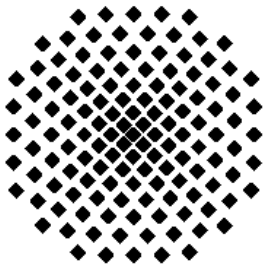


Universität Stuttgart
Fakultät Informatik

Prüfer: Prof. Dr. K. Rothermel

Betreuer: Dipl.-Inf. Jing Tian

begonnen am: 2.1.2002

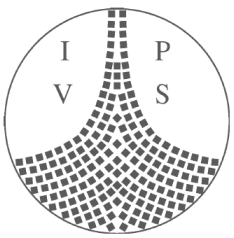
beendet am: 1.7.2002

CR-Klassifikation: C.2.1, C.4, H.3.4, I.6.0

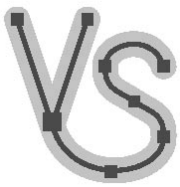
Diplomarbeit Nr. 1989

Integrating Realistic Mobility Models In Mobile Ad Hoc Network Simulation

Illya Stepanov



Institut für Parallele und Verteilte
Systeme (IPVS)
Abteilung Verteilte Systeme
Breitwiesenstr. 20-22
70565 Stuttgart



Abstract

User's mobility is a key feature of the Mobile Ad Hoc Network (MANET). So far most of simulations simplify it with arbitrary movement within the simulation area, which doesn't fully reflect the reality. In the real world the movement of mobile objects, such as people and vehicles, is constrained by environment and their travel behaviour.

This thesis presents more realistic mobility model, the User-Oriented Mobility Model, which adds more details to the simulation and is adaptable to different scenarios. In evaluation section, MANET is simulated with both the proposed mobility model and with random model to demonstrate effect of using more realistic mobility models in simulations.

Acknowledgements

I would like to thank Professor Kurt Rothermel and Professor Vladimir Svjatnyj for giving me an opportunity to write this thesis.

I would like to express my gratitude to members of the CANU Research Group for their help and valuable advises during my writing of the thesis (in alphabetical order): Dr. Christian Becker, Joerg Haehner, Abdelmajid Khelil, Gregor Schiele, and, to my supervisor, Jing Tian.

I would also like to thank my family and my friends for their support during this time. Special thanks to Eugen Bratslavskiy, Jan Mihalyovics, and Kathrin Zeller for editing this thesis.

Separate thankfulness to Marina Kurilova.

Contents

1 Introduction

The mobile ad hoc network (MANET) is formed by a number of wireless mobile devices without using any kind of existing network infrastructure. MANETs are often used in environments, where centralised network is absent or expensive to install and use. Mobile devices in such networks normally use cheap low-energy communication technologies, such as IEEE802.11 or Bluetooth. Therefore, they have limited bandwidth and communication range. This reduces number of neighbours in the communication range, so the mobile devices have to act not only as originators, but also as intermediate routers to help forwarding packets to destination. Applications of these networks are military or rescue operations, tourist guides, etc. Many simulation tools were proposed to evaluate such a networks.

User's mobility is a key feature of MANET. Thus, simulation tools have to include mobility to get a feedback about performance of communication protocols in more real network topologies.

1.1 Motivation

Most evaluations of mobile ad hoc networks are performed using simple random movement models, primarily because of their simplicity and easiness of implementation.

However, in reality people do not walk through walls, cars do not drive into rivers. So, the simulation results, which were obtained with the random models, may not reflect the real situation. For example, in [18] movement of persons was constrained with a graph, abstracting underlying road structure. The simulation showed increase in number of packets being delivered and decrease in average end-to-end packet delay. So, the performance of communication protocols in mobile ad hoc networks could be misinterpreted with the simplified simulation models.

Moreover, there are some protocols, which cannot be evaluated with the random movement models, such as routing protocols, that use regularity of mobility for movement prediction. For example, Spatial Aware Geographic Forwarding searches for a path to destination using geographical area information, based on the fact, that movement of mobile host is restricted by the spatial environment.

Another example: tourists in a city move between points of interest, and information about the sights is transferred into their PDAs. Because of bandwidth limitation, information can not be delivered at once, that's why the sending has to be initiated earlier, and the moment of initiation has to be determined using movement prediction. The scenario cannot be simulated with the random movement model, because this movement cannot be predicted.

Generally, more realistic mobility models are necessary to evaluate more accurately performance of protocols in different environments.

1.2 Thesis Objective

The task of the thesis is to design and integrate realistic mobility models to existing CANU Simulation environment in order to simulate the movement of mobile objects closer to real world. But what level of details should be included in the "realistic mobility model"? In [47] authors mention: "Selecting the correct level of detail (or level of abstraction) for a simulation is a difficult problem. Too little detail can produce simulations that are misleading or incorrect, but adding detail requires time to implement, debug, and later change, it slows down simulation, and it can distract from the research problem at hand". So, it is always a trade-off between simplicity and reality. Model is always constructed via abstraction of reality, including or excluding certain elements.

This also concerns simulation of movement process: even mobility traces reflect the reality with a certain degree of repeatability (maybe in a day, when traces were made, there was a special day for the respondent, so adequate movement pattern wasn't fully obtained). On the other hand, people do not move totally randomly: their movement has certain degree of regularity. So, a suitable compromise has to be found between the totally random movement and predetermined combination of traces, and it has to be properly evaluated to show that it makes any difference from using the random movement.

The most important requirements to this thesis are:

- Spatial constraints, such as infrastructures or obstacles, have to be reflected in the models.
- Behaviour patterns of individual mobile objects have to be included.
- The mobility models have to be seamlessly integrated into the CANUSim simulator.

To evaluate designed mobility models, a set of appropriate scenarios is to be chosen. Random and realistic movement models have to be simulated and compared using Location-Aided Routing Protocol (LAR).

1.3 Structure of this Work

This thesis has the following structure. Chapter 2 presents review of related work in the area of mobility modelling. In Chapter 3, a more realistic model, User-Oriented Mobility Model, is proposed. In Chapter 4, implementation details of the model are described. Chapter 5 presents results of evaluation of the model, and finally, Chapter 6 concludes the thesis and proposes directions for future work.

2 Related Work

Every simulation of a mobile ad hoc network includes the simulation of user's mobility. The following approaches to the mobility simulation were proposed (Fig.2-1).

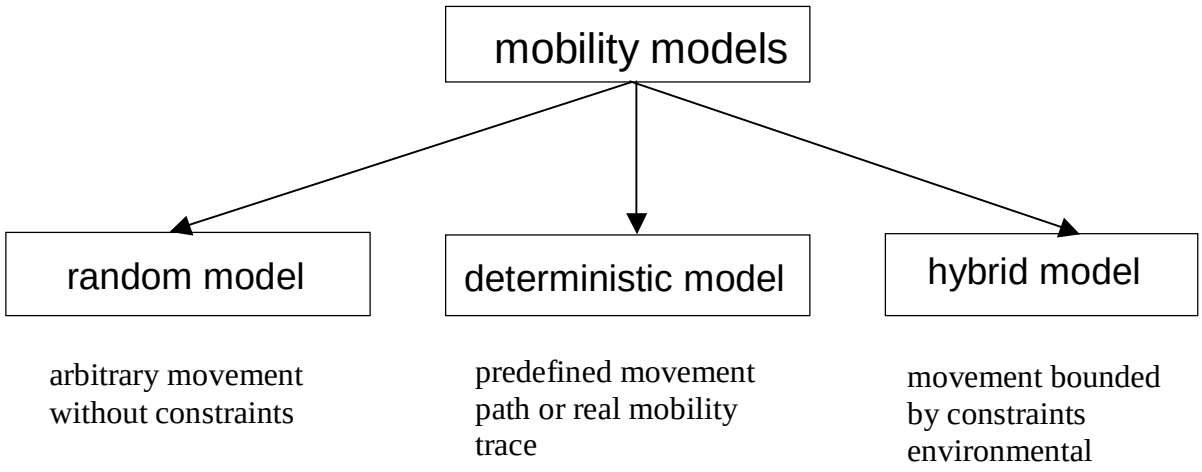


Fig. 2-1: Classification of Mobility Models

Random mobility models simulate random movement within certain area and are simple to implement. Parameters of movement constantly change upon a simulation, thus producing the random motion. Most of simulations of mobile ad hoc networks use such models, which is understandable, because authors were concentrated on developing new communication protocols, rather than on constructing mobility models for the evaluation. The random movement model was sufficient for their scenarios, which mostly included military or rescue operations. But the random movement doesn't reflect most of details of real movement, e.g., environment constraints. It seems to be that people can walk through walls, cars neglect roads and move over each other.

In opposite to random models, *deterministic mobility models* represent real traces of people's movement in the simulation area. Such models can adequately reflect the realistic movement, but are limited to a given area and reflect the reality with a certain degree of repeatability. To obtain such traces, a tremendous work has to be done: proper questionnaires are to be prepared, thousands of respondents are to be asked, result information has to be systematically processed and collaborated – trip surveys take a lot of time and resources.

But people do not move totally randomly, their movement has certain degree of generality, so a suitable compromise has to be found between totally unrealistic, undetermined, random movement, and predetermined combination of traces. *Hybrid mobility models* add more details to the random movement in order to restrict it and to adapt it as close to the realistic movement as possible. Such models are the trade-off between the simplicity of random movement and complexity of deterministic movement.

This chapter presents an overview of different approaches to simulate user's mobility in mobile network.

2.1 Random Mobility Models

Random Walk Mobility Model

Initial simulations of mobile networks were performed in scenarios containing base stations (centralised infrastructure) [1, 2, 3, 5].

Simulation area was approximated with cells (Fig.2-2), assuming that each cell contains a base station. The goal of the mobility simulation was to obtain a list of cells, being visited by a subscriber. In one-dimensional model each cell has only two neighbours (suitable for simulations of movement, which is restricted to two dimensions, for example, on roads, in tunnels, etc.). In two-dimensional model there can be a different number of neighbours (typically 4 or 6, which is suitable for more general case, where terminals can travel in any direction within a covered area, for example, in a city). This a kind of movement is called *Random Walk Mobility Model*.

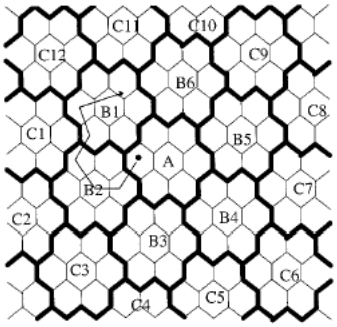


Fig.2-2: Approximation of Area with Cells for Random Walk Model (taken from [3])

The model is defined as follows: in each discrete moment of time *t* a mobile terminal either moves to one of neighbouring cells with probability *q*, or stays at the current cell with probability *1-q* independently from previous movement. If the terminal decides to move to another cell, one of the neighbouring cells is chosen with equal probability.

The users argue that it is the most adequate for the base-station scenario, because the most of mobile subscribers in such a network are likely to be pedestrians. Their movement takes place within a certain geographic area with frequent stop-and-go behaviour and frequent direction change (Fig.2-3). The *Markov chains* and *Markov processes* can be used for mathematical formalisation of the model [1, 3, 4, 6].

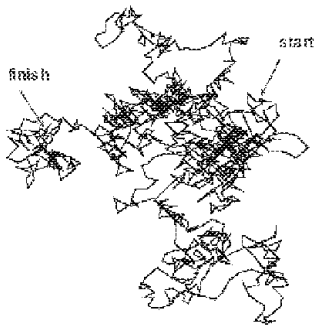


Fig.2-3: Random Walk: 1000 Steps Going Nowhere (taken from [17])

A Markov process is stochastic process, whose future behaviour is determined only by its present state (all past history is encapsulated in the present state) [6]. This property allows the process to formalise the Random Walk, whose behaviour is also determined only by its current state. If the states of a Markov process can take only discrete values, this is called a *Markov chain* (Fig.2-4) and can be described by transition probability matrix *P*. An element P_{ij} of the matrix holds a probability of switching from state i to state j (Fig.2-5).

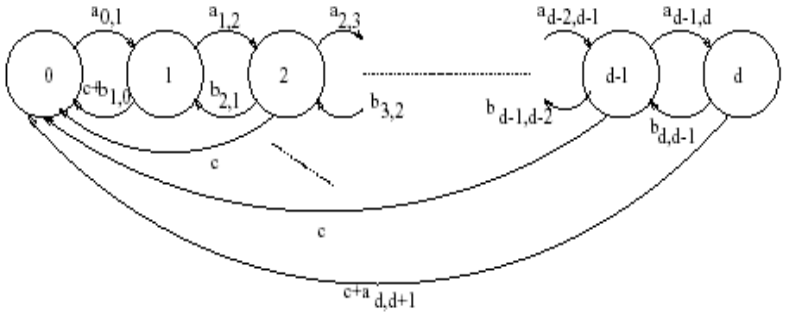


Fig.2-4: Markov Chain (taken from [1])

$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdots \\ p_{10} & p_{11} & p_{12} & \cdots \\ p_{20} & p_{21} & p_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Fig.2-5: Transition Probability Matrix for Markov Chain

Markovian Movement Model

Many authors use the *Markovian Movement Model* [11, 12], which is one-dimensional version of the Random Walk (Fig.2-6), described by a 3x3 matrix (Fig.2-7):

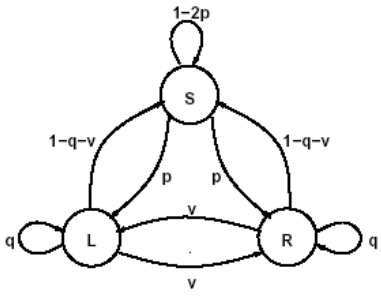


Fig.2-6: State Diagram of Markovian Walk (taken from [11])

$$P = \begin{bmatrix} q & 1-q-v & v \\ p & 1-2p & p \\ v & 1-q-v & q \end{bmatrix}$$

Fig.2-7: Transition Probability Matrix of Markovian Walk

According to the model, a user can be in one of three states: L – left moving state (moves to a cell, which is on the left from the current), R – right moving state (moves to a cell, which is on the right from the current), S – stationary state (stays at the current cell).

The disadvantage of all the described movement models is lacking of reality, because definite destination point is missing, thus making objects' movement chaotic and aimless. Primarily the model can only be used to simulate movement of pedestrians, but frequently changeable speed is far from characterising behaviours of real objects.

Normal Walk Model

Another mobility model - *Normal walk Model* [7] - extends the Random Walk by adding a dependency of movement from the previous state. Let Y_{i-1} denote a direction of the movement at step *i-1*, and direction of step *i* after anticlockwise rotation through an angle Θ_i be Y_i, then

$$Y_i = R(\Theta_i)Y_{i-1}$$

where

$$R(\Theta) = \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix}$$

Consequently $Y_n = R(\Theta_n)R(\Theta_{n-1})...R(\Theta_1)Y_0 = R(\sum_{i=1}^{n}\Theta_i)Y_0$.

Let Z_n denote the coordinate of a node after the *nth* movement, initially $Z_0 = \begin{pmatrix}0\\0\end{pmatrix}$ and $Y_0 = \begin{pmatrix}1\\0\end{pmatrix}$.

Then:

$$Z_n = Z_{n-1} + Y_n = \sum_{i=1}^{n} R(\sum_{j=1}^{i}\Theta_j)\begin{pmatrix}1\\0\end{pmatrix} = \sum_{i=1}^{n}\begin{pmatrix}\cos(\sum_{j=1}^{i}\Theta_j)\\ \sin(\sum_{j=1}^{i}\Theta_j)\end{pmatrix}.$$

A distribution of Θ_i depends on individual behaviour. The Normal walk is *restricted*, if Θ_i is approximated by *discrete* probability distribution and confined to some certain values. For example, restriction of Θ_i to 0 and π makes the mobility model act as one-dimensional Random Walk, restriction to $0, \pm\frac{\pi}{2}, \pi$ leads to mesh layout with four moving directions, restriction to $0, \pm\frac{\pi}{6}, \pm\frac{\pi}{3}, \pi$ approximates the model to hexagonal layout with six directions.

To smooth the curve, the authors distribute Θ_i continuously (normal distribution with zero mean, and then round Z_n to nearest base station). The lower the invariance of Θ_i, the smoother the curve and the larger number of consecutive cell crossings along the same direction.

Brownian Mobility Model

Brownian Mobility Model – simulates traffic of moving objects as traffic of molecular particles. A mobile object can freely move within borders of simulation area and its movement is characterised by a motion vector $\vec{V} = (V, \Theta)$, where:

V – speed of movement,
Θ – direction of movement.

The speed and direction of motion are randomly chosen from $[V_{\min}, V_{\max}]$ and $[\Theta_{\min}, \Theta_{\max}]$ respectively, making objects' movement really random. The parameters are changed after the node travelled a distance d, which is exponentially distributed with a certain mean value [14, 15], thus making the movement more realistic, because the object doesn't change the parameters on every time step, but keeps them constant for some amount of time (Fig.2-8, 2-9).

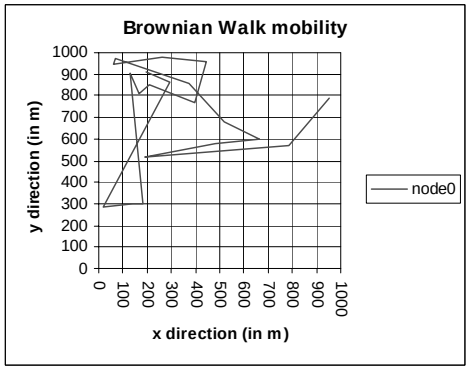


Fig.2-8: Example of Trace of Brownian Mobility Model with d=500m

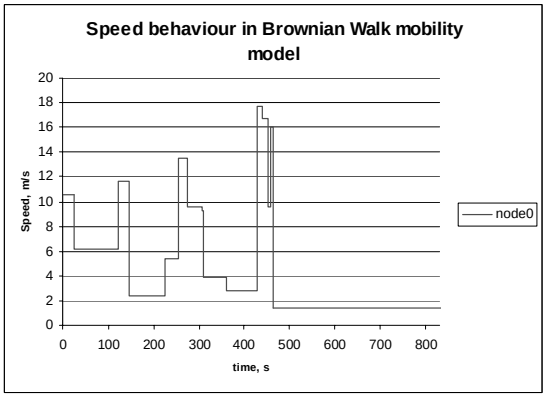


Fig.2-9: Example of Speed Behaviour in Brownian Mobility Model

Incremental Mobility Model

The *Incremental Mobility Model* [13] is based on the Brownian motion, but parameters are changed *incrementally* on every Δt seconds, thus producing the smoother movement (Fig.2-10, 2-11):

$$V(t+\Delta t) = \min(\max(V(t)+\Delta V, 0), V_{\max})$$
$$\Theta(t+\Delta t) = \Theta(t)+\Delta\Theta$$
$$x(t+\Delta t) = x(t)+V(t)*\cos\Theta(t)$$
$$y(t+\Delta t) = y(t)+V(t)*\sin\Theta(t)$$

$$\Delta V = (-A_{\max}*\Delta t, A_{\max}*\Delta t)$$
$$\Delta\Theta = (-\alpha_{\max}*\Delta t, \alpha_{\max}*\Delta t),$$

where:

A_{max} – maximal value of acceleration/deceleration,

α_{max} – maximal value of direction angle change.

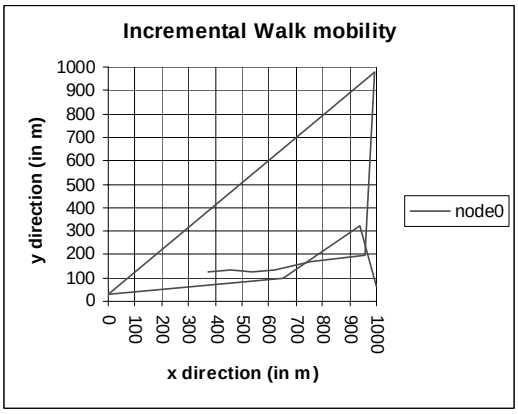


Fig.2-10: Example of Trace of Incremental Mobility Model with t=100s

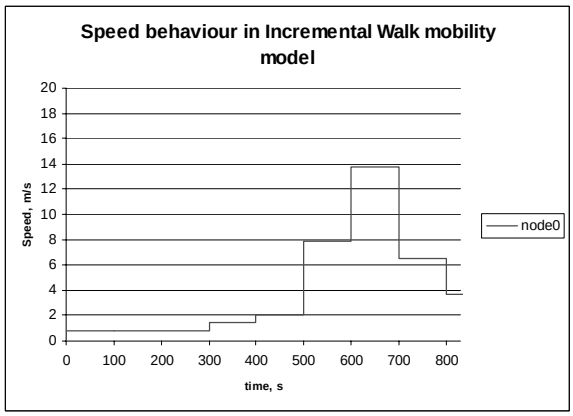


Fig.2-11: Example of Speed Behaviour in Incremental Mobility Model with t=100s

Smooth Mobility Model

A model, presented in [16], the *Smooth Mobility Model*, adds to the Brownian movement a correlation to previous values, therefore avoiding sudden speed ($\frac{\partial}{\partial t}V(t) \to \infty$) and direction (large $\frac{\partial}{\partial t}\varphi(t)$, while V is high) changes. Upon turning, the direction doesn't change at once, but in several time steps until target direction is achieved. The speed has a set of preferred movement speeds, thus making its behaviour more realistic (for example, typical speed in a city is 50-60 km/h, outside the city is 100-120 km/h, and it can be reflected by the model).

A node moves with a constant speed until speed change event occur. The new speed value is chosen from:

$$V^* = \begin{cases} V_{pref1} & p(V_{pref1}) \\ V_{pref2} & p(V_{pref2}) \\ \dots & \dots \\ V_{prefn} & p(V_{prefn}) \\ 0 < V < V_{\max} & 1 - p(V_{pref}) \end{cases}, \qquad V_{pref} = \sum_{i=1}^{n} p(V_{prefn}) < 1$$

The speed changes incrementally with acceleration, which is uniformly distributed in $[-A_{\max}, 0)$ (if V^*<V) or $(0, A_{\max}]$ (if V^*>V).

Time t between two speed changes events is chosen from the exponential distribution of $p(t) = \frac{p_{v^*}}{\Delta t} \times e^{p_{v^*} \times t / \Delta t}$, where:

Δt - duration of single simulation time step

p_{v^*} - probability of changing speed at single time step, $p_{v^*} << 1$

This improved algorithm of speed choice makes its behaviour more realistic, comparatively to speed behaviour in the simple Brownian Walk model (Fig.2-12).

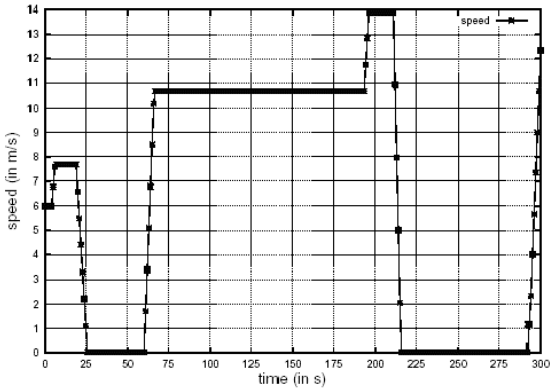


Fig.2-12: Example of Speed Behaviour in Smooth Random Mobility Model in Downtown (taken from [16])

Time of direction change is independent from the speed changing time and determined by its own exponential distribution. New direction value is uniformly distributed in $0 \le \varphi \le 2\pi$. The direction is changed incrementally in several time steps. The total time of direction change, the "curve time", Δt_c, are obtained from a uniform distribution $[\Delta t_{c_{\min}}, \Delta t_{c_{\max}}]$, therefore making the movement *smoother* (Fig.2-13).

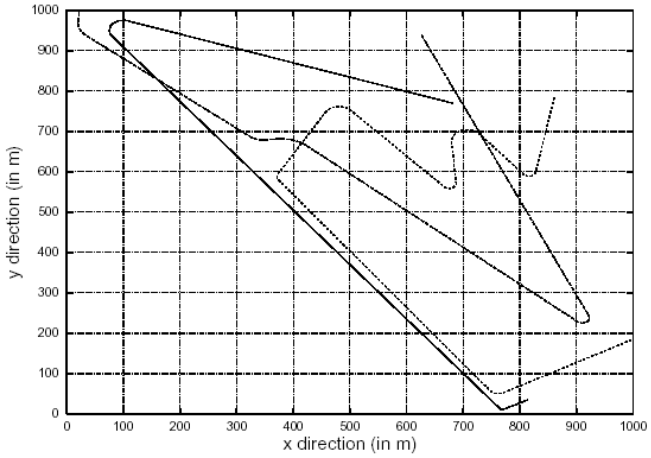


Fig.2-13: Example of Three Traces of Smooth RandomWalk mobility (taken from [16])

Gauss-Markov Mobility Model

The *Gauss-Markov Mobility Model*, which is presented in [19, 20], is a compromise between random and deterministic movement models. It has a tuning parameter, which adapts it to different levels of randomness - from fully deterministic ("fixed motion") to totally random, Brownian motion. The mobility model is expressed with the equation:

$$\overline{V_n} = \alpha\overline{V_{n-1}} + (1-\alpha)\overline{\mu} + \sqrt{(1-\alpha^2)}\,\overline{x_{n-1}}$$

where:

α – tuning parameter, $0 \le \alpha \le 1$,
V_n – object's velocity vector,
x – independent, uncorrelated, stationary Gaussian process with zero mean ($\mu_x = 0$) and standard deviation $\sigma_x = \sigma$, σ-asymptotic standard deviation of V_n, when $n \to \infty$,
μ – asymptotic mean of V_n, when $n \to \infty$.

With $\alpha = 0$ the equation describes random Brownian motion ($\overline{V_n} = \overline{\mu} + \overline{x_{n-1}}$), when $\alpha = 1$ - deterministic constant motion ($\overline{V_n} = \overline{V_{n-1}}$), when $\alpha \in (0,1)$ – random motion with a certain degree of memory (Fig.2-14, 2-15).

