Vivaldi: A Decentralized Network Coordinate System

Frank Dabek, Russ Cox, Frans Kaashoek, Robert Morris

SIGCOMM 2004

Motivation

- Distributed systems benefit when communications latency is minimized
- Explicit measurement is often too much overhead (it's a form of latency/bandwidth consumption itself)
- Latency prediction should be light-weight, accurate, adaptive to changing topology, and distributed (relying on local knowledge alone)

Key observations

- Geographical distance dominates latency (but only roughly approximates actual latency)
- Some nodes have high latency anywhere (slow/ congested links)
- Modeling latency error between nodes as pressure on a spring has nice properties

Vivaldi architecture

- Maps nodes into a simple coordinate system, where distance between coordinates predicts latency
- Gets latency measurements passively, from underlying protocol (relies on steady traffic, reasonable distribution between local and remote nodes)
- Constantly adjusts node coordinates according to latency error feedback
- Completely distributed: coordinates are relative to a handful of known neighbours

Technique

- Minimize global error, calculated as the squared difference between actual and predicted latency for every pair of neighbours
  - This produces spring dynamics (responds quickly, stabilises quickly too)
  - No simple coordinate system can avoid error, even if measurements were perfect and consistent (eg because real latencies violate triangle inequality)

Coordinate System

- There isn't one obviously best coordinate system (spherical? 2D? 3D? 10D?).
- Vivaldi only requires a coordinate abstraction: magnitude, addition and subtraction of coordinates to produce distances
- After some experimentation, the authors believe a planar system with a height metric is suitable (more later)

General algorithm

- Error is modeled as spring tension. To minimize error, adjust coordinates by simulating spring movement:

$$F_{ij} = \left(L_{ij} - \left\|x_i - x_j\right\|\right) \times \mathrm{u}(x_i - x_j).$$

$$F_i = \sum_{j \neq i} F_{ij}.$$

At each interval, move coordinate towards F_i, recompute all forces

Distributed algorithm

- Each node receives neighbour coordinates as well as latency measurement at every communication
- At every sample, adjust own coordinates a small amount according to neighbour’s push
- If two nodes occupy the same position, push away in a random direction (handles starting condition: all nodes at 0)

Adaptive timestep

- How far should a node move in response to one sample? Large movements respond quickly, but produce oscillation
- Some nodes measure latency more accurately than others
- Adjust increment according to self-evaluation of measurement accuracy (accurate measurements reduce movement size)

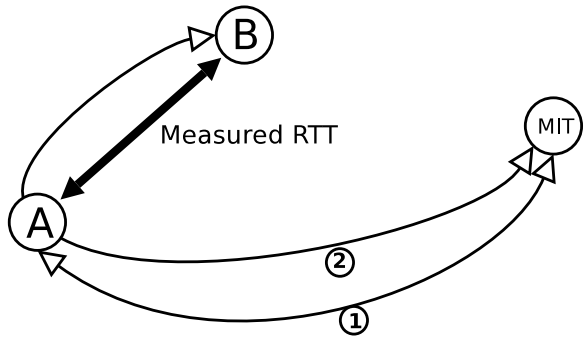
Adaptive timestep 2

- Accuracy is a moving average of the difference between predicted latency and actual
- Accuracy of neighbour affects force vector, so adjustment should scale according to local and remote accuracy

$$\delta = c_c \times \frac{\text{local error}}{\text{local error} + \text{remote error}}$$

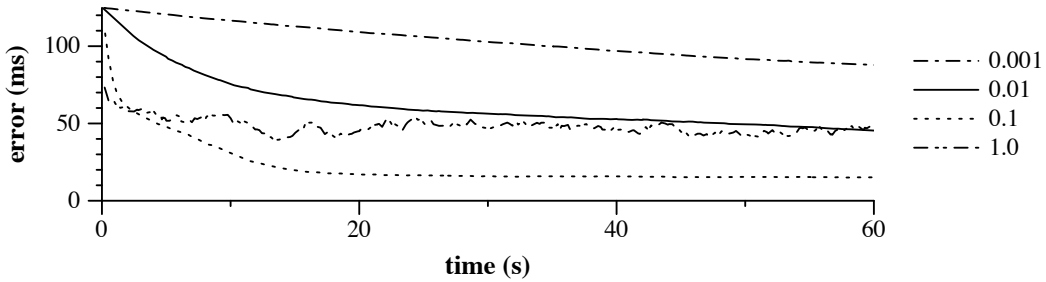
Evaluation

- Two datasets:
  - 192-node PlanetLab simulation, where latencies are measured by direct pair pings
  - 1740 public DNS servers, pair latency measured using King method (measure latency from observer to DNS server A, ask A to recursively query B. The latency between A and B is the difference.)

