Jonathan Ledlie, Paul Gardner, and Margo Seltzer of
Harvard University and Aelitis

NETWORK COORDINATES IN THE WILD

Presented by Nikhil R. Sethi

Outline

- What are Network Coordinates
- Applications
- Background
- Characteristics
- Problems
- Solutions
- Conclusion

Applications

- Network coordinates provide a mechanism for selecting and placing servers efficiently in a large distributed system.
- Performance of Internet Applications
  - Distributed hash tables, web caches, and overlay networks.
  - All require accurate latency estimation between participants.

Background

- Current Methods
  - Proxy measurement
  - Landmark binning
  - Decentralized network embeddings
- Each node has a “network coordinate”
  - The metric distance between two coordinates in the abstract space predicts real world latencies.
- Current Problems
  - Inaccuracy and fragility in presence of triangle inequality violations
- Use Azureus BitTorrent Network
- Million-plus node network

Naysayers

- Debate is turning into a “religious war”
- Naysayers state that coordinate maintenance is too expensive
- Prediction accuracy is worse that direct measurement
- Unproven idea and unlikely to work in practice

Supporters

- Accuracies are reasonable 8-15%
- Maintenance can be built on top of existing application communication. (piggyback)

Azureus

- BitTorrent client
- Initial seeder, new seeders, tracker, and peers.
- Use of network coordinates to:
  - Optimize DHT traversal
  - Help clients choose between exchanging with one client and another.
- Too many clients with pieces to exchange.

More applications

- More complex applications
  - A web cache can be placed at the centroid of a set of web clients that want access to same data.
  - Server hosting a distributed game can be hosted at a machine close to the centroid of the players' coordinates leading to fair access times.

Vivaldi

- The Vivaldi algorithm
  - calculates coordinates as a solution to a spring relaxation problem.
  - Measured latencies are modeled as extensions of springs between mass less bodies
  - Minimum error is found at low-energy state of the spring-system.

VIVALDI$(\vec{x_j}, w_j, l_{ij})$

1 $w_s = \frac{w_i}{w_i + w_j}$

2 $\epsilon = \frac{|\|\vec{x_i} - \vec{x_j}\| - l_{ij}|}{l_{ij}}$

3 $\alpha = c_e \times w_s$

4 $w_i = (\alpha \times \epsilon) + ((1 - \alpha) \times w_i)$

5 $\delta = c_c \times w_s$

6 $\vec{x_i} = \vec{x_i} + \delta \times (\|\vec{x_i} - \vec{x_j}\| - l_{ij}) \times u(\vec{x_i} - \vec{x_j})$

1. Sample confidence
2. Relative Error
4. Exponentially weighted moving average
5. Coordinate
6. Coordinate updated.

Height

- Height
  - An alternative to pure Euclidean distance metric
  - Distance between nodes is measured as their Euclidean distance plus a height above the hypercube that models latency penalty of network links i.e. DSL lines

Neighbor Set

- Each node successively refines its coordinate through periodic updates with other nodes in its neighbor set.
- Information used to maintain this is piggybacked onto existing messages, resulting in no additional overhead (28 bytes per message)
- Algorithm needs to function passively.

Latencies in the Wild

- Previous studies use matrices of inter-node latencies.
  - Infeasible for large networks i.e. Azureus
- Three instruments for Azureus
  - Matrix of a subset of network to act as “ground truth”
  - Clients running on PlanetLab that log every update
  - Statistics injected into Azureus code which were collected by standard clients.

Collection

- PlanetLab
  - 283 nodes running for 24 days collecting 9.5*10^7 latency measurements to 156,658 Azureus nodes.
- Process
  - Summarized each edge with median round trip time
  - Discarded edges with fewer than 4 samples
  - Resulted in 249x2902 matrix with 91% entries containing latency values.

