**Default Setting**

|  |  |
| --- | --- |
| **Line in default\_settings.txt** | **Description** |
| Scenario.endTime = 43200 | Total simulation time in seconds |
| Group.router = EpidemicRouter | Use Epidemic Router as routing algorithm, you can also set it to NewRouter |
| Group.bufferSize = 20M | Boat buffer limit, buffer of PBS is unlimited |
| Group.waitTime = 0, 0 | Random a number in [0, 0] -> fix waitTime = 0, waitTime is the time that needs to wait for finding next available path |
| Group.msgTtl = 300 | Message Time To Live in minutes |
| Group1.groupID = Boat1\_ | Group ID from 1-13 are boats, 14-22 are PBS. But boats ID are actually from 0-12 in Java. we fix message destination “PBS\_Belem13” |
| Group1.routeFile = data/CoDPON/RT\_SRE\_BLM\_CUR.wkt | Boat routing path file |
| Boat1.DP = PBS\_SRE20, PBS\_Belem13, PBS\_Curralinho15 | Boat displacement plan containing PBS that a boat will visit |
| Events1.interval = 864 | Message create interval (864 = 43200/50) |
| Events1.size = 75k | Message partition size |
| Events1.hosts = 13, 21 | PBS ID in Java |

**Message Partitioning**

We need to compare 2 message transferring scheme.

1. Transfer a whole message.

A single message is 10MB (an ultrasound file is 10MB), we generate 50 messages every day in 12 hours.

1. Transfer partitions of a message

We need to partition the message into small packets, the size of a packet is 75Kb, which is the biggest size which any two boat can transfer in the shortest connection. In this way, one message is partitioned into 133 packets.

I changed the MessageCreateEvent (src->input-> MessageCreateEvent.java) so that the message can be generated as partitioned packets instead of a whole file. As you can see in the MessageCreateEvent:

packetNumK = 133

generateNumN = 133

packetNumK is the number of packets that one file is partitioned into. generateNumN is the number of packets that are actually generated to transfer.(In this case they are the same, but I created them initially to simulate Fountain Codes, in Fountain Codes, the generateNumN is 10% more than packetNumK).

In the code I sent to you, the setting is to use partitioned packets, if you want to transfer whole message, you can change the generateNumN to be 1. And you need to change in the default\_setting.txt where Events1.size = 75k. You can change it to 10M instead since 75k is the size of small packet, and 10M is the size of one single file.

We will compare the performance of these two message scheme using two routing algorithm.

**Routing Algorithm**

In our experiments, we basically compare two kinds of routing: EpidemicRouting and OpportunisticRouting. In EpidemicRouting, it's flooding whenver it's possible. The nodes, both boats and PBS(Peer Base Station) will forward their messages to any node that come across(this means when two nodes are close enough, they form a connection and start transferring messages) until the message reaches its destination. In OpportunisticRouting, a node will check every node that come across to see if the other node will go to the message’s destination. It will forward its message to the other node only if the node will go to the message’s destination. Also, after each transferring, the forwarding node will delete the original message that had already been transferred so that it won’t waste the boat’s buffer.

You can find both router in directory: one->src->routing, they are “EpidemicRouter.java” and “NewRouter.java”. The epidemic one is already provided in the simulator and I created the NewRouter using opportunisitc routing, and I modified the ActiveRouter accordingly to call the NewRouter.

**Problem we have now**

In theory, the results of partitioning message using either epidemic router or new router should be strictly better than transferring single messages. However, in the excel files I sent you, you can see this is not the case for epidemic routing. Since there is one message (M14) that successfully delivered in non-partition, but M14 only delivers 1 packet in partitioning method. Although the overall performance for partitioning is better in all cases, we still need to figure it out why the above case happens. I think one possible reason is that there are still some random factor in the code because originally in the simulator, the message destination, waitTime, message retrieve order are all random generated at run time so that each simulation will have different input.