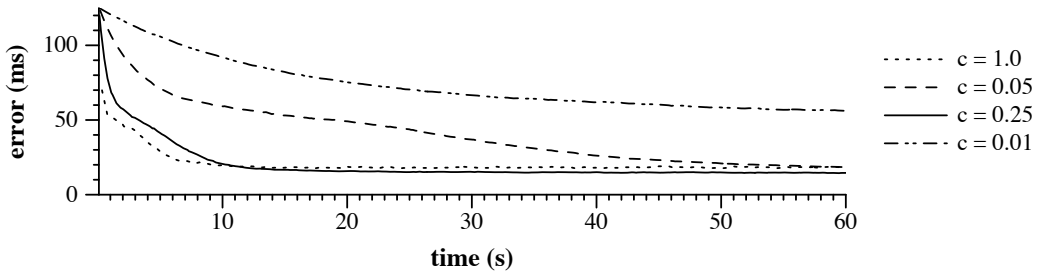


- RTT measured continuously over one week, median used (jitter not reported)

Evaluation: timestep

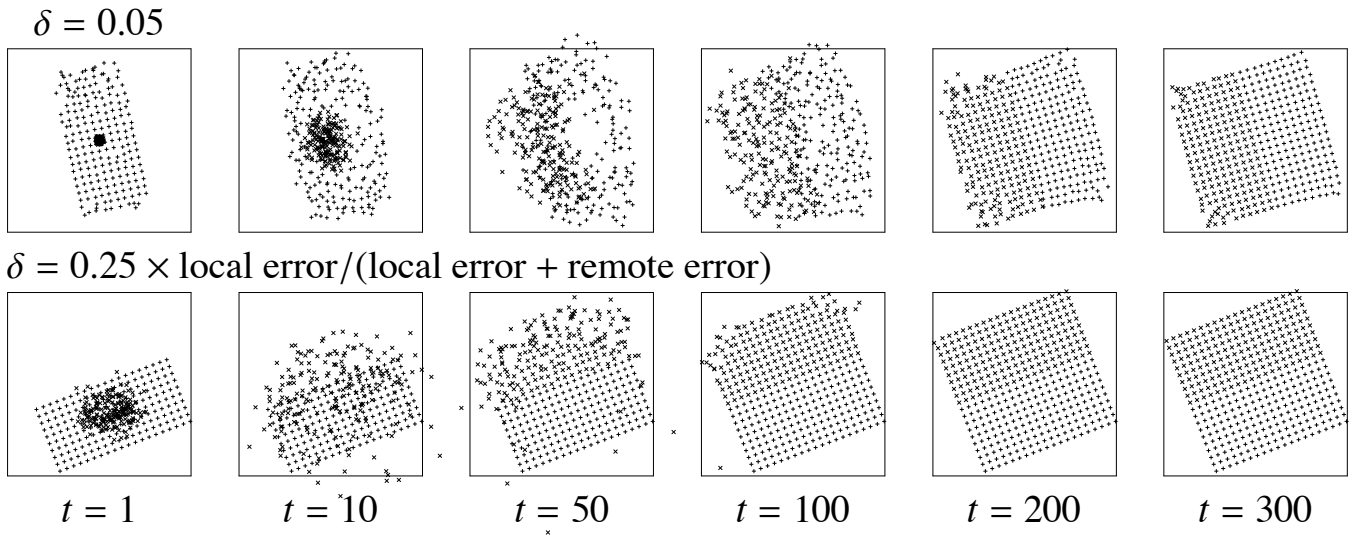


(a)

