(Project Report for CS 237 - Distributed Systems Middleware)

CS237 Project final report

Kyle E. Benson and Zhipeng Huang Donald Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, California 92697
Email: kebenson@uci.edu, zhipengh@uci.edu

I. Algorithm

There are six routing heuristic algorithms that have been tested and compared in the simulation: orthogonal distant path heuristic, new region heuristic, new angle path heuristic, distant-dependent path heuristic, furthest first path heuristic and closest first path heuristic. The New Angle, Distance-Dependent, and Furthest-First heuristics are new contributions from this project.

A. Orthogonal Distant Path Heuristic

The intuition of this heuristic is to avoid the straight path, without diverging from it too much. It strikes a middle ground by choosing a path at an ideal angle of $45°$, which makes the angle at the top orthogonal, hence the name.

Algorithm 1:

begin

/* Sensor, overlay and server node are a, c, b respectively */

$idealDist = |0.5 \cdot dist(a,b)|$

$perpDist = |\sin(angA) \cdot dist(a,c)|$

if $angA > \frac{\pi}{2}$ **or** $angA + angC < \frac{\pi}{2}$ **or** $perpDist > dist(a,b)$ **then** $likelihood = 0$

else $likelihood = 0.5 \cdot ((1.0 - (||perpDist| - |idealDist||/idealDist)^2) + (1.0 - (|\frac{\pi}{2} - angC|/\frac{\pi}{2})^2))$

end

B. New Region Heuristic

The intuition of this heuristic is to avoid regions that have been previously attempted unsuccessfully. We assume that no further attempts to contact such a region will succeed.

C. New Angle Path Heuristic

This heuristic attempts paths along new angles different from the ones previously attempted.

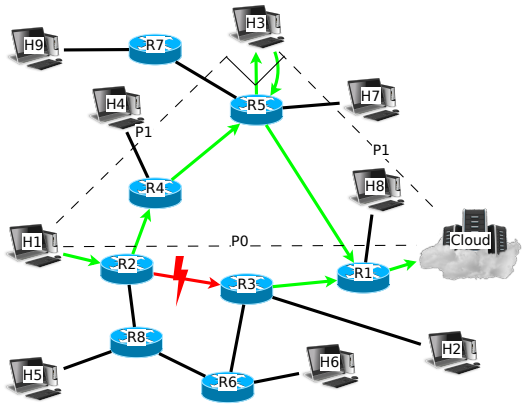

Fig. 1. Orthogonal Distant Path Heuristic

Algorithm 2:

begin

if $peer.region \in regionsAttempted$ **then** $likelihood = 0$

else $likelihood = 1.0$

end

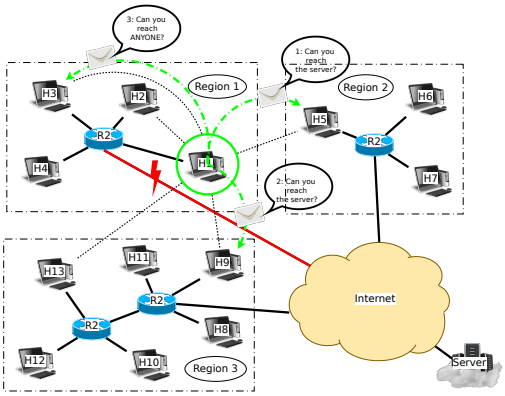

Fig. 2. New Region Heuristic

Algorithm 3:

begin

$angle = Angle(overlay, server)$

if $angle = \pi$ **or** $angle = 0$ **then** $initLikelihood = 0$

else if $angle < \pi$ **then** $initLikelihood = \cos(angle - \frac{\pi}{4})$

else if $angle > \pi$ **then** $initLikelihood = \cos(2 \cdot ((2\pi - angle) - \frac{\pi}{4})/3)$

foreach $path \in pathsAttempted$ **do** $thisLikelihood = |\frac{\sin(angle)}{2}|$