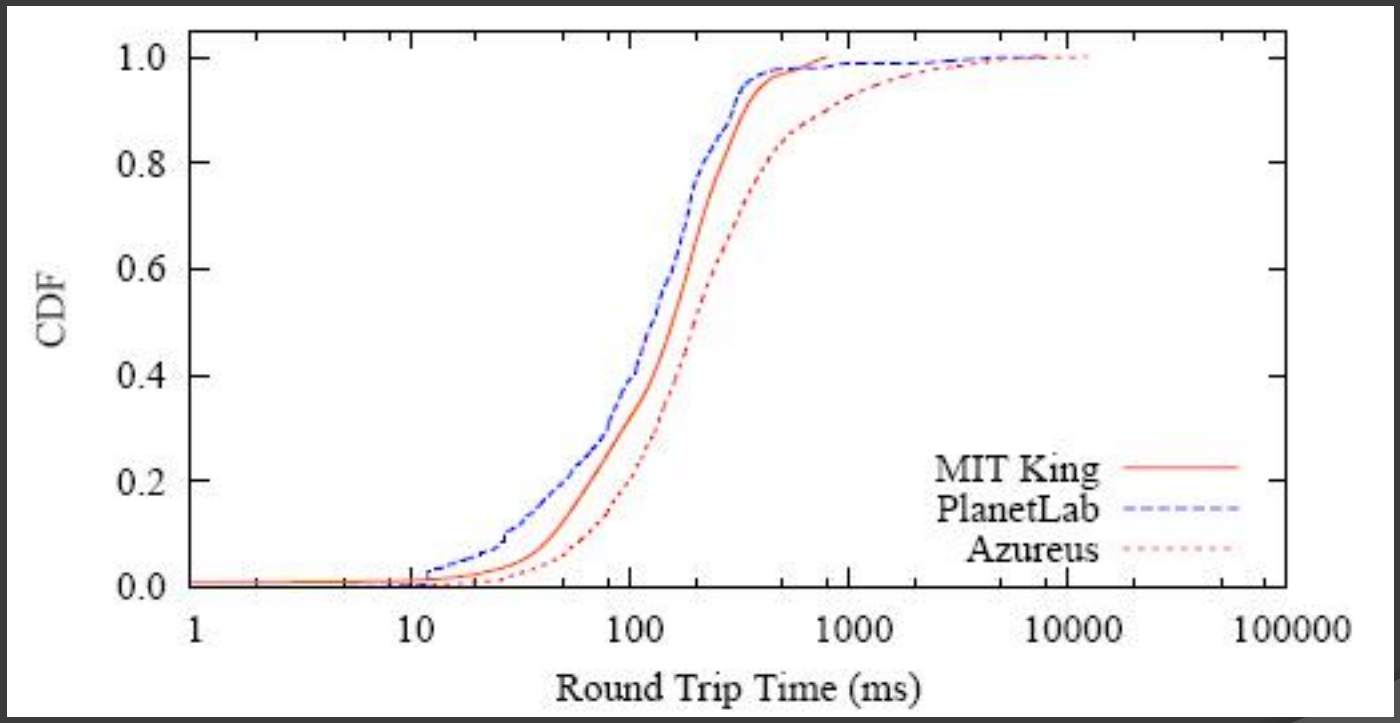
Characteristics

- Round trip times
- Violation of triangle inequality
- Intrinsic dimensionality

Round Trip Times

- Spread
  - Azureus round trip times spread across four orders-of magnitude
  - MITKing data set spreads across three
  - In practice the error between nodes whose distance is near the middle of the latency distribution tends to be the lowest
  - This wide spread is a warning sign that Azureus will have higher error

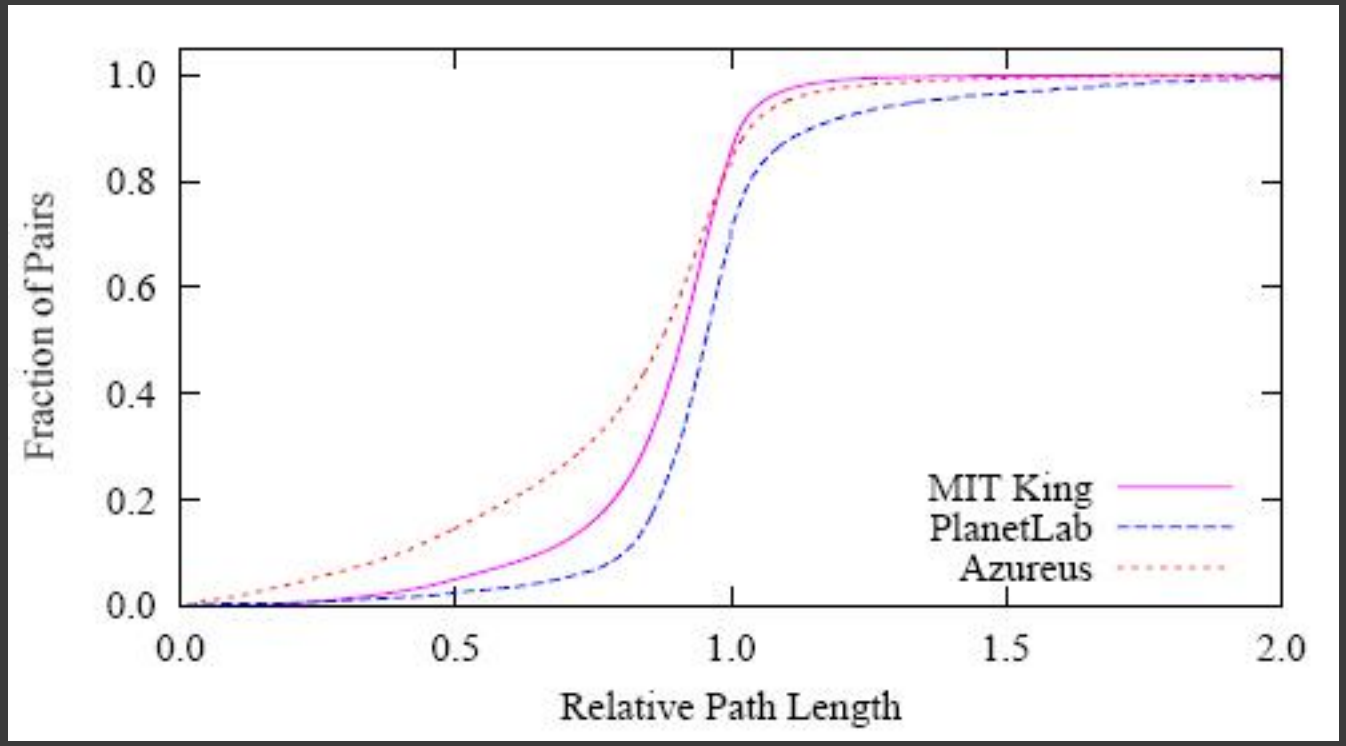
Round Trip Times



Violations of the Triangle Inequality

- No violations + significant number of dimensions = no error
- Tang/Crovella method normalizes severity of each violation so system can be viewed as a whole
- For each node pair we find the shortest path between two that pass through a third node.

