Meridian: A Lightweight Framework for Network Positioning without Virtual Coordinates

Bernard Wong, Aleksandrs Slivkins, Emin Gün Sirer

Patrick Moor

ETH Zürich

January 18, 2006

Outline

Outline

Outline

Outline

Outline

Meridian: A Lightweight Framework for Network Positioning without Virtual Coordinates

Written by Bernard Wong, Aleksandrs Slivkins and Emin Gün Sirer from Cornell University in February 2005.

Meridian: A Lightweight Framework for Network Positioning without Virtual Coordinates

Written by Bernard Wong, Aleksandrs Slivkins and Emin Gün Sirer from Cornell University in February 2005.

Principal Goal

Selecting nodes based on their position in the network.

Applications

- Closest node discovery
- Central leader election
- Target latency constraint systems

Meridian: A Lightweight Framework for Network Positioning without Virtual Coordinates

Written by Bernard Wong, Aleksandrs Slivkins and Emin Gün Sirer from Cornell University in February 2005.

Real Coordinates

- Designated landmark nodes with known position. Non-landmark nodes try to estimate their position using some fancy algorithms based on their latencies to the landmark nodes.
- GPS

Meridian: A Lightweight Framework for Network Positioning without Virtual Coordinates

Written by Bernard Wong, Aleksandrs Slivkins and Emin Gün Sirer from Cornell University in February 2005.

Virtual Coordinates

Use mathematical operations to embed the high-dimensional space of node-to-node latencies into a virtual coordinate space. These virtual coordinates can then be used as if they were real ones. Usually these algorithms introduce significant errors and even worse, they need a global view of the network. Also, they often need to re-calculate the whole embedding once new nodes join and old ones leave.

Meridian: A Lightweight Framework for Network Positioning without Virtual Coordinates

Written by Bernard Wong, Aleksandrs Slivkins and Emin Gün Sirer from Cornell University in February 2005.

Lightweight

Try to keep the space usage at a node as low as possible, ideally constant. The communication overhead should be as low as possible and the network should be flexible enough to adjust rapidly when nodes join or leave.

Meridian...

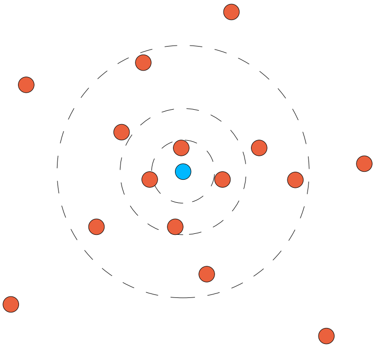
- is a loosely-structured overlay network
- uses direct latency measurements instead of an embedding
- does not try to reconcile the local latencies into a globally consistent coordinate space
- delivers high scalability while balancing the load evenly across all nodes
- enables the small-world phenomenon

The Small-World Phenomenon

A hypothesis that everyone in the world can be reached through a short chain of social acquaitances. Experiment conducted by social psychologist Stanley Milgram who found that two random US citizens were connected by an average path length of six.
Can be transferred to networks where the average path length is short.

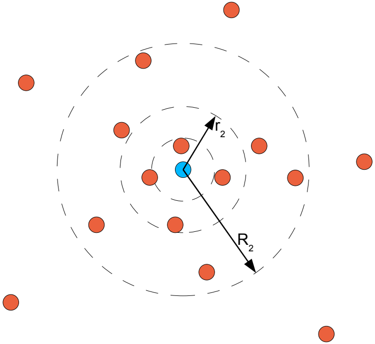
Multi-Resolution Rings

Each Meridian node keeps track of a fixed number of other nodes in the system.
The tracked nodes are put into concentric, non-overlapping rings with exponentially increasing radii.



