Escuela Técnica Superior de Ingenieros de Telecomunicación
Prof. Dr. Vicente Casares-Giner
Dr. Pablo García-Escalle

Institut für Kommunikationsnetze und Rechnersysteme
Prof. Dr.-Ing. Dr. h. c. mult. Paul J. Kühn

Master Thesis

Performance Evaluation of Time-Based and Movement-Based Location Update Schemes

Author: Sebastian Bachmaier
Supervisor: Dr. Pablo García-Escalle
Period: 1 February 2006 - 15 September 2006

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
AuC	Authentication Centre
BS	Base Station
CAC	Call Admission Control
CIR	Carrier-to-Interference-Ratio
CMR	Call-to-Mobility-Ratio
CS	Circuit Switched
CSV	Comma Seperated Values
CTR	Call-to-Timeout-Ratio
ETSI	European Telecommunications Standards Institute
GSM	Global System for Mobile Communications
HLR	Home Location Register
ID	Identification
IMSI	International Mobile Subscriber Identity
LA	Location Area
LST	Laplace Stieltje Transform
LT	Laplace Transform
LU	Location Update
MC	Markov Chain
MM	Mobility Management
MPF	Most Probable First
MSC	Mobile Switching Centre
MT	Mobile Terminal
MT	Mobile Terminal
PCS	Personal Communications Service
PDF	Probability Distribution Function

pdf	Probability Density Function
PEPA	Performance Evaluation Process Algebra
PG	Paging
PS	Packet Switched
RF	Radio Frequency
RLC	Radio Link Control
RNC	Radio Network Controller
RP	Random Process
RRC	Radio Resource Control
RV	Random Variable
SAN	Stochastic Activity Network
SDF	Shortest Distance First
SIR	Signal-to-Interference-Ratio
SNR	signal-to-noise-ratio
SPN	Stochastic Petri Net
SRES	Signed RESult (result of the GSM A3 algorithm)
SRNS	Serving Radio Network Subsystem
SS7	Signalling System #7
TE	Terminal Equipment
TMSI	Temporary Mobile Subscriber Identity
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
VLR	Vistor Location Register

Declaration of Authorship

I hereby declare that I am the sole author of this thesis and used nothing but the specified resources and means.

15 September 2006 S. Bachmaier

Abstract

English

In Personal Communications Service (PCS) networks, location management is a key issue. Mobility tracking operations are executed to maintain known the whereabouts of each mobile terminal. In this work, some modifications on the classical time-based and movement-based registration methods are evaluated. The goal is to provide a unified approach of the time-based and distance-based schemes on one hand, and the movement-based and distance-based strategies on the other hand. Furthermore what is achieved is that the Mobile Terminal sends less location update messages, which results in less contacts with the network and an increase in the uncertainty of the Mobile Terminal position. Although the Mobile Terminal Paging costs are lightly increased, the result is a significant reduction in the Location Update cost, and the net effect is a saving in the total location management cost per call arrival, i.e. the Location Update cost plus the Paging cost.
The studied mechanisms will be compared with the original dynamic schemes by means of a performance evaluation method based on software simulation tools and contrasted to an analysis based on Markovian standard tools. The location management costs for all these dynamic policies will be evaluated.

Spanish

En redes de comunicaciones móviles, la gestión de localización es una cuestión clave. Las funciones de movilidad se ejecutan para mantener localizable cada terminal móvil. En este proyecto, deseamos evaluar algunas modificaciones en los métodos de registro clásicos basados en tiempo y movimiento. El objetivo es proporcionar una visión unificada de los esquemas de tiempo y distancia por una parte, y de los esquemas de movimiento y distancia por otra. Esperamos que el terminal móvil envíe menos mensajes de actualización de posición, que darán lugar a menos contactos con la red y un aumento en la incertidumbre de la posición del terminal móvil. Aunque el coste de búsqueda del terminal móvil aumente ligeramente, contamos con una reducción significativa del coste de actualización de posición, y el efecto final será un ahorro en el coste total por llamada entrante, es decir el coste de actualización de posición más el coste de búsqueda.
Los mecanismos estudiados serán comparados con los esquemas dinámicos originales por medio de un método de la evaluación basado en una herramienta de simulación y luego serán comparado con una análisis basado en las herramientas estándares markovianas. Los costes de gestión de la localización para todas estas políticas dinámicas serán evaluados usando estos métodos de la evaluación.

1 Introduction

1.1 Relevance of the work

For mobile phone operators it is of superior interest to keep operational costs for their network minimal. Mobile phoning is not a privilege to just a few business customers any more, willing to pay any price for being available worldwide. The turnover today is made by the vast amount of private mobile phone users who are very cost sensitive. Those users prefer also to have a small standing charge or none at all. The operators on the other hand have their fixed costs for the network operation which they like to be covered by the standing charge. One part of the network is the signalling subnetwork, responsible for transmission of signalling messages for connection and mobility management. Signalling traffic is of course essential for the operation of the network, but as the customer does does not take notice of these efforts he is generally not willing to pay for it. A simple calculation can show the benefit of saving signalling bandwidth: For a mobile operator with nearly 190 million customers[1], a user cell sojourn time of 120 s and 10 cells per location area (LA) these customer movements result in a LA crossing rate of

$$\frac{19 \cdot 10^7 \text{ customers}}{120\text{s} \cdot 10 \text{ cells/LA}} \approx 158.000 \ \frac{\text{LA crossings}}{\text{s}}.$$

Each of these crossings cause the MT to inform the HLR/VLR about its new location. This procedure takes about 800 ms ([26]) and the messages exchanged can sum up to a total size of 1000 bytes. As can be seen, the needed bandwidth is about 160 MB/s. This transport capacity is needed over wide areas of the network, as MT and VLR are not necessarily geographically close to each other. Although this is only a very rough estimation it shows that even a few percent of reduction in location update (LU) traffic will result in several MB/s less signalling load. However the signalling has other effects than the used transfer capacity. It also generates processing load in various network elements and uses a considerable amount of radio resources in each cell, which is maybe the worst, as the radio bandwidth resource is very scarce. For example Vodafone Germany paid 8.48 billion € for their Universal Mobile Telecommunications System (UMTS) licences for 20 years of usage[2]. Saving only 1 ‰ of the network capacity they were able to save about half a million € per year.

One other encouraging cause to investigate and improve location update procedures lies in the fact that it is not the most active users who cause the most signalling traffic but those moving fast, as they cross many LAs. So, even a user with a very low call rate incurs signalling procedures without yielding revenue. This is of course only true for static LU procedures, while dynamic solutions, based on the individual users behaviour, which are investigated in this work, can avoid these flaws.

Besides the monetary aspect, there is also always an environmental facet in minimising efforts for a given task. In this case it seems a bright idea to minimise needed transport capacity to avoid unnecessary energy consumption in network elements and to reduce useless emission of radio frequency energy,

[1] referring to Vodafone homepage at `http://www.vodafone.com` (date of access: 2006-08-26)
[2] according to the German Bundesnetzagentur press archive at `http://www.bundesnetzagentur.de`

two quite valuable advantages. The latter of these is not only an issue of energy wasting, but as long as the interaction of human cells with radio frequency waves is not fully understood also a measure of precaution.

With cells sizes getting smaller in UMTS and other future cellular networks, all the above mentioned issues get even more important, since the user traverses faster between cells, thus increasing the LU rate.

1.2 Objective of this work

The goal is to reduce signalling traffic caused by the MTs by proposing an improved LU scheme for reasons given in the preceding section. This improved scheme will be explained with the underlying theoretical basics and methods to make it possible for network operators to implement and for standardisation organisations to standardise the improved scheme. The magnitude of the achievable enhancement should be visualised in perspicuous graphics that show the dependency of the enhancements from several parameters of interest. Furthermore any interested reader should be able to continue improving the proposed scheme as all necessary details of the used model are going to be explained.

1.3 Structural Approach

The introductory chapter covers formal aspects of this work, as the relevance of the work, i.e. why it is quite reasonable to treat the subjects of this work, and the objective of the work itself.

Chapter 2 introduces to fundamental mathematical notions that are used for the analytical part of this work as well as in many of the literature references. Among these notions the stochastic processes and their characteristic subtypes will be treated in most detail. The underlying probability theory is assumed to be known, as only short definitions will be made.

Chapter 3 covers mobile networks in a very extensive way. Experienced readers might wish to skip the general and architectural subchapters and start with section 3.3 about Mobility Management. This section however is crucial for the further understanding of the proposed improved location update scheme that will be treated in the main part. Section 3.2 explains why Mobility Management is necessary in cellular networks.

Chapter 4 introduces the different proposed schemes for location update. It is also shown, that for certain parameters the proposed schemes resemble well-known schemes from the literature.

In chapter 5 the Markov model for analytic performance evaluation is explained in detail, giving the prerequisites under which the analysis is true. Then the mathematical treatment is explained, but not conducted however in the scope of this work.

The main part of the work is described in chapter 6. There, the conducted simulations are described and the results are shown in various representations and explained accordingly. Matlab source code linked with this section can be found in the appendix C. The used stochastic activity network model details for this chapter can be found in appendix D.

Finally in chapter 7 conclusions are drawn and the work is summed up. Further possible or necessary improvements in the presented scheme are shown.

2 Theory

This chapter will introduce some of the most important concepts that are used in the model descriptions in later chapters. Some basic definitions of probability theory are repeated. Then the stochastic processes are presented with some of the fundamental theorems related to them.

2.1 Basic Probability Theory

Events in real life can be assigned a probability which just means how often they occur in relation to other events.

$$p_x = \frac{\text{number of occurrences of } x}{\text{number of occurrences of all events}}$$

This was the classical definition of probability until the 1930s when Kolmogorov[1] established the axiomatic definition of probability, based on the theory of sets, given by equations (2.1)-(2.3). A and B represent arbitrary *sets of outcomes* of a random experiment, and are called *events*

$$P[A] \geq 0 \tag{2.1}$$

$$P[\mathcal{S}] = 1 \qquad \text{where } \mathcal{S} \text{ is the complete outcome space, also known as } \textit{certain event} \tag{2.2}$$

$$P[A \cup B] = P[A] + P[B] \qquad \text{if } A \cap B = \varnothing \tag{2.3}$$

Based on these three axioms all other properties of probability theory be derived from. Still the classical definition of relative occurrences can be used to illustrate the meaning of probability more clearly.

To be able to calculate with probabilities is is necessary to introduce the concept of *random variables* (RV). To every outcome of an random experiment a number is assigned (integer or real) which represents the outcome. For RVs upper case letters will be used.
For continuous RVs the probability for the random variable X to take values less than or equal to x is called *probability distribution function* and given by

$$P[X \leq x] = F(x) \tag{2.4}$$

Similarly the chance of X being greater than x is called the *complementary distribution function* and both distribution function and complementary distribution function sum up to 1 as they make up the complete event space.

$$P[X > x] = 1 - F(x) = F^{C}(x) \tag{2.5}$$

[1] Andrey Nikolaevich Kolmogorov; (rus. Андрей Николаевич Колмогоров); ⋆1903-†1987; Russian mathematician whom major contributions in probability theory, topology and algorithmic complexity theory and other fields of mathematics are owed ([1][22])

The probability for a RV to take values in a range (x_1, x_2) is

$$P[x_1 < X \leq x_2] = F(x_2) - F(x_1). \tag{2.6}$$

The derivation of the distribution function is called *probability density function*. The abbreviation "pdf" will be used throughout the text. So the following equivalent relations hold:

$$f(x) = \frac{dF(x)}{dx} \tag{2.7}$$

$$F(x) = \int_{-\infty}^{x} f(\xi)\, d\xi \tag{2.8}$$

With eq. (2.7) a special case of (2.6) can be given:

$$P[x < X \leq x + dx] = F(x) - F(x + dx) = dF(x) = f(x)\, dx. \tag{2.9}$$

This equation gives the probability for X being in the close range of x, which is more or less $P[X = x]$. Not exactly of course, since for continuous RVs there are infinitely many possibilities for X to take (on the number ray of real numbers), so the probability to "hit" exactly a given value x is zero, thus $P[X = x] = 0$. Still, eq. (2.9) can be used in any infinitesimal context.
Next the definition of the *conditional probability* is presented:

$$P[A|B] = \frac{P[A \cap B]}{P[B]} \tag{2.10}$$

It can be interpreted as the probability that an event A happens, if it is already known that B has happened. The conditional probability can be used to derive the *law of total probability*: the complete event space $\mathcal{S}$ is divided into N arbitrary but mutually exclusive events B_i with $i = 1, 2, \ldots, N$. For any event A holds eq. (2.10) with $B = B_i$, i.e. $P[A|B_i]P[B_i] = P[A \cap B_i]$. Summing up both sides of the equation for all i and considering that $\sum_{\text{all i}} P[A \cap B_i] = P[A]$ since all B_i build up the certain event, we arrive at:

$$P[A] = \sum_{i=1}^{N} P[A|B_i] \cdot P[B_i] \tag{2.11}$$

and its continuous form

$$P[A] = \int_{-\infty}^{\infty} P[A|B = x] \cdot f(x)\, dx. \tag{2.12}$$

On the pdf of continuous RVs the *Laplace transform* (LT) can be applied. In this work the LT of an function will be denoted with the same letter, but with a star to remind it being the LT. That is

$$f(x) \quad \circ\!\!-\!\!\bullet \quad f^{\star}(s) = \int_0^{\infty} e^{-sx} f(x)\, dx \qquad \text{if } f(x) = 0 \;\forall\; x < 0 \tag{2.13}$$

2.2 Stochastic Processes (Random Processes)

In chapter 2.1 RVs were introduced. The concept can now be elaborated further if the time dimension is added. If we treat the time dependent behaviour of a RV, we call it a *random process* (RP) or a *stochastic process* [15]. The most basic differentiation among random processes is to divide them according to the point of time at which transitions may occur, i.e. to divide them into *continuous-time* and *discrete-time* processes. Discrete-time processes are rather called *sequences* than processes. The other discrimination dimension is the state space: the RV of the random process can either be a real number or a countable integer. If the RV is a continuous real number then it is a process in the proper sense. Whereas if the RV can take only a countable number of states (whereby this number can still be infinite) it is referred to as *chain* instead of process.

Besides the discrimination according the index and value domain further classifications of RPs can be made according to their stochastic properties. One of the most important classes is that of the Markovian RP, which is described in more detail in the following section.

2.2.1 Markov Process

According to [13] Markov processes are named after a paper of Markov[2] in which he first introduced the concept of a random process with a certain dependency between successive RV which is nowadays known as the *Markov property*. The Markov property states that the further development of the process depends only on the current value of the process but not on those values before that. This property is therefore also called the *memoryless* property, as the history of the RV is of no interest to the further developing of the RV. Mathematically this property can be given in the form (compare [24])

$$P[X(t_n) \leq x_n | X(t), t \leq t_{n-1})] = P[X(t_n) \leq x_n | X(t_{n-1})] \quad \text{if} \quad t_{n-1} < t_n \tag{2.14}$$

This property makes Markov processes especially easy to handle. Figure 2.1 illustrates the meaning of eq. (2.14). Although some different realisations of the process may have gone in different ways in the past, once they "meet" at the same point at a given point in time, their further proceeding will be statistically the same.

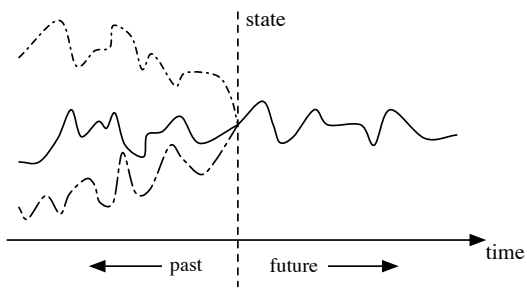


Figure 2.1: In a Markov process the future development does not depend on the past (figure acc. to [16])

[2] Andrey Andreyevich Markov; (rus. Андрей Андреевич Марков); ⋆1856-†1922; Russian mathematician who worked in number theory, analysis and especially in probability theory; initiated by his work the theory of stochastic processes was established as field in mathematics ([23])

2.2.2 Markov Chains

Here a special case of Markov process is introduced, namely a Markov process with discrete state space, as introduced in section 2.2. As the RV can take discrete values only, eq. (2.14) "simplifies" to

$$\begin{aligned} &P[X(t_n) = j | X(t_{n-1}) = i_{n-1}, X(t_{n-2}) = i_{n-2}, X(t_{n-3}) = i_{n-3}, \ldots] = \\ &P[X(t_n) = j | X(t_{n-1}) = i_{n-1}]. \end{aligned} \tag{2.15}$$

It can be shown, that only the negative-exponential distribution function fulfils the requirement of (2.15). Thus the time spent in each state is negative-exponentially distributed for Markov chains.

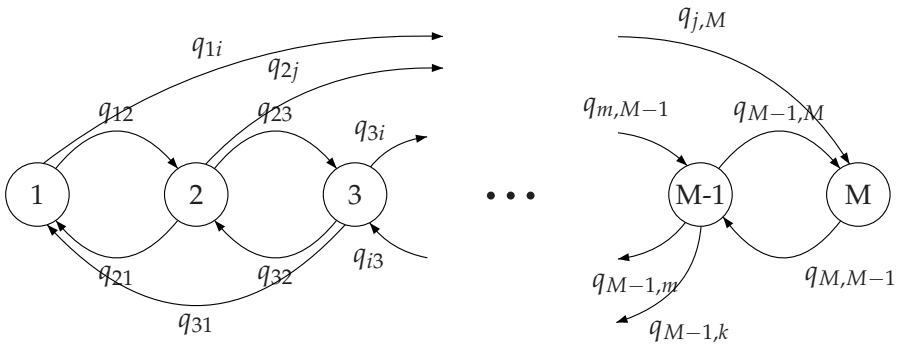


Figure 2.2: State-transition-rate diagram of a Markov chain

In figure 2.2 the state-transition diagram of an example of a Markov chain is shown. These state machines are usually used for visualisation. Note that from the state-transition diagram it cannot be concluded whether a continuous or discrete Markov chain is meant. Only the discrete *state space* is visible. Some further subclasses are mentioned here shortly:

- *homogeneous*: a chain is homogeneous if the probability to change from one state i to another j does not depend on the time we reach this state.
- *absorbing*: if at least one *absorbing state* is present in the chain, the chain is called absorbing; eventually the chain will "end" in such an absorbing state. An absorbing state cannot be left anymore, in other words $p_{ii} = 1$.
- *irreducible*: every state can be reached from *every* other state; this is especially *not* the case for birth-death processes. This means

$$\exists\, m \text{ so that } p_{ij}^{(m)} > 0 \;\forall\, i, j \tag{2.16}$$

  with m as the number of steps.

Discrete-time Markov Chains

For all Markov chain the state space is discrete (else it were a Markov process, compare chapter 2.2), but it is possible to have either discrete time steps on which we consider the chain development or a continuous time axis. The following is true only for the discrete time steps.

For discrete homogeneous Markov chains the probability transition matrix can be given, that gives the

probability for a transition from state i to state j in the next step.

$$\mathbf{p} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\ p_{31} & p_{32} & p_{33} & \cdots & p_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & p_{n3} & \cdots & p_{nn} \end{bmatrix} \tag{2.17}$$

Continuous-time Markov Chains

In contrast to the previous paragraph now Markov chains with continuous-time behaviour are presented in more detail.

For continuous-time Markov chains no transition probability matrix as in eq. (2.17) can be given, as transitions to the next state can occur any point in time. So all transition probabilities depend on a time parameter (for homogeneous chains, else we need two parameters: start time and time span), that means $p_{ij}(h)$ gives the probability that when starting from state i after time span h state j is reached.

$$\mathbf{p}(h) = \begin{bmatrix} p_{11}(h) & p_{12}(h) & p_{13}(h) & \cdots & p_{1n}(h) \\ p_{21}(h) & p_{22}(h) & p_{23}(h) & \cdots & p_{2n}(h) \\ p_{31}(h) & p_{32}(h) & p_{33}(h) & \cdots & p_{3n}(h) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n1}(h) & p_{n2}(h) & p_{n3}(h) & \cdots & p_{nn}(h) \end{bmatrix} \tag{2.18}$$

Each of these transition probabilities can be split up into two partial transitions when considering that for a transition to j at least some intermediate state k has to be traversed first (see figure 2.3). Hence it can be written

$$p_{ij}(t+h) = \sum_{\text{all k}} p_{ik}(t)p_{kj}(h) \tag{2.19}$$

or in matrix notation

$$\mathbf{p}(t+h) = \mathbf{p}(t)\mathbf{p}(h). \tag{2.20}$$

From these *Chapman*[3]*-Kolmogorov Equations* (2.19) the *Kolmogorov-Forward Equations* (2.21) can be derived.

$$\frac{d\mathbf{p}(t)}{dt} = \mathbf{p}(t)\mathbf{q} \tag{2.21}$$

where $\mathbf{q}$ is the *infinitesimal generator* or *transition rate* matrix. It gives the transition rates between all states and for Markov processes does not depend on the

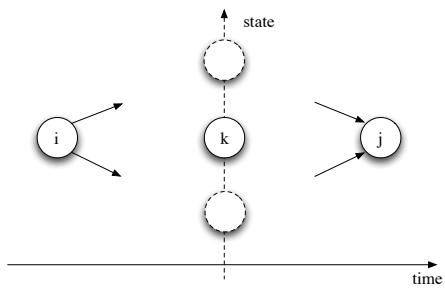


Figure 2.3: To reach state j from state i one of the intermediate states k must be passed

[3]Sydney Chapman; ⋆1888-†1970; British astronomer and geophysicist who derived a special case of this formula for a particular problem ([21][12])

time, but is constant. Henceforward the steady state probabilities $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ are given by

$$\boldsymbol{\pi}\mathbf{q} = 0. \tag{2.22}$$

Birth-Death Processes

The notation of birth-death processes refers in general to processes where state changes are allowed only between neighbouring states in the chain, i.e. the RV can increase (birth) or decrease (death) only by one in one transition. The birth (death) of twins is excluded, as they arrive also in series and not at the exact same time. Birth-death processes are a special case of Markov chain. This results in one of the most easy types of random processes. Figure 2.4 shows the resulting state diagram where no transition are possible except those between neighbouring nodes. Please note the $q_{i,i+1}$ is simplified to λ_i and $q_{i+1,i}$ to μ_i. Eq. (2.17) simplifies to the form given by eq. (2.23), since only transitions between direct neighbour

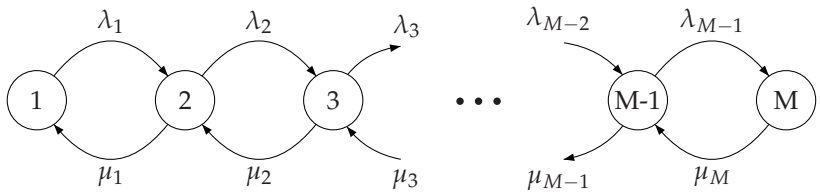


Figure 2.4: Birth-death process

nodes can happen. Thus all transitions with $|j - i| > 1$ are not existing and therefore their probabilities to occur zero and by this also their rate of transition.

$$\mathbf{q} = \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 & \cdots & 0 \\ \mu_1 & -(\lambda_1 + \mu_1) & \lambda_1 & \cdots & 0 \\ 0 & \mu_2 & -(\lambda_2 + \mu_2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mu_M \end{bmatrix} \tag{2.23}$$

Two special cases are the *pure birth process* ($\mu_i = 0\ \forall i$) and the *pure death process* ($\lambda_i = 0\ \forall i$). The pure birth process results in the *Poisson Process* for $\lambda_i = \lambda\ \forall i$.

2.2.3 The Erlang-k distribution

Using the facts of the previous sections, the Erlang-k distribution will be derived. The k-stage service model which is built by k negative-exponentially distributed stages can be considered as a pure birth Markov chain with a finite number of states. A slightly different graphical representation is used here as it is common to indicate the service rates into the circle representing the service unit, instead of drawing it above the arrow.

Recapitulating the pdf of the negative-exponential distribution and its LT

$$f_X(x) = \lambda e^{-\lambda x} \quad \circ\!\!-\!\!\bullet \quad f_X^\star(s) = \frac{\lambda}{s+\lambda} \tag{2.24}$$

a service unit is examined that operates according to this distribution. It is known that the service rate is λ and mean service time is $\frac{1}{\lambda}$. Figure 2.5 depicts this service unit. The "M" above the cirle stands for Markov and indicates that the pdf of the service time is a negative-exponential one. This single service unit will be the building block of the Erlang-k distribution. A new service unit is thought up, that is constructed by sequentially linked exponentially distributed service units. Figure 2.6 shows the new service unit (the whole square). As can be seen it consists of k Markov service units. In contrast to the single stage each service stage now has a service rate of $k\lambda$, i.e. it is sped up by a factor of k which is at the same time the number of stages in the block. So every stage has a mean service time of $\frac{1}{k\lambda}$ and to pass the whole block of k stages, a customer will have to spend in average $k \cdot \frac{1}{k\lambda} = \frac{1}{\lambda}$. We see that the mean time it takes a customer to pass through the system is the same. This

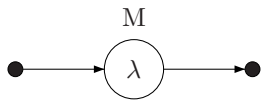


Figure 2.5: A simple negative-exponentially distributed stage

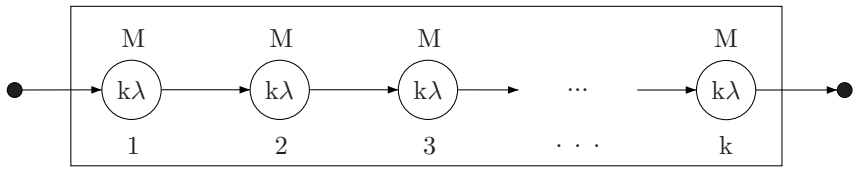


Figure 2.6: Series connection of exp. distr. stages: Erlang-k generation model

result can be derived also from the pdf of the new block. For that we have to derive the pdf of the block first by generalising the assumption that was stated some lines above, namely that the new RV Y which describes the whole block is the summation of k identically RVs X which are negative-exponentially distributed. This fact is stated in equation (2.25).

$$Y = X + X + X + \ldots + X \tag{2.25}$$

Sums of RV can be calculated by convolution in the original domain.

$$f_Y = f_X \circledast f_X \circledast f_X \ldots \circledast f_X \tag{2.26}$$

A much easier way to calculate the sum is by transforming the RVs into the Laplace domain first. In the Laplace domain the convolution is reduced to a multiplication. In our case all summands are the same and the k-fold multiplication simplifies to the kth power of the original pdf.

$$f_Y^\star(s) = \left[\frac{\lambda}{s+\lambda}\right]^k \tag{2.27}$$

By inverse LT the pdf in the time domain is obtained.

$$f_Y(y) = k\lambda(k\lambda y)^{k-1} e^{k\lambda y} \tag{2.28}$$

We can now calculate the mean value of this pdf and get the same result as before, $\frac{1}{\lambda}$. What is more

interesting is the standard deviation $\sigma = \frac{1}{\sqrt{k}\lambda}$. As can be seen σ gets smaller with growing k. In the limit for $k \to \infty$ the Erlang-k distribution equals a constant distribution.

To illustrate this behaviour the pdf (2.28) is shown in figure 2.7 for values of k = [1, 2, 5, 20, 100].

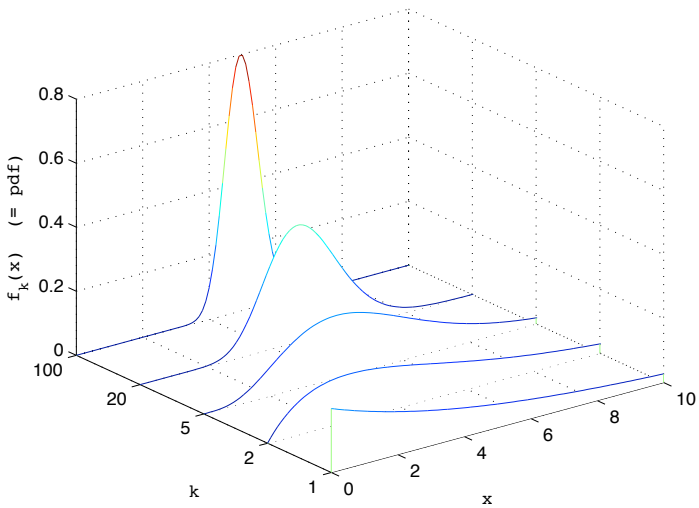


Figure 2.7: Erlang-k distribution for values of k = [1, 2, 5, 20, 100]

2.3 Semi-Markov Process

Opposing to the definition of a Markov process a Semi-Markov process is a process where the interevent times may be distributed arbitrarily. This means that the future development of the process depends not only on the current state but also on the time spent in this state. This means the probability to change to another state is given by

$$P[X(t_n) = j, T_n - T_{n-1} \leq t | X(t_{n-1}) = i_{n-1}]. \tag{2.29}$$

Note that this is not the most general RP as *not* the *whole* past is relevant for the further developing, but the current state *and* the time already spent there. In this more general case the transition rate matrix (cf. eq. (2.21)) *does* depend on the time and is no longer constant. To treat these processes adequately in our later studies two theorems are presented in the following that hold for general systems: the residual lifetime in a state once we start observing it and the "catastrophe process" which gives the probability of an event occurring during an observation interval.

2.3.1 Residual Lifetime

This description and derivation is analogue to references [15] and [13].

We are considering a renewal process with arbitrarily distributed interarrival times. Figure 2.8 illustrates a sequence of events. At a randomly chosen point in time one of the interarrival times will be observed. We are interested in the distribution of the selected interval (called *lifetime* in renewal theory) as well as the distributions of the *forward recurrence time*, also known as *residual lifetime*, and *backward recurrence*

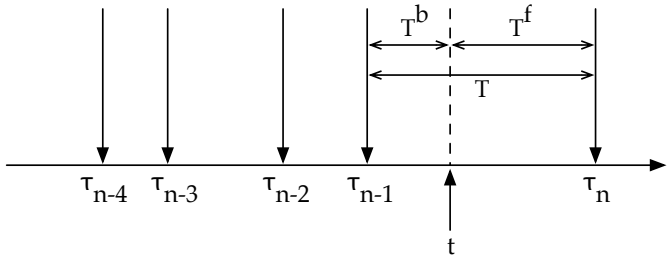


Figure 2.8: Renewal process with one interval to be observed

time, also referred to as *age*. We define RVs T, T^b and T^f for the length of the intervals shown. Further let

$$F(x) = P[\tau_{k+1} - \tau_k \leq x] \tag{2.30}$$

be the distribution of the interarrival times. The distribution surely does not depend on k as it is a renewal process, i.e. with independent and *identically* distributed RVs. To simplify matters later we also introduce $\lambda = 1/E[\tau_{k+1} - \tau_k]$. As we will see our special, selected (observed) interval is not a typical interval, as its distribution will be different from eq. (2.30). This is due to the fact that as an observer it is more likely to observe a longer interval than a shorter interval as the longer ones occupy more space on the time axis than shorter ones. Thus for our selected interval we define a separate distribution function

$$F_T(x) = P[T \leq x] \tag{2.31}$$

and further we define a distribution function for the residual lifetime:

$$F^f(x) = P[T^f \leq x]. \tag{2.32}$$

By reason we state that the probability to observe an interval of length x (which is $f_T(x)\,dx$, since $f_T(x)\,dx = dF_T(x) = F_T(x+dx) - F_T(x) = P[x < T \leq x + dx]$, that means it is the probability that T is in the "close range" of x) is proportional to the length ($f_T(x)\,dx \sim x$) and to the relative occurrence of intervals of that length ($f_T(x)\,dx \sim f(x)\,dx$). So the first intermediate result is

$$f_T(x)\,dx = K \cdot x \cdot f(x)\,dx. \tag{2.33}$$

In this eq. (2.33) we can see the proportionality to x and to $f(x)\,dx$ as well as a constant K that allows the normalisation of $f_T(x)\,dx$ as it is a probability distribution. Executing the normalisation leads to

$$\int_0^\infty f_T(x)\,dx \stackrel{!}{=} 1 = \int_0^\infty K \cdot x \cdot f(x)\,dx = K \cdot \int_0^\infty x \cdot f(x)\,dx = K \cdot E[\tau_k - \tau_{k-1}] = K \cdot \lambda^{-1}. \tag{2.34}$$

We see immediately that $K = \lambda$ and obtain the pdf for RV T:

$$f_T(x) = \lambda x f(x)$$

As next step we state that the position of our point t at which we start our observation is equally dis-

tributed within the interval $(0, x)$, i.e. $f^f(u|x)$ is uniformly distributed in the interval $(0, x)$.

$$f^f(u|x) = \begin{cases} 0 & \text{for } u < 0, \\ 1/x & \text{for } 0 < u < x, \\ 0 & \text{for } u > x. \end{cases} \tag{2.35}$$

Finally we apply the law of total probability and get

$$f^f(u) = \sum_{\text{all x}} f^f(u|x) f_T(x)\, dx = \int_{x=0}^{\infty} \frac{1}{x} \lambda x f(x)\, dx = \lambda \cdot F(x)\big|_{x=u}^{\infty} = \lambda\big(1 - F(u)\big). \tag{2.36}$$

Result (2.36) can also be given in the Laplace domain only by applying the well-known rules of LTs:

$$f^{f\star}(s) = \frac{\lambda}{s}(1 - f^{\star}(s)). \tag{2.37}$$

Using (2.37) easily the mean residual lifetime can be obtained. Here, only the result is given in terms of the mean and variance of the original distribution:

$$E[T^f] = \frac{E[\tau_{k+1} - \tau_k]}{2} + \frac{VAR[\tau_{k+1} - \tau_k]}{2 \cdot E[\tau_{k+1} - \tau_k]}. \tag{2.38}$$

Eq. (2.38) is also known as the residual lifetime paradox, as its result is counterintuitive.