Tang/Crovella Triangle Inequality Test



85% failed

68%

83%

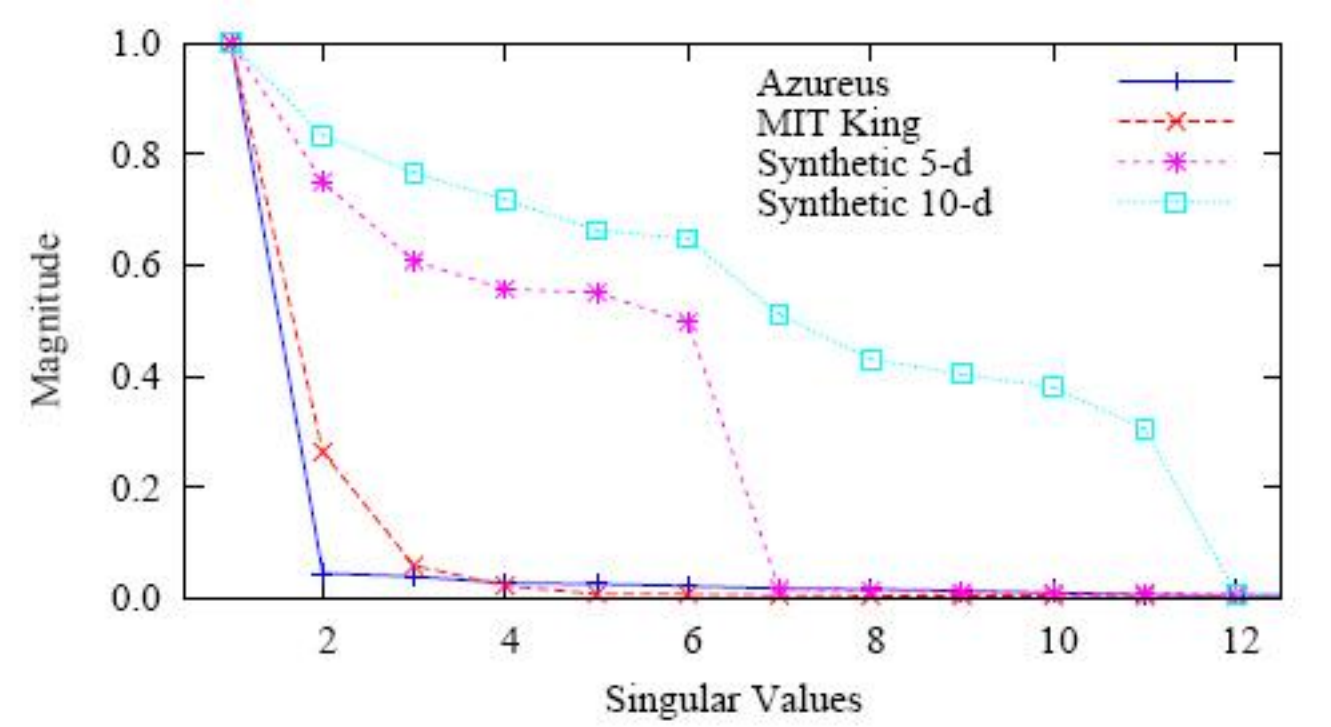
In all three data sets, over half of node pairs fail the test, because there is a third node between the pair that produces a shorter path. A large fraction of these violating pairs have paths that are significantly faster.

This too foreshadows high embedding error in Azureus data set.

Dimensionality

- Network coordinates less useful if a large number of dimensions are needed to capture inter-node latencies.
- Principal Component Analysis (PCA)
  - Technique to hint at number of dimensions required to encompass this information.
  - Requires full matrix, so missing values needed to be filled in Azureus (9%)

Dimensionality



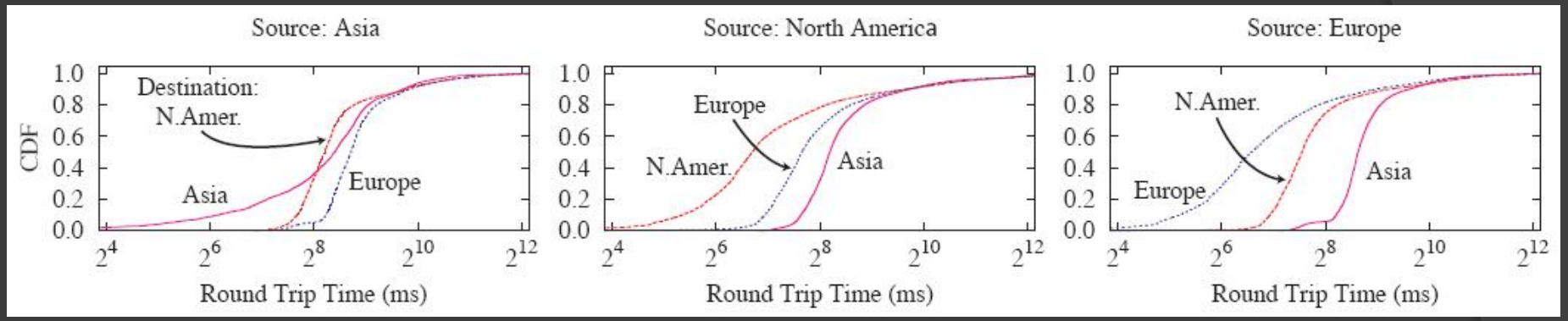
Scree plot shows how much variance each new singular value is capturing, which shows inherent dimensionality of underlying data set. At point where magnitude of singular values becomes zero or close to zero, three dimensions are necessary.

Create synthetic 5/10 dimension systems with 250 random points each, notice the drops after 5 and 10 for those data sets. 4-5 dimensions is appropriate for Internet-scale network coordinates.

Intercontinental Latency Distributions

- More concrete way of examining flatness.
- Euclidean space for the globe?
- Spherical coordinates have high error.
- The world is flat.
  - Traffic flows from Asia and Europe through North America

Intercontinental…



Mapped IP addresses to countries and mapped countries to continents. No messages from Asia to Europe were faster than those from Asia to North America. Same in opposite direction.

All paths between Asia and Europe appear to travel in a line across two oceans.

Using Euclidean metric is sufficient.

Network coordinates in Azureus are fundamentally embeddable, however its round trip time distribution and triangle inequality violations suggest that large error will be exhibited, despite the few dimensions needed.

Techniques to improve network coordinate system

- Latency and Update Filters
- Neighbor Decay
- Measuring Coordinate Systems
- Live Coordinates

Latency and Update Filters

- Two simple filters
  - Latency filter
    - Takes stream of latency measurements from a remote node and turns these into an expected latency value.
    - Anomalous measurements affecting the values. Measure of round-trip would be 1000ms when actual measurements was 200ms.
    - Also expected value could not be fixed at a single value. “plateau shifts”
  - Update Filter
    - Focuses on making coordinates more stable not more accurate
    - When a coordinate has changed enough to cause an application-level reaction.
    - Filter differentiated between constantly evolving system-level coordinates and higher application-level coordinates.

Neighbor Decay - Problem

- Network coordinates should function passively, that is without generating any extra traffic.
  - In case of Azureus they had no control over the selection of nodes talked to, due to the piggybacking nature of information for a coordinate update.
  - Thus, nodes did not have a fixed set of neighbors with expectations for regular exchanges. Some nodes would receive 1-3 updates from a remote node and then never hear from that node again.

Neighbor Decay - Solution

- Instead of refining our coordinate with respect to the remote node from where new information is retrieved, it is refined with respect to all the nodes which have recently received an update.
- Normalizing
  - scale each neighbor by its age i.e. older information receives less weight. This allows nodes that are infrequent to have a lasting, smooth effect on our coordinate.

