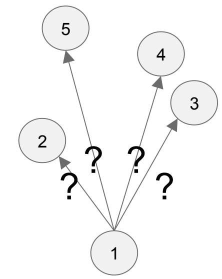
Vivaldi: A Decentralized Coordinate System

Frank Dabek, Russ Cox, Frans Kaashoek and Robert Morris
ACM SIGCOMM Computer Communication Review. Vol. 34. No. 4. ACM, 2004.

Presenter: Andrew and Yibo

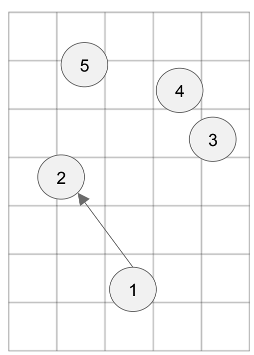
Peer-to-Peer systems

- There are many nodes to communicate with, you want to choose to talk to the node that is closest (lowest RTT)
- One approach is to calculate RTT with each node, and talk to closest node
  - For small clusters or large transfers, this works great!
  - But what about large content distribution systems (i.e. KaZaA, BitTorrent)
  - What about systems with small messages (i.e. DNS)



Peer-to-Peer systems

- You want to put nodes on a coordinate system
  - If your coordinate system approximates RTT well, use it instead of probes!



Coordinate System Requirements

1. Accuracy -- embed Internet with little error
2. Scale to many hosts -- p2p scale
3. Decentralized algorithm -- p2p applications
4. Very little ‘probe’ traffic -- reduce burden on system
5. Adapt to network conditions -- not a static representation

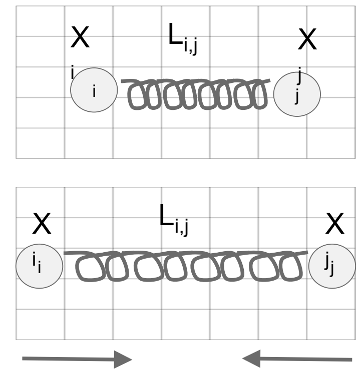
Outline

1. Introduce need for coordinate systems
2. Design of Vivaldi
3. Evaluation of Vivaldi

Vivaldi Network Model

Treat the RTT between two nodes as a spring

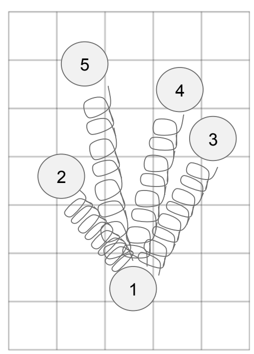
- If distance in coordinates is equal to RTT, no tension in spring
- If distance in coordinates is not equal to RTT, tension in spring



Vivaldi Network Model

Measure error of a particular node (x_i) as the energy in all springs for the node

$$\Sigma_j(L_{i,j} - || x_i - x_j ||)^2$$



Vivaldi Network Model

Measure error of whole system as the energy in all springs

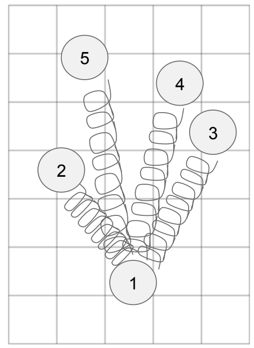
$$E = \Sigma_i\Sigma_j(L_{i,j} - || x_i - x_j ||)^2$$

Goal is to choose coordiantes x that minimize E

Vivaldi Centralized Algorithm

Big idea: for each node i,

1. figure out the total force of the springs between i and all nodes j
2. Move i by that force



Vivaldi Centralized Algorithm

While error(L,x) > tolerance
    For each node i:
        F = 0
        For each node j:
            //error of the spring between i and j
            $e = L_{ij} - ||x_i - x_j||$

            //add error to force vector of this spring
            $F = F + e \times u(x_i - x_j)$

    //move node i by a small step in the direction of the force
    $x_i = x_i + t \times F$

Vivaldi centralized algorithm

While error(L,x) > tolerance
    For each node i:
        F = 0
        For each node j:
            //error of the spring between i and j
            $e = L_{ij} - ||x_i - x_j||$

            //add error to force vector of this spring
            $F = F + e \times u(x_i - x_j)$

    //move node i by a small step in the direction of the force
    $x_i = x_i + t \times F$

We're assuming we know all RTTs for all pairs of nodes…

These RTTs are what we're trying to approximate!