The above given equations are some of the most fundamental results of renewal theory. The residual lifetime pdf will be used in the analysis of the location update model later.

2.3.2 Catastrophe Process

The notion of Kleinrock is adopted here, which uses the term event and "catastrophe" to distinguish the two kinds of events, nevertheless a catastrophe is still an event, but one unrelated to the other kind of events and with a peculiar distribution.

The pdf of the events is named $v(t)$. The pdf for the occurrence of the catastrophe shall be $f(t)$. The question that is of interest here is what be the probability for one event occurring before the catastrophe. We define this probability to $P[B]$. Further we define $P[A_t]$ = "probability that an event occurs 'at' time t". In other words

$$P[A_t] = v(t)\, dt \tag{2.39}$$

Let further $P[B_t|A_t]$ be the conditional probability that *no* catastrophe happened until time t under the condition that at time t an event occurs. This probability is of course given by the complementary distribution function $F^C(t)$. By applying the law of total probability (2.11) we get

$$P[B] = \sum_{\text{for all t}} P[B_t|A_t] P[A_t]. \tag{2.40}$$

Inserting the above stated relations for $P[B_t|A_t]$ and $P[A_t]$ this results in

$$P[B] = \int\limits_{\text{for all } t} F^C(t)v(t)\,dt = \int\limits_0^\infty F^C(t)v(t)\,dt. \tag{2.41}$$

For the special case of $f(t)$ being a negative-exponential pdf, we can insert $F^C(t) = e^{\lambda t}$ and so get

$$P[B] = \int_0^\infty e^{\lambda t}v(t)\,dt = v^\star(\lambda) \tag{2.42}$$

with $v^\star(\lambda)$ being the LT of the pdf which equals to the probability .

3 Mobile Networks

In the following sections an overview to mobile networks is presented. Mobile networks are quite complex systems. The specifications of todays operated networks comprise tens of thousands of pages of details. That is why no real overview of these complex systems can be given. Rather those functionality and architectural aspects are stressed which are of relevance for the understanding of the further chapters about Mobility Management.

3.1 Network Entities and Network Architecture

In this section the basic architecture of a mobile network will be presented as well as the elements it is made up from.

In figure 3.1 the basic architecture of the UMTS network is shown. It stands representatively for any modern cellular network as the structure would be similar. The figure shows the circuit switched (CS) part of the network. The packet switched (PS) part is slightly different, not only in the names of the network elements, but also for their functionality.

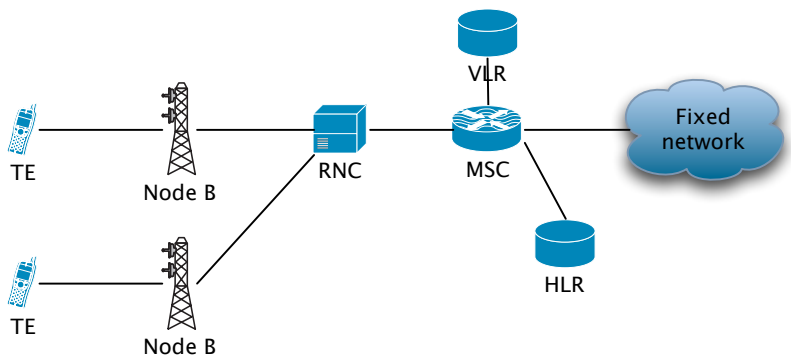


Figure 3.1: Overview of the UMTS CS architecture

In the following a description of the *network elements* (also called entities) is given (partially taken from [34]).

TE The *Terminal Equipment* is commonly known as mobile phone. The term here is used more generally - it can be any kind of mobile device. In UMTS specifications it is actually called *User Equipment* (UE) and in the literature it is often called *Mobile Terminal* (MT). It is the one hand endpoint for functions from all layers of the protocol stack. On the other side the network terminates the functions in different entities for the different layers. But the TE has to accomplish all functions from physical layer up to the application layer for both user and control planes as well as for management plane.

Node B The *Node B* is basically the network element for physical (PHY) layer processing of RF related functions. Its function can be described in short as the interface between RF transmission and wired transmission. This network element is known as *base station* (BS) in GSM networks.

Its tasks comprise:

- Signal Processing
- Measurements for RNC
- (Inner Loop) Power Control
- RF modem

RNC The *Radio Network Controller* is in command of several Node Bs and directs them according to the core networks requests which it gets from the MSC. It is the central node of the radio access network. It manages the calls and radio resources for all the Node Bs which are assigned to it.

Its main duties include:

- Call Admission Control (CAC)
- Radio Resource Control (RRC)
- Radio Link Control (RLC)
- (Outer Loop) Power Control
- Handover control
- Soft Handover
- Ciphering

MSC In short words the *Mobile Switching Centre* is a switching node which supports Mobility Management for the connections it routes.

Its duties - among others - are:

- Routing of calls
- Involved in LU procedures and handover

VLR The *Vistor Location Register* is a database attached to and supporting the MSC. The VLR stores the location area (location areas will be explained later) for a TE if the TE resides in the domain of the VLR. As soon as a TE moves into the domain of the VLR, it also retrieves a copy of the user information from the HLR. In short, it stores the following data:

- Location information about local TEs
- A copy of the HLR information for all the local TEs

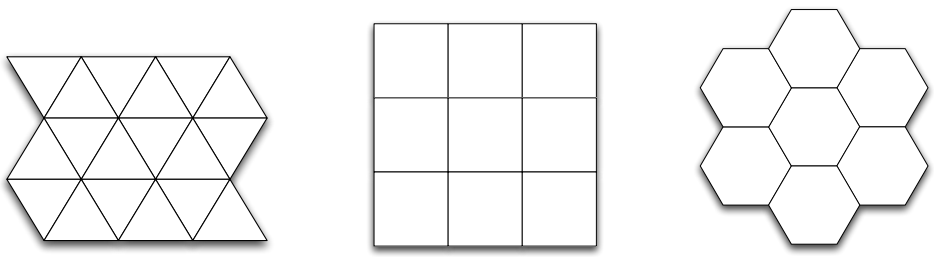
HLR The *Home Location Register* is the central database in the network for user information. When a user is introduced to the network, the users personal data and access rights are stored there. During operation the HLR furthermore stores a reference to the VLR where the TE is currently located. The tasks of the HLR are summarised here:

- Database of the user data (IMSI, access rights, services, ...)
- Involved in authentication procedures
- Knows current VLR

At this point we will not delve any deeper into the architecture but will approach mobile networks from a different point of view.

3.2 Cellular Principle

In contrast to broadcast systems (one-to-many relation) mobile networks realise a one to one connection between communication partners. As the information exchanged between the communication terminals is individual for all pairs of users, a certain amount of transmit capacity is needed. To realise this four kinds of multiplexing methods are used: frequency division, time division, space division and code division. Radio frequency bandwidth is a rare resource and every system is assigned a part of the total usable frequencies. In GSM 900, for instance, the allocated bandwidth is 25 MHz [6], per direction as it is a duplex system. These 25 MHz are divided into 125 channels of 200 kHz each. By a time multiplex method eight mobiles can use one channel at the same time. We see that 1000 users can use the system at the same time. Considering the millions of subscribers this is fairly few and the third "dimension", the space division comes into play. The 1000 people capacity that we calculated to be able to be phoning at the same time can be replicated manifold over the world. This is a result of the decay of the RF intensity with increasing distance from the base station (BS) where the relation $P_{RF,\,free\ space} \sim d^{-2}$ hold, which shows the received RF power dependency on the wave length and on the distance of the transmitter for free space. If not free space propagation is considered the dependency is no longer quadratic but follows the relation $P_{RF,\,real\ prop} \sim d^{-\gamma}$ with $2 \leq \gamma \leq 5$ (cf. [30]). After a particular distance the interfering RF power level is so small that it does not disturb any more. At this point we can reuse one of the 125 frequency channels stated above without any drawbacks. In reality we will accept a reuse at the distance where the Signal-to-Noise-Ratio (SNR) (also known as Signal-to-Interference-Ratio, SIR or Carrier-to-Interference-Ratio, CIR) is big enough. How big actually *is* "big enough" can only be answered in context of the particular transmission system used. For example in GSM a CIR of 18 dB is aimed at, whereas in UMTS even -20 dB can be handled (cf. [30][34]). For the sake of completeness the opposite principle of covering an area with RF is called *common-wave broadcasting* or *single frequency network*. Even though UMTS for example can be operated as a single frequency network, still the cellular principle is applied but the reused resource are the channel codes, not the frequencies.



(a) Tessellation of triangles (b) Tessellation of squares (c) Tessellation of hexagons

Figure 3.2: Regular tessellation shapes

To model the cells that make up the covered area usually hexagons are used. Note that only three regular[1] congruent[2] *tessellation* shapes are possible: triangles, squares and hexagons. All other regular shapes do not fulfil the requirement to divide the 360° of the plane at each vertex without rest [4][35].

[1]regular: the edges of the polygons have the same length
[2]congruent: the shapes for the tessellation are all the same size

The three possibilities are shown in figures 3.2(a)-3.2(c). Triangles divide the plane in $6 \cdot 60°$, squares in $4 \cdot 90°$ and finally hexagons in $3 \cdot 120°$.

Let us now consider what a cell geometrically looks like, so we can decide which one of the three choices to select. A cell is an area fed with some radio frequency (RF) channel(s) according to the mobile network standard in use. The RF is generated by a base station which usually also has some further processing functionality. Under the assumption that a dipole antenna were used the radiation pattern yielded would be concentrical to the antenna. This behaviour is illustrated in figure 3.3(a) which would be according to the ideal radio frequency propagation. The RF field strength is indicated with the shade of

(a) Ideal cell (b) Real cell (c) Model cell

Figure 3.3: Cell approximations

grey, which obviously takes its maximum value in the centre where the antenna is located and decreases with growing distance as a cause of the free space attenuation (as a result of the observation that the signal strength per area decreases as the power of the propagated wave naturally cannot augment, but the area the power is distributed to is growing with growing distance from the centre). In the real world however we cannot take into consideration the free space losses but also losses due to diffraction, reflection, scattering and absorption, to name just a few effects. Hence the radiation pattern and by this also the covered cell area may look entirely different and asymmetrical. Figure 3.3(b) gives an example how a cell may look like. For densely built up urban areas the shape of a cell is usually very dependent on the course of the streets and has a totally irregular shape.

Figure 3.4 gives an example for the RF propagation in such a cell. It shows a part of the city of Stuttgart, Germany. Despite the irregular real-life shapes the hexagonal layout is assumed area-wide for simplification.

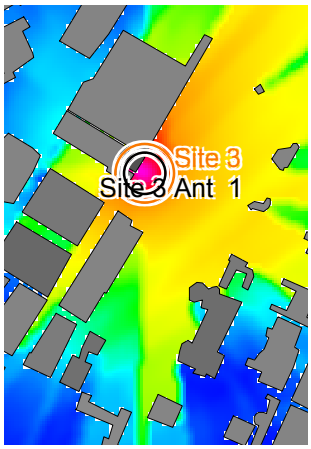


Figure 3.4: RF propagation in urban area - buildings in grey, field strength decreasing from red to blue; generated with WinProp of AWE Communications

As we see the hexagon (cf. figure 3.3(c)) is the best approximation to the RF propagation cell shape, as it approximates best the circle. Also for sectored cells, where a base station supplies several (usually three) antennas with RF, the hexagonal model maps best, as the three sectors are in line with the three times 120° in the hexagon tessellation, as described earlier. We then consider the antennas to be seated in the edge between hexagons opposing to the other layout where the centre of the hexagon was regarded the location of the antenna.

For the above given reasons most cell models in literature the hexagon is used. In some cases it is simpler to consider squares as basic cell shape, but throughout this work this is not the case.

Above, it was explained why the transition from single fre-

quency networks to cellular networks was the logical consequence of an increased bandwidth demand in the network. On the other hand to allow users to move freely between cell boundaries without them noticing, some kind of functionality enabling this has to be provided by the network. These procedures are usually subsumed under the heading Mobility Management. The following chapter will explain these procedures.

3.3 Mobility Management

3.3.1 Introduction

In cellular network methods are necessary that allow users to move freely among cells. The methods and procedures allowing this are subsumed under the headword Mobility Management (MM). In the following the big field of MM is unravelled a bit. For this purpose figure 3.5 will be taken as guidance.

The first distinction to make is whether mobility during the idle phase or during the call phase of the mobile is meant. Idle in this context means, that there is no call ongoing. The mobile is not totally idle in this phase though, as the below explained location management procedures (among others) are still executed.

In the idle phase the mobile is tracked by the network. This is possible as the mobile keeps the network informed about its current location. The purpose of the tracking is an efficient call delivery in case of an incoming call. Be reminded that in a cellular network the cell has to be found out to which the incoming call has to be delivered. So Location Management can be divided into the Location Update part and the Call Delivery part. During the location update procedure the network is informed about the current location area (LA) the mobile resides in. This information is stored in the VLR associated with this LA. If a call is coming in (this is called a *mobile terminated call*, as the origin is in the network and the destination of the call is the MT) first the HLR is queried for the current VLR and then this VLR is queried for the current LA. This phase is subsumed in the "Database Query" box in figure 3.5. The last phase before leaving the idle state of the MT is the paging procedure. As the LA consists of several cells, in all those cells a paging call is issued on a special paging channel to which the mobile listens periodically. After having received the paging call the mobile responds and the call can be put through - the call phase initiates.

To allow mobility of the MT during a call, additional procedures have to assure that the connection does not break off during the crossing of cell boundaries. Remember that the cell boundary is not a fixed line. Only due to measurements of the signal to interference ratio (SIR) can the mobile decide when it has left one cell and enters another (these measurements reflect the network conditions RF-wise). It is then better to receive and transmit data from and to the new BS/Node B. The procedure supporting this transition from one cell to the next while a connection is established is called *handover* or sometimes *handoff*. The handover is initiated by the BS, based on the MT's measurements, and is then announced to the MT via a signalling message.

After this initiation phase the new connection generation phase follows. This phase is composed of two parts: first the radio resources in the new cell have to be allocated for the "new" call. At this point the transition might fail as all radio resources in the cell the MT moves to might be occupied already. This would be quite likely, would the mobile operators not take precautions like reserving a small part of the cells capacity for handoffs from neighbouring cells. Call Admission Control (CAC) for that cell (in the

RNC) will not allow the reserved capacity to be used for calls originating from that cell.
Thus assuming that the resource reservation was successful the next step is to reroute the connection. In the example of an UMTS network this would be in either the RNC or, after a SRNS relocation, in the MSC. Related to the connection routing is data flow control. Data flow control has to secure that during switching from one cell to another as little data as possible is lost. This is accomplished by buffering and multicast of data, which means, that when the connection is rerouted the incoming data is buffered and after successful establishment of the connection in the new cell delivered or, in systems where it is possible to maintain connections with the MT via several BSs, to distribute data to all currently connected cells. To prevent data from being received or sent twice, sequencing is used.

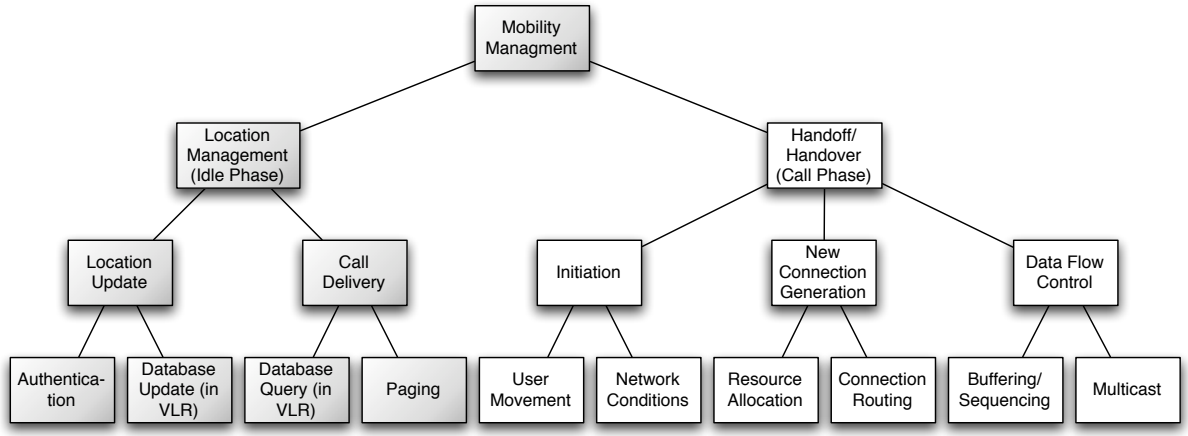


Figure 3.5: Overview of Mobility Management functions (acc. to [3]); the shaded fields are considered in more detail later

Having traversed now the whole graph in figure 3.5 top-down from left to right, we have a global view of what includes MM. The grey shaded areas are covered in more detail in the following sections, as we are interested mostly in LM in this work. However, to optimise future MM procedures it is possible to combine information from LM and HO parts and not to consider them completely isolated any more.

3.3.2 Location Update (LU)

In this paragraph Location Update mechanisms are discussed in detail. As even only this part of the network is a complex topic in itself, to start with, as in the previous sections, an example sequence diagram is given which illustrates a LU procedure. In the paragraph thereafter the LU strategies are discussed which trigger the LU procedure.

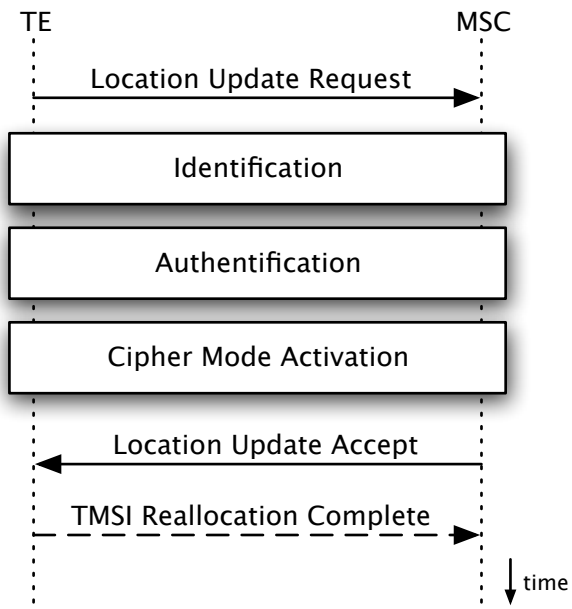


Figure 3.6: Mobility Management Protocol excerpt from [7]

Location Update Procedures

Location Update procedures are necessary to allow the MT to inform the network about its current location. In the context of this paragraph *procedure* is another word

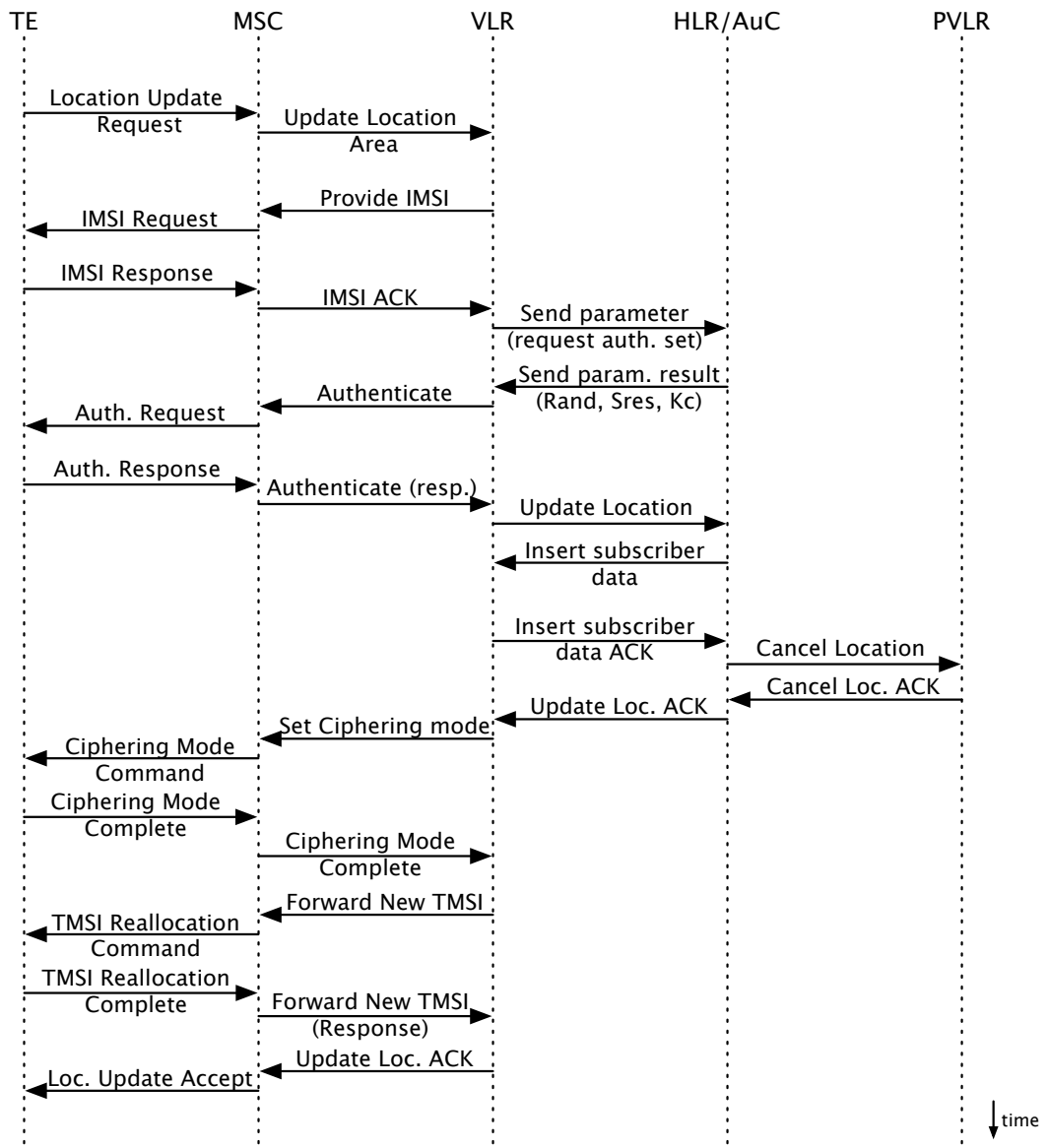


Figure 3.7: Location Update sequence diagram (based on [32])

for *protocol*. It describes the fact that between two of more entities information (via messages) has to be exchanged according to predefined rules.

To start with, a sequence diagram of the MM protocol of GSM and UMTS is shown in figure 3.6. Yet in this diagram for clarity only the very basic messages are shown, namely Location Update Request and Location Update Accept, while the remaining part is given as blocks which show the necessary tasks to end up with a successful LU. Although this diagram illustrates an actual protocol, the blocks that are shown would look similar in any comparable system as the functions are necessary for a secure operation of the network.

During the ...

- *identification* phase the MT has to identify itself for the networks database to be access and update the right database entry
- *authentication* phase the MT has to prove that the identity it announced was the correct one, in

order that no MT can spoof LU messages
- *ciphering activation* the ciphering is activated to ensure link security.

Authentication and ciphering can be optional in some networks as it does not provide protocol functionality but security.

For an example of a more in-depth LU sequence diagram see figure 3.7. There the interactions between (some of) the involved network entities can be seen schematically. On the horizontal axis the network entities are shown and the downward vertical direction represents increasing time in the sequence. In the diagram can be seen, that the LU procedure is initiated by the MT. The MT knows when to initiate the LU procedure by its implemented LU strategy (see below). It then establishes a radio link with the BS (which is not shown) and sends a LU Request message to the MSC responsible for the location area. The MSC relays the request to the VLR, whereas the VLR is usually located together with the MSC. The VLR then asks for the IMSI to identify the MT. After having obtained the information via the MSC, the MSC/VLR ask the HLR/AuC for the authentication information for the respective IMSI. To this end the MSC has to establish a connection to the HLR which can be in another network than the MSC/VLR. Global title translation (SS7 functionality) is used to find the destination. In the AuC a random number RAND is generated. RAND and the users deposited secret key Kc are used by a one-way function A3 to get SRES (Signed RESult). These three values are then transmitted to the MSC which challenges the MT with the same random number RAND. If the MT is the one it pretends to be (by its IMSI) it should be able to generate the same result SRES. If so, the VLR tells the HLR that now it is responsible for the MT and therefore gets a copy of the users database entry from the VLR (Insert Subscriber Data). After acknowledgement the HLR tells the previous VLR (PVLR) that it is no longer responsible for the MT. The ciphering mode is enabled in the next step and the answer of the MT is usually the first ciphered message. A TMSI (Temporary Mobile Subscriber Identity) is generated for the MT to prevent the IMSI from being sent too often, to make it hard to track a MT. After that the LU procedure ends successfully with an LU-Accept message. For more details on this procedure it is referred to [19, p. 465 et sqq.].

Location Update Strategies

In this section some of the possible ways to maintain the location known are shown. These methods are usually called *location update strategies* or *location update schemes*. First a general overview is given (cf. figure 3.8). There a classification scheme for location update methods is presented. Then the methods used in today networks will be presented. Further ideas from the literature are shown and related to the introduced classes.

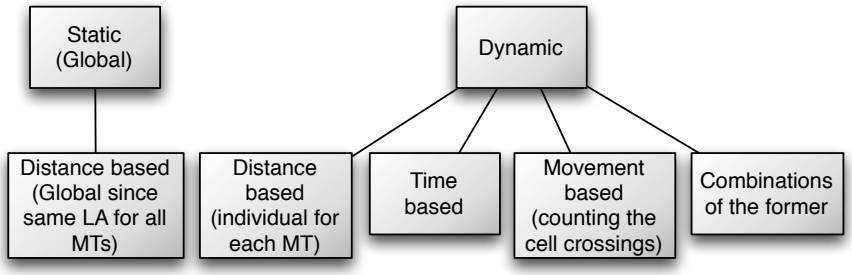


Figure 3.8: Overview over the different LU strategies

Static scheme In todays mobile networks the static/global methods are used. The whole network is split up into *location areas* (LAs). The identifier (ID) of the LA is broadcast by all BS/Node B of the respective LA. According to [36], "each MT compares its registered LA ID with the current broadcast LA ID. Location update is triggered if the two IDs are different." Wong and Leung further note the following essential drawbacks of this widely used scheme:

- For large LA paging is very costly (because of the blanket polling scheme, see section 3.3.3). For the high number of users in future mobile networks, problems might arise.
- Ping-pong effect: if a MT resides between two LA it is possible that by the moving back and forth of the MT many unnecessary LU are triggered.
- The scheme is applied to all users, and the LA is the same for all users. This might be inefficient as this does not allow for the different mobility behaviours of the different users.

Distance-based The distance-based scheme can be compared with a dog on a chain. The dog is allowed to move circularly around the tree where its chain is attached to, within the range of the length of the chain d. Similarly the MT can move around freely, without neither restrictions with respect of numbers of crossed cells, nor the time it spends within the area. But once it leaves the area, i.e. it gets farther from the last point of contact with the network than d, the MT has to send a LU message immediately.

Time-based In time-based methods, the MT has a timer which measures the time since the last contact with the network. If the timer value exceeds a threshold an LU is issued. This method is also used in todays networks, but not as principle scheme but only as secondary mechanism with very long timeout values. The purpose is to keep the network in a consistent state by assuming the MT to be inactive if no LU has occurred for a time longer than the timeout. The MT is then removed from its last VLR.

Movement-based In the movement-based scheme the MT maintains a counter which counts the crossings of cells since the last contact with the network, i.e. since the last LU or since the position of termination of the last call. As soon as the counter exceeds a given threshold (which might be adaptable to the users mobility behaviour) a LU is issued.
The advantage of this method lies in its simplicity, while still being an improvement over the static method.

Profile-based During operation of the network, the network can gather information about the preferred LAs a MT is roaming in, which are then transmitted to the MT as a list. As long as the MT resides within cells on the list, it doesn't has to send any LUs. Only when it moves to a cell not on the list, does it send an LU message. The size of the list, i.e. the number of cells stored in the list is subject to optimisation.

Combinations of different strategies Combinations of different strategies are possible. Sometimes these methods are called *state-based schemes* as the LU is initiated when a certain state is reached, where the state is defined by different conditions of the basic LU schemes explained above.

3.3.3 Paging (PG)

In case a call is coming in the mobile terminal has to be located in the network. This is accomplished by a polling procedure which might look like the following:

1. use of the paging channel of the cell to inquire the mobile to reply to the paging call
2. if the mobile answers the location is known and the call can be put forward/through to the MT
3. if the mobile does not answer the inquiry until a timer expires, the MT is not located in the paged cell

If the outcome of the paging was negative (step 3, timer expires), the procedure has to be repeated until the MT is found in any cell. Figure 3.9 shows the polling cycle in a Nassi-Shneiderman diagram.

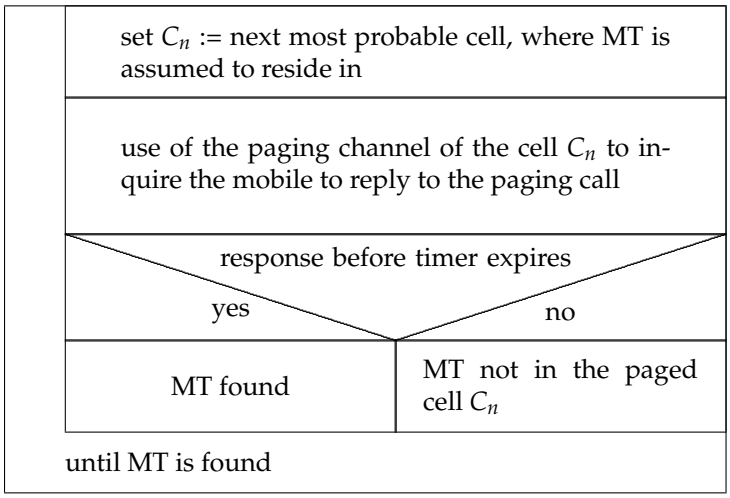


Figure 3.9: Polling cycle (Nassi-Shneiderman diagram)

It is obvious that the mobile is found quickest, if we page the cells first, which have the highest probability to house the MT. This plausible fact that the paging costs are minimised if the most probable locations are paged first was also proven in [27]. To minimise the number of required polling cycles spent per MT is not only of importance for a minimal network signalling message load but also to keep the *paging delay* (or polling delay) small. A small paging delay results in a fast connection setup.

We can already see a tradeoff here, namely the contradiction of fast paging (i.e. small delay) and low traffic in the signalling network. The fastest possibility to get known about the location of a MT would certainly be to page *all* cells of all LA simultaneously. With a delay $D = 1$ the MT is found, but at much too high signalling effort. Paging one cell after another on the other hand might be too time-consuming, as there are constraints on how long a call setup may take. This leads to the possibility to page groups of cells (or whole LAs) simultaneously. But in either case, if we page single cells or groups of cells, the most probable area should be tried first. This would mean to know the probability density function of the location of a MT.