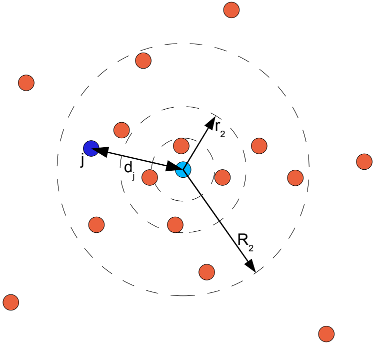
Multi-Resolution Rings

$m > 1$ rings a node manages (fixed)
$r_i = \alpha s^{i-1}$ for $0 < i < m$
$R_i = \alpha s^i$ for $0 \leq i < m - 1$
$r_0 = 0, R_{m-1} = \infty$



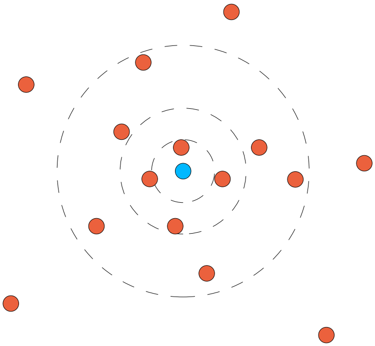
Multi-Resolution Rings

Node measures distance (=latency) d_j to a node j and places that peer in ring i with $r_i < d_j \leq R_i$. Each ring will contain at most k peers. $k = O(\log N)$ is shown to be a good choice.



Multi-Resolution Rings

Favors nearby neighbors (high detail) but also sufficient distant contacts.



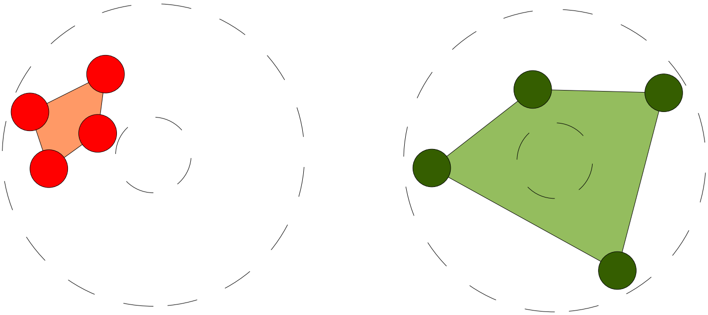
Ring Membership Management

Goals:

- find optimal balance between accuracy and overhead (k nodes per ring)
- geographically distributed ring members
- keep fresh set of nodes (remove old and add new ones quickly)

Geographical Diversity

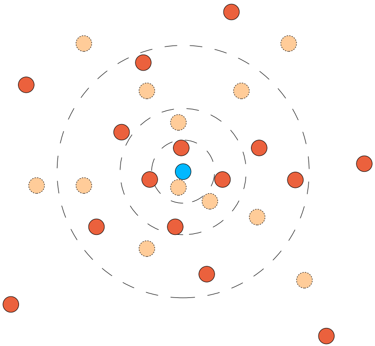
Clustered nodes are useless, so Meridian tries to choose geographically distributed ones.



Geographical Diversity

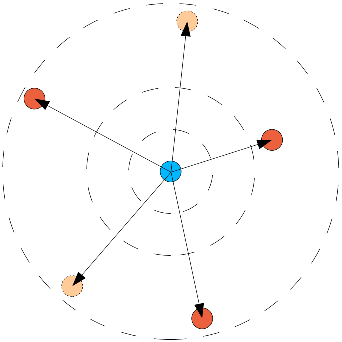
Meridian nodes keep track of k primary and l secondary members per ring.

Nodes periodically re-examine their ring members and choose a primary set with largest diversity.



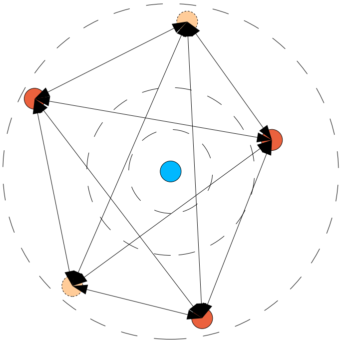
Geographical Diversity

The Meridian node sends a message to every ring member (primary and secondary) and asks for their distances to all the other ring members.



Geographical Diversity

Every node i measures its distance d^i_j to all other nodes j in the same ring and calculates the coordinate tuple $\langle d^i_1, d^i_2, \ldots, d^i_{k+l} \rangle$ where $d^i_i = 0$.

