Performance Modelling of Mobile Networks using MOSEL-2

Diploma Thesis in Computer Science

written by
Patrick Wüchner
born June 12th 1976 in Kronach

Department of Computer Science
(Distributed Systems and Operating Systems)
University of Erlangen-Nürnberg
Germany

School of Computing
University of Glamorgan
Wales, UK

Advisors: **Prof. Dr. rer. nat. Fridolin Hofmann** (Universität Erlangen-Nürnberg),
Prof. Dr. Khalid Al-Begain (University of Glamorgan),
Dipl.-Inf. Jörg Barner (Universität Erlangen-Nürnberg)

Begin: 1st December 2003
Submission: 1st June 2004

Abstract

The GSM/GPRS system is currently the most accepted and most used mobile technology world-wide. Although a lot of effort is done to introduce its successor UMTS, the GSM/GPRS system is still in the focus of performance evaluation to guarantee the user convenience despite increasing utilisation by applying preferably simple and cheap extensions to the system. The easiest way to improve a complex system is to optimise its software components. In general, they can be adapted more easily than hardware components.

In this thesis, the current version of the MOdelling, Specification and Evaluation Language (MOSEL-2 2.1) [1, 2, 12] will be used to model and evaluate the behaviour of the air interface of a single GSM/GPRS cell, which can be considered as the “bottleneck” of the system. Here, especially the effect of delay tolerant users is taken into account. The results of this thesis can then be used to optimise the call admission control (CAC).

The insights into restrictions of modelling and evaluating complex systems with MOSEL-2 2.1 will be used to improve MOSEL-2 by enabling the usage of SPNP 6.1 simulation and by introducing a novel rule pre-processor concept to MOSEL-2. This pre-processor allows the automatic approximation of general distributions by Markovian constructs and therefore the approximative numerical analysis of non-Markovian models.

The thesis will also give an introduction to performance modelling for readers that have little experience in building formal stochastic models. Furthermore, the implementation of the improvements, that lead to the new MOSEL-2 version 2.10, will be explained in detail to extend the MOSEL-2 documentation and to simplify the access to future enhancements of MOSEL-2.

Contents

Chapter 1

Introduction and Background

1.1 Motivation and Overview

Since the first version, the modelling power of the MOdelling, Specification and Evaluation Language MOSEL [1] (→ Chap. 1.5, p. 19) – in the meantime renamed to MOSEL-2 to emphasise the improvements [2] – increased substantially together with the evaluation power of the tools that can be used by MOSEL. Now it is time to prove its applicability for the modelling and evaluation of more complex systems.

In this thesis we will use MOSEL-2 to model different aspects of the air interface of a GSM/GPRS (→ Chap. 1.6.3, p. 28) mobile system, which is nowadays the most popular system for mobile communication world-wide. Of course, there are a lot of publications available that cover the performance evaluation of GSM/GPRS systems (see references) but only a few of them deal with delay tolerant users.

Furthermore, we will use the insights into the limitations we encountered while using MOSEL-2 as a base for more improvements to the tool.

The thesis is organised as follows: The remainder of this chapter will give a brief introduction into different performance modelling and evaluation methods, into MOSEL-2 and into the different generations of mobile (telephone) systems including GSM/GPRS. The focus of Chapter 2 will be the modelling and evaluation of a GSM/GPRS system using the current version (2.1) of MOSEL-2. Chapter 3 will be used to improve MOSEL-2 by enabling SPNP 6.1 simulation and by introducing a rule pre-processor. In Chapter 4 we will use the improved MOSEL-2 and compare the results to the ones obtained in Chapter 2. Finally, a summary and suggestions for future work are given in Chapter 5.

Some of the results of this thesis already have been published in [23].

1.2 Performance Evaluation Procedure – An Overview

The performance estimation of any real system can be done in two major ways [9]: The first way is to measure directly at the existing real system using monitors or benchmarks and perhaps using synthetic workload. But in many cases the real system is unavailable for direct measurements, because it is still under development, or it is too complicated, dangerous (e.g. measuring high-voltage, underwater or explosive systems) or expensive to take measures directly. Therefore, a second way is to build an abstract model of the real system that comprises all interesting features and to do all investigations (→ Chap. 1.4, p. 14) on this model. In this thesis, we will restrict ourselves to the second way.

The communication systems we are going to investigate, can be described as *discrete event systems*. A system is called *discrete event system* if its state can only change at a discrete set of points in time due to the occurrence of an *event*. These points of time may be unknown and irregular due to unpredictable system behaviour (e.g. user interaction). Therefore, the stochastic sojourn time the system remains in a specific state is usually modelled as continuous time random variable.

Due to the fact, that the current state of the discrete event system may be left in different ways, depending on the type of event that will occur next, the subsequent state may not be known for sure. We can handle this nondeterministic behaviour of the system, that is caused by parallel or concurrent events, by considering the probabilities of the state changes of the discrete event system (*probabilistic branching*).

The modelling of systems with stochastic behaviour often is called *stochastic modelling.* For the stochastic modelling of discrete event systems several high-level modelling formalisms based on a graphical system representation exist, e.g. *queueing networks* (→ Chap. 1.3.2, p. 8) or *Petri nets* (→ Chap. 1.3.3, p. 11). Queueing networks are very concise but lack elements for describing complex synchronisation mechanisms. The Petri nets are more suitable for modelling such systems, but quickly grow very complex and not all dependencies (e.g. *guard functions*) can be properly described graphically.

Therefore, the textual modelling language *MOSEL* (→ Chap. 1.5, p. 19) was developed. The textual method allows MOSEL to be very versatile. Furthermore, MOSEL allows to specify not only the systems behaviour but also the desired result measures.

To obtain the desired results, the MOSEL model can be either simulated or analysed numerically (→ Chap. 1.5.2, p. 22). For numerical (Markovian) analysis, the MOSEL model description gets mapped onto into a *stochastic process* class called *continuous time Markov chain* (*CTMC*). This representation of the system can become very large by the so called *state space explosion* but provides a precise mathematical framework that allows fast access to the performance measure results of interest.

Starting with the basic stochastic process underlying each discrete system description, the next section will give an introduction to the different stochastic modelling formalisms mentioned above including a discussion why or why not they should be used directly by the analyst to model a discrete systems behaviour.

Other commonly used methods for system modelling and evaluation, like *series-parallel acyclic directed graphs*, *non-series-parallel task precedence graphs*, *reliability block diagrams*, *fault trees* or *stochastic process algebras* are described in [1].

Stochastic modelling and the evaluation of these models can have different purposes [1, 57]:

- **Performance Analysis:** The measures of interest are usually the long-run averages of system properties, like the mean throughput, the mean response time or the mean utilisation of the system's components that are assumed to be reliable.
- **Reliability Analysis:** The analyst is mainly interested in the time until a component will fail, i.e. the *mean time to failure* ($MTTF$).
- **Availability Analysis:** If the components may fail and can be repaired, the probability that the system is working at a certain point of time is of interest.
- **Dependability Analysis:** Dependability describes the steadiness of the system's service. It comprises reliability and availability measures.
- **Performability Analysis:** The performability analysis investigates the effect of component failures on the system's performance.

Later, we are interested mainly in workload blocking and loss probabilities. Blocking and loss of workload, which is caused by high utilisation of components, is not considered as system or component failure. Therefore, we are dealing with performance modelling and analysis.

1.3 Performance Modelling

As mentioned before, many discrete event systems show nondeterministic, stochastic behaviour, for example due to the unpredictable nature of the systems interaction with its environment (user behaviour). Therefore, discrete event systems are usually described as stochastic processes.

A *stochastic process* $X_t(\omega)$ is a collection of *random variables* $\omega \in \Omega$ indexed in most cases by the parameter *time* $t \in \mathcal{T} \subseteq \mathbb{R}$ and is often used to characterise the change of states of a system over time [6]. The *sample space* Ω of the stochastic process is the set of all possible values of ω [7] and is mapped to the set of real numbers: $X : \Omega \to \mathbb{R}$ [6]. The set of all possible values of $X_t(\omega)$ is also called the *state space* $\mathcal{S}$ of the stochastic process [7].

1.3.1 Markov Chains

A. A. Markov [4]

Markov chains are named after the Russian mathematician *Andrei Andreyevich Markov* (1856-1922) [4], who produced the first results (1906) [5] for discrete-state stochastic processes which fulfil the *Markov property.*

A stochastic process is called a *discrete-state process* or *chain* if the state space is discrete, i.e. $X_t(\omega)$ is defined over a finite or countable set (e.g. $\mathcal{S} \subseteq \mathbb{N}$) [6, 7]. Otherwise, the process is called *continuous-state* stochastic process. We focus on discrete state spaces only.

A chain fulfils the *Markov property*, which is also known as the *memoryless property*, if the entire past history $(t_0, t_1, \ldots, t_{k-1})$ is summarised in the present (t_k) state and therefore, future $(t_{k+1}, t_{k+2}, \ldots, t_n)$ states depend on the present state only, so that the following equation of *probability mass functions* (*PMFs*) holds:

$$P[X_{t_{k+1}} = x_{k+1} | X_{t_k} = x_k, X_{t_{k-1}} = x_{k-1}, \ldots, X_{t_0} = x_0] = P[X_{t_{k+1}} = x_{k+1} | X_{t_k} = x_k] \quad (1.1)$$

If the time parameter space $\mathcal{T}$ is discrete, the Markov chain is called *discrete-time Markov chain* (*DTMC*). In this case, normally the assumption $\mathcal{T} \subseteq \mathbb{N}_0$ is made and to simplify the notation we will denote the DTMC by X_k with $k \in \mathbb{N}_0$. Otherwise, we are using $X(t)$ for representing so called *continuous-time Markov chains* (*CTMCs*).

The memoryless property has two facets [6]:

(1) All information of past states is irrelevant.

(2) The age of the current state, i.e. the time how long the process has been in the current state is irrelevant.

If the chain fulfils both aspects, it is called *Markov chain*. To fulfil the second constraint, the *cumulative distribution function* (*CDF*) of every *inter-event time* (i.e. the time between state transitions) has to be exponentially (CTMC-case) or geometrically (DTMC-case) distributed.

If a stochastic process only fulfils the first aspect, for example because it uses other, non-Markovian distributions, it is called *semi-Markov process* (*SMP*). In this case, the choice of the next state and the time the transition will happen depends on the current state and its age, but still not on any past state.

If an internal *clock structure* is added to the SMP to handle the inter-event times of all possible transitions in one state, comparable to stochastic timed automata and discrete event simulation (→ Chap. 1.4.4, p. 17), the stochastic process is called *generalised semi-Markov process* (*GSMP*).

We can define the one-step transition probabilities $p_{i,j}$ from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ [7]:
DTMC-case: transition at time $n \in \mathcal{T}$:

$$p_{i,j}(n) = P(X_{n+1} = j | X_n = i) \tag{1.2}$$

CTMC-case: transition during time period $[u, v)$ with $u, v \in \mathcal{T}$ and $u \leq v$:

$$p_{i,j}(u, v) = P(X_v = j | X_u = i) \tag{1.3}$$

And for $u = v$:

$$p_{i,j}(u, u) = \begin{cases} 1, & i = j, \\ 0, & otherwise. \end{cases} \tag{1.4}$$

If the Markov chain is *(time-)homogeneous*, i.e. the transition probabilities do not change with time, we can reduce the equations 1.2 and 1.3 to:

DTMC-case: the conditional PMF is independent from the observation time:

$$p_{i,j} = P(X_1 = j | X_0 = i) \tag{1.5}$$

CTMC-case: the conditional PMF only depends on the time difference $t = v - u$:

$$p_{i,j}(t) = P(X_t = j | X_0 = i) \tag{1.6}$$

The transition probabilities are usually collected in a stochastic *transition probability matrix* [7, 1]:

homogeneous DTMC-case:

$$\mathbf{P} = [p_{i,j}] = \begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots \\ p_{1,0} & p_{1,1} & p_{1,2} & \cdots \\ p_{2,0} & p_{2,1} & p_{2,2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{1.7}$$

homogeneous CTMC-case:

$$\mathbf{P}(t) = [p_{i,j}(t)] = \begin{pmatrix} p_{0,0}(t) & p_{0,1}(t) & p_{0,2}(t) & \cdots \\ p_{1,0}(t) & p_{1,1}(t) & p_{1,2}(t) & \cdots \\ p_{2,0}(t) & p_{2,1}(t) & p_{2,2}(t) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{1.8}$$

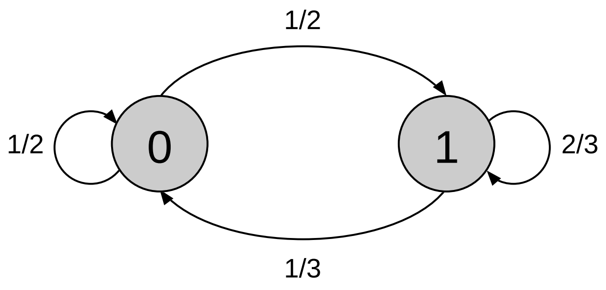

Figure 1.1: Example for a very simple DTMC.

A matrix is called stochastic, if the following conditions hold [8]:

1. The matrix is quadratic (in our case with order $|\mathcal{S}|$):
$$i, j \in \{1, 2, \ldots, |\mathcal{S}|\} \tag{1.9}$$
2. The matrix is non-negative:
$$p_{ij} \geq 0 \quad \forall i, j \in \{1, 2, \ldots, |\mathcal{S}|\} \tag{1.10}$$
3. Every row sums up to 1:
$$\sum_{j=1}^{|\mathcal{S}|} p_{i,j} = 1 \quad \forall i \in \{1, 2, \ldots, |\mathcal{S}|\} \tag{1.11}$$
4. At least one element in each column differs from zero:
$$\exists i \in \{1, 2, \ldots, |\mathcal{S}|\} : p_{i,j} \neq 0 \quad \forall j \in \{1, 2, \ldots, |\mathcal{S}|\} \tag{1.12}$$

Besides the mathematical representation of transition probabilities in matrices, they can be graphically illustrated.

Consider the transition probability matrix of a simple DTMC with state space $\mathcal{S} = \{0, 1\}$.

$$\mathbf{P} = \begin{pmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & p_{1,1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{3} & \frac{2}{3} \end{pmatrix} \tag{1.13}$$

In Figure 1.1 the graphical representation of this DTMC chain is shown. The two filled circles represent the two states of the system. The arcs between different or identical states represent one-step transitions from one state to another or the same state. We label each transition from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ with the conditional PMF of the transition probability.

However, (homogeneous) CTMCs play a more important role in performance modelling than DTMCs. But in contrast to DTMCs, the transition probability matrix $\mathbf{P}$ of a CTMC cannot be built easily from the known system behaviour. Therefore, the *transition rates* $q_{i,j}$ from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ with $j \neq i$ are considered instead. The transition rates are collected in the *transition rate matrix* $\mathbf{Q}$, which is also known as the *infinitesimal generator* of the transition probability matrix or in short *generator matrix* [1, 7]. The transition rate matrix $\mathbf{Q}$ contains the transition rates $q_{i,j}$ for $i \neq j$ and the diagonal elements [1, 14]:

$$q_{i,i} = -\sum_{j \neq i} q_{i,j} \tag{1.14}$$

Theoretically, the transition rates are related to the transition probabilities via [1]:

$$\mathbf{Q} = \lim_{t \to 0} \frac{\mathbf{P}(t) - \mathbf{I}}{t} \tag{1.15}$$

But practically, the transition rate matrix $\mathbf{Q}$ can be created directly out of the known transition rates of the model description. An example is given in Section 1.4.3.

The advantage of Markov chains, which can be used for performance, reliability, availability and dependability analysis [1], is that they provide a precise mathematical framework. Furthermore, Markov chains can be solved using standard numerical solution algorithms (→ Chap. 1.4.2, p. 14).

Unfortunately, Markov chains can only be used, if the model of the system obeys the Markov property. Another disadvantage is, that a lot of real systems consist of workload processed by components, which cannot be easily and intuitively mapped to low level formalisms like Markov chains. Especially concurrent and asynchronous behaviour of parallel system components cannot be described explicitly in Markov chains. Moreover, Markov chains are getting very complex because of state space explosion in larger models. Therefore, high level modelling paradigms like *queueing networks* (→ Chap. 1.3.2, p. 8) or *Petri nets* (→ Chap. 1.3.3, p. 11) and tools for transferring them to Markov chains automatically were developed.

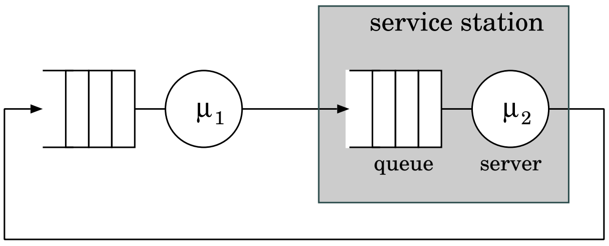

Figure 1.2: Example for a simple queueing network: closed tandem network.