This short introduction to paging mechanisms is given only for the sake of completeness. We will be more interested in the first part of the mobility management, i.e. location update, later. However to calculate the total cost of mobility management (in terms of signalling effort) we will have to consider both LU and PG. In the following some strategies are presented.

Blanket Polling According to [36], "in *blanket polling*, all the cells within the LA in which the MT is located are polled simultaneously when a call arrives. Since the MT is located within the LA, its location can be determined within a single polling cycle." The blanket polling strategy is the PG scheme currently in use for networks which use the static LU scheme. For large LA these methods gets very costly as more cells are paged than necessary for finding the MT.

Sequential Paging In this scheme the cells are paged for the MT sequentially, beginning with the cell of highest probability of presence of the MT. This residence probability can be derived by either the network keeping track of the most probable location (in conjunction with the profile-based LU scheme, see above), or by estimating the route of the MT. The pure sequential PG has the drawback of high and unconstrained delays. If cells are not paged one after another but cells are grouped together the scheme remains the same and the group with the highest MT residence probability is paged and so on. By this the delay is n-times smaller, if n is the group size in cells. The resulting scheme is called *sequential group paging*.

Shortest-Distance First If no other information about the location pdf of the MT is available, it is assumed that the nearer a cell is to the last cell a LU message was sent from, the more likely it is to find the MT there. This is called *Shortest Distance First (SDF)* paging. In other words, the SDF is a MPF under the assumption that the likelihood to find a MT decreases with increasing distance from the last LU. In practice this means that starting from the last LU cell all cells with distance $d = 1$ are paged, then all with $d = 2$, $d = 3$ and so on. In a regular hexagonal shape this means that the circles around the last LU cell are paged. The costs for PG is then $C_{PG}(D) = 3D^2 + 3D + 1$ if the MT resides in ring D around the cell of the last LU.

The SDF scheme is the one used throughout this work to calculate the PG cost.

Group Polling All the above mentioned PG schemes poll MTs one after another, i.e. they use the general polling cycle of figure 3.9 on a per-MT basis only. However, in a highly loaded network it might be a waste of RF capacity to page the MTs individually. A significant gain can be expected, if several users are polled at the same time. In [20] the proposed use of a hash-based bloom-filter method permits for a resource saving PG, especially well performant in future mobile networks.

3.3.4 Mobility Models

The behaviour of an MT with respect to space and time can be modelled by so called *mobility models* which try to model a realistic yet simple user behaviour. In the following a brief overview of the used models is given.

Two groups of mobility models can be distinguished:

1. Models with individual movement behaviour
2. Models with aggregate movement behaviour

The most prominent model of the individual movement class is the (symmetric) random walk model. In this model the MT has an equal probability to move to any other neighbouring cell, i.e. without any preference it moves to any neighbour cell (cf. figure 5.1). This model is also the most simple one, with the drawback of not taking into account any direction.

Another possibility is the use of a so called *Markov model* with or without history. In this model each transition is assigned an individual transition probability $P(j|i)$ given the probability to move to cell j under the condition of being in cell i currently. This models the users movement preference. Furthermore the history can be considered, which means that the set of cells traversed is taken into account for determining the probability to move from i to j.

In the aggregate movement models not an individual user is considered but the totality of users, the *flow* of users. Here the *fluid flow* and *gravity* models are mentioned but not explained. In these models the flow of MT out of and into an LA is given as traffic rate.

For all models, the time spend in a cell has a pdf $f(t)$. Usually and especially for analytical analysis $f(t)$ is assumed to be exponentially distributed.

4 Proposed Mobility Management Schemes

In this chapter the proposed LU methods[1] from [9] are explained. The methods neither belong to the class of pure time-based methods nor to the class of pure distance-based methods, as introduced in chapter 3.3.2, but they blend these two concepts. The proposed methods therefore can be called time-distance based strategies. In case of the modified versions, which will be explained in the respective subchapters, we can call the methods time-distance based strategies with a movement component.

In [9] it is further explained, that the paging area is not bounded in time-based LU methods. Thus blanket polling cannot be used. Therefore the Shortest Distance First PG scheme is applied in conjunction with the proposed LU methods.

4.1 Original Scheme

In the original timeout-based strategy, as already explained in chapter 3.3.2, the LU is triggered after a timer with predefined timeout value T_{LU} expires. The timer is then reset to the original value T_{LU}. The advantage of this method lies in its simplicity and in its adaptiveness to the users profile. Users with a high movement rate could have a shorter timeout period than users with a low movement rate. But to exploit this advantage additional functionality had to be implemented in the network to log previous user behaviour to derive the optimal T_{LU} for that user. On the negative side the method allows not to give an upper bound for the paging area, as the MT could move indefinitely far during the time T_{LU}. This holds true for all the following methods.

The original scheme is included in the performance analysis of this work to have a reference to evaluate the improvement of the proposed new schemes.

4.2 Modified Scheme

In the original time-based scheme the LU message is sent immediately after the timeout occurred. In the modified scheme however the next crossing of a cell boundary is waited for. That means that the MT can move indefinitely long within its currently residing cell *after* the timeout had happened and only when the MT crosses the border to a neighbouring cell the LU is issued. This further reduces the signalling effort by cutting down the LU messages whilst at the same time keeping the same PG cost like for the original scheme. This method was originally proposed in [18]. The modification introduces a kind of movement component with movement threshold $m = 0$, i.e. the next crossing lets the LU happen.

This scheme will be modelled and analysed in chapter 4.2.

[1] NB: LU method, LU scheme and LU strategy are used synonymously throughout this work

4.3 Stop & Wait Scheme

In [9] two new schemes are proposed to further reduce the signalling costs in mobile networks. The first one is discussed here, for the Reset & Wait Scheme see later in section 6.6.6.

The scheme is described in [9] as follows: "Each MT stores the non-empty set of X of cells with a distance of H, in terms of cell, from the last contact cell, and has an LU timer, hereafter called timer. [...] the timer is stopped (or frozen), when the MT moves between or in cells in this set X of cells. [...] The timer is running and enabled when the MT is roaming outside the set X cells. When the timer expires, the MT is always located outside the set X of cells, and it sends an LU message. Each time the MT makes or receives a new call or it sends an LU message, the set X of cells stored in its local memory is updated according to the last contact cell and the timer is reset to zero."

The behaviour of the timer is depicted in figure 4.1. The timer is initialised with the timeout value T_{LU} and then the timer is started. Clearly, the remaining time until the next timeout is decreasing linearly. Only when the MT re-enters the local area (as the non-empty set of cells X will be called in the following), the timer is stopped (frozen) that means the time until next timeout does not change, which is depicted by the flat areas in the figure. After leaving the local area once again, the timer still has the same value as on time of entering the local area earlier. This moving back and forth into and out of the local area can happen multiple times, but at some point in time when the MT is *outside* the local area the timer triggers and the LU message is sent.

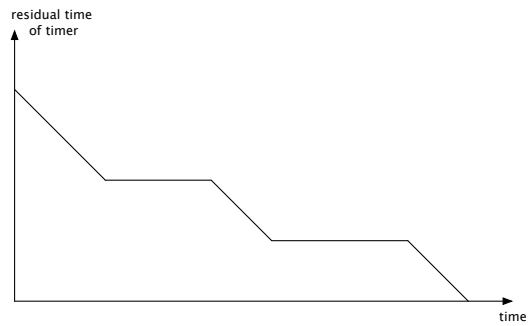


Figure 4.1: Residual time of the timer until next timeout; when the MT resides inside the local area the timer is stopped and thus the residual time until the next timeout persists at the same level

For further illustration figure 4.2 explains the scheme graphically. The MT starts from the centre point in the graphic (◉). First the MT is moving through the grey area, which marks the local area as described above. The timer is not enabled yet in this area. At point (a) the MT leaves the local area and the timer is started. At point (b) however the MT re-enters the local area and the timer is stopped then and the value of the timer is maintained as shown in figure 4.1. After leaving again at point (c) the timer is resumed. Finally at (d) the timer expires and the LU procedure is initiated.

The Stop & Wait scheme will be modelled and analysed in chapter 6.6.4.

4.4 Reset & Wait Scheme

The scheme is described in [9] as follows: "Each MT stores the non-empty set of X of cells with a distance of H, in terms of cell, from the last contact cell, and has an LU timer, hereafter called timer. [...] the timer is disabled and reset to zero when the MT moves between or in cells in this set X of cells. [...] The timer is running and enabled when the MT is roaming outside the set X cells. When the timer expires, the MT is always located outside the set X of cells, and it sends an LU message. Each time the MT makes or receives a new call or it sends an LU message, the set X of cells stored in its local memory is updated

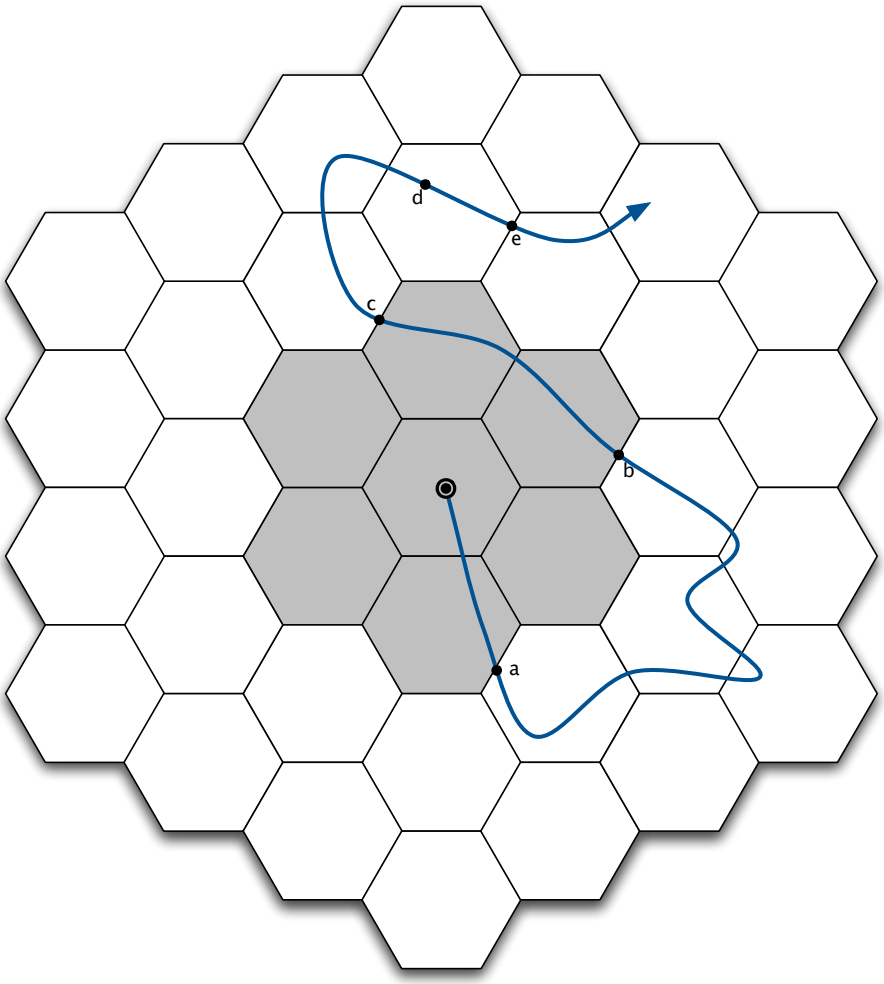


Figure 4.2: Cell mosaic with the local area (shaded grey) and sample movement path of a MT

according to the last contact cell and the timer is reset to zero."

Again for illustration figure 4.2 explains the scheme graphically. The MT starts from the centre point in the graphic (◉). First the MT is moving through the grey area, which marks the local area as described above. The timer is not enabled yet in this area. At point (a) the MT leaves the local area and the timer is started. At point (b) however the MT re-enters the local area and the timer is then stopped and reset to the original value T_{LU} as shown in figure 4.3. After leaving again at point (c) the timer is restarted again. Finally at some point in time, e.g. at (d), the timer expires and the LU procedure is initiated.

This scheme will be modelled and analysed in chapter 6.6.6.

4.5 Modified Stop & Wait Scheme

In this newly proposed method the Stop & Wait strategy is combined with the wait-for-cell-crossing strategy from [18] which is described above (section 4.2, Modified Scheme). This results in the now-called "Modified Stop & Wait method".

For clarification figure 4.2 can be used again: The MT starts from the centre point in the graphic (◉).

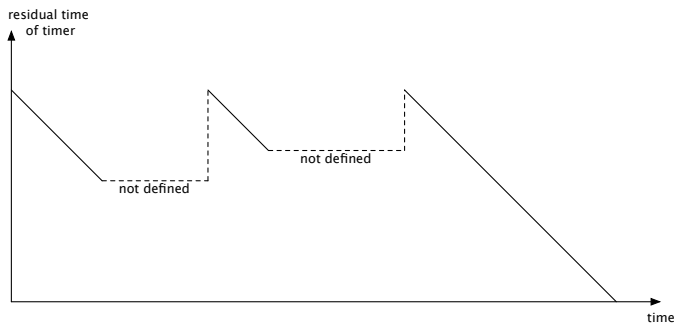


Figure 4.3: Residual time of the timer until next timeout; when the MT gets into the local area the timer is reset and thus the residual time after leaving the local area once again starts at the initial timeout value

First the MT is moving through the grey area, which marks the local area as described above. The timer is not enabled yet in this area. At point (a) the MT leaves the local area and the timer is started. At point (b) however the MT re-enters the local area and the timer is then suspended and its value maintained, just as in the unmodified scheme. After leaving again at point (c) the timer is resumed again. Finally at some point in time (d) the timer expires. However in the modified scheme *no* LU is sent immediately but the MT can move through the currently residing cell. Only when he enters a surrounding cell (indicated in the figure by point (e)) the LU procedure is initiated.

The effect of the combination of both methods should result in a further reduction of signalling in comparison with each of the methods alone.

This scheme will be modelled and analysed in chapter 6.6.5.

4.6 Modified Reset & Wait Scheme

In this newly proposed method the Reset & Wait strategy is combined with the wait-for-cell-crossing strategy from [18] which is described above (section 4.2, Modified Scheme). This results in the now-called "Modified Reset & Wait method".

In figure 4.2 at first the MT is moving through the grey area, i.e. in the local area. The timer is not enabled yet in this area. At point (a) the MT leaves the local area and the timer is started. At point (b) however the MT re-enters the local area and the timer is then stopped and its value is reset to the original value T_{LU}, just as in the unmodified scheme. After leaving again at point (c) the timer is resumed again. Finally at some point in time (d) the timer expires. However in the modified scheme *no* LU is sent immediately but the MT can continue moving through the currently residing cell. Only when he enters a neighbour cell (which is about to happen at point (e) in the figure) the LU procedure is initiated.

The effect of the combination of both methods should result in a further reduction of signalling in comparison with each of the methods alone.

This scheme will be modelled and analysed in chapter 6.6.7.

4.7 Movement-Distance Scheme

In the following for sake of completeness the movement-distance scheme is presented. Not being part of the proposed schemes, it is however a borderline case for certain parameters of the proposed schemes and therefore included here. The scheme is a combination of two of the well-known LU schemes, namely the movement-based and the distance-based scheme. In [5] it is said thereto: "[...] the MT stores the identification of a number of M cells. This set could be an empty set, the whole set of cells within a distance $D-1$ from the cell where the last LU occurred, or some intermediate option between the two previous. Also, the MT has a movement-counter. Each time the MT visits a cell whose identification is memorized, the movement-counter is reset, i.e. set to zero. Otherwise, the movement-counter is incremented in one unit. When the movement-counter reaches a predetermined value d, the MT triggers an LU message, resets its counter and updates the set of M records."

Ibidem it is concluded that the distance-based scheme results in the optimal number of LUs, whereas with the combined movement-distance-based scheme a performance close to the distance-based scheme can be obtained.

The parameter set for which the Stop & Wait and the Reset & Wait equal the movement-distance-based scheme is when the set of cells in the local area for the time-based schemes (X) and for the movement-distance based schemes (M) are the same, i.e. $M = X$, and furthermore the timeout value $T_{LU} = 0$ and the movement threshold $d = 2$. Note that with the first crossing *out of* the local area the movement counter is already incremented by one for the described scheme. That means that only one more movement in the exterior area is possible before an LU is triggered, except for when the MT moves back into the local area, where the counter is reset.

5 Analytical Treatment

In section 3.3.2 a classification for LU strategies was given. In chapter 4 the improved scheme from [9] was introduced which is the subject of investigation of this work. The proposed method has been evaluated there by an analytical performance analysis. In the scope of this work it is given only an overview over the prerequisites of the analysis, the method how to establish the Markov model and the approach to solve the Markov model to yield the desired performance measure.

5.1 Movement Model

5.1.1 Prerequisites

To be able to tract the problem with mathematical ease, in the first step some simplifications are made. These simplifications can be adapted later for better fitting to the actual situation of a cellular network setup.

Equal cell size The cells are assumed to have the same size, irrespective of a actual (physical) cell layout.

Equal cell shape Cells are assumed to be hexagonal shaped.

Random walk The MT is assumed to have no preferred direction. This results in a random walk model of hexagons where a transition to the neighbouring hexagon, i.e. cell, is given by figure 5.1.

Equal cell dwell pdf The MT is assumed to have the same dwell time pdf with the same mean value in all cells. This might be not be true for actual users, as they have preferred location, e.g. their home or their office.

Negative-expontential call arrival pdf Calls are assumed to arrive negative exponentially distributed, which is an appropriate assumption.

5.1.2 Probabilistic Movement Derivations

With the above stated restrictions the model is as follows: The random walk assumption allows the MT to move to any of the neighbouring cells with equal probability of $1/6$ (cf. figure 5.1).

In figure 5.2 the mapping of the 2-dimensional cell layout model to an 1-dimensional Markov chain model is shown. Each state in the Markov chain equals a *ring* of cells around the centre cell (the centre cell is the one where the MT starts after the last LU/call termination). The first state however is constituted of the centre cell itself only. To calculate the transition probabilities λ_i and μ_i between the states of the state diagram (lower part of figure 5.2) we first need to calculate the number of cells in a ring. This

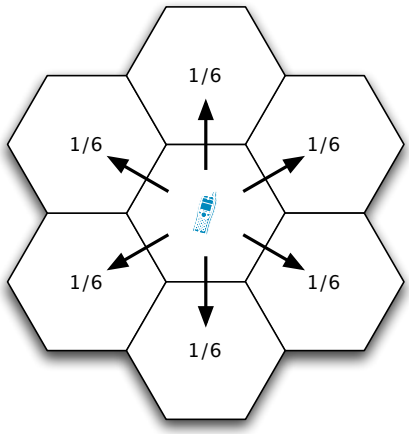


Figure 5.1: Random walk model

is because the transition probability between rings depends only on the number of cell borders to the neighbouring cells and thus indirectly depends on the number of cells in this ring. The reason for that is given in the next paragraph. Table 5.1 gives the number of cells in ring R_i as well as the number of cells in the complete tessellation T_i, where $T_i = \sum_{j=0}^{i} R_j$.

The transitions probabilities α_i' (transition probability from ring i to ring $i+1$, i.e. $\alpha_i' = P\{R_i \rightarrow R_{i+1}\}$) and ω_i' (transition probabilities from ring i to ring $i-1$, i.e. $\omega_i' = P\{R_i \rightarrow R_{i-1}$) as given in table 5.2 are calculated as follows. The probability α_i' is the ratio of the outer borders of ring i, which are given by

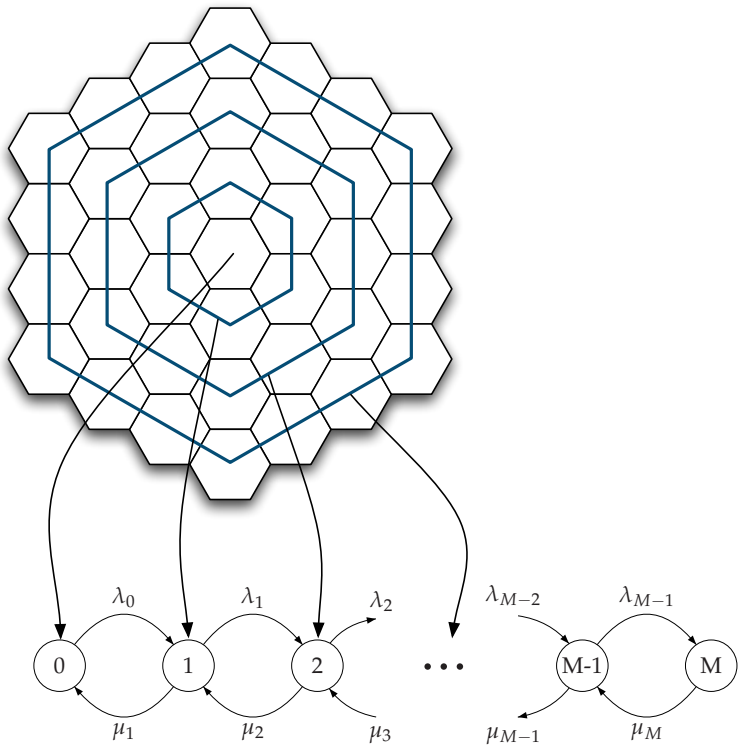


Figure 5.2: Mapping of the 2-dimensional model to an 1-dimensional Markov chain model

Table 5.1: Number of cells in ring i (R_i) and tessellation i (T_i)

i	0	1	2	3	...	k
R_i	1	6	12	18	...	$6k$
T_i	1	7	19	37	...	$3k(k+1)+1$

Table 5.2: Transition probabilities α'_i and ω'_i

i	0	1	2	...	k
α'_i	1	$\frac{1}{2}$	$\frac{5}{12}$	...	$\frac{2k+1}{6k}$
ω'_i	—	$\frac{1}{6}$	$\frac{1}{4}$	...	$\frac{2k-1}{6k}$

$N_{\text{outer},i} = 12i + 6$, divided by all borders of cells in ring i, which is $N_{\text{all},i} = 6 \cdot R_i$. Similarly ω'_i is the ratio of inner cell borders to all cell borders in ring i. The sum of transition probabilities from a given node i is $\alpha'_i + \omega'_i = 2/3$.
Additionally, conditioned transition probabilities α_i and ω_i are defined. These are the probabilities to cross from ring i to ring $i+1$ ($i-1$) under the condition that it is already known that the current ring i is left. They can be calculated from α_i (ω_i) in the following way:

$$\begin{aligned}
\alpha_i &= P\{R_i \to R_{i+1} \mid (R_i \to R_{i+1}) \cup (R_i \to R_{i-1})\} \\
&= \frac{P\{(R_i \to R_{i+1}) \cap \big((R_i \to R_{i+1}) \cup (R_i \to R_{i-1})\big)\}}{P\{(R_i \to R_{i+1}) \cup (R_i \to R_{i-1})\}} \\
&= \frac{P\{R_i \to R_{i+1}\}}{P\{(R_i \to R_{i+1}) \cup (R_i \to R_{i-1})\}} \\
&= {}^3/_2\, P\{R_i \to R_{i+1}\} \\
&= {}^3/_2\, \alpha'_i
\end{aligned}$$

The same relation holds true between ω_i and ω'_i.

The transition *rates* are the product of the rate to leave a cell multiplied by the probability of the direction. Thus $\lambda_i = \alpha'_i \cdot r_{\text{ring } i}$ with the rate to leave a cell $r_{\text{ring } i} = \frac{1}{T_{R_i}}$ and $\mu_i = \omega'_i \cdot r_{\text{ring } i}$. T_{R_i} is derived below.

The pdf for the time the MT stays in a certain ring i is given with the random variable T_{R_i} $\forall\ i \geq 1$. Assuming that the pdf of the cell sojourn time is known and denoted by f_{cell} then the time the MT spends in R_0 is equally distributed, i.e. $f_{R_0} = f_{cell}$. For all other rings it has to be considered that with a certain rate the MT moves to an outer or inner ring and with a certain probability it remains in the same ring, and moves only to another cell of this ring. Figure 5.3 illustrates the way to derive the ring pdf. The ring dwell time T_{R_i} can be composed of the addition of the probability to leave the ring and the probabilities to stay in the ring for one more T_{cell}. Thus we can proceed as follows: With probability $2/3$ we leave the ring after having spent exactly one cell time. But it is also possible, after having spent one cell time in the first cell, to enter the next cell with probability $1/3$ and then leave this cell afterwards with $2/3$. To spend three cell times before leaving the ring is even less likely, as we then change to the second cell with probability $1/3$ and then to the third cell again with probability $1/3$ and then leave with probability $2/3$. This goes on infinitely as in the ring there is no boundary to meet.

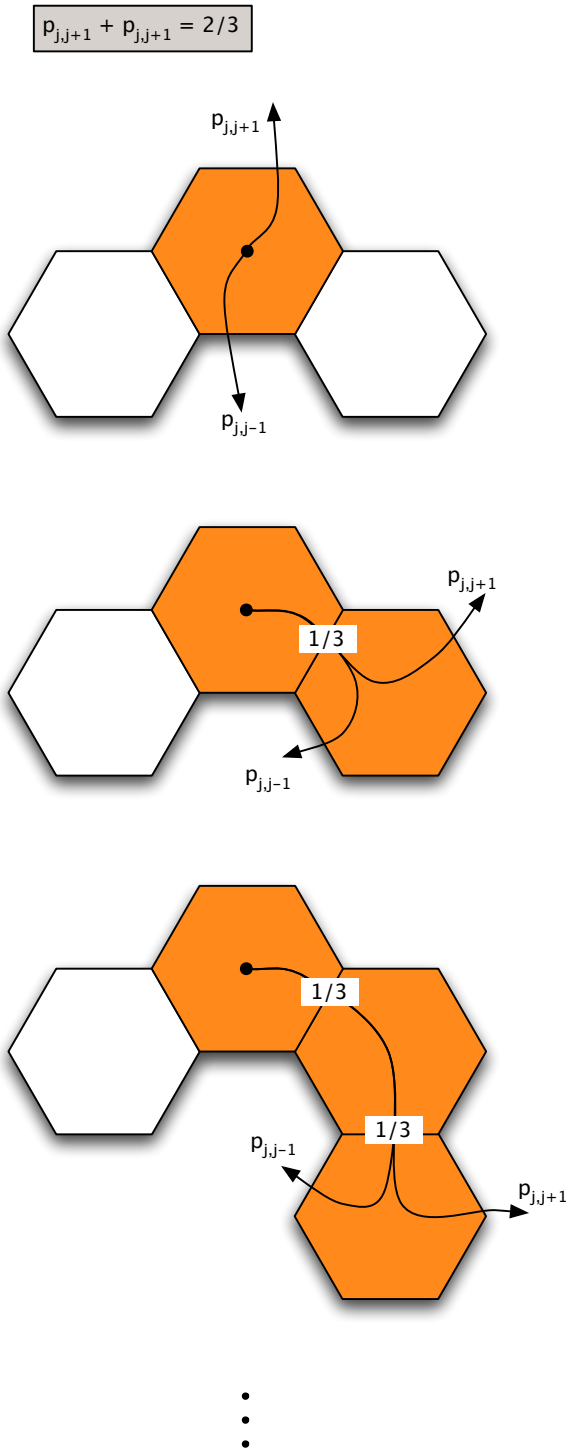


Figure 5.3: Approach for the derivation of the ring sojourn pdf

Mathematically it is possible to state

$$T_{R_i} = T_{cell}\frac{2}{3} + \frac{1}{3}2T_{cell}\frac{2}{3} + \left(\frac{1}{3}\right)^2 3T_{cell}\frac{2}{3} + \dots$$

which can be written as

$$T_{R_i} = \sum_{n=1}^{\infty}\left(\frac{1}{3}\right)^n \cdot n \cdot T_{cell} \cdot \frac{2}{3}.$$

This can be solved by the derivation of the geometrical series and we yield

$$T_{R_i} = \frac{2}{3}T_{cell}\frac{1}{(1-\frac{1}{3})^2} = \frac{3}{2}T_{cell}.$$

As can be seen, the ring dwell time is independent of the actual ring i, i.e. $T_{R_i} = T_R$.

The time T_{T_n} spent in tessellation T_n can be derived from the ring times T_R. To this end we proceed in two steps: first the derivation is done by establishing the probabilistic relations and in the second step a closed form solution is sought. T_{T_n} depends on the ring the MT starts in. Thus the notation has to be extended to $T_{T_{k,n}}$, where k is the ring the MT starts in and n the outermost ring of the tessellation under consideration. The derivation has be done for two special cases: $T_{T_{n,n}}$ and $T_{T_{0,n}}$. The time $T_{T_{n,n}}$ describes the case where the MT enters the tessellation T_n in ring n (the outermost ring) and leaves the tessellation through the same ring n later. $T_{T_{0,n}}$ however is the time the mobile needs to leave tessellation T_n when it is placed in the centre cell (T_0) at the beginning.

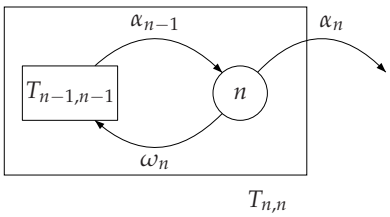


Figure 5.4: Markov chain model for the derivation of the mean tessellation time $\overline{T}_{T_{n,n}}$

1. Derivation of the area dwell time $\mathbf{T_{T_{n,n}}}$: State $T_{n,n}$ represents the case where the MT resides in tessellation T_n by entering this tessellation through ring R_n and leaving through R_n again. The state $T_{n,n}$ (see figure 5.4)is defined recursively and is composed of the sub-states n, representing the MT being in R_n, and $T_{n-1,n-1}$ which represent that the MT resides in one of the inner $n-1$ rings. The time spent in state $T_{n,n}$ is called $T_{T_{n,n}}$, with mean value $\overline{T}_{T_{n,n}}$, which is derived in the following:

$$\begin{aligned}
\overline{T}_{T_{n,n}} &= \alpha_n\overline{T}_R + \omega_n\alpha_n\left(\overline{T}_{T_{n-1,n-1}} + 2\overline{T}_R\right) + \omega_n^2\alpha_n\left(2\overline{T}_{T_{n-1,n-1}} + 3\overline{T}_R\right) + \dots \\
&= \alpha_n\left[\overline{T}_R\sum_{i=1}^{\infty}\omega_n^{i-1}i + \overline{T}_{T_{n-1,n-1}}\sum_{i=1}^{\infty}\omega_n\omega_n^{i-1}i\right] \\
&= \alpha_n\left[\overline{T}_R\frac{1}{(1-\omega_n)^2} + \overline{T}_{T_{n-1,n-1}}\frac{\omega_n}{(1-\omega_n)^2}\right] \\
&\boxed{\text{with } \alpha_n = 1-\omega_n:} \\
&= \frac{1}{\alpha_n}\left[\overline{T}_R + \overline{T}_{T_{n-1,n-1}}\omega_n\right]
\end{aligned}$$

2. Solving the recursion for $\mathbf{T_{T_{n,n}}}$: With $\alpha_n = \frac{2n+1}{4n}$, $\alpha_n = 1 - \omega_n$ and the initial condition $\overline{T}_{T_{0,0}} = T_{cell}$ the recursion equation can be solved and the closed form solution is yielded (Eq. 5.1).

$$\overline{T}_{T_{n,n}} = \frac{3n^2 + 3n + 1}{2n + 1} T_{cell} \tag{5.1}$$

Next, the mean tessellation time $\overline{T}_{T_{0,n}}$ is calculated, by making use of the just derived result for $\overline{T}_{T_{n,n}}$.