Neighbor Decay – Benefits

- Node coordinates do not jump to locations that have high error with respect to other members of neighbor set
- Acts to increase the effective size of the neighbor set, which can lead to higher global accuracy.

Measuring Coordinate Systems

- Relative Error
  - Difference between the expected and actual latencies between two nodes.
  - Global, continuous, and neighbor
    - Neighbor error is used as a proxy for global error when live nodes are performing computations, with large number of neighbors = approx global error.
- Stability
  - Important when a coordinate change triggers application activity.
- Relative Rank Loss (RRL)
  - Can capture application accuracy better than relative error.
  - Determines how well a network coordinate scheme preserves the relative ordering of all pairs of neighbors.
- Relative Application –Level Penalty (RALP)
  - The cumulative penalty for using network coordinates.
  - Using network coordinates for client selection instead of searching through all nodes.

Live Coordinates

- PlanetLab
  - Ran instrumented Azureus clients over three time periods.
  - Crawled approx 10,000 Azureus clients that internally tracked performance of their coordinates using stats inserted in Azuerus code.
    - Included address of remote nodes, remote and local coordinates, perceived latencies to remote nodes, and timestamps. From this RE, stability, RRL, RALP were determined.

First Snapshot – 2D+H

- Two Dimensions + Height
- None of the filtering techniques implemented.

Second Snapshot – 5D

- Then added 3 dimensions
- Dropped height
- Added filtering techniques
- Removing height was fatal.

Third Snapshot – 4D+H

- 4 dimensions + height
- All filtering techniques
- 220 PlanetLab nodes
- Lasted for 3 days
- Logged updates ~40,000 Azureus nodes.

2D+H vs 5D

In all cases 4D+H was more accurate and stable.

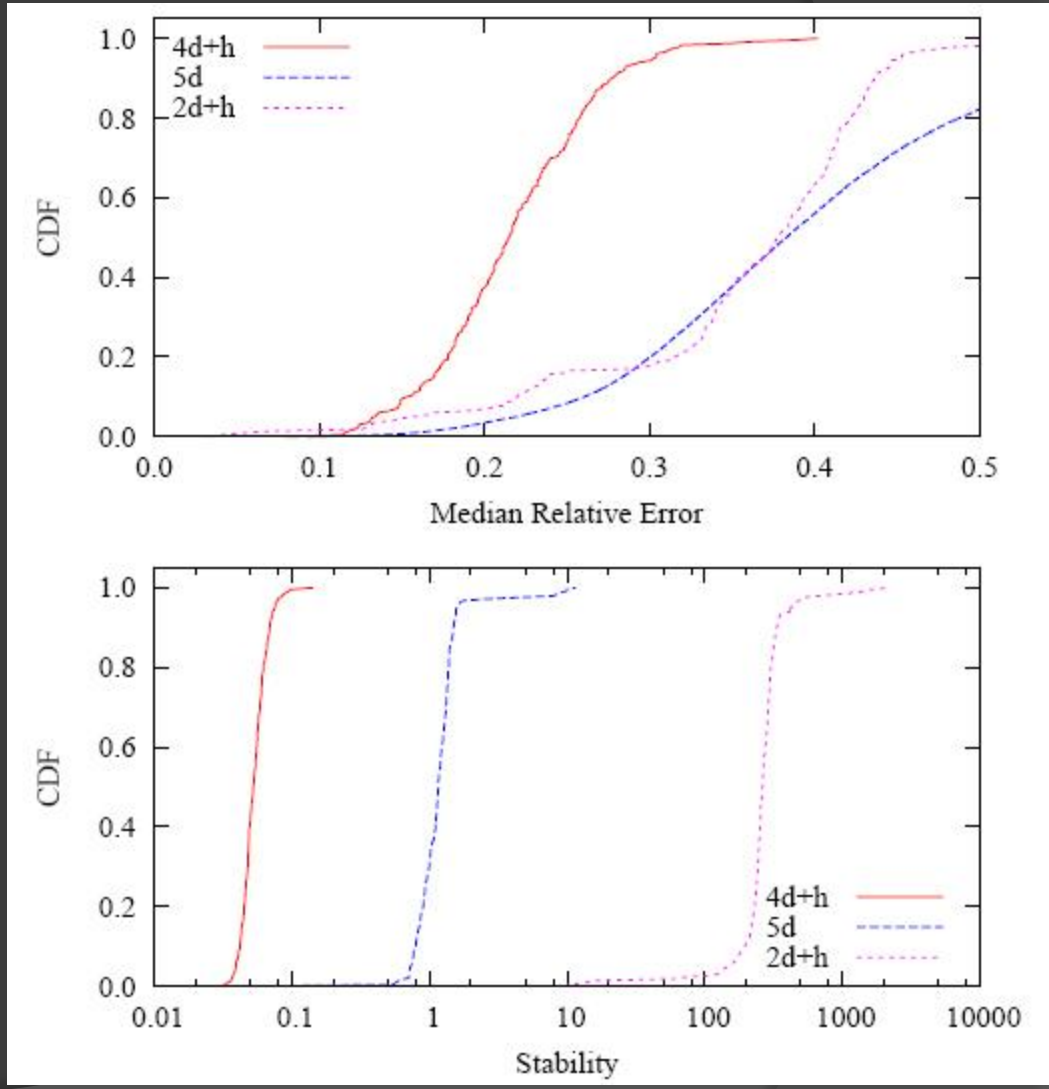
43% improvement in relative error.

4 orders improvement in stability.

First change was 2D+H to 5D

Removal of height damaged accuracy more than the filters aided it.
Given that 2D is sufficient not surprising that addition of 3D did not improve accuracy.