1.3.2 Queueing Networks

A *queueing network* is a set of connected *service stations* which are also called *nodes*. Each service station (node) consists of a *queue* and one or several *servers* and represents a component in the real system. The workload of queueing networks is processed by these *service stations* and often referred to as *jobs*. One or several jobs may wait in the queue while another job is served in the server. The servers might be denoted with an exponentially distributed service rate, i.e. the number of jobs that can be served per time unit. As an example, a simple queueing network is shown in Figure 1.2. This configuration is also called *closed tandem network*, because there is no connection to the environment and because the system consists of two *service stations* in serial. The servers in the tandem network have exponentially distributed service rates of μ_1 and μ_2, respectively.

The properties of each service station are described by (the extended) *Kendall's notation* [7, 1, 9, 14, 17]:

$$\mathrm{A/S/N\ -\ Q}$$

or (extended):

$$\mathrm{A/S/N/B/P/Q}$$

or (for *batch processing*):

$$\mathrm{A/S}^{[a,b]}\mathrm{/N}\ldots$$

With:

A (Arrival process): Declaration of the job inter-arrival time distribution using the following symbols:

M	: exponential (*Markov property* (→ Chap. 1.3.1, p. 4), *Poisson process*)
E_k	: Erlang with k phases
H_k	: hyperexponential with k phases
C_k	: Cox with k phases
PH	: phase-type
D	: deterministic (constant)
G	: general (arbitrary)
GI	: general independent

S (Service process): Declaration of the service time distribution using similar symbols. The service time is the time that the server needs to process a single job. If *batch processing* of several jobs at once is used, **a** defines the minimum, **b** the maximum batch size [14].

N (Number of servers): Number of identical servers in service station.

B (Buffer size): Maximum number of jobs in the service station (in the queue and in service).

P (Population size): Total number of potential jobs that may want to enter the service station, also known as the *calling population* [17].

Q (Queueing discipline): Order in which jobs are served and policy for displacing of jobs in service. If there is no replacement, the discipline is called *non-preemptive*, otherwise *preemptive.*

FCFS	(first-come-first-served): Jobs are served in the order in which they arrived (non-preemptive).
LCFS	(last-come-first-served): The last job that arrived will be served next. This discipline can be preemptive or non-preemptive. Possible resumption policies could be [9, 10]:

PR or PRS	(preemptive resume) : The displaced job will be suspended and finished after resumption.
PRD	(preemptive repeat different) : The displaced job will be stopped and restarted after resumption. The service time will be re-sampled.
PRI	(preemptive repeat identical) : The displaced job will be stopped and restarted after resumption using the same service time as before.

SIRO	(service-in-random-order): The jobs are randomly chosen out of the queue.
RR	(round robin): Each job is given a time slice of specific length. If the job is not completed by the end of this time slice, it is replaced and put back to the (FCFS-)queue. Normally the preemptive resume policy is chosen for this queueing discipline.
PS	(processor sharing): This is similar to *round robin* except that the time slices are considered as infinitesimal short. All jobs are then served simultaneously with a corresponding longer service time.
IS	(infinite server): There is a sufficient number of servers in the service station, so that there will be no jobs lining up in a queue, no displacement and no resumption.
Priorities	Static or dynamic (altering with the passing of time) priorities are assigned to the jobs. The jobs are selected according to their priority. Jobs with the same priority normally follow the FCFS-discipline. If the priorities are said to be preemptive, arriving jobs are displacing jobs in service with lower priorities.

In many cases, the notation is reduced to the first few elements, e.g. *M/M/1*. If omitted, an infinite buffer size, infinite population size and the queueing discipline FIFO is assumed.

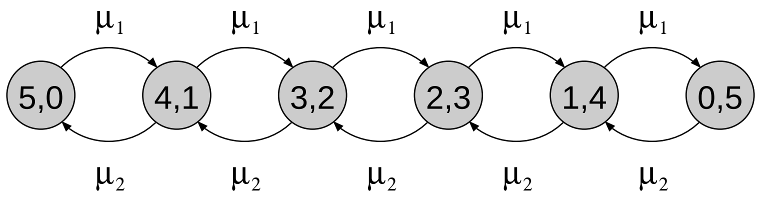

Figure 1.3: CTMC of closed tandem network.

As mentioned above, *queueing networks* are – unlike *Markov chains* (→ Chap. 1.3.1, p. 4) – quite handy and intuitive for modelling systems consisting of resources (service stations) and workload (jobs).

For example: We consider the following parameters for the tandem network shown in Figure 1.2:

- Total number of jobs in the system: $K = 5$
- Both service stations are $M/M/1/K$ systems.
- The *service rate* (i.e. the number of jobs getting processed within a certain time unit) of the first server is μ_1.
- The service rate of the second server is μ_2.

In Figure 1.3 the corresponding *CTMC* of this tandem network is shown. As we can see even in this simple example, the *state space explosion* quickly makes the Markov chain quite large. Furthermore, the components of the real system cannot be seen easily in the Markov chain.

Queueing networks are suitable for modelling systems with a lot of independent service stations (resources), which are used (occupied) by jobs sequentially, and with simultaneous processing of different jobs by different service stations.

Unfortunately, queueing networks are only useful for performance analysis [1] - which would be enough for our application - but also lacks some features like modelling aspects of synchronisation, concurrency and conflict [1, 9].

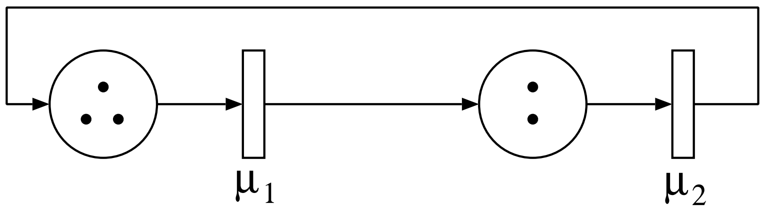

Figure 1.4: Petri net of closed tandem network.

1.3.3 Petri Nets

Petri nets were designed 1962 by Carl Adam Petri [11]. Since then they were permanently extended. A short overview of the different kinds of Petri net is given in [12]. In this thesis we are particularly interested in *generalised stochastic Petri nets* (*GSPNs*), *deterministic and stochastic Petri nets* (*DSPNs*), *extended deterministic and stochastic Petri nets* (*eDSPNs*) and *extended stochastic Petri nets* (*ESPNs*).

In Figure 1.4 the graphical representation of a simple Petri net – again a *closed tandem network* – is shown. As we can see, a basic Petri net consists of *places* (circles), *transitions* (bars), *arcs* connecting places with transitions and *tokens* (dots). The state of a Petri net is defined by the distribution of tokens among the places. This distribution of tokens is also called the (current) *marking* of the Petri net. The marking can change by the *firing* of a transition, i.e. a token is transferred from each input place of this transition to each output place. A transition fires, if it is enabled and the *firing time* expires. A transition is enabled when all input places contain at least one token. The firing time of a transition can be zero. The transition is then said to be *immediate*. The firing times can also show stochastic behaviour, e.g. they can be exponentially distributed. If a Petri net only consists of immediate and exponential transitions, it is a *GSPN*. If it as well contains transitions with constant (i.e. deterministic) firing time, it is a *DSPN*. Uniform distributions are additionally allowed in *eDSPN*. *ESPN* may contain transitions with arbitrarily distributed firing time.

In this thesis, we will represent transitions with ...

... no firing time (immediate) by bars.
(→ Fig. 1.5, p. 12)

... exponentially distributed firing time by unfilled rectangles.
(→ Fig. 1.4, p. 11)

... deterministic or uniformly distributed firing time by striped rectangles.
(→ Fig. 1.7, p. 13)

... arbitrarily distributed firing time by greyly filled rectangles.
(→ Fig. 1.7, p. 13)

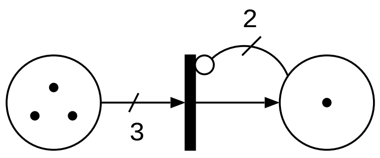

Figure 1.5: Structural extensions of Petri nets.

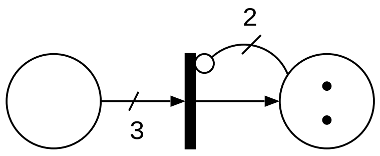

Figure 1.6: Structural extensions of Petri nets (after firing).

In Figure 1.5 two well known structural extensions to basic Petri nets are shown [13, 58]:

- *arc multiplicity*: Arcs may be labeled with a multiplicity. In Figure 1.5 the immediate transition is only enabled, if the input place contains at least three tokens. If the transition fires, three tokens are removed from the input place. However, only one token will enter the output place because the output arc does not have a multiplicity.

- *inhibitor arc*: The transition can be disabled by an inhibitor arc. Inhibitor arcs are represented by arcs with small circles at the transition end (instead of arrows). For disabling, the inhibiting place has to contain at least as much tokens as the multiplicity declares. Originally, inhibitor arcs were introduced without multiplicity to extend (unbounded) Petri nets to the representational power of Turing machines. The multiplicity of inhibitor arcs is a structural extension not for increasing the modelling power, but for increasing the modelling convenience. Inhibitor arcs with multiplicity could be described textually by using so called *guard functions* as well. Unfortunately, there is no common graphical representation for guard functions.

  Actually, we do not reach the power of Turing machines. Instead, we use the (multiple) inhibitor arcs to stress the finite capacity of the places. We assume finite capacity for each place, and therefore *bounded* Petri nets, to keep the potential state space finite, which is the Cartesian product of the occupancy vector of all places, i.e. all possible distributions of tokens among the places.

In Figure 1.6 the same network is shown after the firing of the transition. The transition is now disabled, not only due to the fact, that the input place does not contain at least three tokens, but also because of the active inhibitor arc.

We will stick to Petri nets when building graphical representations of conceptual models that will be described in MOSEL-2 syntax afterwards.

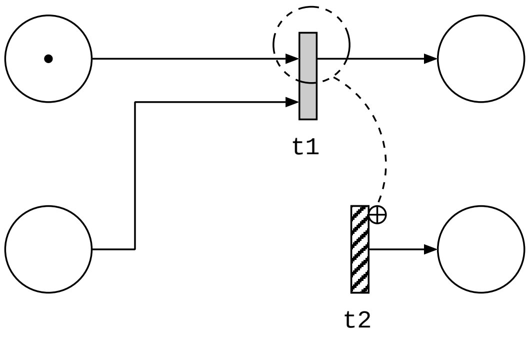

Figure 1.7: Petri net with positive dependence arc.

To reduce the complexity of some Petri net models we will build later (→ Chap. 3.2, p. 64), we develop a novel structural extension to the graphical representation: *positive* and *negative dependence arcs*. Dependence arcs are comparable to inhibitor arcs or can be seen as a graphical representation of a special class of *guard functions*. They as well can disable transitions. The difference between dependence and inhibitor arcs is, that the condition for the disabling of a so called *slave transition* is not a predicate on the number of jobs in a certain place but the current state of another transition which is called *master transition*.

The extension is exemplarily introduced in Figure 1.7. The dependence arc is dotted. There is a small circle marked with a plus sign for positive dependence arcs or a minus sign for negative dependence arcs at the end that is connected to the slave transition that will be controlled. At the other end of the arc is a dotted circle that marks the (part of the) master transition which controls the slave transition. A slave transition with a positive dependence arc can only be enabled, if the marked (part of the) master transition is enabled. A slave transition with a negative dependence arc can only be enabled, if the marked (part of the) master transition is disabled.

In Figure 1.7 the slave transition **t2** is enabled although transition **t1** is disabled, because the master transition **t1** would be enabled, if the lower part, which is ignored by the dependence arc, would not block the transition.

So far, Petri nets seem to be a good compromise between modelling power, complexity and clarity. But for very large systems, the building of graphical representations becomes quite complex and cumbersome. Furthermore, despite all structural extensions the graphical representation of system behaviour will always have to be completed with textual descriptions of subtleties that cannot be expressed properly in graphical form. Therefore, we will use the textual modelling language of MOSEL-2 (→ Chap. 1.5, p. 19) to describe the complete model for evaluation.

1.4 Performance Evaluation

After the model is built, it can be evaluated in different ways. The applicable procedures highly depend on the desired results and on the properties of the model.

1.4.1 Performance Measures

The purpose of performance evaluation is to determine different *performance measures.* The impact of system parameter changes on the performance measures can then be investigated. The obtained knowledge about the causal relationship can then be used to optimise the system.

Common performance measures of interest are [9, 1, 7]:

- the mean response time
- the utilisation
- the mean throughput
- the mean or distribution of the queue length
- the mean waiting time
- the mean work in progress

Furthermore, we will be especially interested in:

- blocking probability (a job cannot be served immediately)
- loss probability (a job gets lost)

1.4.2 Product Form Solution

Solution methods based on *product form* models are mainly used in the context of *queueing networks* (→ Chap. 1.3.2, p. 8). A queueing network is called in product form if it fulfils the following conditions [9, 14, 7]:

- Local Balance Property (necessary and sufficient): A network is in *local balance* iff the rate at which a state of the network is left because a job leaves a certain *node* (*service station*) equals the rate at which the network re-enters the state because of a job arriving at that node. This means, that if the queueing network is mapped to a *CTMC* (→ Chap. 1.3.1, p. 4), the rate at which the CTMC enters a state equals the rate at which the CTMC leaves the same state.
- $M \Rightarrow M$ Property ("Markov Implies Markov", necessary and sufficient): All nodes of the network have to abide the *$M \Rightarrow M$ property*, which is fulfilled iff that node transforms a Poisson (→ Chap. 1.3.1, p. 4) arrival process into a Poisson departure process.

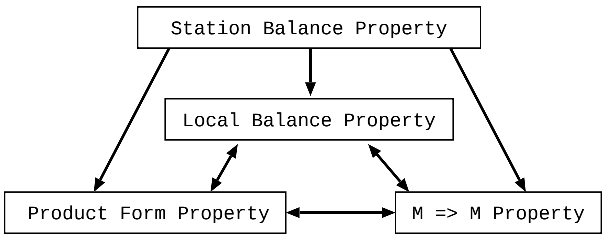

Figure 1.8: Conditions for product form.

- Station Balance Property (sufficient): All queues in the network are considered partitioned into positions. The rates at which jobs enter and leave these partitions have to be equal.

The dependencies of the properties are shown in Figure 1.8 [7, 14].

It can be shown, that a queueing network is in product form, if it consists of the following service station types only [9, 7, 14]:

- *Type-1:* **M/M/m-FCFS** (Limitation: No job classes with different service rates.)
- *Type-2:* **M/G/1-PS**
- *Type-3:* **M/G/∞ IS**
- *Type-4:* **M/G/1-LCFS PR**

For product form queueing networks there exist several theorems for very efficient and exact evaluation without the generation of the state-space [7, 9, 14]:

- **Jackson Theorem:** This theorem provides exact results for open networks with a single class of jobs and an infinite number of jobs. All service stations are following the FCFS queueing discipline and may receive and send jobs from and to the external environment. All arrival and service rates are exponentially distributed but may be load dependent.
- **Gordon/Newell Theorem:** This is an extension to the Jackson-Theorem for handling closed networks. Here, no jobs may enter or leave the system, i.e. the number of jobs is constant. Besides that, all other restrictions of the Jackson-Theorem must be met.
- **BCMP Theorem:** Using the *Baskett, Chandy, Muntz and Palacios*-Theorem (*BCMP-Theorem*), results for all four product form service station types mentioned above can be provided. Furthermore, multiple job classes are allowed, but in nodes with FCFS queueing discipline all job classes must have the same service rate. The network can be open, closed or mixed.
- **Convolution Algorithm:** The convolution algorithm is based on the BCMP theorem but provides an easier way to determine the normalisation constant.

- **Mean Value Analysis:** The mean value analysis is a consequence of the convolution algorithm, but no normalisation constant has to be computed here, because the mean values of the desired results are determined iteratively.

There are also different methods for the approximative analysis of product form nets. The reader is referred to [7] or [9] for details.

The product form solution on one hand is a very efficient way to get results, but on the other hand, these methods impose strong constraints on the models. Therefore, we are searching for more universal evaluation methods.

1.4.3 Markovian Analysis

The *Markovian analysis* is based on *Markov chains* (→ Chap. 1.3.1, p. 4). For this, the model has to fulfil the *Markov property* like, for example, *GSPNs* (→ Chap. 1.3.3, p. 11). Then, the model can be mapped to a *homogeneous CTMC*. This is done by, speaking in terms of Petri nets, reducing the *reachability graph* (all markings that are reachable from the initial marking) by removing all *vanishing markings*, i.e. markings that have at least one enabled immediate transition. The resulting *reduced reachability graph* corresponds to a homogeneous CTMC.

For example, if the two transitions of the Petri net in Figure 1.4 have exponentially distributed firing times with firing rates μ_1 and μ_2, the resulting GSPN is isomorphic to the CTMC in Figure 1.3.