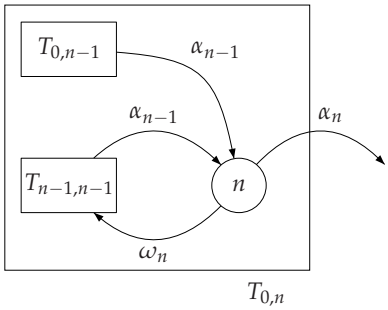


Figure 5.5: Markov chain model for the derivation of the mean tessellation time $\overline{T}_{T_{0,n}}$

1. Derivation of the area dwell time $\mathbf{T_{T_{0,n}}}$: Solving the system of figure 5.5 for the mean sojourn time in state $T_{0,n}$ (= time for entering the Markov chain of figure 5.2 on p. 43 in state 0 and leaving in state n) yields the following recursion equation (with derivation):

$$\begin{aligned}
\overline{T}_{T_{0,n}} &= \alpha_n \left(\overline{T}_{T_{0,n-1}} + \overline{T}_R \right) \\
&+ \omega_n \alpha_n \left(\overline{T}_{T_{0,n-1}} + 2\overline{T}_R + \overline{T}_{T_{n-1,n-1}} \right) \\
&+ \omega_n^2 \alpha_n \left(\overline{T}_{T_{0,n-1}} + 3\overline{T}_R + 2\overline{T}_{T_{n-1,n-1}} \right) \\
&+ \dots \\
&= \alpha_n \left[\overline{T}_{T_{0,n-1}} \sum_{i=0}^{\infty} \omega_n^i + \overline{T}_R \sum_{i=1}^{\infty} i\omega_n^{i-1} + \overline{T}_{T_{n-1,n-1}} \sum_{i=0}^{\infty} i\omega_n^i \right] \\
&= \alpha_n \left[\overline{T}_{T_{0,n-1}} \frac{1}{1-\omega_n} + \overline{T}_R \frac{1}{(1-\omega_n)^2} + \overline{T}_{T_{n-1,n-1}} \frac{\omega_n}{(1-\omega_n)^2} \right] \\
&= \overline{T}_{T_{0,n-1}} + \frac{1}{\alpha_n} \overline{T}_R + \frac{\omega_n}{\alpha_n} \overline{T}_{T_{n-1,n-1}}
\end{aligned}$$

2. Solving the recursion for $\mathbf{T_{T_{0,n}}}$: The innermost ring is a single cell only and the relation has been shown before: the mean ring sojourn time $T_R = {}^3/_2 T_{cell}$. Using this relation as initial condition along with the closed form for $\overline{T}_{T_{n-1,n-1}} = \frac{3n^2-3n+1}{2n-1} T_{cell}$, the single equation recurrence results in

$$\overline{T}_{T_{0,n}} = \left[1 + \sum_{i=1}^{n} \frac{3i^2 + 3i + 1}{2i + 1} \right] T_{cell}. \tag{5.2}$$

Even though the recursion is resolved, the undefined sum remains, so that no closed form solution is possible.

Having derived some preliminary results it is proceeded in derivation of the complete Markov model. The 1-dimensional model according to figure 5.2 is now extended in two senses. First a splitting into two regions is introduced. The first region comprise the inner rings that belong to the start tessellation of the MT. The second region comprise the outer rings, i.e. the region exterior to the "home location.". When the MT moves along in the outer region after some further movement step, which are defined by the movement counter limit D, an LU message will be issued by the MT. Remember that this is the movement-based component of the combined movement-based and time-based LU method. For detailed description refer back to chapters 3.3.2 and 4. The second extension is that a time dimension has to be added to the model in figure 5.2 to be able to model the time-based LUs of our combined LU method. The timer is deterministic as the optimal time-out for the timer has to be found. It is now easy to apply the Coxian phase method to approximate the deterministic timer RV by a combination of negative exponential RV. This is especially easy for a deterministic RV, as was shown in chapter 2.2.3, since the Erlang-k pdf converges to the deterministic pdf for large k. In the next chapter actual values for a sufficient approximation are given.

The complete model that we get thus is shown in figure 5.6.

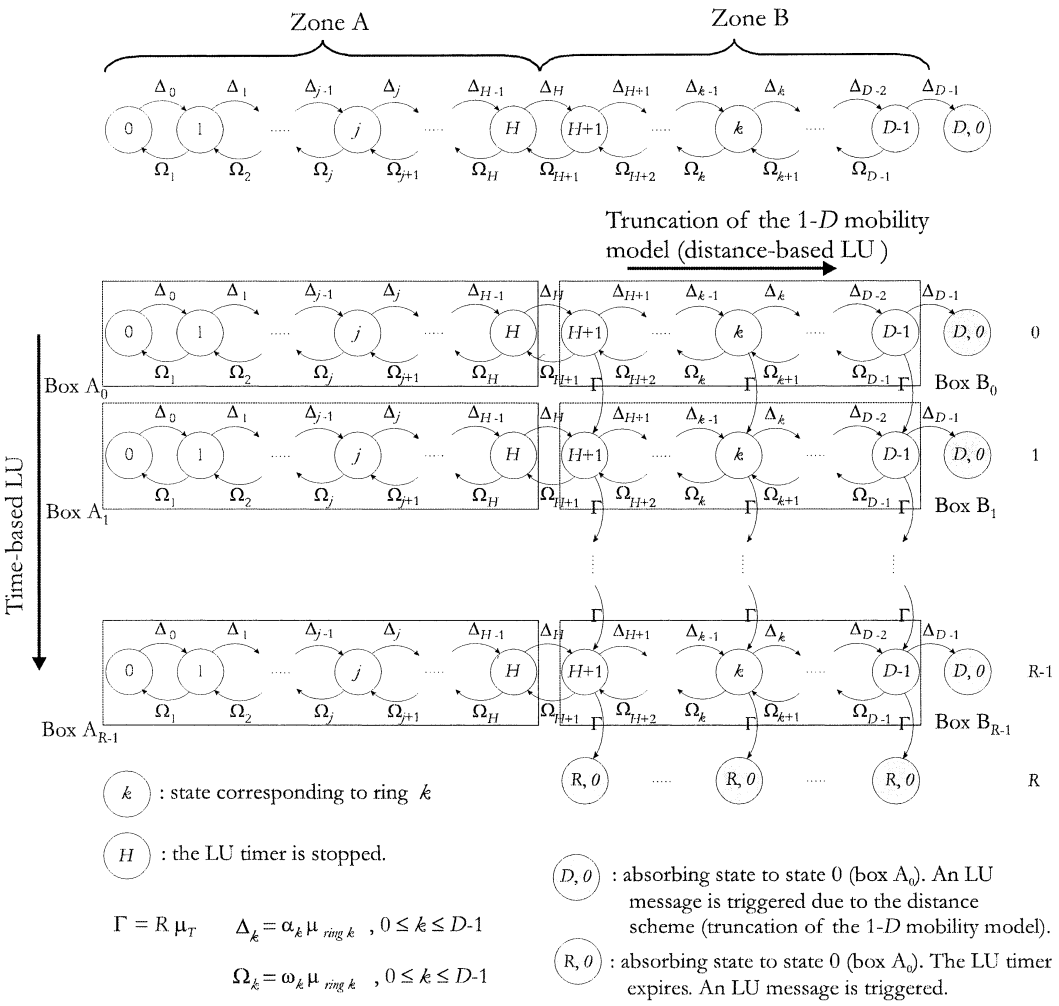


Figure 5.6: Markov chain model for the Stop & Wait scheme (from [9])

5.2 Measures for Performance Evaluation

Every location update and every paging of a MT leads to signalling traffic on the air interface and on the signalling interconnections which link the base station to the mobility management related networks entities. Resources are thus bound to signalling traffic and cannot be used for actual user data transmission.

For each signalling operation[1] *costs* are assigned. First, a *cost factor* appropriate for all the operations considered has to be chosen. This can be, e.g., the bandwidth consumption, the data amount transmitted wireless/wireline, the computational power used for the transaction, or the amount of time resources are blocked. It is also possible to chose a mix of cost factors. In a second step the cost factor is weighted by a *cost function* appropriate for this transaction.
For example: According to the amount of data that is transmitted during a signalling transaction (the cost factor), (relative) costs of $cr_{LU} = 3$ (the cost function) is assigned to the LU procedure and $cr_{PG} = 1$ is assigned to the PG procedure, which expressed that during LU three times more data is transmitted than during a PG. To derive valid values, the specifications of the Mobile Application Protocol and other specifications can be used. In this work though standard values are taken to make results comparable. Akyildiz et al. [2] use $cr_{LU} = \{1, 10, 50\}$ and $cr_{PG} = 1$, Zheng and Regentova [37] use $cr_{LU} = 10$ and $cr_{PG} = 1$ whereas García-Escalle and Casares-Giner [9] and Lee and Lee [18] use $cr_{LU} = 1$ and $cr_{PG} = 1$. The latter values are also used in this work for better comparison with [9]. For the evaluation of the proposed LU scheme the use of the exact costing is not crucial however, as the objective is to show the relative superiority over other methods only.

5.3 Analytical derivation

The analytical derivation is conducted in [9]. Here, only an overview of the general approach is given. For details refer to the original paper.

It should be noted, that no closed-form solution is derived, even though this would maybe possible by making use of symmetries, which is subject to further research. Though, this would only beautify the already-known results. Instead of a closed-form solution, a matrix-based calculation is used. For the evaluation of the state probabilities matrices Matlab is used.

For understanding the general approach of solving for the costs of the proposed method, the Markov model in figure 5.6 is used. It describes the Stop & Wait scheme proposed in [9] but for the Reset & Wait scheme the Markov chain description looks much the same.
The setup is as follows: In the two-dimensional Markov chain the state space is made up from a space and a time component. The states in x-axis represent the space component, where the two-dimensional area is mapped to a one dimensional Markov chain as explained back in figure 5.2. The states (boxes) of the y-axis constitute the time component. As can be seen for states $0 \ldots H$ only horizontal transitions are possible, which means that within the local area the MT can move in space but the timer does not run yet. Only after having left state H and being in a state $\{k|k > H\}$ transitions can happen in the space-domain and in the time-domain. To get a finite state space the absorbing states $D, 0$ are introduced. If the MT reaches as far as to state $D, 0$, i.e. it moved D cells far from the cell of the last LU, a new LU message is sent. In downward direction the absorbing states $R, 0$ are responsible for the actual LU

[1] NB: signalling, (signalling) operation and (signalling) transaction are used synonymously in the context of this chapter

after the timeout occurred. The R time boxes make up the R time phases of an Erlang-R distribution, according to the Coxian phase type model (cf. chapter 2.2.3).

Matrices are established to represent the probability that during the observation interval (which is the inter-call-arrival time; exponentially distributed with rate $1/\lambda_{call}$) no LU events occur $(\mathbf{P}(\text{no LUs}))$, that a time-based LU occurs $(\mathbf{P}(\text{LU}_T))$ and that a distance-based LU occurs $(\mathbf{P}(\text{LU}_D))$. Then the matrix that exactly n time-(distance-)based LUs occur can be established, by making use of the above defined probabilities. With the distribution in n, the average number N of time-(distance-)based LUs can yielded (cf. [9], eqs. (9)-(11)). However the used probabilities $(\mathbf{P}(\text{no LUs}))$, $(\mathbf{P}(\text{LU}_T))$ and $(\mathbf{P}(\text{LU}_D))$ are composed of simpler transition probabilities and are established directly from the possibilities a MT can move through the Markov chain of figure 5.6. These derivations are conducted in [9], eqs. (16)-(28) for the Stop & Wait model and additionally in [9], eqs. (29)-(32) for the Reset & Wait model.

Finally the matrices are evaluated numerically, making use of Matlab. Having the mean number of the LUs and PGs between two calls the costs result directly as shown in chapter 5.2.

6 Simulation

6.1 Introduction

Other than evaluating the proposed LU method analytically, the possibility of simulating the system was used to cross-check the obtained results. The method of simulation has several advantages over the analytical method. As no thorough knowledge about the analysis of queueing networks is needed, less experienced engineers are still able to tackle problems of this kind. Moreover the model can be modelled in a high-level notation, making it easier to understand and thus the probability for errors in the analysis is lower. One last big advantage is the possibility to observe the modelled system under quite general conditions as less mathematical restrictions and assumptions are necessary, e.g. generally distributed pdfs for activities or (nearly) infinite queue lengths may be used.

However simulations have some disadvantages, which should not remain unmentioned. A simulation can be very time consuming, as it is a discrete event simulation, not just an evaluation of an equation system. The simulation time depends on the desired accuracy of the performance measures of interest. Analytical analysis one the other hand can be evaluated to high degree of accuracy without substantial increase in processing power. However a very high accuracy does not necessarily help in understanding the system better and is often needless. A side effect of this is, that while simulation parameters have to be chosen carefully since it takes a long time to obtain results, evaluation of analytical equations can be done for a larger range of values of interest. One of the most important issues with simulations is that for the sake of saving processing power only mean values but not complete pdfs of the performance measures of interest are estimated.

As for the advantages of simulation this method was preferred in this work. The next decision to take is how to simulate the system. In general there are two possibilities: to describe the system in a high-level model/model language specialised to performance simulations or to write a simulator in a usual multi-purpose programming language, where *C(++)* and *Java* are common choices.
The advantage in using a high-level model language lies in the great clearness of the model. In addition the modeller does not have to care about low level functionality, as the modelling environment takes care of, e.g., the correct generation of random variables and the abidance of given confidence intervals (for details on these topics see section 6.4 below). All this results in more rapid model-building. On the other side with the use of multi-purpose languages the programmer retains full control over all possibilities of simulations and is not constrained by the features of the environment.

In this work the high-level approach was chosen, as results were demanded quickly. The Möbius-framework and tool was chosen as previous experiences were positive and know-how was available in the team. In the following the Möbius environment is described briefly.

6.2 Möbius

Möbius™ is a software tool for analysing systems stochastically. It originated from the Möbius project which "is one of the major research projects of the Performability Engineering Research Group (PERFORM) in the Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign"[1], USA.

"Möbius™ is a software tool for modeling the behavior of complex systems. [...] Although Möbius was originally developed for studying the reliability, availability, and performance of computer and network systems, its use has expanded rapidly. It is now used for a broad range of discrete-event systems, from biochemical reactions within genes to the effects of malicious attackers on secure computer systems, in addition to the original applications."[2]

To model complex systems Möbius offers several stochastic modelling formalisms, known from the literature. These are *Stochastic Activity Networks* (SAN), Bucket and Balls (a simplified SAN), Performance Evaluation Process Algebra (PEPA) and Fault Trees. Each of these formalisms have their advantage in a specific application field. Further formalisms can be included in the Möbius tool if necessary.

The model specified in one of the above stated formalisms is called *atomic* in Möbius. In the atomic all elements can be used that the used formalism provides, like places, activities and gates for SANs or nodes, events and logical gates for Fault Trees.

Elementary models (atomics) in either of the above stated formalisms can be agglomerated hierarchically to *composed* models. This option simplifies the construction of larger models consisting of many similar sub-models.

After the complete model has been specified it is necessary to define *reward variables*, which specify the measures interested in. Reward variables are defined in a special area of Möbius by a "reward specification formalism"[3].

"During the specification of atomic, composed, and reward models in the tool, *global variables* can be used to parameterize model characteristics. A global variable is a variable that is used in one or more models, but not given a specific value"[4] In Möbius' *study editor* to each global variable a value is assigned. It is also possible to assign multiple values and functional parameter sweeps to each global variable, allowing the model to be analysed for a variety of parameters. Each combination of value assignments form an *experiment*, which needs to be processed separately in the following.

The models can be processed by transforming them into Markov chains and analysing the Markov chain numerically or by simulating the whole model by a (distributed) discrete event simulation.
While simulating a model is always possible, to use the state based analytical transformation and numerical analysis several preconditions have to be fulfilled:

- use of only exponentially distributed activities (plus *one* deterministically delayed activity though)
- finite state space
- self-stabilizing

Advantages and disadvantages for simulation (taken from [29]):

[1] cf. [11], p. ix
[2] cf. [11], p. ix
[3] cf. [11], p. 4
[4] cf. [11], p. 20

Advantages

- simulations are not restricted to exponentially distributed stochastic processes, but generally distributed activities can be used
- no need for the generation of a state space, which allows for solving of more complex and more detailed models
- current simulation time can be accessed and used for influencing the further model behaviour

Disadvantages

- the performability measure is an estimate only. A confidence interval is given for the measure but the confidence interval itself is based on an estimator and thus may be incorrect.
- to make the confidence interval n times smaller (and thus the result more confident) is an $O(n^2)$ problem, which means for higher accuracy much more effort is needed
- estimation of distributions or even percentiles needs more computational effort whereas for analytic solutions it comes for free
- very long simulation times arise for rare event problems and for very complex models

6.3 Stochastic Activity Networks (SANs)

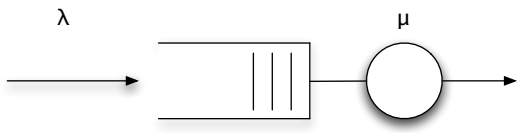
The brief introduction of the elementary building blocks of SANs follow the Möbius Manual ([11]) and the review in [28] where these elements are presented in. The explanation is merely illustrative without any mathematical (graph theoretical) description.

Stochastic Activity Networks, which are very closely related to Stochastic Petri Nets are based on Petri Nets[5]. In an abstract way of defining a Petri Net it is a bipartite[6] directed graph. Petri Nets were invented by Petri[7] who introduced them in his PhD thesis to model concurrency in computer systems. Finite state automata and Petri Nets have the same modelling power. In "classic" Petri Nets there is no notion of time, which means that the transitions between states can occur any time, as long as the respective predicate for a transition is fulfilled. This *non-deterministic* behaviour does not allow the usage of Petri Nets for performance evaluation of the modelled systems. Thus, among the numerous extensions and derivatives that have been proposed, *timed* Petri Nets have evolved, with the above-mentioned Stochastic Petri Nets (SPNs) as a subclass. Each transition has a specified duration it takes from activation to firing of the activity/transition. In SPNs the duration is not given deterministically but by a distribution function. As long as only exponential pdf activities are used, it is possible to transform SPNs into Markov chains (cf. chapter 2.2.2) analysing them analytically. SANs are a derivative of SPNs where the enabling predicates and the output actions are more powerful, allowing for easier to understand as well as smaller models.

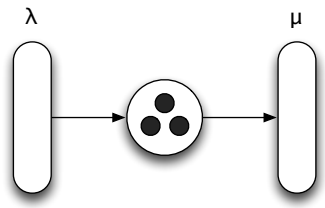
In figure 6.1 four different but equivalent graphical models are shown, representing the same physical problem. Figure 6.1(a) shows a simple M/M/1 system in queueing network representation with arrival rate λ and departure rate μ. Figure 6.1(b) models the same system as a SAN, with the ovals as *activities* and the circle as *place* (for explanations of the model elements see below). In this example the place holds three tokens representing three customers in the servers queue at a certain point in time. The same model entered in Möbius is shown in figure 6.1(c). To analyse the queueing system the system can

[5] also known as place/transition net, in short P/T net
[6] the nodes of the graph can be divided into two disjunct sets of nodes, where there are no edges between nodes of the same set
[7] Carl Adam Petri; *1926; German mathematician and computer scientist; introduced Petri Nets in 1962 [25]

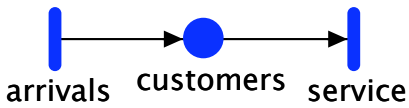
be modelled by the Markov chain from figure 6.1(d). Again arrival and departure rate can be seen. This Markov chain can be solved to obtain the state probabilities and with these the desired performance measure can be obtained.



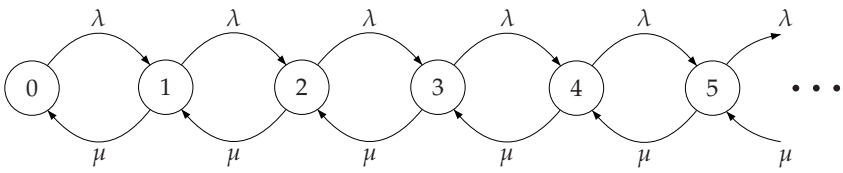
(a) Queueing Network



(b) Stochastic Activity Network



(c) Representation in Möbius



(d) Markov Chain

Figure 6.1: Equivalent models in different representations for a M/M/1 delay system

In the following the model elements of SANs are presented.

Timed Activities "Timed activities are used to represent activities of the modeled system whose durations impact the system's ability to perform."[8] The time duration of an activity can be generally distributed and is given by the *activity time distribution function*. Upon completion of an activity different outcomes are possible. An outcome, known as *case*, is chosen probabilistically according to a *case distribution*.

A timed activity is represented by an oval thick vertical bar.

Instantaneous Activities "Instantaneous activities represent system activities which, relative to the performance variable in question, complete in a negligible amount of time."[9] Like timed activities the instantaneous activities can have cases also.

[8] cf. [28], p. 807
[9] cf. [28], p. 807

An instantaneous activity is represented by an thin vertical line.

Places Places can be seen as a kind of state variables, but "each place can hold a nonnegative number of *tokens*"[10] only. The number of tokens in a place is called the *marking* of that place. "Note that tokens in a place are homogeneous, in that only the number of tokens in a place is known; there is no identification of different kinds of tokens within a place."[11] This is different to coloured Petri nets, for example.

A place is graphically represented by a circle.

Input Gate "Input gates contain both an enabling predicate and input function (on the marking of the places)."[12] Activities are enabled if and only if the *predicate* (which is a boolean function) of the input gate is true. The input function is executed after completion of an activity.

An input gate is graphically represented by a triangle of whose one vertex is in left direction. Usually but not necessarily the left vertex is connected to the place the predicate depends on but the flat right side of the triangle is connected to the related activity always.

Output Gate Output gates consist of an output function. Upon completion of an activity the function in the output gate associated with the respective case of the activity is executed. The function can perform calculations and alter markings of places or global variables.

An input gate is graphically represented by a triangle of whose one vertex is in right direction. The left flat side of the triangle is always connected to a case of an activity whereas the right vertex need not be connected but usually is connected to the place that is changed by the output function.

Lines Lines correspond to directed arcs which connect the above explained elements, to establish a relation between them.

6.4 Discrete Event Simulation

Simulating a system is a third method of analysing and evaluating a system, when analytic evaluation is too complex and prototyping is not feasible, e.g. because of the high costs for the prototype. To simulate a system the following parts have to be conducted:

1. Generate random data as input for the system (Monte-Carlo simulation)
2. Trigger the actions for the events of the random input data
3. Observe the actions to get performance measures.

In the following the most important parts of a discrete event simulator are introduced. Möbius is also a discrete event simulator but as simulation framework it takes care of the topics presented, for convenience of the model developer.

[10] cf. [28], p. 807
[11] cf. [11], p. 45
[12] cf. [28], p. 807

6.4.1 Generation of Random Events

The essential task of the event generator is generating events with a given interarrival time distribution. Most programming languages however, offer a library function for generating *uniformly* distributed RVs only. To obtain RVs with a different pdf, the *Inverse Transform method* is used (cf. [17],[33]). With this method we can generate a RV X with any given distribution function $F(x)$.

We need the inverse function of $F(x)$. For an exponentially distributed RV we yield:

$$F(x) = 1 - e^{-\lambda x}$$

$$x = F^{-1}(y) = -\frac{1}{\lambda}\ln(1-y)$$

Using uniformly distributed values of the interval (0,1) for y, i.e. $y \sim$ UNIFORM(0,1), we get exponentially distributed values with mean value of λ^{-1}. The principle with the inverse function can be seen in figure 6.2.

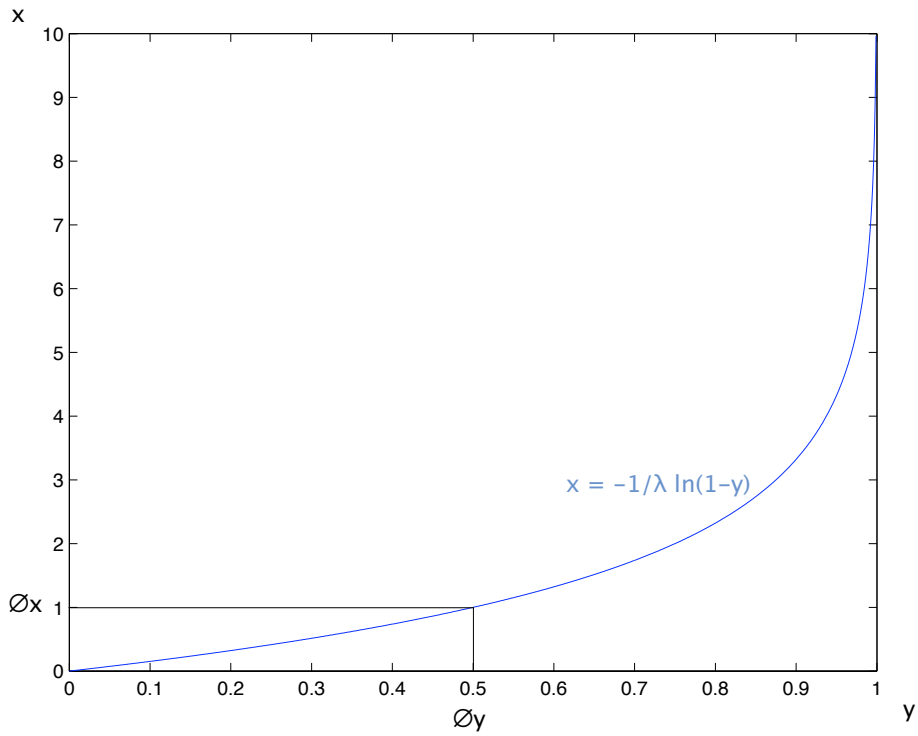


Figure 6.2: Inverse Transform method

When generating uniform numbers we can simplify to $x = -\frac{1}{\lambda}\ln(y)$ because the result will be the same because UNIFORM(0,1) = 1 - UNIFORM(0,1).
For the generation of the actual uniformly distributed numbers, library function can be used which itself are based on one of the many pseudo-random generators, like Lagged Fibonacci or Tauseworthe.

6.4.2 Event Lists

In the section above it was explained how RV can be generated. In the following, it is exemplarily explained how the further processing is done.

The generated RV usually represent the time, as systems are evaluated for their performance where the service times are random. To make the performance measures in a simple system, like the M/M/1 delay system of figure 6.1, we have to keep track of two kinds of events: the arrivals and the departures. In more complex systems more types of events are possible. In the M/M/1 system the inter-arrival times as well as the service times are negative-exponentially distributed. So the generation of RVs as described in section 6.4.1 can be used.
When the simulation system is started the simulation clock is set to zero. Then the point in time for all types of possible events are generated. In the example system this would be an arrival event time. No service time is needed as there is no token in the server for servicing. The arrival event time is added to an *event list*. The simulator clock is then set to the time of the next event in the event list. Then the event is "processed" and for all types of possible and "used" events, new events are generated. In the example a new inter-arrival time has to be generated, as well as a service time since after the arrival one customer is in the system waiting for his servicing to finish. The two time spans are added to the current simulation clock and the two events added to the event list. The list is then sorted according to the time stamp of the events, with the smallest time at the top of the list. The procedure described here repeats then, i.e. the clock is advanced to the next event to happen and after the processing of the event action a new event time is generated and sorted into the event list.

The advantage of the approach with an event list (also called the *asynchronous* approach in [14]) is that there is no waiting time between events. Even if the time between two events is very long, the simulator is very fast, as the clock is just set to the next scheduled event. However more complicated issues are still present as e.g. the rare event issue, where events that occur only very seldom in comparison with the other events result in very long simulation times. More about event lists and modelling of stochastic systems in general can be found in [14], [10], [8] and [17].

6.4.3 Confidence Intervals

In the above sections 6.4.1 and 6.4.2 two basic concepts of discrete event simulations are described. The last basic concept which will be introduced briefly deals with evaluation of the obtained measures. The objective of every simulation is to obtain a good estimation for a measure of interest. E.g. the mean number of customers in the system or the relative time the system has more than five customers.

To get a reliable declaration for the measure several *runs* (also called *batches*) of the same experiment have to be conducted. By averaging the batches the validity of the measure can be increased. To give the credibility of the measure the confidence-interval notation is usually used. The *confidence* (also known as *confidence level*) gives the probability with which the actual value is in the stated *(confidence) interval* around the estimated value. For this calculation it is assumed that the batches are independent of each other and thus the central limit theorem applies. So the pdf over the batches is assumed to be a Gaussian[13] standard normal distribution. Then the relation between the confidence level $1 - \alpha$ and the

[13] Carl Friedrich Gauß; ⋆1777-†1855; German polymath, who obtained the standard normal distribution when examining the method of least squares

interval $\pm z_{\alpha/2}$ can be given mathematically (without derivation) as

$$P\left[\overline{Y} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \overline{Y} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right] = 1 - \alpha$$

where $\overline{Y}$ is the estimated value, μ the actual (unknown) value, σ the standard deviation and n the number of batches. As can be seen the confidence interval gets smaller with higher number of conducted runs. On the hand, for constant interval the confidence level increases with higher number of conducted runs.

Usually the standard deviation σ is also unknown, so the confidence interval should be calculated with the Student[14]-t distribution, using an estimated standard deviation. More about confidence intervals can be found in [17] and [14].

6.5 Simulation Process

To derive the desired results from the original idea, several steps are necessary, which are shown in figure 6.3 and described in the following. However, only the computerised steps are shown, not the design process from the idea to the model.

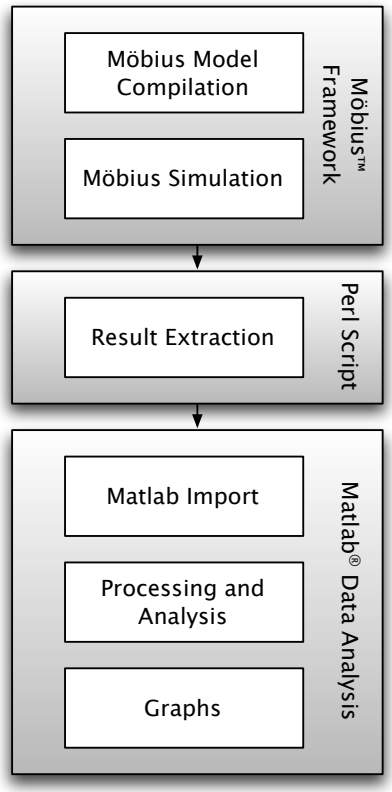


Figure 6.3: Processing (tool) chain for simulations

Initially the idea has to be modelled in a way that is possible to process by a computer. In this work the Stochastic Activity Networks (SAN) is the used formalism. The SAN model is entered to Möbius and

[14] pen name of William Gosset; ⋆1876-†1937; English chemist and statistician of Guinness brewery, who introduced the Student-t distribution for statistical tests on small samples in the brewery process

then compiled to an executable. This executable is self-contained, which means, all parameter sweeps of the study/simulation and all other model information is contained. Möbius is also in charge of the simulation with respect to the random number seeds, the number of batches and the abiding of the confidence interval, where these last two points are related. This simulation step can be considered the first main task in the process. Depending on the model and the parameters, it is also the most time consuming step.

The Möbius output is a comma seperated value file (CSV) and a text file with equivalent information. Both files however contain not the result values only but also some metadata. For further processing in data analysing tools, the bare numeral information is extracted by a Perl script. Perl was choosen as it has powerful yet simple matching filters based on regular expressions. See appendix B for the details of the Perl pre-processing.

For the final analysis Matlab is used. This is necessary as Möbius has neither integrated data analysis nor plotting capabilities, opposed to its predecessor UltraSAN™. Data analysis steps comprise the calculations on the raw data, followed by curve fitting, interpolation and plotting of the processed data. Finally the obtained values of interest are plotted graphically and important characteristic values are displayed. The analysis is the second main part in the simulation process.

6.6 Studies

To compare the simulation result with the analytically derived ones in [9], different studies with certain parameters had to be conducted. In table 6.1 an overview over the simulations is given. Note that simulation and study are used equivalently, even though the word study stresses the specific *parameters* used for this simulation, whereas simulation stresses the *execution* of the study.

Execution times of the studies varied for several reasons. Firstly the number of supporting points that were calculated influences the execution time as for each new data point a new "experiment" is generated by Möbius. The complexity is thus linear, i.e. $O(n)$ in Big-O notation, where n is the number of experiments/data points. But the most influential parameter in regard to computation time is the accuracy which is desired for a reward variable, expressed by the confidence interval for that reward variable. According to [29] the complexity is $O(m^2)$, with m as the factor of increase in accuracy. To reduce the confidence interval by a factor of two therefore needs approximately four times as much batches and thus four times as long. This relation was also observed in this work during trial runs of some studies.
Another less easy to take into account factor influencing the execution time is the complexity of the model. A larger model generally tends to take longer to simulate than a smaller one.
Generation of debug traces resulted in a manifold slower execution time, so this option of Möbius was only used for validation of the respective model and for the actual calculation of data points the optimised executable was used. Finally, as a rule of thumb the simulations took between 1 hour for fewer data points and less accuracy and more than 24 hours for an adequate number of points and accuracy.

