Don’t Love Thy Nearest Neighbor

Cristian Lumezanu
Georgia Tech

Dave Levin, Bo Han, Neil Spring, Bobby Bhattacharjee
University of Maryland
Distributed Internet applications need the ability to find nodes that satisfy latency constraints
Find server that minimizes average latency to players
Theoretical optimum

Nearest neighbor
Find server that minimizes average latency to players (and provides fairness)
Cost optimization in the network coordinate space

Nearest neighbor is not enough

Theoretical optimum

Lowest cost node

Nearest neighbor

A

B

C

D

E
Sherpa

• Overlay network system that finds the lowest cost node under latency constraints
• Broad classes of latency-based cost functions, without knowing all the nodes that we are querying

1. Network coordinates
2. Voronoi regions
3. Compass routing
4. Gradient descent

Overlay setup
Querying/Node discovery
Compass routing
Gradient descent
Nearest neighbor
Lowest cost node
Evaluation

• Two latency data sets:
  – 1715 DNS servers, 213 PlanetLab nodes
  – network coordinate system: Vivaldi

• 1,000 queries: “find centroid of 30 nodes”

\[
\text{cost}(m) = \sum_{i=1}^{N} \frac{d(m, p_i)}{N} + 
\left(\max_i(d(m, p_i)) - \min_i(d(m, p_i))\right)^2
\]
Nearest neighbor is not enough

For 80% of the queries, the node chosen by Sherpa has a lower cost than the nearest neighbor.
For 65% of the queries, the node chosen by Sherpa is among the 10% lowest cost nodes.
Conclusions and Future Work

• Generalized node selection with network coordinates
• Sherpa finds the lowest cost node
• Implementation
• Cost functions
• Other applications: split TCP, route avoidance