Lab2: Analyzing Greedy Shortest Path Algorithms

Shounak Dhar, Kamran Saleem

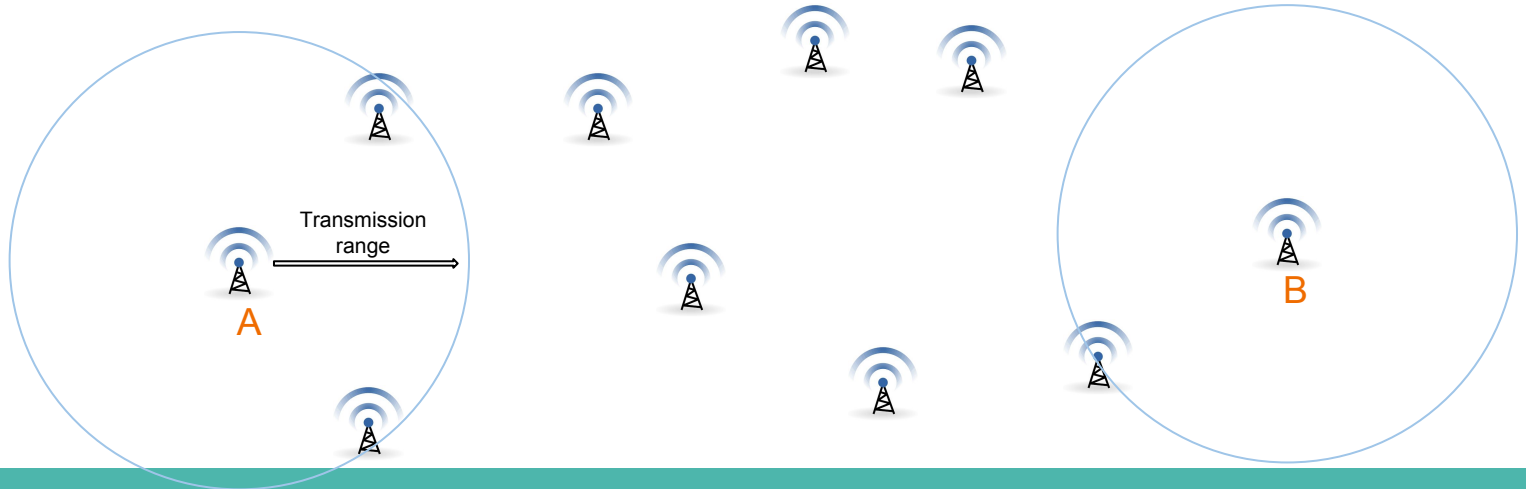
Oct 13, 2015

Problem statement

- Routing packets in wireless networks

Problem statement

- Consider a 2D plane;
- There are transmission towers at various points in the plane
- The farthest a tower can transmit is called it's transmission range
- Your job is to route a packet from a given tower to another given tower



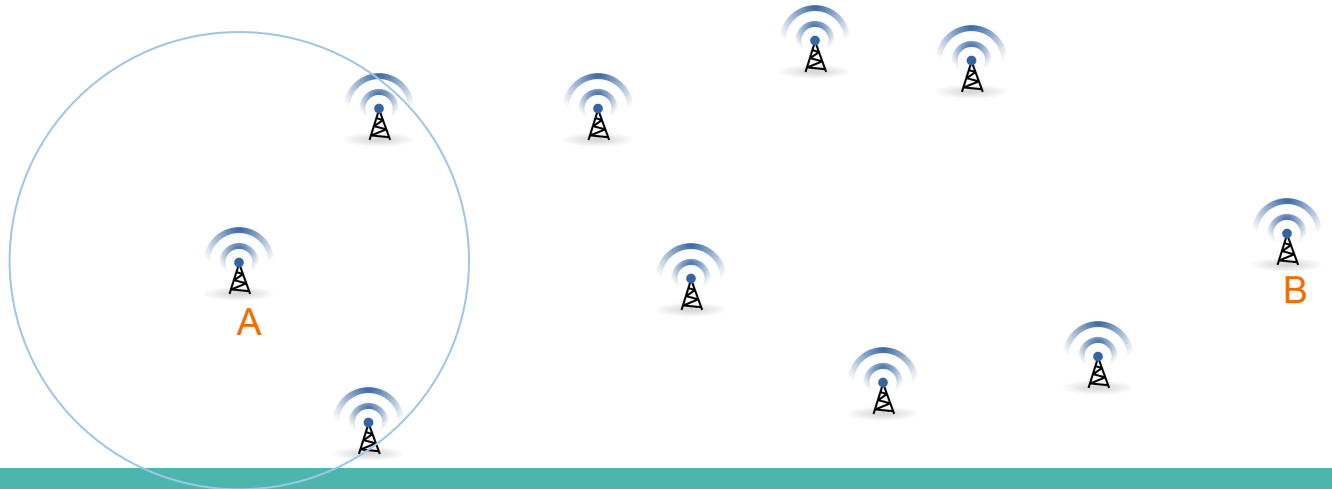
Problem statement (continued...)