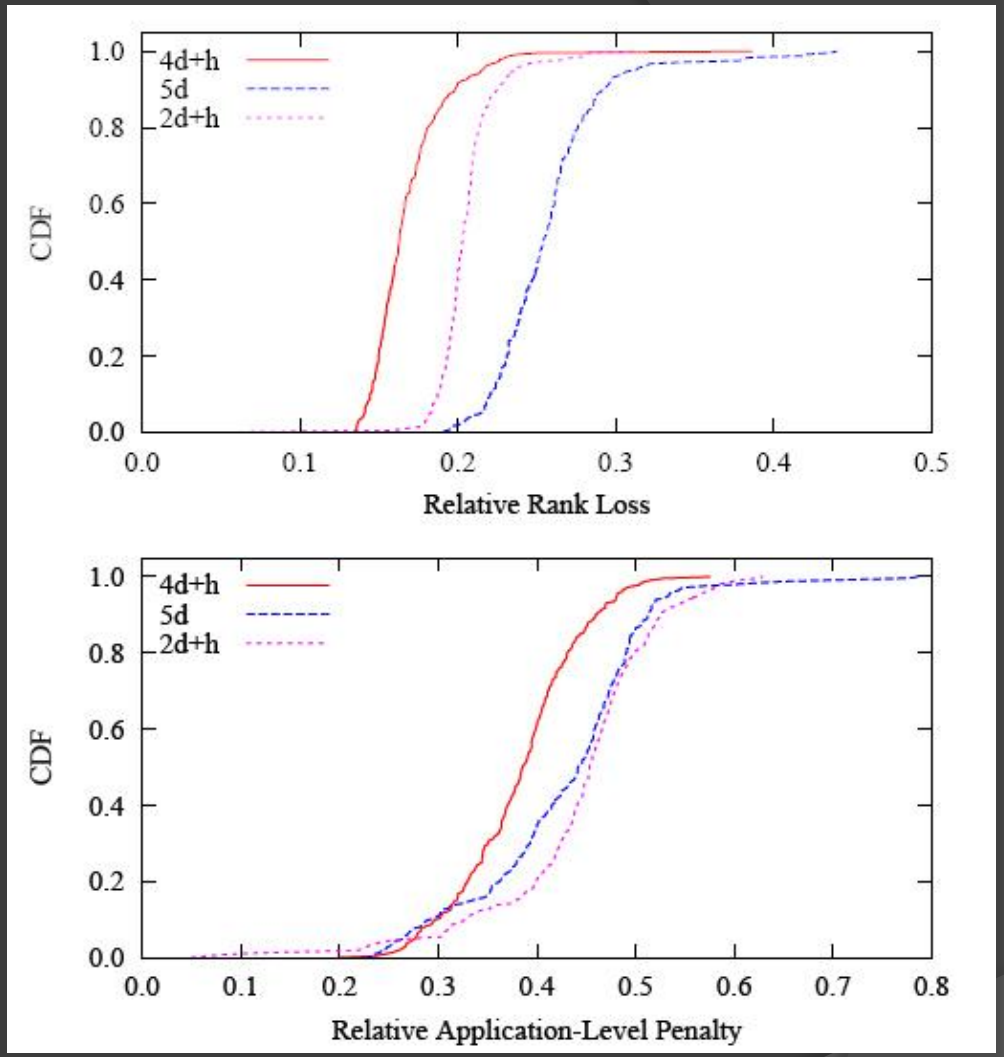
Although 5D is more stable, due to latency filters preventing anomalous measurements reaching updating algorithm.



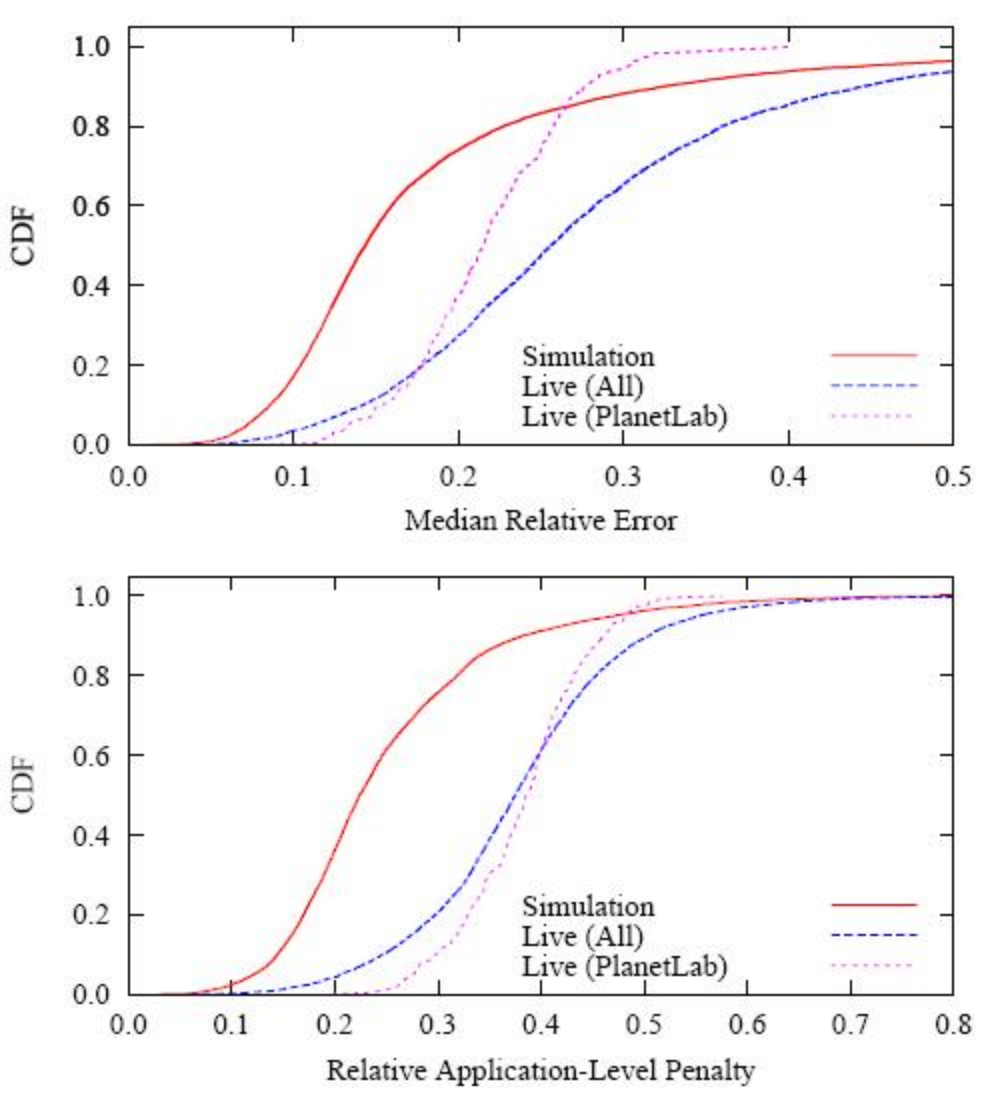
5D vs 4D+H

Introduction of neighbor decay and re-introduction of height created much more accurate coordinate space.