Vivaldi centralized algorithm

While error(L,x) > tolerance

We're assuming we know all

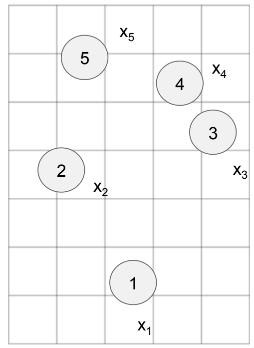
Two changes to make:

1. We need to calculate the coordinates of system using only a few RTTs
2. We need to do this using a distributed algorithm

$x_i = x_i + t \times F$

Vivaldi Distributed algorithm

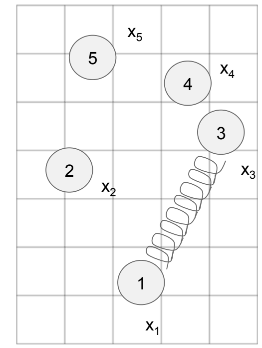
- Each node stores its own coordinate
- When it communicates with another node it measures RTT



Vivaldi Distributed algorithm

- Each node stores its own coordiante
- When it communicates with another node it measures RTT
- Moves itself proportional to the force within the spring

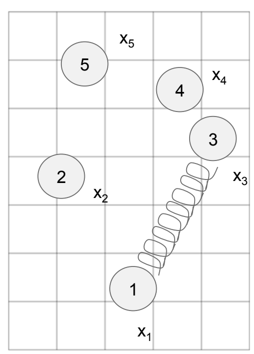
$$x_i = x_i + \delta\,(rtt - || x_i - x_j ||)\; u(x_i - x_j)$$



Vivaldi Distributed algorithm

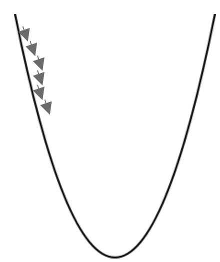
- Each node stores its own coordiante
- When it communicates with another node it measures RTT
- Moves itself proportional to the force within the spring

$$x_i = x_i + \delta\,(rtt - || x_i - x_j ||)\; u(x_i - x_j)$$

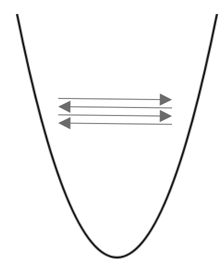


Vivaldi Distributed algorithm

$\delta = .0001$



$\delta = 1$



Vivaldi Distributed algorithm

Adapt δ. Converge quickly with a large δ; as we become more certain of our location, make δ smaller

Vivaldi distributed algorithm

//Given a sample rtt with node j, which has coordinate x_j, error e_j
vivaldi(rtt, x_j, e_j)

> //sample weight balances both local and remote errors
> $w = e_i / (e_i + e_j)$
>
> //calculate wieghted moving average of error of our samples
> e_i = weighted_moving_average(e_i, w, x_i, x_j, rtt)
>
> //Update local coordinates
> x_i = x_i + w (rtt - || x_i - x_j) u(x_i - x_j)

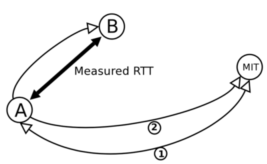
Evaluation methodology

Latency data: two datasets

1) Latency matrix for 192 hosts on PlanetLab network
    a) All pairs ping trace
2) Lacency matrix for 1740 DNS nameservers

rvers?

ative nameservers are on the same subnet

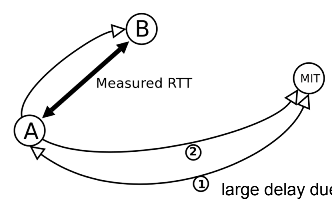


How to define latency?

