

Analysing previous TDMA Scripts

Author - Umang Gupta 2017A8PS0510G

Introduction: We analyse the scripts submitted in previous years and break individual components to better understand the syntax and to utilise any chunk of code that we can.

Prirish - TDMA scripts: First we'll look at the script and see its result. If you simulate the file you'll see that nothing happens.

```
Simulation time:      2100 s
TX:                  0
RX:                  0
Offered load:         0.0
Throughput:           0.0

1 simulation completed in 2.925 seconds
>
```

Nothing was sent and nothing was received, so we'll look at CMD:

```
Simulator IDE: http://localhost:8080/
←[31mSEVERE: org.arl.unet.sim.SimulationAgent/2 > Exception in agent: simulator←[0m
←[31mSEVERE: org.arl.unet.sim.SimulationAgent/5 > Exception in agent: simulator←[0m
←[31mSEVERE: org.arl.unet.sim.SimulationAgent/3 > Exception in agent: simulator←[0m
←[31mSEVERE: org.arl.unet.sim.SimulationAgent/1 > Exception in agent: simulator←[0m
←[31mSEVERE: org.arl.unet.sim.SimulationAgent/4 > Exception in agent: simulator←[0m
```

Again there seems to be some error. "Exception in Agent" -- doesn't really clarify anything. So we'll break down the script:

```
////////////////////////////////////////
// simulation details
def addressList = new ArrayList<Integer>()
for(int i = 0;i<nodes.size();i++)
{
    addressList.add((i+1))
}
```

They create this addressList in the script that is never used again. Reason for creating this is unclear.

```
nodes.each { n1 ->
    nodes.each { n2 ->
        if (n1 < n2) {
            n++
            sum += distance(nodeLocation[n1], nodeLocation[n2])
        }
        propagationDelay[n1-1][n2-1] = (int)(distance(nodeLocation[n1],nodeLocation[n2]) * 1000 / channel.soundSpeed + 0.5)
    }
}
```

This code shows a propagation delay matrix that was generated to calculate the difference in the nodes but again was never used again in the script and the reason for this is unclear.

```
// simulate schedule

simulate time, {

  def node_list = []

  nodes.each { myAddr ->
    node_list << node("${myAddr}", address: myAddr, location: nodeLocation[myAddr])
  }

  node_list.eachWithIndex { n1, i ->
    n1.startup = {
      def phy = agentForService PHYSICAL
      phy[1].frameLength = phy[0].frameLength
      phy[1].dataRate = phy[0].dataRate
      add new TickerBehavior(1000*slot, {
        def node = agentForService(Services.NODE_INFO)
        def slen = schedule[i].size()
        def s = schedule[i][(tickCount-1)%slen]
        if (s) phy << new TxFrameReq(to: s, type: Physical.DATA)
        if(s) println "${node.Address} sent data to ${s}"
      })
    }
  }
}

////////// THIS tick count is mentioned only once //////////
```

If you print `phy[1].frameLength` -- this will return null. Also we don't use these parameters again. We use the schedule matrix for creating slots, we see there is a "slen" variable that is simply an integer that equates to the size of 1 slot of the array. We can simply change it with the number 4 here.

Finally the variable `tickCount` was neither mentioned before and not mentioned after. This is the main reason for getting an error according to me; of course there could simply be an error in the scripting logic.

Also one closing remark in analysing this script; `if(s)` basically means that if the index in the schedule matrix is 1 we'll get data sent to node 1. Now without any "ClearReq()" it is unclear even if the script works, how successful would the script be.