As mentioned before, the *transition rate matrix* $\mathbf{Q}$ can be build directly using the transition rates $q_{i,j}$ from state i to state j and the diagonal elements [1, 14]:

$$q_{i,i} = -\sum_{j \neq i} q_{i,j} \tag{1.16}$$

This leads to the following transition rate matrix for the CTMC in Figure 1.3:

$$\mathbf{Q} = \begin{pmatrix} -\mu_1 & \mu_1 & 0 & 0 & 0 \\ \mu_2 & -\mu_1 - \mu_2 & \mu_1 & 0 & 0 \\ 0 & \mu_2 & -\mu_1 - \mu_2 & \mu_1 & 0 \\ 0 & 0 & \mu_2 & -\mu_1 - \mu_2 & \mu_1 \\ 0 & 0 & 0 & \mu_2 & -\mu_2 \end{pmatrix} \tag{1.17}$$

The transition rate matrix $\mathbf{Q}$ can then be used to calculate the *state probability vector* $\boldsymbol{\pi}$ [1, 14, 7]:

- **transient analysis:**

$$\boldsymbol{\pi}(t)\mathbf{Q} = \frac{d\boldsymbol{\pi}(t)}{dt} \tag{1.18}$$

  together with the initial probability vector $\boldsymbol{\pi}(\mathbf{0}) = (\pi_0(0), \pi_1(0), \ldots)$.

- **steady-state / stationary analysis**

$$\boldsymbol{\pi}\mathbf{Q} = \mathbf{0} \tag{1.19}$$

The major task is then to solve equation 1.18 or 1.19, which can be done in several ways:

- **direct (exact) methods:**
  - **steady-state:** The homogeneous linear equation 1.19 can be directly solved by using, e.g., the well known *Gaussian elimination* method [15, 7] or the *Grassmann algorithm*, that is based on the Gaussian elimination method, but is less sensitive to rounding and cancellation errors [7].
  - **transient:** The solving of differential equation 1.18 is a very difficult task. It can be done, e.g., for simple M/M/1-systems by using Bessel functions [16].
- **numerical methods:**
  - **steady-state:**
    - iterative numerical method [9, 14]
    - recursive numerical method [9]
    - power method [7, 10, 14]
    - Jacobi method [7, 14]
    - Gauss-Seidel method [7, 10]
    - method of successive over-relaxation [7, 10]
  - **transient:**
    - standard, Fox-and-Glynn or stiff uniformisation [7, 10, 14]

As soon as the *state probabilities* are calculated, the *marginal probabilities* can be derived and using these, all other performance measures can be calculated [9, 14].

The *Markovian analysis* is a very powerful and universal method for solving models that fulfil the *Markov property*. Unfortunately, *Markov chains* are growing very fast with increasing model size and therefore a lot of memory is needed for solving big models.

As mentioned before, *Markov chains* are not very convenient for being created as system description directly by the analyst. But many high-level modelling packages include tools for the automatic generation of the underlying Markov chain, that can then be solved mathematically.

Another disadvantage is, that not all system behaviour can be adequately described with exponentially distributed firing times and that only for a few non-Markovian Models there exist numerical solutions, like DSPNs [2].

1.4.4 Discrete Event Simulation

The *discrete event simulation* (*DES*) does not make such high demands on the model that has to be evaluated. The DES can deal with general as well as memoryless distributions. Furthermore, it is able to handle very large models, it just takes more CPU time [18, 10]. In DES, the state of the system is still described by its marking, but in addition to that, a continuous clock vector is introduced which describes the *remaining firing times* (*RFTs*, *state sojourn times*, *transition clocks*) [18, 12]. This extension makes the description of the models state equivalent to a GSMP (→ Chap. 1.3.1, p. 4).

The execution of the discrete event simulation represents the dynamic behaviour of the model and thus determines the desired values [1]. The following algorithm describes a simplified simulation run [12]:

1. initialisation:
    - set the global simulation clock to zero
    - set the start state/marking of the Petri net
    - define the termination event, e.g. a certain simulation length
2. determine the set of enabled transitions (goto step 8 if set is empty)
3. determine the firing time of each enabled transition by drawing a sample from each associated probability distribution.
4. find the transition with the shortest remaining clock time (i.e. firing time) and increment the simulation clock. (The simulation algorithm has to ensure, that there is only a single transition that can fire at a certain time.)
5. goto step 8 if the termination event was reached
6. compute the new marking obtained by the firing of the transition and the marking dependent results
7. continue with step 2
8. end of simulation, provide results

The simulation runs are repeated very often and/or the simulation length is very high, so that it is ensured that every state of the model is reached many times. Then the relative quantities of the occurrence of each state will approximate the state probabilities (*Monte Carlo method*) [2, 19, 20].

The great advantage of DES is, that we can evaluate stochastic processes that are described as GSMP. So we can theoretically use any distribution of firing times for modelling the behaviour of the real system. Another advantage is, that the need for memory during the computation is almost independent of the model size.

Unfortunately, the simulation of large models and models with rare events consumes a lot of CPU time to obtain enough accuracy of results. Remedy may result from using different *speed-up methods* like *importance splitting* [21, 10] or *importance sampling* [22, 10] techniques. But more often non-Markovian system behaviour is approximated by Markovian models to be able to use analytic-numerical solution methods.

1.5 Introduction to MOSEL-2

The *MOdelling, Specification and Evaluation Language* version 2 (*MOSEL-2*) was – like its predecessor *MOSEL* [1] – developed at the University of Erlangen, Germany. In the beginning of this work MOSEL-2 is available as version 2.1, that comprises all features presented in [2] and [12].

The main idea behind MOSEL-2 is to provide a system description language that is:

- not evaluation tool- but system-oriented
- easy to understand and to easy to learn but powerful
- universally applicable but evaluation method independent

With other words, the user should not have to learn a new modelling language, just because he has to change a single parameter (e.g. the distribution of a transition's firing time) in his system description and therefore has to change the evaluation method and tool as well.

MOSEL-2 does not provide implementations of methods for solving models - neither by (numerical) analysis nor simulation. Rather, it comprises translators for several already existing, well-tested and mature performance evaluation tools.

In this section, a brief introduction to the usage of MOSEL-2 will be given using a simple example similar to the *closed tandem network* shown before (→ Fig. 1.4, p. 11).

Description of the rough work-flow of the performance evaluation procedure using MOSEL-2 (→ Fig. 1.9, p. 20) [23]:

1. Inspecting the real system, the modeller builds a high-level, textual system description using the MOSEL-2 specification language[1]. Furthermore, he specifies the desired performance and/or reliability measures. This description is used as an input file for the MOSEL-2 environment (→ Chap. 1.5.1, p. 21).
2. The MOSEL-2 environment checks the input file for syntactical and semantical errors. If no error occurred, the MOSEL-2 model gets translated to the system description of the evaluation tool selected by the user.
3. The chosen tool is started by the MOSEL-2 environment.
4. The tool evaluates the model by using numerical analysis methods or discrete event simulation (→ Chap. 1.5.2, p. 22).
5. After the evaluation the tool saves its results in one or more tool-specific output files.
6. The MOSEL-2 environment parses the tool-specific output file(s) and provides the results in a single textual and, if specified, graphical result file (→ Chap. 1.5.3, p. 23).

[1]If the system is very complex, he might draw an abstract conceptual model using graphical syntax, e.g. queueing networks or Petri nets, beforehand.

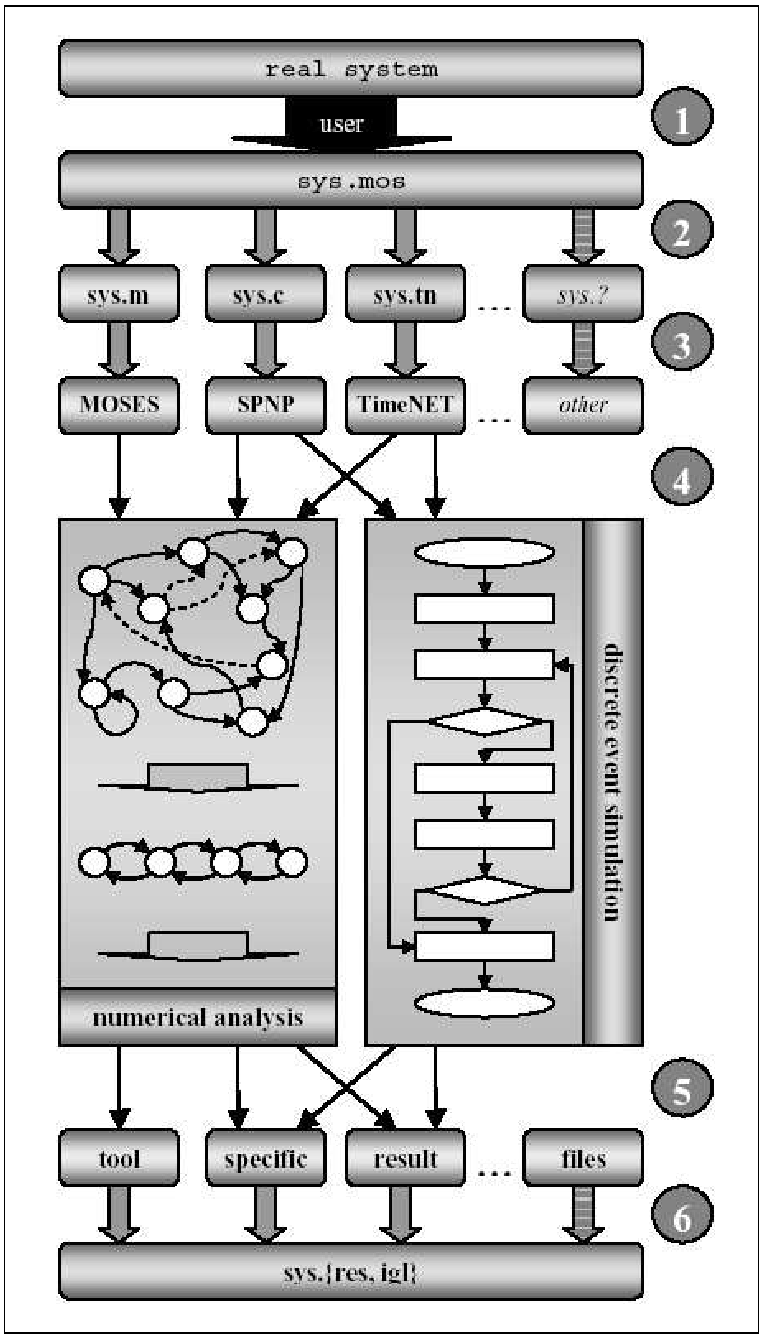

Figure 1.9: The modelling and analysis process in the MOSEL-2 environment.

1.5.1 System Description in MOSEL-2

We will now do step one, and build a possible description of the closed tandem network (→ Fig. 1.4, p. 11) in MOSEL-2. can be[2]:

```
 1  CONST rate := 4;                      /* rate of first transition           */
 2  CONST delay := 0.5;                   /* delay of second transition         */
 3  CONST tokens := 3;                    /* number of tokens in closed system  */
 4  CONST init1 := 2;                     /* initial number of tokens in place1 */
 5  CONST init2 := tokens-init1;          /* initial number of tokens in place2 */
 6  NODE place1[tokens] := init1;         /* 'place1' with capacity 'tokens'    *
 7                                         * and 'init1' initial tokens         */
 8  NODE place2[tokens] := init2;         /* 'place2' with capacity 'tokens'    *
 9                                         * and 'init2' initial tokens         */
10  ASSERT place1 + place2 = tokens;      /* the number of tokens in the system *
11                                         * always has to be 'tokens'          */
12  FROM place1 TO place2 RATE rate;      /* expon. transition from 'place1' to *
13                                         * 'place2' with parameter 'rate'     */
14  FROM place2 TO place1 AFTER delay;    /* determ. transit. from 'place2' to  *
15                                         * 'place1' with parameter 'delay'    */
16  PRINT avg_no_of_tokens_in_place1 := MEAN(place1);
17  PRINT avg_no_of_tokens_in_place2 := MEAN(place2);
18  PRINT usage_of_place1 := PROB(place1 != 0);
19  PRINT usage_of_place2 := PROB(place2 != 0);
20  PRINT DIST place1;
21  PICTURE "token distribution of place1"
22  CURVE DIST place1
23  YLABEL "PROB"
24  XLABEL "Number of Jobs"
```

The MOSEL-2 model description can be divided into six parts:

- constant and parameter part (lines 1 to 5)
- node part (lines 6 to 11)
- function and condition part (unused)
- rule part (lines 12 to 15)
- result part (lines 16 to 20)
- picture part (lines 21 to 24)

A more detailed description of the MOSEL-2 syntax and semantics can be found in [2]. MOSEL-2 language extensions for supporting non-Markovian distributions, that can be evaluated with the help of SPNP 6.1.2a simulation (→ Chap. 1.5.2, p. 22), are provided in [12].

[2]The line numbers specified in the first column are not part of the MOSEL-2 model description. They only serve referencing.

1.5.2 Evaluation Methods

MOSEL-2 is available as version 2.1. MOSEL-2 2.1 is able to use the following evaluation tools:

- *MOSES* [25]
- *TimeNET* (version 3.0) [2, 24]
- *SPNP* (version 6.1.2a) [12, 10]

The evaluation tools and methods that can actually be used, depend on the type of the model (→ Chap. 1.4, p. 14) [2, 10]:

Type of		**MOSES**	**SPNP**		**TimeNET**	
Model	**Result**	**num. ana.**	**num. ana.**	**DES**	**num. ana.**	**DES**
GSPN	stationary	X[1]	X[2]	X	X	X
	transient	-	X[3]	X	X	X
DSPN	stationary	-	-	X	X[4]	X
	transient	-	-	X	X[4]	X
eDSPN	stationary	-	-	-[5]	?[6]	X
	transient	-	-	-[5]	-	X
ESPN	stationary	-	-	X[7]	-	-
	transient	-	-	X[7]	-	-

There are further restrictions affecting the applicability of the different tools. See [2] for details.

The user can specify the desired tool and its options while invoking the MOSEL-2 environment from *command line*, e.g. with:

```
mosel2 -csk -o +SIMULATION tandem.mos
```

For a list and description of all command line options see [2] or [12] or call MOSEL-2 with option "-h".

[1]Methods: power (= Jacobi), power2 (= Gauss Seidel), LPU (default), Crout and Grassmann [25]

[2]Methods: SOR, Gauss Seidel and power [10]

[3]Methods: standard uniformisation, Fox and Glynn uniformisation, stiff uniformisation [10]

[4]At most one non-exponential timed transition may be enabled at any time. (But DSPNs with several deterministic transitions enabled at the same time can be approximatively evaluated.) [2]

[5]The uniform distribution did not work in SPNP 6.1.2a. [12]

[6]According to [2], the steady-state analysis of a model containing uniform distributions should be possible, if not more than one uniform or deterministic transition is enabled at any time. The evaluation of such models is neither forbidden by MOSEL-2 2.1 nor by TimeNET. Nevertheless, no results could be obtained even for very simple models ("Error in StatAnalysis/src/erand.c: Segmentation fault").

[7]The following SPNP 6.1.2a distributions besides DSPN distributions are supported by MOSEL-2 2.1: beta, Cauchy, deterministic, gamma, geometric, hyperexponential, hypoexponential, lognormal, normal, Pareto, Poisson. [12]

```
Simulation of "tandem2.mos" by SPNP (length: 1000, runs: 1000)

Results:
  avg_no_of_tokens_in_place1 = 1.258
  avg_no_of_tokens_in_place2 = 1.742
  usage_of_place1 = 1
  usage_of_place2 = 0.964

Distribution of "place1":
  P{place1 = 0} = 0
  P{place1 = 1} = 0.778
  P{place1 = 2} = 0.186
  P{place1 = 3} = 0.036
```

Figure 1.10: Textual tandem network results (tandem.res).

1.5.3 Results

The results can be provided in two different ways as specified in the result and picture part: textual (→ Fig. 1.10, p. 23) and graphical (→ Fig. 1.11, p. 23).