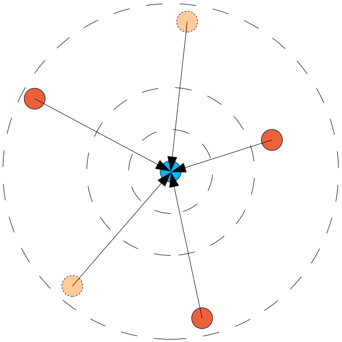


Geographical Diversity

All the tuples are sent back to the central Meridian node.

Message complexity: $2(k+l)+2(k+l)^2$

Assuming 100 byte request packets, 50 byte probe packets and $k = l = \log(2000)$ this results in about 52 KB communication overhead per ring. Over a ring management period of 5 minutes this is less than 180 B/s.



Geographical Diversity

The Meridian node uses a greedy algorithm to determine the most diverse k-node subset:

1. Start with the $k + l$-dimensional polytope spawned by all the $k + l$ tuples.
2. Remove the tuple that yields to minimal volume reduction and also drop that dimension.
3. Do so until only k tuples are left. Those form the new primary node set, the remaining l become the secondary one.

Nodes unreachable during the ring membership managment phase are dropped from the node set.

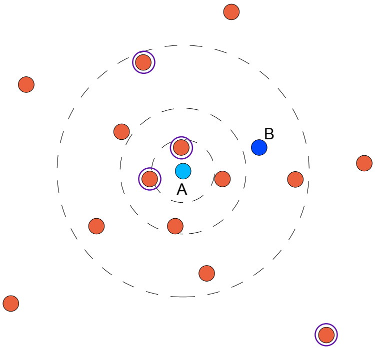
Gossip Protocol

Goal

Each node should discover and mantain a small set of pointers to a sufficiently diverse set of nodes in the network.

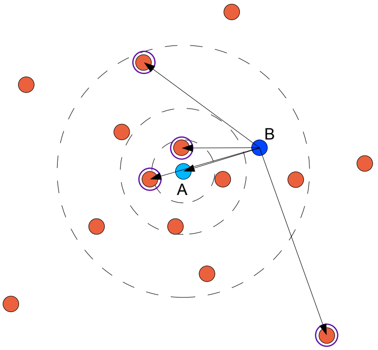
Gossip Protocol

Each node A randomly picks a node B from each of its rings and sends a gossip packet to B containing a randomly chosen node from each of its rings.



Gossip Protocol

On receiving the packet, node B determines through direct probes its latency to A and to each of the nodes contained in the gossip packet from A.
The newly discovered nodes are put into the corresponding rings as secondary members.



Gossip Protocol

Each node is expected to receive m gossip packets and to initiate m^2 probes per gossip period. Further, the node receives m^2 probes from neighbors of its neighbors.

Assuming 9 rings ($m = 9$), a probe packet size of 50 bytes and a gossip packet size of 100 bytes, an average of 21 KB is used per period.

Distributed over 60 second gossip cycles, that's less than 350 B/s and independent of system size!

Initial Gossip

- New nodes need to know at least one address of an existing Meridian node
- They fetch the whole peer set of the existing node and put the nodes in their own rings
- Now they start gossiping normally

Closest Node Discovery

Goal

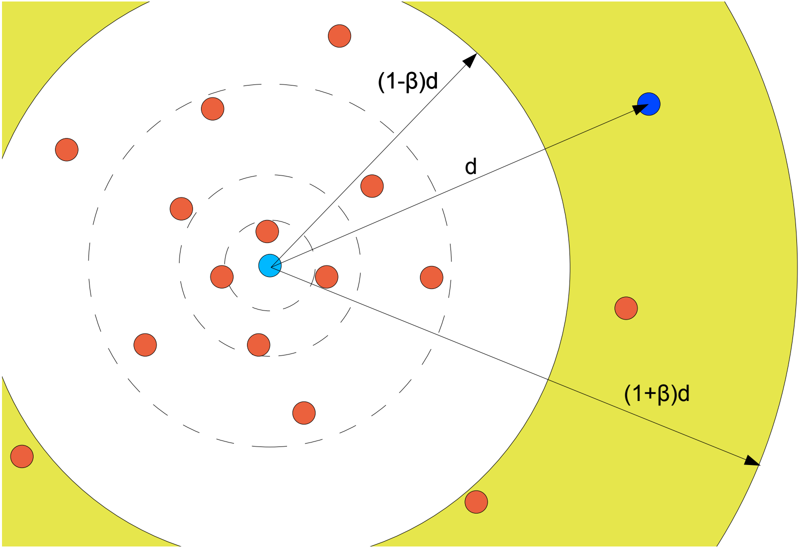
Find closest Meridian node to a given target node (not necessarily a Meridian node)