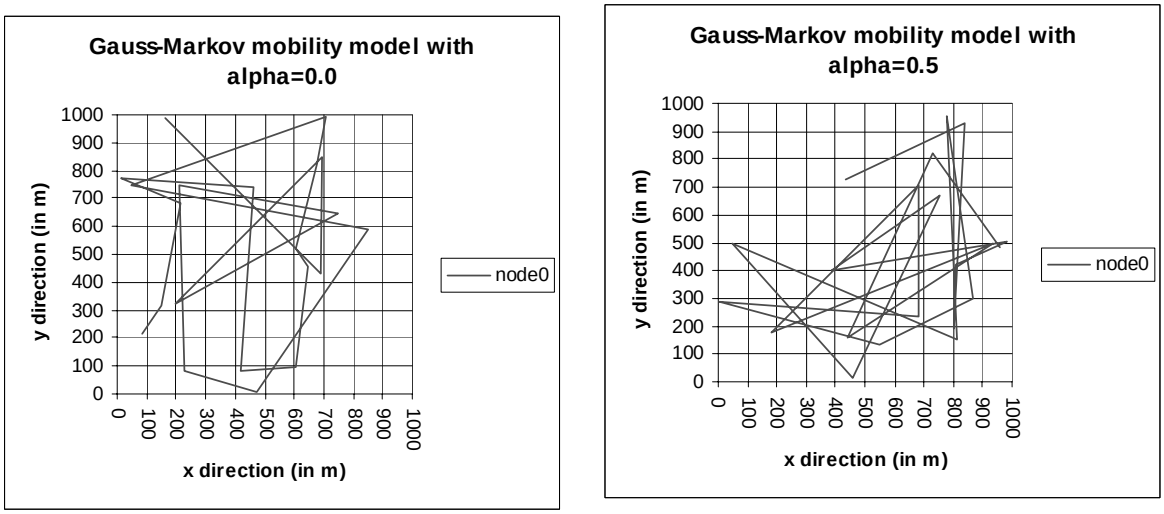


Fig.2-14: Example of Trace of Gauss-Markov Mobility Model

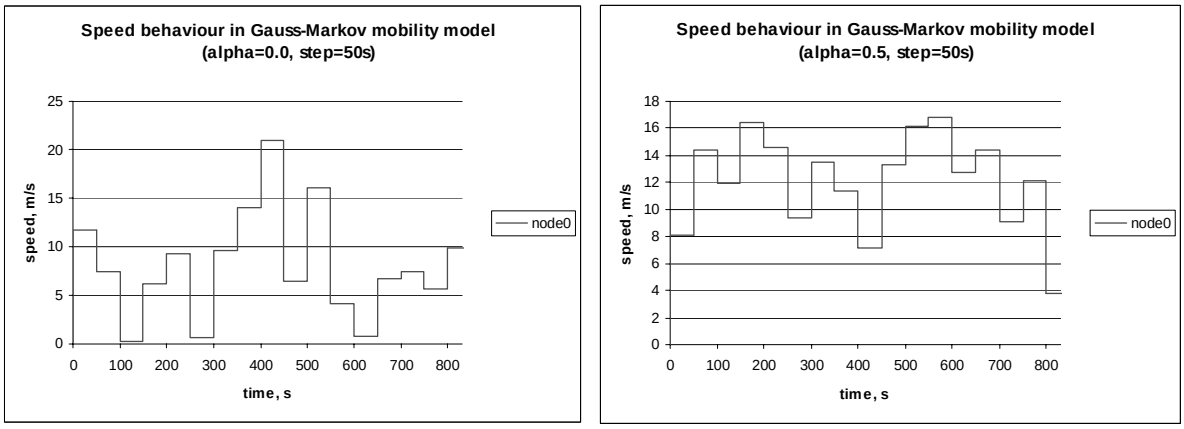


Fig.2-15: Example of Speed Behaviour in Gauss-Markov Mobility Model

The disadvantages of the Brownian and derived movement models are that they do not consider sense of movement: objects constantly modify their motion parameters, but a definite destination point of the movement is missing.

Random Waypoint Mobility Model

The destination point and speed of motion are the characteristics of the *Random Waypoint Mobility Model* [23-25]. This model is widely used in simulations of mobile ad hoc networks and is implemented under the wide majority of MANET simulators ([21, 22]). The model is defined as followed: at the beginning all the nodes are located at initial positions, then they choose destination points within simulation area and the speed of movement V from $[V_{\min}, V_{\max}]$. The speed remains constant during the movement. After arriving to the destination, a node stays there for some time (between $[t_{p_{\min}}, t_{p_{\max}}]$), and then the movement

continues. The examples of traces and speed behaviour in the Random Waypoint mobility model are shown in Fig.2-16, 2-17.

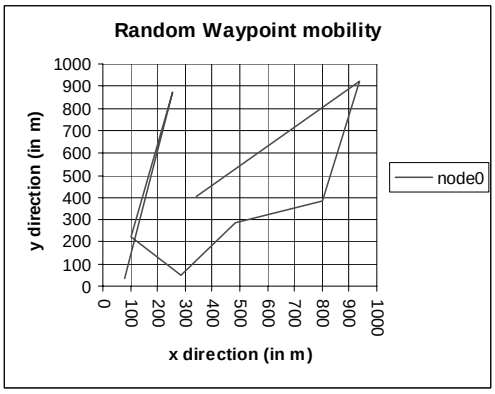


Fig.2-16: Example of Trace of Random Waypoint Mobility Model

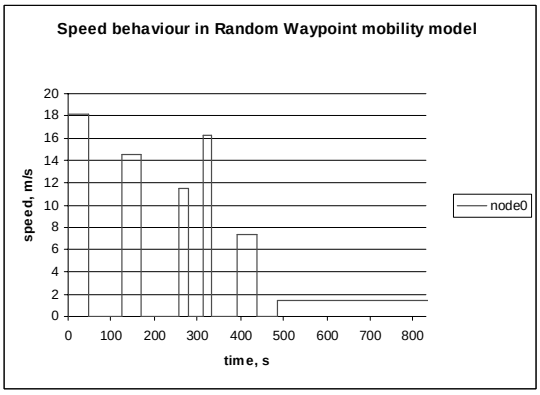


Fig.2-17: Example of Speed Behaviour in Random Waypoint Mobility Model

Some mobility models simulate movement of mobile objects as the movement of group of moving objects. This is very useful for several scenarios, for example to simulate a group of tourists in a city or visitors in a museum.

Vector Mobility Model

In [29] the authors use the *Vector Mobility Model*, which is an extendable version of the Brownian Walk model. The result mobility vector $\vec{M}$ is expressed as a sum of *base* mobility vector $\vec{B}$ and *deviation vector* $\vec{V}$:

$$\vec{M} = \vec{B} + \alpha\vec{V}$$

where:

α – acceleration factor

By properly adjusting the acceleration factor and choosing the base and deviation vectors, several models can be described:

- *Gravity model* – nodes try to be closer to a certain point (for example, closer to the Base Station in order to get a better radio signal), in such a case the base vector is directed to the point;
- *Location dependent model* – represents a collective mobility pattern in the area (for example, for the road), the base vector is defined by area;
- *Targeting model* – the node moves to a certain target and upon approaching the velocity is decreased up to zero, the base vector points to the destination;
- *Group motion model* – motion of a group of nodes, the base vector is the group's movement.

The deviation vector in these models represents the node's individual movement behaviour (deviation from the base vector).

Reference Point Group Mobility Model

A very similar model is used in [31] – the *Reference Point Group Mobility Model*. Each group has an associated logical "centre" point and group motion vector $\vec{V}_g$. The node's movement vector is calculated as a sum of the group motion vector and a random motion vector $\overrightarrow{RM}$. The vector $\overrightarrow{RM}$ has a length, which is equally distributed within a certain radius from centre of the group, and a direction, which is uniformly distributed within $[0, 2\pi]$. The path, which the group will follow, is defined as a set of checkpoints. Each time the centre of the group arrives to a current checkpoint, the new $\vec{V}_g$ and the next checkpoint are chosen and the movement continues.

Boids Model

Several authors applied the *Boids Model* to simulation of group motion [35-38]. Boids were originally created to model complex motion of dynamic group in zoology (flocks of birds, schools of fish, herds of animals). Biologists have found that grouping in animals' behaviour is created by a attraction, which modulates desire of each member to join the group with desire to maintain a sufficient distance from the nearby creatures [40]. Assuming, that the group movement is a result of interaction between single nodes, the movement is simulated as motion of individual nodes. Three sub-models define the system: perception model (what a field of sight a node has), behavioural model (what a node decides upon its current) and movement model (how a node actually moves, which is based on the behavioural model). The following behaviours lead to grouping:

- Group Centring: node tries to stay close to the nearby group-mates;
- Velocity Matching: node tries to match the velocity with the nearby group-mates;
- Collision Avoidance: node tries to avoid collisions with the nearby group-mates.

The combination of these behaviours produces the motion. This mobility model was used to simulate the Art Gallery, the Event Hall and the Battle Field scenarios [35], which differ in the speed of the movement and the distance between the nodes.

2.2 Deterministic Mobility Models

Another class of mobility model, *deterministic mobility models*, is represented by *mobility traces*, which were obtained from real life using different sociological methods, such as trip surveys, personal questionnaires, etc.

For example, dynamic location management algorithm in [46] was simulated with travel patterns from a database, which contained results of the trip survey, conducted by the Regional Municipality of Waterloo. Using this data, the author argued that the results of his work adequately reflect the reality.

Upon the survey, the information about trips per day was obtained from every household member over 5 years of age. The trip data contained information about time of starting and ending the trip, its origin and destination locations, purpose of the trip. The trip purposes were classified on:

- work,
- work-related,
- school,
- serve passenger,
- shopping,
- social-recreation,
- personal business,
- return home,
- other.

Information was recorded in the *Activity Duration* and *Activity Transition* matrices. The activity duration matrix (Fig.2-18) is used to store information about a particular trip, while the activity transition matrix (Fig.2-19) serves for switching between different activities during the simulation process (Fig.2-20).

The advantage of the model is correspondence to reality, but on the other hand, it requires high preparation overhead and expenses to make a survey in order to obtain the source data.

person type	*time period*	*activity*	*duration*	*cumulative probability*
0	7	6	330	0.948718
0	7	6	360	0.961538
0	7	6	400	0.974359
0	7	6	460	0.987179
0	7	6	540	1.000000
0	7	7	0	0.125654
0	7	7	5	0.235602
0	7	7	10	0.413613
0	7	7	15	0.539267
0	7	7	20	0.596859

Fig.2-18: Activity Duration Matrix (taken from [46])

person type	*time period*	*previous activity*	*next activity*	*cumulative probability*
1	4	8	1	0.351724
1	4	8	2	0.393103
1	4	8	3	0.420690
1	4	8	4	0.475862
1	4	8	5	0.696552
1	4	8	6	0.793103
1	4	8	7	0.986207
1	4	8	8	1.000000
1	4	8	9	1.000000
1	4	9	1	0.000000

Fig.2-19: Activity Transition Matrix (taken from [46])

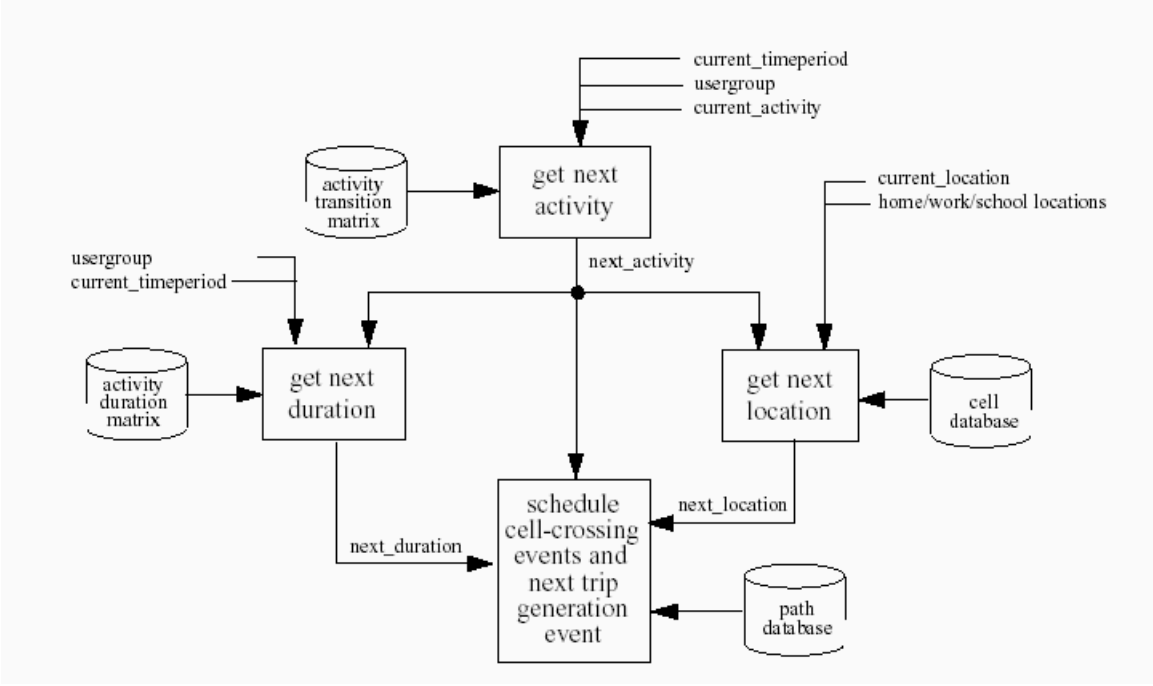


Fig.2-20: Structure of Activity-Based Mobility Model (taken from [46])

2.3 Hybrid Mobility Models

Hybrid mobility models combine random movement with regularity of deterministic movement.

Global-Local Mobility Model

For example, in [8] a mobility model is built as a two-level hierarchy: the *Global Mobility model* (*GMM*) and the *Local Mobility model* (*LMM*). GMM is used to create intercell movements, while LMM is used to model intracell movements. The presence of the global model is motivated by a fact that most of mobile users exhibit some regularity in their daily movement. This regularity can be best characterised by a number of *user mobility patterns* (*UMP*), recorded in a profile for each user and indexed by the time of occurrence. The precious of UMP is decreased to small deviations by making *user's actual path* (*UAP*) as edited version of UMP inserting a new cell, deleting a cell from the path or changing cells, thus producing some irregularity. LMM represents the movement within a cell and can be described by three variables: position, speed, and direction. The position of an object in the next step X_{n+1} can be determined from:

$$X_{n+1} = AX_n + BU_n + W_n$$

where:

$$X_n = [\, x(n)\ \dot{x}(n)\ r_x(n)\ y(n)\ \dot{y}(n)\ r_y(n)\,]^T$$

$$U_n = \begin{bmatrix} u_x(n) \\ u_y(n) \end{bmatrix} \quad W_n = \begin{bmatrix} w_x(n) \\ w_y(n) \end{bmatrix}.$$

x(n) and y(n) – position at step n,
$r_x(n)$ and $r_y(n)$ – two-dimensional random acceleration,
A and B – state and disturbance transition matrixes,
U_n – driving command (two-dimensional position increment),
W_n – discrete time Gaussian white noise (deviation of movement).

Movement Circle and Movement Track Models

Another random mobility model with profile extension is used in [9, 10]: the *Movement Circle* model (*MC*) and the *Movement Track* model (*MT*). The MC model is based on assumption that wherever a user moves from a location, he eventually returns to the start point. The movement is modelled as different circle-like patterns (Fig.2-21). A number in the circle identifies each state and indicates a location in the area. The MT model is the less restricted version of MC and starts and ends with stationary or boundary state, which are not equal to the initial state (Fig.2-22). The MC and MT models are used to describe the regular movement behaviour of a node. To describe the random behaviour, the authors use the Markov Chains.

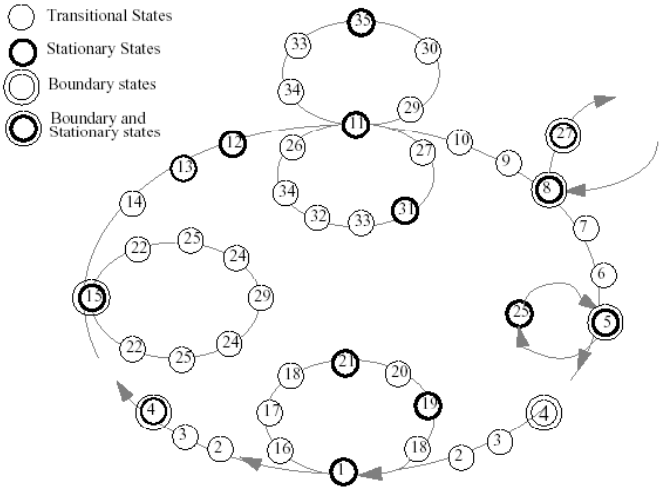


Fig.2-21: Example of MC model (taken from [9])

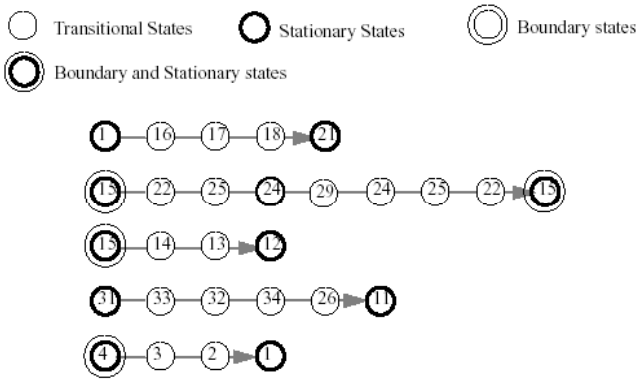


Fig.2-22: Example of MT model (taken from [9])

Several authors attempted to get more realistic movement by adding obstacles to the existing models.

Random Waypoint Mobility Model with Obstacles

For example, in [26] three scenarios were created: *conference*, *event coverage* and *disaster* (Fig.2-23, 2-24, 2-25). Nodes use Random Waypoint mobility, but their speed is limited according to typical speed in such an environment. Furthermore they avoid obstacles upon the movement. If an obstacle crosses a straight line between two communicating nodes, the link is considered broken, thus making the communication to be opaque to obstacles.

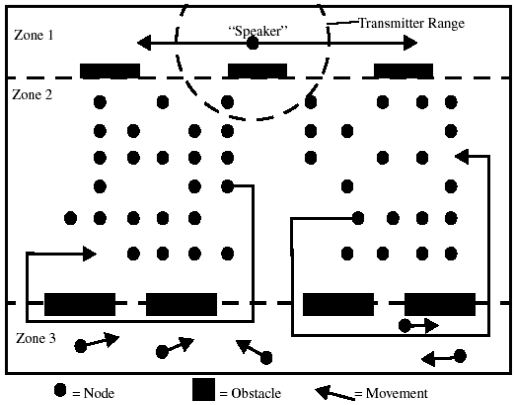


Fig.2-23: Example of Conference Scenario (taken from [26])

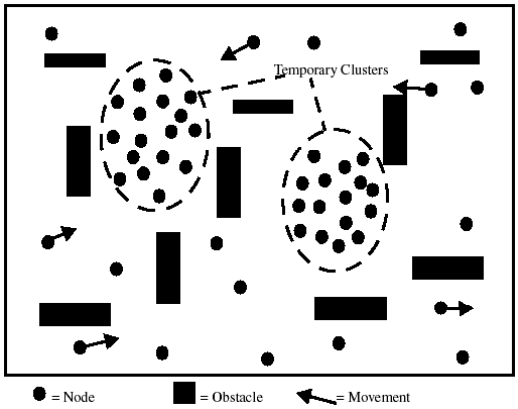


Fig.2-24: Example of Event Coverage Scenario (taken from [26])

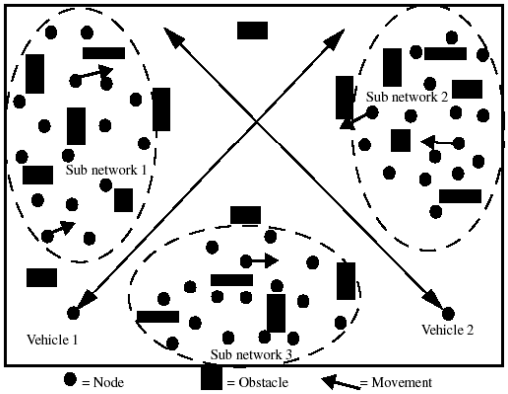


Fig.2-25: Example of Disaster Area Scenario (taken from [26])

Restricted Random Waypoint Mobility Model

The *Restricted Random Waypoint* mobility model is used in [27]. Simulation area consists of three "big" movement areas ("towns") and three highways, connecting the towns. Within a town the node uses the Random Waypoint mobility model, but with certain probability it can move to another town by a "highway" between (Fig.2-26).

In [28] the *Urban* scenario was simulated. The area consists of a number of regularly placed buildings and streets (Fig.2-27), representing the centre of a modern city. Nodes move only on streets; the buildings are not transparent to radio signals and stress communication.

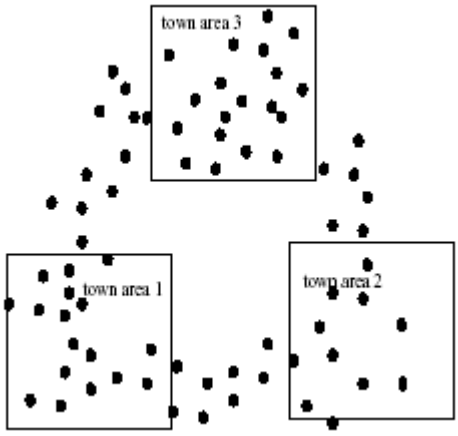


Fig.2-26: Simulation Area with Three Towns (taken from [27])

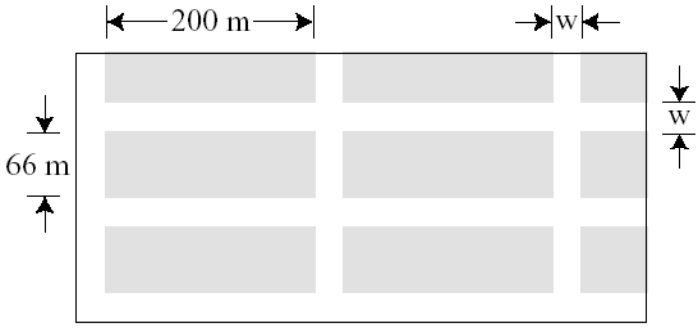


Fig.2-27: Urban Scenario (taken from [28])

Graph Walk Mobility Model

Another movement model, which restricts movement area for the Random Waypoint movement, is the *Graph Walk* mobility model, proposed in [18]. A complex city centre is approximated with a *graph*, in which vertices represent points of interest for mobile objects and edges represent connections between them. The node's movement is restricted with the

edges and takes place between the vertices. Every vertex has the same weight value, in other words, it is as attractive for a mobile user as any other vertex.

At the beginning nodes are located at initial positions at vertices, then they randomly choose destination points and move there with the certain speed (between $[V_{\min}, V_{\max}]$). After arriving to the destination, a node stays there for some time ($[t_{p_{\min}}, t_{p_{\max}}]$), and then its movement continues. The idea of the model is equal to the Random Waypoint Walk mobility, but the movement takes place only on edges of the graph (Fig.2-28).

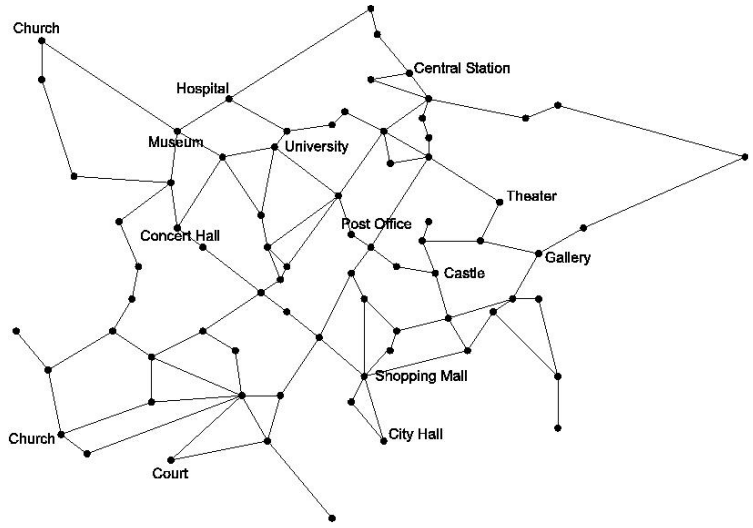


Fig.2-28: Graph of Stuttgart Centre (taken from [18])

Simulation with the Graph Walk movement model showed the difference in performance of communication protocols in comparison to commonly used Random Waypoint movement model. Assignment of "attraction" value to every point can extend the model, thus making the trips more realistic.

3D-Movement Model

In [30] the *3D-movement* model was used. Users move straight on square shaped floors of a building, until they change direction (turn left, right or back according to Poisson distribution) and then continue the straight motion (Fig.2-29). If a turning point is located in the staircase region, the user moves horizontally or vertically with a certain probability. The speed of horizontal or vertical motions is uniformly distributed in [0, $V_{\max}$] and [0, $V_{\max}'$] respectively.

Integrated Mobility Model

A complex mobility model is used in [34]. The authors argue, that modern simulations of mobile networks require the usage of more realistic mobility models than fluid or random models. Three levels of the mobility model are introduced:

- *City Area model* level: set of area zones, connected via high-capacity routes;
- *Area Zone model* level: set of streets and set of building blocks;
- *Street Unit model* level: highways, streets with traffic-light(s)-controlled flow, high/low-priority streets.

