BonnMotion

a Mobility Scenario Generation and Analysis Tool

Documentation

Version: June 10, 2009

Contents

1 Legal notice

2 Contact information

eMail:
Nils Aschenbruck ⟨tacnet@cs.uni-bonn.de⟩
URL:
`http://bonnmotion.cs.uni-bonn.de/`
Postal address:
Nils Aschenbruck
Institute of Computer Science IV, University of Bonn
Roemerstr. 164, 53117 Bonn, Germany

3 Introduction

BonnMotion is a Java software which creates and analyses mobility scenarios. It is developed within the Communication Systems group at the Institute of Computer Science IV of the University of Bonn, Germany, where it serves as a tool for the investigation of mobile ad hoc network characteristics. The scenarios can also be exported for the network simulators ns-2, GloMoSim/QualNet, COOJA, and MiXiM.

4 Installation

To use this software, you need to have a JDK or JRE installed. It has been compiled and tested with Java v1.5.0_18.

During the installation, a few other shell scripts / batch files are created in the "bin" folder, which you can move and rename as you like:

- "bm" is a wrapper that starts the BonnMotion application. Starting it without command line parameters prints a detailed help message.
- "compile" re-compiles the sources. By default, only applies to files that were changed after the last compilation. To re-compile all sources, use "compile all".
- "makedoc" uses javadoc to create a source code documentation.

This distribution includes the compiled class files so if you do not want to modify the code, there is no need for you to run the compile or makedoc scripts.

4.1 Installation on UNIX operating systems

If you are using a UNIX platform, simply unpack the archive and run the "install" script. You will then be asked for the location of your Java binary path, i.e. the directory containing the Java interpreter and possibly the Java compiler and the javadoc utility.

NOTE: We have not yet run our shell scripts on other platforms than Linux or Cygwin. We would be happy if you informed us about changes necessary to run them under other operating systems so we can incorporate them into our distribution.

4.2 Installation on Microsoft Windows operating systems

NOTE: The batch files for MS Windows have not yet been thoroughly tested.

If you are using Microsoft Windows, please edit the two variables JAVAPATH and BONNMOTION in the "install.bat" and then execute it.

5 Running

All applications described below are started via the "bm" wrapper script. The syntax is as follows:

bm ⟨parameters⟩ ⟨application⟩ ⟨application parameters⟩

The application can be a mobility model or e.g. the Statistics application used to analyse scenario characteristics.

Starting the script without command line parameters prints further help. Usage examples are given in the sections below.

6 Scenario generation

Currently, there are several mobility models available, which are introduced later on with some comments on implementation details. For further details on the models themselves, please refer to the literature. Various synthetic models were proposed during last decade. There have been several general surveys [Camp et al. 2002, Bettstetter 2001, Bai / Helmy 2004, Musolesi / Mascolo 2008] as well as some specific ones for vehicular models [Hoogendoorn / Bovy 2001].

There are two possibilities to feed input parameters into the scenario generation: The first is to enter the parameters on the command line, and the second is to have a file containing the parameters. These two methods can also be combined; in this case, the command line parameters override those given in the input file.

The scenario generator writes all parameters used to create a certain scenario to a file. In this way, settings are saved and particular scenario parameters can be varied without the need to re-enter all other parameters.

Important parameters used with all models are the following: The node number is set with -n, the scenario duration (in seconds) with -d and the -i parameter specifies, how many additional seconds at the beginning of the scenario should be skipped. With -x and -y, the

width and height (in metres) of the simulation area are set. With -R, the random seed can be set manually.

Cutting off the initial phase is an important feature and therefore, -i has a high default value: It has been observed that with the Random Waypoint model, nodes have a higher probability of being near the center of the simulation area, while they are initially uniformly distributed over the simulation area. In our implementation of the Manhattan Grid model, all nodes start at (0,0) for simplicity.

Usage examples:

```
bm -f scenario1 RandomWaypoint -n 100 -d 900 -i 3600
```

This creates a Random Waypoint scenario with 100 nodes and a duration of 900 seconds. An initial phase of 3600 seconds is cut off.