Table 6.1: Excerpt of conducted simulations

Study Number	Study Name	Details	
1	[9]; Fig. 6; Original Scheme	# of CMRs	46
		# of CTRs	272
		Simulation Time	100
		SAN model design	averaging LUs and PGs over simulation time
		Confidence Interval	0.01 with 95%
2	[9]; Fig. 6; Original Scheme; Improved SAN	similar to Study 1, but ...	
		SAN model design	averaging LUs and PGs over intercall time
		Confidence Interval	0.01 with 95%
3	[9]; Fig. 6; Modified Scheme	similar to Study 2, but ...	
		# of CMRs	19
		# of CTRs	37
4	[9]; Fig. 6; S&W, H=0	similar to Study 2, but ...	
		Local Area (H)	0
		# of CMRs	23
		# of CTRs	25
		Confidence Interval	0.005 with 95%
5	[9]; Fig. 6; S&W; H=0; modified	similar to Study 4, but ...	
		Local Area (H)	0
		# of CMRs	19
		# of CTRs	51
		Confidence Interval	0.01 with 95%
6	[9]; Fig. 6; R&W; H=0	similar to Study 2, but ...	
		Local Area (H)	0
		# of CTRs	140
		Confidence Interval	0.005 with 95%
7	[9]; Fig. 6; R&W; H=0; modified	similar to Study 6, but ...	
		# of CMRs	37
		# of CTRs	90
8	Deterministic vs. Erlangian timer	similar to Study 7, but ...	
		Confidence Interval	0.01 with 95%
9	S&W modified time-based vs. S&W movement-distance-based	similar to Study 5, but ...	
		Timeout T_{LU}	0
		Confidence Interval	0.01 with 95%
10	R&W modified time-based vs. R&W movement-distance-based	similar to Study 5, but ...	
		Timeout T_{LU}	0
		Confidence Interval	0.01 with 95%

6.6.1 Study 1: Original Scheme

Used SAN Model

The SAN model used for the validation of the original scheme is documented in the following. Find the model in figure D.1 in the appendix. The upper part of the model is responsible for the (1D-)movement of the MT. This element is central to the model and is used in all models of this work. The three cases of the cell activity represent the three possible ways the MT can move after the cell dwell time has passed: to a cell of an inner ring (only if not already in the innermost cell), to a cell of the same ring (if not already in the innermost cell) and to a cell of an outer ring.

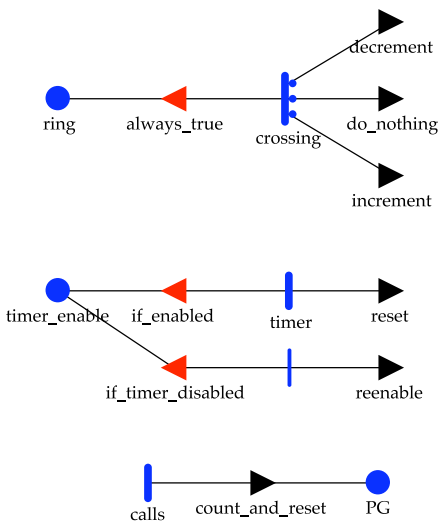


Figure 6.4: SAN modelled in Möbius for time-based MM

The middle part of the model shows the timer function. The timer is running always except for when it is reset after paging. This reset of the timer is done by stopping and restarting the timer. After the timeout the ring location is set to the innermost (centre) cell, to start over again.
The lower part simulates the arrival of calls and counts the paged cells in bucket PG. Moreover the ring location is reset, so that the MT starts in the innermost cell again, and the timer is disabled and thus reset.

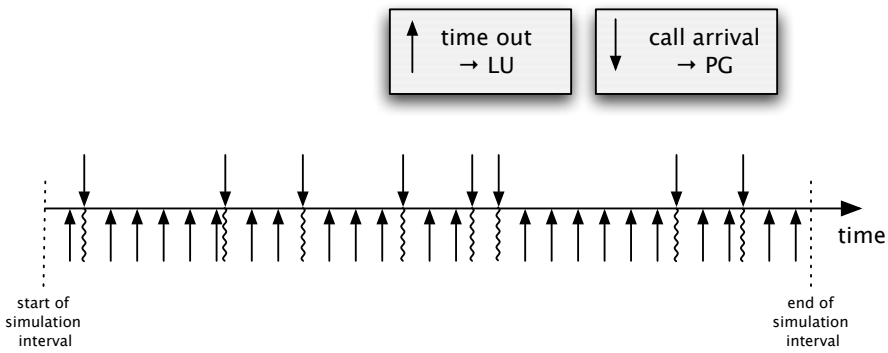


Figure 6.5: Discrete events of the stochastic MM point process

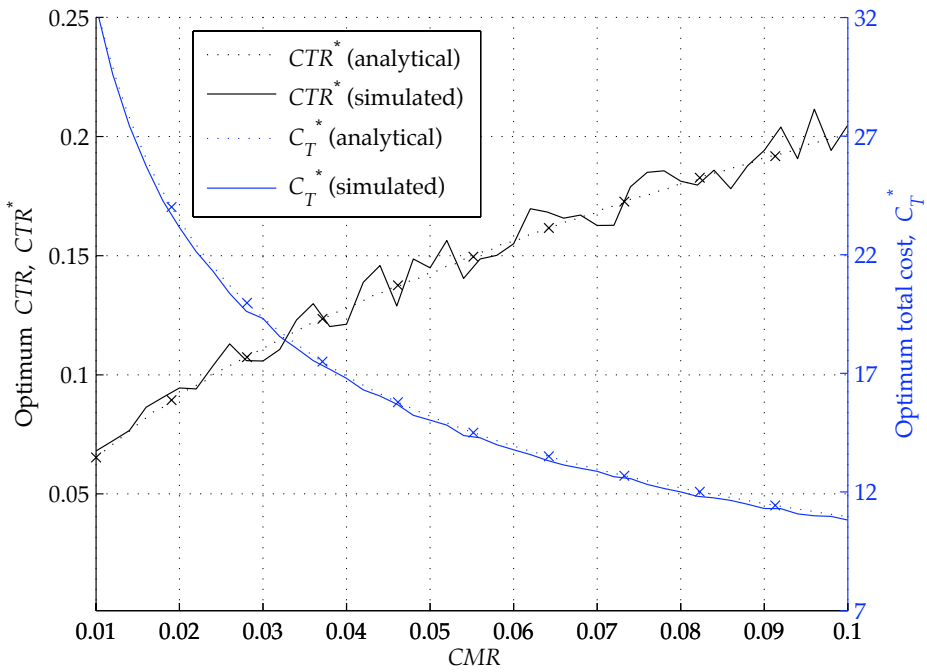


Figure 6.6: Curve for Original MM Scheme as in Figure 6 in [9]

In this model the simulation time is 100 seconds. The LU cost and PG cost that are accumulated in that time are divided by 100 as $r_{call} = 1/\text{second}$ and thus in 100 seconds 100 calls occur. By that the LU cost and PG cost between two calls are obtained. This is shown in figure 6.5, where the point process is sketched with upward arrows representing timeouts (LUs) and downward arrows for call arrivals (PGs). As can be seen, the timeouts are equidistant as they are deterministic, whereas the calls arrive randomly.

Results

The result graphs are shown in figure 6.6. The total costs C_T^* are given for the simulation as well as for the analytical calculation. The analytical curve and the simulation points of the optimal call-to-timeout ratio CTR^* are plotted also.

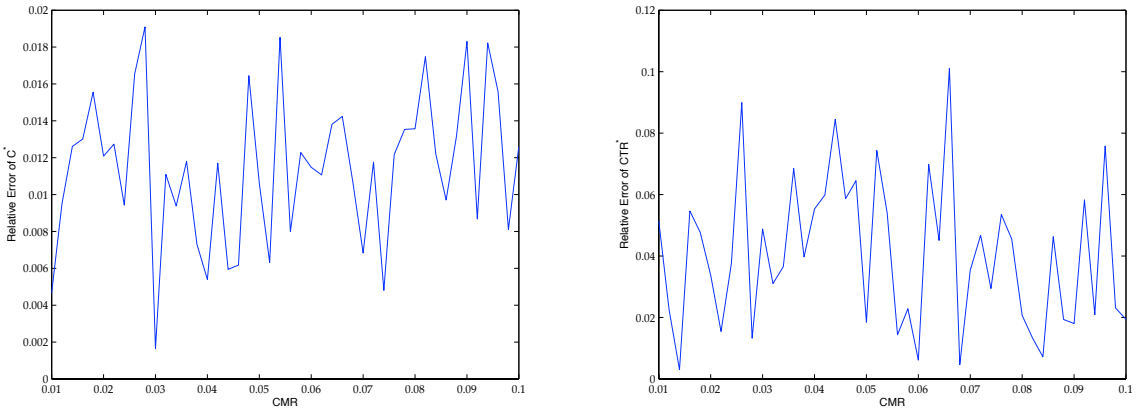
The *deviation* between the respective analytic and simulated curve is called (relative) *error* η. The relative error is defined as the absolute difference between the two curves divided by the absolute value of the curve without error. In equation 6.1 the vectors $\bar{a}$ and $\bar{b}$ refer to the vector of (y-)data points of each curve.

$$\eta = \frac{|\bar{a} - \bar{b}|}{|\bar{a}|} \tag{6.1}$$

In figure 6.7 the errors for C_T^* and CTR^* are given. The mean values are

$$\bar{\eta}_{C_T^*} = 0.0114 \text{ and} \tag{6.2}$$

$$\bar{\eta}_{CTR^*} = 0.0404. \tag{6.3}$$



(a) Relative Error of C_T^*

(b) Relative Error of CTR^*

Figure 6.7: Relative Errors of the simulated results in comparison to the analytically derived results for Study 1

The results for the relative error are not satisfying, especially not for the CTR^* with more than 4 % mean deviation. The high deviation can also be seen directly in figure 6.6 as the strong ripple. The reason for the high variance in the simulated CTR^* lies in the broad minimum of the CTR curve. Whereas the actual C_T^* is not very sensitive to simulation variations, the CTR^* is very hard to place exactly because of the above mentioned broad minimum of C_T. This is shown in figure 6.8 where the C_T is plotted against the CTR for all CMRs. It is intuitively clear that with the broad minimum in C_T it is hard to find the optimal CTR. Little variations in the simulated data are sufficient to shift the minimum by several percent.

To estimate the error we can expect without further error processing, an *error propagation* calculation is carried out. First we put on record that

- CTR depends on C_T
- the C_T curve was calculated with a confidence interval of 1 % with 95 % probability, i.e $\Delta C_{T,\,\text{relative}} = 1\%$.

The error of CTR is calculated as

$$\Delta CTR \approx \left| \frac{d\,CTR}{d\,C_T} \right| \Delta C_T \tag{6.4}$$

where ΔCTR and ΔC_T are the absolute errors, not the relative ones, which are given by the confidence interval.

The differential coefficient $\left|\frac{d\,CTR}{d\,C_T}\right|$ is approximated from the graph for $CMR = 1$. $\left|\frac{d\,CTR}{d\,C_T}\right|$ is between 0.05 and 0.1. With this the error ΔCTR can be estimated with equation 6.4 and then transformed to the relative error for CTR, $\Delta CTR_{\,\text{relative}}$. $\Delta CTR_{\,\text{relative}}$ can be expected to be in the range of 20 % indeed.

To elude the impact of the simulation data accuracy, in a new approach the minimum is not calculated from the actual data, but first a curve fitting is done and with the fitted data the minimum for the total costs (C_T^*) is sought.

The fitted curve should fit the actual data as good as possible. To validate the *goodness* of the fitting, the residuals have to be examined. But generally, the curve used for the fitting should have a relation to the underlying physical model. The physical model for the MM in question is thus derived in the following.

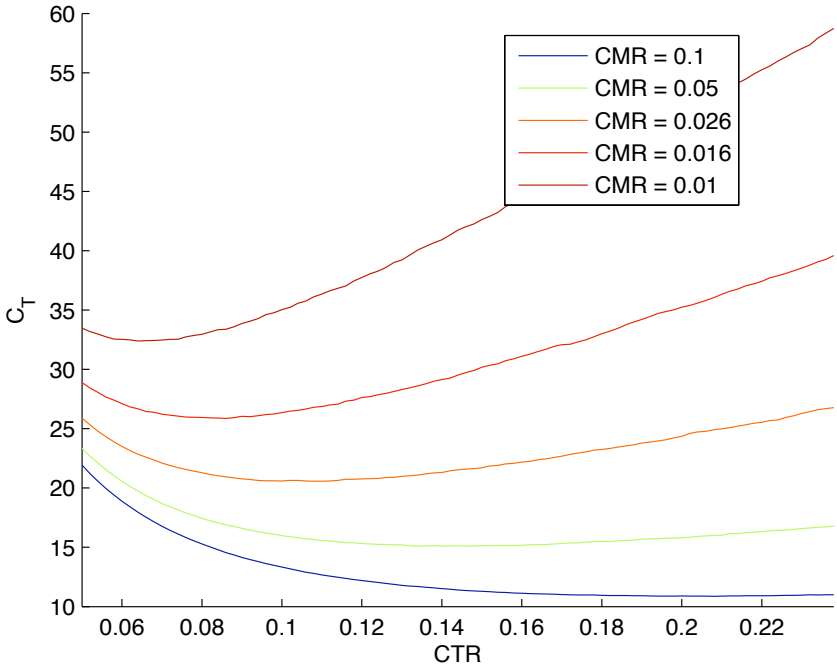


Figure 6.8: C_T-over-*CTR* curve for some *CMR* values; the minimum of the curve is "broad"

The total cost for the MM is composed of the LU cost and the PG cost. In the time-based MM model the costs depend on the timers timeout T_{LU}. It is intuitively clear that $C_{LU} \sim \frac{1}{T_{LU}}$. The paging costs however are not directly linked to the timer. To derive the relation between C_{PG} and T_{LU} it was first considered, that the is a relation between T_{LU} and the ring n where the timeout occurs and therefore between T_{LU} and $\overline{T}_{T_{0,n}}$. C_{PG} again depends on n. In other words, by knowing the relation between mean tessellation sojourn time and ring and further considering the dependency of the PG cost from the ring the MT resides in when a call arrives, a relation between sojourn time and PG cost can be established. However the obtained result for $\overline{T}_{T_{0,n}}$ (eq. 5.2) is unexpectedly nontrivial (opposed to the result for a MT starting from the outer ring which is quite simple) so further analytical use is not possible. That means, there is no easy-to-grasp relation between the PG costs and the timeout value T_{LU}.

The curve fitting used is thus based on the two parts, the $1/x$ part for the LU costs and a 1st order polynomial for the PG costs. The 1st order polynomial was chosen because the residuals of *CTR* after fitting were lower than 1%. In figure 6.9 a magnification is shown of the area where the minimum is located. As can be seen, the location of the minimum differs for the unprocessed raw data and the fitted and interpolated data.

For the such fitted C_T-versus-*CTR* curves the results for the optimal *CTR* are much smoother and more accurate (see figure 6.10). The mean error between the analytical and the simulated costs reduce to

$$\overline{\eta}_{C_T^*} = 0.0083 \text{ and} \tag{6.5}$$

$$\overline{\eta}_{CTR^*} = 0.0091. \tag{6.6}$$

The improvement for C_T is unincisive, whereas the value for *CTR* improves about more than a factor of 4 (compare with original values in (6.2) and (6.3)).

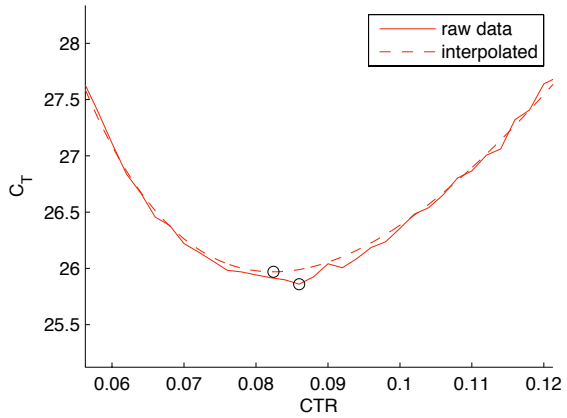


Figure 6.9: Consequence of fitting: for $CMR = 0.016$ the minimum of the raw data and of the fitted and interpolated data differ about 4%

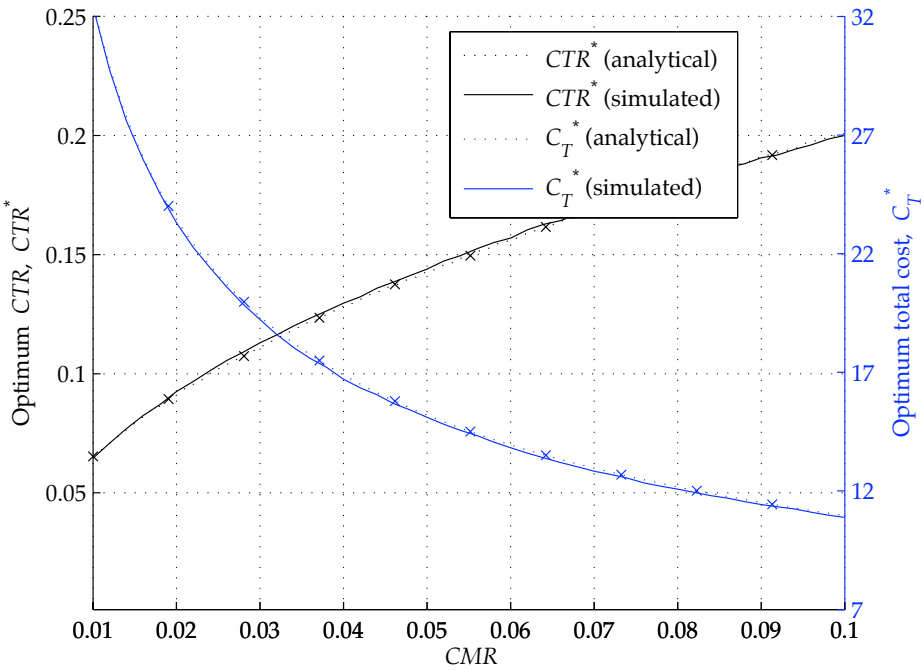


Figure 6.10: Equivalent to figure 6.6, but improved by curve fitting

6.6.2 Study 2: Original Scheme; Improved SAN

Used SAN Model

In the model of Study 1 the cost performance variables depend on the simulation time, so that the performance variables have to be normalised by the simulation time to get the actual costs. This is considered a drawback as the dependence has to be considered when changing the simulation time, elsewise leading to wrong results. More to the point is however that the Möbius feature of calculating the confidence interval cannot be used adequately. For the model of Study 1 it would be appropriate to simulate as long as until the delta of the cost per call would be less than a given threshold. However this is not possible since the simulation time is a fixed variable in Möbius that cannot be changed during

execution of the simulation. In an improved model which is presented in the next paragraph the costs are directly available from the performance variables and thus a suitable termination condition can be used, which depends on the accuracy of the performance variable.

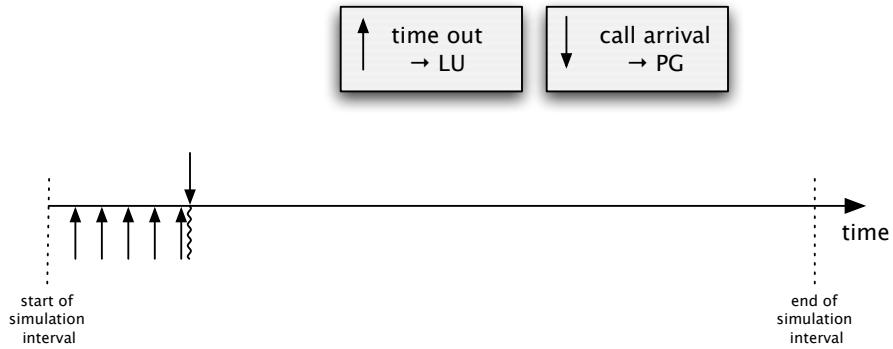


Figure 6.11: Discrete events of the stochastic MM point process; in the improved model the observation interval is not fixed manually but as long as the intercall time

Figure 6.11 shows the principle of the observation of the stochastic process in the improved way. The LUs are counted only up to the point where a call arrives. When a call arrives the PG cost is calculated according to the ring the MT is located at that point in time. After that no more call arrivals and no more LU events are counted. The advantage in this method is that the LU and PG costs are normalised to one intercall time. By averaging over a lot of batches, the mean LU and PG costs between two call result directly.
The simulation time is set to a much higher value than the observation time to guarantee that no calls occur outside the simulation time. In this simulation the simulation time is set to 100 s which is 100 times the mean intercall time of 1 s. As the intercall times are negative exponentially distributed the probability for an intercall time to be greater than t is

$$F^{C}_{\text{exp}}(t) = e^{-\lambda t}$$

and with $\lambda = \frac{1}{\text{s}}$ and $t = 100\text{s}$

$$F^{C}_{\text{exp}}(100\text{s}) = e^{-\frac{1}{s}100s} \approx 3.7 \cdot 10^{-44}$$

which is fairly and negligible small. The high simulation time does not effect the execution time however, as, like chapter 6.4.2 (event list handling) explained, new events are only created when the conditions for them to happen are fulfilled. So, as after the first incoming call all events in the model are disabled, no new events are created and thus the next event to happen is the termination of the simulation.

Results

The results of this study matched the results of study 1. The truncation effect (because of the fixed simulation time) of Study 1 is therefore negligible. Therefore no graphs or further results are given for this study. Nevertheless the improved SAN model was used in all following studies, because of the advantage of obtaining the costs directly, i.e. without the necessity to consider the simulation time.

6.6.3 Study 3: Modified Scheme

The modelling for this scheme is very similar to Studies 1 and 2, with the following differences:

- on firing of the timer activity *no* LU is issued
- during crossing of cells it is checked whether a timeout happened before and a flag is set correspondingly
- if the flag (represented by a SPN-place) is set, a LU is issued and counted immediately

For details on this model and the simulation parameters it is referred to appendix D.5. There the model for the modified Reset & Wait scheme is given for local area of $H = 0$. For Study 3 the only difference is that the `local_area`-variable H is set to `-1` instead, i.e. there is no local area. For an explanation of the local area parameter H, see chapter 4.

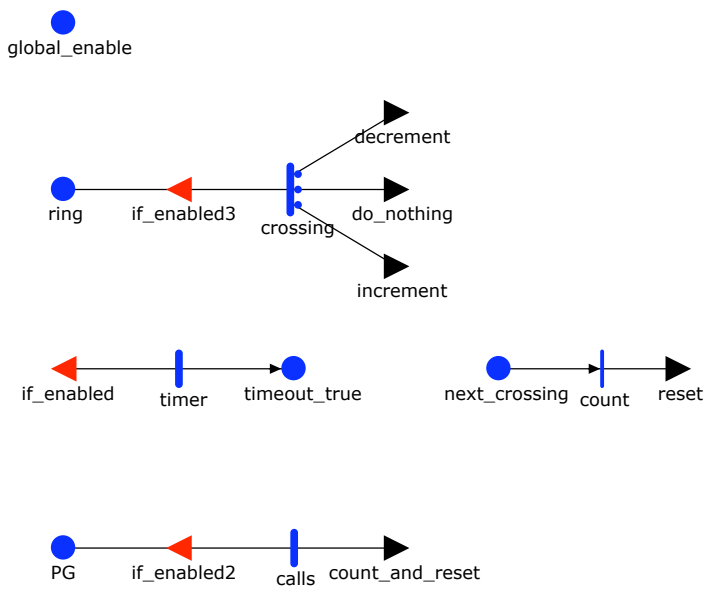


Figure 6.12: SAN modelled in Möbius for time-based modified R&W MM

Central element in the model shown in figure 6.12 is the movement part with the three possible outcomes to move to the inner ring (`decrement`), to stay in the same ring (`do_nothing`) and to move to the outer ring (`increment`). Below that, the timer activity is shown which is directly connected to the `timeout_true`-place. If the timer fires this place gets the value 1. The next crossing of the activity `crossing` will activate then `next_crossing` and thus be counted as LU. The ring location is reset and the timer is restarted. The lower path in the model shows the call activity. It is different in comparison to the model of Study 1 only in the way that the whole model will be stopped after the first occurrence of a call by disabling the `global_enable`-place.

Figure 6.13 shows the results of the simulation. 46 *CMR* values were generated, each with 37 *CTR* values for finding the optimum *CTR*. No fitting was used for this simulation. The total cost curve covers the analytical curve very well, while the *CTR* curve has variations again due to the broad minimum. The mean error of the total cost is $\overline{\eta}_{C_T^*} = 0.39\%$ whereas for the *CTR* it is $\overline{\eta}_{CTR^*} = 3.3\%$.

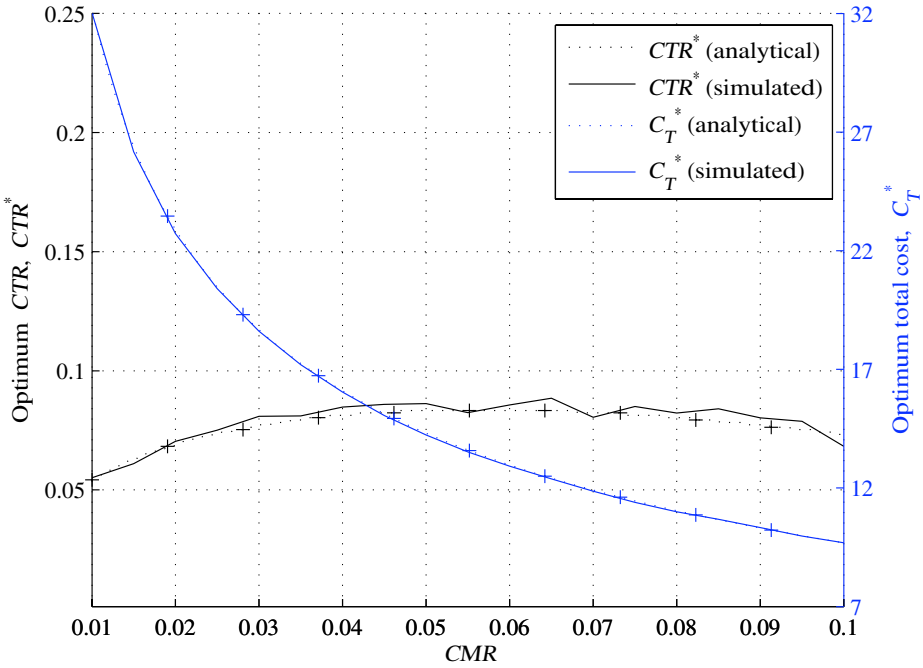


Figure 6.13: Total cost C_T^* and optimum call-to-timeout-ratio CTR^* for the Modified Scheme with analytical results from [9] for comparison

6.6.4 Study 4: Stop & Wait Scheme, H=0

To model this strategy the original SAN was altered with respect to the LU/timer part. However, the general mobility model with the cell crossings is retained as well PG/calls part. The timer part although is more complex. On leaving the local area the `now_outof_LA` the `save_time` function (see in the middle of figure 6.14) is executed which calls the internal Möbius function `BaseModelClass::LastActionTime` to obtain the current simulation time and save it in a temporary variable and then the timer is enabled. If the timer expires a LU message is "sent" and counted. If however the MT moves back into the local area before the timeout expired, the difference of the previously saved time and the current time is calculated and this time span is subtracted from the original timeout value for the running experiment. This is since only the remaining time has to be used as the timeout value the next time the timer starts running, i.e. when the MT has left the local area once more. After the LU message the timeout value has to be reset to the original value for the running experiment. The further details on the used model can be found in appendix D.2.

For Study 4 a local area distance of $H = 0$ was chosen, to make it possible to compare the outcomes to the analytical values in the above cited paper ([9]).

23 *CMR* values have been simulated for the x-axis, sweeping though 25 *CTR* values for each *CMR*. The result graph is shown in figure 6.15. The mean deviation between the analytical and the simulated curves is $\overline{\eta}_{C_T^*} = 0.39\%$ for C_T^* and $\overline{\eta}_{CTR^*} = 3.3\%$ for CTR^* respectively. Fitting as explained in Study 1 was used.

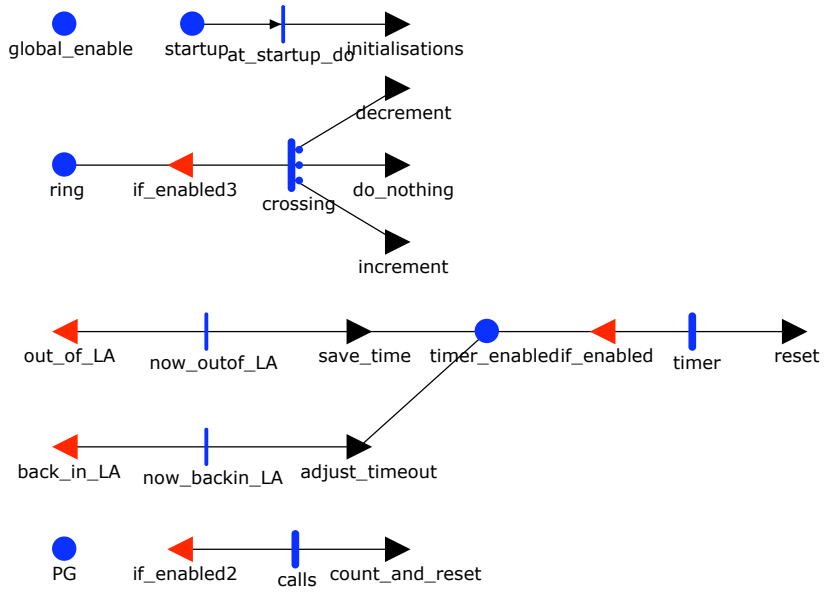


Figure 6.14: SAN modelled in Möbius for time-based S&W MM

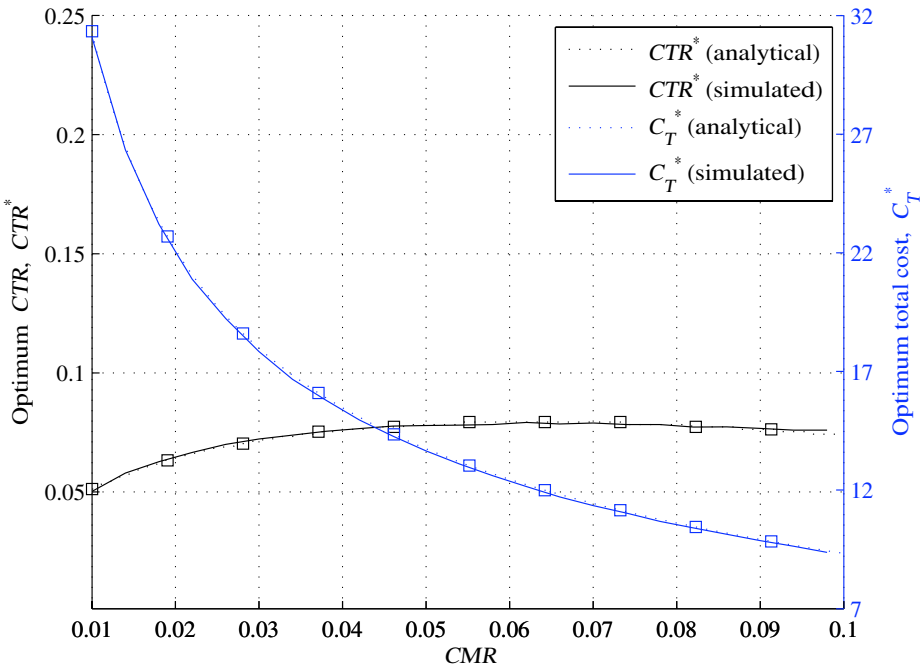


Figure 6.15: Total cost C_T^* and optimum call-to-timeout-ratio CTR^* for the Stop & Wait Scheme with analytical results from [9] for comparison

6.6.5 Study 5: Modified Stop & Wait, H=0

For simulating this method the model from Study 4 (section 6.6.4) was enriched with some parts from the model for the modified scheme from Study 3 (section 6.6.3). In this model the time spent out of the local area is saved on entering the local area. After leaving the local area once again, the timer is restarted with the remaining time. The additional component in the model is that after the timeout happened no immediate LU is triggered. It is only with the next crossing (`crossing`-activity fires) that the LU procedure is triggered. For details see appendix D.3 also.