Algorithm

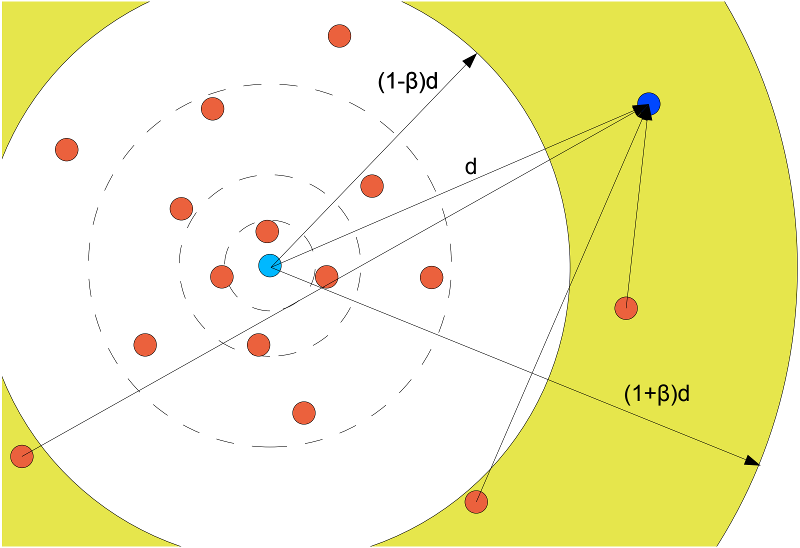
1. Measure distance d to target node T
2. Ask all ring-nodes within range of $(1-\beta)d$ to $(1+\beta)d$ for their distance to T
3. If the distance d_i of the closest node i is smaller than βd, start over from node i
4. Terminate otherwise

$0 \leq \beta < 1$, where a large β reduces errors at the expense of hop counts.