Neighbor decay allowed nodes to triangulate their coordinates.



Live vs. Simulated



Comparison of statistics from live nodes to simulated nodes show that accuracy can be improved by 45%

Barriers to Accuracy

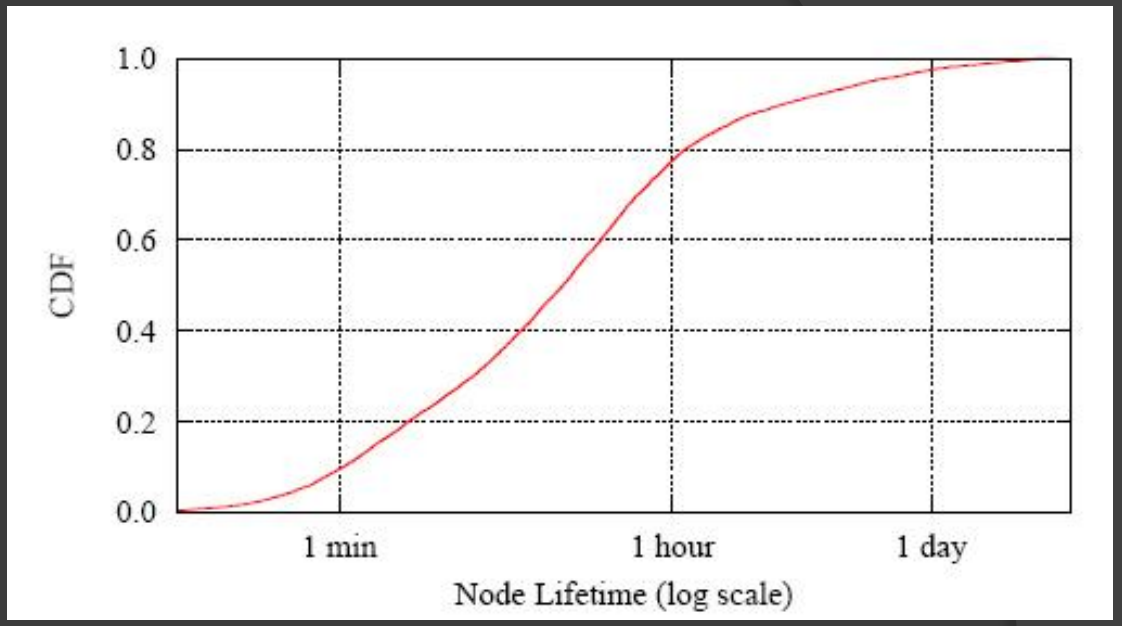
- Churn
- Drift
- Corruption and Versioning
- Intrinsic Error
- Latency Variance

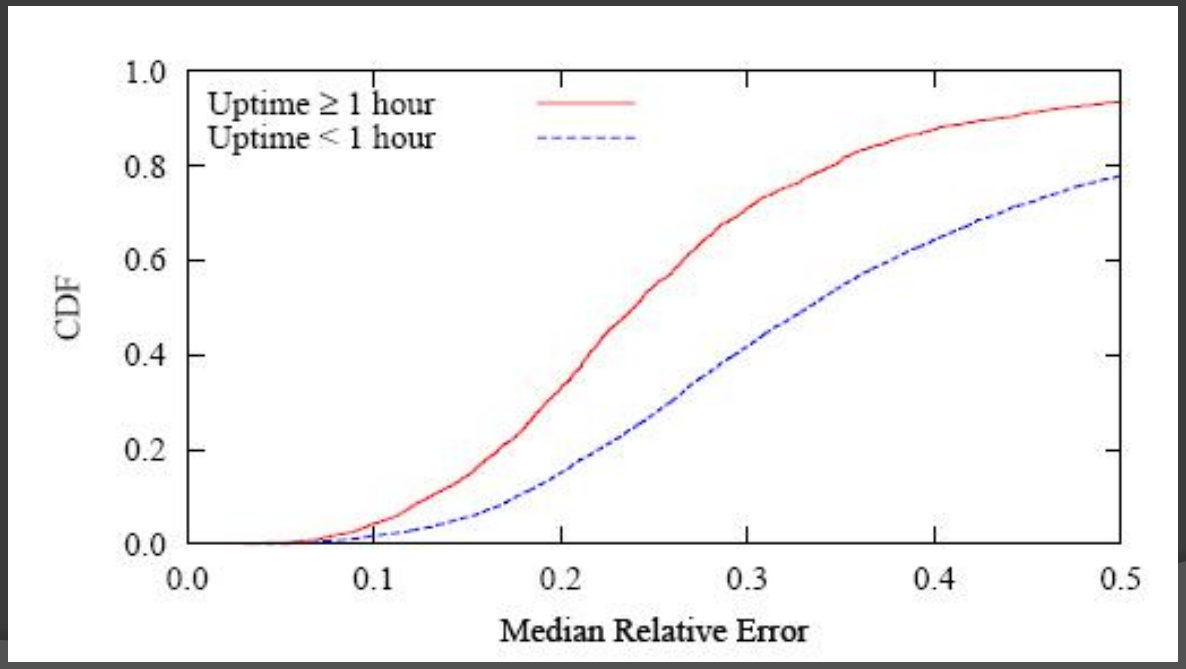
Churn

- Given an existing, stable system, how quickly can a new node find a stable, accurate coordinate.
- Three solutions
  - Nodes could perform a rapid initial triangulation process before shifting to a lower update rate. Passivity is an issue
  - “greedy optimization” instead of stepping once through update, nodes repeat until a local min has been reached
  - Instead of starting from scratch when restarting a client, have it begin where it left off.

Churn

78% of nodes stayed in system for less than one hour. Difficult to incorporate newcomers with coordinates starting at origin.





Nodes that have been in system for more than one hour have more accurate coordinates. Suggests that churn hurts convergence.