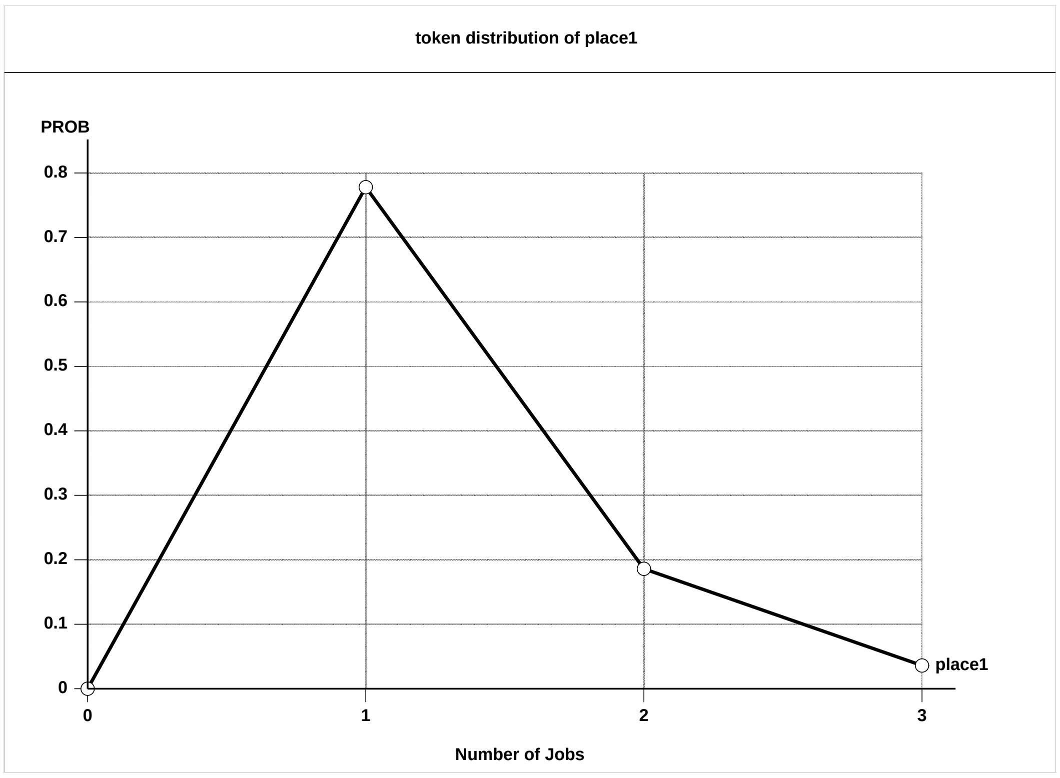

Figure 1.11: Graphical tandem network results (tandem.igl).

1.6 Mobile Communication Systems

Mobile communication systems can be categorised by several attributes:

- dimension, scale, distance between processors [26, 27]:
  - up to approx. 2 m: *personal area network* (*PAN*)
  - up to approx. 1 km: *local area network* (*LAN*)
  - up to approx. 10 km: *metropolitan area network* (*MAN*)
  - up to approx. 1000 km: *wide area network* (*WAN*)
- transmission technology [26, 27]:
  - broadcast
  - multicast
  - unicast
  - point-to-point
- usage
- protocol
- speed
- type of data

As an example we will focus on the GSM/GPRS mobile system and use MOSEL-2 later to model and evaluate the air interface of a single GSM/GPRS cell, which can be considered as the "bottleneck" of the GSM/GPRS system.

In this section a short introduction into these mobile communication technologies will be given: In Sections 1.6.1 to 1.6.4 the different generations of mobile phone systems are described. Generation 2.5 (2.5G) includes the GSM/GPRS technology.

1.6.1 1G: Analog Cellular Mobile Radio Telephone

Wireless telephone systems can be divided into two groups: mobile telephone systems and cordless telephone systems. Cordless telephones usually consist of a base station and a telephone, both used together in a single building. The range between those to parts is up to 300 meters and most systems are nowadays following the DECT standard [27].

However, we are interested in mobile telephone systems. The currently most popular mobile telephone system *GSM* ($\rightarrow$ Chap. 1.6.2, p. 25) is a successor of the first cellular mobile radio telephone systems that used analog voice signalling.

Examples of this first analog systems, that are also referred to as *first generation* (*1G*) mobile phone systems, are:

- mobile radio telephone system (early 20th century, e.g. CB-radio) [27]: single channel, i.e. only sending or receiving exclusively
- improved mobile telephone system (*IMTS*, 1960, USA) [27]: big cells (100km), only 23 channels but 2 frequencies, one for sending and one for receiving
- A-Netz (1958, Germany) [3]: connection establishment from mobile phone only
- B-Netz (1972, Germany, Austria, Netherlands, Luxembourg) [3]: connection establishment also from fixed network if location of mobile phone is known
- advanced mobile phone system (*AMPS*, 1982, USA, Great Britain (TACS), Japan (MCS-L1)) [27, 26, 3]: smaller cells (10-20km) allow lower transmission power and closer reuse of frequencies (*cell concept*, *space division multiple access*, *SDMA*), 832 full-duplex channels by *frequency division multiple access* (*FDMA*)
- C-Netz (1986, Germany) [3]: digital signalling (but still analog voice transmission), handover possible

1.6.2 2G: AMPS, CDMA and GSM

The first systems using digital voice signalling are often referred to as *second generation* (*2G*) mobile phone systems. The introduction of digital mobile phones was done in two different ways in Europe and the USA.

Launch of AMPS and CDMA

All over the USA, a single analog system (*AMPS*) existed. The second generation of this system is the backward compatible *D-AMPS* [26]. *D-AMPS* was introduced in 1991 based on the standards IS-54 and IS-135, but parallel several competitors tried to introduce their own standards. The only standard that survived besides *D-AMPS* is IS-95, the *code division multiple access* (*CDMA*) variant which is based on direct sequence spread spectrum (→ Chap. 1.6.4, p. 32) [3, 26, 27]. A modified form of *D-AMPS* named *PDC* is in use in Japan since 1993 [3, 26].

Launch of GSM

In Europe, it was just the other way round: Almost every country had its own analog system. Therefore, the telecommunication authorities of all European countries agreed on building a common digital system from scratch without compromises for backward compatibility, called *GSM* (now: *global system for mobile communication*) [27]. The first specifications were already done in 1982 by the *Groupe Spéciale Mobile* [3]. Later, GSM was developed and standardised by the *European Telecommunications Standards Institute* (*ETSI*) [30]. The first countries started using GSM in 1992 [3].

Nowadays, almost all countries in the world (except the USA, Japan and perhaps some smaller countries) are using GSM. Even in the USA it achieves some acceptance [26].

Specification of GSM

Due to the fact, that the whole GSM specification extents over 5000 pages [26], we will only have a look at the basic concepts – mainly of the physical and medium access layer – that are necessary to understand the air interface that will be modelled later.

FDMA: In Germany (and several other countries), four *ultra high frequency* (*UHF*, 0.3-3 GHz) bands are reserved for GSM [3, 26]:

- 890-915 MHz: D-Netz uplink
- 935-960 MHz: D-Netz downlink
- 1710-1758 MHz: E-Netz uplink
- 1805-1880 MHz: E-Netz downlink

The 1800 MHz bands were added later in 1994 [3]. The corresponding system is called DCS 1800, but it is essentially GSM [27].

The division of a frequency band into distinct bands for uplink and downlink is called *frequency division duplex* (*FDD*) [3] in opposite to *time division duplex* (*TDD*), where uplink and downlink use the same frequency but different TDMA time slots (see below).

Each frequency band is divided into 124 carrier frequencies with 200 kHz bandwidth [26]. The division of the whole base band into several carrier frequencies is called *frequency division multiplexing* (*FDM*). Because in mobile phone systems the multiplexing is used to handle the medium access, i.e. the sharing of the frequency resources among users, we call this method *frequency division multiple access* (*FDMA*) [3].

Theoretically, this would allow 2 x 124 full duplex communication channels system-wide, which is definitively not enough. Therefore, more multiplexing techniques are used.

TDMA: The digital encoding of the voice connections allows the transmission of the voice data in smaller, compressed packages. Thus, we do not need a continuous voice connection but only a continuous stream of voice packets with high frequency. This allows us to split a single carrier frequency into consecutive frames of eight *time slots*. Each of these time slots can be used by a different user. The merged time slots of a single user form the *physical channel* of this user. This concept is called *time division multiple access* (*TDMA*) and is demonstrated in Figure 1.12.

Frames are organised in a *frame hierarchy* (multi-frames, super-frames and hyper-frames), but this hierarchy is not in the focus of this thesis. See [27], [26] or [3] for details.

Now 2 x 992 full duplex communication channels are available system-wide.

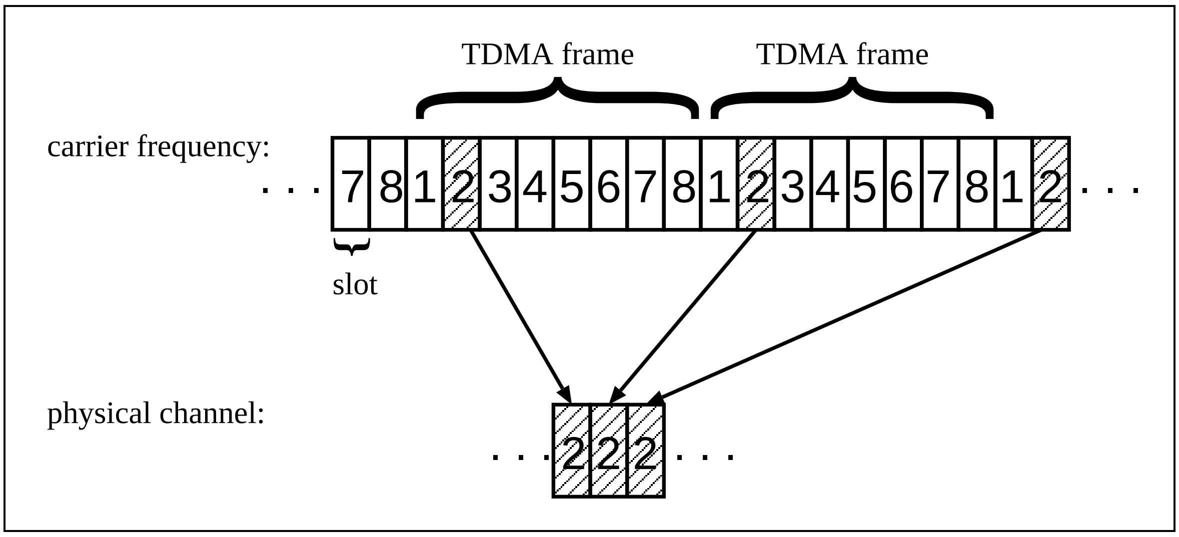

Figure 1.12: Time division multiple access.

Each carrier frequency can transmit 270883 bit/s. This is shared among eight users, so each user gets a gross data rate of about 33.9 kbit/s. After considering overhead and error correction, only 13 kbit/s are available for the voice connection over a single physical channel [26].

If the physical channel is used for a circuit switched data-over-GSM connection, which was introduced in autumn 1994, then only 9.6 kbit/s were available at first. Nowadays, data-over-GSM connections reach 14.4 kbit/s by using more efficient error correction methods, that were introduced in summer 2000 by the first German providers [3, 29].

SDMA: GSM also uses *space division multiple access* (*SDMA*). The whole area, that should be covered by the service, gets divided into small parts, called *cells* (*cell concept*). Each cell has its own *base transceiver station* (*BTS*), which is the fixed counterpart of the mobile station, the antenna of the service provider [3]. Every cell can now use a subset of all carrier frequencies. This subset can be reused in cells, that are far enough away to avoid mutual interference.

In Figure 1.13 28 GSM cells are shown that are using seven different sets of carrier frequencies (A-G). Adjacent cells may not use the same frequencies. Therefore, we can assemble seven cells to a so called *cluster*. In each cluster all frequencies can be used, providing that no frequency is used by different cells of one cluster.

The SDMA method is quite flexible, because not only the size of the cells but also the number of cells per cluster (i.e. the *cluster size*) can be changed according to the number of users in a certain area [3, 26].

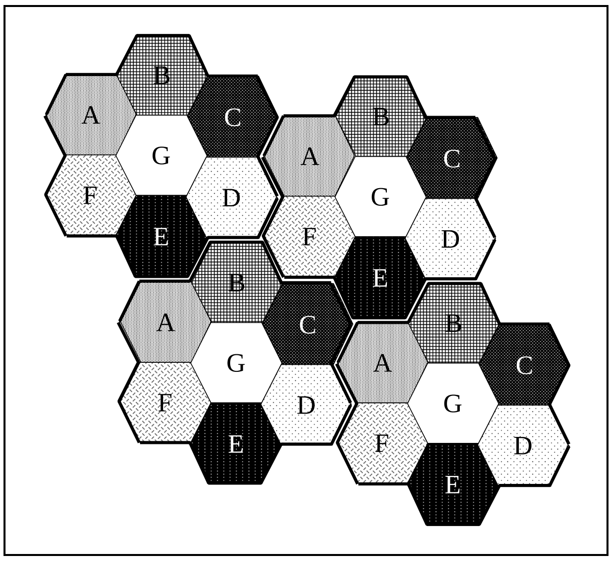

Figure 1.13: 28 GSM cells in 4 clusters of size 7.

1.6.3 2.5G: GSM with HSCSD, GPRS and EDGE

Several extensions were made to GSM to improve its versatility. The extended GSM is referred to as *Generation 2.5* (*2.5G*), because the new concepts enhance GSM with features planned for the third generation (→ Chap. 1.6.4, p. 32). Most extensions aim on higher efficiency and higher bit rate for data transmission.

HSCSD

The *high-speed circuit switched data* (*HSCSD*) concept was introduced by the first providers in 1999. With HSCSD a user is allowed to use up to four physical channels for his data connection, if available. Therefore, he is able to achieve a gross data rate of up to 57.6 kbit/s. Of course, the allocated channels can only be used by this user, who has to pay for it – even if the connection is idle. [29, 3]

GPRS

In contrast to the circuit switched data connections above, the *general packet radio service* (*GPRS*) uses packet switched data transfer. GPRS was like GSM standardised by ETSI. It was introduced in Germany in 2000 [28, 29].

With GPRS one or several *physical channels* can be used for data packet transmissions by one or several users. A physical channel, that is used for GPRS traffic is called *packet data channel* (*PDCH*) [28]. The number of physical channels that may be used by the GPRS system as PDCH is specified in the *partitioning scheme*.

scheme	remaining data rate
CS-1	9.05 kbit/s
CS-2	13.4 kbit/s
CS-3	15.5 kbit/s
CS-4	21.4 kbit/s

Table 1.1: GPRS coding schemes

Three different partition schemes can be used [28]:

- *Complete partitioning*: Some of the eight physical channels are dedicated to ordinary, circuit switched GSM connections (voice or data). The remaining channels are reserved for GPRS connections. This scheme is very inflexible, because unused channels of one service cannot be used by the other service.
- *Complete sharing*: Any unused physical channels may be allocated by a conventional GSM or by a GPRS connection. Normally GSM connections are given higher priority than GPRS connections, so PDCHs can be displaced by GSM connections.
- *Partial sharing*: This scheme is a mixture of complete partitioning and complete sharing. Some physical channels are reserved for conventional GSM, some for GPRS. The remaining channels may be used by both, whereas GSM connections again have higher priority.

Obviously, a single carrier frequency can provide a maximum of eight PDCHs. Even if a BTS provides several frequencies for its cell, a mobile station can only use up to eight PDCHs simultaneously, because the mobile stations are normally only able to handle a single carrier frequency for uplink and a single one for downlink.

The GPRS data rate a single PDCH will offer to the users cannot be easily specified. It depends on several factors [28, 31, 27]:

- The gross data rate of a single PDCH is 21.4 kbit/s. This includes the overheads for the GSM physical layer (synchronisation, guard times) and for the GPRS air interface.
- The *forward error correction* (*FEC*) channel coding is done dynamically depending on the current radio conditions. The four different GPRS *coding schemes* are shown in Table 1.1.
- The *automatic repeat request* (*ARQ*) protocol of GPRS consumes more bandwidth if necessary.
- More overhead is produced by higher and partly application specific protocol layers.

For this thesis we assume ideal radio conditions and therefore a data rate of 21.4 kbit/s per PDCH. We do not take higher protocol layers into account to provide results that are independent from certain applications. As mentioned before, a single user may be granted up to eight PDCHs for sending GPRS data. Therefore, the maximum data rate a single user

reliability class	lost SDU[1] probability	duplicate SDU probability	out of sequence SDU probability	corrupt SDU probability
1	10^{-9}	10^{-9}	10^{-9}	10^{-9}
2	10^{-4}	10^{-5}	10^{-5}	10^{-6}
3	10^{-2}	10^{-5}	10^{-5}	10^{-2}

Table 1.2: GPRS reliability classes

delay class	SDU size 128 byte		SDU size 1024 byte	
	mean	95 percentile	mean	95 percentile
1	$< 0.5s$	$< 1.5s$	$< 2s$	$< 7s$
2	$< 5s$	$< 25s$	$< 15s$	$< 75s$
3	$< 50s$	$< 250s$	$< 75s$	$< 375s$
4	unspecified			

Table 1.3: GPRS delay classes

can obtain is 171.2 kbit/s, but this situation will be very unlikely. Including higher layers, a user will in practice achieve a maximum data rate of about 43.2 kbit/s [34].

Each carrier frequency can only handle up to 32 GPRS data connections (*temporary block flows*, *TBFs*, sessions) simultaneously, because the TBFs are internally identified by a *TFI* (*temporary flow identity*) of 5 bit length [28].

