Practice #4 Report

20170481 Yuseung Lee

Task 1: Mobility Helper

[1.1]
In *Task1::Task1()*, I set the variable *size* to 6, *width* to 5, and *height* to 30. In *Task1::CreateNodes()* I changed the *"GridWidth"* parameter of the *SetPositionAllocator* to *UintegerValue(3),* so it allocates 3 nodes each row and switches to the next row.

[1.2]
(1.2.1)
The pairs of nodes that are 1 hop are away from each other are: (A, B), (A, C), (A, D), (A, E), (A, F), (B, C), (B, D), (B, E), (B, F), (C, D), (C, E), (C, F), (D, E), (D, F), (E, F). We can see that every pair of nodes are 1 hop away.

(1.2.2)
In an ad hoc network two nodes are 1 hop away if they are contained in each other's wireless link coverage. For example, Node A and D are 1 hop away because Node A lies in Node D's link coverage area and Node D lies in Node A's link coverage area.

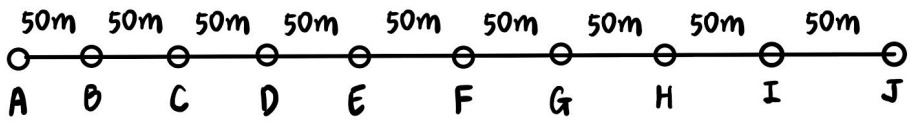
[1.3]
When I set *width*=20 & *height*=30, Node C and D were still 1 hop away. But when I changed the *width* to 21, Node C and D were not 1 hop away anymore. So the maximum value of the width for C and D to be 1 hop away is **20m.**

[1.4]
When I set *width*=5 & *height*=51, Node A and D were still connected through 1 hop. But when I changed the *height* to 52, Node A and D were not connected anymore and thus nodes [ABC] were not connected to any of [DEF]. So *height* **greater than or equal to 52m** would disconnect [ABD] & [DEF].

Task 2: Ping Application

(In this task, I took the distance for CD in [1.3] which is 50m, and used it as the width.)

[2.1]



Above is the initial deployment of the given network.

[2.2]
(2.2.1)

```
ns3@ubuntu:~/Documents/ns-allinone-3.31/ns-3.31$ ./waf --run scratch/task2
Waf: Entering directory `/home/ns3/Documents/ns-allinone-3.31/ns-3.31/build'
[1996/2067] Compiling scratch/task2.cc
[2027/2067] Linking build/scratch/task2
Waf: Leaving directory `/home/ns3/Documents/ns-allinone-3.31/ns-3.31/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.032s)
Creating 10 nodes 50 m apart.
Starting simulation for 100 s ...
PING 10.0.0.10 - 56 bytes of data - 84 bytes including ICMP and IPv4 headers.
64 bytes from 10.0.0.10: icmp_seq=0 ttl=58 time=1303 ms
64 bytes from 10.0.0.10: icmp_seq=1 ttl=58 time=304 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=58 time=5 ms
```

```
64 bytes from 10.0.0.10: icmp_seq=94 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=95 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=96 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=97 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=98 ttl=58 time=5 ms
64 bytes from 10.0.0.10: icmp_seq=99 ttl=58 time=5 ms
--- 10.0.0.10 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99999ms
rtt min/avg/max/mdev = 5/20.97/1303/132.9 ms
```

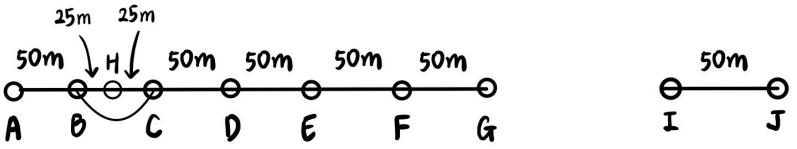
The above is the terminal output of *task2.cc.* We can see that the ping application sends a total of 100 packets from Node C to J and prints out the information in the received packets in C: ICMP sequence number, TTL and time. In this simulation there was 0% packet loss so every ping from Node C to J has succeeded.

(2.2.2)

```
Task2::InstallApplications ()
{
  // Ping from Node C to Node J
  V4PingHelper ping (interfaces.GetAddress (9));
  ping.SetAttribute ("Verbose", BooleanValue (true));

  ApplicationContainer p = ping.Install (nodes.Get (2));
  p.Start (Seconds (0));
  p.Stop (Seconds (totalTime) - Seconds (0.001));

  // Move Node H to between B and C
  Ptr<Node> node = nodes.Get (7);   // Get Node H
  Ptr<MobilityModel> mob = node->GetObject<MobilityModel> ();

  Simulator::Schedule (Seconds (totalTime / 3), &MobilityModel::SetPosition, mob, Vector (75, 0, 0));
}
```

I changed the code as above which moves Node H to a position between B and C. In the last line, I scheduled the simulator to move Node H to position (75, 0, 0) after one third of the total simulation time.