$newLikelihood* = thisLikelihood$

end

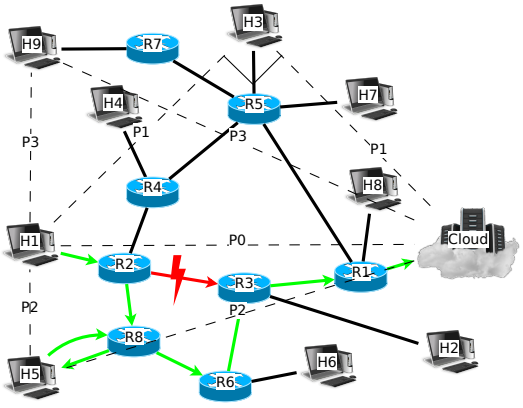

Fig. 3. New Angle Path Heuristic

Algorithm 4:

begin

/* minDist = some reasonably small number */

$idealDist = 0.5 \cdot dist(sensor, server)$ $dist = dist(sensor, overlay)$ **if** $dist <= minDist$ **then** $likelihood = 0$

else if $minDist < dist < idealDist$ **then** $likelihood = dist^2/idealDist^2$

else if $dist >= idealDist$ **then** $likelihood = idealDist/dist$

end

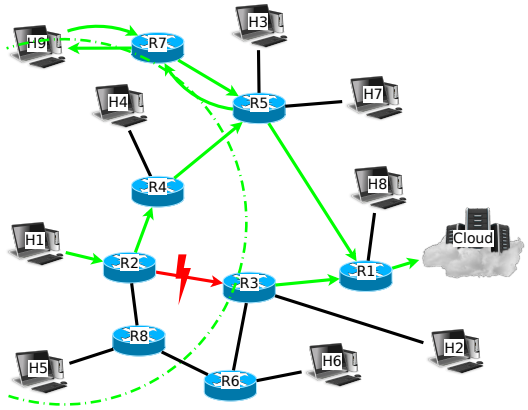

Fig. 4. Distance-Dependent Path Heuristic

D. Distance-Dependent Path Heuristic

This heuristic attempts paths that use overlay nodes at an ideal distance from the sensor. This ideal distance is chosen as a radius that reaches half-way to the server.

E. Furthest First Path Heuristic

This heuristic considers overlay peers located further away to be more likely candidates.

Algorithm 5:

begin

/* minDistance = some constant small number */

if $dist(a, b) < minDistance$ **then** $likelihood = 0$

else $likelihood = (dist(a, b) - minDistance)/dist(a, b)$

end

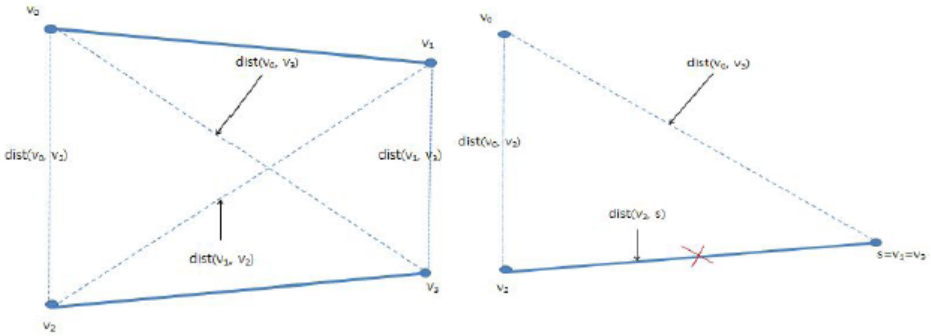

Fig. 5. Furthest First Path Heuristic

F. Closest First Path Heuristic

The intuition of this heuristic is to contact nearby overlay nodes that have found a path out of the local region. We found, however, that this approach does not always work well in practice, likely due to the path similarity inherent with nearby nodes.

II. Conclusion

In this paper, we described our recent additions to the GeoCRON simulator. In particular, we:

- Switched from Rocketfuel to BRITE for creating ns-3 network topologies
- Added the New Angle heuristic, which repeatedly tries overlay nodes at diverse angles
- Added the Furthest-First heuristic, which attempts to contact overlay nodes furthest from the source
- Added the Distance-Dependent heuristic, which tries to pick overlay nodes at an ideal distance from the source, preferring nodes further away over those close by

Algorithm 6:

begin

/* maxDistance = some constant large number */

if $dist(a, b) > maxDistance$ **then** $likelihood = 0$

else $likelihood = (maxDistance - dist(a, b))/maxDistance$

end

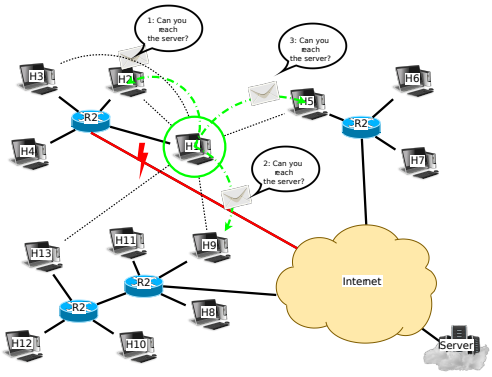

Fig. 6. Closest First Path Heuristic

- Defined a new method for assigning regions to nodes in which the region of study is broken up into a grid, where each cell is a constant size

We ran simulations on topologies of 10,000 nodes and 25 regions, comparing the peformance of each heuristic with each other. The results were inconclusive, demonstrating that further refinement and/or combinations of heuristics are necessary to improve the delivery ratio. For example, the New Angle and Distance-Dependent heuristics may be combined with varying weights to pick nodes both far away from the source as well as along diverse paths.