Whether a connection request can be accepted is decided by a *call admission control* (*CAC*) [28] algorithm according to the partitioning scheme, *quality of service* (*QoS*) requirements and further criteria. The QoS requirements are defined when the session is set up and stored in a session specific parameter set called *packet data protocol* (*PDP*) [31]. They can be specified by the user regarding [3]:

- *service precedence* (high, medium, low)
- *user data throughput*
- *reliability class* (→ Table 1.2, p. 30)
- *delay class* (→ Table 1.3, p. 30)

Several possible modifications to GSM/GPRS were developed to improve its performance. Two of them, *EDGE* and *DOVE* will be described next.

[1]*SDU*: *service data unit*, i.e. single GPRS data packet.

EDGE

In 2004, 75 mobile network providers in 50 countries intend to introduce the *enhanced data (rates) for GSM/global evolution* (*EDGE*) technology [29, 32, 26, 33]. EDGE provides a new physical layer and can be introduced in two different ways [32]:

- To improve the circuit switched data transfer over GSM like HSCSD: *enhanced circuit-switched data* (*ECSD*).
- For improving GPRS: *enhanced GPRS* (*EGPRS*)

We will only have a brief look at the latter way in this thesis and will refer to it as EDGE.

EDGE provides data rates up to 384 kbit/s (theoretically up to 473.6 kbit/s on the physical layer) by introducing new modulation techniques, error-tolerant transmission methods and improved link adaption mechanisms [32]. But practically, including all overhead of higher layers, a user will see a maximum data rate of about 118.4 kbit/s (downlink) [33].

The main improvement of the data rates is done by changing the modulation technique [32, 26]: Conventional GPRS used *Gaussian minimum shift keying* (*GMSK*), which is a kind of *phase modulation*. Using GMSK each shift in the phase (symbol, baud) represents one bit. EDGE uses *8-phase shift keying* (*8PSK*). Here, every shift in phase can represent three bits. The data rate is therefore increased by factor three. The disadvantage is, that the "distance" between the single symbols is shorter, i.e. symbols might be misinterpreted more easily, especially in very poor radio conditions, where GMSK will show better performance. Therefore, EDGE provides nine different *modulated coding schemes* (*MCS*), the lower four (MCS1-MCS4) using GMSK, the upper five (MCS5-MCS9) using 8PSK. The data rates of the EDGE coding schemes MCS1-MCS4 are slightly higher than the data rates of the GPRS coding schemes CS1-CS4 because of differences in the header and payload sizes.

EDGE furthermore provides re-segmentation for retransmitting packets in a lower coding scheme if the transmission failed because of decreasing radio conditions. Therefore, the link-controlling algorithm, which chooses the coding scheme for the next sequence of data packets, can be more aggressive. Further enhancements like addressing window improvements, greater measurement accuracy and the improved interleaving procedure are described in [32].

DOVE

The *delay of voice end-user* (*DOVE*) [35, 36] is a theoretical concept to enhance the efficiency of existing GSM/(E)GPRS networks by modifying the *CAC* algorithm.

Consider a GSM/GPRS system with GSM voice and GPRS data service sharing the GSM TDMA slots obeying a *complete partitioning scheme* and imagine a high utilisation of the voice service. Then the GPRS data rates will decrease drastically, because the voice service has higher priority and displaces the data service. DOVE tries to decrease this effect by delaying an arriving voice connection request from taking away the last PDCH from GPRS. The delay will be short enough, so that it will be accepted by the user.

Nevertheless, this short delay will have two effects:

- In the meantime the data service is able to operate and the data rate will therefore not decrease that fast.
- Another voice connection may terminate and the released resources can be used by the waiting voice connection instead of displacing the PDCH.

The quantitative impact of this concept on the data rate will be discussed in Chapter 2.1.2.

1.6.4 3G: UMTS

The *Universal Mobile Telecommunications System* (*UMTS*) is another name for the *third generation* (*3G*) technology *wide-band CDMA* (*W-CDMA*). The W-CDMA technology was suggested by Ericsson as a possible implementation of the first 3G considerations called *IMT*[3]*-2000* made by the *International Telecommunication Union* (*ITU*) in 1992 [26]. W-CDMA/UMTS could stand up to the competitor *CDMA2000*, suggested by Qualcom [29, 26]. UMTS gets standardised by the *Third Generation Partnership Project* (*3GPP*), which was founded in 1998 and consists of several "organisational partners" including ETSI [29, 38].

Nowadays, the first mobile networks "support UMTS", the first UMTS mobile phones are sold and the UMTS specification still enjoys frequent releases [39, 40]. Beginning with "Release 99" [37] they aim at introducing the third generation of mobile systems by partly reusing GSM infrastructure and partly building new technology from scratch.

One of the first improvements implemented was the third generation air interface called *UMTS Terrestrial Radio Access Network* (*UTRAN*). As mentioned above, UMTS/UTRAN uses W-CDMA as medium access method. W-CDMA is a variant of CDMA with a bandwidth of 5 MHz. The CDMA method will be explained in the next section:

CDMA: Code Division Multiple Access

The *code division multiple access* (*CDMA*) is yet another multiplexing technique for medium access that was developed by Qualcom. CDMA is not used together with but instead of FDMA and TDMA (except "some" FDMA for the cell concept). Therefore in CDMA, every mobile station sends and receives at the same time on the same frequency band. The separation of information is done by different, mobile station specific coding of the logical bits by using several so called *chip sequences*, which are vectors of bipolar bits, i.e. the logical 0 is represented by -1 and the logical 1 is represented by +1. The chip sequence of all mobile stations are unique (for each frequency band) and orthogonal to each other. Each bit a mobile station wants to send or receive gets represented by the chip sequence (in case of a logical 1) or its negation (in case of a logical 0). If several senders transmit data at the same time, their transmissions get superposed. The receiver can restore the information of a specific sender by using the senders chip sequence.

[3]International Mobile Telecommunications

Example: Imagine two senders: S_1 and S_2. The chip sequence of sender S_1 is $\mathbf{C_1} = (+1, +1, -1, -1)$. The chip sequence of sender S_2 is $\mathbf{C_2} = (-1, +1, -1, +1)$. Sender S_1 wants to transmit the information $B_1 = 1$ and S_2 wants to send information $B_2 = 0$. The information B_1 is encoded by sender S_1 using its chip sequence $\mathbf{C_1}$, which results to the encoded information $\mathbf{E_1} = \mathbf{C_1} = (+1, +1, -1, -1)$. Sender S_2 encodes B_2 by using chip sequence $\mathbf{C_2}$, which results to $\mathbf{E_2} = \overline{\mathbf{C_2}} = (+1, -1, +1, -1)$. The encoded information $\mathbf{E_1}$ and $\mathbf{E_2}$ will then be sent by S_1 and S_2, respectively. A possible receiver R_3 will pick up the overlay of both transmissions: $\mathbf{E_3} = \mathbf{E_1} + \mathbf{E_2} = (+1, +1, -1, -1) + (+1, -1, +1, -1) = (+2, 0, 0, -2)$. If the receiver R_3 wants to decode $\mathbf{E_3}$ to get the information sent by S_1, he calculates the normalised inner product $\mathbf{E_3} \bullet \mathbf{C_1} = \frac{(+2)\cdot(+1)}{4} + \frac{(0)\cdot(+1)}{4} + \frac{(0)\cdot(-1)}{4} + \frac{(-2)\cdot(-1)}{4} = 0.5 + 0 + 0 + 0.5 = +1$ The result tells R_3 that S_1 sent information $B_1 = 1$. If R_3 wants to get the information sent by S_2 he uses $\mathbf{C_2}$ instead of $\mathbf{C_1}$: $\mathbf{E_3} \bullet \mathbf{C_2} = \frac{(+2)\cdot(-1)}{4} + \frac{(0)\cdot(+1)}{4} + \frac{(0)\cdot(-1)}{4} + \frac{(-2)\cdot(+1)}{4} = -0.5 + 0 - 0 - 0.5 = -1 \Rightarrow S_2$ sent information $B_2 = 0$.

More detailed information about the CDMA algorithm can be found in [27], [26] or [3].

The number of users CDMA can handle depends on the number of chips per chip sequence, but the higher the number of chips per sequence the more chips have to be send for transmitting one bit. This more of information can only be handled, if the available bandwidth is increased by the factor chips per sequence (*spread spectrum* method).

Using W-CDMA a maximum *chip rate* of 3,84 Mchip/s can be achieved, theoretically. This allows bit-rates up to 2Mbit/s, under good radio conditions and presuming a non-moving "mobile" station located in a so called "Hot Spot" [29, 37].

The description and rating of further UMTS features would go far beyond the scope of this thesis. We refer the interested reader to [3], [37] and [40].

Chapter 2

Modelling and Evaluation of Mobile Systems with MOSEL-2

We will now use MOSEL-2 to examine the air interface (downlink) of a single GSM/GPRS (→ Chap. 1.6.3, p. 28) cell which can be considered as the "bottleneck" of the GSM/GPRS system. Before building the MOSEL-2 model description, we will abstract from the real system by building conceptual models using Petri nets (→ Chap. 1.3.3, p. 11).

2.1 Markovian Approach

We will only use Markovian distributions (→ Chap. 1.3.1, p. 4) for our models in the beginning. This will keep the models small and concise. The models can then be represented as *GSPNs* (→ Chap. 1.3.3, p. 11).

2.1.1 GSM System

We consider a pure GSM network as introduced before (→ Chap. 1.6.2, p. 26) and observe the behaviour of a single GSM cell with a single carrier frequency, which results in a maximum of eight available TDMA channels. Later we could easily extend that model to take into account additional carrier frequencies and/or handovers from adjacent cells. Anyway, we will not take handovers and retrials into account in this thesis. The interested reader is referred to [28], [31], [41] and [42].

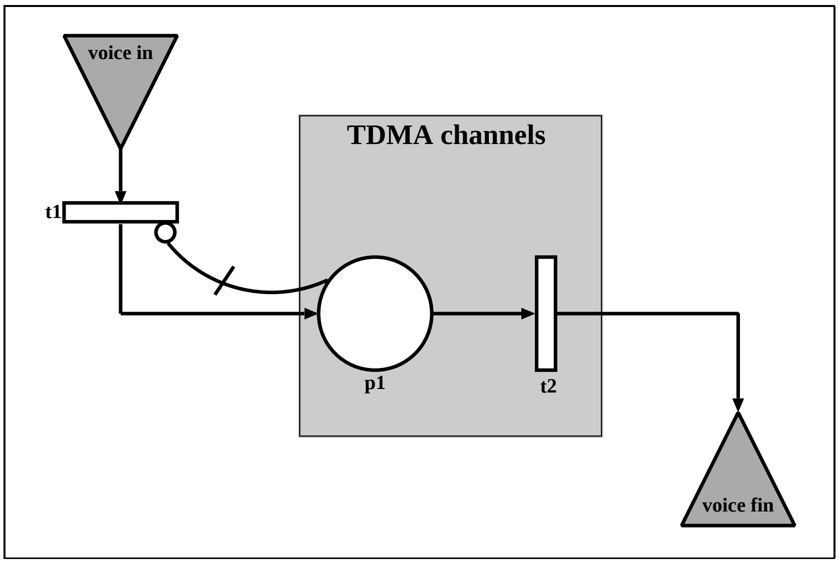

Figure 2.1: Conceptual GSPN model of GSM air interface.

Conceptual Model

A possible conceptual representation of the abstracted air interface of a single GSM cell by a GSPN is shown in Figure 2.1:

- Arrival process: New voice connection requests (voice calls or data-over-GSM) arrive according to a Poisson process. This is represented by transition **t1** which has an exponentially distributed firing rate (0.005 . . . 0.015). We assume a high and therefore approximately constant number of potential users per cell. Therefore, similar to [41] and [42], we do not take into account the state dependency of the arrival process, i.e. we do not consider the dependency of the arrival rate from the (quite constant) number of idle users in the cell at is it done for example in [28].

- Unless all eight TDMA channels are occupied, the requested voice connection gets established. Each voice connection uses a single TDMA channel. Voice connections in service are equivalent to tokens located in place **p1**, which has a maximum capacity of eight. If the capacity is reached, **t1** gets disabled by the inhibitor arc with multiplicity eight. In this case, new connection requests are ignored and get lost.

- Service process: As in several related publications [41, 28, 31], we assume independent service processes with exponentially distributed call holding times with a mean duration of 100 seconds and therefore a mean service rate for transition **t2** of 1/100 per active connection (*infinite server*). Therefore, the service rate of **t2** is state-dependent: it depends on the current number of tokens in place **p1**.

The underlying Markov Chain

The simple GSM system can be seen as loss system with finite capacity $C = 8$. Its state space can be described by a *stochastic process* $\mathbf{X}(t) = \mathbf{C}(t)$, where $\mathbf{C}(\mathbf{t})$ is the number of active connections at time t. Because of the *memoryless property* of all state changes, the stochastic process can be described as a *CTMC* (→ Chap. 1.3.1, p. 4). The *CTMC* can then be solved as described in Chapter 1.4.3. This can be done automatically and therefore very comfortably by using MOSEL-2 (→ Chap. 1.5, p. 19), especially when the models are growing more complex later.

The MOSEL-2 Model

We will now describe the GSPN model of the simple GSM system with the help of the MOSEL-2 modelling language:

```
/** CONSTANT AND PARAMETER PART *********************************************/
// user behaviour
PARAMETER v_ar := 0.005 .. 0.015 STEP 0.002;  /* mean voice call arrival rate */
CONST     v_ht := 100;                        /* mean voice call holding time */
CONST     v_sr := 1/v_ht;                     /* mean voice call service rate */

// network configuration
CONST     freq_c  := 1;                        /* carrier frequencies in cell */
CONST     chans_f := 8;                        /* physical channels per freq. */
CONST     chans   := freq_c * chans_f;           /* overall numb. of channels */

/** NODE PART ***************************************************************/
NODE p1[chans] := 0;                                                  /* p1 */

/** RULE PART ***************************************************************/
FROM EXTERN TO p1 RATE v_ar;                                          /* t1 */
FROM p1 TO EXTERN RATE v_sr * p1;                                     /* t2 */

```

Note, that we do not have to specify the inhibitor arc explicitly in MOSEL-2. The capacity of place **p1** is well defined and cannot be exceeded.

In addition to the model architecture and behaviour, MOSEL-2 allows to specify the desired result measures. We are mainly interested in the probability of unsatisfied connection requests:

```
/** RESULT PART *************************************************************/
PRINT v_loss := PROB(p1==chans);               /* voice call loss probability */

/** PICTURE PART ************************************************************/
PICTURE "voice call loss probability"
PARAMETER v_ar
CURVE v_loss
XLABEL "voice call arrival rate"
YLABEL "voice call loss probability"
```

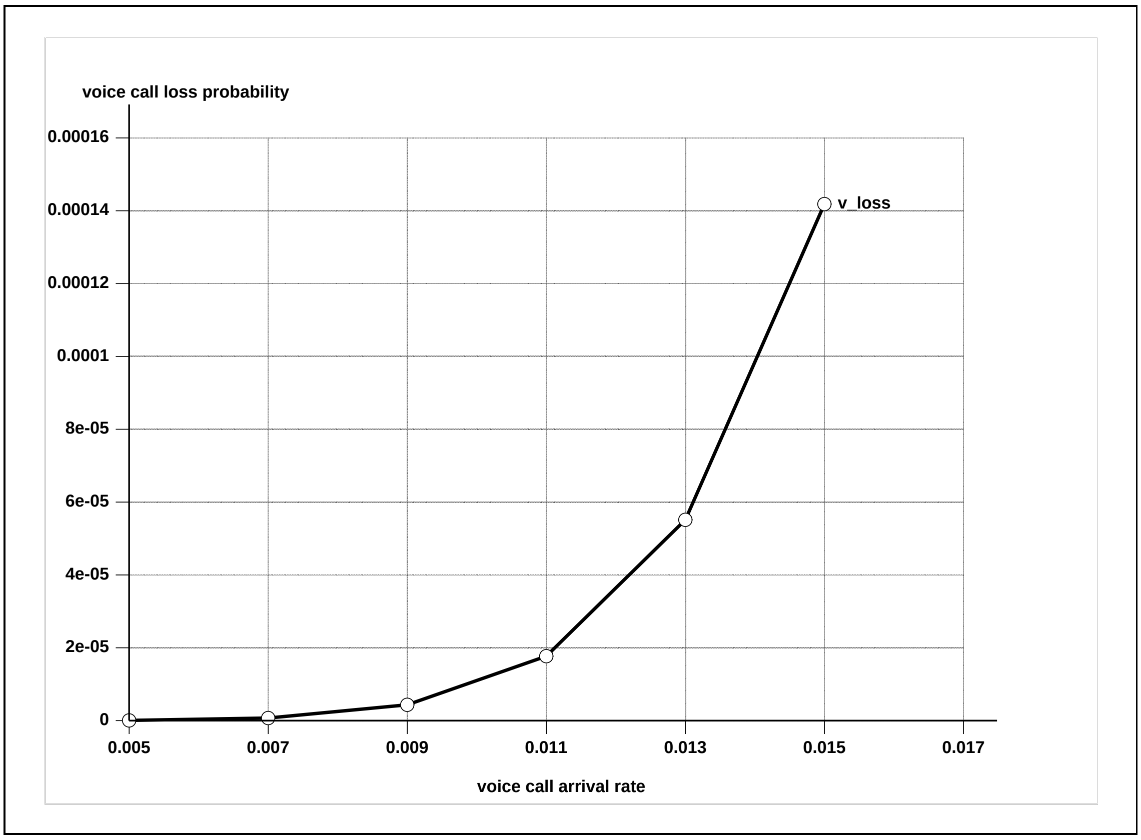

Figure 2.2: GSM: voice call loss probability versus voice call arrival rate.