(2.2.3)



Above is the graph after Node H has been moved to a position between B and C. Since I set the position of H to be the exact middle of B and C, the distance of BH and HC are both 25m. Also there is no node between Node G and I because we found out in Task 1 that the maximum link coverage radius for each node is about 50m.

(2.2.4)

The main differences between routing tables for ping application from C to J and from J to C are as following:

	Ping from C to J	Ping from J to C
1	In intermediate nodes (D/E/F/G), the path to the destination node (J) is still valid (flag is set UP).	In intermediate nodes (E/F/G), the path to the destination node (C) is invalid (flag is set DOWN).
2	The source node (C) has a valid path to the destination node (J) with 7 hops.	The source node (J) is still IN_SEARCH for the path to the destination node (C).

The differences derive from the behavior of AODV and more generally distance vector routing. Since AODV is an on-demand protocol, the routing table in each node is updated only when it receives data from its neighbor. So in the first simulation (C to J), even after H is moved nodes D/E/F/G still think that it has a valid path to J because they don't receive the RREP packet coming from J, and thus don't update their routing tables. They are only updated by the RREQ packet from C. But in the second simulation (J to C), nodes E/F/G have invalidated their route to C because they receive neither RREQ nor RREP packet, which indicates that they are left out of the path from J to C.

The second difference is from the fact that in distance vector protocols, routing tables are only shared by neighboring nodes. In the first simulation, C has neighbors B, H, D and node D has a path to J. So C maintains its path to J which goes through D. But in the second simulation, J only has neighbor I, and I doesn't have a path to C anymore so J invalidates its path C. Then J searches for a path to C that doesn't go through I, but there is no other neighbor so it is still IN_SEARCH.

Task 3: Traceroute Application

[3.1]

(3.1.1)

```
Creating 6 nodes.
Starting simulation for 100 s ...
Traceroute to 10.0.0.6, 30 hops Max, 56 bytes of data.
1 10.0.0.3 264 ms 0 ms 0 ms
2 10.0.0.5 1 ms 1 ms 1 ms
3 10.0.0.6 10 ms 2 ms 2 ms

Trace Complete
```

Above is the terminal output of traceroute from A to F. We can see that the chosen route from A to F is A→C→E→F.

(3.1.2)

```
Creating 6 nodes.
Starting simulation for 100 s ...
Traceroute to 10.0.0.6, 30 hops Max, 56 bytes of data.
1 10.0.0.3 264 ms 0 ms 0 ms
2 10.0.0.5 1 ms 1 ms 1 ms
3 10.0.0.6 10 ms 2 ms 2 ms

Trace Complete
```

The above is the terminal output with B moved to (0, 50), and we can see that the chosen route is the same as before: A→C→E→F. Also, when I swapped the position of B and C, the route was still the same, even though this time C is at (25, 25). There are alternative paths with the same number of hops (A→B→E→F, A→D→E→F) but the simulator keeps choosing the same one. From these observations I concluded that the simulator doesn't choose a path by the node's position, but instead each node assigns an order of priority on the other nodes within its link coverage and prioritise the routes that go through nodes of high priority. In the above case we can assume that Node A has chosen Node C as the highest priority node and thus chooses the path with C regardless of C's position.

[3.2]

(3.2.1)

```
Starting simulation for 100 s ...
Traceroute to 10.0.0.5, 30 hops Max, 56 bytes of data.
1 10.0.0.4 261 ms 0 ms 1 ms
2 10.0.0.5 4 ms 1 ms 1 ms

Trace Complete
```

Above is the terminal output for the traceroute from C to E. There were two possible choices (C→B→A→E and C→D→E), and the chosen route was C→D→E. The difference between the paths C→B→A→E and C→D→E is that the former has 3 hops whereas the latter has 2 hops. On the other hand the former path has a total distance of 90m, which is shorter than that of the latter path: 100m.

(3.2.2)

From simulation results in (3.2.1) we can conclude that NS3 prioritizes the path with the smallest number of hops. And from the results in (3.1) we can say that when there are multiple choices of paths with the least number of hops, then the simulator tends to choose the path that contains nodes with high priorities. (This node priority could be derived by some algorithm in NS3 or randomly choose a node in the link coverage.)

(3.2.3) Path A: 1000 hops of 50m / Path B: 1001 hops of 30m

#3.2.3.1

The total distance of path A (50,000m) is much higher than path B (300,30m). However from the simulation result in (3.2.1) we concluded that the simulator prioritizes the path with the least number of hops. Therefore we can predict that the simulator will choose path A, since it has less number of hops than path B.

#3.2.3.2

A path with shorter total distance (path B) takes less time for the nodes to transfer data between each other. In contrast, a path with less numbers of hops (path A) takes less time for the intermediate nodes to receive data, process it, and then send it to the next node. I think the path with less number of hops is better because in a real ad hoc network could have different levels of efficiency so a large number of intermediate nodes increases the possibility of an error within the path. Also in mobile networks, having less number of nodes could decrease the possibility of an intermediate node going out of the link coverage.