Latency ?= minimum RTT

Not for King, since King can report a RTT less than true value

Use median to filter out transient congestion and packet loss



large delay due to high load at nameserver A >> delay btw A and B

Using the data

Using RTT matrices as inputs to a packet-level network simulator

Each nodes run the decentralized Vivaldi algorithm

Limitation of the simulator: RTTs do not vary over time, no queueing delays

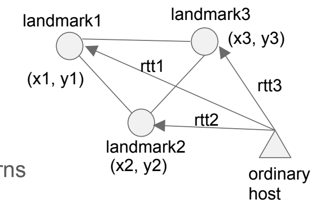
Why not simulating queueing delay?

Because this needs modeling underlying network infrastructure (model a model!)

Just stick to real data

Evaluation

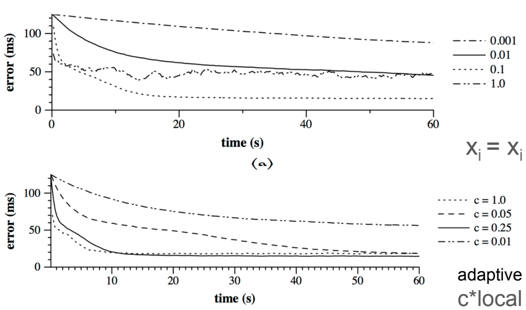
1. Effectiveness of the adaptive time-step δ
2. How well Vivaldi handle high-error nodes
3. Vivaldi's sensitivity to communication patterns
4. Vivaldi's repsonsiveness to network changes
5. Vivaldi's accuracy compared to that of global network positioning (GNP)



(x4, y4)

Effectiveness of the adaptive time-step δ

Network error: median of all nodes errors



(b)

fixed δ

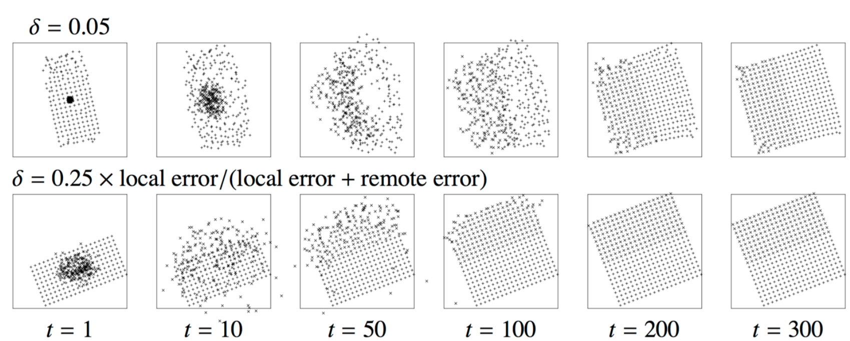
$$x_i = x_i + \delta\,(rtt - \|x_i - x_j\|)\,u(x_i - x_j)$$

adaptive δ =
c*local error/(local error + remote error)

local error = abs(predicted rtt - actual rtt)/actual rtt

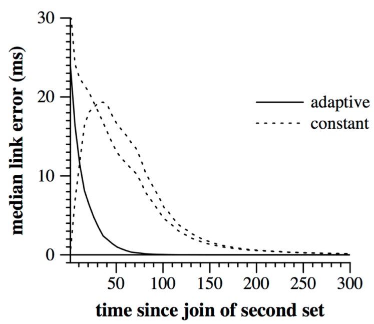
How well Vivaldi handle high-error nodes

Evolution of a stalbe 200-node network after 200 new nodes join



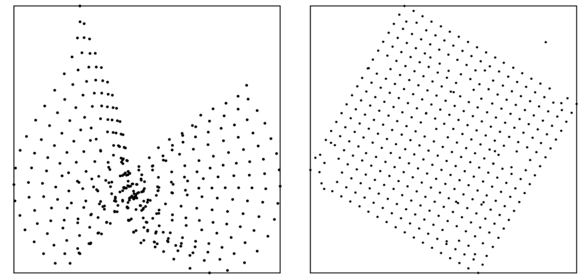
How well Vivaldi handle high-error nodes