Model Evaluation

We are using the MOSEL-2 evaluation environment to translate the MOSEL-2 model into the *c-based stochastic Petri net language* (*CSPL*), which acts as input for *SPNP* [10]. SPNP then transforms the CSPL models into a CTMC and performs the steady-state analysis. Finally, the results of SPNP get parsed by MOSEL-2 and are presented in a tool-independent format.

Performance of the Evaluation Methods: To get a feeling for the time exposure of different evaluation methods and model sizes, we provide information about the duration of every model presented and evaluated in this thesis from now on. The duration measures where obtained using the Linux program *time(1)*. All models were evaluated using a dual i686 PC with i386-Linux (kernel 2.4.23), two *Intel*® *XEON*™ CPUs (2×2.2 GHz) and 2 Gbyte RAM.

The model above was evaluated within 0.680s/0.270s/0.940s (user/system/real).

Results

The graphical results of our GSM evaluation are shown in Figure 2.2.

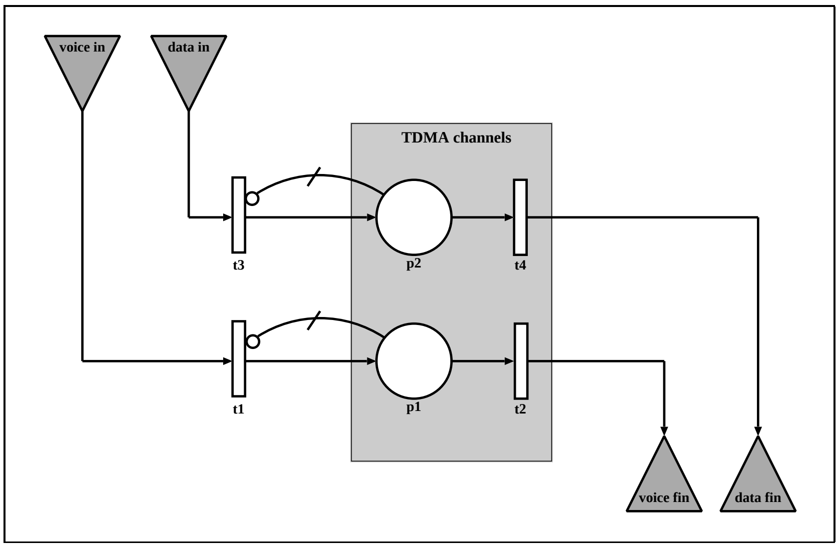

Figure 2.3: Conceptual GSPN model of GSM/GPRS system.

2.1.2 GPRS System

We will now extend the GSM model introduced above by adding GPRS.

Conceptual Model

The conceptual model is shown in Figure 2.3.

In addition to the GSM model we now have:

- Burst arrival process: We model the GPRS data service on burst level. The data bursts arrive according to a Poisson process modelled by transition **t3**, which has exponentially distributed firing rate $(0.1 \ldots 2.5)$.
- An overall number of 32 bursts per carrier frequency can be accepted for being serviced in place **p2** (→ Chap. 1.6.3, p. 28). Furthermore, PDCHs can be preempted by arriving GSM calls, which have higher priority. One TDMA channel is reserved to GPRS, which leads to a partial sharing scheme (→ Chap. 1.6.3, p. 28) without reservations for voice calls. Therefore, the capacity of place **p1** decreases by one from eight to seven.
- Burst service process: Similar to [28] we assume, that the burst size is geometrically distributed with a mean burst size of 10 kbyte. Furthermore, we assume good radio conditions (CS-4) and therefore a data rate of 21.4 kbit/s per PDCH. The burst rate of a single PDCH is then equal to $\frac{21.4}{8 \cdot 10}$. The burst rate of all PDCHs is divided among all GPRS users and modelled by **t4**, which is comparable to *processor sharing* (*PS*).
- The simultaneous use of GSM and GPRS within the same pool of TDMA channels (cell) requires the use of two channels for signalling [35, 28]. So we can only use up to five channels for GSM voice connections.

MOSEL-2 Model

```
/** CONSTANT AND PARAMETER PART *********************************************/
// GSM user behaviour
PARAMETER v_ar := 0.005 .. 0.015 STEP 0.002;  /* mean voice call arrival rate */
CONST     v_ht := 100;                        /* mean voice call holding time */
CONST     v_sr := 1/v_ht;                     /* mean voice call service rate */

// GPRS data behaviour
PARAMETER d_ar := 0.1, 1.5, 2.5;              /* mean data burst arrival rate */
CONST     d_sz := 10 * 8;                      /* mean data burst size (kbit) */

// network configuration
CONST     freq_c    := 1;                      /* carrier frequencies in cell */
CONST     chans_f   := 8;                      /* physical channels per freq. */
CONST     chans     := freq_c*chans_f;            /* overall numb. of channels */
CONST     chans_sig := 2;                     /* channels used for signalling */
CONST     chans_use := chans-chans_sig;       /* channels available for users */
CONST     chans_gpr := 1;                       /* channels reserved for GPRS */
CONST     chans_gsm := chans_use - chans_gpr; /* max. numb. of voice channels */
CONST     gprs_max  := 32 * freq_c;           /* maximum number of GPRS users */
CONST     pdch_dr   := 21.4;                   /* data rate per PDCH (kbit/s) */
CONST     burst_sr  := pdch_dr/d_sz;           /* burst service rate per PDCH */

/** NODE PART ***************************************************************/
NODE p1[chans_gsm]      := 0;                                        /* p1 */
NODE p2[gprs_max]       := 0;                                        /* p2 */

/** RULE PART ***************************************************************/
FROM EXTERN TO p1 RATE v_ar;                                          /* t1 */
FROM p1 TO EXTERN RATE v_sr * p1;                                     /* t2 */
FROM EXTERN TO p2 RATE d_ar;                                          /* t3 */
FROM p2 TO EXTERN RATE burst_sr * (chans_use - p1);                   /* t4 */

/** RESULT PART *************************************************************/
PRINT v_loss := PROB(p1==chans_gsm);           /* voice call loss probability */
PRINT d_loss := PROB(p2==gprs_max);            /* data burst loss probability */

/** PICTURE PART ************************************************************/
PICTURE "call/burst loss probability"
PARAMETER v_ar
CURVE v_loss
CURVE d_loss
XLABEL "voice call arrival rate"
YLABEL "call/burst loss probability"
```

Model Evaluation

The model evaluation using MOSEL-2 with SPNP and steady-state numerical analysis took 2.600s/0.630s/3.210s (user/system/real).

Results

The graphical results of our GSM/GPRS model are shown in Figure 2.4. We can see, that the voice call loss probability is much higher compared to a pure GSM system (→ Fig. 2.2, p. 38), which is obvious, because three channels are now reserved for signalling and GPRS. However, the utilisation of the GPRS service has no influence on the GSM service. In contrast, the dependency of the data service from the voice utilisation is evident.

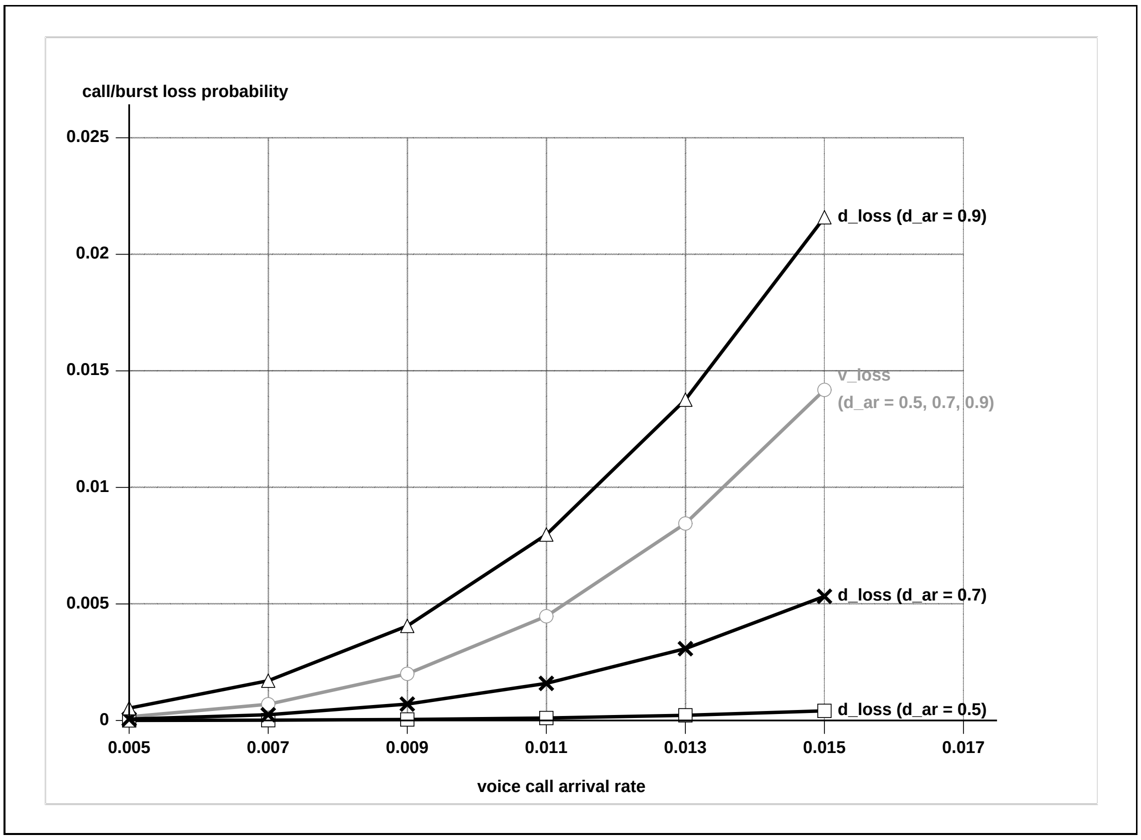

Figure 2.4: GPRS: voice call and data burst loss probability versus voice call arrival rate.

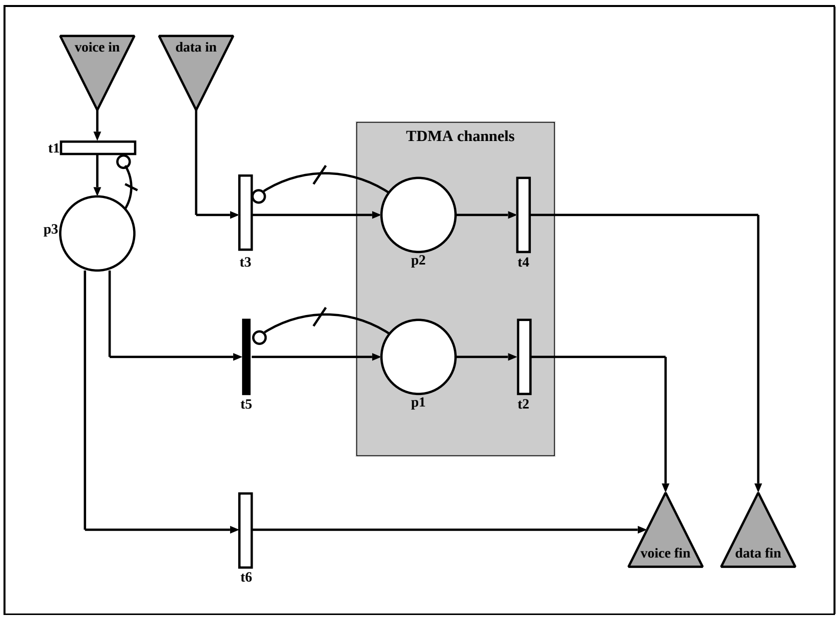

Figure 2.5: Conceptual GSPN model of GSM/GPRS system with DeTVoC.

2.1.3 Refining the GPRS Model: DeTVoC

Until now, we assumed that GSM voice connection requests that cannot be served immediately are lost an once. But in reality the user will at least wait a few seconds, hoping that he still might be served. We call this behaviour *delay tolerant voice call* (*DeTVoC*).

Conceptual Model

DeTVoC can be modelled as shown in Figure 2.5.

Additional elements:

- Arriving voice connection requests (**t1**) are now transfered to place **p3**, where they can wait for being served. Place **p3** has a high but finite capacity (e.g. 100) for keeping the state space finite and tractable. Calls arriving at **p3** can be assigned to a free TDMA channel immediately by transition **t5**, if available. Otherwise, the call has to wait in place **p3**.
- A user will not wait endlessly for the acceptance of his connection request. The maximum time a user will wait is modelled by the (infinite server) transition **t6**, which has an exponentially distributed firing rate to keep the model Markovian.
- We can now differentiate between the call loss probability, i.e. the probability that a call does not get served at all, and the call blocking probability, i.e. the probability that a call cannot be served immediately.

- The voice call loss probability P_{loss} (`v_loss`) gets calculated as:

$$P_{loss} = 1 - P_{served} = 1 - \frac{\lambda_S}{\lambda_I} \tag{2.1}$$

  with:

  - λ_S: rate of served connection requests (**t2**)
  - λ_I: rate of incoming calls (arrival process **t1**)

  This calculation requires that place **p3** does not reach its capacity.

- The call blocking probability equals the probability that all TDMA channels are occupied.

- We use the mean waiting time ($\frac{1}{\mu_{t6}}$) as parameter from (almost) 0 to 15 seconds. A waiting time of 0 seconds would be equivalent to the system without DeTVoC. The call blocking probability then equals the call loss probability.

- The immediate transition **t5** has higher priority than the timed transition **t6**.

MOSEL-2 Model

Besides the model architecture, we change the set of parameters and the result measures to check the influence of DeTVoC on the system behaviour. Note, that immediate transitions always have higher priority than timed transitions. Therefore, we do not have to specify priorities for **t5** and **t6** explicitly.

```
/** CONSTANT AND PARAMETER PART ***********************************************/
// GSM user behaviour
PARAMETER v_ar := 0.005 .. 0.015 STEP 0.002;  /* mean voice call arrival rate */
CONST     v_ht := 100;                        /* mean voice call holding time */
CONST     v_sr := 1/v_ht;                     /* mean voice call service rate */
PARAMETER v_dt := 1E-8, 7, 15;                /* mean voice call waiting time */
CONST     v_dr := 1/v_dt;                    /* mean voice call aborting rate */

// GPRS data behaviour
CONST     d_ar := 0.7;                        /* mean data burst arrival rate */
CONST     d_sz := 10 * 8;                      /* mean data burst size (kbit) */

// network configuration
CONST     freq_c    := 1;                      /* carrier frequencies in cell */
CONST     chans_f   := 8;                      /* physical channels per freq. */
CONST     chans     := freq_c*chans_f;           /* overall numb. of channels */
CONST     chans_sig := 2;                     /* channels used for signalling */
CONST     chans_use := chans-chans_sig;       /* channels available for users */
CONST     chans_gpr := 1;                       /* channels reserved for GPRS */
CONST     chans_gsm := chans_use - chans_gpr; /* max. numb. of voice channels */
CONST     gprs_max  := 32 * freq_c;           /* maximum number of GPRS users */
CONST     pdch_dr   := 21.4;                   /* data rate per PDCH (kbit/s) */
CONST     burst_sr  := pdch_dr/d_sz;           /* burst service rate per PDCH */

/** NODE PART *****************************************************************/
NODE p1[chans_gsm]     := 0;                                            /* p1 */
NODE p2[gprs_max]      := 0;                                            /* p2 */
NODE p3[100]           := 0;                                            /* p3 */

/** RULE PART *****************************************************************/
FROM EXTERN TO p3 RATE v_ar;                                            /* t1 */
FROM p1 TO EXTERN RATE v_sr * p1;                                       /* t2 */
FROM EXTERN TO p2 RATE d_ar;                                            /* t3 */
FROM p2 TO EXTERN RATE burst_sr * (chans_use - p1);                     /* t4 */
FROM p3 TO p1;                                                          /* t5 */
FROM p3 TO EXTERN RATE v_dr * p3;                                       /* t6 */

/** RESULT PART ***************************************************************/
PRINT v_loss  := 1 - ((v_sr * MEAN(p1)) / v_ar);     /* voice call loss prob. */
PRINT v_block := PROB(p1==chans_gsm);      /* voice call blocking probability */
PRINT d_loss  := PROB(p2==gprs_max);           /* data burst loss probability */
PRINT p3_full := PROB(p3==100);               /* probability of full place p3 */

/** PICTURE PART **************************************************************/
PICTURE "call blocking, call loss and burst loss probability"
PARAMETER v_ar
CURVE v_block
CURVE d_loss
CURVE v_loss
XLABEL "voice call arrival rate"
YLABEL "call blocking, call loss and burst loss probability"
```

