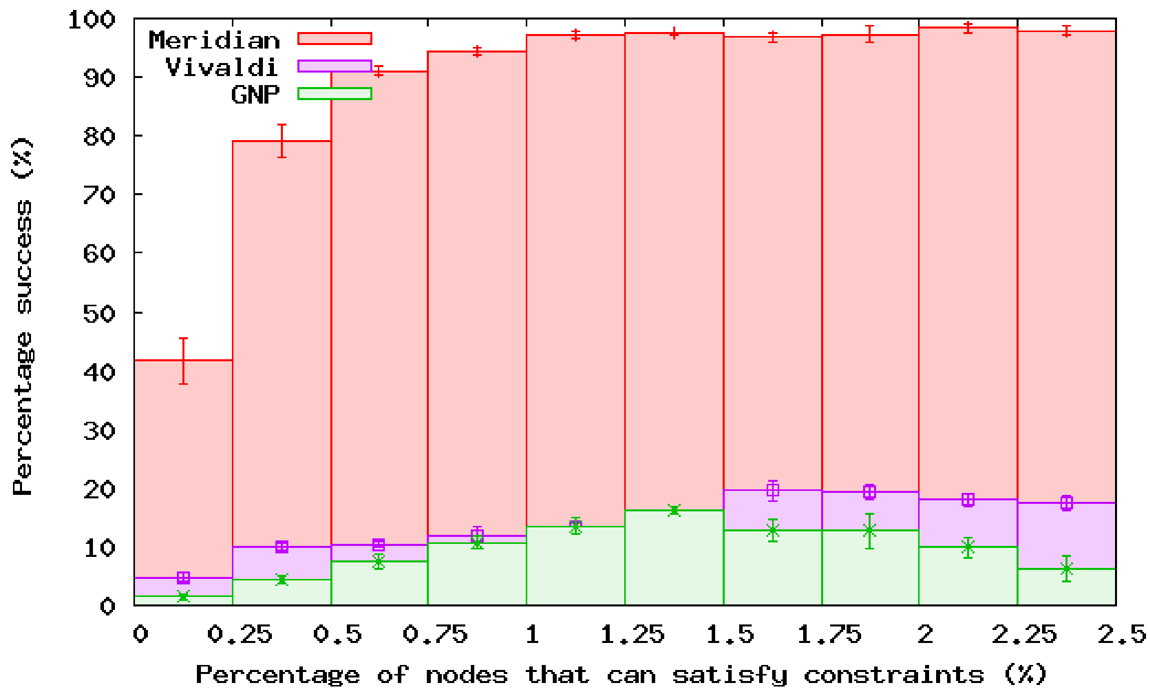
- Meridian incurs significantly less relative error



Evaluation: Multi-constraint System

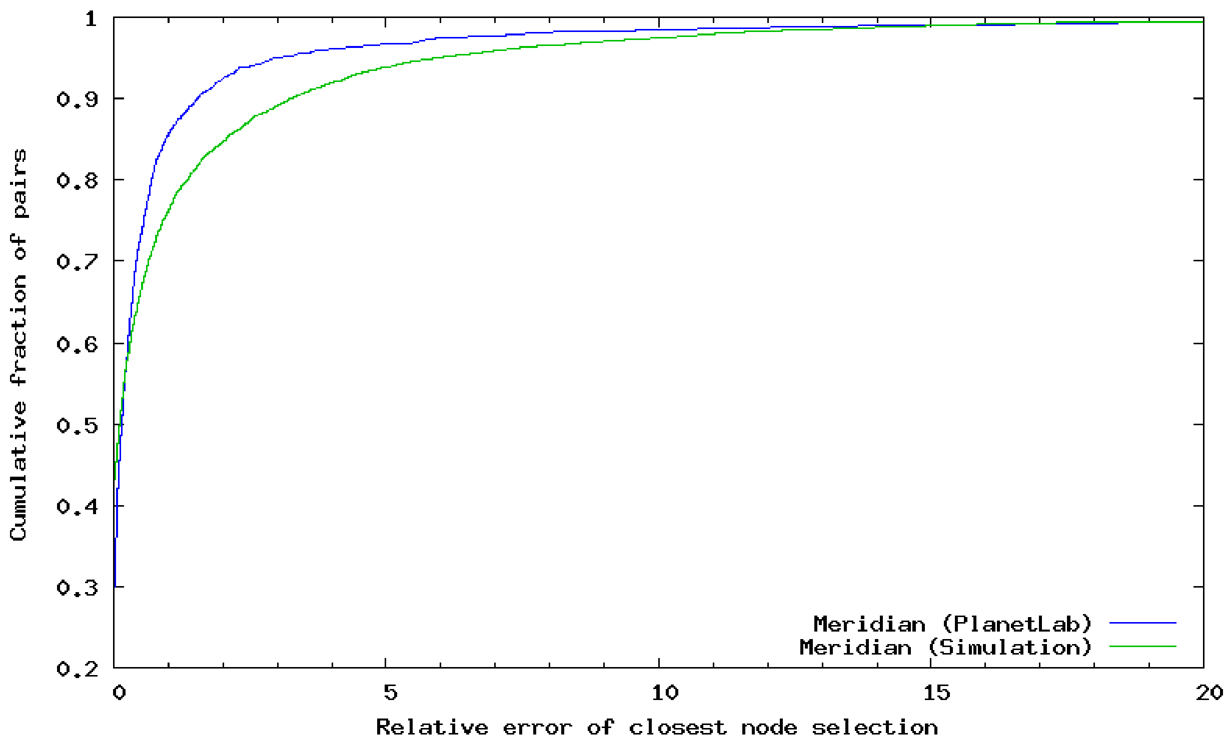
- Categorized multi-constraint queries by its difficulty
  - Difficulty a measure of the number of nodes in solution space
- Success rate for queries that can be satisfied by only 0.5% of the nodes:

2 Constraints		3 Constraints		4 Constraints	
Meridian: 91%	VC: 35%	Meridian: 90%	VC: 19%	Meridian: 91%	VC: 11%



Evaluation: PlanetLab Deployment

- A PlanetLab deployment of 166 nodes shows the closest node discovery accuracy to be very close to the simulation results
- Have expanded deployment to 325 PlanetLab nodes supporting all 3 applications and MQL



Implementation

- Includes query language and the 3 protocols
- Works with firewalled hosts
- Can use DNS queries, TCP connect times, and Meridian UDP packets to measure latency
- Optimizations:
  - Measurement cache reduces query latency
  - Ring management scheme to select more diverse peers

ClosestNode.com

- ClosestNode.com is a DNS redirection service that returns the IP address of closest node to the client
  - e.g. cobweb.closestnode.com will resolve to the closest *CobWeb* DHT node to the requesting client
- Requires minimal changes to the service
  - Linking the Meridian library and calling one function at startup
  - Or add standalone Meridian server to start script
- No changes required for the client
- Can register your service at:
  - http://www.closestnode.com

Conclusions

- A lightweight accurate system for selecting nodes
- Combines query routing with active measurements
- An order of magnitude less error than virtual coordinates
- Solves the network location problem directly
  - Does not need to be paired with CAN
- Code, data, demos and more information at

  http://www.cs.cornell.edu/People/egs/meridian