Practice #4: Distance Vector Routing in NS3

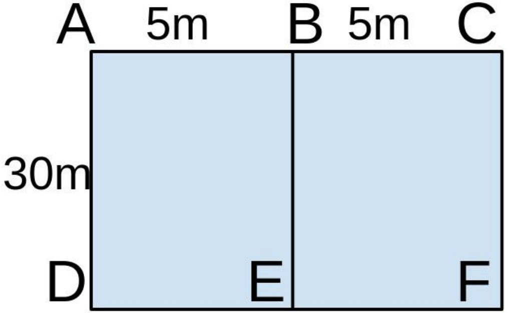
Boyan Kostadinov (boyanyk@kaist.ac.kr), Sangwoo Kim (woodmac@kaist.ac.kr)

Due: 12/1 Tue, 11:59 pm.

Goal

- Better understand ad-hoc distance vector routing by means of practical examples.
- In this task, you are required to use **NS-3** for network simulation. That allows us to simulate a wireless connection between the nodes, as wireless is necessary for ad-hoc.
- You can assume that every node in this practice is a wireless device.

Task 1 - MobilityHelper Tutorial



1. Write a program to simulate a simple network grid.
   Copy the **task1.cc** file to **scratch/** directory and run by **"`./waf --run scratch/task1`"** to execute the code. **(15pts)**

   1.1. **Setup a grid of nodes as shown in the picture on this slide (not drawn to scale) (5 pts)**
   - Use the same width and height as shown in the figure. Those, as well as the number of nodes, can be changed in the constructor.
   - You need to end up with 6 nodes across 2 rows.
   - Look at `Task1::Task1()` for the variables and at `Task1::CreateNodes()` for the grid setup code.
   - In the report, write 1-2 sentences about how you did it.

   1.2. **Run the file using the command described above. Look at the `task1.routes` file available in the scratch directory.**

   1.2.1. Which pairs of nodes are one hop away from each other? **(1pt)**

   1.2.2. Why/How are they one hop away? **(3pts)**

   1.3. **Now, change the width of the grid. What is the maximum value (of the width) such that C and D are still one hop away from each other? Please use integer values only (3pts)**

   1.4. **Now, change the height of the grid. What value would cause nodes [ABC] to not be connected to any of [DEF]? (3pts)**

Task 2 - Ping application

2. Create a one-dimensional 10 node deployment. For that, use height 0 and the distance for CD you found in 1.3.

Modify skeleton file task2.cc (15pts)

2.1. Draw a diagram/graph of this deployment (include edge distances) and name each node (A to J). (3pts)

2.2. Look at `Task2::InstallApplications`. This function assigns a 'ping' application to a given node, using its IP - it is setup to ping from node C to node J.

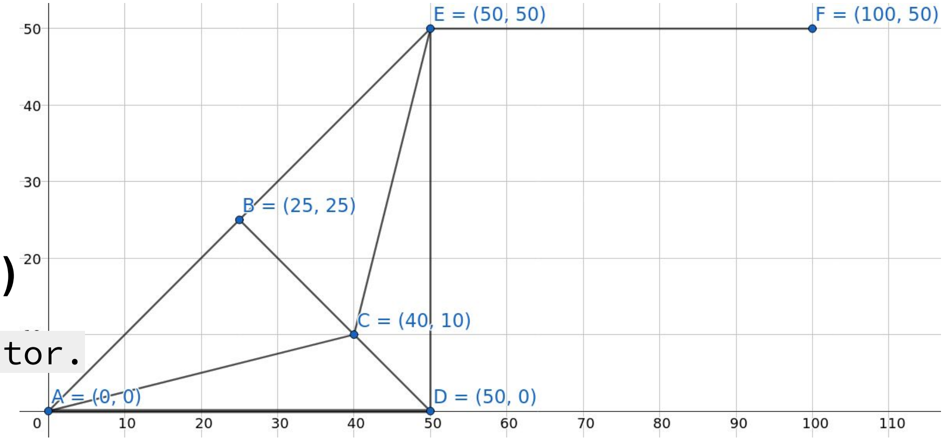
2.2.1. Run the code. What is the terminal output? Include a screenshot (doesn't need to capture the full output) **(2pts)**

2.2.2. Look at the rest of the function. It contains code to move a node to a different position. Uncomment the code. Modify it such that you move node H in a position between nodes B and C. **(3pts)**

2.2.3. Draw a new graph of this deployment. Only show edges where connection between nodes exists **(2pts)**

2.2.4. Save the routing table `task2.routes` to a different file. Then, change the code to ping from node J to node C. Again, move node H to the position between nodes B and C. What are the differences between the routing tables? Discuss **(5pts)**

Task 3



3. Various distance routing situations (20pts)

3.1. Recreate diagram using `ListPositionAllocator.`
You may find the following resources useful:
ListPositionAllocator; MobilityHelper

3.1.1. Look at `Task2::InstallApplications`
The ‘traceroute’ function works similarly to the ‘ping’ function. It is currently executing traceroute from node C to node E (IF you’ve implemented the position allocator correctly). Run traceroute from A -> F. Show terminal output **(3 pts)**

3.1.2. Move node B to (0,50) and run again. What do the results tell you? What do you think of the alternative paths? What might have caused the simulator to select this path? Discuss briefly **(5pts)**

Task 3 - Traceroute application

3. Various distance routing situations

3.2. Recreate diagram from Slide 14: Appendix 1: Task 3.2 diagram. You can use the same task3.cc file from 3.1, just comment out the code from 3.1 clearly.