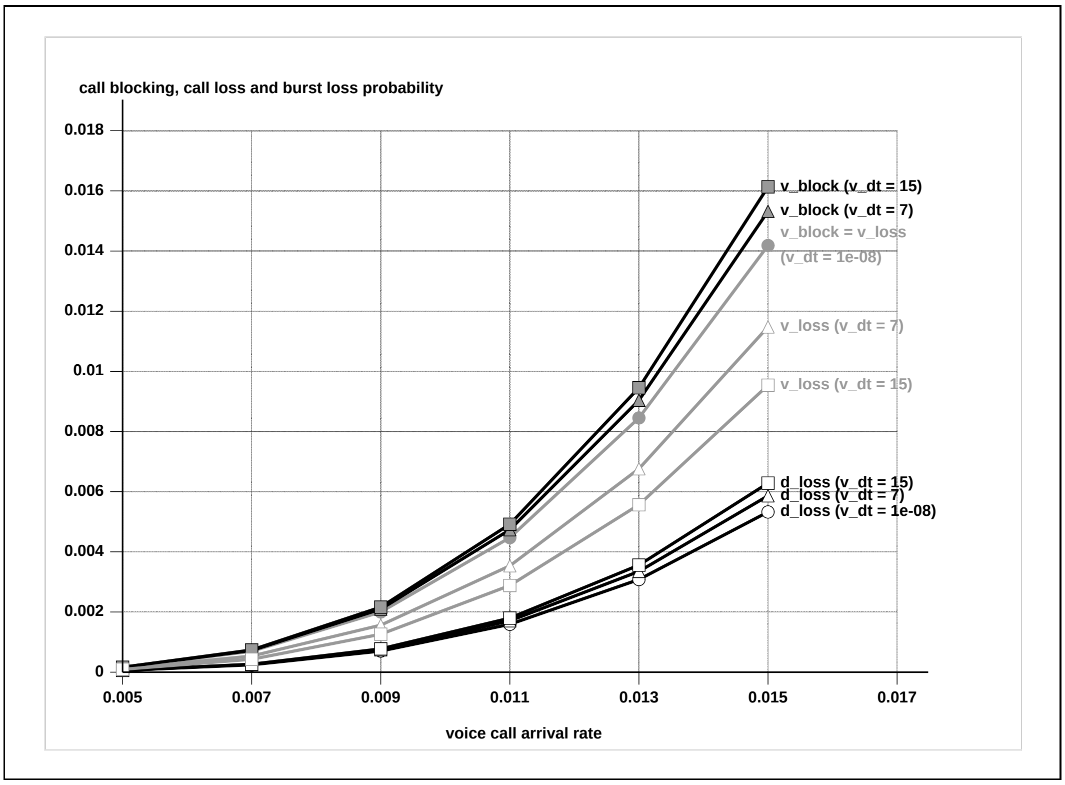

Figure 2.6: GPRS/DeTVoC: blocking of voice calls and loss of calls and data bursts.

Results

The results were obtained within 9.030s/0.650s/9.690s (user/system/real). They were produced using MOSEL2 with SPNP's steady-state analysis and are shown in Figure 2.6.

As predicted, the voice call loss probability approaches the voice call blocking probability for very short user waiting times. The reduction of the voice call loss probability with increasing user waiting time can be seen clearly. The voice call blocking probability increases with longer waiting times because of a higher utilisation of the GSM channels. A higher utilisation of GSM also leads to a higher data burst loss probability.

The textual output file provides further information about the applicability of the assumption made before: The probability of place **p3** reaching its capacity in this configuration is very low ($< 10^{-30}$) and can definitely be assumed to be zero.

2.1.4 Improving CAC Algorithm: DOVE

We could see in Figure 2.6, that despite DeTVoC the probability of unserved voice connection requests is still higher than the probability of lost data bursts. We assume, that voice call loss is much more undesirable than data loss, because data loss may be automatically repaired by ARQ methods. Nevertheless, the goal is of course to balance the probability of lost voice calls and lost data bursts.

A possible workaround would be to cancel the reservation of the one TDMA channel for the GPRS system, which results in a complete sharing scheme.

MOSEL-2 Model with Complete Sharing Scheme

This is done in a slightly modified MOSEL-2 model. Note, that data burst loss can now have two disjunctive causes:

1. The available PDCHs are serving the maximum of 32 GPRS users already.
2. There is no TDMA channel available for setting up a PDCH.

```
[...]
CONST     chans_gpr := 0;                         /* channels reserved for GPRS */
[...]
PRINT d_loss  := PROB(p2==gprs_max) + PROB(p1==chans_use); /* data burst loss */
[...]
```

Results of Model with Complete Sharing Scheme

The results were obtained using MOSEL-2 and SPNP's steady-state analysis within 10.460s/0.700s/11.090s (user/system/real). They are presented in Figure 2.7.

We can see, that now the situation is just the other way round. Unfortunately, the partitioning schemes do not provide finer granularity than single TDMA channels. Therefore, we have to find another way to get a compromise between the last two approaches.

Conceptual Model with DOVE

One method to solve this problem is the introduction of DOVE (→ Chap. 1.6.3, p. 31). Remember, that the main principle of this method is to delay the voice connection, that will displace the last PDCH in a complete sharing scheme. This will have two effects:

1. Data bursts can still be served, as long the voice user is delayed. Therefore, the data burst loss probability will not rise that fast.
2. The last PDCH may not be displaced, because another TDMA channel might be available for the voice user in the meantime.

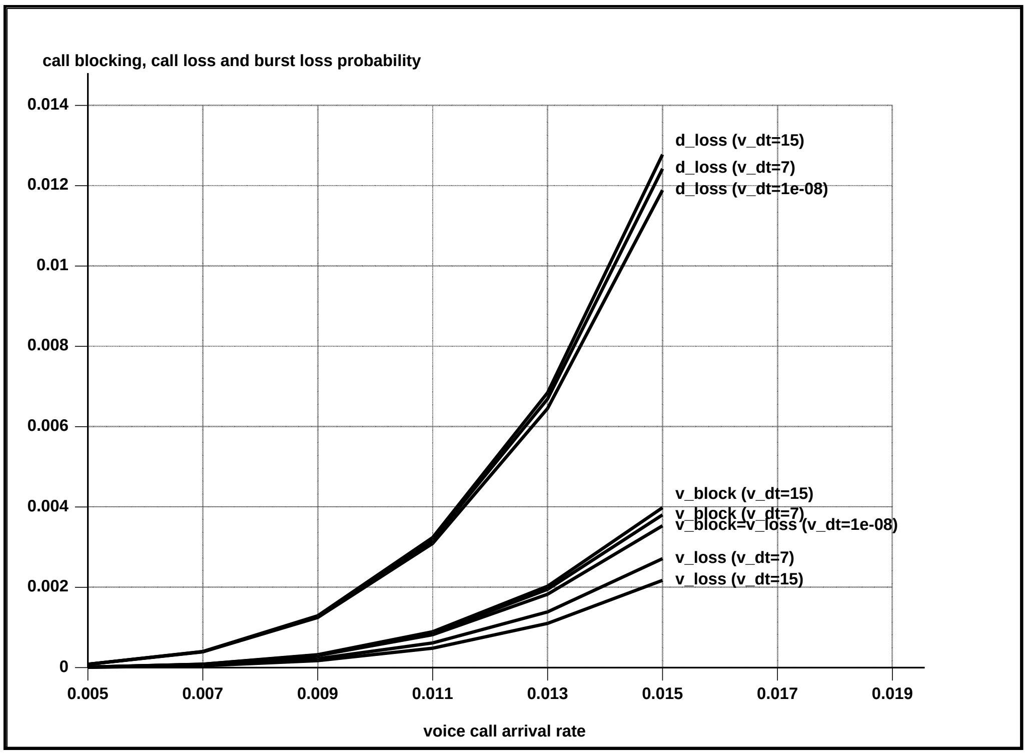

Figure 2.7: Results of GPRS/DeTVoC system with complete sharing scheme.

DOVE allows us to fine-tune the balance between data burst and voice call loss by changing the delay a voice user has to wait before preempting the last PDCH. The longer we choose this delay, the more the system behaves like a system with partial sharing scheme and a reserved channel for GPRS. The shorter we choose the delay, the more the system behaves like a system with complete sharing scheme.

The conceptual model is shown in Figure 2.8 and comprises the following changes:

- The inhibitor arc of the immediate transition **t5** now again disables the transition when only one PDCH is left (similar to partial sharing scheme).
- The timed transition **t7** instead allows preempting the last PDCH by an arriving voice connection request after an exponentially distributed waiting time. This transition will be modelled as infinite server, because every call will have to be delayed by the same time, regardless of the number of already waiting calls.
- For the model evaluation we will take the mean DOVE-delay as parameter between (almost) 0 and 25 seconds.

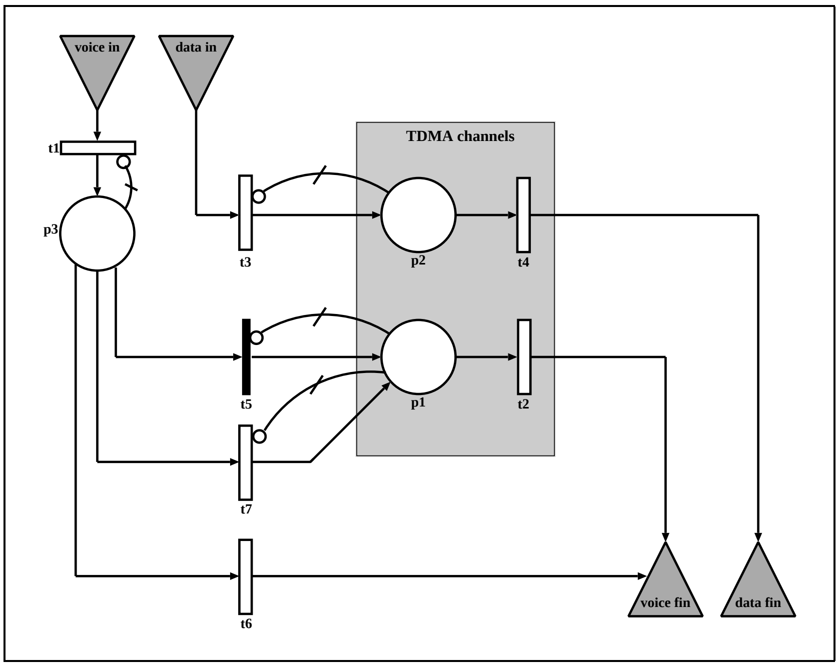

Figure 2.8: Conceptual GSPN model of GSM/GPRS system with DOVE.

MOSEL-2 Model with DOVE

For concise results, we will now keep the mean waiting time of waiting users (DeTVoC) constant at 7 seconds. Instead, we will take the DOVE delay as parameter. Furthermore, we will only take the data burst loss and the voice call loss probability into account. Including the introduction of DOVE, the MOSEL-2 model now looks like:

```
/** CONSTANT AND PARAMETER PART ********************************************/
// GSM user behaviour
PARAMETER v_ar := 0.005 .. 0.015 STEP 0.002;  /* mean voice call arrival rate */
CONST     v_ht := 100;                        /* mean voice call holding time */
CONST     v_sr := 1/v_ht;                     /* mean voice call service rate */
CONST     v_dt := 7;                          /* mean voice call waiting time */
CONST     v_dr := 1/v_dt;                    /* mean voice call aborting rate */

// GPRS data behaviour
CONST     d_ar := 0.7;                        /* mean data burst arrival rate */
CONST     d_sz := 10 * 8;                      /* mean data burst size (kbit) */

// network configuration
CONST     freq_c    := 1;                      /* carrier frequencies in cell */
CONST     chans_f   := 8;                      /* physical channels per freq. */
CONST     chans     := freq_c*chans_f;           /* overall numb. of channels */
CONST     chans_sig := 2;                     /* channels used for signalling */
CONST     chans_use := chans-chans_sig;       /* channels available for users */
CONST     chans_gpr := 0;                       /* channels reserved for GPRS */
CONST     chans_gsm := chans_use - chans_gpr; /* max. numb. of voice channels */
CONST     gprs_max  := 32 * freq_c;           /* maximum number of GPRS users */
CONST     pdch_dr   := 21.4;                   /* data rate per PDCH (kbit/s) */
CONST     burst_sr  := pdch_dr/d_sz;           /* burst service rate per PDCH */
PARAMETER dove_dt   := 1E-8, 7, 25;                     /* delay time for DOVE */
CONST     dove_sr   := 1/dove_dt;                         /* DOVE service rate */
CONST     dove_res  := 1;  /* number of PDCHs saved from immediate preemption */
CONST     dove_vch  := chans_use - dove_res;    /* always available for voice */

/** NODE PART ***************************************************************/
NODE p1[chans_gsm]     := 0;                                           /* p1 */
NODE p2[gprs_max]      := 0;                                           /* p2 */
NODE p3[100]           := 0;                                           /* p3 */

/** RULE PART ***************************************************************/
FROM EXTERN TO p3 RATE v_ar;                                           /* t1 */
FROM p1 TO EXTERN RATE v_sr * p1;                                      /* t2 */
FROM EXTERN TO p2 RATE d_ar;                                           /* t3 */
FROM p2 TO EXTERN RATE burst_sr * (chans_use - p1);                    /* t4 */
IF (p1 < dove_vch) FROM p3 TO p1;                                      /* t5 */
FROM p3 TO EXTERN RATE v_dr * p3;                                      /* t6 */
FROM p3 TO p1 RATE dove_sr * p3;                                       /* t7 */

/** RESULT PART *************************************************************/
PRINT v_loss  := 1 - ((v_sr * MEAN(p1)) / v_ar);     /* voice call loss prob. */
PRINT d_loss  := PROB(p2==gprs_max) + PROB(p1==chans_use); /* data burst loss */

/** PICTURE PART ************************************************************/
PICTURE "call and burst loss probability"
PARAMETER v_ar
CURVE d_loss
CURVE v_loss
XLABEL "voice call arrival rate"
YLABEL "call and burst loss probability"
```

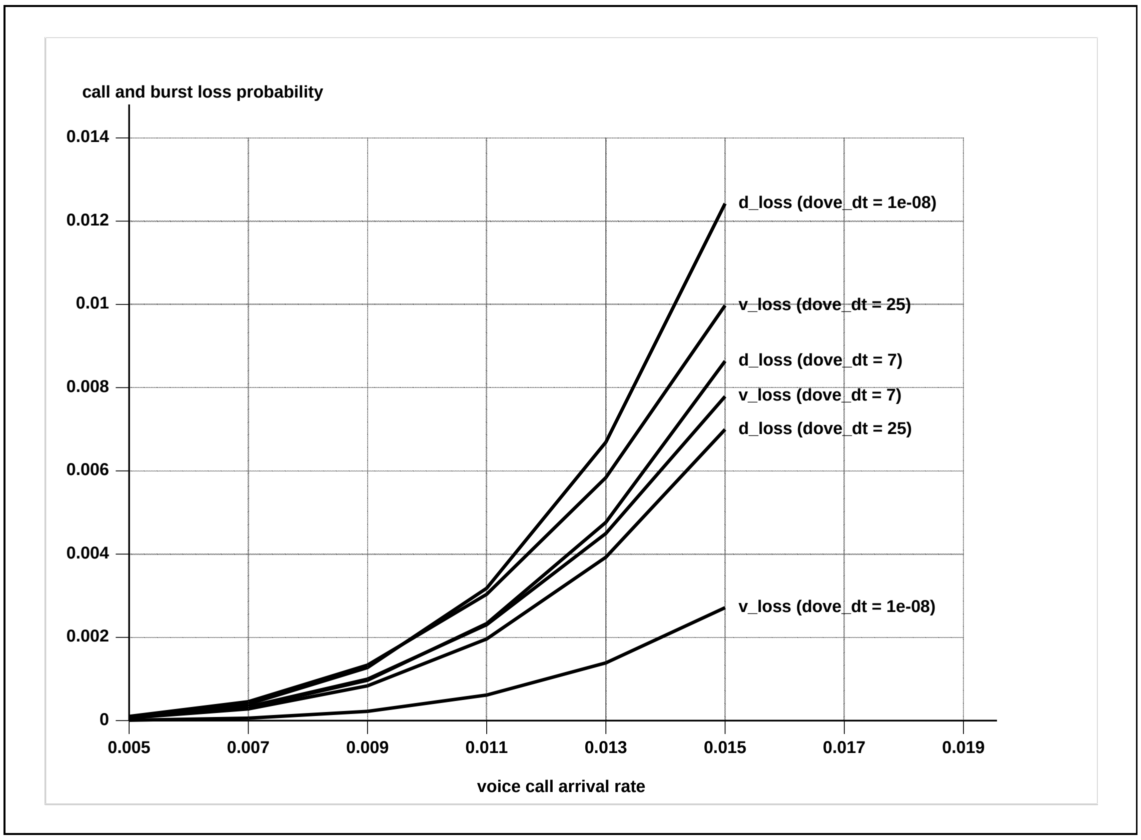

Figure 2.9: GPRS/DeTVoC with DOVE: voice call and data loss probability.

