Dynamic Mobility

The aim of the following simulations was to set the trajectory of a mobile node dynamically using data provided by another node. The data is of the form [speed (mps),heading (degrees)]. Here Node B follows a lawnmower pattern and whenever another comes within its range of contact , it receives the data from Node B and starts following the specified trajectory. Initially Node A is static, but the mobility of node A is set to true . That is, it is a mobile node but not moving initially.

Code Explanation:

In the dynamic-mobility-sim script we first initialize and describe the parameters of node A, B

```
simulate T, {
  def n1 = node('0', address: 6, location: [-2.km, 0, -1.km], mobility:
true, stack: { container ->
           container.add 'ping', new dynamic_mobility_node_B()
         })
 def n = node('AUV-3', address: 100, location: [0.km, 0.km, -1.km],
mobility: true, stack: { container ->
           container.add 'ping', new dynamic_mobility_node_A()
         })
```

In the dynamic-mobility-A script we try to ping node B if it is in range :

```
 add new TickerBehavior(5000, {



             phy << new ClearReq()
             println "Sending data to : " + y
             phy << new DatagramReq(to:y, data : "If near send data" ,
protocol:PING_PROTOCOL)
             println "n0 ${node.location}"
         })
```

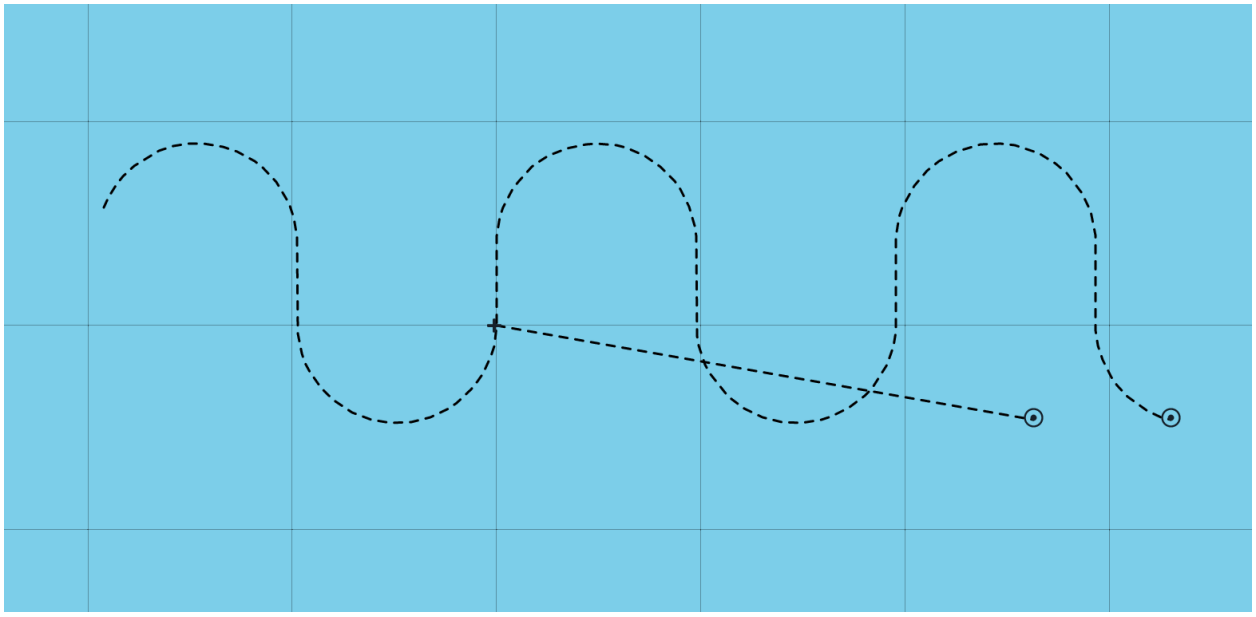
Then in dynamic-mobility-B script we check if we received the message from A , and then send the data in the form [speed,heading] (here speed ,heading was taken to be [20,100])

```
if (msg instanceof DatagramNtf && msg.protocol == PING_PROTOCOL ){

        String data = new String(msg.data)
        println "Data received as from 100: " + data
        if(node.address==6)
        {
        phy << new ClearReq()
        phy << new DatagramReq(to: 100 ,data :[20,100],
protocol:PING_PROTOCOL)
        }


   }
```

Then on Node A if we successfully receive the data array , we set the speed and heading to the specified values

```
if (msg instanceof DatagramNtf && msg.protocol == PING_PROTOCOL )
      {
        ArrayList<Integer> collectB=new ArrayList<Integer>()
        collectB.addAll(msg.data)
        int sp=msg.data[0]
        int hd=msg.data[1]


        node.heading=hd
        node.speed=sp
   }
```

The Simulation can also be visualized on the UNET sim Map . Here is a snapshot from the simulation.



(A demonstration video link will be added soon)