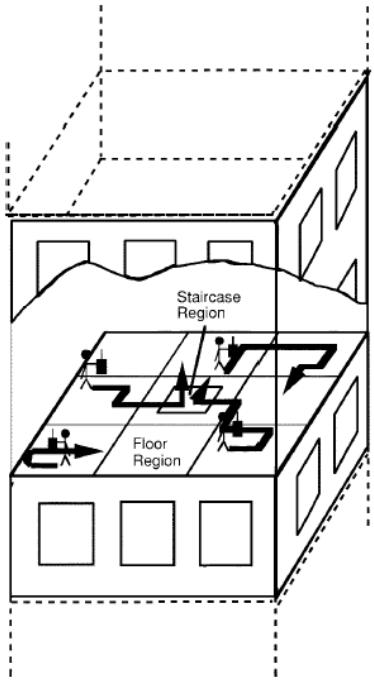


Fig.2-29: Example of 3D-movement Model (taken from [30])

The City Area model (Fig.2-30) describes user movement within a city area environment. The parameters of the model include:

- Geographical area definition (area zones and high-capacity routes);
- Population (different groups of users by mobility rate, for example, cars, pedestrians, etc., their mobility and traffic behaviour);
- Time period (rush hours, busy hours, etc.).

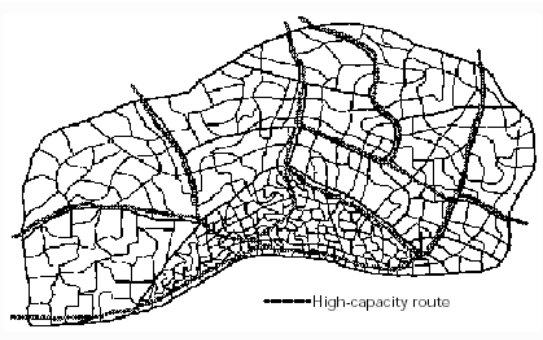


Fig.2-30: Example of City Area Model (taken from [34])

The Area Zone model (Fig.2-31) represents traffic behaviour in a microcell area. The parameters of the model include:

- Geographical area (street network and building blocks);
- Population and its traffic behaviour;
- Time period.

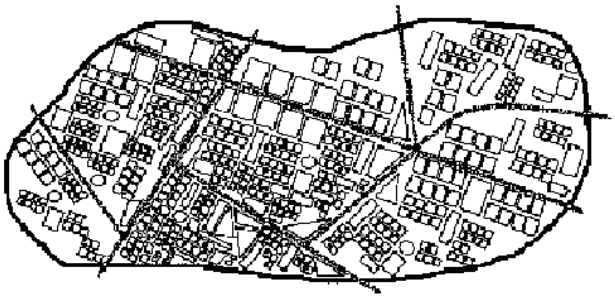


Fig.2-31: Example of Area Zone Model (taken from [34])

The Street Unit model (Fig.2-32) describes mobility behaviour with an accuracy of few meters. In order to develop such a model, a detailed analysis of motion is required. The parameters of the model include:

- Geographical area (one or more street segments connected at crossroads; a single street segment is characterised by length, number of lines and capacity – the number of cars per hour);
- Population and car driver's behaviour;
- Time period;
- Relations between cars' speed and density.

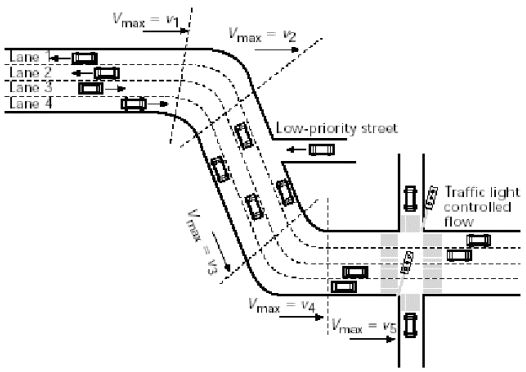


Fig.2-32: Example of Street Unit Model (taken from [34])

In order to improve the model, the authors use data from Transportation Theory: typical speeds in different parts of a city, characterisations of mobile units according to the mobility behaviour, density of population in different areas, etc. As a result, the *Integrated Mobility Modelling Tool* (IMMT) was proposed (Fig.2-33), which consists of several integrated components.

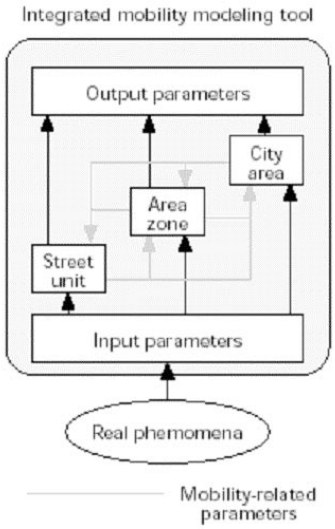


Fig.2-33: Integrated Mobility Modelling Tool (taken from [34])

3 Conception of User-Oriented Mobility Model

From the review of related work, it's clear that the new, more complex mobility models are required. This chapter proposes a mobility model, called User-Oriented Mobility Model, which reflects environment constraints, includes profile-based trips and simulates movement behaviour during the motion to destination.

The chapter is structured as follows. Section 3.1 presents the structure of the mobility. Section 3.2 describes the Environment Model, which is as a combination of the most popular standards, used to describe a geographical environment in digital form. Section 3.3 presents the Trip Model, which is based on the Activity-Based Travel Demand Modelling Approach [41-44], but expresses trip sequences using automata. Section 3.4 discusses the Movement Behaviour Model. Finally, Section 3.5 concludes the chapter.

3.1 Elements of the Mobility Model

According to requirements to the thesis, the mobility model has to include the following elements.

The basic element is the *Environment Model.* In this thesis only the spatial environment is considered. The spatial environment constrains the movement of mobile objects: people do not walk through walls; cars do not drive through lakes, etc.

At the next step it is important to answer the question, why do mobile objects move? According to Activity-Based Travel Demand Modelling Approach, "Travel is a demand that arises through people's needs and desires to participate in activities" [41]. People move, because they want to do something in a certain place. For example, they go shopping, to restaurants, to work, visit sights, etc. Desire to execute a certain action makes them change locations. People move between elements of the Environment Model, like shops, restaurants, enterprises, monuments, etc. Elements of the Environment Model get another meaning from perspective of the *Trip Model* – points of interest. The Trip Model is built on top of the Environment Model.

But these two models are not sufficient to simulate the movement with all details, because it is not defined, how people will move to destination. The simple and widely used solution – the constant speed motion – does not adequately reflect realistic behaviours. For example, cars do not drive over each other, but influence the movement behaviour of neighbouring vehicles: when a car ahead slows down, succeeding cars will also decrease the speed. It means that the realistic *Movement Behaviour Model* has also to be included in the mobility model.

The resulting mobility model, the User-Oriented Mobility Model, is integration of the following elements:

- Environment Model, representing spatial constraints;

- Trip Model, imitating trip planning decisions;
- Movement Behaviour Model, simulating movement to destination.

By varying these components, the model can be tuned to simulate movement of different classes of mobile objects in various environments. The model is *oriented* on a mobile user, being simulated, because miscellaneous mobile objects have their own movement area, movement behaviour and perform personal trips, but the model can be adjusted to reflect these elements.

In the next sections the components of the model will be precisely described.

3.2 Environment Model

The presence of the Environment Model is very important for the mobility model. It stores elements, which constrain the area of movement and serve as destination points for movement process.

3.2.1 Related Environment Models

Information about environments is widely used by different kinds of applications, e.g. location-aware applications, car navigation systems, etc. The most popular standards, used for description of environments in digital form, are Geographic Data Files (GDF) and Geography Markup Language (GML).

3.2.1.1 GDF-Standard

GDF [50] was primary developed to specify road network data in digital form for car navigation systems.

The GDF-data contains three major parts (*catalogues*):

- Feature Catalogue
- Attribute Catalogue
- Relationships Catalogue

Feature Catalogue

The Feature Catalogue stores topological elements of the environment. Elements of the following types can be included:

- Roads and Ferries;
- Administrative Areas;
- Settlements and Named Areas;
- Land Cover and Use;
- Brunnels;
- Railways;

- Waterways;
- Road Furniture;
- Services;
- Public Transport;
- General Features.

The Feature Catalogue can be extended with the *User-Defined Features*.

Elements of the environment are described at different *presentation levels* (Level-0 – Level-2).

Level-0 specifies the elements at the level of *geometrical primitives* (Point, Line, Area, etc.) (Fig.3-1). Geometrical primitives are specified in coordinates, which refer to the national (local) grid system, but they also can be converted to World Geodetic System (WGS).

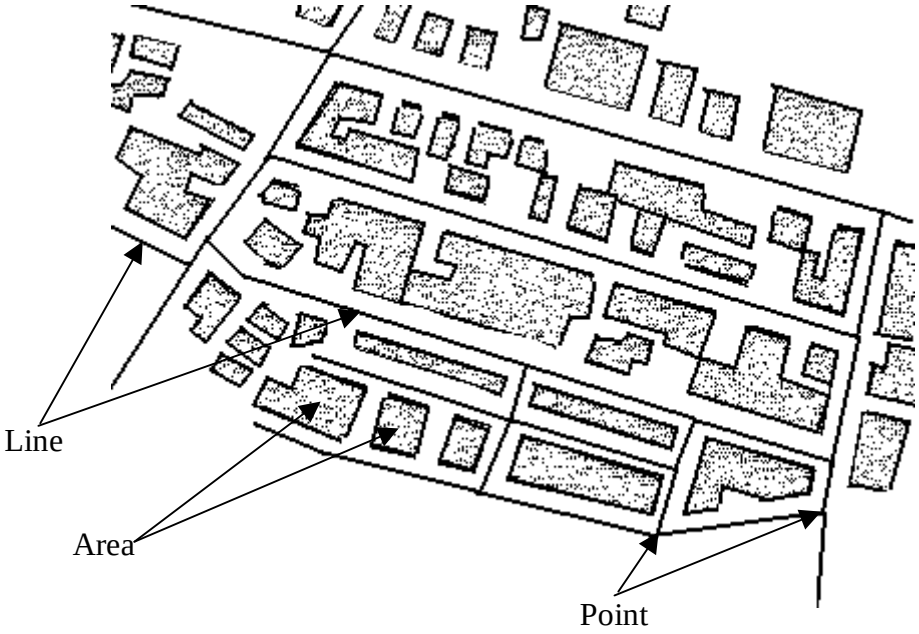


Fig.3-1: Example of GDF Elements of Level-0

At the Level-1, elements from the basic level get initial *semantic meaning*, for example, *Road Elements* (part of a Road, bounded by two points), *Buildings*, *Traffic Signs*, *Pedestrians Crossings*, etc. (Fig.3-2).

Furthermore, at the Level-2 simple features from the Level-1 are combined into more complex features, for example, several Road Elements form a *Road*, Junctions form an *Intersection*, etc. (Fig.3-3).

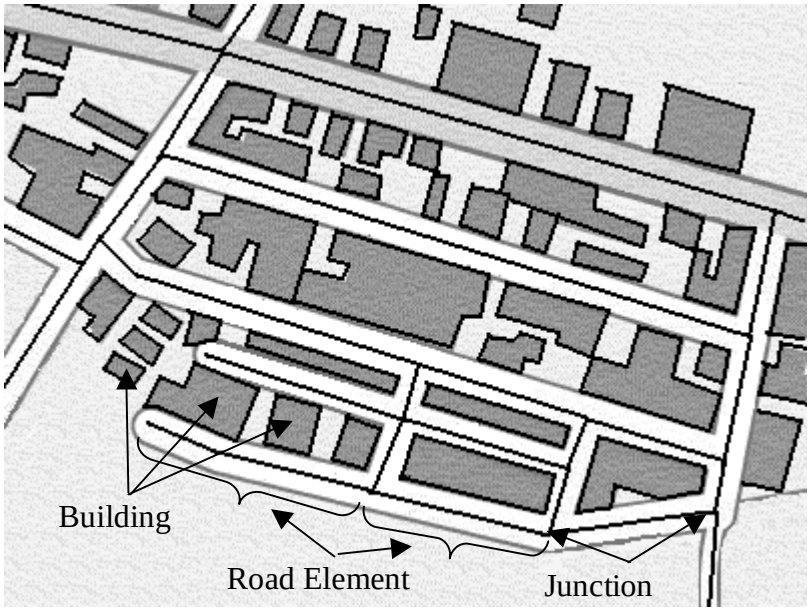


Fig.3-2: Example of GDF Elements of Level-1

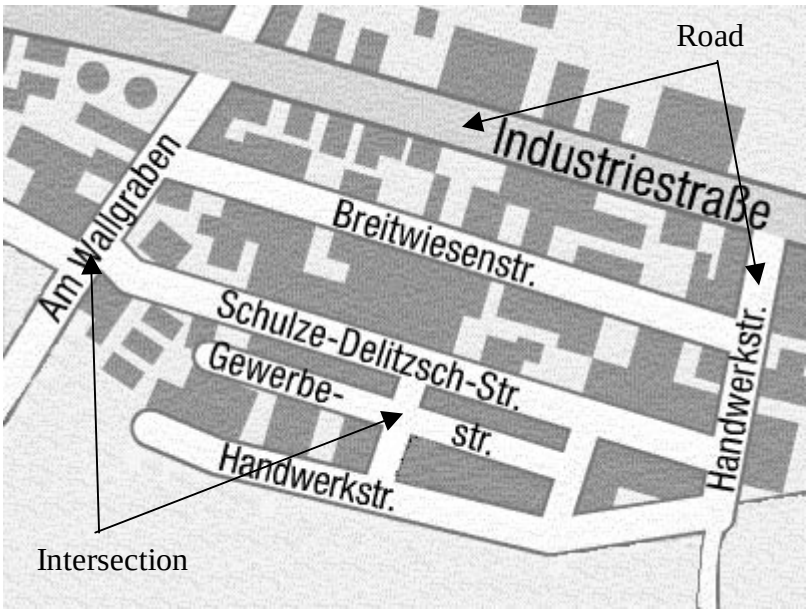


Fig.3-3: Example of GDF Elements of Level-2

Description of features at different presentation levels simplifies working with the model, for example, data from Level-2 can be used for high-level trip planning (travel from city A to city B by freeway C). Information from Level-1 makes it more detailed (travel only by particular parts of the freeway).

All possible kinds of elements of different presentation levels, which can be included to the model, can be obtained from the GDF specification [50].

Attribute Catalogue

In the *Attribute Catalogue*, *properties and particularities* of objects are described. For example, for a Road Element can be defined the following properties:

- Number of Lanes,
- Direction of Traffic Flow,
- Opening Period

Attributes for a Restaurant can include:

- Credit Card Acceptance,
- Opening Period,
- Telephone Number

Some of attributes of Traffic Sign:

- Direction,
- Textual Contents of a Traffic Sign

A full list of possible attributes for different elements can be obtained from the GDF specification [50].

Relations Catalogue

The *Relations Catalogue* is similar to the Attributes Catalogue, but it stores *relations* among features. Relations can be defined as *properties, involving more than one object* (Fig.3-4).

Relations can be:

- Reflexive, irreflexive or non-reflexive
- Symmetric, asymmetric or non-symmetric
- Transitive, intransitive or non-transitive

A relationship R is *reflexive* over set S if and only if $\forall x \in S \quad (x,x) \in R$.
A relationship R is *irreflexive* over set S if and only if $\forall x \in S \quad (x,x) \notin R$.
A relationship R is *non-reflexive* over set S if and only if it is neither reflexive, nor irreflexive.

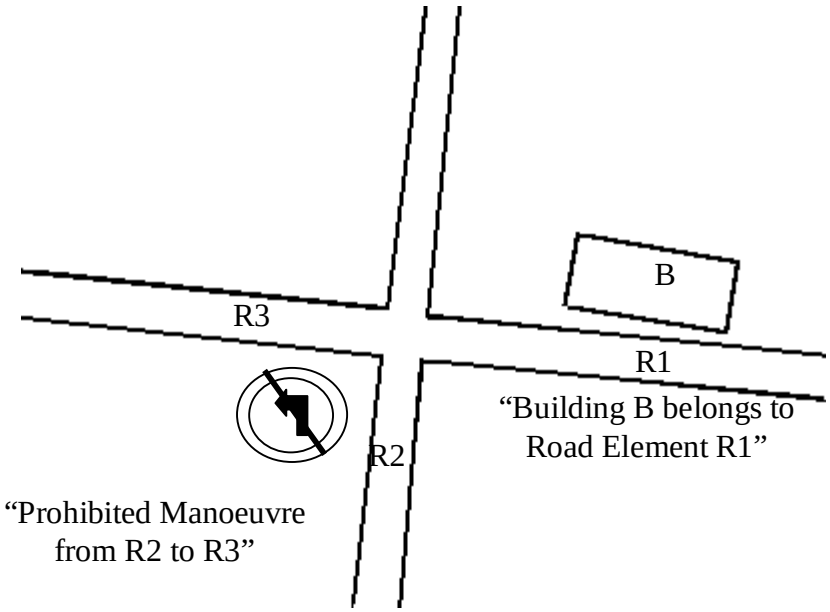


Fig.3-4: Example of Relationships

A relationship R is *symmetric* if and only if $\forall x_i \quad (x_1, x_n) \in R \text{ and } (x_n, x_1) \in R$.
A relationship R is *asymmetric* if and only if $\forall x_i \quad (x_1, x_n) \in R \text{ and } (x_n, x_1) \notin R$.
A relationship R is *non-symmetric* if and only if it is neither symmetric, nor asymmetric.

A relationship R is *transitive* if and only if $\forall x, y, z \quad (x, y) \in R, (y, z) \in R \text{ and } (x, z) \in R$.
A relationship R is *intransitive* if and only if $\forall x, y, z \quad (x, y) \in R, (y, z) \in R \text{ and } (x, z) \notin R$.
A relationship R is *non-transitive* if and only if it is neither transitive, nor intransitive.

In Fig.3-4, relation "Prohibited Manoeuvre from R2 to R3" is irreflexive (it doesn't prohibit to turn over from R2 to R2), asymmetric (it doesn't prohibit to turn from R3 to R2); relation "Building B belongs to Road Element R1" is non-reflexive (doesn't mean that building belongs to itself or building doesn't belong to itself) and symmetric (a road element can be joined to a building, for example, upon the calculation of path from the house).

All possible kinds of relationships can be obtained from the GDF specification [50].

Conclusion

GDF proposes a rich description of area using different levels of details. Not only elements themselves are described, but also their properties and relations with other elements. Advantage (and disadvantage) of GDF is high level of standardisation – standard defines all possible elements of the model, their attributes and relations with other elements, using hard-coded identifiers, but the model can be extended using user-reserved identifiers.

3.2.1.2 GML-Standard

GML (Geography Markup Language) [48] is an XML-based language for encoding geographical information.

GML doesn't explicitly define all possible kinds of elements to be included in the model. Instead, it provides basic schemas to encode:

- Feature (single element of the environment)
- Geometry of a feature
- Collection of features (several elements, merged with same semantic meaning)
- An association between features

Feature

Feature is encoded inheriting the basic feature (AbstractFeature) with common properties (e.g., id, name, description, bound area) and extending it with its own properties.

Geometry of feature

Geometry of a feature is specified using following geometrical primitives:

- Point
- Line String (non-closed polyline)
- Line Ring (closed polyline)
- Polygon (approximated with outer boundary and possible set of inner boundaries, using Line Rings)
- Multi Point (collection of points)
- Multi Line String (collection of Line Strings)
- Multi Polygon (collection of Polygons)
- Multi Geometry (collection of different geometrical elements)

An example of description of feature with geometry and attributes in GML is shown in Fig.3-5.

GML doesn't define fixed coordinate system and uses (some) spatial reference system (SRS).

```
<School ID="145l">
  <description>Balmoral Middle School</description>
  <NumStudents>987</NumStudents>
  <NumFloors>3</NumFloors>
  <extentOf>
    <Polygon srsName="epsg:27354">
      <outerBoundaryIs>
        <LinearRing>
          <coordinates>
            491888.999999459,5458045.99963358
            491904.999999458,5458044.99963358
            491908.999999462,5458064.99963358
            491924.999999461,5458064.99963358
            491925.999999462,5458079.99963359
            491977.999999466,5458120.9996336
            491953.999999466,5458017.99963357
          </coordinates>
        </LinearRing>
      </outerBoundaryIs>
    </Polygon>
  </extentOf>
</School>
```

Fig.3-5: Example of GML Data (taken from [49])

Collection of features

A complex feature can also contain other features (*collection* of features), which allows specifying the environment in different presentation levels (lower levels are encapsulated at higher levels) (Fig.3-6).

```
<CityModel fid="Cm1456">
  <dateCreated>Feb 2000</dateCreated>
  <gml:featureMember>
    <River fid="Rv567">....</River>
  </gml:featureMember>
  <gml:featureMember>
    <Museum fid="Ms568">....</Museum>
  </gml:featureMember>
  <gml:featureMember>
    <Road fid="Rd812">....</Road>
  </gml:featureMember>
…
</CityModel>
```

Fig.3-6: Example of the Complex Feature (taken from [48])

Association between features

Relations between features are encoded using *associations*. For example, relation "Building belongs to Road Element" is encoded as association between the building and the road (Fig.3-7).

```
<School ID="145l">
  <description>Balmoral Middle School</description>
  <NumFloors>3</NumFloors>
  …
<belongsTo xlink:type="simple" xlink:href="#1234"/>
</School>

<RoadElement ID="1234">
  <description>Road Element 1234</description>
  <NumLanes>2>/NumLanes>
…
</RoadElement>
```

Fig.3-7: Example of an Association between Features

Conclusion

GML standardises only basic schemas to encode spatial data, so an application is supposed to be aware of exact notation, used to describe features and attributes. For example, one GML-derived standard can specify to store number of road lanes in "NumLanes" attribute, while another will use "size". The same is also applicable to tags, used to define the element itself and therefore makes the applications incompatible with different GML-sources.

The reviewed environment models are very close to each other:

- Both specify the environment as collection of topological elements (Road, Museum, Hotel, Restaurant, Ferry Connection, etc.).
- Elements in both models represent the environment at different presentation levels (Road→Road Element, City Area→Building, etc.).
- Geometry of elements is specified, using common geometrical primitives (Point, Line, Polygon, etc.).
- Elements contain additional, feature-specific data (opening time of museum, maximum speed, which is allowed on the road, etc.).
- Relations between elements are specified (Building belongs to Road, Prohibited Manoeuvre from Road Element$_1$ to Road Element$_2$).

This similarity will be used later to create a combination of two models.

3.2.2 Requirements to the Environment Model

The purpose of the Environment Model is to store topological elements of the movement area, which constrain the motion and serve as a destination points for the movement process.

In order to make the mobility model more realistic, a detailed Environment Model is required. But the detailed Environment Model requires high preparation overhead, so existing data sources (in GDF or GML format) have to be reused. To achieve this, the Environment Model has to be built on the top of existing environment models, provide the same functionality and include:

- Description of features of the area at different presentation levels,
- Geometry of elements,
- Attributes of elements,
- Relationships between elements.

3.2.3. Description of the Environment Model

Features

The Environment Model includes elements having different degrees of generality. For example, area contains two roads and a living area. Roads are composed of a number of road elements; living area is composed of a number of buildings, etc. (Fig.3-8)

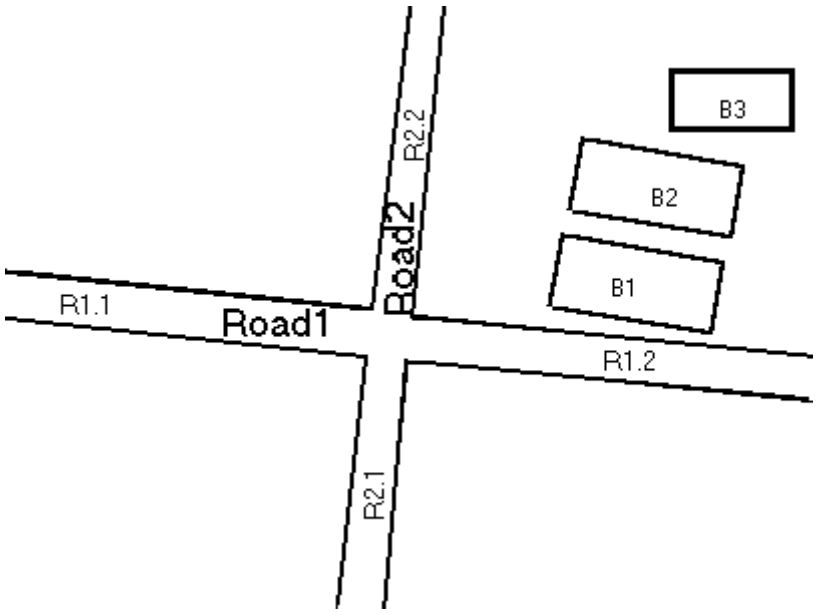


Fig.3-8: Example of an Area

This could be represented with the following hierarchy of elements (Fig.3-9). Different levels of representation can be constructed from elements of corresponding levels. For example, the basic level contains simple elements, such as road elements, buildings, junctions, etc. The next level is composed of more generic elements, such as road or living area. Elements at the higher level encapsulate elements from the lower levels (Fig.3-10).