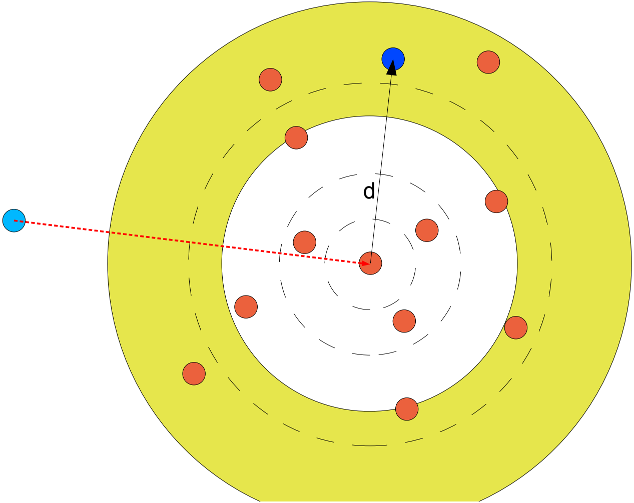
Closest Node Discovery



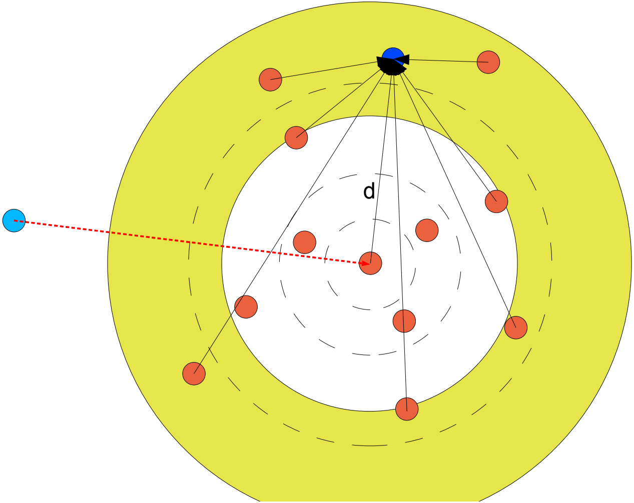
Closest Node Discovery



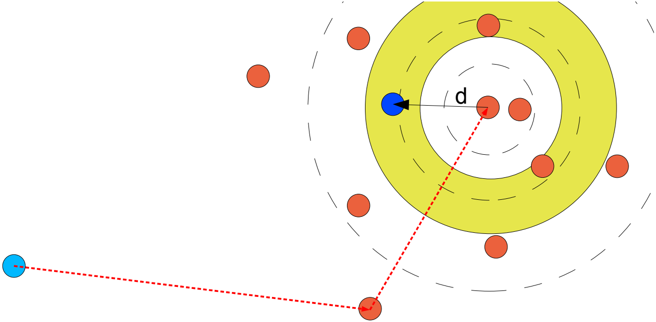
Closest Node Discovery



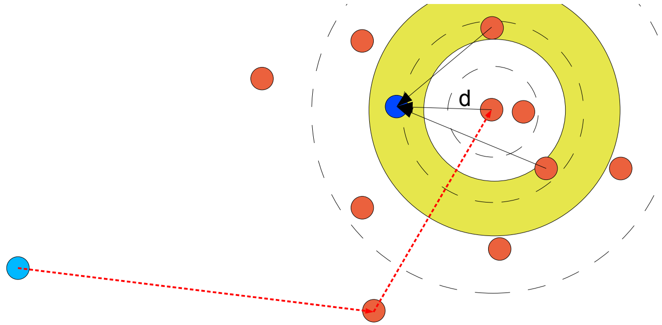
Closest Node Discovery



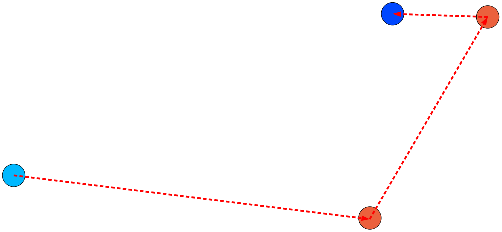
Closest Node Discovery



Closest Node Discovery



Closest Node Discovery



Central Leader Election

Goal

Find a Meridian node with lowest average latency to a given set of nodes (not necessarily Meridian nodes).

Central Leader Election

Goal

Find a Meridian node with lowest average latency to a given set of nodes (not necessarily Meridian nodes).

Can be solved using a slight variation of closest node discovery:

- Replace single target node T with a set of target nodes T
- Replace d with $d_{avg} = \frac{1}{|T|} \sum_{i=1}^{|T|} d_i$

Target Latency Constraint System

Goal

Find a set of nodes satisfying certain latency constraints.

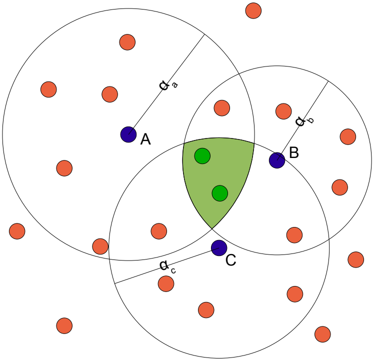
Constraints given as $\langle target_i, range_i \rangle$ for $0 < i \leq u$.

Target Latency Constraint System

Goal

Find a set of nodes satisfying certain latency constraints.

Constraints given as $\langle target_i, range_i \rangle$ for $0 < i \leq u$.
Example: $\langle A, \alpha_a \rangle, \langle B, \alpha_b \rangle, \langle C, \alpha_c \rangle$