- You will try 3 different algorithms to solve this problem
  - Greedy Perimeter Stateless Routing
  - Dijkstra's algorithm with minimum latency
  - Dijkstra's algorithm with minimum hops

- You will repeat this for different values of transmission range (that we will provide as input)

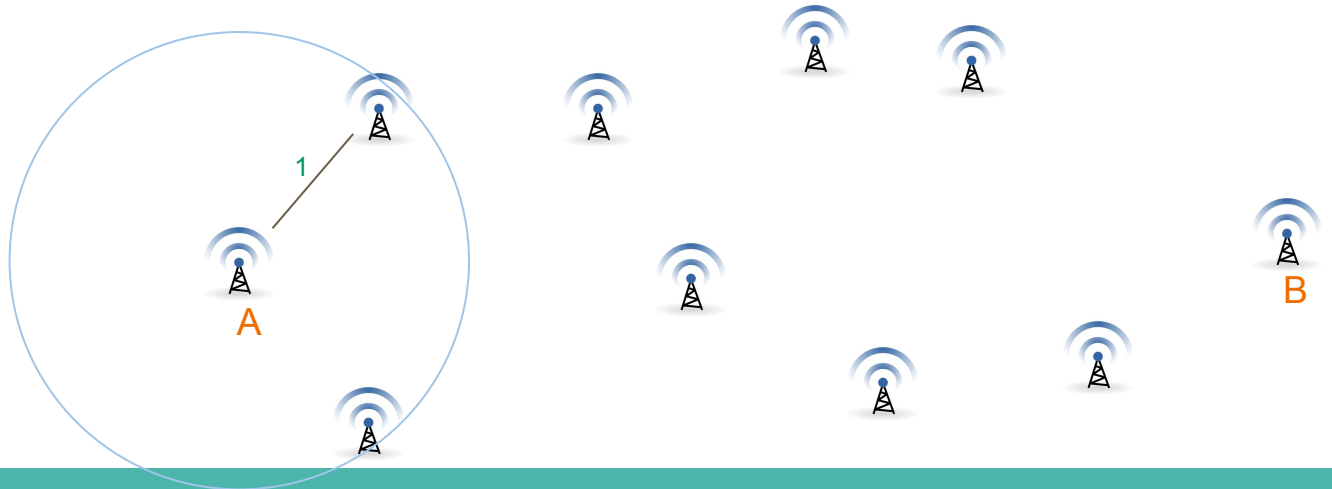
GPSR

- Greedy Perimeter Stateless Routing
  - Connects the source to a vertex within its radius that is closest to the sink
  - If no vertex is closer, moves to perimeter mode (we won't implement this)
  - In this assignment, restrict GPSR to failing when it cannot find a closer vertex