Analysing Aditya TDMA simulation scripts

Good thing about this script is that there were no errors to debug but bad was that there was no rx or received messages while there was some tx or transmitted messages.

```
=====
TX Count      RX Count      Loss %
-----
      2          0        100.0

15 simulations completed in 28.133 seconds
>
```

```

//add new WakerBehavior(400000, {
simulate T,{

  // def ClusterAgen = new Test_Agent()
  // node '1', address: 1, location: [0.km, 0.km, 0], stack : {container -> container.add 'Cluster',ClusterAgen }
  //node '2', address: 2, location: [0, 0.km, 0], stack : {container -> container.add 'Cluster',ClusterAgen }

  node 'D', address: 30, location: [180.m, 0, -1000.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

  node 'E', address: 32, location: [220.m, 0, -1000.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

  node 'C', address: 31, location: [200.m, 0, -1000.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

  node 'G', address: 20, location: [180.m, 0, -500.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

  node 'F', address: 22, location: [220.m, 0, -500.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

  node 'B', address: 21, location: [200.m, 0, -500.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

  node 'A', abcd: 10, address: 10, location: [200.m, 0.m, 0.m], stack: { container ->
    container.add 'ping', new Test1_Agent()

  }

}

//println "Channel Sound Speed: 1500 m/s Delay: 15.33 s"

```

So we create custom nodes at custom locations and with specified addresses and use the container to add “ping” module from Test1_Agent(), so since there is nothing else in the script that does anything so we’ll look more into the Test1_Agent().

```

if(addr==30)
{
  add new WakerBehavior(0000, {
    phy[1].powerLevel = -48.dB;    // must be non- positive
    phy << new ClearReq()
    phy << new DatagramReq(to: 31 ,data : neighbouraddr , protocol:cluster_protocol)
    println "data sent by 30 to 31"
  })
}

if(addr==32)
{
  add new WakerBehavior(90, {
    phy[1].powerLevel = -48.dB;    // must be non- positive
    phy << new ClearReq()
    phy << new DatagramReq(to: 31 ,data : neighbouraddr1 , protocol:cluster_protocol)
    println "data sent by 32 to 31"
  })
}

if(addr==31)
{
  add new WakerBehavior(180, {
    phy[1].powerLevel = -28.dB;    // must be non- positive
    // println "data sent by 31 to 21 kkkkkkkkkkkkyyyyyy"
    println "${collectC} in startup in addr: 31"
    phy << new ClearReq()
    phy << new DatagramReq(to: 21 ,data : collectC , protocol:cluster_protocol)
    println "data sent by 31 to 21"
  })
}

```

This is the on startup behaviour from the script and what it does is checks the address and schedules a datagram to a specific node, however the first number in the WakerBehaviour makes very little sense. This is because in most other simulation scripts we'd have a rate or some sort of float variable that schedules the node messages. However there isn't a way to properly check which portion of the script works or if there is some logical error in this.

If we look at the process message module:

```

void processMessage(Message msg) {
    if(addr == 31){
        //add new WakerBehavior(260,{
        println "message processed in address:"+addr
    }
    if (msg instanceof DatagramNtf && msg.protocol == cluster_protocol && ( msg.from == 30 || msg.from == 32) )    //notification recieved
    {
        println "DATA Received by ${addr}, DATA VALUE is ${msg.data} "

        //temp = msg.data[0]
        //println temp
        collectC.addAll(msg.data)
        println "now total value is ${collectC}"
        //check++
        //if(check==2){
        //    //println "khaskkjhsdkaj"
        //    startup()
        //}

    }
    //})
}

if(addr == 21){
    //add new WakerBehavior(500,{
    println "message processed in address:"+addr
}
if (msg instanceof DatagramNtf && msg.protocol == cluster_protocol && ( msg.from == 20 || msg.from == 22 || msg.from == 31) )    //notification recieved
{
    println "DATA Received by ${addr}, DATA VALUE is ${msg.data}"
    //temp = msg.data[0]
    //println temp
    collectB.addAll(msg.data)
    println "now total value is ${collectB}"
    checkk++
    //if(checkk==3){
    //    startup()
    //}
}
//})
}

```

The process message script checks the datagram and sees msg.from parameter but we still have no idea if this is working.