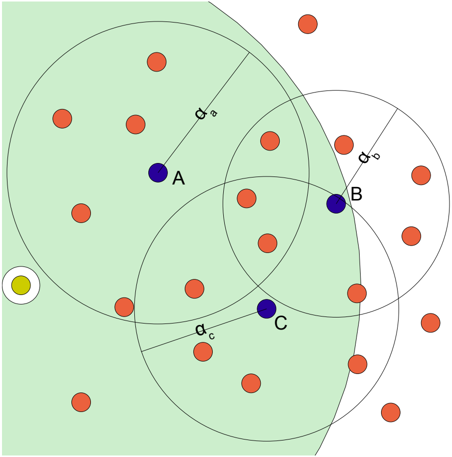


Target Latency Constraint System

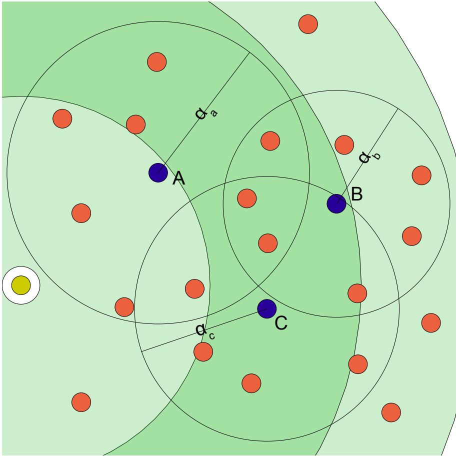
Algorithm

1. Measure latencies d_i to target nodes and calculate distance to solution space as $s = \sum_{i=1}^{u} \max(0, d_i - range_i)^2$
2. Terminate if $s = 0$ (Node fulfills all latency constraints)
3. Otherwise, query all peers j that are within $\max(0, (1-\beta)\cdot(d_i - range_i))$ to $(1+\beta)\cdot(d_i + range_i)$ for their distances to target nodes
4. Calculate s_j for every peer
5. Terminate if any $s_j = 0$, because that node fulfills all constraints
6. Otherwise, forward the request to node j with $s_j < \beta s$ (if available)

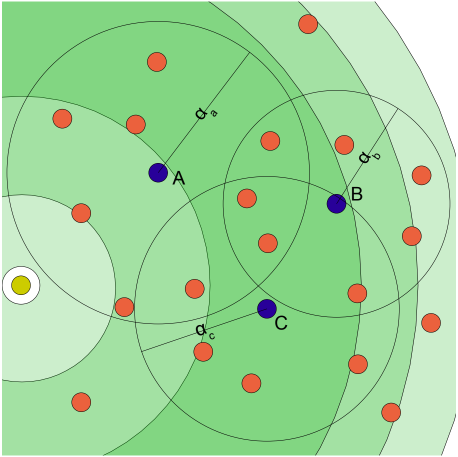
Target Latency Constraint System



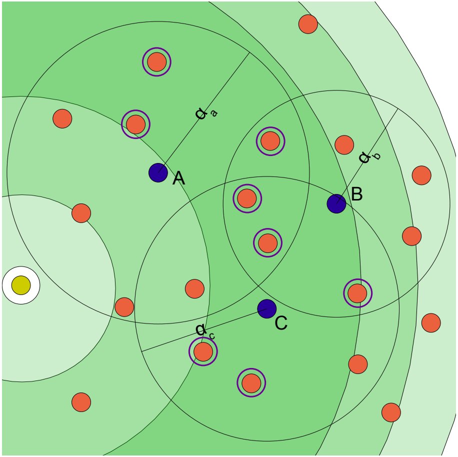
Target Latency Constraint System



Target Latency Constraint System



Target Latency Constraint System



Analysis Summary

m fixed forever, k and l grow with number of nodes, usually $k = l = \log(N)$

Storage

Storage requirement per node: $O(m(k+l)) = O(\log N)$

Communication

Ring membership management: $O((k+l)^2) = O(\log^2 N)$
Gossip protocol: $O(m^2) = O(1)$

Analysis Summary

The paper proves some further theoretical statements:

- Small ring cardinalities suffice to ensure good quality (under certain reasonable assumptions)
- Nearest-Neighbor returns exact or near-exact neighbors in logarithmic number of hops
- The system is load-balanced if the ring sets of different nodes ar stochastically independent.

Some Terms

$B_{ui} = B_u(2^i) =$ closed ball of Meridian nodes of radius 2^i around node u
$S_{ui} \subset B_{ui} \setminus B_{u(i-1)} =$ i-th ring of Meridian node u

Some Terms

$B_{ui} = B_u(2^i) =$ closed ball of Meridian nodes of radius 2^i around node u
$S_{ui} \subset B_{ui} \setminus B_{u(i-1)} =$ i-th ring of Meridian node u

Definition

A pair uv of Meridian nodes is ϵ-nice if node u has a neighbor w within distance ϵd_{uv} from v, and $w \in S_{ui}$ where $2^{i-1} < d_{uv}(1+\epsilon) \leq 2^i$. The rings are ϵ-nice if all pairs of Meridian nodes are ϵ-nice.

