LAMPIRAN

1. *Default Setting*

```
#
# Default settings for the simulation
#

## Scenario settings
Scenario.name = default_scenario
Scenario.simulateConnections = true
Scenario.updateInterval = 0.1
# 43200s == 12h
Scenario.endTime = 10000000
Scenario.endTime = 86400

## Interface-specific settings:
# type : which interface class the interface belongs to
# For different types, the sub-parameters are interface-
specific
# For SimpleBroadcastInterface, the parameters are:
# transmitSpeed : transmit speed of the interface (bytes
per second)
# transmitRange : range of the interface (meters)

# "Bluetooth" interface for all nodes
btInterface.type = SimpleBroadcastInterface
# Transmit speed of 2 Mbps = 250kBps
btInterface.transmitSpeed = 250k
btInterface.transmitRange = 10

# High speed, long range, interface for group 4
highspeedInterface.type = SimpleBroadcastInterface
highspeedInterface.transmitSpeed = 10M
highspeedInterface.transmitRange = 10

# Define 6 different node groups
Scenario.nrofHostGroups = 1

## Group-specific settings:
# groupID : Group's identifier. Used as the prefix of host
names
# nrofHosts: number of hosts in the group
# movementModel: movement model of the hosts (valid class
name from movement package)
```

```
# waitTime: minimum and maximum wait times (seconds) after
reaching destination
# speed: minimum and maximum speeds (m/s) when moving on a
path
# bufferSize: size of the message buffer (bytes)
# router: router used to route messages (valid class name
from routing package)
# activeTimes: Time intervals when the nodes in the group
are active (start1, end1, start2, end2, ...)
# msgTtl : TTL (minutes) of the messages created by this
host group, default=infinite

## Group and movement model specific settings
# pois: Points Of Interest indexes and probabilities
(poiIndex1, poiProb1, poiIndex2, poiProb2, ... )
#       for ShortestPathMapBasedMovement
# okMaps : which map nodes are OK for the group (map file
indexes), default=all
#       for all MapBasedMovent models
# routeFile: route's file path - for MapRouteMovement
# routeType: route's type - for MapRouteMovement


# Common settings for all groups
# Group.movementModel = ShortestPathMapBasedMovement
Group.movementModel = MapBasedMovement
Group.router = SprayAndWaitRouter
# Group.router = EpidemicRouter
Group.bufferSize = 5M
Group.waitTime = 0, 120
# All nodes have the bluetooth interface
Group.nrofInterfaces = 1
Group.interface1 = btInterface
# Walking speeds
Group.speed = 1.5, 2.5
# Message TTL of 300 minutes (5 hours)
Group.msgTtl = 120

Group.nrofHosts = 50

# group1 (pedestrians) specific settings
Group1.groupID = p

# group2 specific settings
#Group2.groupID = c
# cars can drive only on roads
```

```
#Group2.okMaps = 1
# 10-50 km/h
#Group2.speed = 2.7, 13.9

# another group of pedestrians
#Group3.groupID = w

# The Tram groups
#Group4.groupID = t
#Group4.bufferSize = 50M
#Group4.movementModel = MapRouteMovement
#Group4.routeFile = data/tram3.wkt
#Group4.routeType = 1
#Group4.waitTime = 10, 30
#Group4.speed = 7, 10
#Group4.nrofHosts = 2
#Group4.nrofInterfaces = 2
#Group4.interface1 = btInterface
#Group4.interface2 = highspeedInterface

#Group5.groupID = t
#Group5.bufferSize = 50M
#Group5.movementModel = MapRouteMovement
#Group5.routeFile = data/tram4.wkt
#Group5.routeType = 2
#Group5.waitTime = 10, 30
#Group5.speed = 7, 10
#Group5.nrofHosts = 2

#Group6.groupID = t
#Group6.bufferSize = 50M
#Group6.movementModel = MapRouteMovement
#Group6.routeFile = data/tram10.wkt
#Group6.routeType = 2
#Group6.waitTime = 10, 30
#Group6.speed = 7, 10
#Group6.nrofHosts = 2


## Message creation parameters
# How many event generators
Events.nrof = 1
# Class of the first event generator
Events1.class = MessageEventGenerator
# (following settings are specific for the
MessageEventGenerator class)
```

```
# Creation interval in seconds (one new message every 25 to
35 seconds)
# 1 menit = 60
Events1.interval = 600, 900
# Message sizes (500kB - 1MB)
Events1.size = 10k
# range of message source/destination addresses
Events1.hosts = 1, 1
Events1.tohosts = 24, 24
# Message ID prefix
Events1.prefix = M


## Movement model settings
# seed for movement models' pseudo random number generator
(default = 0)
MovementModel.rngSeed = 1
# World's size for Movement Models without implicit size
(width, height; meters)
MovementModel.worldSize = 4500, 3400
# How long time to move hosts in the world before real
simulation
MovementModel.warmup = 1000

## Map based movement -movement model specific settings
MapBasedMovement.nrofMapFiles = 4

MapBasedMovement.mapFile1 = data/roads.wkt
MapBasedMovement.mapFile2 = data/main_roads.wkt
MapBasedMovement.mapFile3 = data/pedestrian_paths.wkt
MapBasedMovement.mapFile4 = data/shops.wkt

## Reports - all report names have to be valid report
classes

# how many reports to load
Report.nrofReports = 2
# length of the warm up period (simulated seconds)
Report.warmup = 0
# default directory of reports (can be overridden per
Report with output setting)
Report.reportDir = reports/baru
# Report.reportDir = reports/shortestpath/TTL
# Report.reportDir = reports/Lcopies
# Report classes to load
Report.report1 = MessageStatsReport
```

```
Report.report2 = BufferOccupancyReport
#Report.report3 = MessageGraphvizReport

## Default settings for some routers settings
#ProphetRouter.secondsInTimeUnit = 30
SprayAndWaitRouter.nrofCopies = 6
SprayAndWaitRouter.binaryMode = true

## Optimization settings -- these affect the speed of the
simulation
## see World class for details.
Optimization.cellSizeMult = 5
Optimization.randomizeUpdateOrder = true


## GUI settings

# GUI underlay image settings
GUI.UnderlayImage.fileName = data/helsinki_underlay.png
# Image offset in pixels (x, y)
GUI.UnderlayImage.offset = 64, 20
# Scaling factor for the image
GUI.UnderlayImage.scale = 4.75
# Image rotation (radians)
GUI.UnderlayImage.rotate = -0.015

# how many events to show in the log panel (default = 30)
GUI.EventLogPanel.nrofEvents = 100
# Regular Expression log filter (see Pattern-class from the
Java API for RE-matching details)
#GUI.EventLogPanel.REfilter = .*p[1-9]<->p[1-9]$
```