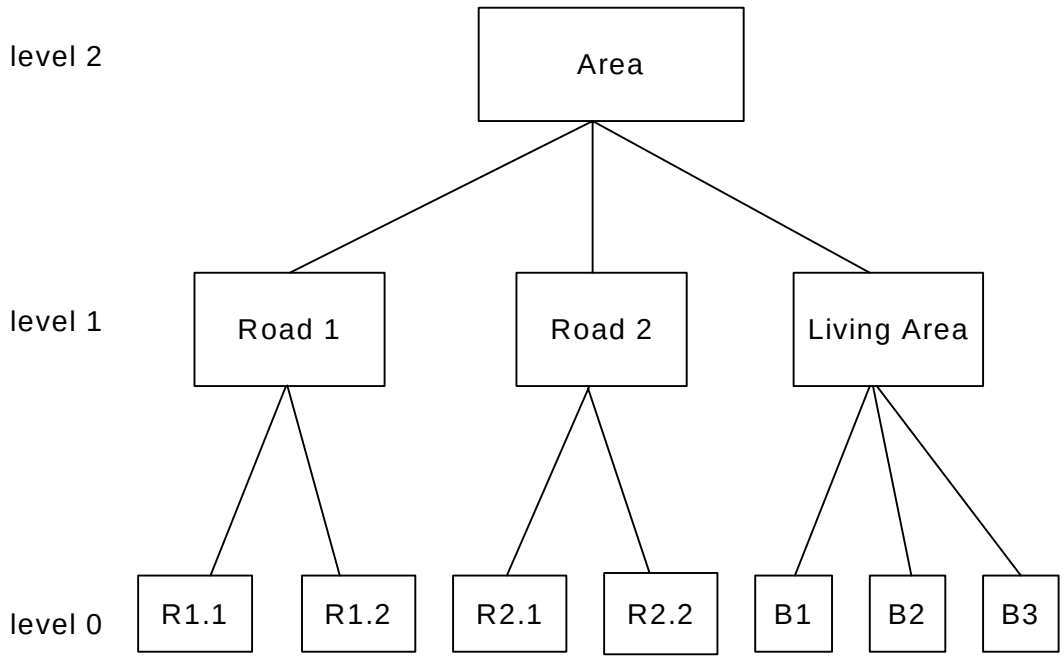


Fig.3-9: Hierarchy of Elements of the Area

Geometry

Geometry of features in the Environment Model is described through commonly used geometrical primitives, such as Point, Line, Polygon, etc. The coordinates of points are specified using World Geodetic System.

Properties and Relations

Every feature has its properties and participates in relations with the other elements.

Feature-specific properties (attributes) and relations between elements are represented as key-value pairs (Fig.3-11).

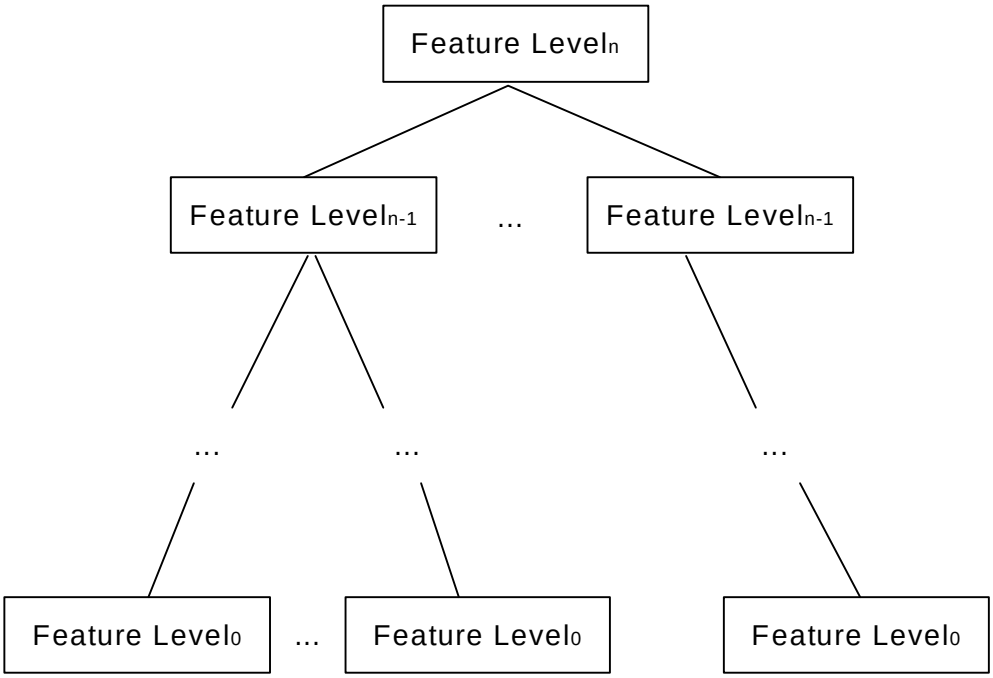


Fig.3-10: Encapsulation of Elements of Low Levels at High Levels

Road

id=”1234”
“type”=”freeway”
“Number of Lanes”=”2”
“Maximum Speed”=”120”
“Intersects”=”1235, 1236, 1237”
“Prohibited Manoeuvre”=”1235, 1236”

Fig.3-11: Description of Road with Properties and Relations

But defining the attributes and relations with key-value pairs arises the following problem: a user of the model upon making queries has to be familiar with the possible key names and corresponding values. For example, he has to be aware that speed limitation can be obtained with “Maximum Speed” key, neither with “Speed Limit” nor with “Speed Limitation”.

Standardisation of possible keys and values of properties and attributes is required.

This problem becomes more significant upon combining the GDF and GML-derived approaches. For example, the “Road Element” in GML and GDF is described differently (Fig.3-12).

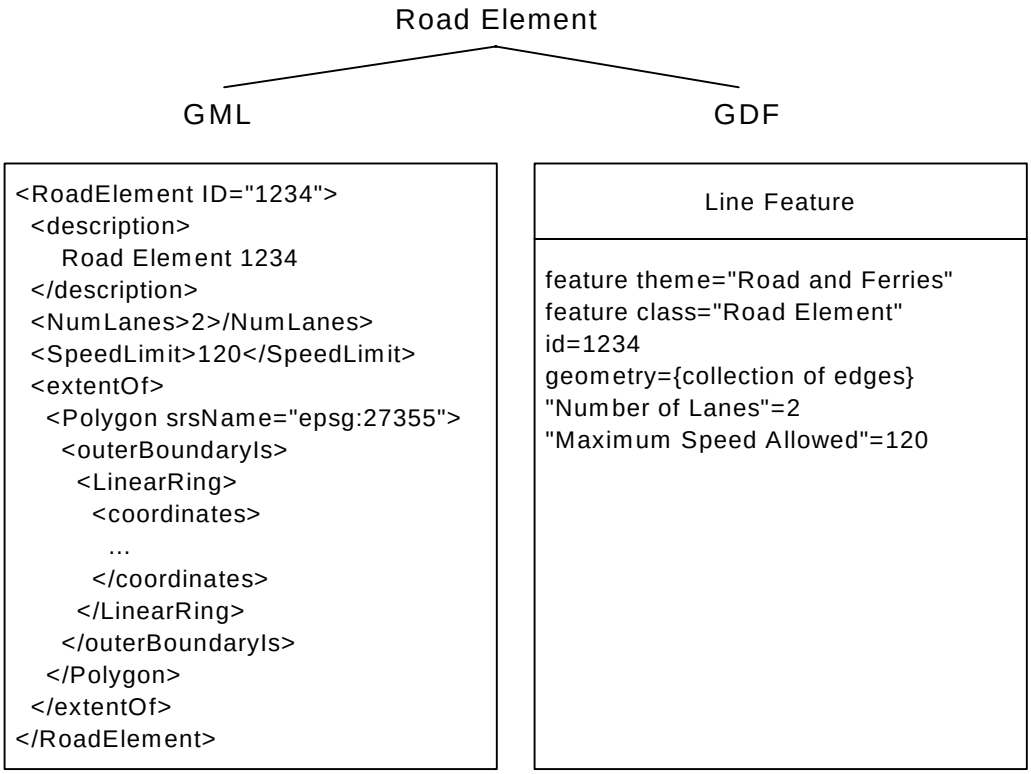


Fig.3-12: Comparison of Description of “Road Element” in GDF and GML

This example shows, that in spite of the generality, the models use different naming conventions. For example, in GML the speed limit on the road can be defined with “SpeedLimit” property, while GDF uses “Maximum Speed Allowed”. Moreover, GML does not define feature-specific attributes - they are standardised in GML-derived specifications and the various specifications use different naming conventions. In order to use the Environment Model, component, which performs queries, has to be familiar with the naming conventions used to describe a feature.

Because GDF explicitly defines all possible names for the features and their properties, GDF was chosen as a primary standard for the Environment Model. It means that (Fig.3-13):

- Components, which perform queries, have to use GDF-naming to obtain a specific feature from the model or its property.

- Upon the loading from the source, which does not use GDF-naming conventions, all the naming is converted into the GDF format.

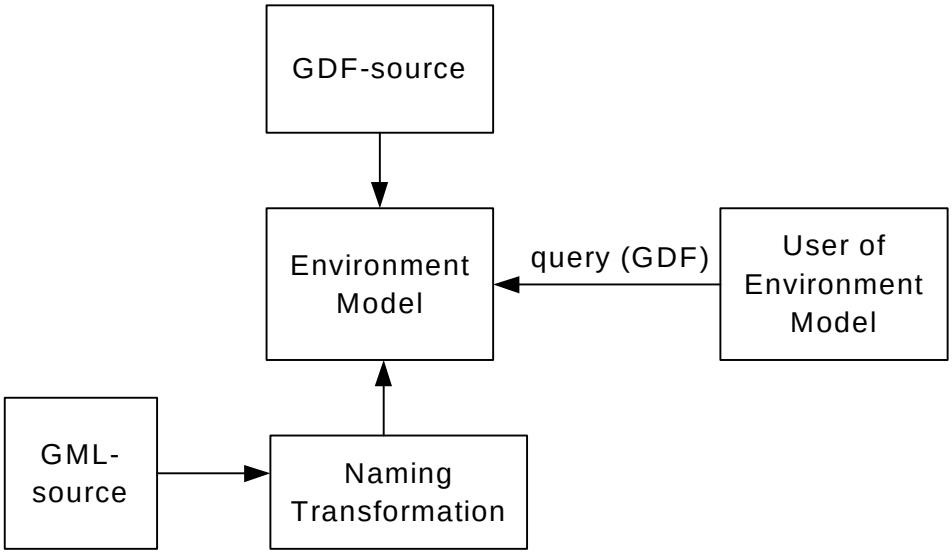


Fig.3-13: Sources of Environment Model and Naming Translation

3.2.4 Summary of the Environment Model

The Environment Model was created as a combination of the GDF and GML models, which allows reusing the existing data sources.

Model stores the topological elements at different presentation levels (City→Area→Zone→Street). Elements on higher levels encapsulate elements from lower levels. Geometry of elements is described using geometrical primitives (Point, Line, Area). Additional feature-specific properties and relations between elements are specified as key-value pairs, using GDF as the basic standard for key/value naming conventions.

3.3 Trip Model

For more realistic mobility modelling it is important to consider that people neither move with a goal to make a certain number of steps in different directions nor move between randomly chosen points of area. Movement of persons has high value of repeatability, and in order to reflect it individual trip planning decisions have to be included in simulation. This section presents several approaches to include trips into the simulation: Local Trip Model, Global Trip Model, Combined Trip Model, and Location-Probabilistic Trip Model.

3.3.1 Activity-Based Approach to Trip Modelling

Activity-Based Travel Demand Modelling approach defines travel through demand to participate in *activities*:

"No one would think about how many trips to make when developing a plan for a day; rather, one would think about what she wants to or needs to do, where the activities can or need be engaged, and, only then, would think about how to visit these places. Importantly, how many trips will be made depends on how the visits to different places are sequenced and combined into trip chains" [45].

Movement is described with a *trip chain* (Fig.3-14). The trip chain is a sequence of actions to be executed; execution of an action requires person to change location (for example, shopping requires going to shops, lunch requires going to restaurants, etc.). After finishing current activity, the next activity is chosen and new movement is initiated. The movement takes place because of switching between activities. The approach puts primacy of trip purpose over travel itself.

Input data for the model is collected upon trip surveys. Results of the survey are recorded in *Activity Transition matrix* (Table 3-1) and *Activity Duration matrix* (Table 3-2).

Activity Transition matrix stores *transitions* between activities for each person depending on *time of the day*. At different time of day unlike changes between activities are possible. For example, after work, at 12:00 a person is likely to go to lunch, but at 17:00 he is more likely to go home.

Activity Duration matrix stores the information about duration of person's activities at certain time period. At various time periods, the same activity can last for different amount of time. For example, a lunch in a restaurant at 12:00 can take 30 minutes, but after 19:00 it might take 3 hours.

Person	Time period	Current Activity	Next Activity
1	<8:00	1	2
1	8:00-12:00	2	3
1	12:00-13:00	3	2
1	13:00-17:00	2	4
1	17:00-22:00	4	1
2	<9:00	10	11
…	…	…	…

Table 3-1: Example of Activity Transition Matrix

Person	Time period	Activity	Location	Duration
1	0:00-24:00	1	(523, 345)	10-12 h
1	8:00-12:00	2	(234, 455)	3:30-4:00
1	12:00-18:00	2	(234, 455)	4:00-6:00
1	11:00-13:00	3	(454, 667)	0:30-1:00
1	16:00-24:00	3	(333, 555)	3:00-4:00
…	…	…	…	…

Table 3-2: Example of Activity Duration Matrix

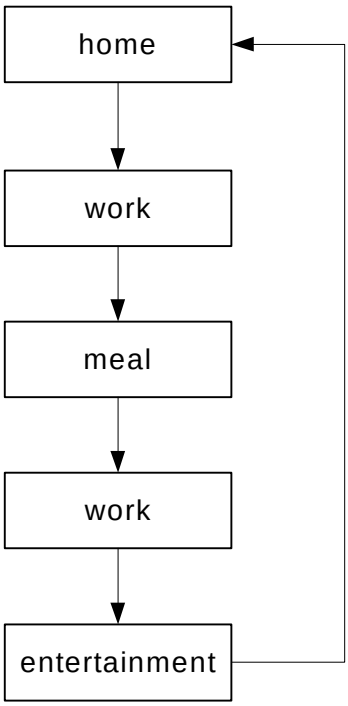


Fig.3-14: Example of Activities Sequence

The above-mentioned activity matrices are defined for one specific day only. But person's behaviour can vary in different days. In such a case, *aggregation* of activities is used (Fig. 3-15).

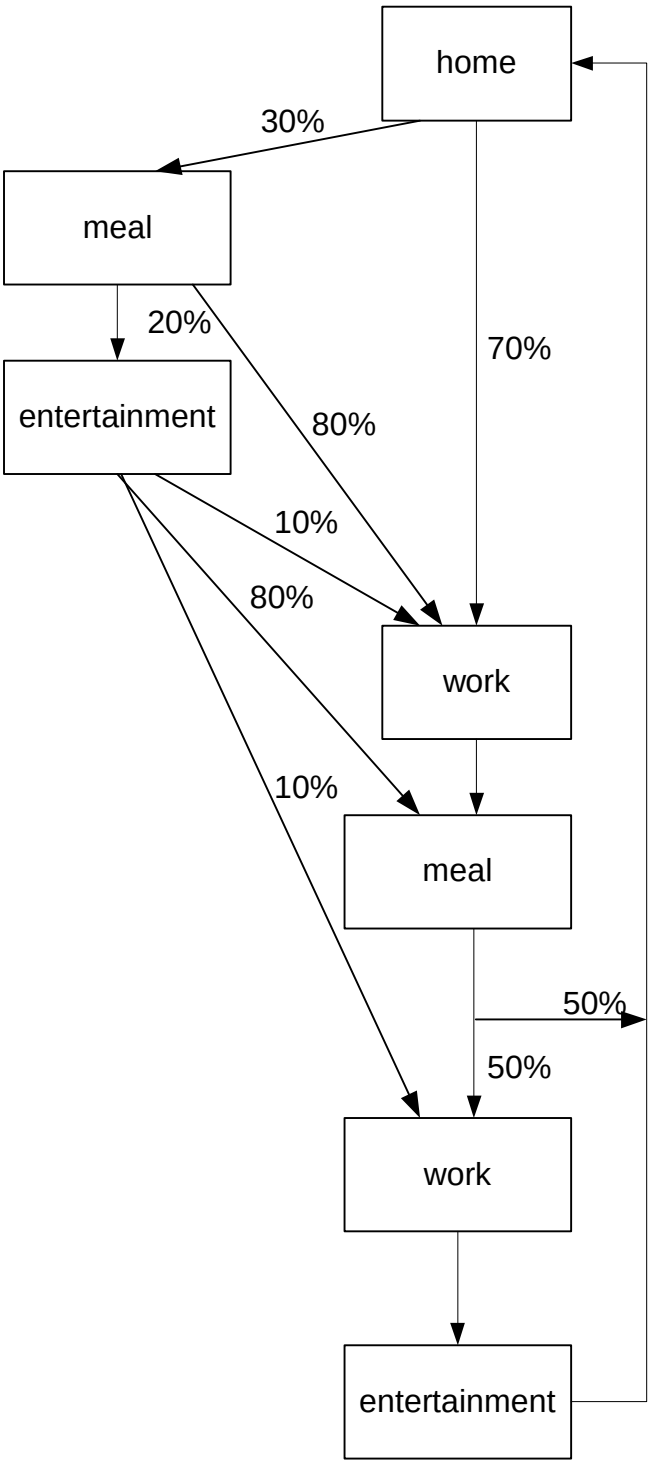


Fig.3-15: Example of Aggregated Activities Sequence

The aggregated activity sequence summarises several results and the next activity is chosen from a set of alternatives with a certain probability. New *Aggregated Activity Duration matrix* (Table 3-3) and *Aggregated Activity Transition matrix* (Table 3-4) contain a new column – probability of switching to particular activity between states, which is 100% and therefore is omitted for a non-aggregated matrix.

Person	Time period	Current Activity	Next Activity	Probability
1	<9:00	1	2	0.7
1	<9:00	1	3	0.3
1	8:00-12:00	3	2	0.8
1	8:00-12:00	3	4	0.2
…	…	…	…	…

Table 3-3: Example of Aggregated Activity Transition Matrix

Person	Time period	Activity	Location	Duration	Cumulative Probability
1	<9:00	1	(523, 345)	10-12 h	1.0
1	8:00-12:00	2	(234, 455)	(234, 455)	0.8
…	…	…	…	…	…

Table 3-4: Example of Aggregated Activity Duration Matrix

The Activity-Based Travel Demand concept will be used later to add a *purpose of movement* to the mobility model.

3.3.2 Automata of Activity Sequence

Typical sequence of activities is defined with a set of vertices, which identify activities to be executed, and transitions between them. Such a representation is similar to *Automata* representation (Fig.3-16).

Each activity has an associated set of possible locations, where the action can be executed. For example, a person can have lunch in one of his favourite restaurants; educational activity can be executed at a university, school, or academy, which is also depends on individual's behaviour. To determine possible locations, where a particular activity can be executed, semantic meaning of elements of the Environment Model can be used.

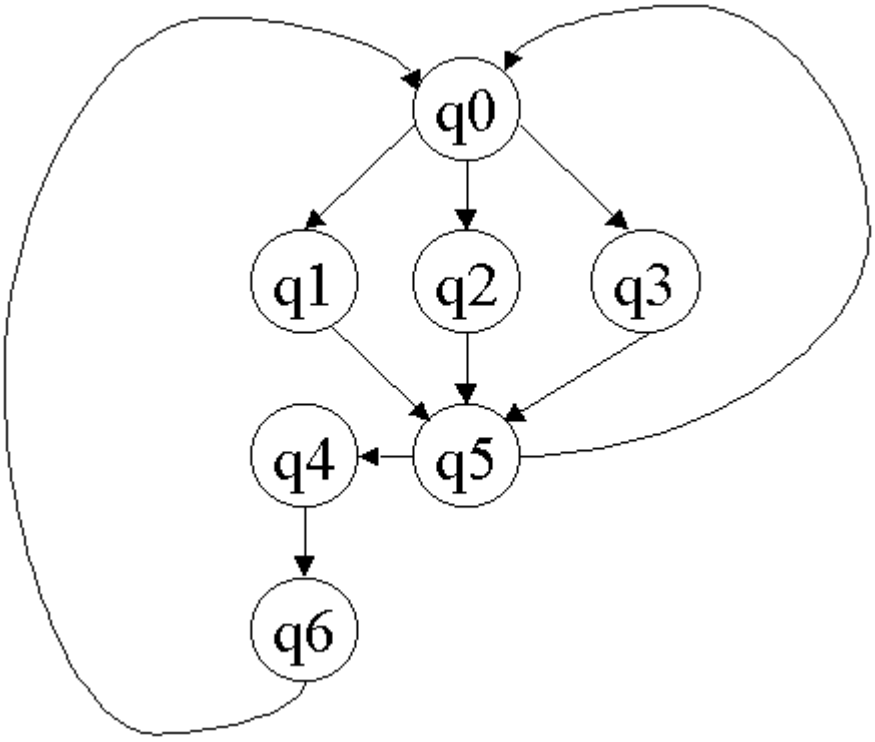


Fig.3-16: Example of Automaton of Activity Sequence

The Environment Model is a *static* model, which contains all topological elements of the model, while Trip Model is *dynamic* model, because upon execution of different activities different semantic elements of Environment Model can be used.

Classical automaton has the following parameters:

- Set of states Q
- Input alphabet Σ
- Transition function δ
- Initial state q_0

In order to describe an activity sequence with the automaton, the above-mentioned parameters have to be defined. The set of states Q contains activities to be executed. Initial state q_0 also belongs to the set as well. Each state $q_i \in Q$ contains an associated set of locations, where the activity can be executed, with preference of one location to another, and duration of activity's execution:

$$q_i \rightarrow \begin{Bmatrix} loc_1 & p(loc_1) \\ loc_2 & p(loc_2) \\ \dots & \\ loc_n & p(loc_n) \end{Bmatrix} \qquad \sum_{k=1}^{n} p(loc_k) = 1$$

Assume, that the input alphabet Σ contains a single element α with a value, which is uniformly distributed between 0 and 1.

Transition function δ has to choose the next state from one or more neighbouring states, where transition from current state is possible, depending on input value and taking into account the preference of one state to another.

Lets assume, that each transition from one state in source activity sequence has a certain *weight* value p, which defines preference of switching to destination state from current, and sum of all weights of all transitions from the same state is equal to 1:

$$0 \le p_{q_i q_j} \le 1$$
$$\sum_{k=1}^{|Q|} p_{q_i q_k} = 1$$

Transition function $\delta(q_i, a)$ from the state q_i to next state q_j can be based on the comparison of input value with weights of all transitions from the current state.

Let q_i be the current state, $q_i \in Q$, and Q' is a set of all states, which are reachable from q_i (the transition from q_i is possible and associated weight value is greater than 0):

$$Q' = \{r_0, r_1 \dots r_m\}$$
$$p_{q_i r_l} > 0, \quad \sum_{l=0}^{m} p_{q_i r_l} = 1$$

Then, particular transitions can be performed as:

$q_j = r_0$, when $0 \le a \le p_{q_i r_0}$
$q_j = r_1$, when $p_{q_i r_0} < a \le p_{q_i r_1}$
…
$q_j = r_m$, when $p_{q_i r_{m-1}} < a \le p_{q_i r_m}$

Summary:

$$q_j = \delta(q_i, a) = r_l, \qquad \begin{cases} l = 0 : 0 \le a \le p_{q_i r_l} \\ l > 0 : 0 < a - \sum_{s=0}^{l-1} p_{q_i r_s} \le p_{q_i r_l} \end{cases}$$

This automata approach will be used to describe different trip models.

3.3.3 Overview of Trip Models

The activity sequences can be defined for different number of nodes and therefore could be classified into:

- Local Trip Model
- Global Trip Model
- Combined Model

The Location-Probabilistic Trip Model can be used as a mathematical expression of these models.

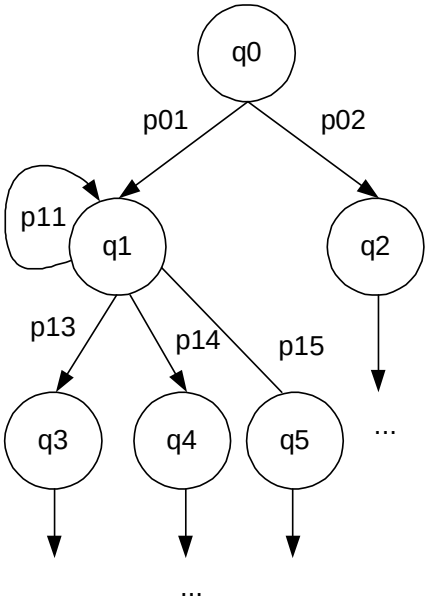
Local Trip Model

In the Local Trip Model each person has its own trip plan for a day and its own preferable locations for execution of activities, and therefore, gets its own automaton of activity sequences (Fig.3-17).

The advantage of such approach is correspondence to reality, when trip decisions of every person are simulated independently, but the model requires high preparation overhead (a separate automaton has to be defined for every node) and increases the resource requirements for a simulation (activity sequences are stored independently).

Global Trip Model

In the *Global Trip Model*, all nodes share the same activity sequence, which represents typical human behaviour in the area. The destinations of various individual trips differ, because the same activity is executed at different places (for example, persons can have lunch in different restaurants). Most of related trip models, which use the activity sequence, assume the same sequences for all the nodes (for example, [9, 10]) primarily because of simplicity, smaller preparation overhead and less resource requirements to the simulation, comparatively to the Local Model, but the result trips do not fully reflect individual travel behaviour and can be primarily used to simulate the movement of persons belonging to the same group.



$p(\delta(q_i, a))$:

	q0	q1	q2	q3	q4	q5	
q0 [	0	p01	p02	0	0	0	...]
q1 [	0	p11	0	p13	p14	p15	...]

...

Fig.3-17: Automaton of Activity Sequences in Local Trip Model

Combined Approach

Combined approach is a trade-off between the local and global models: different local models are combined into one Global Model, thus introducing a trade-off between complexity of a number of local models and simplicity of one Global Model.

Upon combination, states and transition probabilities are joined together.