There are certain other aspects that need to be improved in the future. First of all, a detailed comparison between the six heuristics should be explored regarding aspects such as difference in convegence time, latency, expected delivery ratio, etc. Moreover, the purpose of the simulation of the six algorithms should be more clear.

III. Results

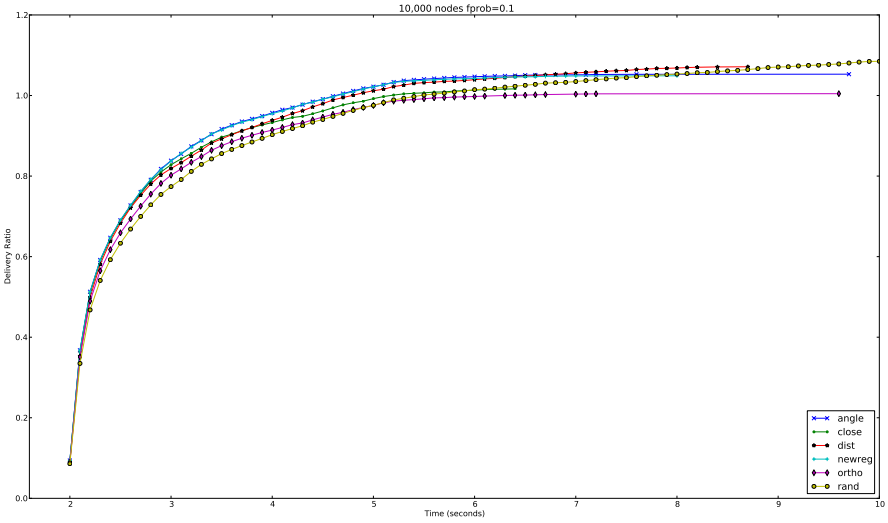

Fig. 7. Delivery ratio of the various heuristics for a failure probability of 0.1. One can notice the way the heuristics sometimes overcome another's delivery ratio as time evolves and different paths are considered. NOTE: due to some necessarily aggressive data cleaning, the delivery ratio is exaggerated and skewed.

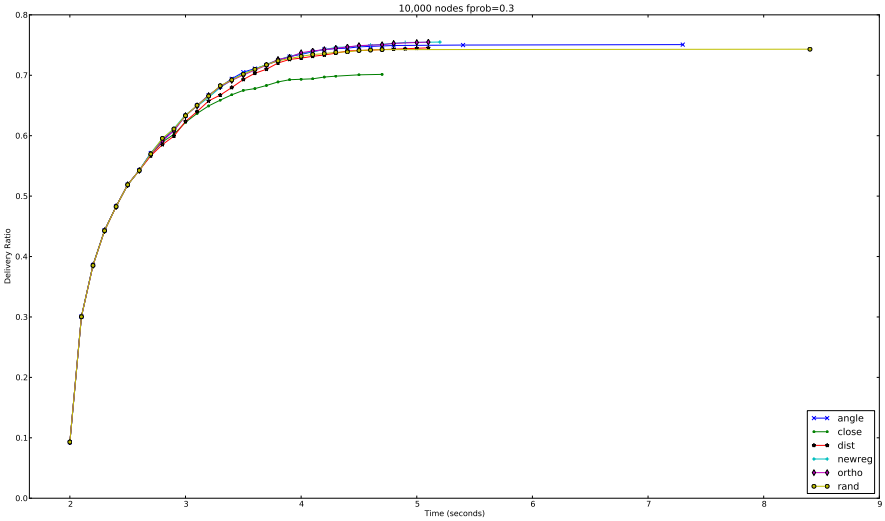

Fig. 8. Delivery ratio for a failure probability of 0.3. Note that the more restrictive heuristics stop finding alternative connections before those that pick from more diverse areas, such as Random and New Angle. This pattern, as well as the relatively poor performance of Nearest Neighbor, grows proportionally to the failure probability.