2. **Penambahan *Density***

```
Scenario.name = compare-%%Group.movementModel%%-
%%Group.router%%-%%Group.nrofHosts%%-
%%MovementModel.rngSeed%%


# Group.movementModel = ShortestPathMapBasedMovement

Group.movementModel = MapBasedMovement

Group.router = [EpidemicRouter; SprayAndWaitRouter;]

Group.nrofHosts = [25; 50; 75; 100; 125;]
```

```
MovementModel.rngSeed = [1; 2; 3; 4; 5;]
Report.nrofReports = 2
Report.report1 = BufferOccupancyReport
Report.report2 = MessageStatsReport
Report.reportDir = reports/baru/density
```

3. **Penambahan *Buffer Size***

```
Scenario.name = compare-%%Group.movementModel%%-
%%Group.router%%-%%Group.bufferSize%%-
%%MovementModel.rngSeed%%

Group.movementModel = MapBasedMovement
# Group.movementModel = ShortestPathMapBasedMovement
Group.router = [EpidemicRouter; SprayAndWaitRouter;]
Group.bufferSize = [5M; 10M; 15M; 20M; 25M;]

MovementModel.rngSeed = [1; 2; 3; 4; 5;]
Report.nrofReports = 2
Report.report1 = BufferOccupancyReport
Report.report2 = MessageStatsReport
Report.reportDir = reports/baru/buffer
```

4. **Penambahan TTL**

```
Scenario.name = compare-%%Group.movementModel%%-
%%Group.router%%-%%Group.msgTtl%%-%%MovementModel.rngSeed%%

Group.movementModel = MapBasedMovement
# Group.movementModel = ShortestPathMapBasedMovement
Group.router = [EpidemicRouter; SprayAndWaitRouter;]
Group.msgTtl = [60; 120; 180; 240; 300;]
```

```
MovementModel.rngSeed = [1; 2; 3; 4; 5;]

Report.nrofReports = 2

Report.report1 = BufferOccupancyReport

Report.report2 = MessageStatsReport

Report.reportDir = reports/baru/TTL
```

5. **Penambahan L*copies***

```
Scenario.name = compare-%%Group.movementModel%%-
%%SprayAndWaitRouter.nrofCopies%%-
%%SprayAndWaitRouter.binaryMode%%-%%MovementModel.rngSeed%%


Group.movementModel = [MapBasedMovement;
ShortestPathMapBasedMovement;]

Group.router = SprayAndWaitRouter

SprayAndWaitRouter.nrofCopies = [5; 7; 9; 11; 13;]

SprayAndWaitRouter.binaryMode = [true]


MovementModel.rngSeed = [1; 2; 3; 4; 5;]

Report.nrofReports = 2

Report.report1 = BufferOccupancyReport

Report.report2 = MessageStatsReport

Report.reportDir = reports/baru/Lcopies
```