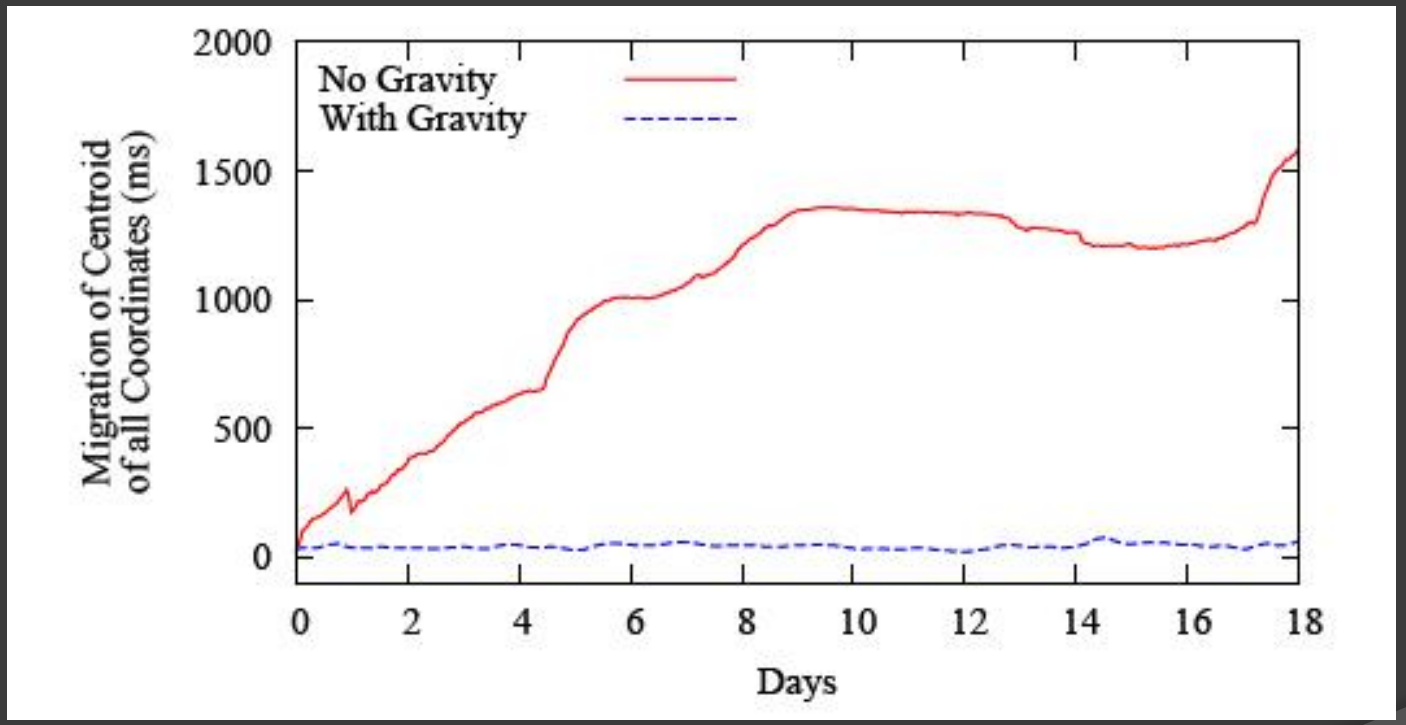
Drift

- Monitoring over several months revealed that coordinates migrated in a fairly constant direction. Not random.
- Absolute coordinates do matter.
- Applications tend to make assumptions on max distance away from “true” origin.
- Cause of origin migration is driven by compressing the globe into a small number of dimensions.

Drift Solution

- Straw-man Solution
  - Continuously re-define origin as centroid of system.
  - Requires accurate statistical sampling of coordinate distribution.
- Gravity
  - Apply a polynomial increasing gravity to coordinates as they become farther away from “true” origin.

Drift


No Gravity
With Gravity
Migration of Centroid of all Coordinates (ms)
2000
1500
1000
500
0
0
2
4
6
8
10
12
14
16
18
Days