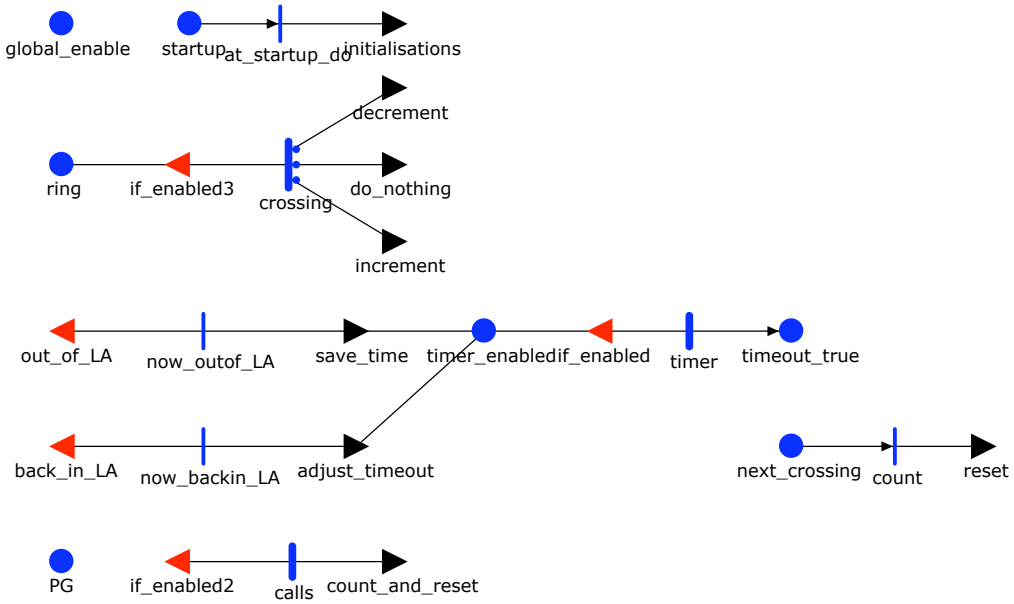


Figure 6.16: SAN modelled in Möbius for time-based modified S&W MM

The outcome of the scheme is given in figure 6.17. The simulated total cost match the analytical costs very good. However the *CTR* values differ again for the above given reasons. But since for this model the optimal *CTR* values are in the close range of zero the percentage error becomes very high and is not given here.

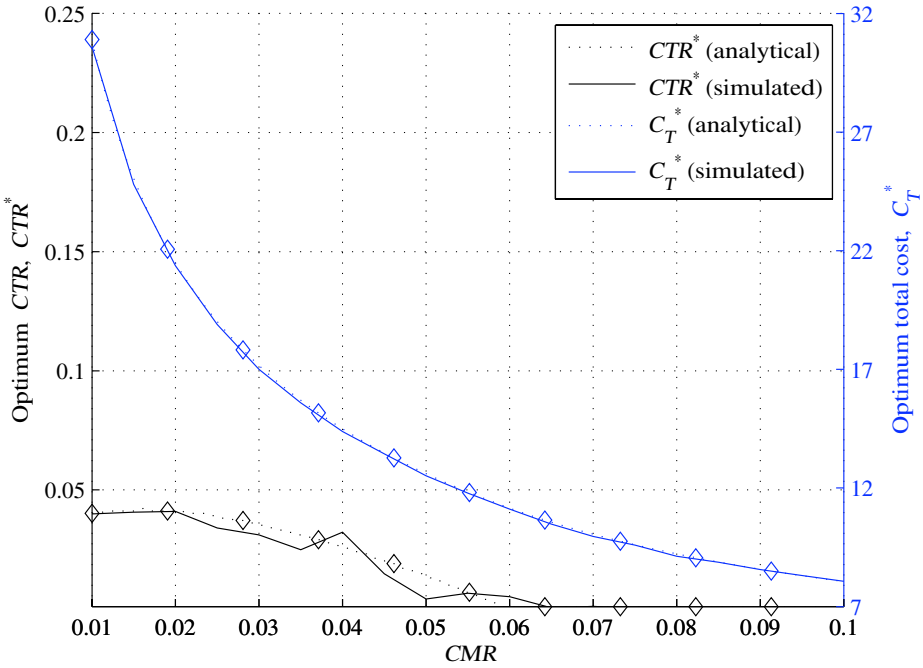


Figure 6.17: Total cost C_T^* and optimum call-to-timeout-ratio CTR^* for the Modified Stop & Wait Scheme with analytical results from [9] for comparison

6.6.6 Study 6: Reset & Wait, H=0

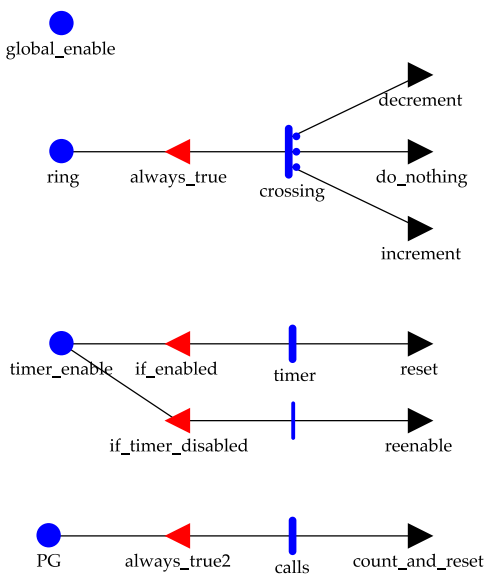


Figure 6.18: SAN modelled in Möbius for time-based R&W MM

The model in figure 6.18 resembles the model from Study 1. This is since the model changed only slightly between those studies. For details see appendix D.4. For the original scheme in Study 1 the local area parameter $H = -1$ was set, i.e. no local area is present. In the proposed Reset & Wait mechanism the

local area is present and given by the parameter $H, H > 0$. To make results comparable to [9] again the parameter was fixed to $H = 0$. The only visible difference now in comparison to figure D.1 is the presence of the `global_enable`-place. After the first call this place is disabled so that no further activities in the model can fire. Therefore the second branch of the timer activity, which re-enables the timer if it is disabled is not used in this model, but is a relict of the model approach which was summing up for 100 s.

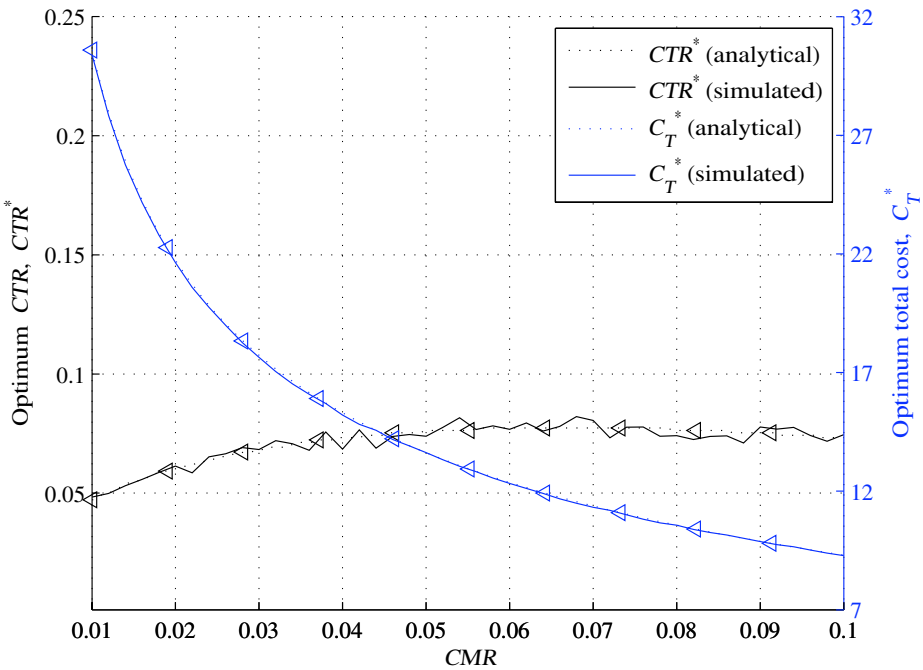


Figure 6.19: Total cost C_T^* and optimum call-to-timeout-ratio CTR^* for the Reset & Wait Scheme with analytical results from [9] for comparison

The result graph in figure 6.19 shows a significant reduction in the total cost in comparison with the previous studies. The reason for this is that the timer is reset every time the local area is entered. The PG costs are not increased by this measure for $H = 0$, as the process renews at that point, i.e. it starts as if the last contact with the network had happened just then.

6.6.7 Study 7: Modified Reset & Wait, H=0

For simulating this method the model from Study 6 (section 6.6.6) was enriched with some parts from the model for the modified scheme from Study 3 (section 6.6.3). The used model is the same as in the modified scheme in section 6.6.3, but this time with $H = 0$ instead of $H = -1$. Therefore it is referred to figure 6.12 and the model is not shown here. For further details see appendix D.5 also.

Figure 6.20 shows the results for this scheme. The total costs are reduced to a minimum for this scheme, in relation to the other schemes. Note also that for growing *CMR* the optimal timeout value T_{LU} fastly converges to zero. In praxis this means, there is no timer needed at all, for users with a mobility profile in the *CMR* region in question. The scheme would therefore result in a distance-movement scheme, with the H determining the distance component and the delayed (until next cell crossing) LU operation the movement component.

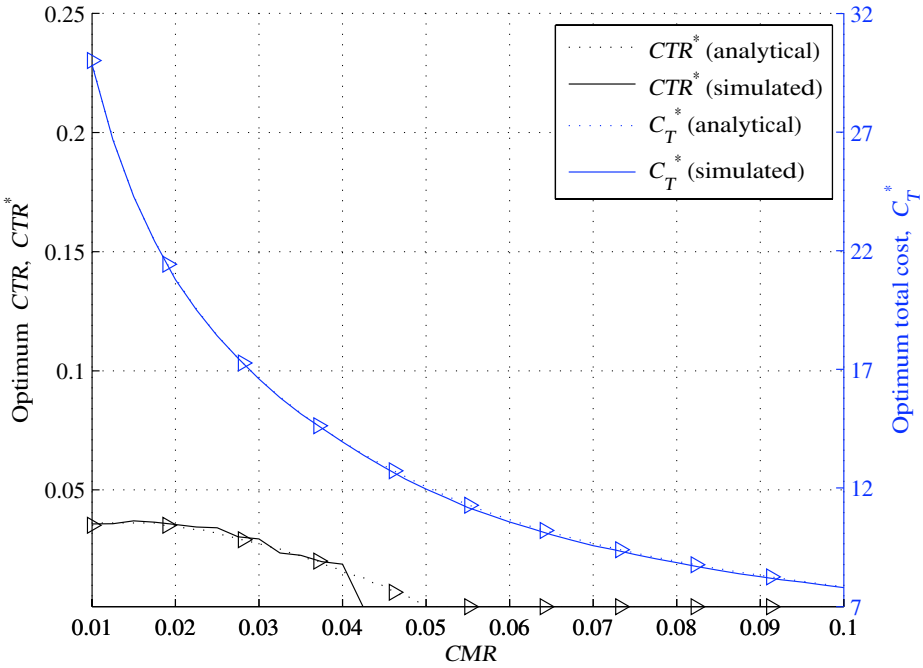


Figure 6.20: Total cost C_T^* and optimum call-to-timeout-ratio CTR^* for the Modified Reset & Wait Scheme with analytical results from [9] for comparison

6.6.8 Study 8: Deterministic timer vs. Coxian phase-type timer

In real world systems timers are mostly deterministic, as the system is optimised for a specific deterministic value and there is no reason to variate this value. Furthermore the implementation in Software gets easier and the use of interrupt-generating hardware-implemented timer functionality on microcontrollers is possible. But for analytical calculations the phase-type decomposition of deterministic "Random Variables" into exponential stages is advised (cf. chapter 2.2.3 and [15]).

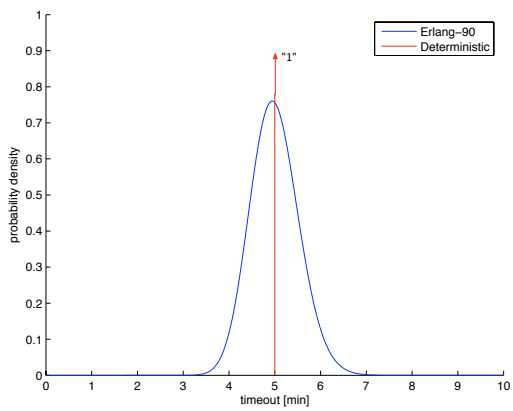


Figure 6.21: Timer value of 5 (minutes) with deterministic pdf and with Erlang-90 distributed pdf

For deterministic RV the phase-type decomposition in k exponential stages results in an Erlang-k distribution. The number of stages k determines thereby the accuracy of the approximation (cf. chapter 2.2.3). In [9], $k = 90$ was chosen which results in a coefficient of variation of $c_v \approx 10.5\%$. To investigate the deviation in the total model that is made by this approximation, two Möbius models where compared by simulation: one with deterministic timer and one with the above stated timer with Erlang-90 distribution. The two pdfs are given in figure 6.21. For the deterministic RV the pdf is a Dirac-impulse.

The comparison was conducted only for one model exemplarily. The findings should apply to all models in a similar order of magnitude. So the model chosen is the Study 7 (modified R&W, H=0). The confidence interval was chosen to 1 % relative error with 95 % probability.

The output graphs are shown in figure 6.22. We can see that the costs tend to be smaller when using a Erlang distribution in comparison to a deterministic timer. Nevertheless the difference is within the confidence interval which means within the accuracy of the simulation. We conclude that even though the coefficient of variation is about 10 % the difference in the difference in the simulated cost outcome is less than 1 %. This again is a result of the broad minimum of the C_T-versus-CTR curve.

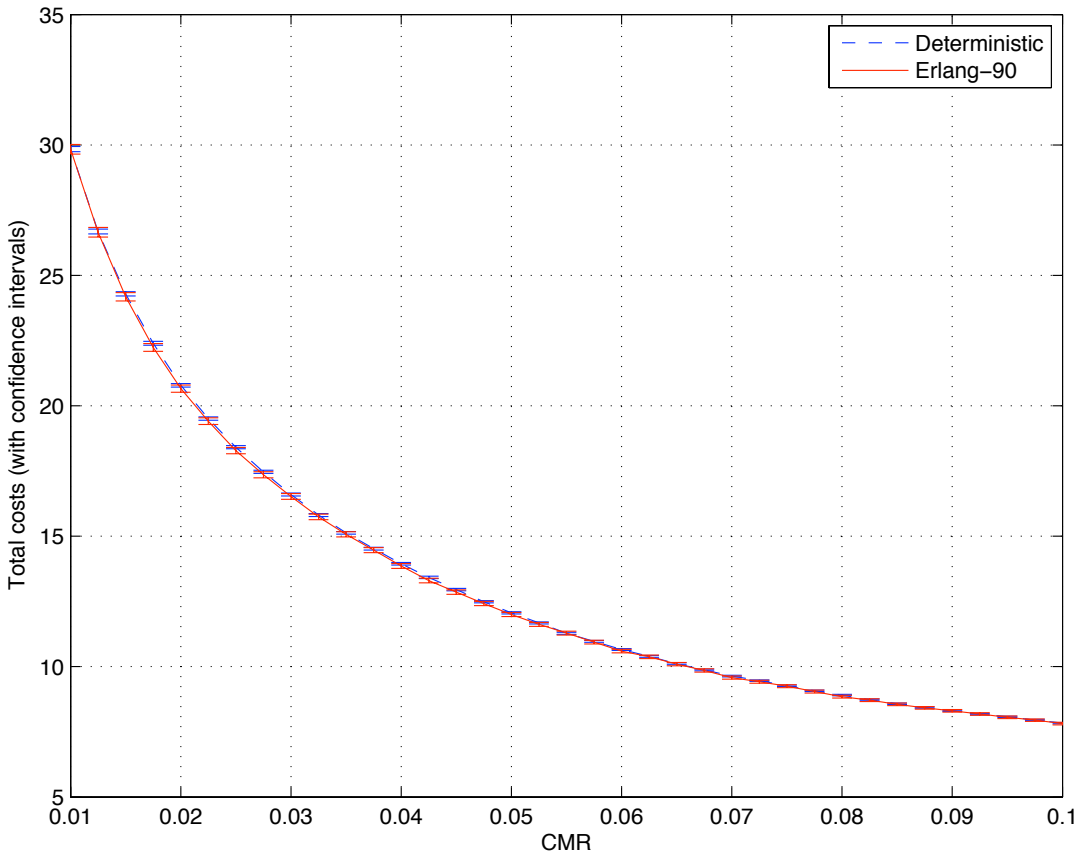


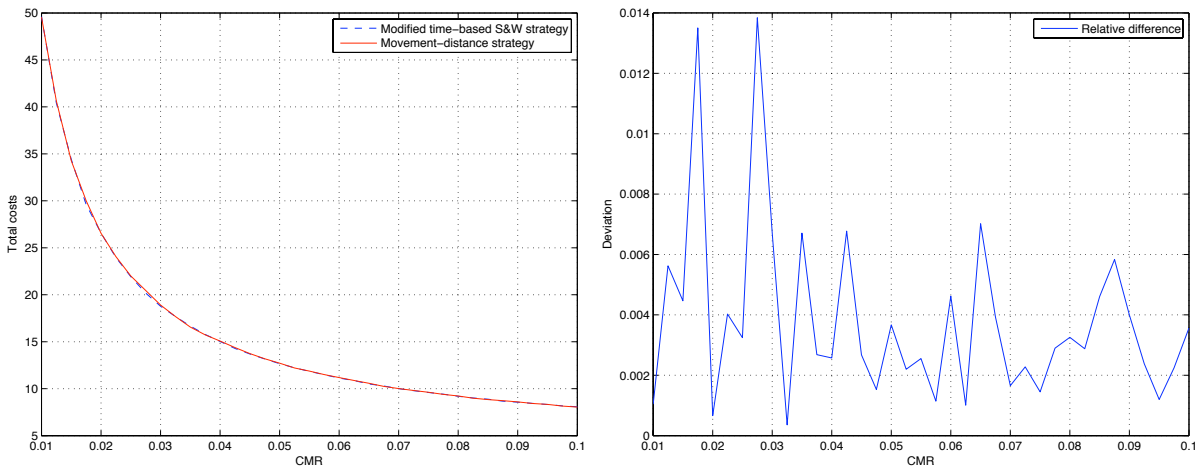
Figure 6.22: Total-cost function for deterministic and Erlang-90 distributed timer; the differences lie within the accuracy of the model

6.6.9 Borderline cases

Two special cases were researched to check the validity of the modified S&W and modified R&W models. These two models converge to a movement-distance based strategy for timer values $T_{LU} = 0$. (compare also chap. 4.7 and [9], chap. 2.2.2).

For the modified S&W strategy with timer $T_{LU} = 0$ and local area $X(H)$, the model is equivalent to a movement-distance strategy with movement threshold $d = 2$ and local area $M(H)$, where the movement counter is reset on entering the local area.

The results show that the models yield indeed the same total cost for both models. The lines lie on each other nearly perfectly. See figure 6.23 for details. 37 call-to-mobility values (from 0.01 to 0.1 in steps of 0.0025) were simulated for the x-axis. As the timer value as well as the movement counter are not subject to an optimisation but fixed values in this cross-check simulation the total number of experiments in this simulation is only 37.



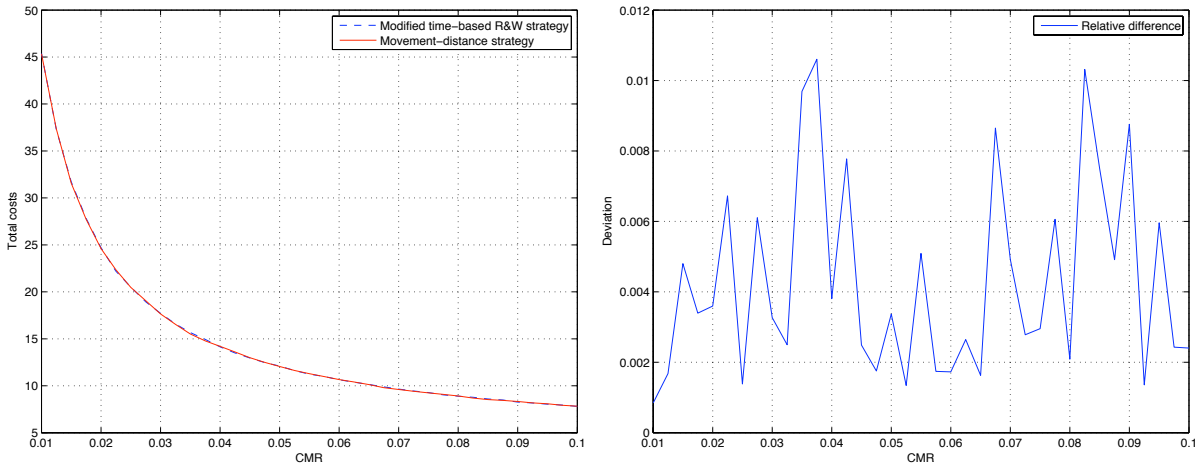
(a) Total costs (b) Deviation between models

Figure 6.23: Comparison of time-based vs. movement-distance-based S&W strategies

The fitting is an indicator that the more complex modified S&W model is valid.

For the modified R&W strategy with timer $T_{LU} = 0$ and local area $X(H)$, the model is equivalent to a movement-distance strategy with movement threshold $d = 2$ and local area $M(H)$, where the movement counter is reset on entering the local area.

The results show that the models yield indeed the same total cost for both models. The lines lie on each other nearly perfectly. See figure 6.24 for details. 37 call-to-mobility values (from 0.01 to 0.1 in steps of 0.0025) were simulated for the x-axis. As the timer value as well as the movement counter are not subject to an optimisation but fixed values in this cross-check simulation the total number of experiments in this simulation is only 37.



(a) Total costs

(b) Deviation between models

Figure 6.24: Comparison of time-based vs. movement-distance-based R&W strategies

7 Conclusions and Outlook

The methods proposed in [9] have been investigated with respect to their savings in mobility management costs. It can be concluded that the proposed schemes, more precisely the Modified Reset & Wait scheme, should be implemented in current networks for several reasons. The improvement in signalling effort ranges from 8 % to 28 % in the range of the researched call-to-mobility ratio for the Modified Reset & Wait scheme. This gain can be yielded with nearly no effort. Even if only *one* memory place is used to store the cell ID of the last cell where a LU occurred (or a call terminated, respectively) the above mentioned improvement results which is notable. With todays mobile terminals microcontroller-based processor architecture and on-chip memory (e.g. the Intel PXA architecture, Freescale Semiconductor MXC300-30 or the Texas Instruments OMAP architecture), the hardware effort to implement this method is negligible. The algorithm is simple and therefore bears no dangers of unwanted side-effects.

Even though the method is simple to implement in the MT, the network has to implement some feature to derive the optimal timeout value T_{LU} by observing the users past mean *CMR* value. This can be avoided by setting a fixed timeout T_{LU} for all subscribers, which is sub-optimal but not far from the minimum, as the cost is not very sensitive to the timeout. If the Modified Reset & Wait scheme is used, the timer can even be $T_{LU} = 0$, i.e. no timer functionality has to be used. The proposed scheme corresponds to a movement-based scheme (cf. chapter 6.6.9) with movement threshold of $M = 2$ then. But even when the proposed scheme is used with $H > 0$ and $T_{LU} > 0$ the additional effort is marginal but the savings effect is notable.

As next step in the investigation of the proposed schemes, it is proposed to simulate more realistic cases, where some of the strong restrictions in the model are loosened. Interesting cases are the behaviour under

- different mobility models (no random walk, but directional)
- different cell layout, with irregular shapes, overlay networks and just different cell sizes
- different parameter ranges, e.g. for the *CMR* range

If after some more research the schemes manifest to bring the wanted improvement effect over a broad range of conditions, standardisation within the 3GPP or ETSI committees should be striven for.

Acknowledgements

I owe many thanks to many people that helped rendering this work possible. Firstly I like to thank the circle of people at the guest university, the Universidad Politécnica de Valencia (UPV), namely Professor Vicente Casares-Giner and Dr. Pablo García-Escalle, for welcoming me so heartily and for their good support in any kind. Dr. García tutored me in an excellent way, by giving help where needed and challenging me at the same time. For assistance in all administrative duties at the UPV I would like to thank Jesus Alonso-Urbano and his student assistants.

The second circle of people I owe thanks to is at my home university. Professor Kühn not only allowed me to do the external work at the UPV but at the same time established the contact to his colleague at the UPV and convinced him of taking the extra effort of taking me as a student. Dr. Wizgall of the InfoTech Office supported me in the forefront of the staying abroad by giving many helpful tips. The Office of International Affairs of the University of Stuttgart and the Erasmus delegate of my faculty helped me with the necessary administrative obligations.

In the deepest dept however I am to Sandra Möller who only agreed with a heavy heart to my leaving for nearly nine months. She gave me mental encouragement during my staying abroad and moreover supported me in the scheduling, structuring and correction of this work.

I also always enjoyed the chats I had with my lab colleagues and their introductions to Spanish and Valencian culture and customs. Furthermore I would like to thank my diploma fellow Fabian Thrum with whom I had many fruitful technical discussions and I wish him to find a great career perspective. I also thank all the other friends I became acquainted with, for the good times we spent together and for their everlasting patience in teaching me a little Spanish.

Finally I would like to thank the European Union for the Erasmus program that facilitate student exchanges across Europe and for the financial support.

A Möbius Distributed Simulation

With the transition from UltraSAN to Möbius the ability for distributed simulation was introduced. With this method the single experiments of a complete study are spread over the available Mobius-machines for computation. The advantage of this is clearly a shorter total simulation time for a given study model. As the concept is not given in detail in the manual, the basics of distributed simulation are summarised in the following.

To use remote simulation on both the client and the server remote connection command have to be available. Möbius allows the use of either `rsh` (*remote shell*) or `ssh` (*secure shell*) commands. Which remote shell command is actually used is entered to Möbius in menu SETTINGS – OS COMMANDS – REMOTE SHELL COMMAND of the Project Manager Window (see figure A.1).

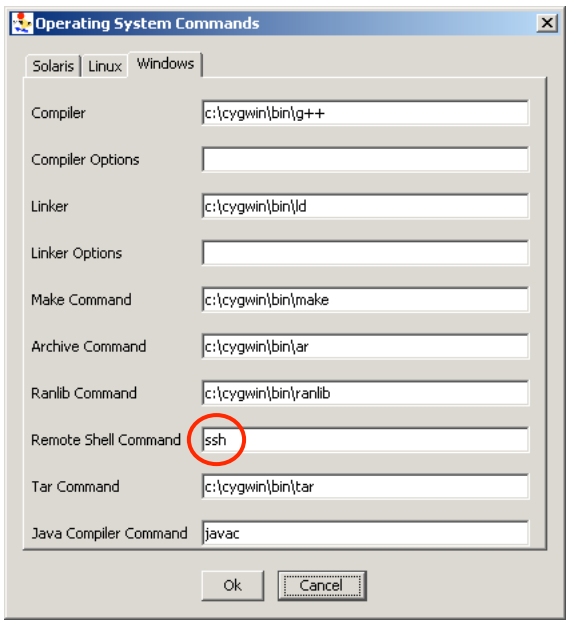


Figure A.1: The OPERATING SYSTEM COMMANDS window

To install rsh or ssh on Windows® Operating Systems the use of Cygwin™ is the premier choice as Cygwin has to be installed anyway to use Möbius. For the use of `rsh` the *inetutils* package of the *Net* category has to be installed, which is not selected by default. For `ssh` the *openssh* package of the *Net* category has to be installed, but this is selected anyhow in the default installation. In the following the configuration and use of `ssh` is shown in detail, as the Möbius manual is not very extensive on these topics.

On all machines that will be used as simulation servers, the sshd (*ssh deamon* = ssh server) has to be set up. In current Cygwin installations, a configuration script is delivered which simplifies the configurations of sshd a lot. Initiating the command by

```
ssh-host-config
```

the following tasks are executed:

- generation of RSA and DSA keys in `/etc/` by use of the `ssh-keygen` command
- generation of the `/etc/ssh_config` file (configuration for outgoing ssh)
- generation of the `/etc/sshd_config` file (configuration for incoming ssh)
- installation of sshd as a Windows service

Besides these main jobs the script takes care about many details during execution, like checking for already existing files, removing those, copying results to the right places and more.

By using

```
ssh-host-config -y
```

the configuration is further simplified as all questions asked during script execution will automatically be answered by *yes* which is fine for a new installation.

Finally the Windows service daemon is started by

```
cygrunsrv -S sshd
```

and the ssh server is operable on the simulation server. It is now possible to `ssh` to the server from any other machine. To try this enter

```
ssh <RemoteMachine> -l <UserName> ls
```

and the home directory listing of user `<UserName>` on `<RemoteMachine>` should be printed. If this does not work it might be because of a firewall on the server. Ensure that TCP port 22 (ssh) is open. For how to do that rely on your system's firewall documentation, as the options vary greatly. If after opening port 22 it still might not work it might be that the account `<UserName>` is an password free Windows account.

To stop the service daemon use

```
cygrunsrv -E sshd
```

but this is only given for sake of completeness at this point.

Möbius is not able to use password protected ssh connections. Authentication thus relies on the key-concept of ssh. For this reason the keys generated by `ssh-keygen` have to be without passphrase. Be sure to just press return if asked for the passphrase (but the `ssh-host-config` script does not ask anyway). Additionally, the keys generated, which identify the client machine, have to be copied to every server machine.

Firstly, generate keys on the client by running the

```
ssh-keygen
```

command which generates the necessary keys and states the storing location. Usually something like `~/.ssh/id_rsa` or `~/.ssh/id_dsa` for the private key and `~/.ssh/id_rsa.pub` or `~/.ssh/id_dsa.pub` for the public key are created. To copy the public key `~/.ssh/id_rsa.pub` (or `~/.ssh/id_dsa.pub`) from the client to the server machines enter

```
scp ~/.ssh/id_rsa.pub <UserAtServer>@<Server>:~/.ssh/authorized_keys
```

or respectively

```
scp ~/.ssh/id_dsa.pub <UserAtServer>@<Server>:~/.ssh/authorized_keys2
```

on the client machines command line.

By this the client's key gets into the server's trusted key list, so the client gets authorised automatically on connection setup and it is not prompted for the password, which is required for Möbius.

Be careful with the `scp` command though, as it replaces the `authorized_keys` file. If you already have an `authorized_keys` file on this machine, it is of course possible to just add the new key to the existing ones. An editor can be used for this or the new key can be appended by the `>>` file operator.

For example use

```
scp ~/.ssh/id_rsa.pub <UserAtServer>@<Server>:~/.ssh/tmp_auth_key
ssh <UserAtServer>@<Server> "cd .ssh; cat tmp_auth_key >> authorized_keys"
```

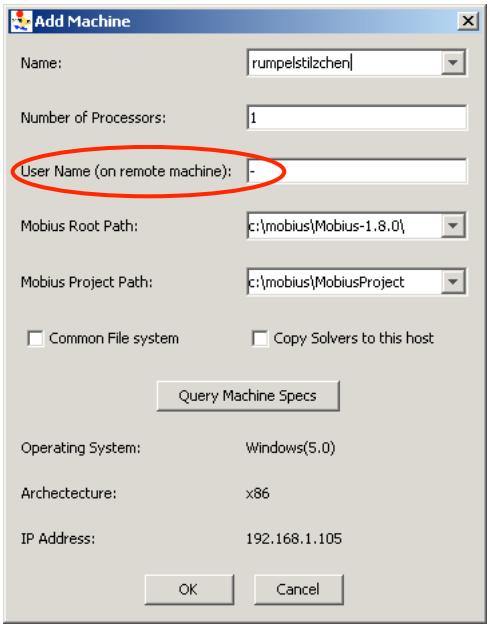


Figure A.2: The ADD MACHINE dialogue window; the User Name field is not working yet (as of version 1.8.3)

On the simulator client, ie, the machine that triggers the simulation on behalf of an user request, only the ssh client is required. In general the ssh client needs no further configuration. But since a bug in Möbius, which prevents the correct use of the user field in the menu SETTINGS – NETWORK MACHINES – ADD MACHINE additional configuration is needed for the distributed simulation to work. The problem in this place is that the users name on the client machine and on the several server machines is not necessarily the same. As mentioned before this couldn't be helped with the user field of the corresponding Möbius dialogue (compare figure A.2), so a `ssh_config` file has to be generated on the client machine. In this `ssh_config` file users names for hosts can be indicated. For this run

```
nano ~/.ssh/config
```

which will open the `ssh_config` file or generate a new one if it does not exist on the client yet. Instead of `nano` any editor can be used of course.