Median link errors: median of all link errors



Vivaldi's sensitivity to communication patterns

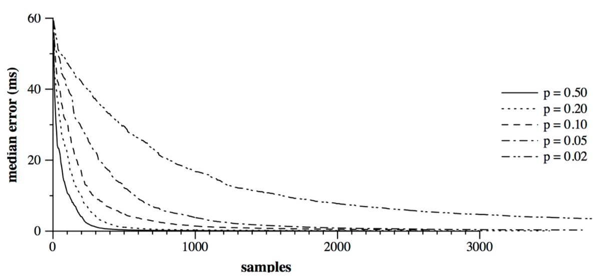
Pattern 1: communicate with four neighbors
Pattern 2: communicate with both neigbhors & long-distance hosts
(get a global sense of their place in the network)



How much long-distance comm. is necessary?

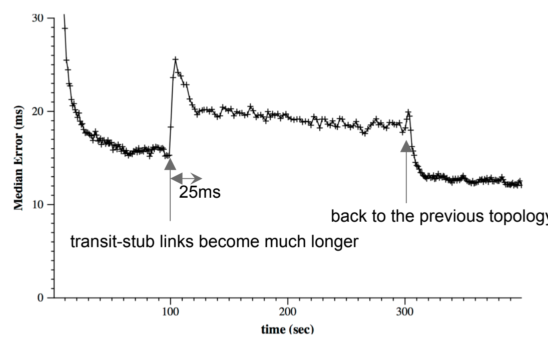
A grid of 400 nodes. Each node is assigned 4 neighbors and 4 faraway random nodes.

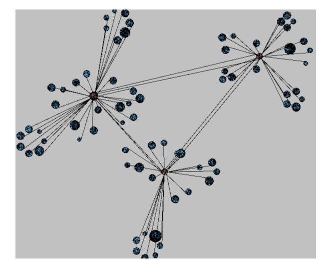
At each step, each nodes chooses a faraway node with probability p among these 8 nodes.



Adapting to network changes

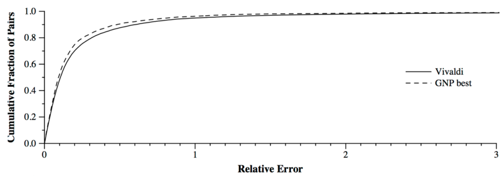
Use ITM tool to generate a 'transit-stub' topology of 100 hosts



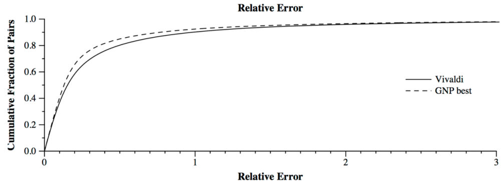


Accuracy

Compared with GNP best (Lowest median error)

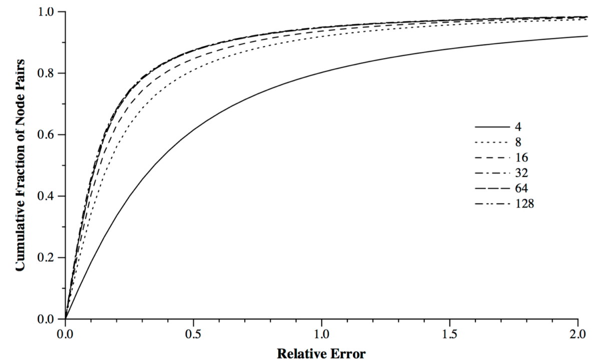


PlanetLab



King

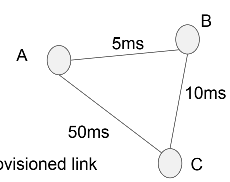
Accuracy vs. the number of neighbors



Suitability for embedding?