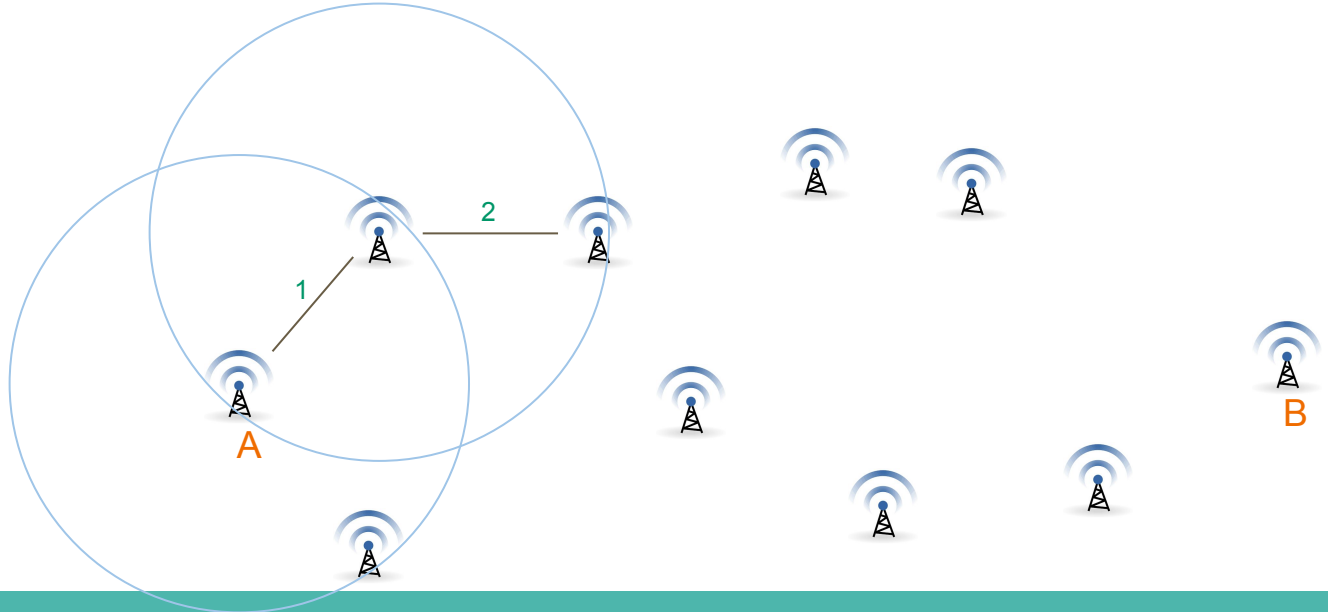
GPSR

- Greedy Perimeter Stateless Routing



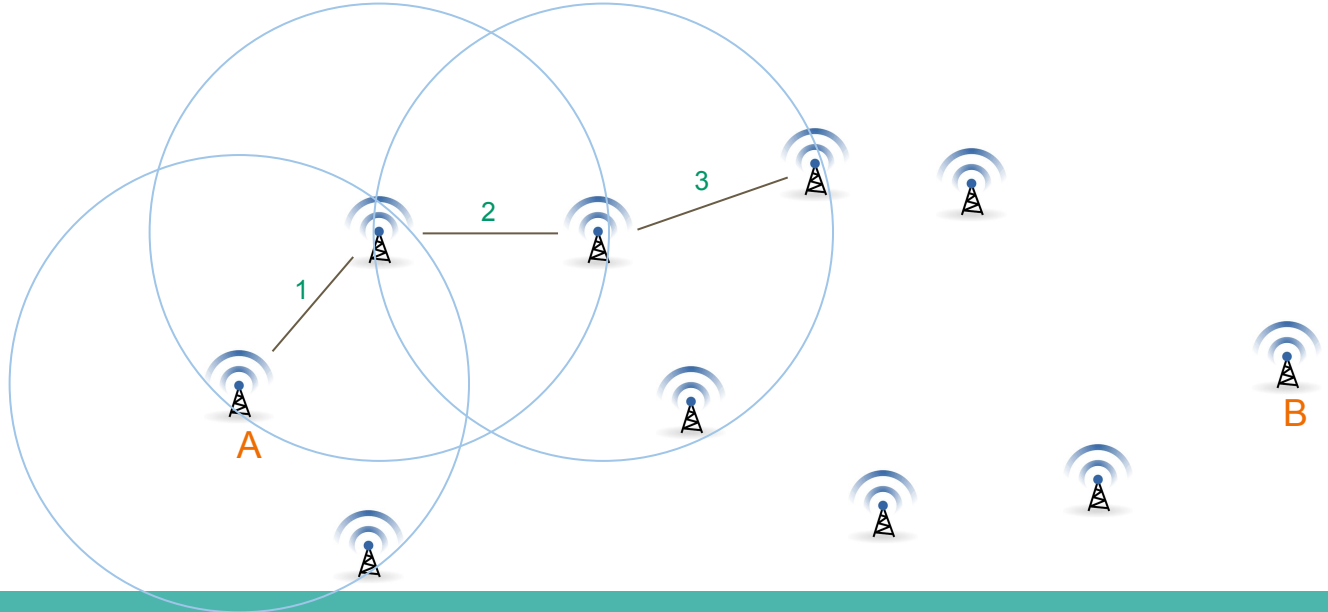
GPSR

- Greedy Perimeter Stateless Routing



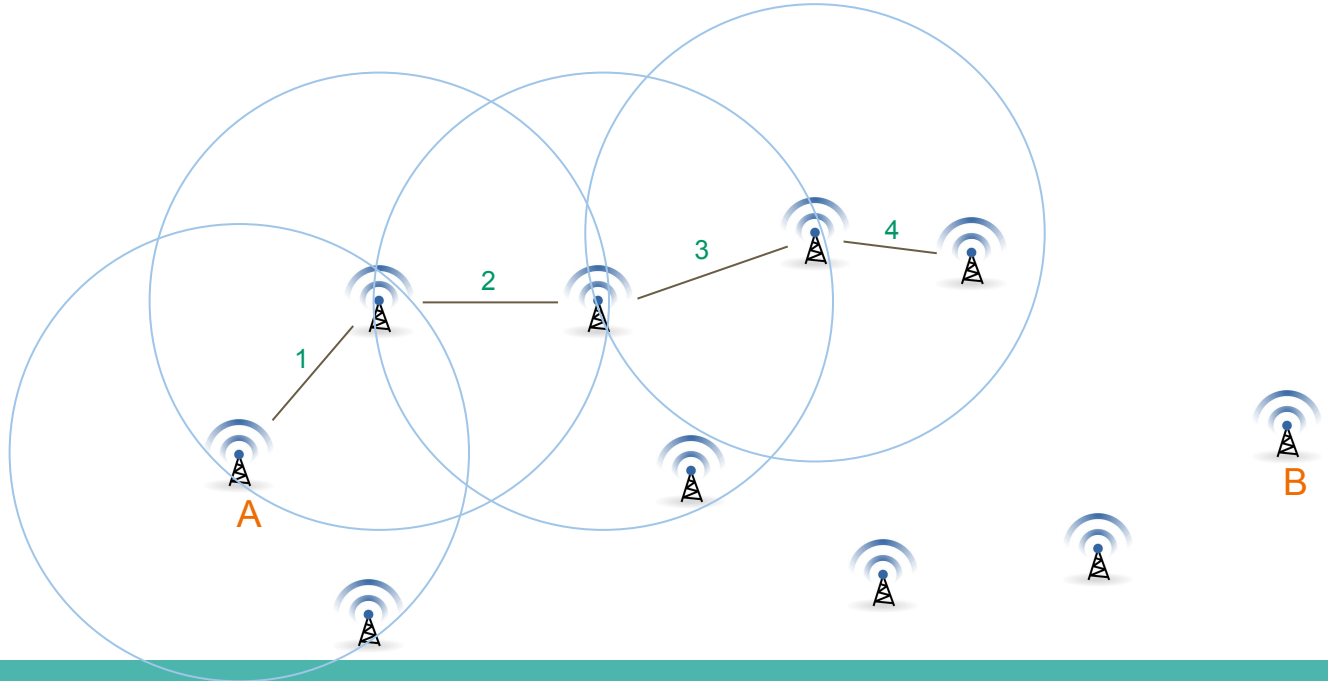
GPSR

- Greedy Perimeter Stateless Routing



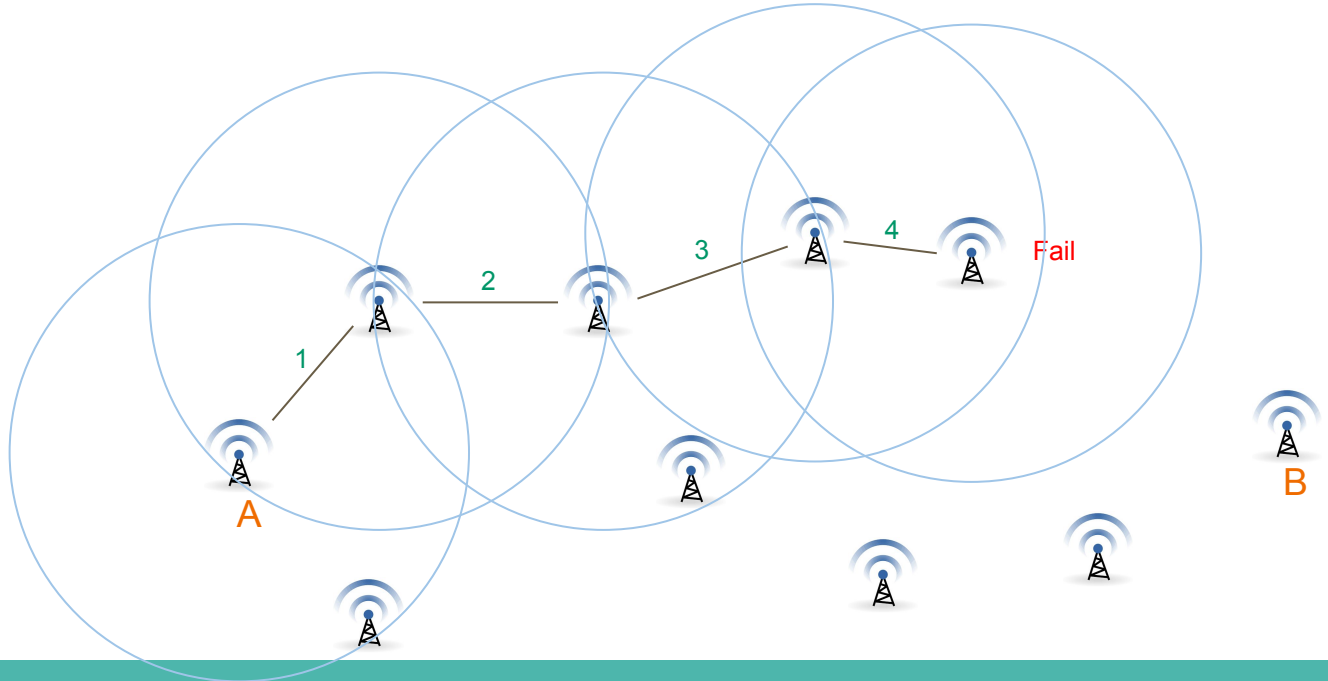
GPSR

- Greedy Perimeter Stateless Routing



GPSR

- Greedy Perimeter Stateless Routing



Djikstra's Algorithm

- Finds shortest path after analyzing adjacent neighbors of every vertex
- Greedy Algorithm since it minimizes path at each vertex
- http://optlab-server.sce.carleton.ca/POAnimations2007/DijkstrasAlgo.html

Provided Files

- VertexNetwork.java
- Driver.java
- Vertex.java
- Edge.java
- Program2.java

Do not modify

Modify