In this file add a line

```
User <UserName>
```

where `<UserName>` is the name you would have to state with the `-l` option of `ssh` otherwise. If several servers are used for distributed simulation and on all of these machines different user accounts have to be used, in the `ssh_config` file for every host a User entry has to be added, for example:

```
Host <ServerX>
  User <UserNameOnX>
Host <ServerY>
  User <UserNameOnY>
```

For further details on the structure and other configuration possibilities of the `ssh_config` file see the man page

```
man ssh_config
```

After having created the `~/.ssh/config` file make sure only the owner has write permissions – otherwise the use of the `~/.ssh/config` fails. Use for example

```
chmod 700 ~/.ssh/config
```

for that purpose.

After having followed these steps a Query Machine Spec button should work and OS, Architecture and IP address of the server should be stated in the corresponding fields of the dialogue window A.2. If it does not work yet it might be of an unacknowledged RSA key fingerprint which Möbius is not able to accept on its own. Make sure the to log in to the server by hand once and accept the RSA key fingerprint of the server so that server gets to the client's `known_hosts` file, by executing

```
ssh <RemoteMachine> ls
```

again, but note that other than before the `-l` command for specifying the user name is not required any more as the `~/.ssh/config` is used now. The answer to a question like

```
RSA key fingerprint is 89:7e:08:24:67:c4:9f:24:14:30:08:13:73:02:7b:c2.
Are you sure you want to continue connecting (yes/no)?
```

should be *yes*, if the fingerprint of the host is correct. Finally Möbius is ready for distributed simulation.

A good introduction to `ssh` can be found at [31].

B Perl Scripting

Output files of Möbius contain a lot of information, as data (the actual simulated performance measure *values*) and meta data (describing the *circumstances* of the simulation and giving information *about the type* of data) are intertwined. For quick evaluation of the results the information content is too big. The main interest is in the obtained values of the Reward Variables. In the predecessor of Möbius, UltraSAN, the integrated graphing feature made quick evaluation possible. With Möbius the extraction of these values from the result file is necessary, for which Perl has been used during this work.

Two result files are generated by Möbius. A plain text (.txt) file and a comma separated value (.csv) file (see excerpt of this file at B.1 for one experiment) are generated. The latter has the advantage of being directly openable in spreadsheet programs like *OpenOffice Calc*, *Microsoft Excel* or *The MathWorks Matlab*.

```
Experiment 1
Experiment 1,Start Time:,Mon May 15 13:10:50 CEST 2006
Experiment 1,Finish Time:,Mon May 15 13:10:55 CEST 2006
Experiment 1,Elapsed Running Time:,4.726
Experiment 1,Total CPU Time:,4.006
Experiment 1,Batches Completed:,200
Experiment 1,Experiment Name:,Experiment 1
Experiment 1,Global Variable Settings:
Experiment 1,double,end_time,100.0
Experiment 1,short,local_area,-1
Experiment 1,double,r_call,0.1
Experiment 1,double,r_cell,10.0
Experiment 1,double,timeout,0.4
Experiment 1, Results
Experiment 1,Result Type,Name,Start Time,End Time,Mean,,Confidence Interval,
 Failed To Converge
Experiment 1,Mean,LU,0.0,end_time,|2.4432500000E02 ,+/-,2.5503759699E-01
Experiment 1,Mean,PG,end_time,end_time,1.1043500000E02 ,+/-,6.6586470660E00
```

Figure B.1: Excerpt of Möbius .csv output file

To extract the relevant information the Perl-Script B.2 is used. After having read the input file (which is the Möbius output file), it looks sequentially for one of the lines with a data value and then separates the value from the meta data contained in that line. The extracted data is written to a new output file. The fragment of B.1 would result in the single line

```
10.0,0.4,2.4432500000E02,1.1043500000E02
```

in the new output file, which contains again comma separated values for easy importing in Matlab. In Matlab the importing of csv-files results in a matrix containing the values. The further processing can be done using the powerful Matlab-routines.

```
# 1: Get command line values:
    if ($#ARGV !=1) {
        die "Usage:  $0 inputfile outputfile\n";
    }
    ($infile,$outfile) = @ARGV;
    if (! -r $infile) {
        die "Can't read input $infile\n";
    }
    if (! -f $infile) {
        die "Input $infile is not a plain file\n";
    }

# 2: File I/O
open(IN, $infile);
@lines = <IN>;                # Read it into an array
close(IN);                    # Close the file
open(OUT, ">$outfile");       # Open output file

# 3: Match interesting parts and write to output file
$finding = 0;
foreach $line (@lines) {
    if ($line =~ /double,r_cell/) {
        $line =~ /\d+\.\d+/;
        print OUT $&;
        print OUT ",";
        $finding++;
    }
    if ($line =~ /double,timeout/) {
        $line =~ s/Experiment.\d+,double,timeout,//;
        chomp($line);
        chop $line;
        print OUT $line;
        print OUT ",";
        $finding++;
    }
    if ($line =~ /LU,0.0,end_time/) {
        $line =~ s/Experiment.\d+,Mean,LU,0.0,end_time,//;
        $line =~ s/\s*,+.*//gi;
        $line =~ s/\n+/,/gi;
        print OUT $line;
        $finding++;
    }
    if ($line =~ /PG,end_time,end_time,/) {
        $line =~ s/Experiment.\d+,Mean,PG,end_time,end_time,//;
        $line =~ s/\s*,+.*//gi;
        $line =~ s/\n+/,/gi;
        print OUT $line;
        $finding++;
    }
    if ($finding == 4) {
        print OUT "\n";
        $finding = 0;
    }
}
close(OUT);
```

Figure B.2: Perl script for extracting data from csv-file

C Matlab source

C.1 data_analyse.m

The data_analyse-Tool is written as a Matlab function. The function is configurable by five parameters and has no return value. If no input parameters are given, the interactive mode is activated.

Input parameters (with allowed range in parentheses):

- chosenCurve; (1-6); to chose the model
- numCMR; (integer); number of simulated CMR values
- debugPlotsOnOff; (y/n); if intermediate plots should be displayed
- errorPlotsOnOff; (y/n); if error plots should be generated
- fittingChoice; (0-2); what kind of curve fitting should be used for finding the minimum (no/$\frac{1}{x}$/polynomial fitting)

The coarse function structure is given in the following. First some checking for invalid parameters is done. Then, if curve fitting was chosen, the total-cost-over-CTR curve is fitted by using the method of minimum squares. The resulting function is then interpolated at 5000 points to get a smooth curve and thus an exact minimum. If intermediate plots were chosen to be displayed these show the original data as well as the fitted and interpolated curves, together with the minimum points. In the next step the main graph is generated where the analytical result from [9] and the simulated results from this work are opposed each other. In the last step the deviation between the both curves is computed and both plotted and given as mean deviation.

In the following the used data structure is explained briefly: The obtained costs, that have been esti-

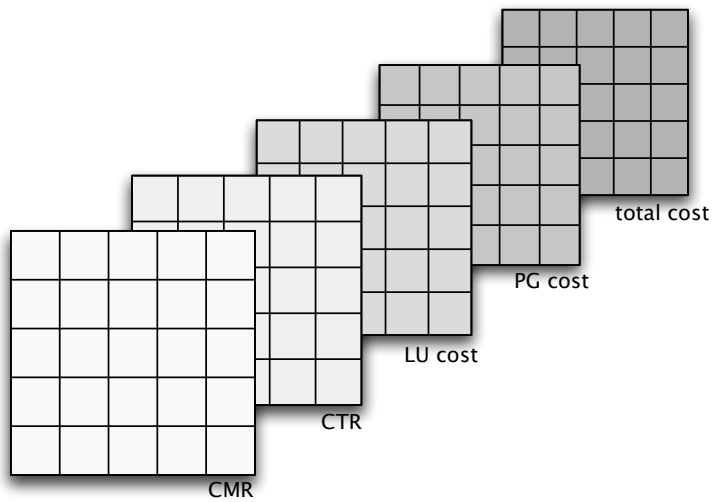


Figure C.1: Layout of the complete 3D *data*-array

	1	2	3	4	...	...
1	0.05	0.06	0.07	0.08		
2	0.05	0.06	0.07	0.08		
3	0.05	0.06	0.07	0.08		
4	0.05	0.06	0.07	0.08		
⋮						

Figure C.2: Layout of the *CMR*-array in *data*-array

mated by the simulation, depend on both the Call-to-Mobility-Ratio (CMR) as well as on the Call-to-Timeout-Ratio (CTR). Thus a three-dimensional array layout as shown in figure C.1 was used to handle the data efficiently. The LU cost and PG cost are added and result in the total cost, which is added to the data structure as a new slice. Further slices are added to the data structure if necessary. Slice 1 and slice 2 represent the CMR and the CTR, which are part of the data structure as two-dimensional slices, albeit being one-dimensional only. The reason for that lies in the fact that the cost data cannot be referenced by the CMR and CTR directly, as Matlab solely allows integer indices. Therefore the CMR/CTR values has to be stored separately to make correct referencing possible. The layout of one of the 2D-slices of figure C.1 (the CMR-slice in fact) is shown in figure C.2.

```
function data_analyse2(chosenCurve, numCMR, debugPlotsOnOff, errorPlotsOnOff,
fittingChoice)
%
%plot_data: extract data from Mobius generated file and plot
%

close all; %close all open figure windows

if nargin == 0
    %enter interactive mode
    curveChoices = '\n Please select the curve to compare with: \n  [1]: Original Scheme
\n  [2]: Modified Scheme \n  [3]: S&W (H=0) \n  [4]: S&W (H=0) (modified) \n  [5]: R&W
(H=0) \n  [6]: R&W (H=0) (modified) \n  \n ';
    chosenCurve = input(curveChoices);

    numOfCMRs = '\n How many CMR values have been calculated? \n \n ';
    numCMR = input(numOfCMRs);

    debugPlots = '\n Do you like to see the pre-processing plots? (y/n) \n \n ';
    debugPlotsOnOff = input(debugPlots,'s');

    errorPlots = '\n Do you like to get error calculations and plots? (y/n) \n \n ';
    errorPlotsOnOff = input(errorPlots,'s');

    fittingChoiceTxt = '\n Which curve fitting should be used:  \n [0]: no fitting \n
[1]: a1*1/x + a2*x + a3 \n [2]: 4th order polynomial \n \n ';
    fittingChoice = input(fittingChoiceTxt);
end

if exist('out.csv','file') ~= 2
    disp(' Exracting data from result file ...');
    !perl extract_data.pl Results_results.csv out.csv
    disp(' Listo!');
end

load costcmrctr-a;      %load analytically derived data

load out.csv;           %load preprocessed simulation result file; this yields a (length
(out)x4) matrix
format long;            %set number format to long, else only 5 digits
out = sortrows(out,[1,2]);      %order file to get all CMRs grouped together, 2nd order
acc. to CTR
out = [out (out(:,3) + out(:,4))];  %add LU cost and PG cost to get total cost, now
(length(out)x5) matrix

%CMR = 46;   %numero de CMRs; THIS HAS TO BE SET MANUALLY!!!
numCTR = length(out)/numCMR;  %numero de CTRs para descubrir el minimo

%check for plausibility of CMR (= CMR is factor of length(out))
if numCTR ~= round(numCTR)
    error('Your entered a CMR value that is not valid! Further execution aborted...');
end

%make new dimension to get a (CMRxCTRx5) matrix
for i = 0:numCMR-1
    data(i+1,:,:) = out(i*numCTR+1:(i+1)*numCTR,:);
end

if fittingChoice == 1
    %fit data to f(x) = a1*1/x  +  a2*x  +  a3
    for i = 1:numCMR
        t = data(i,:,2);
        y = data(i,:,5);
        X = [ones(size(t)) ; 1./t ; t];
        a = X'\y';
        T = data(i,:,2)';
        Y = [ones(size(T))  1./T  T]*a;
        data_smooth(i,:,:) = [T  Y];
    end
```

```
elseif fittingChoice == 2
    %fit data to 4th order polynomial
    for i = 1:numCMR
        t = data(i,:,2);
        y = data(i,:,5);
        p = polyfit(t,y,4);
        Y = polyval(p,t);
        data_smooth(i,:,:) = [t' Y'];
    end
elseif fittingChoice == 0
    %no fitting
    for i = 1:numCMR
        t = data(i,:,2);
        y = data(i,:,5);
        data_smooth(i,:,:) = [t' y'];
    end
else
    error('Your fitting choice was wrong! Aborting...');
end

%interpolate fitted curve to 5000 values
for i = 1:numCMR
    data_i(i,:,:) = [data_smooth(i,1,1):(data_smooth(i,numCTR,1)-data_smooth(i,1,1))/
5000:data_smooth(i,numCTR,1);interp1(data_smooth(i,:,1),data_smooth(i,:,2),data_smooth
(i,1,1):(data_smooth(i,numCTR,1)-data_smooth(i,1,1))/5000:data_smooth(i,numCTR,
1),'spline')]';
end

%find minimum of data
for i = 1:numCMR
    [ymd(i),imd(i)] = min(data(i,:,5));
end

%find minimum of curve approximation
for i = 1:numCMR
    [ymc(i),imc(i)] = min(data_smooth(i,:,2));
end

%find minimum of interpolated curve approximation
for i = 1:numCMR
    [ymi(i),imi(i)] = min(data_i(i,:,2));
end

if strcmp(debugPlotsOnOff,'y')

    %plot the raw data
    hold on     %all curves in one graph
    for i = 1:numCMR
        plot(data(i,:,2),data(i,:,5),'r');
    end
    pause       %wait for key

    %plot fitted curves
    for i = 1:numCMR
        plot(data_smooth(i,:,1),data_smooth(i,:,2),'b');
    end
    pause

    %plot minima of data
    plot(data(1,imd,2),ymd,'ro');   %data( 1 ,imd,2) can be used as the numCTR is the
same for all i

    %plot minima of curve approximation
    plot(data_smooth(1,imc,1),ymc,'bo');   %data( 1 ,imc,2) can be used as the numCTR is
the same for all i

    %plot minima of interpolated curve approximation
    plot(data_i(1,imi,1),ymi,'go');   %data( 1 ,imi,2) can be used as the numCTR is the
same for all i
```

```
    pause
    hold off

    %plot graph with CTR_opt and C_opt over numCMR
    [AX,H1,H2] = plotyy(1./data(:,1,1),ymc,1./data(:,1,1),out(imc,2));
    set(AX,'xlim',[0.01 0.1]);
    set(AX(2),'ylim',[0.05 0.25]);
    %set(H1,'LineStyle','-');
    %set(H2,'LineStyle','--');
    %set(H1,'Marker','x')
    %set(H2,'Marker','o')
    hold on
    [AX,H1,H2] = plotyy(1./data(:,1,1),ymd,1./data(:,1,1),out(imd,2));
    %axis([0.01,0.1,min(y/100),max(y/100)])
    set(AX,'xlim',[0.01 0.1]);
    set(AX(2),'ylim',[0.05 0.25]);
    %set(H1,'LineStyle','-');
    %set(H2,'LineStyle','--');
    %set(H1,'Marker','x')
    %set(H2,'Marker','o')
    set(H1,'color','green')
    set(H2,'color','green')
    pause
    %close

    %[AX,H1,H2] = plotyy(1./data(:,1,1),ymi,1./data(:,1,1),out(imi,2));
    [AX,H1,H2] = plotyy(1./data(:,1,1),ymi,1./data(:,1,1),data_i(1,imi,1)); %a bug is
here!
    %axis([0.01,0.1,min(y/100),max(y/100)])
    set(AX,'xlim',[0.01 0.1]);
    set(AX(2),'ylim',[0.05 0.25]);
    %set(H1,'LineStyle','-');
    %set(H2,'LineStyle','--');
    %set(H1,'Marker','x')
    %set(H2,'Marker','o')
    set(H1,'color','red')
    set(H2,'color','red')
    pause
    close

end %debugPlotsOnOff

%-----------------------------------------------------------------------------
CT   = CLUTa   + CLUDa   + CPGa;
CT_s = CLUTa_s + CLUDa_s + CPGa_s;

CTtc   = zeros(2,puntoscmr);
CTRtc  = zeros(2,puntoscmr);
CTst   = zeros(2,puntoscmr);
CTRst  = zeros(2,puntoscmr);
CTrt   = zeros(2,puntoscmr);
CTRrt  = zeros(2,puntoscmr);

Rind=1;
for salcelind=1:2
    for i=1:puntoscmr
        [CTtc(salcelind,i),I]=min(CT(Rind,salcelind,1:puntos,i));
        CTRtc(salcelind,i)=CTR(I);
    end
end

Rind=2;
for salcelind=1:2
    for i=1:puntoscmr
        [CTrt(salcelind,i),I]=min(CT(Rind,salcelind,1:puntos,i));
        CTRrt(salcelind,i)=CTR(I);
        [CTst(salcelind,i),I]=min(CT_s(Rind,salcelind,1:puntos,i));
        CTRst(salcelind,i)=CTR(I);
```

```
    end
end

close all;
figure(1);

ticados   = 10;
espaciado = puntoscmr/ticados;

switch chosenCurve
    case 1
        line(CMR(1:espaciado:puntoscmr),CTRtc
(1,1:espaciado:puntoscmr),'LineStyle','none','Marker','x','Color','k');
        lines(1) = line(CMR(1:puntoscmr),CTRtc
(1,1:puntoscmr),'LineStyle',':','Color','k');
    case 2
        line(CMR(1:espaciado:puntoscmr),CTRtc
(2,1:espaciado:puntoscmr),'LineStyle','none','Marker','+','Color','k');
        lines(1) = line(CMR(1:puntoscmr),CTRtc
(2,1:puntoscmr),'LineStyle',':','Color','k');
    case 3
        line(CMR(1:espaciado:puntoscmr),CTRst
(1,1:espaciado:puntoscmr),'LineStyle','none','Marker','s','Color','k');
        lines(1) = line(CMR(1:puntoscmr),CTRst
(1,1:puntoscmr),'LineStyle',':','Color','k');
    case 4
        line(CMR(1:espaciado:puntoscmr),CTRst
(2,1:espaciado:puntoscmr),'LineStyle','none','Marker','d','Color','k');
        lines(1) = line(CMR(1:puntoscmr),CTRst
(2,1:puntoscmr),'LineStyle',':','Color','k');
    case 5
        line(CMR(1:espaciado:puntoscmr),CTRrt
(1,1:espaciado:puntoscmr),'LineStyle','none','Marker','<','Color','k');
        lines(1) = line(CMR(1:puntoscmr),CTRrt
(1,1:puntoscmr),'LineStyle',':','Color','k');
    case 6
        line(CMR(1:espaciado:puntoscmr),CTRrt
(2,1:espaciado:puntoscmr),'LineStyle','none','Marker','>','Color','k');
        lines(1) = line(CMR(1:puntoscmr),CTRrt
(2,1:puntoscmr),'LineStyle',':','Color','k');
    otherwise
        close all;
        error('The selection of the curve to compare was not valid. Aborting
execution ...');
end

lines(length(lines)+1) = line(1./data(:,1,1),data_i(1,imi,1),'Color','k');   %draws the
simulated CTR* for comparison
%line(1./data(:,1,1),out(imd,2),'Color','r');
%hl=legend(': Optimum {\it CTR}, {\it CTR}^*',': Optimum total cost, {\it C_T}^*',' ',':
Original scheme',': Enhanced scheme, [3]',': S&W scheme, {\it H}=0',': {\it Modified}
S&W scheme, {\it H}=0',': R&W scheme, {\it H}=0',': {\it Modified} R&W scheme, {\it H}
=0');

ax1      = gca;
axis(ax1,[0.01 0.1 0.001 0.25]);
set(ax1,'YColor','k');
grid on;
ax2      = axes('YaxisLocation','right','Ycolor','b','Color','none');
axis(ax2,[0.01 0.1 7 32]);

switch chosenCurve
    case 1
        line(CMR(1:espaciado:puntoscmr),CTtc
(1,1:espaciado:puntoscmr),'LineStyle','none','Marker','x','Color','b');
        lines(length(lines)+1) = line(CMR(1:puntoscmr),CTtc
(1,1:puntoscmr),'LineStyle',':','Color','b');
    case 2
        line(CMR(1:espaciado:puntoscmr),CTtc
```

```
(2,1:espaciado:puntoscmr),'LineStyle','none','Marker','+','Color','b');
        lines(length(lines)+1) = line(CMR(1:puntoscmr),CTtc
(2,1:puntoscmr),'LineStyle',':','Color','b');
    case 3
        line(CMR(1:espaciado:puntoscmr),CTst
(1,1:espaciado:puntoscmr),'LineStyle','none','Marker','s','Color','b');
        lines(length(lines)+1) = line(CMR(1:puntoscmr),CTst
(1,1:puntoscmr),'LineStyle',':','Color','b');
    case 4
        line(CMR(1:espaciado:puntoscmr),CTst
(2,1:espaciado:puntoscmr),'LineStyle','none','Marker','d','Color','b');
        lines(length(lines)+1) = line(CMR(1:puntoscmr),CTst
(2,1:puntoscmr),'LineStyle',':','Color','b');
    case 5
        line(CMR(1:espaciado:puntoscmr),CTrt
(1,1:espaciado:puntoscmr),'LineStyle','none','Marker','<','Color','b');
        lines(length(lines)+1) = line(CMR(1:puntoscmr),CTrt
(1,1:puntoscmr),'LineStyle',':','Color','b');
    case 6
        line(CMR(1:espaciado:puntoscmr),CTrt
(2,1:espaciado:puntoscmr),'LineStyle','none','Marker','>','Color','b');
        lines(length(lines)+1) = line(CMR(1:puntoscmr),CTrt
(2,1:puntoscmr),'LineStyle',':','Color','b');
end

lines(length(lines)+1) = line(1./data(:,1,1),ymi,'Color','b');   %draws the simulated
CT* for comparison

set(get(ax1,'Ylabel'),'String','Optimum {\it CTR}, {\it CTR}^*');
set(get(ax2,'Ylabel'),'String','Optimum total cost, {\it C_T}^*');
hy1=get(ax1,'Ylabel');
hx=xlabel('{\it CMR}');
ticsejes = 6;
ylimits  = get(ax2,'YLim');
yinc    = (ylimits(2)-ylimits(1))/(ticsejes-1);
set(ax2,'YTick',[ylimits(1):yinc:ylimits(2)]);

hy1=get(ax1,'Ylabel');
hy2=get(ax2,'Ylabel');

set(ax1,'FontName','Palatino');
set(ax2,'FontName','Palatino');
set(hy1,'FontName','Palatino');
set(hy2,'FontName','Palatino');
set(hx,'FontName','Palatino');

%legend(lines,'Test','test222')
legend(lines,'{\it CTR}^* (analytical)','{\it CTR}^* (simulated)','{\it C_T}^*
(analytical)','{\it C_T}^* (simulated)');
%-----------------------------------------------------------------------

pause;
close all;

%----- post-processing -----
if strcmp(errorPlotsOnOff,'y')
    %error statistics and plots
    switch chosenCurve
        case 1
            y_calc =  interp1(CMR(1:puntoscmr),CTtc(1,1:puntoscmr),1./data(:,1,1));
        case 2
            y_calc =  interp1(CMR(1:puntoscmr),CTtc(2,1:puntoscmr),1./data(:,1,1));
        case 3
            y_calc =  interp1(CMR(1:puntoscmr),CTst(1,1:puntoscmr),1./data(:,1,1));
        case 4
            y_calc =  interp1(CMR(1:puntoscmr),CTst(2,1:puntoscmr),1./data(:,1,1));
        case 5
            y_calc =  interp1(CMR(1:puntoscmr),CTrt(1,1:puntoscmr),1./data(:,1,1));
        case 6
```

```
            y_calc =  interp1(CMR(1:puntoscmr),CTrt(2,1:puntoscmr),1./data(:,1,1));
    end
    erCT = abs(y_calc'-ymc)./y_calc';
    plot(1./data(:,1,1),erCT);
    line([1./data(1,1,1) 1./data(end,1,1)],[mean(erCT) mean
(erCT)],'LineStyle','-','Color','k');
    legend('|\Delta {C_T}^*|','Mean |\Delta {C_T}^*|');
    grid on;
    %axis([0.01 0.1 0 0.025]);
    disp('Mean Delta C_T* = ');
    disp(mean(erCT));
    pause
    close

    switch chosenCurve
        case 1
            y_calc =  interp1(CMR(1:puntoscmr),CTRtc(1,1:puntoscmr),1./data(:,1,1));
        case 2
            y_calc =  interp1(CMR(1:puntoscmr),CTRtc(2,1:puntoscmr),1./data(:,1,1));
        case 3
            y_calc =  interp1(CMR(1:puntoscmr),CTRst(1,1:puntoscmr),1./data(:,1,1));
        case 4
            y_calc =  interp1(CMR(1:puntoscmr),CTRst(2,1:puntoscmr),1./data(:,1,1));
        case 5
            y_calc =  interp1(CMR(1:puntoscmr),CTRrt(1,1:puntoscmr),1./data(:,1,1));
        case 6
            y_calc =  interp1(CMR(1:puntoscmr),CTRrt(2,1:puntoscmr),1./data(:,1,1));
    end
    erCTR = abs(y_calc'-data_smooth(1,imc,1))./y_calc';
    plot(1./data(:,1,1),erCTR);
    line([1./data(1,1,1) 1./data(end,1,1)],[mean(erCTR) mean
(erCTR)],'LineStyle','-','Color','k');
    legend('|\Delta {CTR}^*|','Mean |\Delta {CTR}^*|');
    grid on;
    %axis([0.01 0.1 0 0.025]);
    disp('Mean Delta CTR* = ');
    disp(mean(erCTR));
    pause
    close
end
```

C.2 compare.m

To compare the LU strategy for deterministic and Erlang-distributed timeouts a separate Matlab-script was used. It is a simplified `data_analyse.m`-script, enhanced by error bars.

```
clear;  %to remove all interfering variables from workspace

if exist('out_e.csv','file') ~= 2
    !perl extract_data_comparison.pl Results_results_erlang.csv out_e.csv
end
if exist('out_d.csv','file') ~= 2
    !perl extract_data_comparison.pl Results_results_det.csv out_d.csv
end

load out_e.csv;            %load preprocessed simulation result file; this yields a
(length(out)x4) matrix
load out_d.csv;               %load preprocessed simulation result file; this yields a
(length(out)x4) matrix
format long;                    %set number format to long, else only 5 digits
out_e = sortrows(out_e,[1,2]);      %order file to get all CMRs grouped together, 2nd
order acc. to CTR
out_e = [out_e (out_e(:,3) + out_e(:,5)) (out_d(:,4) + out_d(:,6))];  %add LU cost and
PG cost to get total cost, now (length(out)x5) matrix

out_d = sortrows(out_d,[1,2]);      %order file to get all CMRs grouped together, 2nd
order acc. to CTR
out_d = [out_d (out_d(:,3) + out_d(:,5)) (out_d(:,4) + out_d(:,6))];  %add LU cost and
PG cost to get total cost, now (length(out)x5) matrix

numCMR = 37;
numCTR = length(out_e)/numCMR;  %numero de CTRs para descubrir el minimo

%make new dimension to get a (CMRxCTRx5) matrix
for i = 0:numCMR-1
    data_e(i+1,:,:) = out_e(i*numCTR+1:(i+1)*numCTR,:);
end
%make new dimension to get a (CMRxCTRx5) matrix
for i = 0:numCMR-1
    data_d(i+1,:,:) = out_d(i*numCTR+1:(i+1)*numCTR,:);
end

%find minimum of data
for i = 1:numCMR
    [ymd(i),imd(i)] = min(data_d(i,:,7));
end
%find minimum of data
for i = 1:numCMR
    [yme(i),ime(i)] = min(data_e(i,:,7));
end

hold on;
for i = 1:numCMR
    plot(data_e(i,:,2),data_e(i,:,7),'r');
end
plot(data_e(1,ime,2),yme,'bo');
pause;
close;

plot(1./data_d(:,1,1),ymd,'--');
hold on;
plot(1./data_e(:,1,1),yme,'r');
hold off;
legend('Deterministic','Erlang-90');
xlabel('CMR');
ylabel('Total costs');
XLim([0.01 0.1]);
grid on;
%print -depsc2 'DetErl.eps'
%print -dpdf 'DetErl.pdf'
pause;
close;

for i = 1:37
    errd(i) = sqrt(data_d(i,imd(i),4)^2+data_d(i,imd(i),6)^2);
```

```
    erre(i) = sqrt(data_e(i,ime(i),4)^2+data_e(i,ime(i),6)^2);
end

errorbar(1./data_d(:,1,1),ymd,errd,'--');
hold on;
errorbar(1./data_e(:,1,1),yme,erre,'r');
grid on;
legend('Deterministic','Erlang-90');
xlabel('CMR');
ylabel('Total costs (with confidence intervals)');
XLim([0.01 0.1]);
print -depsc2 'DetErl.eps'
%print -dpdf 'DetErl.pdf'
pause;
close;

plot(1./data_d(:,1,1),abs(ymd-yme)./ymd);
xlabel('CMR');
ylabel('Deviation');
XLim([0.01 0.1]);
grid on;
print -depsc2 'DetErlError.eps'
%print -dpdf 'DetErlError.pdf'

mean(abs(ymd-yme)/ymd)

pause;
close;
```

D Möbius Model Reference and Details

D.1 Study 1

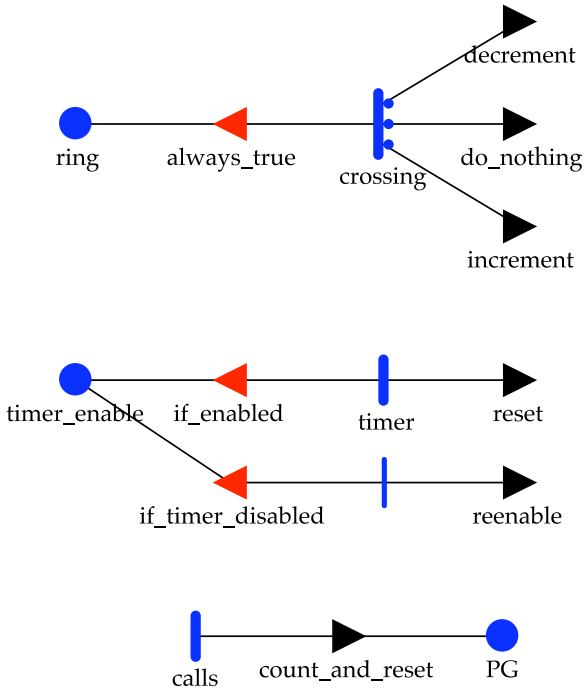


Figure D.1: SAN modelled in Möbius for time-based MM

Bucket Attributes:	
Place Names	**Initial Markings**
PG	0
ring	0
timer_enable	1

Timed Activity:	**calls**
Exponential Distribution Parameters	**Rate** `r_call`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	**crossing**
Exponential Distribution Parameters	**Rate** `r_cell`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return((2.0*ring->Mark()-1)/(6.0*ring->Mark()));`
	
	case 2
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return(1.0/3.0);`
	
	case 3
	
	`if (ring->Mark() == 0)`
	` return(1.0);`
	`else`
	` return((2.0*ring->Mark()+1)/(6.0*ring->Mark()));`

Timed Activity:	**timer**
Deterministic Distribution Parameters	**Value** `timeout`
Activation Predicate	(none)
Reactivation Predicate	(none)

Instantaneous Activities Without Cases:
at_once

Input Gate:	always_true
Predicate	`true`
Function	`;`

Input Gate:	if_enabled
Predicate	`(timer_enable->Mark() >= 1) && (ring->Mark() > local_area)`
Function	`;`

Input Gate:	if_timer_disabled
Predicate	`timer_enable->Mark() == 0`
Function	`;`

Output Gate:	count_and_reset

Function	`short m = ring->Mark();`
	`PG->Mark() += 3*m*(m + 1) + 1;`
	`ring->Mark() = 0;`
	`timer_enable->Mark() = 0;`

Output Gate:	decrement
Function	`ring->Mark()--;`

Output Gate:	do_nothing
Function	`//does nothing`

Output Gate:	increment
Function	`ring->Mark()++;`

Output Gate:	reenable
Function	`timer_enable->Mark() = 1;`

Output Gate:	reset
Function	`ring->Mark() = 0;`

Performance Variable Model: LUaPG		
Top Level Model Information	Child Model Name	time_based_mobility
	Model Type	SAN Model