The set of states in result automaton contains all the states from source automats without duplicates:

$$Q = Q_1 \cup Q_2 \cup \ldots \cup Q_n - Q_1 \cap Q_2 - \ldots - Q_1 \cap Q_n - \ldots - Q_{n-1} \cap Q_n$$

The elements in result transition probability matrix contain normalised sums of transition probabilities between equal states in the source matrices:

$$p_{q_i q_j} = \frac{\sum_{k=1}^{|Q|} p_{q_i q_j (k)}}{\sum_{k=1}^{|Q|} \sum_{r=1}^{|Q'|} p_{q_i q_r (k)}}$$

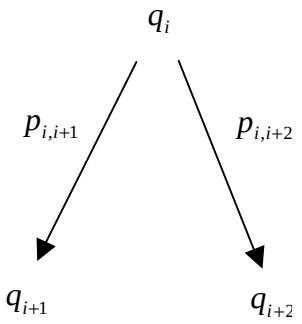
This is a reasonable combination of the above-mentioned models, because it preserves the states from the original models and associates cumulative transition probabilities.

Location-Probabilistic Trip Modelling Approach

The above-mentioned automats perform transitions between states and chose one location as the place for execution of current activity:

$$q_i \rightarrow \begin{Bmatrix} loc_1 & p(loc_1) \\ loc_2 & p(loc_2) \\ ... & \\ loc_n & p(loc_n) \end{Bmatrix} \quad : \quad loc_k, \quad p(loc_k)$$

The transitions between the states are performed with correspondence to associated transition probabilities. The state with higher transition probability can be selected more often, than state with the lower one:



$$p_{i,i+1} > p_{i,i+2} \rightarrow \quad p(q_{i+1}) > p(q_{i+2})$$

Therefore, the probability, that a particular location associated with the state q_{i+1}, will be visited, is higher, than one in q_{i+2}.

Probability, that a particular location loc_k within the state q_i will be chosen, can be determined as:

$$p(loc_k | q_i) = \begin{cases} p(q_i) * p(loc_k) & loc_k \in q_i \\ 0 & loc_k \notin q_i \end{cases}$$

Because the same location can belong to several states, cumulative probability, that the location will be selected, can be determined from:

$$r(loc_k) = \sum_{s=1}^{|Q|} p(loc_k | q_s)$$

So, automats with complex activity sequence can be approximated with simple location-probability mapping: locations within frequently chosen states have a higher probability, and are more likely to be chosen upon the simulations of trip sequences.

This model is a mathematical expression of the above-mentioned models. It neglects the activities sequences as main mobility factor and shifts the focus to locations. A certain visiting probability is assigned to every location, which determines the attractiveness of the place for a user. The advantage of the model is its simplicity, but the model can be used only in short-time simulations, because it badly reflects the daily behaviour of users.

3.3.4 Summary of Trip Models

The Trip Model adds trips to the simulation of mobility process. The trip chain is defined as a sequence of activities and represented in the model with automats. Each state in the automaton contains a set of locations, where the activity can be executed, and duration of the activity. The transitions between the states have a certain associated probability.

The automats are classified into:

- Local Trip Automaton – defines a trip sequence for a single node;
- Global Trip Automaton – defines a common trip sequence for all the nodes;
- Combined Trip Automaton – a combination between the local and global approaches and offers the variability of many local models and the simplicity of single model.

Location-probabilistic Trip Model simplifies the automats by assigning a visiting probability to a location.

These models can be used to reproduce trip sequences during the simulation.

3.4 Movement Behaviour Model

Another important part of the mobility model is the Movement Behaviour Model, which defines node's behaviour during movement between source and destination points.

Most of existing models just simplify it with constant speed motion, which doesn't fully reflect reality. For example, speed behaviour of a car dependents on behaviour of the vehicle in front of it, and certain densities of vehicles on the same road can cause traffic jams. People also demonstrate such a behaviour, but with comparatively larger densities.

Mobile nodes do not move with constant speed, but also mutually influence movement behaviour of each other. This section describes different approaches to express such behaviour and to include it into the simulation of mobility.

3.4.1 Classification of Movement Behaviour Models

Movement behaviour models can be classified according to degree of inter-node influence into:

- models, which *neglect* mutual influence between mobile nodes
- models, which *reflect* mutual influence between mobile nodes

Models, which *neglect* the mutual influence between nodes, simulate movement of a node without taking into account behaviour of its neighbours. Most of these models are described in "Related Work" section of the thesis and just approximate the motion with constant speed motion. But for some scenarios, for example, for a scenario with transport vehicles, more sophisticated movement behavioural models with reflection of the mutual influence are required.

Models, which *reflect* the mutual influence between nodes, simulate movement behaviour of a node with respect to behaviour of neighbouring nodes. Such a reflection can be included in different ways – from expressing dependency between density and average speed of mobile units to making a complex collaboration of movement vectors of adjacent nodes. Next sections propose some related approaches to implement such behaviour.

3.4.2 Fluid Traffic Model

Fluid Traffic Model [32, 33] expresses traffic as a relationship between three parameters: speed (v), density (k), and volume of traffic (q).

Assuming that all vehicles in a group have equal speeds, the volume is defined as:

$$q=kv \qquad (1),$$

from which it's clear that if density of vehicles on a road goes up, speed goes down and vice versa.

"Dynamic equation" of fluid is defined as:

$$\frac{dv}{dt} = -c^2 k^n \frac{\partial k}{\partial x} \qquad (2)$$

where:

c – nonnegative constant with dimension of speed,
n – parameter, which specifies traffic model in use:

Greenshield's (linear) Model (n = 1),
Greenberg's Model (n = -1),
Drew's Models ($n \in [-0.5;0.5]$).

The solution of (2) for v gives:

$$v = \begin{cases} \frac{ck_c^{(n+1)/2}}{n+1}(1-(\frac{k}{k_c})^{(n+1)/2}) & n \neq -1 \\ c\log\frac{k_c}{k} & n = -1 \end{cases} \qquad (3)$$

where:

k_c - "jam" density (v = 0),

Greenshield's Model is one of the mostly used models in simulations, because "it well describes the speed-density relation for relatively low densities" [32]. The solution (3) for n=1 gives:

$$v = v_0(1-\frac{k}{k_c}), \qquad (4)$$

where:

$v_0 = \frac{ck_c^{(n+1)/2}}{n+1}$ - free running speed (k = 0).

The chart of Greenshield's relationship between speed and density is shown in Fig.3-18. The speed of vehicle on a free road (k=0) is equal to free-running speed. With increase of density, speed value decreases up to zero at "jam density" k_c.

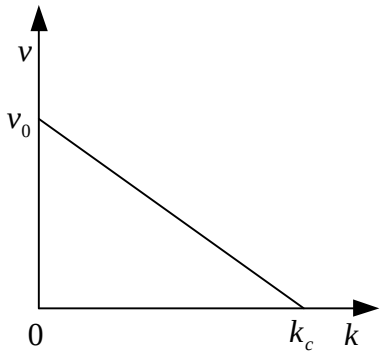


Fig.3-18: Greenshield's Relationship between the Speed of a Traffic and its Density

Advantage of the model is its simplicity, but many researchers state, that the model doesn't adequately reflect relation between density and speed, which is not linear in reality.

3.4.3 Intelligent Driver Model

"Intelligent Driver Model" [51] is based on the fact that behaviour of a vehicle is highly dependent on behaviour of vehicle in front: the drivers keep a certain distance between each other. If a car in front brakes, the succeeding vehicles will also slow down.

The distance S_α from car α to ahead-going car α-1 is defined as (Fig.3-19):

$$S_\alpha(t) = [x_{\alpha-1}(t) - x_\alpha(t) - l_\alpha] \qquad (1)$$

where:

$x_{\alpha-1}(t)$ - location of vehicle α-1 at time t,
$x_\alpha(t)$ - location of vehicle α at time t,
l_α - length of vehicle α

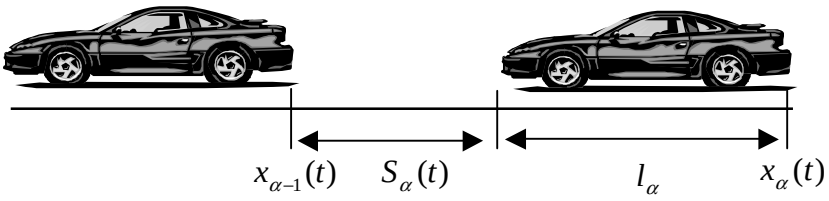


Fig.3-19: Definition of Distance between two Cars

The distance S_α has to match "effective desired distance" between the cars S^*, which is in the turn defined as sum of minimal distance and "safety" distance:

$$S^* = S_0 + \max(v_\alpha T + \frac{v_\alpha \quad \Delta v_\alpha}{2\sqrt{ab}}, 0) \qquad (2)$$

where:

S_0 - minimal, "jam" distance between cars
v_α - speed of vehicle α
T - time step of monitoring

$v_\alpha T$ - safety distance

Δv_α - velocity difference between vehicles α and α-1,
$\Delta v_\alpha = [v_\alpha(t) - v_{\alpha-1}(t)]$

a - acceleration of vehicle α

b - desired acceleration of vehicle α

Result acceleration of vehicle α is expressed as:

$$\dot{v}_\alpha = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{S^*}{S_\alpha}\right)^2\right] \qquad (3)$$

where:

v_0 - desired velocity

δ - exponent, which defines acceleration behaviour

On a free road ($S_\alpha \to \infty$): $v_\alpha \to v_0$ and $\dot{v} \to 0$ – the vehicle moves with the desired speed. If distance to ahead-going car S_α increases, then acceleration decreases and can be negative ("braking" behaviour).

The model better describes relationship between vehicles behaviour, but is more difficult to implement and includes many time-consuming arithmetical operations (square root, power, division), which have to be executed for every node on every timestep in order to recalculate its movement behaviour, that will impact the total time of simulation.

3.4.4 Boids Model

The Boids Model ([35-38]) is based on an assumption, that movement vector of a mobile host is the result of collaboration of small movement vectors, which represent atomic steering behaviours. For example, group motion is a combination of the following behaviours (Fig.3-20):

- Separation: keep a certain distance from local neighbours,
- Alignment: align the movement with movement of flock neighbours,
- Cohesion: steer to average position of local neighbours.

The same idea could be used to obtain node's motion vector with respect to behaviour of neighbouring nodes: separation, alignment and cohesion together will pretend nodes from colliding and arrange their speeds.

In [39] additional steering behaviours, which can be included into the Boids Model, are described:

- seek – to steer a character towards a specific static position in global space;
- flee – inverse of seek, velocity of a character is aligned away from the target;
- pursuit – the same, as seek, but target is moving;
- evasion – inverse of pursuit, a character tries to steer away from predicted position of target character;
- offset pursuit – a character tries to steer on path, which lies near the target;
- arrival – the same, as seek, but instead of moving through the target at the full speed, a character slows down as it approaches to destination;
- obstacle avoidance – a character tries to avoid collisions with obstacles, which lie directly in front of it;
- wander – type of random steering, but with small displacements from current steering direction, thus producing smooth motion;
- path following – to steer along a predetermined path;
- wall following – to approach and to steer along the "wall", keeping certain distance from it;
- leader following – one or more characters follow the other character ("leader").

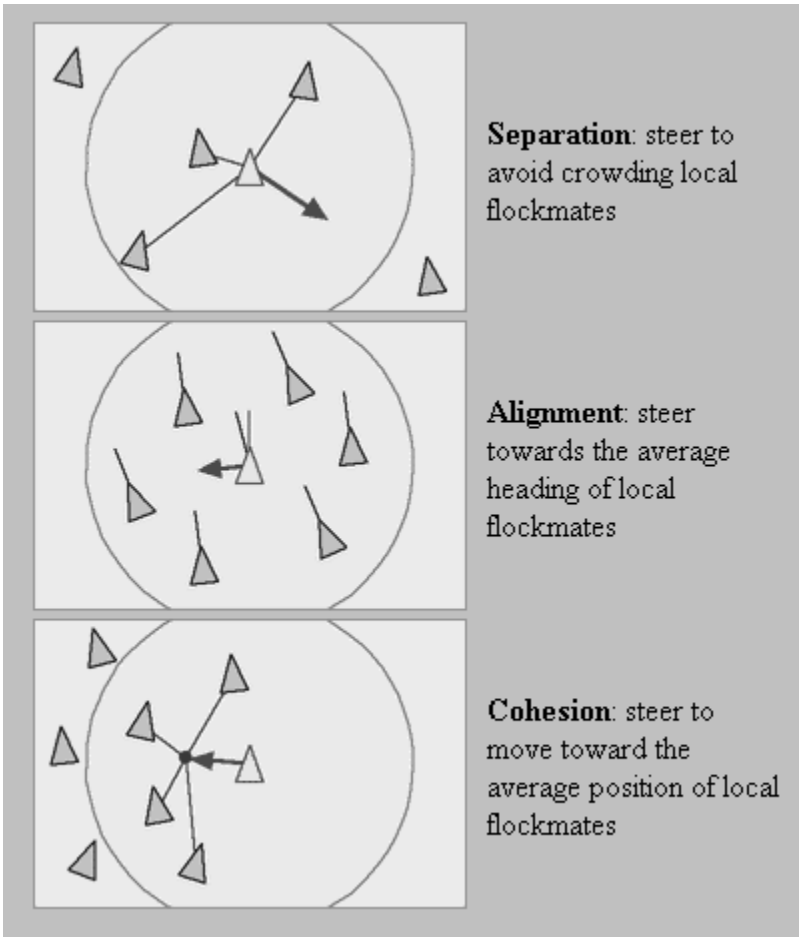


Fig.3-20: Atomic Steering Behaviours, Leading to Group Movement (taken from [36])

These behaviours can be added to the above-mentioned combination of behaviours to extend the model. For example, *seek* and *arrival* behaviours may be used to move a node to destination, *obstacle avoidance* and *path following* may be used to produce environment-constrained movement, etc.

Possibility of combining different atomic steering behaviours makes the Boids Model very powerful to produce movement in various scenarios.

3.4.5 Summary of Movement Behaviour Models

The actual movement of mobile nodes (movement behaviour) between source and destination can be simulated in different ways – from simple constant speed motion to more complex models, which are based on collaboration between neighbouring nodes. Collaborative models are more complex and require additional simulation resources, but are capable to produce more realistic movement behaviour, comparatively to the simple models. Three models belonging to this class were described: Fluid Traffic Model, which is based on relation between the speed and density, "Intelligent Driver Model", which is based on keeping a certain distance between vehicles, and Boids Model, which is based on aggregation of atomic steering behaviours.

3.5 Conception Summary

The proposed User-Oriented Mobility Model is result of integration of several models (Fig.3-21).

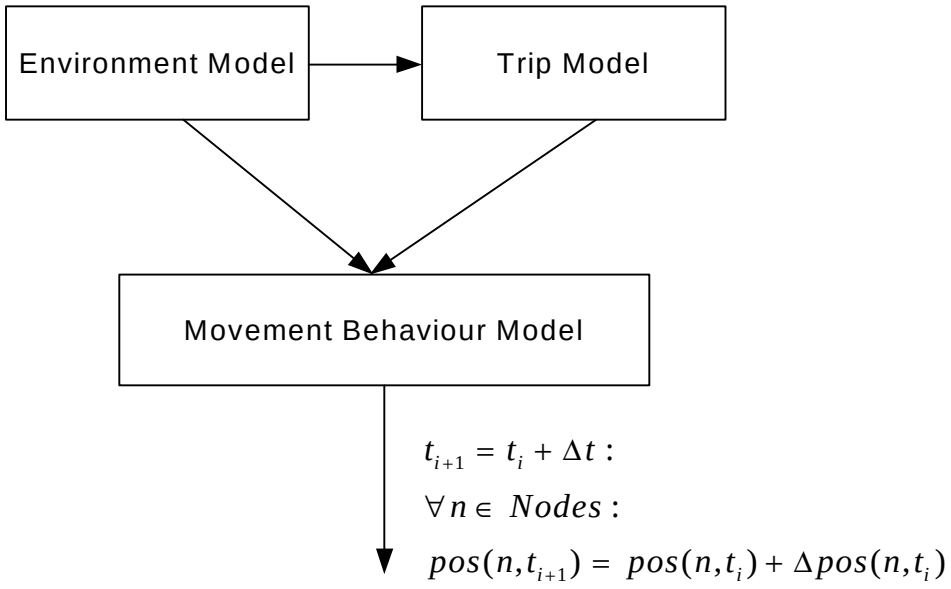


Fig.3-21: Structure of proposed User-Oriented Mobility Model

These sub-models are *macro parameters* of the mobility model. By varying these components, the model can be tuned to simulate movement in specific environment. Scenario under evaluation defines choice of this or that particular sub-model. For example, to simulate movement of cars in “City Scenario”, the Environment Model should contain road network, the Trip Model should produce trips that are likely to be made by the cars in this area and the Movement Behaviour Model should reflect mutual influence of nodes using the Fluid Traffic Model or the Intelligent Driver Model. To simulate movement of pedestrians in a park, the Environment Model should hold pedestrians’ movement area, the Trip Model should construct common trips and the Movement Behaviour Model can be simplified with a low speed motion with a set of preferable speeds.

4 Implementation of User-Oriented Mobility Model

This chapter explains implementation details of the mobility model for CANU Simulation Environment (CANUSim) [54]. As mentioned before, the model is a combination of the Environment, Trip and Movement Behaviour Models, so implementation details of each of these models will be described in order to show the mechanism of their integration.

This chapter is structured as follows. Section 4.1 describes API, offered by the CANUSim to include custom mobility model into simulation. The Environment, Trip and Movement Behaviour models are described in Sections 4.2, 4.3, 4.4 correspondingly. Finally, Section 4.5 concludes the chapter.

4.1 CANUSim Mobility API

CANUSim is a simulation environment for MANET. It performs simulation with certain discrete interval ("simulation time-step"). The scenario for simulation is defined in an XML-file. The simulator is split into packages; for mobility models two packages are important:

- `core` – contains core classes of the simulator
- `movement` – contains base classes for custom mobility models implementations

The core of the simulator is `Universe` object (`sim.core.Universe`), which aggregates other objects and is responsible for the simulation. Mobile node in the CANUSim is represented by instance of `Node` class (`sim.core.Node`). For mobility modelling `position` property of `Node` is important, which defines its current location in the simulation area. Every `Node` object has an associated set of *extension modules*. The extension module gets control on every simulation time step and performs required operations. For example, *Communication* extension initiates sending of packets to other nodes, etc. Movement component is implemented as such an extension module and performs changing of the node's position on every time step according to mobility model. For example, the Random Waypoint Mobility Model makes it according to formula:

$$pos_{t+1} = pos_t + \vec{V} * \Delta t$$

where:

pos_t - location of node at time t
$\vec{V}$ - node's movement vector
Δt - duration of single time step

Because the simulation is based on time-steps, mobility model has to consider the duration of time steps upon calculating node's position at the next time step.

`Movement` (`sim.movement.Movement`) is the base class for all the implementations of mobility models and has the following properties and methods:

- `minSpeed` – minimal speed of the movement
- `maxSpeed` – maximal speed of the movement
- `initialize()` – derived classes should override it to perform custom initialisation (the method will be called after finishing processing scenario file and before starting the simulation)
- `act()` – derived classes should override it to perform custom operations on every time step (for example, changing of node's position, if necessary)
- `load()` – derived classes should override it to perform custom processing of XML-source

The `minSpeed` and `maxSpeed` properties have to be initialised according to expectations of minimal and maximal speed values since the simulator relies on them upon calculation of internal state variables.

The CANUSim mobility components are summarised in Fig.4-1.

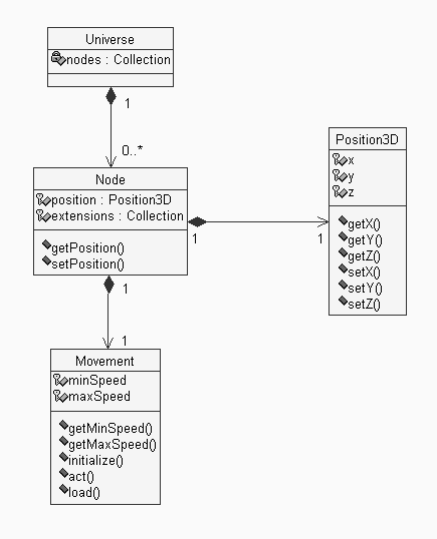


Fig.4-1: CANUSim Mobility Components

4.2 Implementation of the Environment Model

The Environment Model is implemented to conform GDF and GML standards of environment representation in digital form. The implementation is split into two parts: model itself (`gid` package) and reader of GDF files (`gdfreader` package) (Fig.4-2). Parser of GML data sources can be implemented in similar manner.

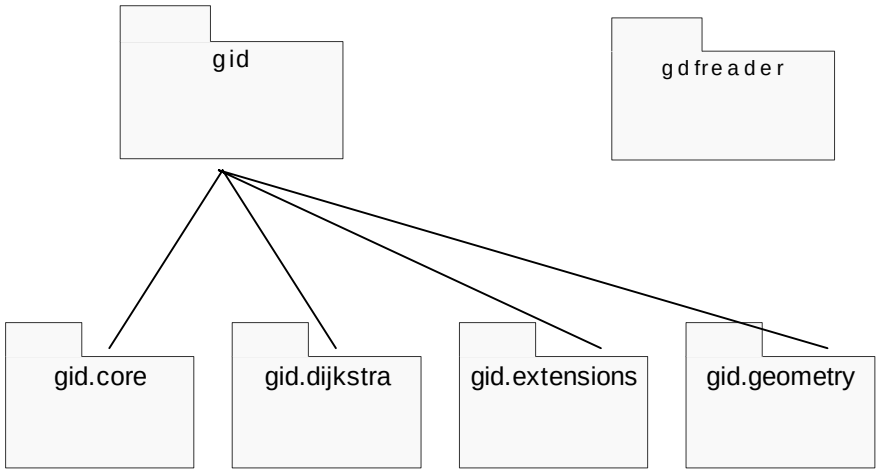


Fig.4-2: Structure of Implementation of the Environment Model

4.2.1 gid package

`gid` package is split into following sub-packages:

- `gid.core` – contains core elements of the model
- `gid.dijkstra` – contains implementation of Dijkstra planar graph traversal algorithm, reflecting movement area of cars
- `gid.extensions` – contains additional modules
- `gid.geometry` – contains geometrical primitives

gid.core package

Core element of implementation of the Environment Model is `GID` object (`gid.core.GID`), which contains a collection of elements of the environment (such as road, restaurants, museums, etc.). Single element of the environment is represented by instance of `GIDElement` class (`gid.core.GIDElement`).

The most important methods and properties of `GID` are:

- `elements` – collection of elements of the environment
- `graph` – graph of movement area, constructed from the collection of elements
- `rebuildGraph()` – rebuilds graph of movement area
- `mapEdgeToElement()` – maps edge of the graph to associated `GIDElement`

To ease calculation of routes between points, the `GID` constructs graph of the movement area. In the graph vertices represent movement attraction points of the area (restaurants, museums) and edges represent road connections between them, so classical planar graph traversal algorithms can be applied to find a path between points. In spite of the fact that the movement area differs for various mobile objects (e.g. cars, pedestrians, etc.), the `GID` constructs a single graph for all types of objects. To find a path for particular mobile node, attributes of movement area have to be taken into account, in other words, edge under consideration has to be mapped to the corresponding `GIDElement` and its properties have to be inspected.

`GID` proposes an implementation of Dijkstra shortest-path algorithm to calculate routes for cars (`gid.dijkstra.Dijkstra`), reflecting one-way and closed for movement roads. For other types of mobile nodes, custom path selection algorithm has to be implemented, taking into account movement area for this class of nodes.

Single element of the environment is created by instance of `GIDElement` class with the following properties:

- `id` – element's unique identifier
- `class_code` – specifies class of the element (GDF class code)
- `subclass_code` – specifies subclass of the element (GDF subclass code)
- `geometry` – specifies geometry of the element
- `children` – collection of references to elements of lower levels
- `attributes` – map of string-to-string key-value pairs (as defined by GDF specification)
- `relations` – map of string-to-string key-value pairs (as defined by GDF specification)

gid.geometry package

Geometry of an element is defined using common geometrical primitives, such as Point, Line, Polyline (series of lines), Polygon (closed Polyline) (Fig.4-3).

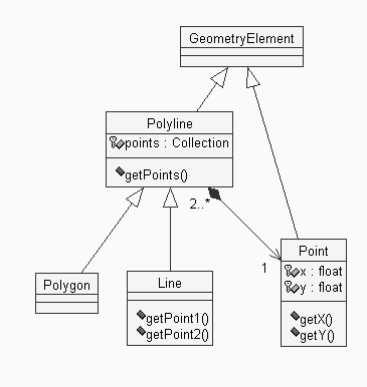


Fig.4-3: Structure of gid.geometry package

The model can be extended with new geometrical classes, deriving them from the `GeometryElement`.

The overall structure of classes, composing the Environment Model is shown in Fig.4-4.

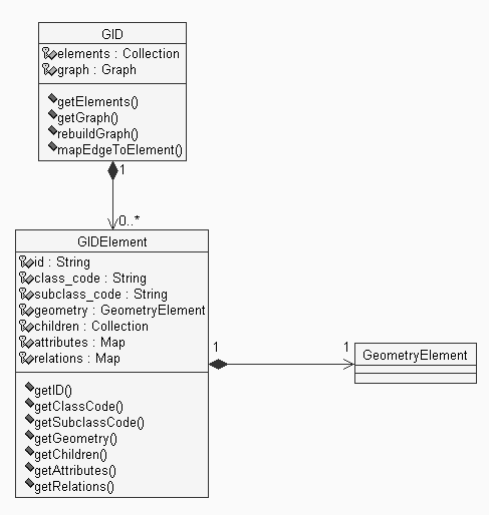


Fig.4-4: Class diagram of the Environment Model

The Environment Model can be added to the simulation with following line in simulation scenario:

<extension class="gid.core.GID"/>

There can be only one `GID` object per simulation.