A scenario is saved in two files: The first, with the suffix ".params", contains the complete set of parameters used for the simulation. The second, with the suffix ".movements.gz" contains the (gzipped) movement data.

Now, you can take a look at the "scenario1.params" file to see all parameters used for the simulation, and if you wanted to create a similar scenario, only with a higher mobility, you can do this with

```
bm -f scenario2 -I scenario1.params RandomWaypoint -h 5.0
```

which takes the whole parameter set from scenario1.params and overrides the maximum speed with 5 m/s.

6.1 The Random Waypoint model ("RandomWaypoint")

Our implementation includes a feature to restrict the mobiles' movements: With "-d 1", nodes move only along the x-axis, with "-d 2", nodes move either along the x- or along the y-axis (with a probability of 0.5 each) and with the default "-d 3", it is the classical Random Waypoint model as you know it.

Another feature is the "-c" switch that causes positions to be chosen from a circle that fits into the simulation area rather than from the simulation area itself.

Instead of choosing new destinations uniformly distributed from the simulation area, "attraction points" can be defined with the "-a" parameter, followed by the data characterising the attraction points. Each attraction point is defined by four floating point numbers, in this order: <x-coordinate>, <y-coordinate>, <intensity>, and <standard deviation>.

- The coordinates give the attraction point's position.
- The intensity levels weight the attraction points: A point with an intensity x times as high as another point's will also attract a node with a probability which is x times as high.
- The last parameter is the standard deviation of the Gaussian distribution with mean 0 that is used to determine the nodes' distances to the attraction point on each of the

two dimensions (i.e., the distance to the attraction point does not follow a Gaussian distribution, but a Rayleigh distribution).

The parameters for several attraction points are simply concatenated, seperated by commas. For example, to place two attraction points on the simulation area, the first at (100,100) with intensity 1 and standard deviation 20, the second at (350,200) with intensity 1.5 and standard deviation 31, use "-a 100,100,1,20,350,200,1.5,31".

6.2 The Manhattan Grid model ("ManhattanGrid")

The Manhattan Grid model is introduced in [ETSI 1998]. In this model, nodes move only on predefined paths. The arguments -u and -v set the number of blocks between the paths. As an example, "-u 3 -v 2" places the following paths on the simulation area:

```
+ - + - + - +
|   |   |   |
+ - + - + - +
|   |   |   |
+ - + - + - +
```

Our implementation contains some (reasonable) modifications of the Manhattan Grid model:

1) An additional parameter we introduce is the minimum speed of a mobile node. This is helpful because the speed of a mobile can be arbitrarily close to 0 and since the model defines that the speed is to be updated in _distance_ intervals, there can be very long periods of very slow node movement without this parameter.

2) The possibility to have nodes pause was added with help of two additional parameters: The pause probability (if a node does not change its speed, it will pause with that probability) and the maximum pause time.

Note that it is especially important to cut off an initial phase with this model, because in our implementation, all nodes start at the same position (0,0).

6.3 Gauss-Markov models

6.3.1 The original Gauss-Markov model ("OriginalGaussMarkov")

This implementation of the Gauss-Markov model follows the publication [Liang / Haas 1999]. In this implementation, the mean velocity vector mu is not specified directly; instead, the norm is specified using "-a" and a random vector with this norm is assigned to each station. Of course, a norm of 0 yields only the vector (0,0). The implementation also allows the user to specify a maximum speed. A velocity vectors with a larger norm will be multiplied with an appropriate scalar to reduce the speed to the maximum speed.

The model has been adapted to deal with scenario borders in the following way: If a station moves onto the border, its velocity vector as well as its expected velocity vector are "mirrored".

6.3.2 The Gauss-Markov model ("GaussMarkov")

This is the implementation of the Gauss-Markov model, which rather follows the description in [Camp et al. 2002], though it is not the same. The main commonalites are that for each

mobile node, two seperate values are maintained instead of one speed vector: The mobile's speed and its direction of movement. Also the default method of handling mobile nodes that move out of the simulation area is closely related to [Camp et al. 2002]: Nodes may continue to walk beyond the area boundary, which causes the next movement vector update not to be based on the prior angle, but on an angle that brings the nodes back onto the field. Therefore, the field size is automatically adapted to the node movements after scenario generation.