Some Terms

$B_{ui} = B_u(2^i) =$ closed ball of Meridian nodes of radius 2^i around node u
$S_{ui} \subset B_{ui} \setminus B_{u(i-1)} = i$-th ring of Meridian node u

Definition

A ring S_{ui} is well-formed, if it is distributed as a random k-node subset of B_{ui}.

Some Terms

$B_{ui} = B_u(2^i) =$ closed ball of Meridian nodes of radius 2^i around node u
$S_{ui} \subset B_{ui} \setminus B_{u(i-1)} =$ i-th ring of Meridian node u

Definition

Algorithm $\mathcal{A}(\beta_0)$ is an algorithm that forwards the query for target q from node u to w if $d_{wt} < \frac{1}{\beta_0} d_{ut}$.

Some Terms

$B_{ui} = B_u(2^i) =$ closed ball of Meridian nodes of radius 2^i around node u
$S_{ui} \subset B_{ui} \setminus B_{u(i-1)} =$ i-th ring of Meridian node u

Definition

Let u be the nearest neighbor of node q. Node v is a γ-approximate nearest neighbor of q if $d_{vq} \leq \gamma d_{uq}$.

Some Terms

$B_{ui} = B_u(2^i) =$ closed ball of Meridian nodes of radius 2^i around node u
$S_{ui} \subset B_{ui} \setminus B_{u(i-1)} =$ i-th ring of Meridian node u

Definition

$\mathcal{A}$ is γ-approximate if for any query it finds a γ-approximate nearest neighbor, and does so in at most $2\log(\Delta)$ steps.

A Theorem
without proof

Theorem

If the rings are ϵ-nice, $\epsilon \leq \frac{1}{8}$ then
(a) $\mathcal{A}(2)$ is 3-approximate,
(b) $\mathcal{A}(\beta_0)$ is $(1+\epsilon)$-approximate, $\beta_0 = 1 + O(\epsilon^2)$.
(c) if we use a larger threshold $\beta_0 = 1 + \gamma, \gamma \in (\epsilon, \frac{1}{2})$ then
$\mathcal{A}(\beta_0)$is$(1 + \epsilon + 2\gamma)$-approximate.

The theoretical results have been verified in two different ways:

- A simulation based on real-world latencies
- A physical deployment on PlanetLab

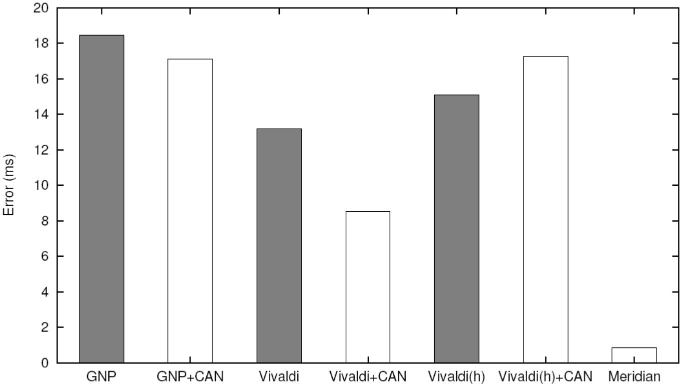
Simulation

They collected pairwise latencies between 2500 internet nodes.

Setup

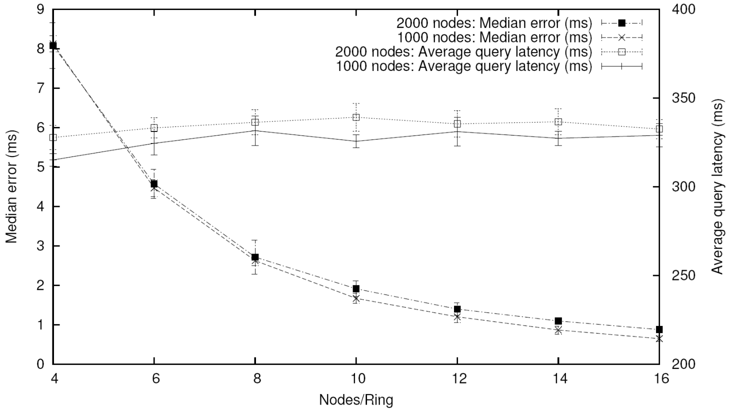
- 2000 Meridian nodes, 500 target nodes
- $k = 16$ nodes per ring, $m = 9$ rings per node
- Acceptance threshold $\beta = \frac{1}{2}$
- Innermost ring radius $\alpha = 1ms$, Ring grow factor $s = 2$