4.2.2 gdfreader package

`gdfreader` package provides functionality of loading the Environment Model from GDF-source. This is done by `GDFReader` class (`gdfreader.GDFReader`) directly after its instantiation.

Processing of source file is performed in two steps:

1) Loading of elements of data structure
2) Resolving links between the elements (e.g., relationships, hierarchy of elements, etc.)

After finishing the source processing, the parser adds the elements to the Environment Model and calls `GID.rebuidGraph()` to restructure graph of movement area, in order to ease recalculation of paths between the elements.

The parser is capable to extract from source file only part of the whole area. Upon the extraction, clipping is performed (Fig.4-5):

- if an element is outside the requested part, do not process it
- if an element is within this region, add it to the model
- if an element is on border of the area, clip its geometry with limiting edges

A bound area for clipping is defined with rectangle. The bounding area can be omitted; in this case the whole area will be loaded.

The `GDFReader` has to be added to the simulation scenario **only after** the `GID` extension. The `GDFReader` accepts the following parameters via XML:

- `min_x`, `max_x`, `min_y`, `max_y` – define the bounding rectangle
- `param` – the name of GDF-source file

For example, a line:

```
<extension name="Reader" class="gdfreader.GDFReader"
 min_x="4000" max_x="6000" min_y="3000" max_y="4000"
 param="source.gdf" />
```

will add to the simulation `GDFReader` object and request to parse GDF-source "source.gdf", processing only the elements from rectangle (4000, 3000)-(6000, 4000).

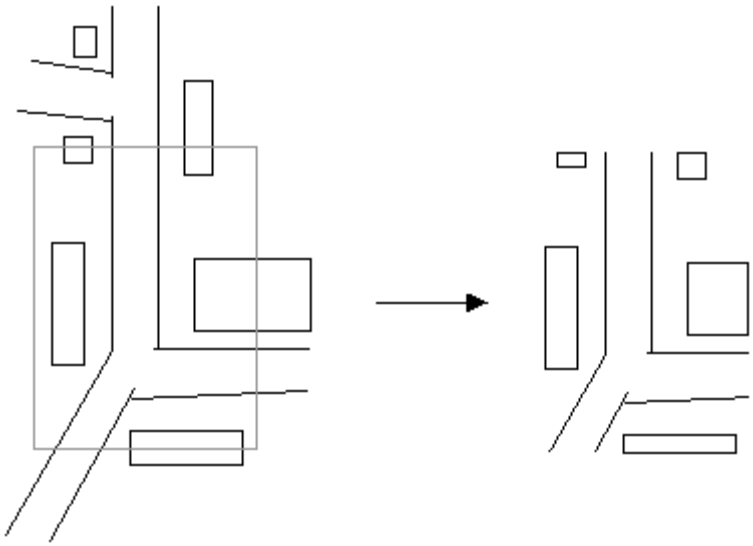


Fig.4-5: Clipping of the Environment by GDFReader

4.3 Implementation of Trip Models

Trip models are responsible for choosing a new trip for mobile node. Because trip surveys were outside of this diploma thesis, the implemented trip models produce only random trips with different degree of regularity. The following trip models are currently implemented (Fig.4-6). The implementations of trip models are sited into `uomm.generators` package.

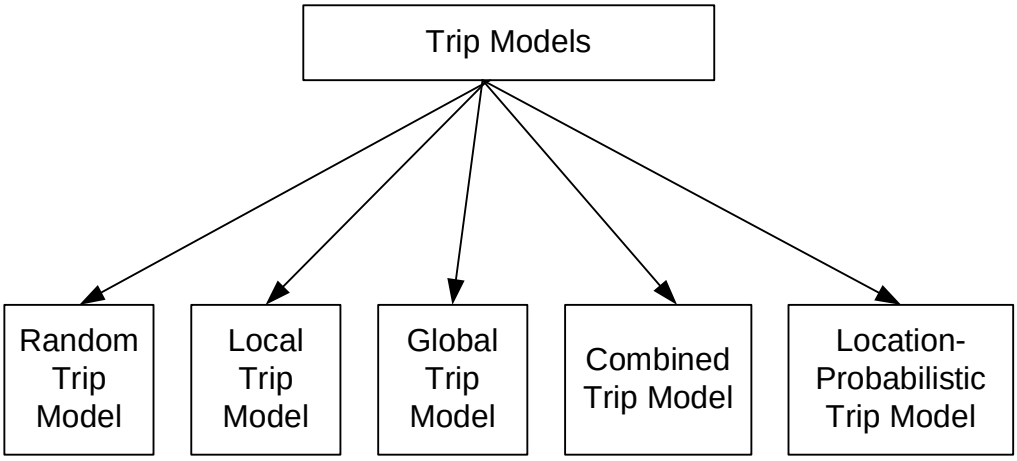


Fig.4-6: Currently implemented Trip Models

Trip Model has to implement two interfaces:

- `InitialPositionGenerator` (`uomm.core.InitialPositionGenerator`) – is used to get position for initial node's placement
- `TripGenerator` (`uomm.core.TripGenerator`) – is used to construct a trip to destination (sequence of points to pass)

Trip models use graph as a source for the initial placement and trips generation. The following position and trips generators are currently implemented:

- `RandomInitialPositionGenerator` (`uomm.generators.RandomInitialPositionGenerator`) – places a node on a random vertex of the graph (part of Random Trip Model)
- `ShortestPathTripGenerator` (`uomm.generators.ShortestPathTripGenerator`) – generates a path to randomly chosen vertex according to Dijkstra algorithm (part of Random Trip Model)
- `LocalTripGenerator` (`uomm.generators.LocalTripGenerator`) – generates a trip to destination point, that is

defined by current state of Local Trip Automaton (implementation of Local Trip Model)
- `GlobalTripGenerator` (`uomm.generators.GlobalTripGenerator`) – generates a trip to destination point, that is defined by current state of Global Trip Automaton (implementation of Global Trip Model)
- `CombinedTripGenerator` (`uomm.generators.CombinedTripGenerator`) – generates a trip to destination point, that is defined by current state of Combined Trip Automaton (implementation of Combined Trip Model)
- `LocationProbabilisticTripGenerator` (`uomm.generators.LocationProbabilisticTripGenerator`) – generates trips between the vertexes according to visiting probability (implementation of Location-Probabilistic Trip Model)

By default, paths between points are calculated using Dijkstra shortest-path algorithm from the `GID`, so only trips suitable for cars are generated. To generate trips for other kinds of mobile objects, corresponding path-searching algorithm has to be implemented and registered at the Trip Model as the main path-searching algorithm.

More complicated models (Local, Global, Combined and Location-Probabilistic trip models) add regularity of trip sequences using automatons.

Random Trip Model

Random Trip Model constructs trips between random vertices of the graph according to Dijkstra shortest-path algorithm. The model is implemented in two classes: `RandomInitialPositionGenerator` and `ShortestPathTrip Generator`. The model can be included into the simulation scenario with following XML-tags:

<extension name="PosGen"
class="uomm.generators.RandomInitialPositionGenerator"/>
<extension name="TripGen"
class="uomm.generators.ShortestPathTripGenerator"/>

Local Trip Model

The trip generator according to the Local Trip Model is implemented in `LocalTripGenerator` (`uomm.generators.LocalTripGenerator`) class. The generator creates a separate activity automaton for every node. The automatons are constructed randomly in three steps:

1) A "*pool of locations*" is created – contains points, where different activities can be executed. Size of the pool is `statePoolSize` × `locationsPerState` points.

2) A "*pool of states*" is created. Size of the pool is `statePoolSize` states. Each state defines a particular activity to be executed and includes `locationsPerState` locations, chosen from the "*pool of locations*".
3) Automata are created – every automaton contains `statesPerAutomata` states, randomly chosen from the "*pool of states*", with random transitions between them.

To improve regularity of movement within the area, the generator saves the "*pool of locations*" to external data storage and reuses it upon the successive executions. The Local Trip Model can to be included into the simulation scenario with following XML-tag:

```
<extension name="TripGen"
class="uomm.generators.LocalTripGenerator">
<poolsize>8</poolsize>
<statesize>10</statesize>
<automatasize>4</automatasize>
<filesrc>locations.pool</filesrc>
</extension>
```

The `LocalTripGenerator` accepts the following parameters:

- `poolsize` – number of states in the "*pool of states*"
- `statesize` – number of locations in each state
- `automatasize` – number of states in single automaton
- `filesrc` –name of data source for the "*pool of locations*". If the source exists, then pool is constructed from the source, otherwise it is constructed with random locations and is stored in the source.

Global Trip Model

In the Global Trip Model there is a single automaton for all the nodes; thus the model doesn't use the "*pool of states*". The trip generator according to the Global Trip Model is implemented in `GlobalTripGenerator` (`uomm.generators.GlobalTripGenerator`) class. The automaton is constructed in two steps:

1) A "*pool of locations*" is created – contains points, where activities can be executed. Size of the pool is `statesPerAutomata` × `locationsPerState` points.
2) A automaton is constructed – contains `statesPerAutomata` states with random transitions between them; each state includes `locationsPerState` locations, chosen from the "*pool of locations*".

The generator saves the "*pool of locations*" to external data storage and reuses it upon the successive executions. The Global Trip Model has to be included into the simulation scenario with following XML-tag:

```
<extension name="TripGen"
class="uomm.generators.GlobalTripGenerator">
```

```
<statesize>10</statesize>
<automatasize>4</automatasize>
<filesrc>locations.pool</filesrc>
</extension>
```

The following parameters are defined for `GlobalTripGenerator`:

- `statesize` – number of locations in each state
- `automatasize` – number of states in the automaton
- `filesrc` –name of data source for the "*pool of locations*". If the source exists, then pool is constructed from the source, otherwise it is constructed with random locations and is stored in the source.

Combined Trip Model

In the Combined Trip Model there is a single automaton for all the nodes, but it is constructed upon aggregation of individual automata. Thus it is identical to the Local Trip Model, but it requires additional step for combining the automata. The Combined Trip Model is implemented in `CombinedTripGenerator` (`uomm.generators.CombinedTripGenerator`) class. The automaton is constructed in four steps:

1) A "*pool of locations*" is created – contains `statePoolSize` × `locationsPerState` points for activities execution.
2) A "*pool of states*" is created. Size of the pool is `statePoolSize` states, each state includes `locationsPerState` locations, chosen from the "*pool of locations*".
3) Automata are created – every automaton contains `statesPerAutomata` states, randomly chosen from the "*pool of states*", with random transitions between them.
4) The automata are combined into single one according to the rules from the chapter 3.3.3.

The generator saves the "*pool of locations*" to external data storage and reuses it upon the successive executions. The `CombinedTripGenerator` can be added to the simulation with following XML-tag:

```
<extension name="TripGen"
class="uomm.generators.CombinedTripGenerator">
<poolsize>8</poolsize>
<statesize>10</statesize>
<automatasize>4</automatasize>
<filesrc>locations.pool</filesrc>
</extension>
```

The `CombinedTripGenerator` recognises following parameters:

- `poolsize` – number of states in the "*pool of states*"
- `statesize` – number of locations in each state

- `automatasize` – number of states in single automaton
- `filesrc` –name of data source for the "*pool of locations*". If the source exists, then pool is constructed from the source, otherwise it is constructed with the random locations and is stored in the source.

Location-Probabilistic Trip Model

In the Location-Probabilistic Model locations get a visiting probability according to the state, to which they belong.

This trip generator is implemented in `LocationProbabilistic TripGenerator` (`uomm.generators.LocationProbabilisticTrip Generator`) class. The probabilities are assigned in three steps:

1) A "*pool of locations*" is created – contains `statePoolSize` × `locationsPerState` points, where activities can be executed.
2) A "*pool of states*" is created. Size of the pool is `statePoolSize` states, each state includes `locationsPerState` locations, chosen from the "*pool of locations*".
3) Probabilities of locations to be visited are calculated, as described in the chapter 3.3.3.

The generator saves the "*pool of locations*" to external data storage and reuses it upon the successive executions. It can be added to the simulation with following XML-tag:

```
<extension name="TripGen"
class="uomm.generators.LocationProbabilisticTripGenerator">
<poolsize>8</poolsize>
<statesize>10</statesize>
<filesrc>locations.pool</filesrc>
</extension>
```

For the `LocationProbabilisticTripGenerator` the following parameters are defined:

- `poolsize` – number of states in the "*pool of states*"
- `statesize` – number of locations in each state
- `filesrc` –name of data source for the "*pool of locations*". If the source exists, then the pool is constructed from the source, otherwise it is constructed with the random locations and is stored in the source.

The structure of the trip models is shown in Fig.4-7.

4.4 Integration with Movement Behaviour Models

The above-described Environment Model and trip models have to be combined with the Movement Behaviour Model in order to obtain the result mobility model. As result of the integration, the Movement Behaviour Model reflects node's movement

along a path, which determined by the Trip Model, with respect to environment constraints.

The base class for implementations of movement behaviour models is `uomm.core.UserOrientedMovement`. It is derived from the base class for mobility models in the CANUSim – `sim.movements.Movement`.

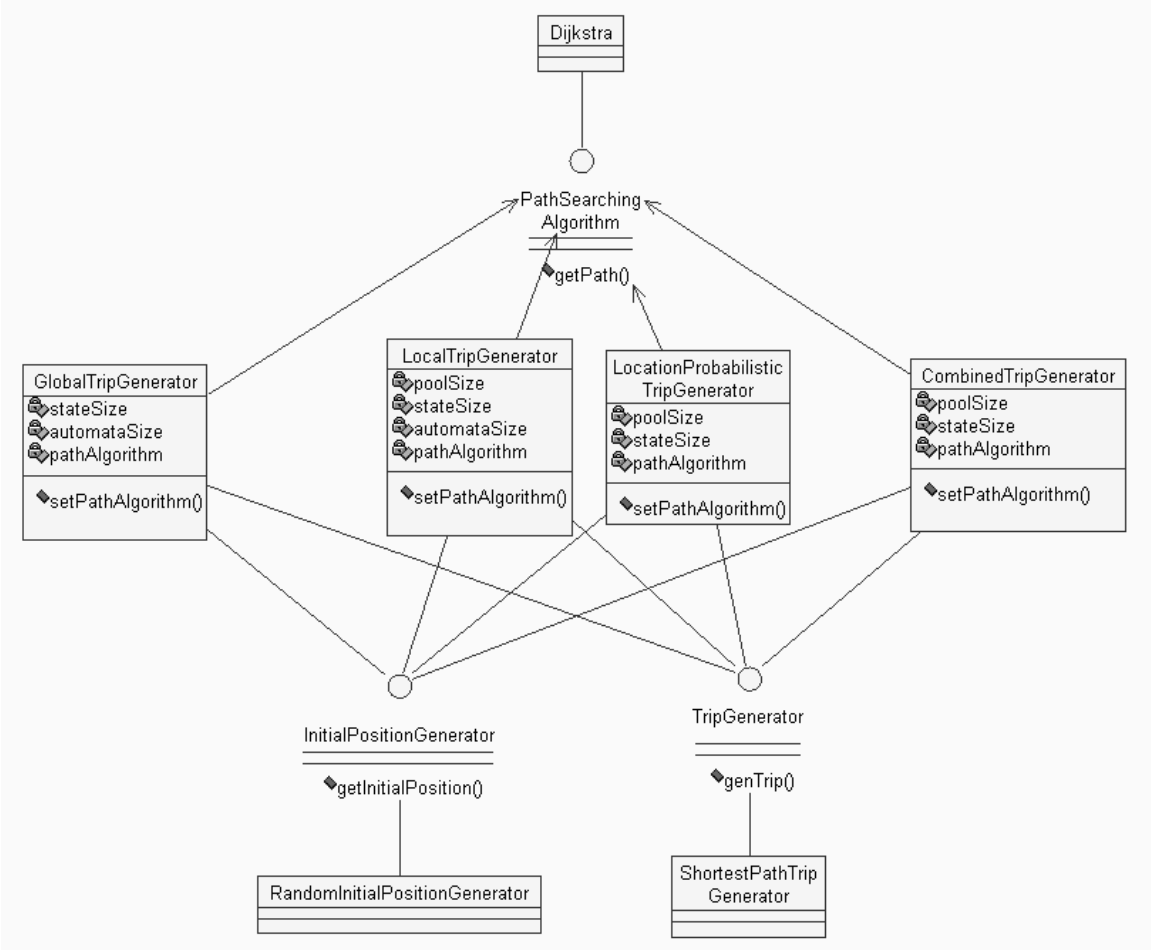


Fig.4-7: Structure of Trip Models

Class `uomm.core.UserOrientedMovement` has following significant methods and properties:

- `initialPositionGenerator` – link to object, which selects node's initial position
- `tripGenerator` – link to object, which generates trips, accomplished by the node

- `initialize()`– is called before starting the simulation to perform initialisation
- `act()` – is called on every time step to perform node's position changing

Custom implementation of the Movement Behaviour Model has to implement method `initialize()` to set a location of the node according to the `InitialPositionGenerator`. In the method `act()` it has to modify node's position according to speed behaviour and path of movement, as defined by the Trip Model. The sample implementation of the model in pseudocode is shown in Fig.4-8.

Different movement behaviour models can be implemented in this fashion. Currently only the constant-speed motion is implemented in the class `ConstantSpeedMovement` (`uomm.core.ConstantSpeedMovement`). The model simulates node's movement to a destination, as defined by the Trip Model. The movement area is approximated with edges of the graph of environment. The speed value varies between $[v_{\min}, v_{\max}]$, but during the movement along current edge it is constant. After arriving to destination, the node stays there between $\lfloor t_{p\min}, t_{p\max} \rfloor$ and then initiates a new trip.

`ConstantSpeedMovement` can be added to the list of node's extension modules with following lines:

<extension class="uomm.core.ConstantSpeedMovement"
initposgenerator="TripGen" tripgenerator="TripGen">
<minspeed>8.33</minspeed>
<maxspeed>13.88</maxspeed>
<minstay>0.0</minstay>
<maxstay>100.0</maxstay>
</extension>

For the `ConstantSpeedMovement` the following parameters are defined:

- `initposgenerator` –unique name of associated `TripGenerator`
- `initposgenerator` –unique name of associated `InitialPositionGenerator`
- `minSpeed` – minimal speed value (in m/s)
- `maxSpeed` – maximal speed value (in m/s)
- `minStay` – minimal stay duration at destination (in s)
- `maxStay` – maximal stay duration at destination (in s)

Class diagram of integration of different models into single mobility model is shown in Fig.4-9.

```
// mobile node
Node node;

// associated InitialPositionGenerator
InitialPositionGenerator initPosGen;

// associated TripGenerator
TripGenerator tripGen;

// path of the movement
Path path;

initialize()
{
  // set initial position of the node according to the InitialPositionGenerator
  node.position = initPosGen.getInitialPosition();
}

act()
{
  // select new path, if current is empty
  if (path is empty)
    chooseNewPath();

  // move the node to the next location in path
  vect = (path[0] - node.position)*speed*timeStepDuration;
  node.position.add(vect);

  // check if current location is reached
  if (node.position==path[0])
    path.remove(0);
}

chooseNewPath()
{
  if (movement in all directions is prohibited)
  {
     // we have to do something, for example, imitate “border effect”:
     // move to boundary point of the area with infinite speed
     // (imitate arrival into the simulation area)
     path[0] = boundaryPoint;
     speed = INFINITE;
  }
  else
  {
    // generate new trip
    path = tripGen.genTrip();
    speed = rand(minSpeed, maxSpeed);
  }
}
```

Fig.4-8: Sample Implementation of Movement Behaviour Model

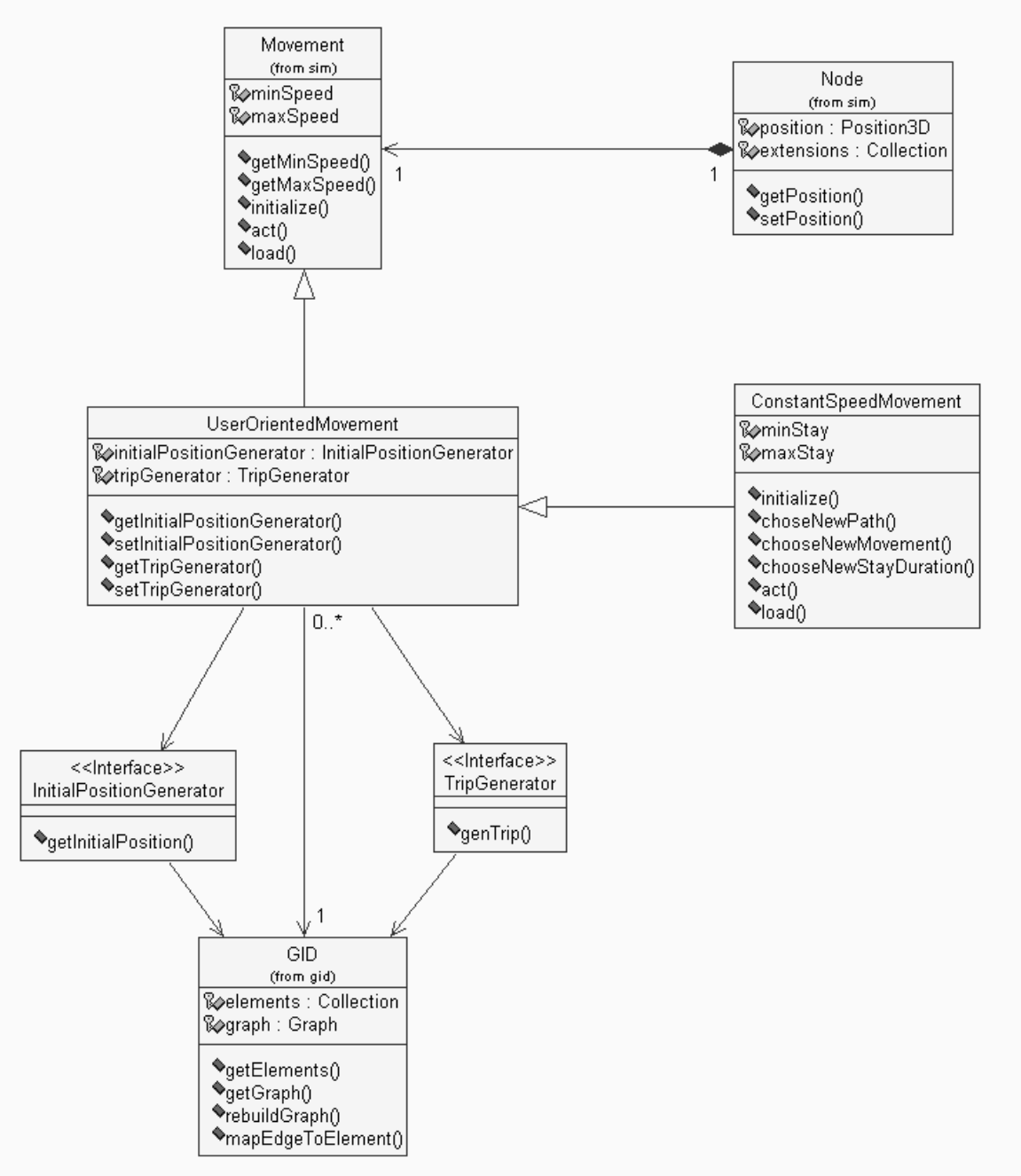


Fig.4-9: Integration of Models into the Mobility Model

4.5 Implementation Summary

This chapter presented details of implementation of the mobility model. The implementation is split into three parts:

- Implementation of Environment Model (package `gid`)
- Implementation of Trip Model (package `uomm.generators`)
- Implementation of Movement Behaviour Model (package `uomm.core`)

The Environment Model is implemented as a collection of elements of movement area. To simplify recalculation of paths between points, the model constructs a graph of the movement area. In the graph, vertices represent points of the environment, and edges represent connections between them. The trip models, implemented in the thesis, generate random trips in the simulation area, preserving certain sequences in activities execution. The Movement Behaviour Model integrates with these models by simulating movement along the path, which is determined by the Trip Model, with correspondence to environment constraints, as defined by the Environment Model.

5 Evaluation of the Mobility Model

This chapter presents results of evaluation of the User-Oriented Mobility Model to show different influences on performance of MANET, caused by simple random mobility models and more realistic models.

This chapter is structured as follows. Section 5.1 presents simulation environment and simulation scenario. Section 5.2 describes the Location-Aided Routing protocol, being chosen for the evaluations. Section 5.3 analyses the simulation results. Finally, Section 5.4 concludes the chapter.