The main difference to [Camp et al. 2002] is that new speed and direction of movement are simply chosen from a normal distribution with a mean of the respective old value (the standard deviation is specified on the command line using -a and -s). Speed values are constrained to a certain interval that can be specified on the command line using -m and -h: If a newly chosen speed value is outside of this interval, it is changed to the closest value inside of the interval (which is either the minimum or the maximum value).

The behaviour described above can be modified with several command line switches: Using -b, the size of the simulation area is fixed and nodes simply "bounce" at the area boundaries. Using -u, the speed values outside of the valid speed interval are adapted in a way that leads to a uniform distribution of node speeds (instead of peaks around the interval boundaries).

6.4 The Reference Point Group Mobility model ("RPGM")

The implementation of this model [Hong et al. 1999] includes the possibility to have "dynamic" groups: When a node comes into the area of another group, it changes to this new group with a probability that can be set with "-c <probability>". Deactivate this feature with "-c 0". Note that when this feature is activated, "empty" groups may be moving along the simulation area and nodes coming into their areas may change their memberships to these.

6.5 Static scenarios ("Static")

By default, nodes in static scenarios are homogeneously distributed over the simulation area. There are two possibilities for non-homogeneous node distributions:

- Attraction points can be defined with the "-a" parameter; a detailed explanation of this feature is given in the "Random Waypoint" section.
- With the "-l" parameter, the simulation area can be divided into several areas with different node densities along its x-axis. Given the number n of density leves, each of the n areas will contain a fraction of approximately $2 * k/(n * (n + 1)), 1 <= k <= n$, of the nodes. (The density decreases from left to right.)

The following example shall illustrate this: Distributing 150 nodes on a 300m x 300m field with 3 density levels, approx. 75 nodes will lie within the rectangle area with the corners (0,0) and (100,300), approx. 50 nodes will lie within (100,0) and (200,300) and approx. 25 nodes will lie within (200,0) and (300,300).

6.6 Disaster Area model ("DisasterArea")

Creates scenarios according to [Aschenbruck et al. 2007].

- Tactical areas can be defined with the "-b" parameter following a list of coordinates for the area (seperated by ","). The last three values in this list are type, wanted groups and transportgroups.

- Type can be 0 for incident location, 1 for patients waiting for treatment area, 2 for casualties clearing station, 3 for technical operational command and 4 for ambulance parking point.
- For every specified incident location there must be a patients waiting for treatment area specified, for every patients waiting for treatment area a casualties clearing station and for every ampulance parking point a casualties clearing station.

7 Converting scenarios to other formats

The native format in which BonnMotion saves the movement traces is node-by-line waypoint based. This means that there is one line for each node. This line contains all the waypoints. A waypoint is a position at which the movement of a node (e.g. direction, velocity) changes. A waypoint consists of: o the simulation time in seconds at which the waypoint is reached by the node o the x and y coordinates of the position of the waypoint

7.1 ns-2

The "NSFile" application is used to generate two files that can be integrated into a TCL script to start an ns-2 simulation via the "source" command.

The file with the suffix ".ns_params" sets some variables needed to set up the simulation in an array named "val": the keys "x" and "y" refer to width and height of the simulation area, "nn" refers to the number of nodes and "duration" to the duration of the simulation.

The file with the suffix ".ns_movements" schedules the movements of the node objects that are expected to be in an array named "node_", numbered starting at 0. The simulator object is expected to be in the variable "ns_".

As a side note, the "NSFile" application places an additional margin around the simulation area, because ns-2 versions up to 2.1b9a regularly crash when nodes move at the border of the simulation area. (Actually, this has only been observed with the Manhattan Grid model up to now, but this procedure is generally carried out just to play it safe.)

Usage example:

```
bm NSFile -f scenario1
```

creates the two files "scenario1.ns_params" and "scenario1.ns_movements".