Simulation



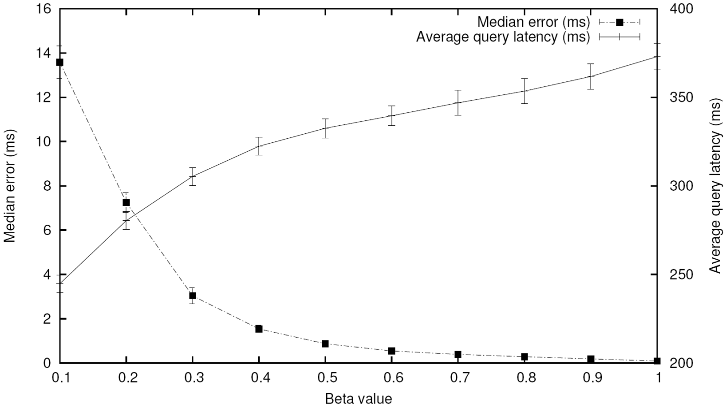
Dark bars show the inherent embedding error, light ones the median error for nearest-neighbor discovery

Simulation



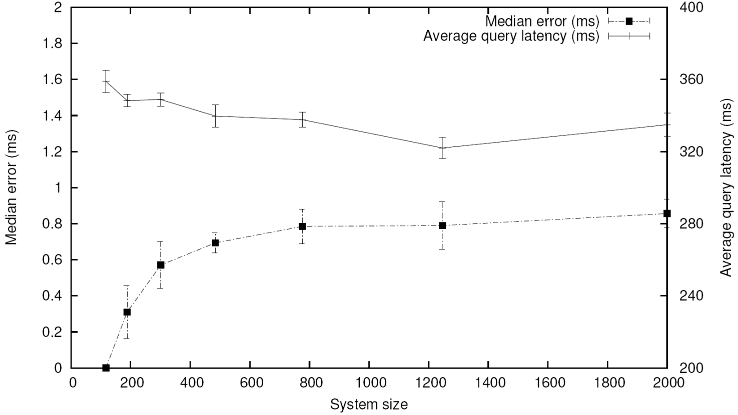
Error is reduced with more nodes per ring, while the latency remains about constant

Simulation



Increasing β improves accuracy, while the average number of hops increases

Simulation



Error and latency remain unaffected as the network size increases

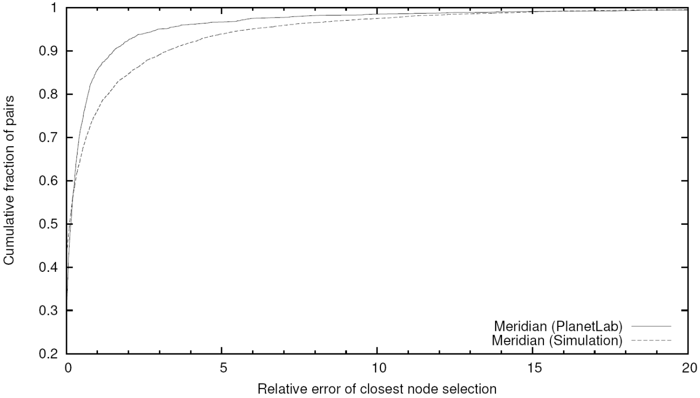
Physical Deployment

Setup

- Deployed on 166 PlanetLab machines
- 1600 different targets
- $k = 8, \beta = \frac{1}{2}, \alpha = 1ms, s = 2$

They determined the closest node by querying every machine and compared the result with the one Meridian provided.

Physical Deployment



The relative errors of simulation and deployment compared.

Conclusions

- Truly lightweight
- Scales well
- Accurate both in theory and practice
- Simple (easy to implement)

Questions?