3.2.1. Run traceroute from node C to node E. Do you think there would be a significant difference between the two possible paths? **(2pts)**

3.2.2. From previous two experiments (3.1, 3.2.1), what conclusion can you draw about what paths NS3 prioritises? Why could that be? **(2pts)**

3.2.3. Now, imagine that instead of 2 hops of 50m, there were 1000 hops of 50m (path A), and instead of 30 hops of 30m, there were 1001 hops of 30m (path B).

3.2.3.1. What route would the simulator take? Why? **(3pts)**

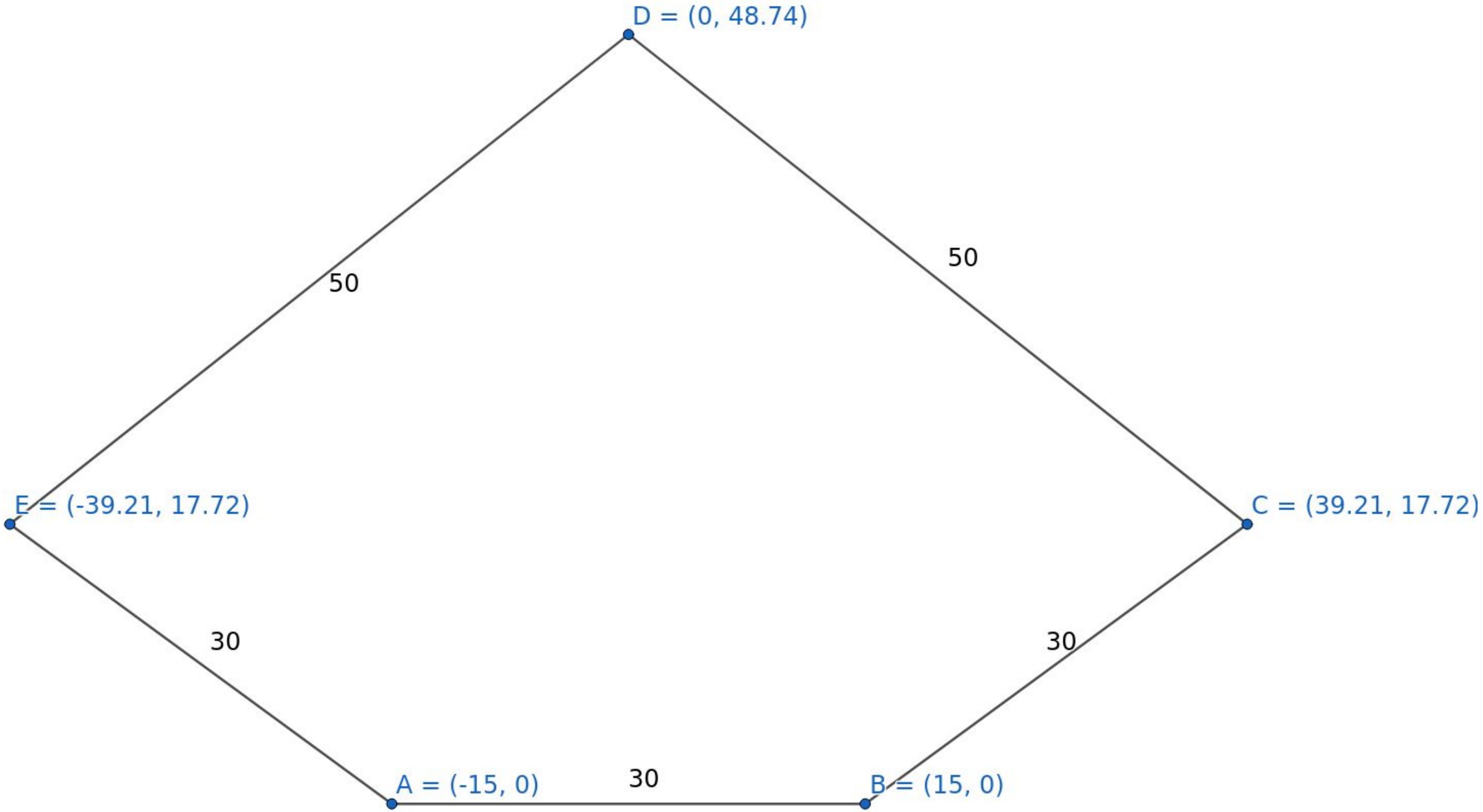
3.2.3.2. What are the pros and cons of each path? Which one do you think is better? Discuss in 3-4 sentences **(5pts)**

Submission Requirements

Submit your report in **pdf format**, and the files for each task with the filename of ***taskX_20xxxxxx_YourName.cc.*** **The report should be no more than 5 pages.** Submit a zip file on KLMS.

Due: 12/1 Tue, 11:59 pm. *Plagiarism & Late submission: 0 points.*

Appendix 1: Task 3.2 diagram



Q&A From Lab Session

Q1 - Is it fine to modify the source code by quite a lot? Or is there some kind of copyright issue?
A - Modifying the skeleton code is fine as long as it still works. However, don’t modify code outside of the scratch directory.

Q2 - In task2, I'm confusing the expression. After 2.2.2 I think the node is moved to the particular position. But in .2.2.4 it said again move.... Do I have to get back to the original code after 2.2.2? Or is it just emphasizing the condition?
So the difference in two routing tables is where the ping starts from right?
the only difference!
Thank you!!
A - It says move again just to emphasize - 2.2.4 requires the same node movement as 2.2.2. Then, when executing the ping command there should be a difference in the routing tables, depending on source and target of ping.

Q3 - Oh, also what does it mean by:

```
/**
   * \param argc is the command line argument count
   * \param argv is the command line arguments
   * \return true on successful configuration
  */"
```

Above the Run function of task1.cc
A - it is just standard code (and comments) for passing arguments to a c++ program. It is not necessary to use it