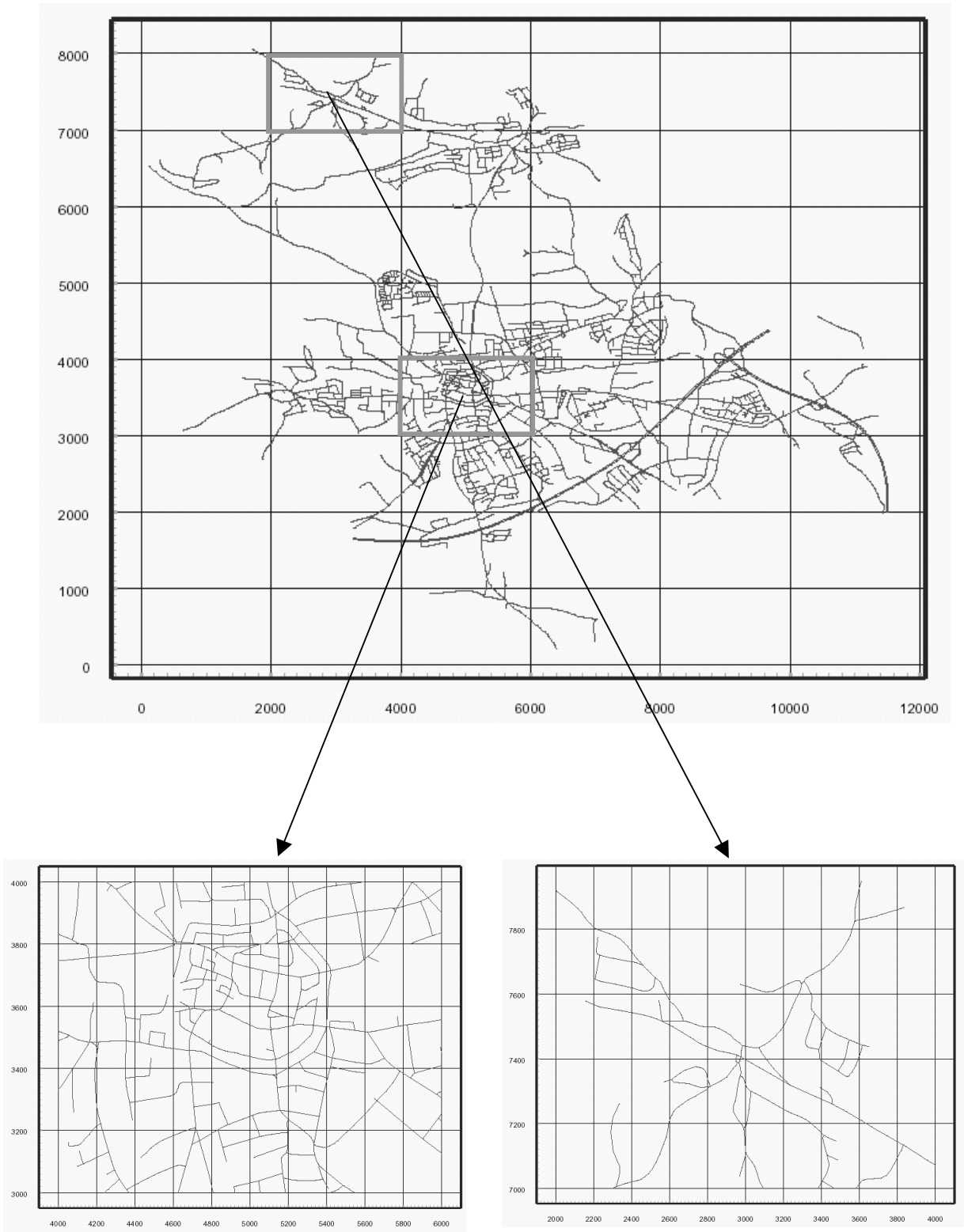
5.1 Simulation Environment

In simulation scenario car traffic in city area was simulated. Two parts of the city were investigated (further mentioned as “City Centre” and “Urban Area”). They were obtained from a real digital map in GDF format [52], preserving details of environment, therefore one-way roads and closed for movement roads were reflected in the model. Boundary dimensions of areas were 2 km by 1 km (Fig.5-1). Because the mobile nodes assumed to be cars, only the road network was considered as movement area. In graph, created by the Environment Model, City Centre contained 1194 locations and 1534 interconnections, summary length of the road network was 28,325 km. Urban Area contained 463 locations and 511 interconnections, summary length of the road network in this case was 10,346 km.

The simulations were performed using GlomoSim Simulation Environment [22]. The mobility models were implemented under CANU Simulation Environment [54] and it was used as generator of mobility patterns for the GlomoSim. Networks with different number of nodes were simulated - from 30 to 150. Netwotk traffic was created by 30 CBR (Constant Bit Rate) connections, using Location-Aided Routing protocol [14] for packet delivery, which was chosen because it uses location information and, therefore, should be dependent on mobility model in use. Another argument for Location-Aided Routing was the existing implementation for the GlomoSim environment. All the nodes had the same transmission range of 250 m, which is typical communication range for such a scenario. Total time of simulation was 900 sec. Presented results were calculated as average after at least 25 passes.

During the simulation, nodes moved between vertices of the graph on a shortest path, which was calculated using Dijkstra algorithm [53]. After arrival to a destination, node stayed there for a pause time between $\lfloor t_{p\min}, t_{p\max} \rfloor$, then a new destination was chosen and the movement continued. If a node was located at a point, from which all the movements were prohibited, it moved to bordering point of the area with infinite speed, thus simulating “border effect” (disappearing of one node from simulation and arrival of another node at boundary point). This was done to disallow all the nodes to be blocked at certain point. At the beginning the nodes were located at random vertices. The Movement Behaviour Model was approximated with constant speed motion with speed values between $[v_{\min}, v_{\max}]$. Two different trip models were used:

- Random Trip Model – generated trips between randomly chosen points of the area
- Local Trip Model – generated trips according to the Local Trip Automata



City Centre *Urban Area*

Fig.5-1: Areas, Used for the Evaluation of User-Oriented Mobility Model

For the Local Trip Model, trip sequences were constructed randomly for each simulation pass. For every area the “pool of locations” was created with points, where activities can be executed. The pool contained 80 locations. The pool of locations was reused upon the successive simulations. Then a “pool of activities” was constructed, which represented typical human deeds (work, shopping, etc.). Each activity could be performed at 10 points, randomly chosen from the pool of locations. Total 8 activities were created. Activity sequence for every node included transitions with random weights between 4 activities, randomly chosen from the pool of activities.

Parameters of the scenario are summarized in Table 5-1.

Parameter	Value
Time of simulation	900 s
Number of nodes	30, 50, 75, 100, 150
Size of simulation area	2 km by 1 km
Transmission range	250 m
Routing protocol	LAR Scheme 1
Number of traffic connections	30
Traffic type	CBR
Packet rate	5 p/s
Packet size	64 byte
Simulation Areas	City Centre (for User-Oriented Mobility Model only)
	Urban Area (for User-Oriented Mobility Model only)
	Rectangle Bounded Area (for Random Waypoint Mobility Model only)
Trip models (for User-Oriented Mobility Model only)	Random Trip Model
	Local Trip Model
Number of possible locations to execute a particular activity (for Local Trip Model only)	10
Total number of activities (for Local Trip Model only)	8
Number of activities per Node (for Local Trip Model only)	4
Movement Behaviour Model	Constant Speed Motion
Movement Speed	30-50 km/h
Pause Time	0 to 100 s

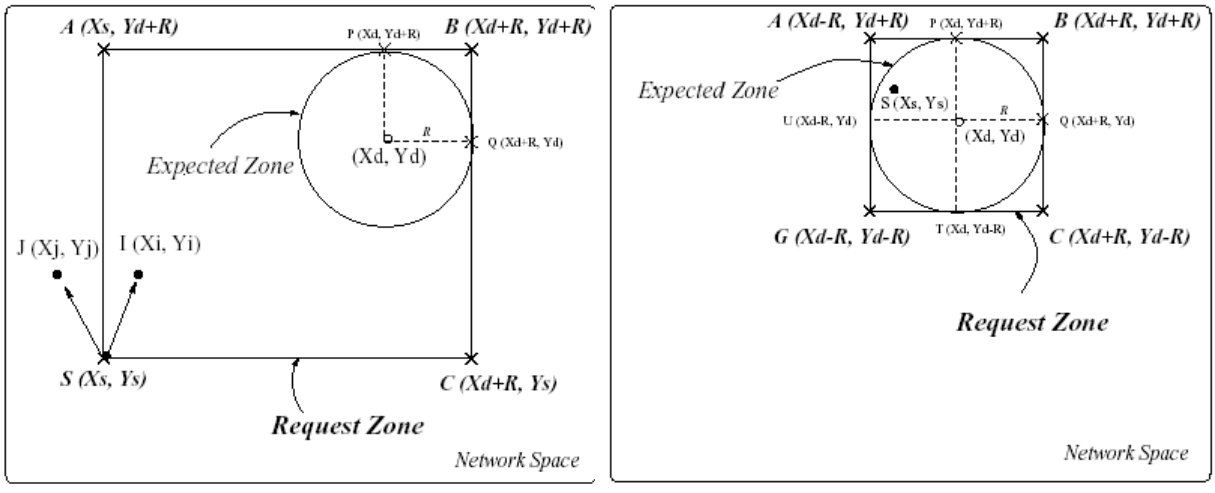
Table 5-1: Simulation Parameters

5.2 Location-Aided Routing Protocol for MANET

Location-Aided Routing (LAR) protocol [14] uses “limited” flooding to find a route to recipient. There are two methods of reducing the flooding proposed; in this thesis only LAR Scheme 1 is used.

Assuming, that originator of a packet located at time t_1 at (x_s, y_s) and knows: (a) location of recipient (x_d, y_d) at time t_0, and (b) average moving speed of the recipient v_d. So, the sender can define expected zone, where the recipient can be found, with circular region of radius $r = v(t_1 - t_0)$, centred at (x_d, y_d). The flooding is limited to request zone, which includes both the location of source and possible location of receiver. Upon searching a route, the sender broadcasts a *route request* packet to all its neighbours. The receiver of the packet compares the desired destination with its own identifier and if they match, it replies with a *route reply* packet. If the node is not destination of the packet, then it checks if it is located within the request zone. If so, it rebroadcasts the request packet to all its neighbours, otherwise it simply discards it. To avoid redundant retransmissions, each node rebroadcasts the same packet only once. If the originator of route searching did not receive route reply packet for a certain timeout interval, it reinitiates route request. Information about the request zone is recorded in header of the packet.

The request zone is defined as following. If a sender is located within expected zone, then the area is limited with rectangle $(x_d - r, y_d - r), (x_d - r, y_d + r), (x_d + r, y_d + r), (x_d + r, y_d - r)$ and with rectangle $(x_s, y_s), (x_s, y_d + r), (x_d + r, y_d + r), (x_d + r, y_s)$ otherwise (Fig.5-2).



(a) Source node outside the Expected Zone

(b) Source node within the Expected Zone

Fig 5-2: Definition of Request Zone (taken from [14])

Parameters of the protocol, used in the simulation, are listed in Table 5-2.

Parameter	Value
Request timeout	2 s
Maximum route length	9 hops
Request seen lifetime	30 s

Table 5-2: LAR Simulation Parameters

5.3 Simulation Results

During the simulation, mobility models were compared by distribution of nodes in the simulation area, data packets delivery ratio, routing protocol overhead and average end-to-end packet delay.

Distribution of nodes in the simulation area

To calculate a distribution of nodes in simulation area, the area was divided into cells, and relative number of nodes in each cell was considered. Size of the cell should be neither too big, nor too small: big cell sizes lead to inaccurate results, while too small sizes increase complexity of results processing. Experimentally it was determined, that cell size 100x100m reflects results with sufficient precision [55]. As metrics for nodes distribution, Herfindahl and normalised Gini coefficients of inequality for cells are used [55, 57, 58].

The coefficients are defined as following. Let $f(i)$ is a function, which assigns to each cell relative number of nodes, located at the cell during the observation. The Herfindahl coefficient of nodes distribution (HC) is defined as sum of squares of relative number of nodes in each cell:

$$HC = \sum_{i=1}^{N} \left(\frac{f(i)}{\sum_{j=1}^{N} f(j)} \right)^2$$

The Herfindahl coefficient is minimal (HC= $\frac{1}{N}$), when nodes are equally distributed over the cells, and maximal (HC=1), when all the nodes are located in single cell. The coefficient expresses the absolute concentration of nodes in the area, e.g. if all the nodes are located in a small number of cells.

The Gini coefficient (GC) is based on the Lorenz curve, which is used to depict inequalities in distribution. To construct the curve, the relative numbers of nodes per cell have to be sorted in increasing order:

$$f(1) \le f(2) \le \ldots \le f(N)$$

The curve is constructed connecting the points:

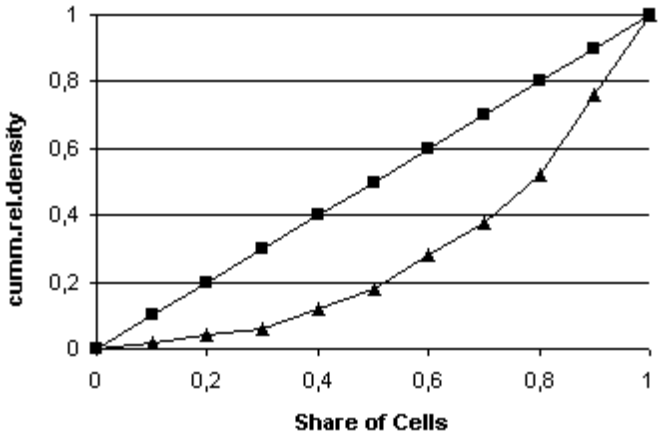
$$(0,0),(u_1,v_1),(u_2,v_2),...,(u_N,v_N)$$

The coordinates (u_i,v_i) are calculated as:

$$u_i = \frac{i}{N} \qquad v_i = \sum_{k=1}^{i} \frac{f(k)}{\sum_{p=1}^{N} f(p)}$$

If the nodes are equally distributed, then the curve is equal to 45-degree line. The bigger the area between the Lorenz curve and 45-degree line, the higher is the inequality in the nodes distribution. The Gini coefficient is defined as ratio of the area between the Lorenz curve and 45-degree line and the area between the 45-degree line and the x-axis (Fig.5-3).

Cell	i	Cell Share	Number of Nodes	Rel. Number of Nodes
8	1	0,1	1	0,02
7	2	0,2	1	0,02
3	3	0,3	1	0,02
4	4	0,4	3	0,06
9	5	0,5	3	0,06
1	6	0,6	5	0,1
10	7	0,7	5	0,1
2	8	0,8	7	0,14
5	9	0,9	12	0,24
6	10	1	12	0,24



HC=0,1632
GC=0,428 GC*=0,476

Fig.5-3: Lorenz Curve and Gini Coefficient

The Gini coefficient can be calculated as follows:

$$GC = \frac{2 \times \sum_{i=1}^{N} i \times f(i)}{N \times \sum_{i=1}^{N} f(i)} - \frac{N+1}{N}$$

The coefficient is minimal (GC=0), when the nodes are equally distributed over the cells, and is maximal (GC=$\frac{N-1}{N}$), when all the nodes are located at the single cell. The normalised Gini coefficient (GC*) is used to get values between 0 and 1 and is expressed as:

$$GC^* = GC \times \frac{N}{N-1}.$$

The Gini coefficient reflects the relative densities between the cells.

Values of coefficients of inequality for different mobility models are shown in Table 5-3. During the simulation, the area was divided into 200 cells (N=200). The results were calculated as average after 25 passes, obtaining 45 distributions per pass.

The distribution of mobile nodes in Random Waypoint Mobility Model is closer to the uniform distribution, than in other reviewed models [56]. In the User-Oriented Mobility Model with the Random Trip Model the distribution of nodes is non-uniform, because of distribution of movement attraction points in the area (Fig.5-4). The application of the Local Trip Model increases the inequality, because the trips now take place between limited numbers of points (Fig.5-5). The coefficients are bigger for the Urban Area, which is explainable because of smaller size of the area and presence of "empty" cells, where nodes cannot be located.

Coefficients for compared mobility models	Random Waypoint Mobility Model	User-Oriented Mobility Model in the "City Centre" with the Random Trip Model	User-Oriented Mobility Model in the "Urban Area" with the Random Trip Model	User-Oriented Mobility Model in the "City Centre" with the Local Trip Model	User-Oriented Mobility Model in the "Urban Area" with the Local Trip Model
Normalised Gini Coefficient of distribution	0,216594	0,629878429	0,781026863	0,703762715	0,797997298
Herfindahl Coefficient of distribution	0,005705	0,016303335	0,023430153	0,021429051	0,023339259

Table 5-3: The Coefficients of Inequality of Nodes Distribution in Compared Mobility Models

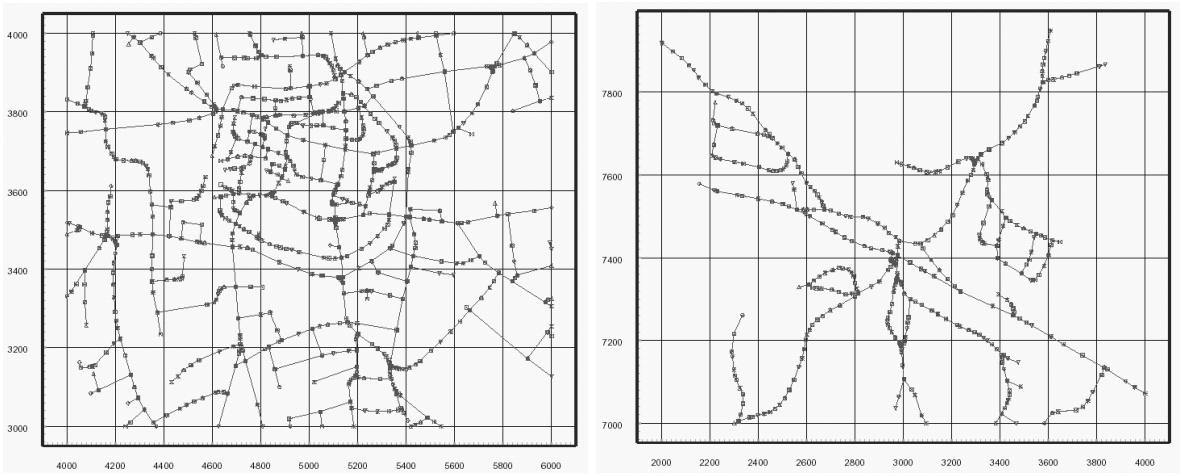


Fig.5-4: The Distribution of Movement Attraction Points in the Areas with the Random Trip Model

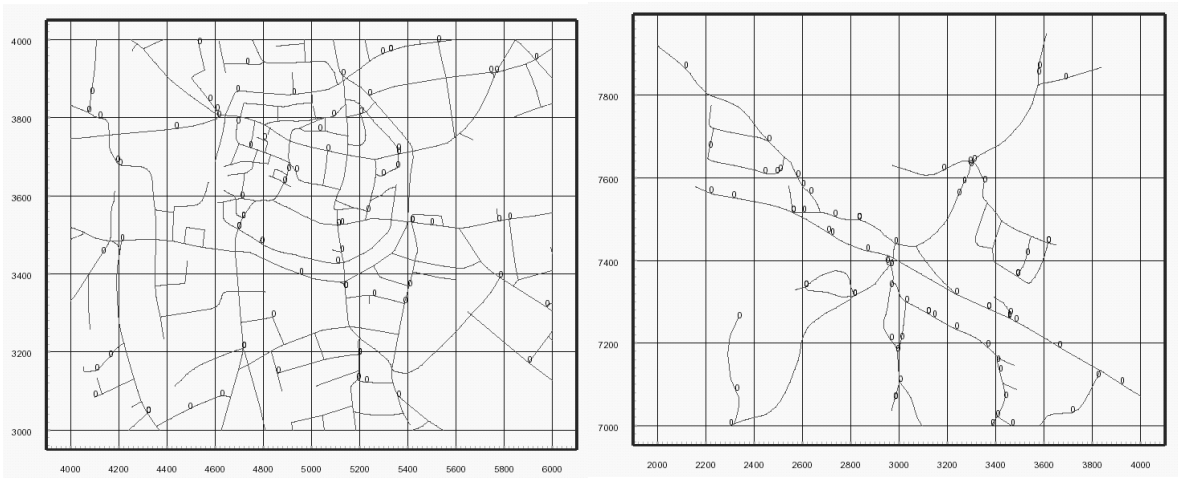


Fig.5-5: The Distribution of Movement Attraction Points in the Areas with the Local Trip Model

Data packets delivery ratio

Data packets delivery ratio was calculated as percentage of data packets being delivered to total number of data packets being sent. From Fig.5-6 it's clear that the simulation with the User-Oriented Mobility Model in urban area constantly shows higher percentage of packets being delivered. This is explainable with more constrained movement area. The nodes are more densely concentrated on graph, so each node has more neighbours, in comparison to other areas, which reduces possibility of partitions in the network. This is proven by results of calculations of average number of neighbours per node in the simulation (Fig.5-7). The same factor explains the higher delivery ratio in centre area comparatively to rectangle-bounded area in the Random Waypoint Mobility Model, with relatively small number of nodes (30-50). With bigger number of nodes (n>50), the rectangle-bounded area shows higher percentage of packets delivery. This can be explained with the fact, that LAR uses flooding upon route

discovery. Flooding on graph requires detour along edges and more nodes get involved in it, comparatively to flooding in the rectangle-bounded area, which has almost uniform distribution of nodes. The flooding in the rectangular area can be spread on the same distance with less number of hops. Since maximum number of hops per route is limited (in the simulation it was set to 9), therefore there are more chances to find a route to recipient in the rectangle-bounded area, than with graph, and simulation results prove it. The usage of Local Trip Model leads to bigger inequalities of nodes distribution, when more nodes get concentrated around particular locations, therefore connectivity between these dense zones reduces and a route to recipient in other zone can be found with less probability. Number of locations in the Local Trip Model was limited to 80, so probability that both sender and receiver will be located in different zones is higher, than if in the same one (79/80 vs. 1/80), which explains results of comparison of Random and Local trip models (Fig.5-8).

Average End-to-End Delay

End-to-end packet delay was calculated as average time for all data packets being successfully delivered between origination of packet at sender side and receiving it at recipient side. From Fig.5-9 it's clear that with relatively small number of nodes (n=30) the both graph-approximated areas show smaller packet delay, in comparison to the rectangular area. It can be explained with the fact, that these areas are more constrained, so the nodes are more closer located and route to recipient can be found rapidly. This also explains smaller packet delay for urban area with n<150. With more number of nodes (n>30), city centre shows greater packet delay, than the rectangular area, because spatial constraints force to use more hops and detour along the graph, thus making forwarding paths larger. Usage of the Local Trip Model limits the distribution of nodes within the area: more nodes get concentrated around particular locations and fewer nodes are located between these dense areas. When route discovery packet will be sent to another zone, it'll take longer to find a path, because of changeable topology of the network due to high mobility of nodes. Because sender and receiver are more likely to be located in different dense zones (as shown above), it explains difference between the results with the Local and Random trip models (Fig.5-10).

Protocol Routing Packet Overhead

Routing packet overhead was calculated as percentage of route request packets to total number of packets being sent (route discovery, data send, data relay). The simulation showed larger number of routing packets sent in the graph-approximated areas, in comparison to the rectangular area (Fig.5-11). This can be explained with bigger inequality of nodes distribution, when there are more nodes, concentrated around particular location, so upon the flooding more local neighbours get involved into retransmission of routing request packets. This also explains difference between the Local and Random trip models (Fig.5-12). Smaller routing packet overhead for urban area is explained with square of the area, when the nodes are closely located to each other and route to recipient can be found without initiation of additional route rediscoveries. Surprisingly, the simulation did not show huge number of routing packets per data packet during the flooding. This can be explained with the source code of LAR implementation for GlomoSim, from which is stated, that a node doesn't initiate a new route discovery for each data packet sent to the same recipient. Instead, if the discovery is already in progress, the data packet is put to the buffer and the node waits for reply to the already sent route request. Because route request timeout interval was set to 30 s and data messages were generated every 0.2 s, so route requests were originated 30/0.2=150 times less, as was initially expected.

