Virtual Coordinates for Internet Hosts

Reference:
Motivation

- To select the closest host in a set without sending probes to measure round-time latencies to all hosts

- Probing takes time and is not efficient for
  - P2P systems in which the set of hosts to consider is large
  - Each piece of data to be retrieved from the selected host is small (e.g., DNS)
Objective

- Synthetic coordinates for Internet hosts such that
  - the distance between the coordinates of any two Internet hosts predicts the communication latency between the hosts with a small error
  - Coordinates in virtual space rather than physical space because Internet latencies violate the triangle inequality

- Each host can compute its own coordinates after communicating with a small set of other hosts.
Global Network Positioning (GNP)

- The first coordinate system
  - T. S. Eugene Ng and Hui Zhang, “Predicting Internet Network Distance with Coordinates-Based Approaches,” Proceedings IEEE INFOCOM, June 2002.

- GNP uses several landmarks (special hosts) with known coordinates. An ordinary host computes its coordinates using latencies measured from probing three or more landmarks.
Vivaldi

- Fully distributed; no special host, each host runs the same protocol
- Scalable, has low message overhead, and adapts to network traffic and topology changes
- \textit{Height} introduced as a coordinate to improve prediction accuracy of Internet latencies
  - It captures transmission delay on the access link of a single-homed host
Prediction Error

\[ E = \sum_i \sum_j (L_{ij} - \|x_i - x_j\|)^2 \]

where

- \( L_{ij} \) actual RTT between nodes \( i \) and \( j \)
- \( x_i \) coordinates assigned to node \( i \)
- \( \|x_i - x_j\| \) distance between the coordinates of nodes \( i \) and \( j \)

Minimizing the squared-error function is equivalent to minimizing the energy in a physical mass-spring network.
Physical Spring Network Analogy

- Consider each pair of hosts as nodes connected by a spring with its rest length $L_{ij}$ set to the actual RTT

![Diagram showing three scenarios:
- A single spring at rest (N1 to N2 with 100 units).
- Longer spring (N1 to N2 with 150 units).
- Shorter spring (N1 to N2 with 50 units).]

Each force has a magnitude proportional to displacement.
Force between node i and node j

By Hook's Law:

\[ F_{ij} = (L_{ij} - \| x_i - x_j \|) \times u(x_i - x_j) \]

Scalar quantity: displacement of the spring from rest

Unit vector which gives direction of the force on node i

Force vector \( F_{ij} \) can be viewed as an error vector, which has a direction
Network of Springs

- The net force $F_i$ on a node $i$ is the sum of the forces from other nodes:
  \[ F_i = \sum_{j \neq i} F_{ij} \]

- The network may come to rest at a local (rather than global) minimum energy state.
Centralized Algorithm

```
// Input: latency matrix and initial coordinates
// Output: more accurate coordinates in x
compute_coordinates(L, x)
    while (error (L, x) > tolerance)
        foreach i
            F = 0
        foreach j
            // Compute error/force of this spring (1)
            e = L_{ij} - ||x_i - x_j||
            // Add the force vector of this spring to the total force (2)
            F = F + e \times u(x_i - x_j)
            // Move a small step in the direction of the force. (3)
            x_i = x_i + t \times F
```

Force on node i from node j
Simple Vivaldi algorithm after a single measurement

```
// Node i has measured node j to be rtt ms away, and node j says it has coordinates x_j.
simple_vivaldi(rtt, x_j)

// Compute error of this sample. (1)
e = rtt - \|x_i - x_j\|

// Find the direction of the force the error is causing. (2)
dir = u(x_i - x_j)

// The force vector is proportional to the error (3)
f = dir \times e

// Move a a small step in the direction of the force. (4)
x_i = x_i + \delta \times dir
```

Constant step size

Force on node i from node j
Coordinate update in Vivaldi

\[ x_i = x_i + \delta \times (rtt - \|x_i - x_j\|) \times u(x_i - x_j). \]

- This update for an individual measurement is identical to an individual force calculated within the loop of the centralized algorithm (multiplied by \( \delta \)).

- There is no guarantee that the updates will converge to a global minimum. The system may come to rest at a local minimum.
How to choose time step $\delta$?

System converges fast with larger $\delta$ but may jump back and forth across a low energy valley that a smaller $\delta$ would explore.

Another problem: How to handle nodes that have a high error in coordinates?
Adaptive time step $\delta$

- Simple adaptive time step as a fraction ($c_c < 1$) of the node’s estimated error

$$\delta = c_c \times \text{local error}$$

- The estimated error of the remote node should also be taken into consideration

$$\delta = c_c \times \frac{\text{local error}}{\text{local error} + \text{remote error}}$$
Vivaldi algorithm with adaptive time step

\[ \text{vivaldi}(\text{rtt}, x_j, e_j) \]

\[
\begin{align*}
// & \text{constants } c_e \text{ and } c_c \text{ are tuning parameters} \\
// & \text{Sample weight balances local and remote errors. (1)} \\
w &= e_i / (e_i + e_j) \\
// & \text{Compute relative error of this sample. (2)} \\
e_s &= \| x_i - x_j \| - \text{rtt} / \text{rtt} \\
// & \text{Update weighted moving average of local error. (3)} \\
e_i &= e_s \times c_e \times w + e_i \times (1 - c_e \times w) \\
// & \text{Update local coordinates. (4)} \\
\delta &= c_c \times w \\
x_i &= x_i + \delta \times (\text{rtt} - \| x_i - x_j \|) \times u( x_i - x_j)
\end{align*}
\]
Evaluation Methodology