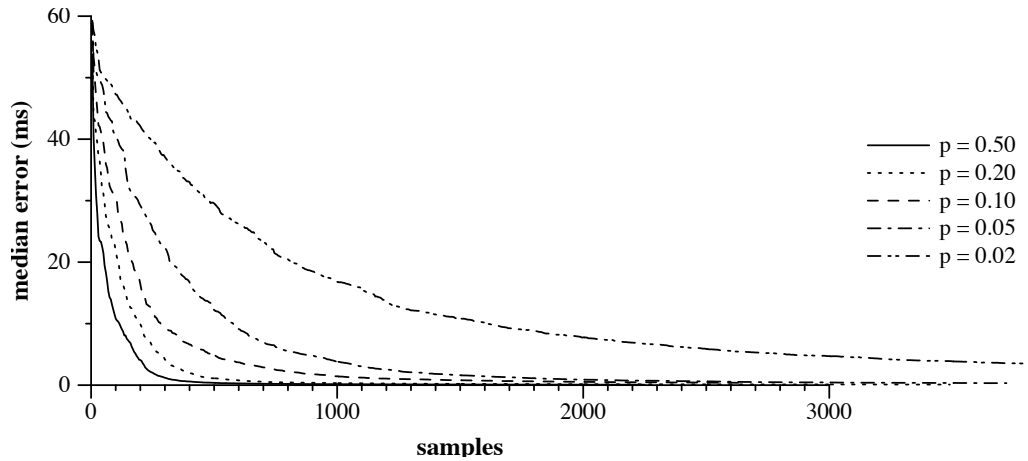


Adaptive timestep stabilizes new network better than fixed

Also has a dramatic effect on stability when new nodes join

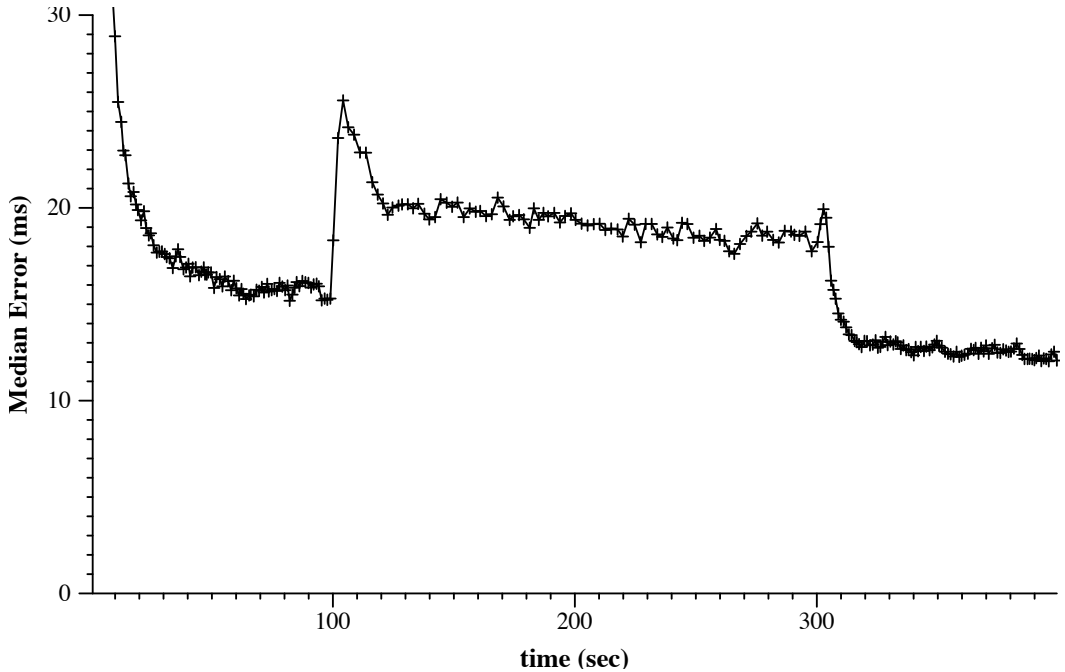


Global knowledge



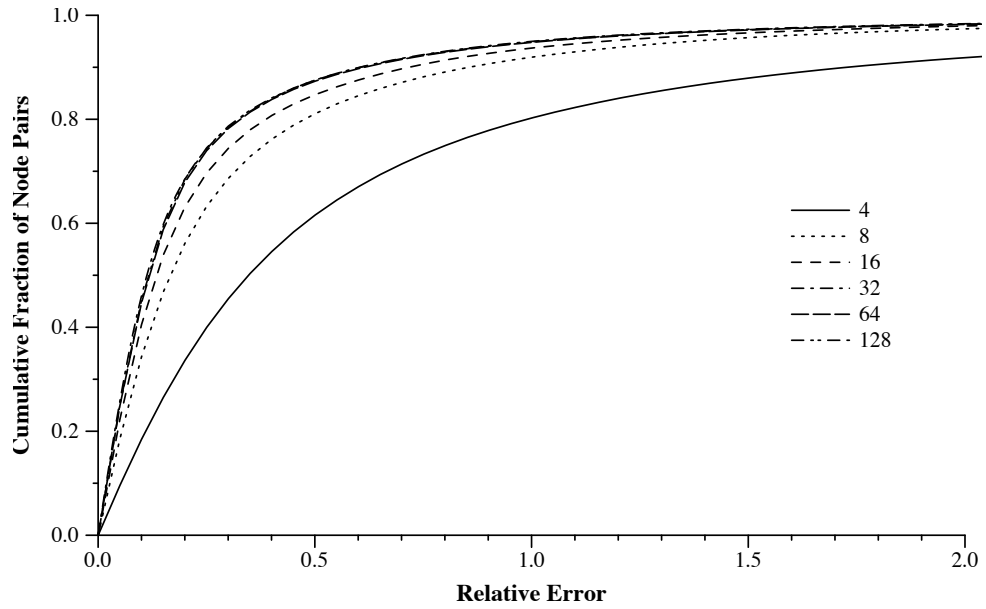
- p: probability that neighbour is distant
- 5-10% distant node measurement is beneficial. Vivaldi's performance is dependent on underlying protocol's choice of nodes.