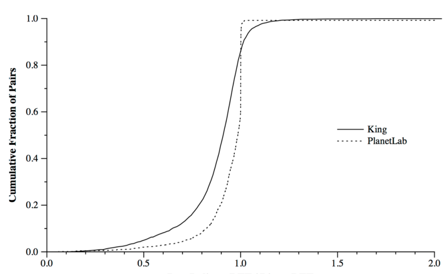
Triangluar inequaltiy violation

In Euclidean space, triangular inequality holds.
In network context, not necessary.



poorly provisioned link

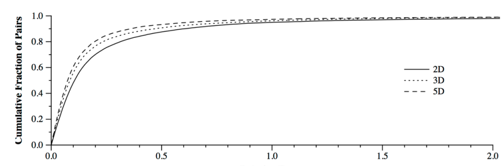
lowest indirect path / direct path = (5+10)/50



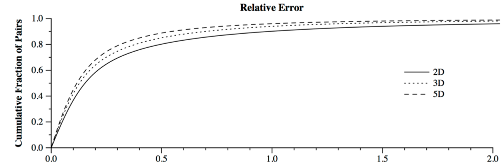
Conclusion: suitable

Euclidean space

$$
\begin{aligned}
[x_1, \cdots, x_n] - [y_1, \cdots, y_n] &= [x_1 - y_1, \cdots, x_n - y_n] \\
\|[x_1, \cdots, x_n]\| &= \sqrt{x_1^2 + \cdots + x_n^2} \\
\alpha \times [x_1, \cdots, x_n] &= [\alpha x_1, \cdots, \alpha x_n]
\end{aligned}
$$



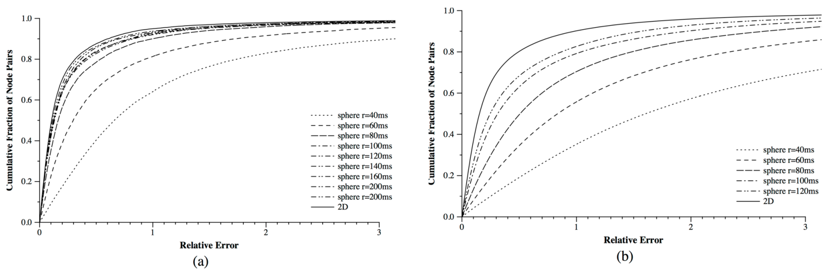
PlanetLab



King

Spherical coordinates

To model the shape of Earth



Euclidean space with heights

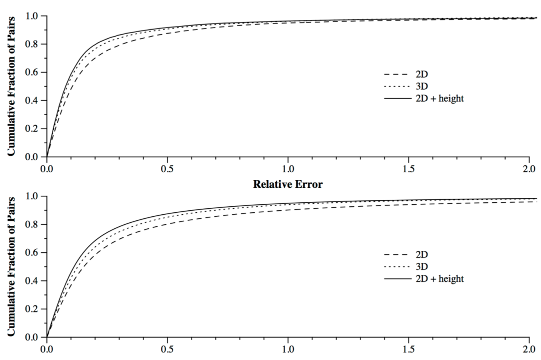
Euclidean space assumption: latency propotional to gegraphic distance

Access link could be slow in the case of cable modems and telephone modems

A height dimension for the access link

$$
\begin{aligned}
[x, x_h] - [y, y_h] &= [(x - y), x_h + y_h] \\
\big\|[x, x_h]\big\| &= \big\|x\big\| + x_h \\
\alpha \times [x, x_h] &= [\alpha x, \alpha x_h]
\end{aligned}
$$

Accuracy



Graphical comparison

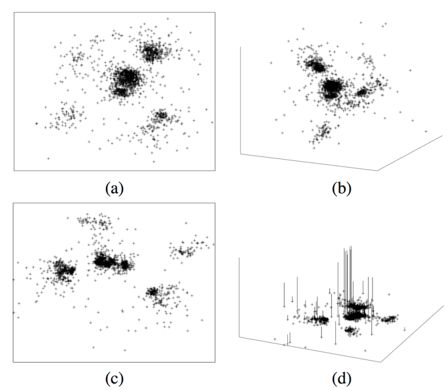
2-D Vivaldi w/o heights

3-D Vivaldi w/o heights

Dataset: King

3-D Vivaldi w/ heights projected to 2-D

Heights



Discussion

Strengths:

- Very elegently designed solution
- Evaluation shows the strenght of the solution

Weaknesses:

- Is the need still there?
  - How many p2p systems still out there?
  - Heterogenious distributed systems?