- Packet-level network simulator running with RTT data
- Internet RTT latency data
  - Matrix of inter-host RTTs
  - Compute Vivaldi coordinates from a subset of these RTTs
  - Check accuracy of RTT computed to the corresponding entry in the full RTT matrix
- Two measured data sets
  - PlanetLab: 192 hosts, all-pairs pings, median RTT = 76 ms
  - King: 1740 DNS servers, median RTT = 159 ms
- Two synthetic data sets
  - Grid: perfect 2D coordinates
  - ITM: a topology generator
Basic idea in King method

- First DNS query is for a name in the domain of A. It returns the latency to A.
- Second query is for a name in the domain of B, but is sent first to A as a recursive server.
  - The difference between two queries is the latency between A and B
King data set

- Name servers are much more geographically diverse than PlanetLab hosts
- Avoid domains which are served by multiple name servers
- Around 100 million measurements were made over a week
  - median was used as the final RTT to filter out effects of congestion and packet loss
- About 10% of the original nodes were removed because they were obvious outliers in the data set
Using the Data

- Simulation setup
  - RTT matrix as input to a packet-level simulator
  - Each node sends a packet to some other node once a second
    - Send RPC packet
    - Delay by $\frac{1}{2}$ RTT time
    - Uses measured RTT of RPC to update coordinates
  - Error of a link is absolute difference between predicted and actual; error of a node is median of its link errors; error of system is median of node errors

- Limitation
  - The RTTs do not vary over time
Impact of Adaptive Time Step

3D for King data set

Static $\delta$

Adaptive $\delta$

large delta when node error is high, small delta when node error is low
**Evaluation - Robustness**

- **$\delta = 0.05$**
  - $t = 1$
  - $t = 10$
  - $t = 50$
  - $t = 100$
  - $t = 200$
  - $t = 300$

- **$\delta = 0.25 \times \text{local error}/(\text{local error} + \text{remote error})$**

- The evolution of a stable 200-node network after 200 new nodes join
  - **Constant $\delta$:** new nodes confuse the old (stable) nodes
  - **Adaptive $\delta$:** allows new nodes to find their places quickly without disturbing the existing order
Evaluation - Communication Patterns

- **(Left)** Sampling only nearby nodes can lead to coordinates that preserve local relationship but are far from correct globally.
- **(Right)** If nodes contact distant nodes as well, the accuracy will be improved.
Simulation of a grid of 400 nodes
- Each node has 8 neighbors; 4 are nearby and 4 are faraway
- Choose a faraway neighbor with probability $p$ at each time step
Adapting to Network Changes

- A synthetic “Transit-Stub” topology of 100 hosts using ITM
- 6D coordinates
  - $t = 100$: a transit stub link increased in length by a factor of 10
  - $t = 300$: the original length of the transit stub link restored
Adapting to Network Changes (cont.)

- **Joining cost**
  - Once there are enough nodes in the system, the joining cost for a new node is a small constant number of samples
  - regardless of the size of the network

- **Initiate a 1000-node network using King dataset**
  - Wait for the network to converge
  - Add 1000 new nodes, one at a time
  - New nodes converge after about 20 samples
Selection of a coordinate space

- Only inter-host latencies that satisfy the triangle inequality can be fit into a coordinate space.

\[ |x_1 - x_2| + |x_2 - x_3| \geq |x_1 - x_3| \]

- Internet latencies quite often do not satisfy this inequality.
Internet latencies

Many node pairs violate inequality
  - Most are small violations due to measurement inaccuracy
  - Only 5% severe violations
Euclidean Spaces

A higher dimension => better accuracy, higher comm. cost

PlanetLab

King
Spherical coordinates

Not so good – Internet paths do not wrap around the earth
Height vector

- Consists of a Euclidean coordinate augmented with a height
  - Euclidean: models a high-speed Internet core with latencies proportional to geographic distance
  - Height: models the time to travel the access link from the host to the core

- A packet travels the source node's height, then travels in the Euclidean space, then travels the destination node's height
Height vector

- Vector operations in this space

\[
[x, x_h] - [y, y_h] = [(x - y), x_h + x_h]
\]

\[
\| [x, x_h] \| = \| x \| + x_h
\]

\[
\alpha [x, x_h] = [\alpha x, \alpha x_h]
\]
Height Vector

Cumulative Fraction of Pairs vs. Relative Error for PlanetLab and King, showing 2D, 3D, and 2D + height cases.
Graphical Comparison (King data)

(c) Height vectors projected onto XY plane
Conclusions

- **Proposed a synthetic coordinate** which minimizes prediction error of Internet latencies between 2 hosts
  - Euclidean coordinate with height vector

- **Scaling to a large number of hosts**
  - A simple and decentralized algorithm (no special nodes)
  - Measurements to some distant nodes are necessary
  - Adaptive time step improves convergence
  - A new joining node can converge after about 20 samples

- **Adaptive to changing network conditions**
The End