Adaptability



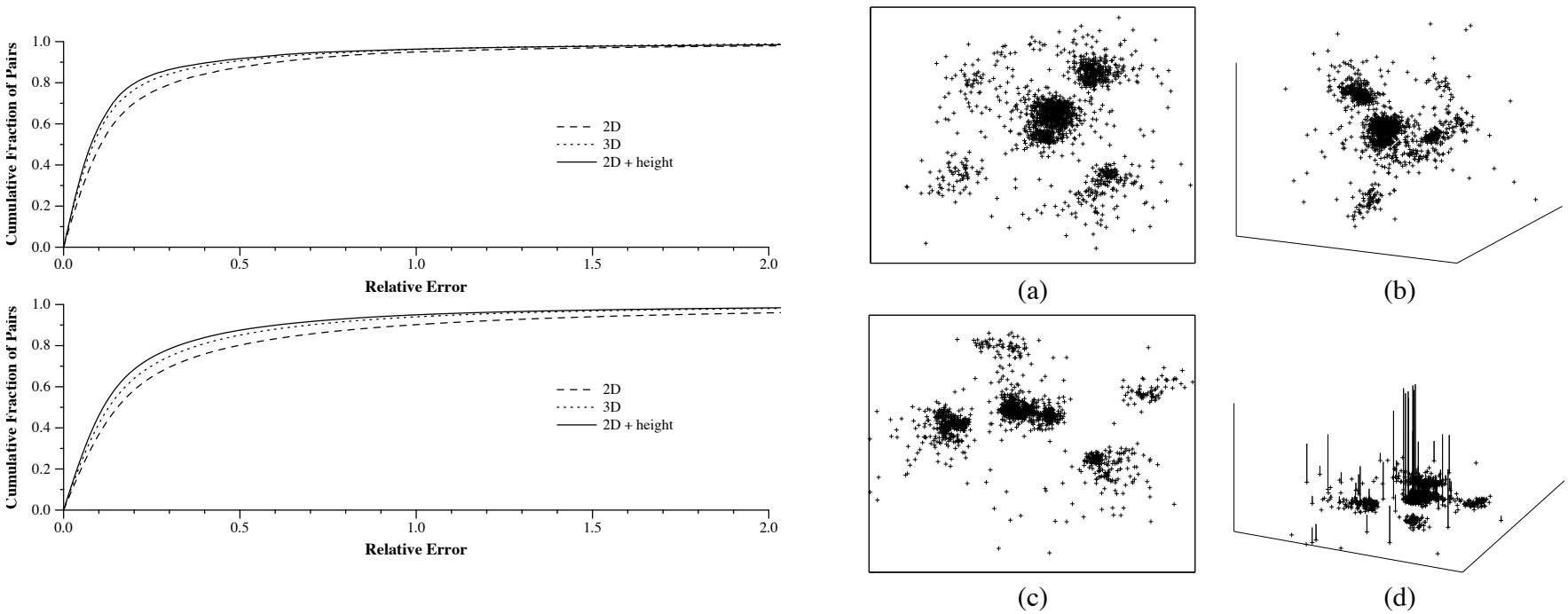
- transit link latency increased by 10 at T=100
- restored at T=300
- looks adaptable

Accuracy



- Number of neighbours has a big impact
- Preferring local nodes to remote improves accuracy
- Vivaldi competitive with GNP without global landmarks

Coordinate systems 2



- Experiments reveal that spherical systems are no more accurate than cartesian at modeling the internet: paths don't wrap around
- 2-D accuracy can be improved with a height vector (not a third dimension, just an addition to the distance from every node)

Conclusions

- It is possible to accurately predict node latency
  - using a simple coordinate system
  - without explicit probes
  - or global knowledge