7.2 Glomosim / Qualnet

The "GlomoFile" application creates files with the suffixes ".glomo_nodes" and ".glomo_mobility", which can be used with Glomosim (2.0.3) and Qualnet (3.5.1). Use the "-q" switch for Qualnet: This causes nodes to be numbered starting at 1, not at 0.

7.3 XML

The "SPPXml" application is used to generate mobility files in XML format according to the XML schema proposed by Horst Hellbrck as a standardised mobility file format for

the research program "Schwerpunktprogramm 1140" (`http://www.tm.uka.de/forschung/SPP1140/`) of the DFG (Deutsche Forschungsgemeinschaft).

SPPXml has an additional parameter "-r" that allows to specify a uniform radio range for all nodes. If this parameter is omitted, a radio range of 250m is used. BonnMotion does not use this radio range in any way. It is however required by the XML schema.

The XML schema corresponding to the XML files generated by "SPPXml" is defined in: `http://www.i-u.de/schools/hellbrueck/ansim/xml/sppmobtrace.xsd`

The contrib directory furthermore includes a python program "sppmob2bonnmotion.py" that convertw XML mobility traces to the BonnMotion format. It has been tested with Python 2.3.

7.4 IntervalFormat

The native format implies that during the simulations for each event the current node positions have to be calculated based on the waypoints. If there are many events, this may have a negative impact on the runtime of a simulation. An alternative is to use an interval based approach. The nodes are regarded as stationary for an interval. The positions of the nodes are updated periodically after each interval by a specific position update event. By doing so, the current node positions do not have to be calculated for each event. However, the number events is increased, which may also influence the runtime of a simulation negatively. A factor that has a major impact in this context is the interval length. Smaller intervals yield higher accuracy but also more events. Overall, it is a trade-off between the number of events and the runtime per event.

Trace files in the BonnMotion's native trace format can be transformed to an interval-based format using the IntervalFormat application. The interval length can be specified using the -l option. The default value is one second. The interval trace format is an interval-by-line based. This means that there is one line for each interval of each node. A line consists of:

- the node number
- the simulation time in seconds (in intervals)
- the x and y coordinates of the position of the node for the interval

The IntervalFormat application prints the waypoints (ordered by node and time) for every interval step.

- The used interval can be specified using the -l switch.
- Using the -s switch the header can be skipped

8 Scenario analysis

8.1 The Statistics application

The "Statistics" application is used to analyse a given scenario. There are two modes of operation: The default is to calculate "overall" statistics (averaged over the simulation time) and the other mode is to calculate "progressive" statistics (values of metrics for certain points in time).

In its default mode, the application creates a file with the suffix ".stats" containing the following information:

- The *relative mobility* is independent of the transmission range and is calculated according to [Johansson et al. 1999].
- The other figures are dependent on the transmission range, which is given in the first column of every line.
- The second column gives the average node degree: To how many of the other nodes is one node connected?
- The third column gives the average number of partitions: 1 means the network is connected at all times, values larger than 1 indicate that this is not the case.
- The fourth column gives the degree of separation: How likely is it that two randomly chosen nodes are within a connected component at a randomly chosen point in time?
- The fifth column gives the average link duration; only links that go up after the simulation start and go down before the simulation end are taken into account.
- The sixth column gives the standard deviation of the previous measure.
- The seventh column gives the number of links considered in the calculation of the two previous values.
- The eighth column is like the fifth, but this time, _all_ links are counted.
- The ninth column gives the total number of links.

Alternatively to the statistics which are averaged over the whole simulation time, the devolution of certain characteristics can be calculated. See the command line help and the following examples.

Usage examples:

bm Statistics -f scenario1 -r 50,75,100

This writes the averaged statistics to "scenario1.stats" for transmission ranges of 50, 75, and 100 meters.

bm Statistics -f scenario1 -r 75 -P

This creates a file scenario1.stats_75.part, which gives the number of partitions each time it changes.

bm Statistics -f scenario1 -r 50,100 -N -M 10

This creates the files "scenario1.stats_50.nodedeg" and "scenario1.stats_100.nodedeg" which show the devolution of the node degrees. Furthermore, the files "scenario1.stats_50.mincut" and "scenario1.stats_100.mincut" show the minimal cut of the topology every 10 seconds. It is reasonable to specify such a time span for computations that cost much CPU time.