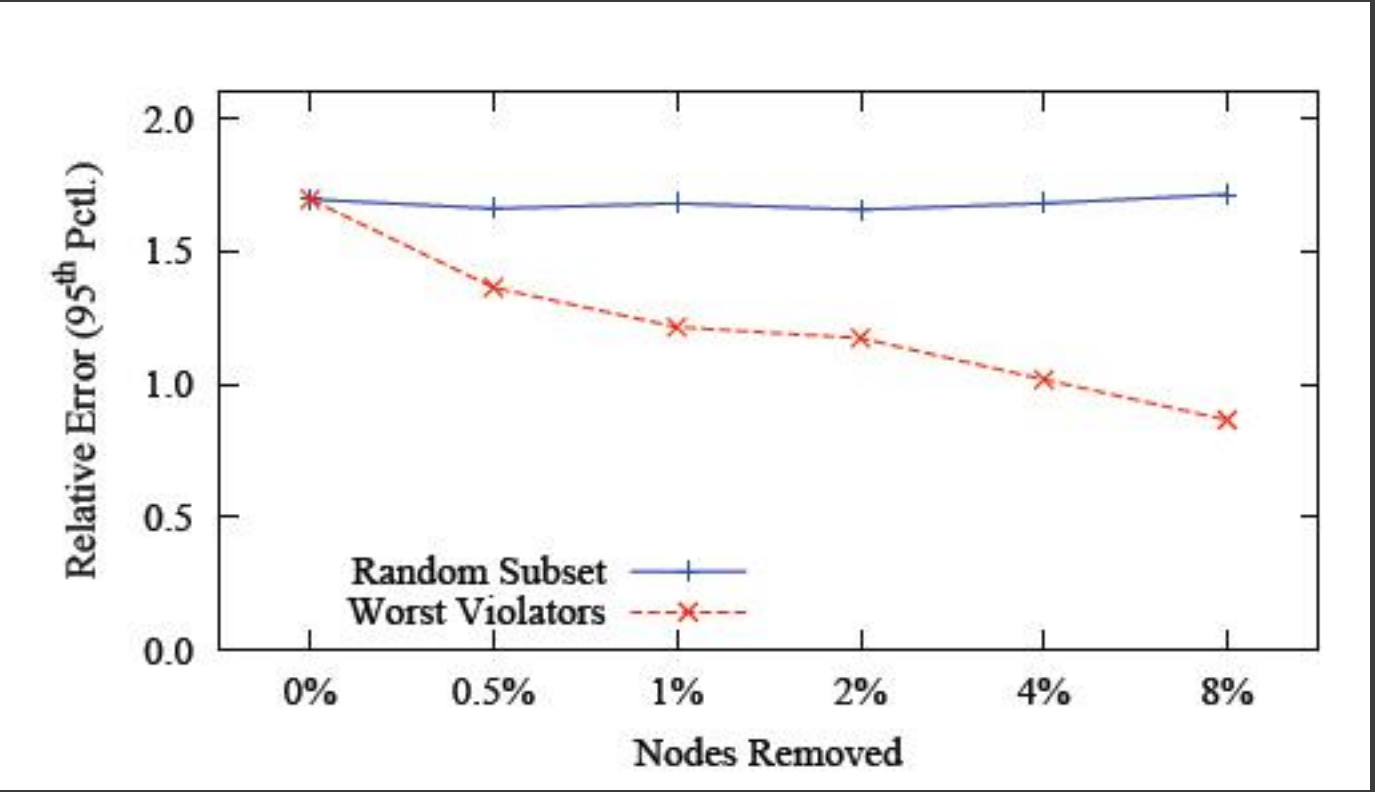
Corruption and Versioning

- Users can choose when to upgrade
  - Causes problem when all users not running same version of code. (13%)
- Malicious behavior
  - Trivial to install client that respond with random values like the MPAA ☹

Intrinsic Error

- Violations of triangle inequality occur frequently on Azureus
- Height manages violations that access-link latency.
- Found that by removing a small number of the worst violators causes a large improvement in global accuracy.
- Not only do they damage their own coordinates, damage reverberates through system.

Intrinsic Error

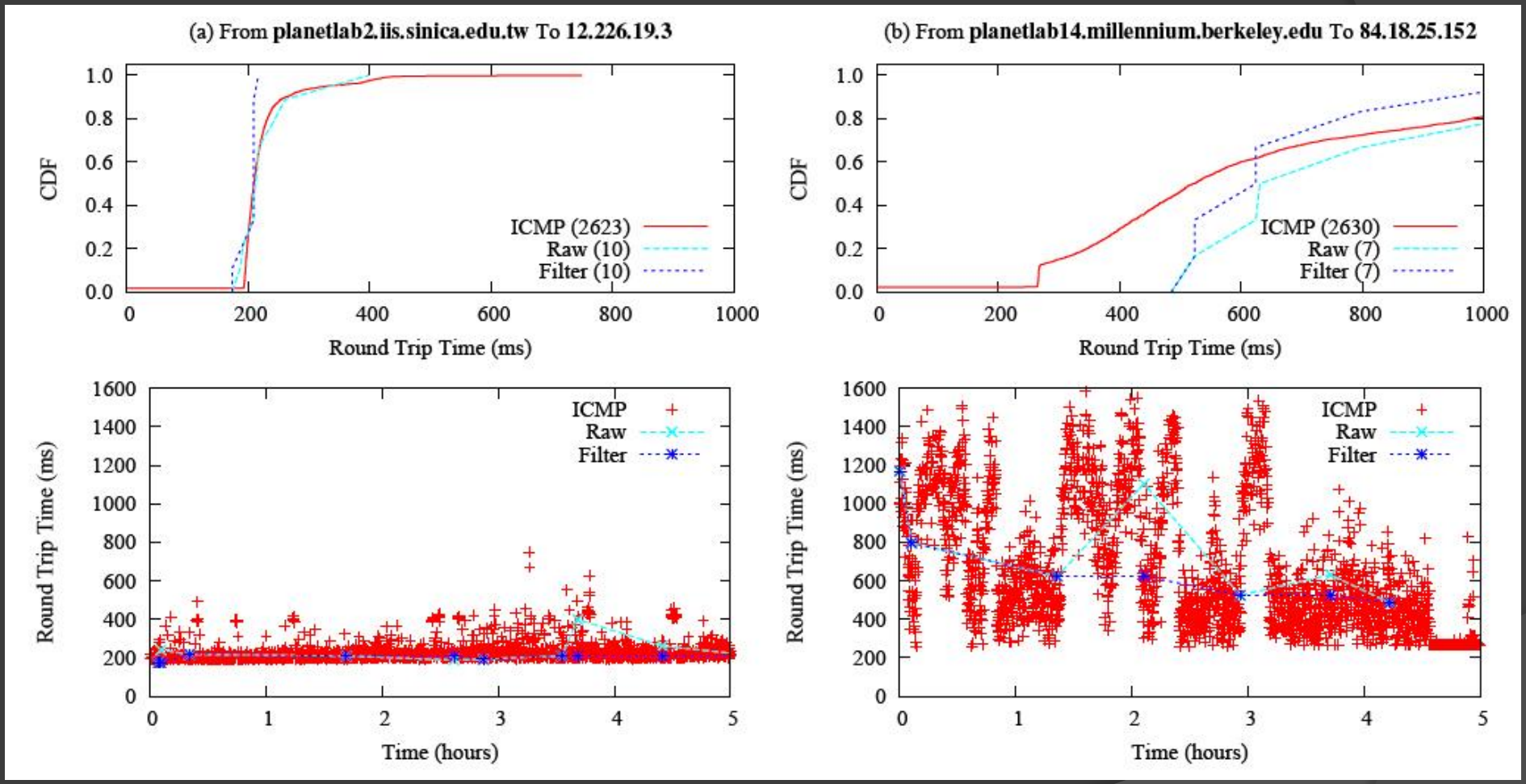


Removing worst .5% of nodes leads to 20% improvement in accuracy.

Latency Variance

- If variances are very large what does it mean to predict the latency from one node to another?
- Latency filters

Latency Variations



Related Work

- Clustering
  - Latency prediction through clustering nodes based on:
    - IP address prefixes
    - Automatic formation through cluster size and amenability.
    - Nodes that are similar distances away from fixed landmarks place themselves in same cluster.

Conclusion

- Network coordinates in the wild to behave differently than tame ones in PlanetLab
- HOWEVER.
  - These wild coordinates can be tamed…
  - Through latency filters, update filters, neighbor decay, coordinate memory, gravity, and violator exclusion