So I'll write a custom script to see if the Test1_agent() has any decipherable errors. This is strictly because there are a lot of print statements in the test agent script and nothing shows up in the simulation.

```

ArrayList<Integer> neighbouraddr=new ArrayList<Integer>()
ArrayList<Integer> neighbouraddr1=new ArrayList<Integer>()
ArrayList<Integer> collectC=new ArrayList<Integer>()
ArrayList<Integer> collectB=new ArrayList<Integer>()
ArrayList<Integer> neighbouraddr3=new ArrayList<Integer>()
ArrayList<Integer> neighbouraddr4=new ArrayList<Integer>()

def datapacket = PDU.withFormat {
    length(16)                // 16 byte PDU {protocol data unit}
    uint8('Type')             //datapacket of 16 bytes
    uint8('CHaddr')
    //uint8('CHaddr')
    //uint8('CHaddr')
    padding(0xff)             // padded with 0xff to make 16 bytes
}

```

They create data packets in the test agent script that were not used again maybe because simply using datagram is easier; nevertheless they are a part of the script and can be skipped. Also the declared arrays collectC and collectB are mentioned later in the script but are never

assigned any value and i believe are the main reason we have simulation errors in this particular script. (Null exceptions while compilation)

Another **big** issue is the unavailability of cluster_protocol.groovy file. It's not there in any of the folders and so we can't say what exactly it does. However if we go by the name it's the clustering protocol that will implement some sort of arrangement or topology for the nodes. In our simulation scripts we'll remove this and work with waker behaviour for now to see if that portion works.

We fixed some problems with the previous scripts, added ClearReq() and there were no errors but there is still zero transmission.

```
data sent by 20 to 21
data sent by 30 to 31
data sent by 32 to 31
data sent by 22 to 21
data sent by 31 to 21

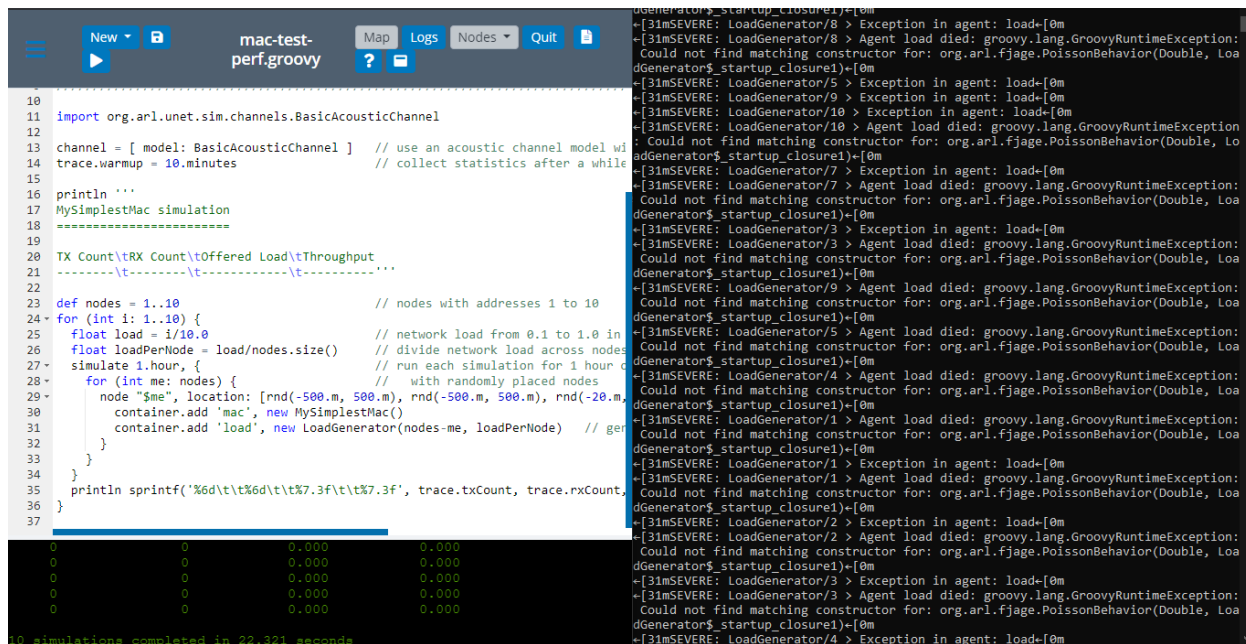
1 simulation completed in 2.410 seconds
>
```

```
1,      20,      [180, 0, -500]
2,      21,      [200, 0, -500]
3,      22,      [220, 0, -500]
4,      30,      [180, 0, -1000]
5,      31,      [200, 0, -1000]
6,      32,      [220, 0, -1000]
7,      10,      [200, 0, 0]

0          0          0.000          0.000
```