6. **Penambahan L*copies* dan *Density***

```
Scenario.name = compare-%%Group.movementModel%%-
%%Group.nrofHosts%%-%%SprayAndWaitRouter.nrofCopies%%-
%%SprayAndWaitRouter.binaryMode%%-%%MovementModel.rngSeed%%


Group.router = SprayAndWaitRouter

Group.movementModel = [MapBasedMovement;
ShortestPathMapBasedMovement;]

Group.nrofHosts = [25; 50; 75; 100; 125;]
```

```
SprayAndWaitRouter.nrofCopies = [5; 7; 9; 11; 13;]

SprayAndWaitRouter.binaryMode = [true]


MovementModel.rngSeed = [1; 2; 3; 4; 5;]

Report.nrofReports = 2

Report.report1 = BufferOccupancyReport

Report.report2 = MessageStatsReport

Report.reportDir = reports/baru/Lcopies_Density
```

7. ***Listing Program Average Buffer Occupancy***

```
/*
 *
 *
 */
package report;

/**
 * Records the average buffer occupancy and its variance
with format:
 * <p>
 * <Simulation time> <average buffer occupancy % [0..100]>
<variance>
 * </p>
 *
 *
 */
import java.util.*;
//import java.util.List;
//import java.util.Map;

import core.DTNHost;
import core.Settings;
import core.SimClock;
import core.UpdateListener;

public class BufferOccupancyReport extends Report
implements UpdateListener {

/**
 * Record occupancy every nth second -setting id
({@value}).
 * Defines the interval how often (seconds) a new snapshot
of buffer
```

```
 * occupancy is taken previous:5
 */
public static final String BUFFER_REPORT_INTERVAL =
"occupancyInterval";
/** Default value for the snapshot interval */
public static final int DEFAULT_BUFFER_REPORT_INTERVAL =
3600;

private double lastRecord = Double.MIN_VALUE;
private int interval;

private Map<DTNHost, Double> bufferCounts = new
HashMap<DTNHost, Double>();
private int updateCounter = 0;  //new added


public BufferOccupancyReport() {
      super();

      Settings settings = getSettings();
      if (settings.contains(BUFFER_REPORT_INTERVAL)) {
            interval =
settings.getInt(BUFFER_REPORT_INTERVAL);
      } else {
            interval = -1; /* not found; use default */
      }

      if (interval < 0) { /* not found or invalid value ->
use default */
            interval = DEFAULT_BUFFER_REPORT_INTERVAL;
      }
}

public void updated(List<DTNHost> hosts) {
      if (isWarmup()) {
            return;
      }

      if (SimClock.getTime() - lastRecord >= interval) {
            lastRecord = SimClock.getTime();
            printLine(hosts);
            updateCounter++; // new added
      }
            /**
            for (DTNHost ho : hosts ) {
                  double temp = ho.getBufferOccupancy();
                  temp = (temp<=100.0)?(temp):(100.0);
                  if
(bufferCounts.containsKey(ho.getAddress()))
```

```
                        bufferCounts.put(ho.getAddress(),
(bufferCounts.get(ho.getAddress()+temp))/2);
                  else
                  bufferCounts.put(ho.getAddress(), temp);
            }
            }
      */
}

/**
 * Prints a snapshot of the average buffer occupancy
 * @param hosts The list of hosts in the simulation
 */

private void printLine(List<DTNHost> hosts) {
      /**
      double bufferOccupancy = 0.0;
      double bo2 = 0.0;
      for (DTNHost h : hosts) {
            double tmp = h.getBufferOccupancy();
            tmp = (tmp<=100.0)?(tmp):(100.0);
            bufferOccupancy += tmp;
            bo2 += (tmp*tmp)/100.0;
      }

      double E_X = bufferOccupancy / hosts.size();
      double Var_X = bo2 / hosts.size() - (E_X*E_X)/100.0;

      String output = format(SimClock.getTime()) + " " +
format(E_X) + " " +
            format(Var_X);
      write(output);
      */
      for (DTNHost h : hosts ) {
            double temp = h.getBufferOccupancy();
            temp = (temp<=100.0)?(temp):(100.0);
            if (bufferCounts.containsKey(h)){
                  //bufferCounts.put(h,
(bufferCounts.get(h)+temp)/2); seems WRONG

                  bufferCounts.put(h,
bufferCounts.get(h)+temp);
                  //write (""+ bufferCounts.get(h));
            }
            else {
            bufferCounts.put(h, temp);
            //write (""+ bufferCounts.get(h));
            }
      }
```

```
}


@Override
public void done()
{

      for (Map.Entry<DTNHost, Double> entry :
bufferCounts.entrySet()) {

            DTNHost a = entry.getKey();
            Integer b = a.getAddress();
            Double avgBuffer =
entry.getValue()/updateCounter;
            write("" + b + ' ' + avgBuffer);

            //write("" + b + ' ' + entry.getValue());
      }
      super.done();
}


}
```