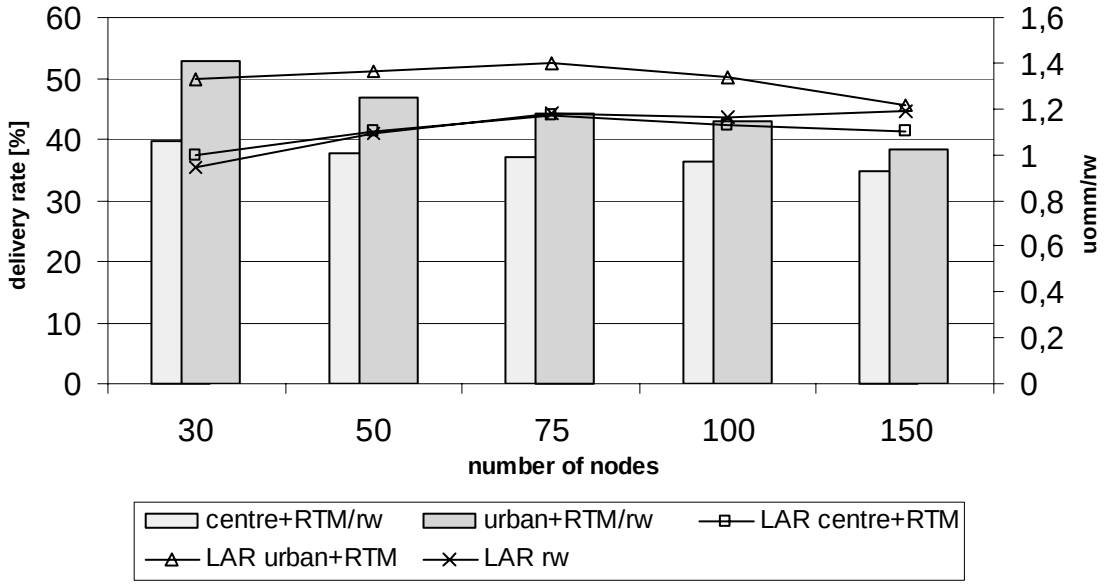


Fig.5-6: Data Packet Delivery Ratio for the Random Waypoint Mobility Model and for the User-Oriented Mobility Model with the Random Trip Model in both Areas

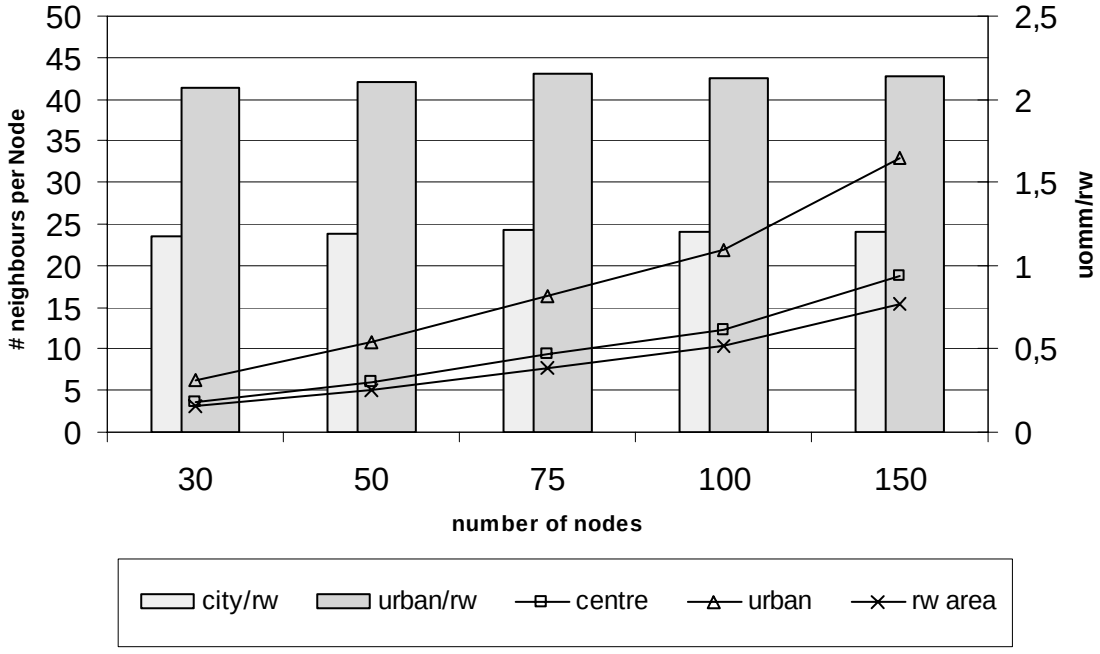


Fig.5-7: Average Number of Neighbours per Node for the Random Waypoint Mobility Model and for the User-Oriented Mobility Model in both Areas

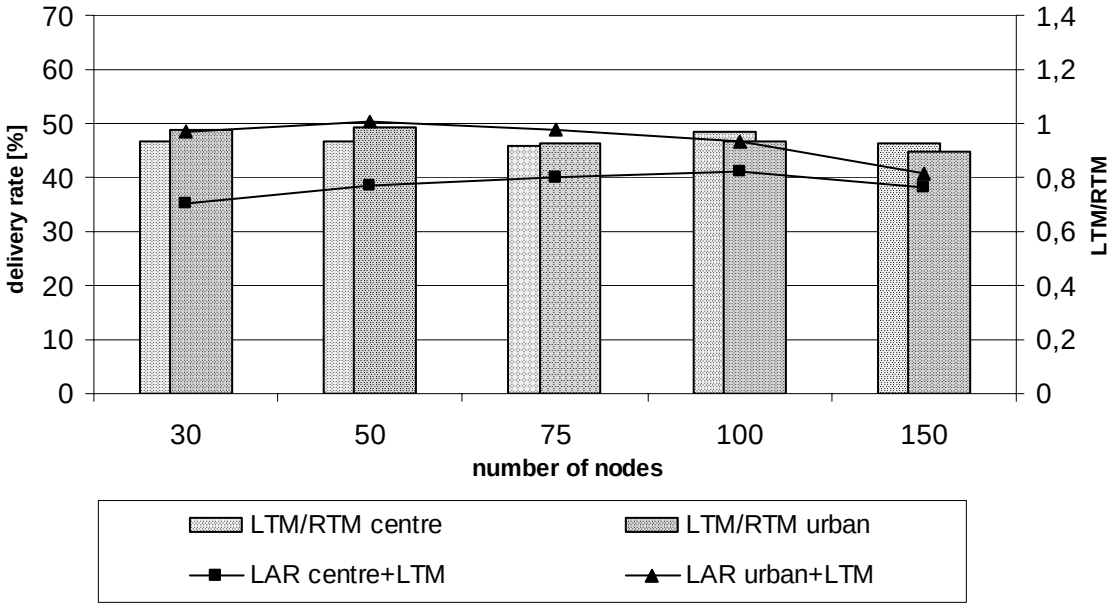


Fig.5-8: Comparison of Packet Delivery Ratio for the User-Oriented Mobility Model with the Random and Local Trip Models

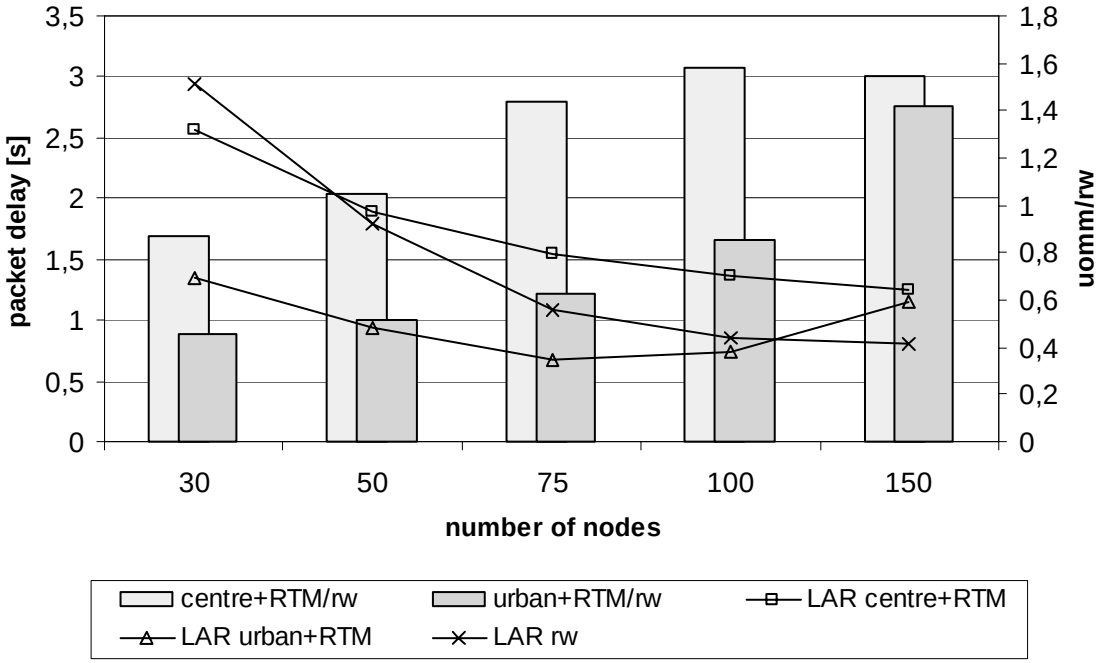


Fig.5-9: Average End-to-End Delay for the Random Waypoint Mobility Model and for the User-Oriented Mobility Model with the Random Trip Mode in two Areas

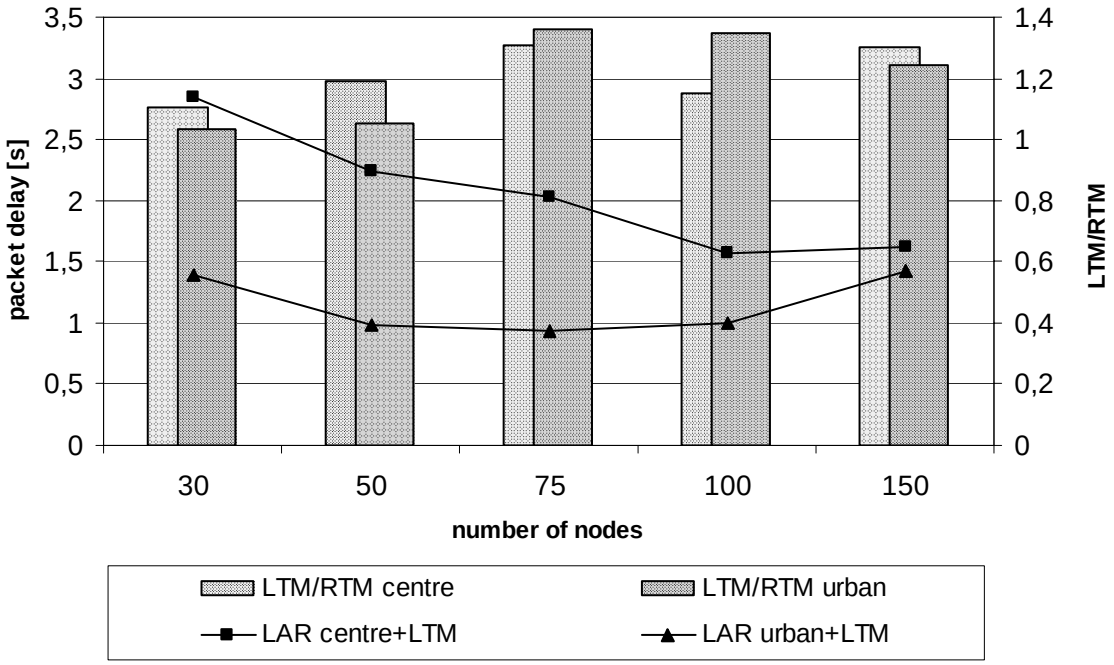


Fig.5-10: Comparison of Average End-to-End Delay for the User-Oriented Mobility Model with the Random and Local Trip Models

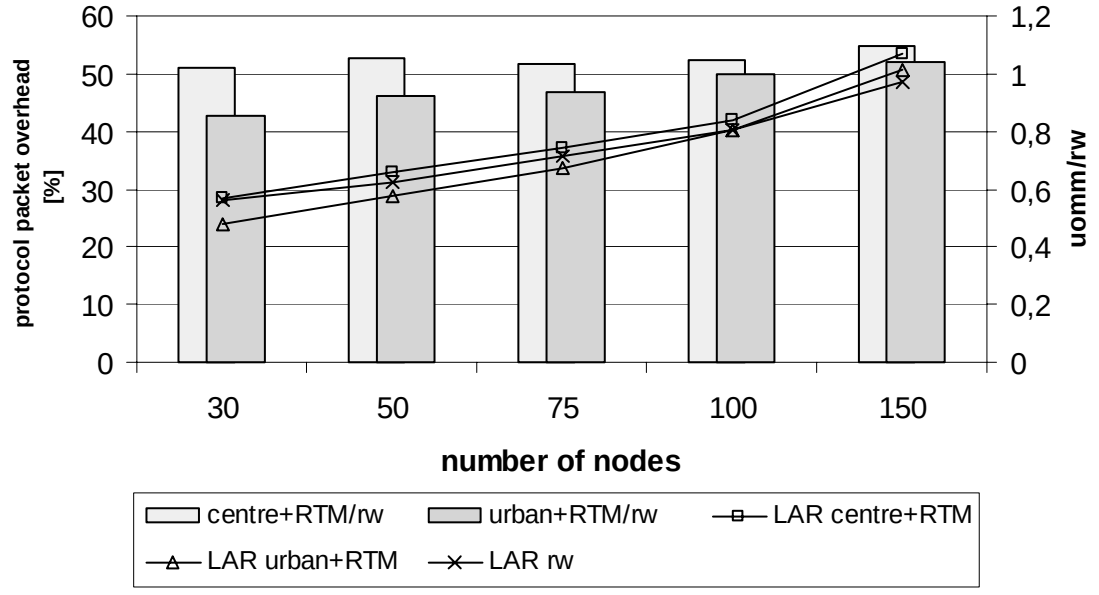


Fig.5-11: Routing Protocol Packet Overhead for the Random Waypoint Mobility Model) and for the User-Oriented Mobility Model with the Random Trip Model in two Areas

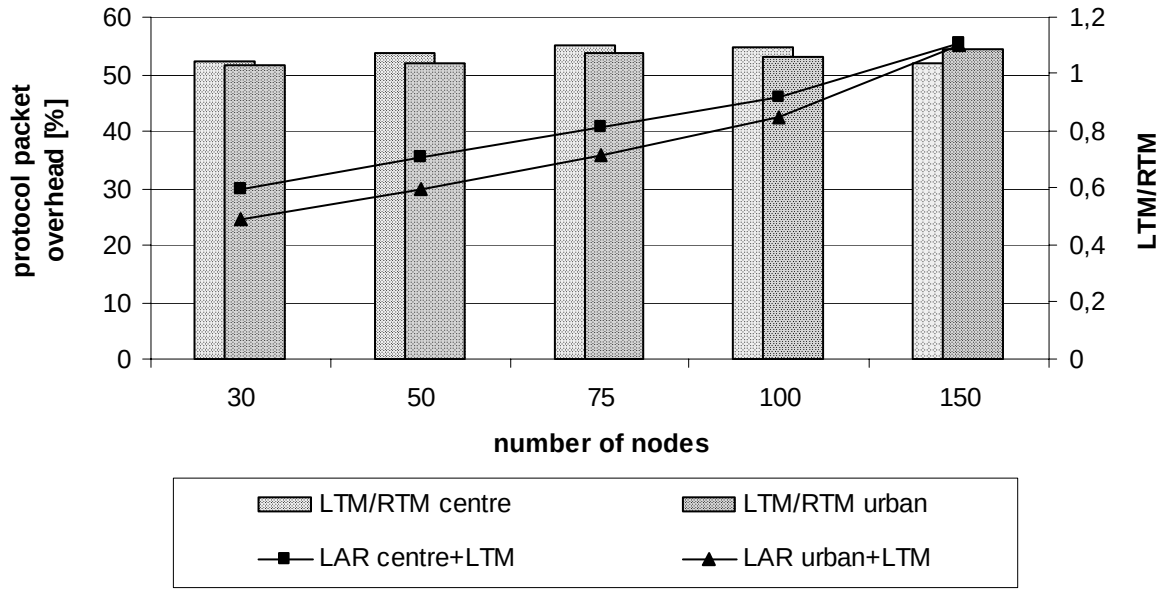


Fig.5-12: Comparison of Routing Protocol Packet Overhead for the User-Oriented Mobility Model with the Random and Local Trip Models

5.4 Simulation Summary

During the evaluations, car traffic in city area was simulated with both the User-Oriented and Random Waypoint mobility models to prove difference in performance of the MANET with the random and more realistic mobility models. Location-Aided Routing was chosen as routing protocol for the evaluation, primarily because it uses location information and, therefore, should be dependent on mobility model in use. Another argument for Location-Aided Routing was the existing implementation for the GlomoSim environment. The simulation showed difference in results with the random and more realistic mobility models. Primary explanatory factors of the difference were dissimilarity of movement areas and inequality of nodes distribution.

6 Summary and Conclusions

Users' mobility is a key feature of the mobile ad hoc network. Most of simulations simplify it with random movement, which is only suitable for limited number of scenarios. In reality the movement of persons has a certain value of regularity, which has to be reflected in simulations. Moreover, there are protocols, which rely on movement prediction mechanisms (such as Model-Based Routing or Spatial-Aware Geographic Forwarding) and therefore cannot be fully evaluated with the random movement.

In this thesis a more realistic mobility model was proposed, called User-Oriented Mobility Model. The model is based on the Environment Model, containing detailed information about simulation area; the Trip Model, reflecting trips made by persons in the area; and the Movement Behaviour Model, which simulates movement behaviour upon touring to destination. The Environment Model uses data from real digital maps in common formats, thus solving the problem of creation of detailed data sources. The Trip Model is based on the Activity-Based Travel Modelling Approach, but expresses the trips using Automata of Activity Sequences. A number of Movement Behaviour models, suitable for simulating the behaviour of different kinds of mobile objects, can be included into the model.

In evaluation part, the Location-Aided Routing Protocol was simulated with both the User-Oriented Mobility Model and Random Waypoint Mobility Model. The results showed difference in usage of both models, which was caused by constraints of movement area and unequal distribution of mobile nodes in more realistic mobility model.

In future work more routing protocols will be simulated with the User-Oriented Mobility Model. Also there are plans to included obstacles into communication model to make the evaluations more realistic. More metrics to compare different movement areas have to be found. "Border effect" in a simulation area has to be precisely investigated.

References

1. J.S.M. Ho, I.F. Akyildiz, "A Mobile User Location Update and Paging Mechanism Under Delay Constrains", Wireless Networks, vol.1, no.4, pp.413-425, 1995.

2. I.F. Akyildiz, S.M. Ho, Y. -B. Lin, "Movement-Based Location Update and Selective Paging for PCS Networks", IEEE/ACM Trans. Networking, vol.4, pp.629-638, Aug. 1996.

3. W.S. Jeon, D.G. Jeong, "Performance of Improved Probabilistic Location Update Scheme for Cellular Mobile Networks", IEEE Transactions on Vehicular Technology, accepted, 2000.

4. M. Schopp, "Use Modelling and Performance Evaluation of Distributed Location Management for Personal Communication Services", Proceedings of the 15th International Teletraffic Congress (ITC 15), Washington, DC, June 1997 pp. 23-34.

5. M.M. Zonoozi, P. Dassanayake, "User Mobility Modeling and Characterization of Mobility Patterns", IEEE Journal on Sel. Areas in Communications, 15(7): 1239-1252, Sept. 1997.

6. R.P. Saldana, M.C.C. Changho, "On Random Walk Models and Markov Chains", Proceedings of the Philippines Computing Science Congress, November 29 - December 1, 2000.

7. I.F. Tsai, R.H. Jan, "The Lookahead Strategy for Distance-Based Location Tracking in Wireless Cellular Networks", ACM Mobile Computing and Communications Review, vol. 3, 1999, pp. 27-38.

8. T. Liu, P. Bahl, "Mobility Modeling, Location Tracking, and Trajectory Prediction in Wireless ATM Networks", in the IEEE Journal on Special Areas in Communications, Special Issue on Wireless Access Broadband Networks, Vol. 16, No. 6, (August 1998): 922-936.

9. G.Y. Liu, G.Q. Maguire, "A Predictive Mobility Management Scheme for Supporting Wireless Mobile Computing", Proceedings of IEEE International Conference on Universal Personal Communications (ICUPC'95), Tokyo, Japan, November 6-9, 1995.

10. G.Y. Liu, G.Q. Maguire, "Efficient Mobility Management Support for Wireless Data Services ", Proceedings of 45th IEEE Vehicular Technology Conference (VTC'95), Chicago, Illinois, July 26-28, 1995.

11. A. Bar-Noy, I. Kessler, M. Sidi, "Mobile Users: To Update or not to Update?", Wireless Networks journal, Vol. 1, pp. 175-186, 1995.

12. C. -C. Chiang, "Wireless Network Multicasting", PhD thesis, University of California, Los Angeles, 1998.

13. Z.J.Haas and M.R.Pearlman, "The Performance of a New Routing Protocol for the Reconfigurable Wireless Networks", Proc. ICC '98, pages 156-160.

14. Y.-B. Ko and N. H. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks", in ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom'98), October 1998.

15. Samir R. Das, Robert Castaneda, Jiangtao Yan, and Rimli Sengupta, "Comparative performance evaluation of routing protocols for mobile ad hoc networks", in 7th International Conference on Computer Communications and Networks (IC3N), pages 153-161, October 1998.

16. Christian Bettstetter, "Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks", in Proceedings 4th ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (WSWiM'01), Rome, Italy, July 2001.

17. R. Landau, "Random Walk Simulation", http://nacphy.physics.orst.edu/ComPhys/MONTE/mc3/node4.html

18. J. Tian, J. Hähner, C. Becker, I. Stepanov, K. Rothermel, "Graph-based Mobility Model for Mobile Ad Hoc Network Simulation", proceedings of 35th Annual Simulation Symposium, in cooperation with the IEEE Computer Society and ACM, San Diego, California, April 2002.

19. M. Sanchez, P. Manzoni and Z.J. Haas, "Determination of Critical transmission Range in Ad-Hoc Networks", In Proceedings of Multiaccess Mobility and Teletraffic for Wireless Communications 1999 Workshop (MMT'99), October 6th-8th, Venice, Italy.

20. B. Liang, C. Hass, "Predictive Distance-based mobility management for PCS Networks", IEEE INFOCOM'99, New York, NJ, March 1999.

21. "The Network Simulator - ns-2", http://www.isi.edu/nsnam/ns/

22. "GloMoSim: Global Mobile Information Systems Simulation Library", http://pcl.cs.ucla.edu/projects/glomosim/

23. B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", In Proceedings of the 7th ACM International Conference on Mobile Computing and Networking, pages 85-96, Rome, Italy, July, 2001.

24. J. Broch, D.A. Maltz, D.B. Johnson, Y. -C. Hu, and Jorjeta Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, ACM, Dallas, TX, October 1998.

25. D.B. Johnson, D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

26. P. Johanson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," Proceedings of ACM/IEEE MOBICOM'99, Seattle, WA, August 1999, pp. 195-206.

27. L. Blazevic, S. Giordano, J. Y. Le Boudec, "Self Organized Terminode Routing", Technical Report No. DSC/2000/040, Swiss Federal Institute of Technology, Lausanne, December 2000.

28. Douglas S. J. De Couto and Robert Morris, "Location Proxies and Intermediate Node Forwarding for Practical Geographic Forwarding", MIT Laboratory for Computer Science, June, 2001.

29. X. Hong, T. Kwon, M. Gerla, D. Gu and G. Pei, "A Mobility Framework for Ad Hoc Wireless Networks", Proceedings of ACM Second International Conference on Mobile Data Management (MDM '2001), Hong Kong, Jan, 2001.

30. T. S. Kim, M. Y. Chung, and D. K. Sung, "Mobility and traffic analyses in three-dimensional PCS environments," IEEE Transactions on Vehicular Technology, vol. 47, pp. 537-545, May 1998.

31. X. Hong, M. Gerla, G. Pei and Ch.-Ch. Chiang, "A group mobility model for ad hoc wireless networks", Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, United States, Seattle, Washington, 1999.

32. I. Seskar, S.V. Marie, J. Holtzman, J. Wasserman, "Rate of Location Area Updates in Cellular Systems", IEEE VTC'92, Denver, CO, May 1992.

33. G.P. Pollini, C. -L. I, "A Profile-Based Location Strategy and Its Performance", IEEE Journal on Selected Areas in Communications, Volume 15, Number 8, October 1997.

34. J.G. Markoulidakis, G.L. Lyberopoulos, D.F. Tsirkas, E.D. Sykas, "Mobility Modeling in Third-Generation Mobile Telecommunications Systems", IEEE Personal Communications, pp. 41-56, August 1997.

35. J. -M. Ho, J.S. Mehta, D.S. Tan H. Tanabe, S. Zhou, "Design and Evaluation of an Individually Simulated Mobility Model in Wireless Ad Hoc Networks", Pending submission to Communication Networks and Distributed Systems Modeling and Simulation Conference 2002, San Antonio, Texas.

36. Boids web-site by Craig Reynolds, www.red3d.com/cwr/boids

37. C.W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", in Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings), pages 25-34.

38. D.C. Brogan, J.K. Hodgins, "Group Behaviors for Systems with Significant Dynamics", Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3, pp 528-534.

39. C.W. Reynolds, "Steering Behaviors For Autonomous Characters", Proceedings of the Game Developers Conference, pages 763-782, 1999.

40. E. Shaw, "Schooling in Fishes: Critique and Review", In L. Aronson, E. Tobach, D. Leherman, and J. Rosenblatt, editors, Development and Evolution of Behavior, pages 452-480, 1970.

41. Eric I. Pas. Recent Advances in Activity-Based Travel Demand Modeling. Proceedings of Activity-Based Travel Forecasting Conference, June 2-5, 1996.

42. Chapin, F.S. Human Activity Patterns in the City. John Wiley & Sons, New York, 1974.

43. Hagerstrand, T. What about people in regional science? Papers in Regional Science, 24, 7-21, 1970.

44. Hagerstrand, T. The Impact of Transport on the Quality of Life. Fifth International Symposium on Theory and Practice in Transport Economics, Greece, 1973.

45. Ryuichi Kitamura. Applications of Models of Activity Behavior for Activity Based Demand Forecasting. Proceedings of Activity-Based Travel Forecasting Conference, June 2-5, 1996.

46. John Scourias and Thomas Kunz. An activity-based mobility model and location management simulation framework. In Proceedings of the Second ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM'99), Seattle, USA, pages 61-68, August 1999.

47. John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun-chan Lan, Ya Xu, Wei Ye, Deborah Estrin, Ramesh Govindan, and John Heidemann. Effects of Detail in Wireless Network Simulation. SCS Communication Networks and Distributed Systems Modeling and Simulation Conference, September, 2000.

48. Geography Markup Language (GML) 2.0, OpenGIS® Implementation Specification, OGC Document Number: 01-029, http://www.opengis.net/gml/01-029/GML2.html, 20 February 2001.

49. Ron Lake. Introduction to GML Geography Markup Language, http://www.jlocationservices.com/company/galdos/articles/introduction_to_gml.htm

50. Geographic Data Files (GDF) Home Page, http://www.ertico.com/links/gdf/gdf.htm

51. M. Treiber and D. Helbing, Explanation of Observed Features of Self-Organization in Traffic Flow, preprint cond-mat/9901239 (1999).

52. Tele Atlas Homepage, http://www.teleatlas.com

53. E.W. Dijkstra, “A note on two problems in connection with graphs“, Numerische Mathematik, 1, pp.269-271 (1959).

54. CANU Project Homepage, http://canu.informatik.uni-stuttgart.de

55. Uwe Kubach, Kurt Rothermel, “Estimating the Benefit of Location-Awareness for Mobile Data Management Mechanisms”, to appear in proceedings of Pervasive 2002.

56. Christian Bettstetter, Christian Wagner, “The Spatial Node Distribution of the Random Waypoint Mobility Model”, Technische Universitaet Muenchen, Institute of Communication Networks, Munich, Germany, In Proc. of the 1st German Workshop on Mobile Ad-Hoc Networks (WMAN'02), Ulm, Germany, March 25-26, 2002.

57. A.B.Atkinson, “Wealth, Income and Inequality”, Oxford University Press, Inc., second edition, 1981.

58. A.Sen, “On Economic Inequality”, Clarendon Paperbacks, second edition, 1993.

References

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst
und nur die angegebenen Hilfsmittel verwendet habe.

_____

Illya Stepanov