The trace count is zero. The script for this is uploaded as Test1_Agent_Sim.groovy. Something to ask about is what are the random numbers in the beginning of the waker behaviour.

Looking into MAC perf script



```
10 import org.arl.unet.sim.channels.BasicAcousticChannel
11
12 channel = [ Model: BasicAcousticChannel ] // use an acoustic channel model with
13 trace.warmup = 10.minutes // collect statistics after a while
14
15
16 println '''
17 MySimpleMac simulation
18 =====
19
20 TX Count\tRX Count\tOffered Load\tThroughput
21 -----\t-----\t-----\t-----
22
23 def nodes = 1..10 // nodes with addresses 1 to 10
24 for (int i: 1..10) {
25     float load = i/10.0 // network load from 0.1 to 1.0 in
26     float loadPerNode = load/nodes.size() // divide network load across nodes
27     simulate 1.hour, { // run each simulation for 1 hour
28         for (int me: nodes) { // with randomly placed nodes
29             node "$me", location: [rnd(-500.m, 500.m), rnd(-500.m, 500.m), rnd(-20.m, 20.m)]
30             container.add 'mac', new MySimpleMac()
31             container.add 'load', new LoadGenerator(nodes-me, loadPerNode) // generate
32         }
33     }
34 }
35 println sprintf('%6d\t%6d\t%7.3f\t%7.3f', trace.txCount, trace.rxCount,
36 )
37
38 0 0 0.000 0.000
39 0 0 0.000 0.000
40 0 0 0.000 0.000
41 0 0 0.000 0.000
42 0 0 0.000 0.000
43
44 10 simulations completed in 22.321 seconds
```

```
+ [31mSEVERE: LoadGenerator/8 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/8 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/5 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/5 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/10 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/10 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/7 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/7 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/3 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/3 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/9 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/9 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/5 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/4 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/4 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/1 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/1 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/2 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/2 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/3 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/3 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
+ [31mSEVERE: LoadGenerator/4 > Exception in agent: load-[0m
```

Running the script we see no result in the simulation window but on the cmd window we see the exception in agent and because we have 10 nodes we see a repeated error. This shows a problem in fjage script that is not accessible for the user and removing fjage libraries altogether gives you an error during simulation.

These scripts are layered so but for the moment we see there is some issue with the LoadGenerator script. Also the MySimpleMac() is the script from samples in Unet and not from the Github file. Here is an error we get from simulating the load generator file:

```
+ [31mSEVERE: LoadGenerator/10 > Exception in agent: load-[0m
+ [31mSEVERE: LoadGenerator/10 > Agent load died: groovy.lang.GroovyRuntimeException: Could not find matching constructor for: org.arl.fjage.PoissonBehavior(Double, LoadGenerator$_startup_closure1)-[0m
```

There is no clear way to resolve this that I can see. Which fjage files can be included for this script to work is a question to resolve for the moment. Another problem is why is the pre-written module raising an exception in a script. It should ideally be maintained by the unet developers and thus there is a higher chance of there being some sort of a logical problem with the script.

Closing Remarks:

Although there were some issues with the script I'd recommend still going through the scripts and through these errors so you can only put in the working code chunks into your simulation.

Going through these errors will also get you an idea of how the simulation works. I would also recommend putting some `println` statements because that gives you the idea of what takes place when you're running your simulation scripts.