Performance Variable : LU			
Affecting Models	time_based_mobility		
Impulse Functions	time_based_mobility->timer		
	(Reward is over all Available Models) `return(1.0);`		
Reward Function	*(Reward is over all Available Models)*		
Simulator Statistics	Type	Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.01

Performance Variable : PG			
Affecting Models	time_based_mobility		
Impulse Functions			
Reward Function	*(Reward is over all Available Models)* `return(time_based_mobility->PG->Mark());`		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.01

D.2 Study 4

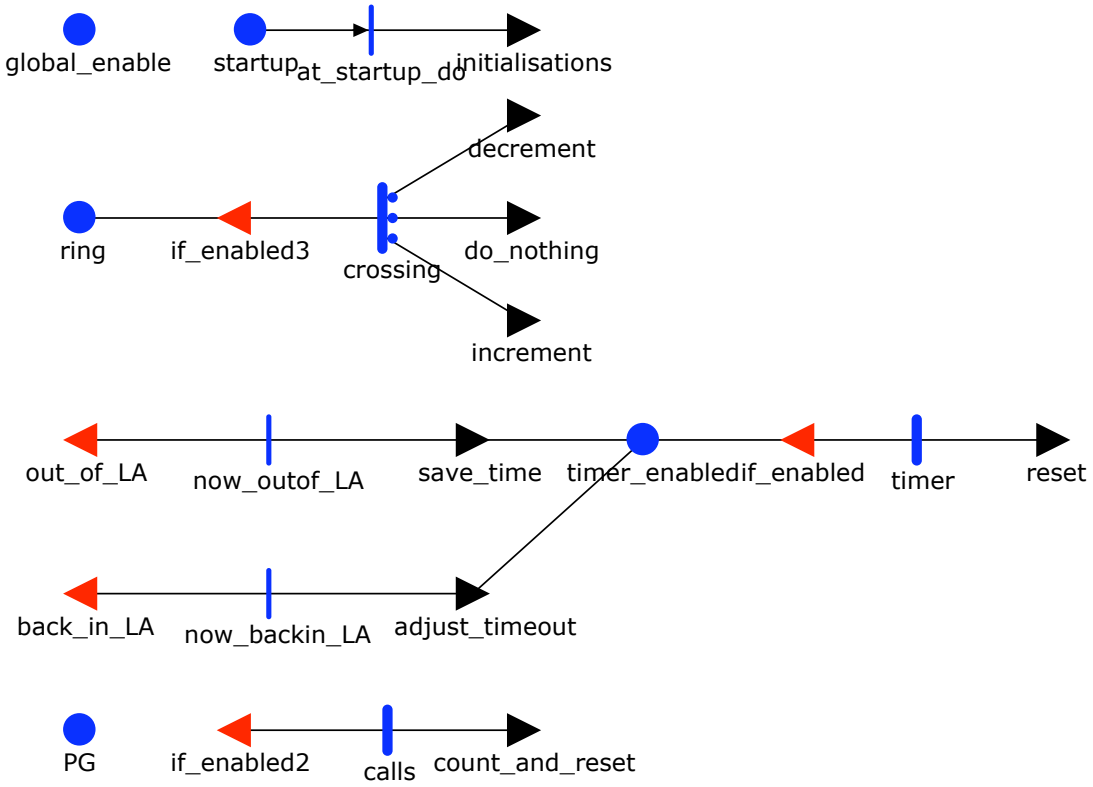


Figure D.2: SAN modelled in Möbius for time-based S&W MM

Bucket Attributes:	
Place Names	**Initial Markings**
PG	0
global_enable	1
ring	0
startup	1
timer_enabled	0

Timed Activity:	**calls**
Exponential Distribution Parameters	**Rate** `r_call`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	**crossing**
Exponential Distribution Parameters	**Rate** `r_cell`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return((2.0*ring->Mark()-1)/(6.0*ring->Mark()));`
	
	case 2
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return(1.0/3.0);`
	
	case 3
	
	`if (ring->Mark() == 0)`
	` return(1.0);`
	`else`
	` return((2.0*ring->Mark()+1)/(6.0*ring->Mark()));`

Timed Activity:	**timer**
Deterministic Distribution Parameters	**Value** `timeout`
Activation Predicate	(none)
Reactivation Predicate	(none)

Instantaneous Activities Without Cases:
at_startup_do
now_backin_LA
now_outof_LA

Input Gate:	back_in_LA
Predicate	`(ring->Mark() <= local_area) &&` `(timer_enabled->Mark() == 1)`
Function	`;`

Input Gate:	if_enabled
Predicate	`(global_enable->Mark() >= 1) && (timer_enabled->Mark() == 1)`
Function	`;`

Input Gate:	if_enabled2
Predicate	`global_enable->Mark() >= 1`
Function	`;`

Input Gate:	if_enabled3
Predicate	`global_enable->Mark() >= 1`
Function	`;`

Input Gate:	out_of_LA
Predicate	`(ring->Mark() > local_area) && (timer_enabled->Mark() == 0)`
Function	`;`

Output Gate:	adjust_timeout

Function	`timeout = timeout -`
	` (BaseModelClass::LastActionTime - saved_time);`
	`//inside_LA = true;`
	`timer_enabled->Mark() = 0;`

Output Gate:	count_and_reset

Function	`short m = ring->Mark();`
	`PG->Mark() += 3*m*(m + 1) + 1;`
	`global_enable->Mark() = 0;`
	`timeout = original_timeout;`
	`timer_enabled->Mark() = 0;`

Output Gate:	decrement
Function	`ring->Mark()--;`

Output Gate:	do_nothing
Function	`//do not change the ring location`

Output Gate:	increment
Function	`ring->Mark()++;`

Output Gate:	initialisations
Function	`original_timeout = timeout;`

Output Gate:	reset

Function	`ring->Mark() = 0;`
	`timeout = original_timeout;`
	`timer_enabled->Mark() = 0;`

Output Gate:	save_time
Function	`saved_time = BaseModelClass::LastActionTime;` `timer_enabled->Mark() = 1;`

Performance Variable Model: LUaPG		
Top Level Model Information	Child Model Name	time_based_mobility
	Model Type	SAN Model

Performance Variable : LU			
Affecting Models	time_based_mobility		
Impulse Functions	time_based_mobility->timer		
	(Reward is over all Available Models) `return(1.0);`		
Reward Function	*(Reward is over all Available Models)*		
Simulator Statistics	Type	Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.005

Performance Variable : PG			
Affecting Models	time_based_mobility		
Impulse Functions			
Reward Function	*(Reward is over all Available Models)* `return(time_based_mobility->PG->Mark());`		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.005

Range Study Variable Assignments for Study range in Project tiempo_stop_sb:							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
end_time	double	Fixed	100.0	-	-	-	-
local_area	short	Fixed	0	-	-	-	-
original_timeout	double	Fixed	0.0	-	-	-	-
r_call	double	Fixed	1.0	-	-	-	-
r_cell	double	Functional	[1.0,10.0]	0.2	Additive	n/x	100.0
saved_time	double	Fixed	0.0	-	-	-	-
timeout	double	Incremental	[Incremental Range]	0.0020	Additive	-	-

D.3 Study 5

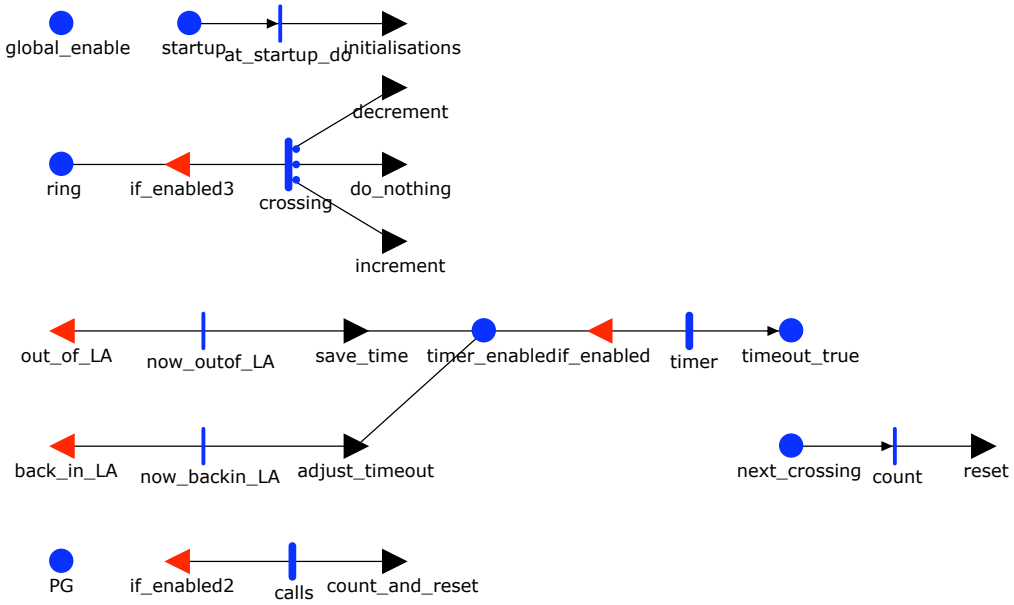


Figure D.3: SAN modelled in Möbius for time-based modified S&W MM

Bucket Attributes:	
Place Names	**Initial Markings**
PG	0
global_enable	1
next_crossing	0
ring	0
startup	1
timerout_true	0
timer_enabled	0

Timed Activity:	**calls**
Exponential Distribution Parameters	**Rate** `r_call`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	**crossing**
Exponential Distribution Parameters	**Rate** `r_cell`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return((2.0*ring->Mark()-1)/(6.0*ring->Mark()));`
	
	case 2
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return(1.0/3.0);`
	
	case 3
	
	`if (ring->Mark() == 0)`
	` return(1.0);`
	`else`
	` return((2.0*ring->Mark()+1)/(6.0*ring->Mark()));`

Timed Activity:	**timer**
Deterministic Distribution Parameters	**Value** `timeout`
Activation Predicate	(none)
Reactivation Predicate	(none)

Instantaneous Activities Without Cases:
at_startup_do
count
now_backin_LA
now_outof_LA

Input Gate:	back_in_LA
Predicate	`(ring->Mark() <= local_area) &&` `(timer_enabled->Mark() == 1)`
Function	`;`

Input Gate:	if_enabled

Predicate	`(global_enable->Mark() >= 1) &&`
	`(timer_enabled->Mark() == 1) &&`
	`(timeout_true->Mark() == 0)`

Function	`;`

Input Gate:	if_enabled2
Predicate	`global_enable->Mark() >= 1`
Function	`;`

Input Gate:	if_enabled3
Predicate	`global_enable->Mark() >= 1`
Function	`;`

Input Gate:	out_of_LA
Predicate	`(ring->Mark() > local_area) && (timer_enabled->Mark() == 0)`
Function	`;`

Output Gate:	adjust_timeout

Function	`timeout = timeout -`
	` (BaseModelClass::LastActionTime - saved_time);`
	`timer_enabled->Mark() = 0;`

Output Gate:	count_and_reset

Function	`short m = ring->Mark();`
	`PG->Mark() += 3*m*(m + 1) + 1;`
	`global_enable->Mark() = 0;`
	`timeout = original_timeout;`
	`timer_enabled->Mark() = 0;`

Output Gate:	decrement

Function	`ring->Mark()--;`
	`if ((timeout_true->Mark() == 1) && (ring->Mark() > local_area))`
	` next_crossing->Mark() = 1;`

Output Gate:	do_nothing

Function	`//do not change the ring location`
	`if ((timeout_true->Mark() == 1) && (ring->Mark() > local_area))`
	` next_crossing->Mark() = 1;`

Output Gate:	increment

Function	`ring->Mark()++;`
	`if ((timeout_true->Mark() == 1) && (ring->Mark() > local_area))`
	` next_crossing->Mark() = 1;`

Output Gate:	initialisations
Function	`original_timeout = timeout;`

Output Gate:	reset

Function	`ring->Mark() = 0;`
	`timeout = original_timeout;`
	`timer_enabled->Mark() = 0;`
	`timeout_true->Mark() = 0;`

Output Gate:	save_time
Function	`saved_time = BaseModelClass::LastActionTime;` `timer_enabled->Mark() = 1;`

Performance Variable Model: LUaPG		
Top Level Model Information	Child Model Name	time_based_mobility
	Model Type	SAN Model

Performance Variable : LU			
Affecting Models	time_based_mobility		
Impulse Functions	time_based_mobility->count		
	(Reward is over all Available Models) `return(1.0);`		
Reward Function	*(Reward is over all Available Models)*		
Simulator Statistics	Type	Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.01

Performance Variable : PG			
Affecting Models	time_based_mobility		
Impulse Functions			
Reward Function	*(Reward is over all Available Models)* `return(time_based_mobility->PG->Mark());`		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.01

Range Study Variable Assignments for Study range in Project tiempo_stop_mod_stop:							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
end_time	double	Fixed	100.0	-	-	-	-
local_area	short	Fixed	0	-	-	-	-
original_timeout	double	Fixed	0.0	-	-	-	-
r_call	double	Fixed	1.0	-	-	-	-
r_cell	double	Functional	[1.0,10.0]	0.5	Additive	n/x	100.0
saved_time	double	Fixed	0.0	-	-	-	-
timeout	double	Incremental	[Incremental Range]	0.0010	Additive	-	-

D.4 Study 6

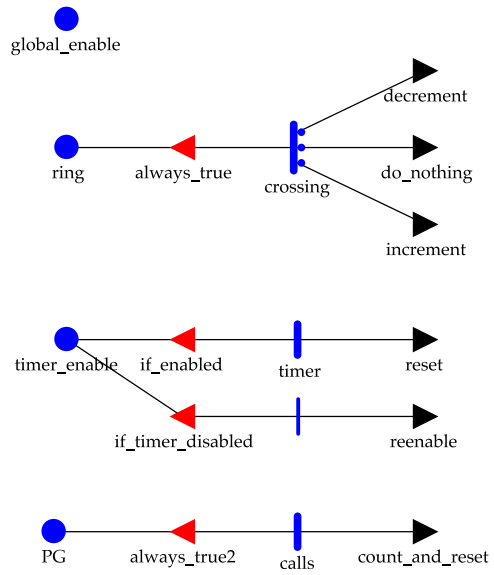


Figure D.4: SAN modelled in Möbius for time-based R&W MM

Bucket Attributes:	
Place Names	**Initial Markings**
PG	0
global_enable	1
ring	0
timer_enable	1

Timed Activity:	**calls**
Exponential Distribution Parameters	**Rate** `r_call`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	**crossing**
Exponential Distribution Parameters	**Rate** `r_cell`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return((2.0*ring->Mark()-1)/(6.0*ring->Mark()));`
	
	case 2
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return(1.0/3.0);`
	
	case 3
	
	`if (ring->Mark() == 0)`
	` return(1.0);`
	`else`
	` return((2.0*ring->Mark()+1)/(6.0*ring->Mark()));`

Timed Activity:	**timer**
Deterministic Distribution Parameters	**Value** `timeout`
Activation Predicate	(none)
Reactivation Predicate	(none)

Instantaneous Activities Without Cases:
at_once

Input Gate:	always_true
Predicate	`(global_enable->Mark() == 1)`
Function	`;`

Input Gate:	always_true2
Predicate	`(global_enable->Mark() == 1)`
Function	`;`

Input Gate:	if_enabled

Predicate	`(timer_enable->Mark() >= 1)`
	`&& (ring->Mark() > local_area)`
	`&& (global_enable->Mark() == 1)`

Function	`;`

Input Gate:	if_timer_disabled
Predicate	`timer_enable->Mark() == 0`
Function	`;`

Output Gate:	count_and_reset

Function	`short m = ring->Mark();`
	`PG->Mark() += 3*m*(m + 1) + 1;`
	`ring->Mark() = 0;`
	`timer_enable->Mark() = 0;`
	`global_enable->Mark() = 0;`

Output Gate:	decrement
Function	`ring->Mark()--;`

Output Gate:	do_nothing
Function	`//does nothing`

Output Gate:	increment
Function	`ring->Mark()++;`

Output Gate:	reenable
Function	`timer_enable->Mark() = 1;`

Output Gate:	reset
Function	`ring->Mark() = 0;`

Performance Variable Model: LUaPG		
Top Level Model Information	Child Model Name	time_based_mobility
	Model Type	SAN Model

Performance Variable : LU			
Affecting Models	time_based_mobility		
Impulse Functions	time_based_mobility->timer		
	(Reward is over all Available Models) `return(1.0);`		
Reward Function	*(Reward is over all Available Models)*		
Simulator Statistics	Type	Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.005

Performance Variable : PG			
Affecting Models	time_based_mobility		
Impulse Functions			
Reward Function	*(Reward is over all Available Models)* `return(time_based_mobility->PG->Mark());`		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.005

Range Study Variable Assignments for Study range in Project tiempo_modified_sb:

Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
end_time	double	Fixed	100.0	-	-	-	-
local_area	short	Fixed	0	-	-	-	-
r_call	double	Fixed	1.0	-	-	-	-
r_cell	double	Functional	[1.0,10.0]	0.2	Additive	n/x	100.0
timeout	double	Incremental	[Incremental Range]	2.5E-4	Additive	-	-

D.5 Study 3 + 7

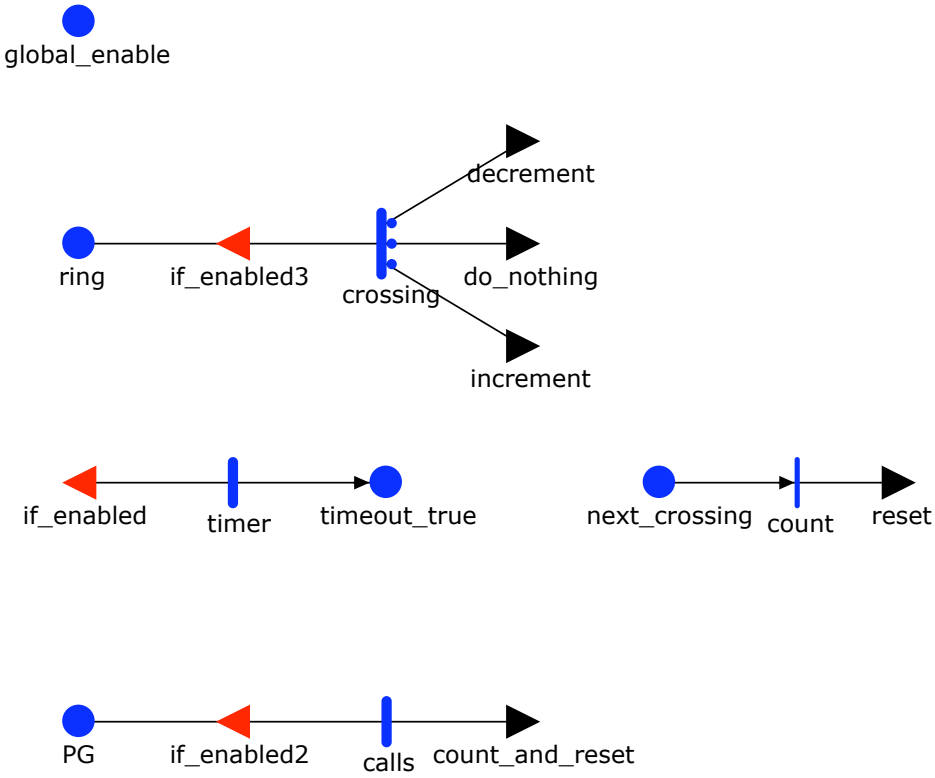


Figure D.5: SAN modelled in Möbius for time-based modified R&W MM

Bucket Attributes:	
Place Names	**Initial Markings**
PG	0
global_enable	1
next_crossing	0
ring	0
timerout_true	0

Timed Activity:	**calls**
Exponential Distribution Parameters	**Rate** `r_call`
Activation Predicate	(none)
Reactivation Predicate	(none)

Timed Activity:	**crossing**
Exponential Distribution Parameters	**Rate** `r_cell`
Activation Predicate	(none)
Reactivation Predicate	(none)

Case Distributions	**case 1**
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return((2.0*ring->Mark()-1)/(6.0*ring->Mark()));`
	
	case 2
	
	`if (ring->Mark() == 0)`
	` return(0.0);`
	`else`
	` return(1.0/3.0);`
	
	case 3
	
	`if (ring->Mark() == 0)`
	` return(1.0);`
	`else`
	` return((2.0*ring->Mark()+1)/(6.0*ring->Mark()));`

Timed Activity:	**timer**
Deterministic Distribution Parameters	**Value** `timeout`
Activation Predicate	(none)
Reactivation Predicate	(none)

Instantaneous Activities Without Cases:
count

Input Gate:	if_enabled

Predicate	`(global_enable->Mark() >= 1) &&`
	`(ring->Mark() > local_area) &&`
	`(timeout_true->Mark() == 0)`

Function	`;`

Input Gate:	if_enabled2
Predicate	`global_enable->Mark() >= 1`
Function	`;`

Input Gate:	if_enabled3
Predicate	`global_enable->Mark() >= 1`
Function	`;`

Output Gate:	count_and_reset

Function	`short m = ring->Mark();`
	`PG->Mark() += 3*m*(m + 1) + 1;`
	`global_enable->Mark() = 0;`

Output Gate:	decrement

Function	`ring->Mark()--;`
	`if ((timeout_true->Mark() >= 1) && (ring->Mark() > local_area))`
	`{`
	` /* timeout_true might be > 1 if several`
	` timeouts occur without the MT moving */`
	` next_crossing->Mark() = 1;`
	`}`
	`/* reset timeout_true in any case: either LU is done or timeout`
	` should not have happened and timer is reset */`
	`timeout_true->Mark() = 0;`

Output Gate:	do_nothing

Function	`//do not change the ring location`
	`if (timeout_true->Mark() >= 1){`
	` /* timeout_true might be > 1 if several`
	` timeouts occur without the MT moving */`
	` next_crossing->Mark() = 1;`
	` timeout_true->Mark() = 0;`
	`}`

Output Gate:	increment

Function	`ring->Mark()++;`
	`if (timeout_true->Mark() >= 1){`
	` /* timeout_true might be > 1 if several`
	` timeouts occur without the MT moving */`
	` next_crossing->Mark() = 1;`
	` timeout_true->Mark() = 0;`
	`}`

Output Gate:	reset
Function	`ring->Mark() = 0;`

Output Gate:	save_time
Function	`saved_time = BaseModelClass::LastActionTime;` `timer_enabled->Mark() = 1;`

Performance Variable Model: LUaPG		
Top Level Model Information	Child Model Name	time_based_mobility
	Model Type	SAN Model

Performance Variable : LU			
Affecting Models	time_based_mobility		
Impulse Functions	time_based_mobility->count		
	(Reward is over all Available Models) `return(1.0);`		
Reward Function	*(Reward is over all Available Models)*		
Simulator Statistics	Type	Interval of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	0.0
		Stop Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.005

Performance Variable : PG			
Affecting Models	time_based_mobility		
Impulse Functions			
Reward Function	*(Reward is over all Available Models)* `return(time_based_mobility->PG->Mark());`		
Simulator Statistics	Type	Instant of Time	
	Options	Estimate Mean	
		Include Lower Bound on Interval Estimate	
		Include Upper Bound on Interval Estimate	
		Estimate out of Range Probabilities	
		Confidence Level is Relative	
	Parameters	Start Time	end_time
	Confidence	Confidence Level	0.95
		Confidence Interval	0.005

Range Study Variable Assignments for Study range in Project tiempo_modified_sb:							
Variable	Type	Range Type	Range	Increment	Increment Type	Function	n
end_time	double	Fixed	100.0	-	-	-	-
local_area	short	Fixed	0	-	-	-	-
r_call	double	Fixed	1.0	-	-	-	-
r_cell	double	Functional	[1.0,10.0]	0.25	Additive	n/x	100.0
timeout	double	Incremental	[Incremental Range]	5.0E-4	Additive	-	-

E Supplementary CD-ROM

To preserve the work and make the results reconstructible a CD-ROM was compiled and deposited. Its content is summarised here:

- The thesis itself as Adobe® PDF file in PDF Version 1.4
- All LaTeX sourcecode files of the thesis with all figures in PDF format as well as in the original, editable format
- The BibTeX file with the bibliography of all cited and not-cited literature entries
- All literature files, like conference papers and specifications, if available in electronic format
- The Möbius SAN models discussed in this work, i.e. Study 1 - Study 10
- Matlab and Perl scripts

The directory structure of the CD-ROM is given in the following:

```
- LaTeX source
    * chapters
    * figs
- Literature
    - master_thesis2.bib
    * pdfs
- Matlab and Perl
- Studies
    * Study 1
    * ...
    * Study 10
- thesis.pdf
```

Bibliography

[1] Andrei Nikolajewitsch Kolmogorow, Feb 2006. URL http://de.wikipedia.org/wiki/Andrei_Nikolajewitsch_Kolmogorow.

[2] I. F. Akyildiz, J. S. M. Ho, and Yi-Bing Lin. Movement-based location update and selective paging for PCS networks. *Networking, IEEE/ACM Transactions on*, 4(4):629–638, 1996. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=11235&arnumber=532871.

[3] Ian F. Akyildiz, Janis McNair, Joseph Ho, Hüseyin Uzunalioglu, and Weyne Wang. Mobility Management in Current and Future Communication Networks. *IEEE Network Magazine*, 12(8):39–49, July/August 1998.

[4] Suzanne Alejandre. What Is a Tessellation?, 2006. URL http://mathforum.org/sum95/suzanne/whattess.html.

[5] Vicente Casares-Giner and Pablo García-Escalle. An Hybrid Movement-Distance-Based Location Update Strategy for Mobility Tracking. *Proc. 5th European Wireless Conference, Mobile and Wireless Systems beyond 3G*, pages 121–127, Feb 2004.

[6] Jörg Eberspächer, Hans-Jörg Vögel, and Christian Bettstetter. *GSM - Switching, Services and Protocols*. John Wiley & Sons, 2. edition, March 2001. URL http://www.wiley.co.uk/wileychi/commstech/eberspacher1.html.

[7] *3GPP TS 29.002; Mobile application part (MAP) specification*. ETSI, 4.8.0 edition, 1994.

[8] George S. Fishman. *Principle of Discrete Event Simulation*. John Wiley & Sons, 1978.

[9] Pablo García-Escalle and Vicente Casares-Giner. Performance Evaluation of Stop & Wait and Reset & Wait Time-Based Location Update Strategies. *IEICE Transactions on Communications*, E88-B(7): 2971–2984, July 2005. URL http://ietcom.oxfordjournals.org/cgi/reprint/E88-B/7/2971.

[10] Geoffrey Gordon. *System Simulation*. Prentice-Hall, 1978.

[11] PERFORM Performability Engineering Research Group. *Möbius - Model-Based Environment for Validation of System Reliability, Availability, Security, and Performance - User Manual*. University of Illinois at Urbana-Champaign, 451 Coordinated Science Laboratory, MC-228, 1308 West Main Street, Urbana, IL 61801-2307, USA, 1.8.0 edition, 2005. URL http://www.mobius.uiuc.edu/.

[12] John Kingman. Kolmogorov and Contemporary Mathematics, 2003. URL http://www.emis.de/newsletter/newsletter49.pdf.

[13] Leonhard Kleinrock. *Queueing Systems*, volume I: Theory. John Wiley & Sons, 1975. URL http://www.lk.cs.ucla.edu/.

[14] Hisashi Kobayashi. *Modeling and Analysis*. The Systems Programming Series. Addison-Wesley, 1978.

[15] Paul J. Kühn. Teletraffic Theory and Engineering. Pfaffenwaldring 47, 70569 Stuttgart, Germany, Winter term 2001.

[16] Paul J. Kühn. Lecture "Teletraffic Theory and Engineering". Blackboard notes, Winter term 2004.

[17] Averill M. Law and W. David Kelton. *Simulation, Modeling and Analysis*. McGraw-Hill, 3. edition, 2000. URL `http://www.mhhe.com/engcs/industrial/lawkelton/`.

[18] Goo Yeon Lee and Yong Lee. Performance analysis of an modified time-based location registration scheme. pages 492–497, 2003. URL `http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=1243175&isnumber=27859&punumber=8808&k2dockey=1243175@ieeecnfs&query=1243175%3Cin%3Earnumber&pos=0`.

[19] Michel Mouly and Marie-Bernadette Pautet. *The GSM System for Mobile Communications*. Cell & Sys, 1. edition, 1992.

[20] Pars Mutaf and Claude Castelluccia. Hash-Based Paging and Location Update Using Bloom Filters. *Mobile Networks and Applications*, 9(6):627–631, December 2004. URL `http://www.inrialpes.fr/planete/people/mutaf/monet05dec.pdf`.

[21] J. J. O'Connor and E. F. Robertson. Sydney Chapman, October 2003. URL `http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Chapman.html`.

[22] J. J. O'Connor and E. F. Robertson. Andrey Nikolaevich Kolmogorov, January 1999. URL `http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Kolmogorov.html`.

[23] J. J. O'Connor and E. F. Robertson. Andrei Andreyevich Markov, December 1996. URL `http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Markov.html`.

[24] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York, 4. edition, 2002. URL `http://www.mhhe.com/engcs/electrical/papoulis/`.

[25] Carl Adam Petri. *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[26] P. Reichl, D. Kesdogan, K. Junghärtchen, and M. Schuba. Simulative Performance Evaluation of the Temporary Pseudonym Method for Protecting Location Information in GSM Networks. In R. Puigjaner, N. Savino, and B. Serra, editors, *Computer Performance Evaluation - Modelling Techniques and Tools*, September 1998. URL `http://userver.ftw.at/~reichl/`.

[27] Christopher Rose and Roy Yates. Minimizing the average cost of paging under delay constraints. *Wireless Networks*, 1(2):211–219, June 1995. URL `citeseer.ist.psu.edu/rose95minimizing.html`.

[28] W. H. Sanders and J. F. Meyer. METASAN: a performability evaluation tool based on stochastic activity networks. In *ACM '86: Proceedings of 1986 ACM Fall joint computer conference*, pages 807–816, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press. ISBN 0-8186-4743-4. URL `http://portal.acm.org/citation.cfm?id=324642&dl=ACM&coll=GUIDE&CFID=78806565&CFTOKEN=84921240#`.

[29] William H. Sanders and et al. *UltraSAN - User's Manual - Version 3.0*. University of Illinois at Urbana-Champaign, 451 Coordinated Science Laboratory, MC-228, 1308 West Main Street, Urbana, IL 61801-2307, USA, 3.0 edition, 1995. URL `http://www.crhc.uiuc.edu/UltraSAN/`.

[30] Michael Schopp. Mobile Communications II. University of Stuttgart, Pfaffenwaldring 47, Stuttgart, Germany, 2005.

[31] Kimmo Suominen. Getting started with SSH. URL `http://kimmo.suominen.com/docs/ssh/`.

[32] Sami Tabbane. Location Management Methods for Third-Generation Mobile Systems. *IEEE Communications Magazine*, 35(8):72–78, 83–4, August 1997. URL `http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=606034&isnumber=13299&punumber=35&k2dockey=606034@ieeejrns&query=%28+%28+%7E%7Elocation+management+methods+for+third-generation+mobile+systems%7E%7E%3Cin%3Emetadata+%29+%3Cand%3E+%28+tabbane%3Cin%3Emetadata+%29+%29&pos=0`.

[33] Fabian Thrum. Adaptive Session Admission Control in Multiservice Mobile Cellular Networks. Master's thesis, Universidad Politécnica de Valencia & Fachhochschule der Telekom Leipzig, ETSIT, Camino de Vera, s/n, 46022 Valencia, Spain, August 2006.

[34] Bernhard Walke, Marc Peter Althoff, and Peter Seidenberg. *UMTS - Ein Kurs*. J. Schlembach Fachverlag, 2. edition, 2002.

[35] Eric W. Weisstein, editor. *CRC Concise Encyclopedia of Mathematics*. Chapman & Hall/CRC, 2. edition, 2003. URL `http://www.crcpress.com/shopping_cart/products/product_detail.asp?sku=C3472&parent_id=&pc=`.

[36] Vincent W.-S. Wong and Victor C. M. Leung. Location management for next generation personal communication networks. *IEEE Network*, 14(5):18–24, September/October 2000. URL `citeseer.ist.psu.edu/wong00location.html`.

[37] Jun Zheng and Emma Regentova. Hybrid Location Update Scheme for Future PCS Networks. *IEICE Transactions on Communications*, E88-B(1):388–391, January 2005.