8.2 The LinkDump application

The LinkDump application prints information about every single link within a certain time span of a scenario. This information can e.g. be used for a simulator that does not directly model mobility at all.

Using the -d switch, only the link durations are printed. This offers more insight into the distribution of link durations than the averaged value calculated by the Statistics application. In this case, the -w switch prevents that wrong durations are printed out due to links that go up before simulation begin or go down after simulation end.

8.3 The Dwelltime application

The Dwelltime application creates statistics how long nodes stay in which area of the simulated region.

9 Scenario visualisation

"Visplot" is a very simple application that writes those positions to a file where a mobile changes its speed or direction. This file can simply be visualised using e.g. gnuplot.

10 Acknowledgments

The first versions of this software were designed and implemented by Michael Gerharz and Christian de Waal. The development of these versions were supported in part by the German Federal Ministry of Education and Research (BMBF) as part of the IPonAir project.

Since 2008 BonnMotion is partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053.

Further extension have been designed and implemented by Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, Tim Heinrich, Patrick Peschlow, and Matthias Schwamborn.

References

[Aschenbruck et al. 2007]
ASCHENBRUCK, Nils, GERHARDS-PADILLA, Elmar, GERHARZ, Michael, FRANK, Matthias und MARTINI, Peter: Modelling Mobility in Disaster Area Scenarios. In: *Proc. 10th ACM-IEEE Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Syste ms (MSWIM)* (2007), S. 4–12

[Bai / Helmy 2004]
BAI, Fan und HELMY, Ahmed: *A Survey of Mobility Models*. 2004. – `http://nile.usc.edu/~helmy/important/Modified-Chapter1-5-30-04.pdf`

[Bettstetter 2001]
BETTSTETTER, Christian: Mobility modeling in wireless networks: categorization, smooth movement, and border effects. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5 (2001), Nr. 3, S. 55–66

[Camp et al. 2002]
CAMP, Tracy, BOLENG, Jeff und DAVIES, Vanessa: A Survey of Mobility Models for Ad Hoc Network Research. In: *Wireless Communication and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications* 2 (2002), Sep., Nr. 5, S. 483–502

[ETSI 1998]
European Telecommunications Standards Institute (ETSI): *Universal Mobile Telecommunicatios System (UMTS) - Selection procedures for the choice of radio transmission technologies of the UMTS*. UMTS 30.03 version 3.2.0, TR 101 112. 1998

[Hong et al. 1999]
HONG, Xiaoyan, GERLA, Mario, PEI, Guangyu und CHIANG, Ching-Chuan: A Group Mobility Model for Ad Hoc Wireless Networks. In: *Proc. of the ACM Int. Workshop on Modelling and Simulation of Wireless and Mobile Systems (MSWiM)* (1999), S. 53–60

[Hoogendoorn / Bovy 2001]
HOOGENDOORN, S. P. und BOVY, P. H. L.: State-of-the-art of vehicular traffic flow modelling. In: *Journal of Systems and Control Engineering - Special Issue on Road Traffic Modelling and Control* 215 (2001), Nr. 4, S. 283–304

[Johansson et al. 1999]
JOHANSSON, Per, LARSSON, Tony, HEDMAN, Nicklas, MIELCZAREK, Bartosz und DEGERMARK, Mikael: Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In: *Proc. of the Mobicom* (1999), S. 195–206

[Liang / Haas 1999]
LIANG, Ben und HAAS, Zygmunt J.: Predictive distance-based mobility management for PCS networks. In: *Proc. of the IEEE Infocom* (1999), S. 1377–1384

[Musolesi / Mascolo 2008]
MUSOLESI, Mirco und MASCOLO, Cecilia: Mobility Models for Systems Evaluation. In: *State of the Art on Middleware for Network Eccentric and Mobile Applications (MINEMA)* (2008)