Provided Files

- VertexNetwork.java
  - public void gpsrAllPairs(boolean print);
    - calls the GPSR algorithm for all pairs of vertices and displays the number of successful runs as well as the average time taken for these successful runs
  - public void dijkstraLatencyAllPairs(boolean print);
    - calls Dijkstra's algorithm (for minimum latency) for all pairs of vertices and displays the number of successful runs as well as the average time taken for these successful runs
  - public void dijkstraHopsAllPairs(boolean print);
    - calls Dijkstra's algorithm (for minimum hops) for all pairs of vertices and displays the number of successful runs as well as the average time taken for these successful runs

Provided Files

- Driver.java
  - main method
  - parses arguments to form transmissionRange Arraylist
  - Invokes gpsrPath, dijkstraPathLatency and dijkstraPathHops for all values of transmissionRange

Provided Files

- Vertex.java
  - Data Structure for a vertex

- Edge.java
  - Data Structure for an edge

Provided Files

- Program2.java
  - ONLY File to be modified (may add more .java files as needed)
  - overrides gpsrPath, dijkstraPathLatency, dijkstraPathHops

Assignment Submission Guidelines

- Please submit single .zip through Assignment section on Canvas
- All files should be at the root of your .zip archive

Tips

- In Eclipse make use of Ctrl+ SPACE
- Make as much use of Piazza as possible (no question is stupid)
- Afraid of posting personal code? - use Private