Results of Model with DOVE

Again, we are using MOSEL-2 and steady-state analysis provided by SPNP. The results were obtained within 22.400s/0.780s/23.180s (user/system/real) and are shown in Figure 2.9.

We can see, that for short DOVE delay times the results approach the behaviour of a complete sharing scheme (→ Fig. 2.7, p. 47). On the other hand, the results approach the behaviour of a partial sharing scheme (→ Fig. 2.6, p. 45) for long delay times.

Medium delays can be regarded as good compromise between both sharing schemes. Furthermore, the delays can be specified quite dynamically and flexibly.

2.2 Non-Markovian Approach

Until now, we imposed a strong constraint on the model. We assumed, that all transitions have either exponentially distributed random firing time or fire immediately after enabling. The resulting GSPNs fulfil the Markov property and allow simple, convenient and fast evaluation. Nevertheless, the exponential distribution is not well-suited for modelling some behaviour of the real system.

2.2.1 Reconsidering Exponential Distributions

Sticking to the GSM/GPRS/DOVE model with DeTVoC (→ Fig. 2.8, p. 48), we will now discuss the applicability of the exponential distribution for modelling the different timed transitions.

t1 and t2: call arrival and service process

According to [43], classical telephone traffic is appropriately described by (negative-) exponentially distributed holding times and call inter-arrival times. Therefore, we consider Poisson processes as adequate instrument to approximate these processes. (The dwell time, i.e. the time a user stays in one cell before handover, instead can be modelled more appropriately using lognormal distribution [46]. This has to be taken into account when considering handovers.)

t3 and t4: data arrival and service process

Comparable to the Internet, the types for the most common data transmitted over GPRS are WWW and FTP [35]. This will result into so called *heavy-tailed* distributions for file sizes and therefore heavy-tailed burst lengths. This behaviour is not appropriately described by the exponential distribution transition **t4**. Instead, Weibull, Pareto, lognormal, phase-type or general exponential distributions are suggested by several publications [43, 47, 48, 49, 50] to model the higher variability of inter-arrival and service times for self-similar data traffic.

In practice, all too often only the first two moments, i.e. the mean and the standard deviation (or variance or square coefficient of variation), are available as empirically obtained information about the random variables' distribution [45, 48]. Therefore, it would be practical to use only these two parameters without making any other questionable assumptions [48]. A possible distribution to work with would then be a widely general distribution, characterised only by its mean and variance, that we will call *empirical distribution* (*EMP*) later (→ Chap. 3.2, p. 64).

t6: DeTVoC

The time a user will wait for the establishment of his connection before giving up will be distributed upon a limited range. There will be no user waiting forever and only a low number of users waiting less than a second. Therefore, the user waiting time will be approximated better by a normal or uniform distribution.

t7: DOVE

DOVE uses a timeout that might be state dependent for adaptive CAC but deterministic. Therefore, a deterministic firing time for transition **t7** will be much more realistic than an exponentially distributed random firing time.

2.2.2 Limitations of MOSEL-2

The use of the non-Markovian distributions explained above within a MOSEL-2 model causes the following problems:

- MOSEL-2 using MOSES only allows the usage of models equivalent to GSPNs. Therefore, only exponential distributions are allowed. [2]
- MOSEL-2 using TimeNET only allows *extended DSPNs* (*eDSPNs*), i.e. only exponential, deterministic and uniform distributions can be used. Furthermore, models with more than one deterministically distributed transition enabled simultaneously can only be simulated or approximatively analysed. Models with more than one uniformly distributed transition enabled simultaneously can only be simulated. [2] (→ Chap. 1.5.2, p. 22)
- Concerning the distributions mentioned above, MOSEL-2 2.1 would be able to handle deterministic, lognormal, normal, Pareto and several phase-type distributions using SPNP 6.1.2a simulation [12]. Unfortunately, SPNP 6.1.2a was a non-official version of SPNP and is no longer available.

Based on this restrictions, we are currently only able to improve the behaviour modelling of DeTVoC (**t6**) and DOVE (**t7**) by introducing uniform and deterministic distributions.

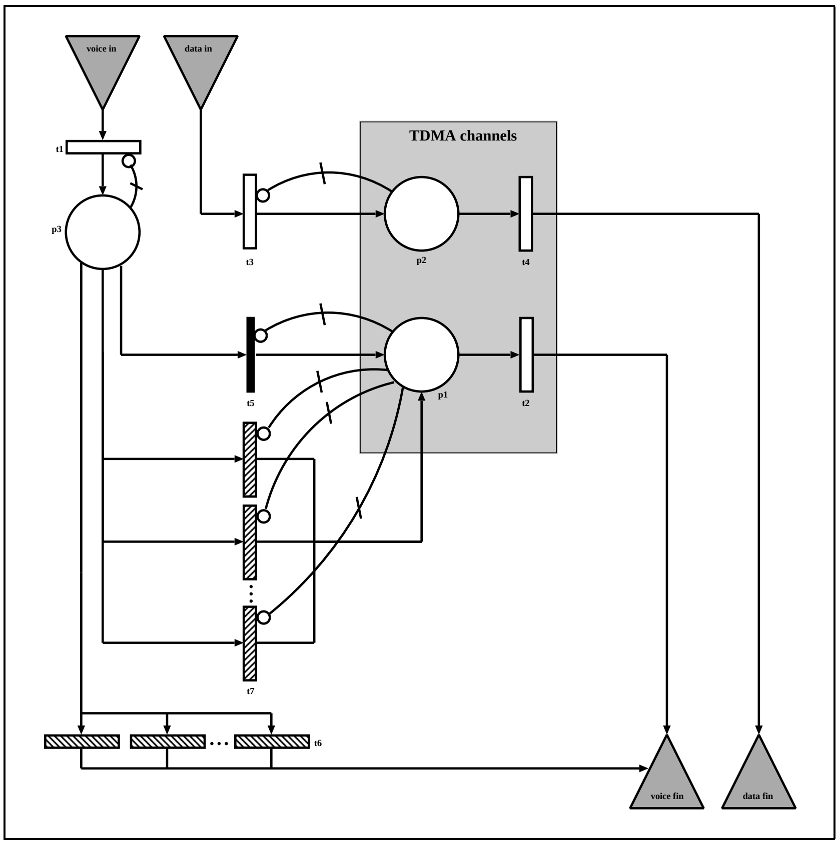

Figure 2.10: Conceptual eDSPN model of GSM/GPRS system with DOVE.

2.2.3 Conceptual non-Markovian Model of GSM/GPRS/DOVE System with DeTVoC

The improvements lead to the conceptual eDSPN model shown in Figure 2.10:

- We cannot specify the state dependent firing times for the infinite server transitions **t6** and **t7**, that now have non-exponentially distributed firing times, as easily as we did for exponential distributions. Therefore, we have to specify a single transition for every token in place **p3**. These transitions then will be provided with state dependent enabling functions, that ensure, that for every token located in place **p3** one **t6** transition and one **t7** transition gets enabled.
- For **t6** we specify a set of transitions with a deterministic firing time of seven seconds.
- For **t7** we specify a set of transitions with uniformly distributed firing times from five to 10 seconds.

2.2.4 MOSEL-2 Model of non-Markovian GSM/GPRS/DOVE System with DeTVoC

```
/** CONSTANT AND PARAMETER PART ***********************************************/
// GSM user behaviour
PARAMETER v_ar := 0.005 .. 0.015 STEP 0.002;  /* mean voice call arrival rate */
CONST     v_ht := 100;                        /* mean voice call holding time */
CONST     v_sr := 1/v_ht;                     /* mean voice call service rate */
CONST     v_dt_min := 5;                   /* minimum voice call waiting time */
CONST     v_dt_max := 10;                  /* maximum voice call waiting time */

// GPRS data behaviour
CONST     d_ar := 0.7;                        /* mean data burst arrival rate */
CONST     d_sz := 10 * 8;                      /* mean data burst size (kbit) */

// network configuration
CONST     freq_c    := 1;                      /* carrier frequencies in cell */
CONST     chans_f   := 8;                      /* physical channels per freq. */
CONST     chans     := freq_c*chans_f;           /* overall numb. of channels */
CONST     chans_sig := 2;                     /* channels used for signalling */
CONST     chans_use := chans-chans_sig;       /* channels available for users */
CONST     chans_gpr := 0;                       /* channels reserved for GPRS */
CONST     chans_gsm := chans_use - chans_gpr; /* max. numb. of voice channels */
CONST     gprs_max  := 32 * freq_c;           /* maximum number of GPRS users */
CONST     pdch_dr   := 21.4;                   /* data rate per PDCH (kbit/s) */
CONST     burst_sr  := pdch_dr/d_sz;           /* burst service rate per PDCH */
CONST     dove_dt   := 7;                              /* delay time for DOVE */
CONST     dove_res  := 1; /* number of PDCHs saved from immediate pre-emption */
CONST     dove_vch  := chans_use - dove_res;     /* always available for voice */

/** NODE PART *****************************************************************/
NODE p1[chans_gsm]     := 0;                                            /* p1 */
NODE p2[gprs_max]      := 0;                                            /* p2 */
NODE p3[20]            := 0;                                            /* p3 */

/** RULE PART *****************************************************************/
FROM EXTERN TO p3 RATE v_ar;                                            /* t1 */
FROM p1 TO EXTERN RATE v_sr * p1;                                       /* t2 */
FROM EXTERN TO p2 RATE d_ar;                                            /* t3 */
FROM p2 TO EXTERN RATE burst_sr * (chans_use - p1);                     /* t4 */
IF (p1 < dove_vch) FROM p3 TO p1;                                       /* t5 */

@<1..20>{                                                               /* t6 */
 IF p3 >= # FROM p3 TO EXTERN AFTER v_dt_min .. v_dt_max;               /* t6 */
}                                                                       /* t6 */

@<1..20>{                                                               /* t7 */
 IF p3 >= # FROM p3 TO p1 AFTER dove_dt;                                /* t7 */
}                                                                       /* t7 */

/** RESULT PART ***************************************************************/
PRINT v_loss  := 1 - ((v_sr * MEAN(p1)) / v_ar);     /* voice call loss prob. */
PRINT d_loss  := PROB(p2==gprs_max) + PROB(p1==chans_use); /* data burst loss */
PRINT p3_full := PROB(p3==20);

/** PICTURE PART **************************************************************/
PICTURE "call and burst loss probability"
PARAMETER v_ar
CURVE d_loss
CURVE v_loss
XLABEL "voice call arrival rate"
YLABEL "call and burst loss probability"
```

We changed the following elements:

- Lines 6-7: The voice call user's mean waiting time is replaced by a time range from `v_dt_min := 5` to `v_dt_max := 10` seconds. These values are used for the uniform distribution of transition **t6**.
- Line 24: The delay time for DOVE is now constant at 7 seconds. This value is used as parameter for the deterministic distribution of transition **t7**.
- Line 31: To avoid very rare events, keeping the model smaller and shorten simulation time, we decrease the capacity of place **t3**. The applicability of the smaller capacity will be proven with the results provided by line 51.
- Lines 40-42: The former single rule of transition **t6** is now replaced by a whole set of rules to handle the infinite server property of **t6**. A loop construct generates as many rules as place **p3** can hold. Each generated rule gets its own enabling function. The loop construct gets expanded by the MOSEL-2 syntax pre-processor. The expanded loop would be:

  ```
  IF p3 >= 1 FROM p3 TO EXTERN AFTER v_dt_min .. v_dt_max;
  IF p3 >= 2 FROM p3 TO EXTERN AFTER v_dt_min .. v_dt_max;
  IF p3 >= 3 FROM p3 TO EXTERN AFTER v_dt_min .. v_dt_max;

                                ...

  IF p3 >= 20 FROM p3 TO EXTERN AFTER v_dt_min .. v_dt_max;
  ```

  More information about MOSEL-2 loops can be found in [2]. `AFTER v_dt_min .. v_dt_max` describes a firing time with uniform distributed firing time from `v_dt_min` to `v_dt_max`.
- Lines 44-46: Similar to transition **t6** we use a loop construct to generate all rules necessary to describe the infinite server behaviour of transition **t7**.
- Line 51: The results of this measure will prove, that a capacity of 20 for place **p3** still will be sufficient.

2.2.5 Results

We evaluate this modes using MOSEL-2 and TimeNET by starting MOSEL-2 with the following command line options described in [2]:

```
mosel2 -Tsvo simulation -o "-S -Ton 95 50 10 1 0 50 0 600" thesis_2_2_4.mos
```

This starts TimeNET's discrete event simulation method and restricts the overall time used for the simulation of a single tool-specific input file to 10 minutes. Because of five different *parameter settings*, i.e. five possible combinations of parameters, five input files are generated by MOSEL-2 and simulated by TimeNET. The overall process took 116m/121s/60m (user/system/real).

Although TimeNET should be able to solve the above model by simulation, we are not able to obtain any reasonable results:

```
Stationary analysis of "thesis_2_2_4.mos" by TimeNET

Parameters:
  v_ar = 0.005

Results:
  v_loss = nan
  d_loss = nan
  p3_full = nan

=====================================================

Parameters:
  v_ar = 0.007

Results:
  v_loss = nan
  d_loss = nan
  p3_full = nan

=====================================================

Parameters:
  v_ar = 0.009

Results:
  v_loss = nan
  d_loss = nan
  p3_full = nan

[...]
```

The following error messages are generated by TimeNET:

```
SIMULATING THE DSPN ...
[...]
NOTE: A statistic seems to have no variance,
100 successive equal samples (value: nan) recorded.
Check the result definitions for deterministic behaviour.
```

Unfortunately, the reasons for this errors could not be determined within the bounds of this thesis.

Chapter 3

Improving MOSEL-2

Two deficiencies were discovered above:

1. Only a few non-Markovian distributions, i.e. deterministic and uniform, are available for evaluation with MOSEL-2 1.2 at the moment.
2. The evaluation of non-Markovian models using simulation is a very time-consuming task.
3. The TimeNET simulation seems to be unreliable for some models.

To come around these problems, this chapter will be used to improve MOSEL-2 ...

1. ... by enabling the handling of SPNP 6.1 simulation that allows the usage of a lot of other distributions.
2. ... by introducing a pre-processor for automatic approximation of general distributions by Markovian constructs. This will allow us to evaluate non-Markovian models approximately by faster numerical analysis.

The changes lead to the current MOSEL-2 version 2.10. All references to lines of code are (more or less) based on this MOSEL-2 version.

3.1 Using SPNP 6.1 Simulation

MOSEL-2 2.1 was recently enhanced to use the discrete event simulation provided by SPNP 6.1.2a [12]. Unfortunately, SPNP 6.1.2a is no longer available. To enable MOSEL-2 to work with the simulation method provided by the official and still available SPNP version 6.1, several changes had to be done. While introducing these changes, special care was taken, to leave the handling of other methods than SPNP and the handling of the numerical analysis of SPNP untouched.

3.1.1 Problem Description

The main problem was caused by the fact, that SPNP 6.1 – in contrast to SPNP 6.1.2a – does not support the CSPL-function `expected()` (or `cum_expected()`) in simulation mode.

`expected()` is used to calculate the mean of the return values from a marking dependent reward function that is passed as parameter. The return value of `expected()` then can be used within CSPL to calculate more complicated result measures, that even may contain return values of further `expected()` calls. The final values can then be printed to the output stream using the CSPL-function `pr_value()`.

Instead of `expected()`, SPNP 6.1 does only support the CSPL-function `pr_expected()` (or `pr_cum_expected()`). Unfortunately, this function does not provide a return value that could be used for further calculations, but only prints the calculated mean to the output stream.

Furthermore, the format of the output of `pr_expected()` and `pr_cum_expected()` differs from the `pr_value()` output.

When using SPNP 6.1 simulation, the output file can have two parts:

1. The first part is produced using numerical analysis. For this, it seems, that each non-Markovian distribution is substituted by an exponential distribution. It appears as if the first parameter of the original non-Markovian distribution simply is taken as parameter (firing rate) for the exponential distribution, which does not make any sense in most cases. This part may even show results, if `pr_value()` or other CSPL result output functions are used, that are only allowed for numerical analysis (i.e. `pr_mtta()`). The results, of course, are useless as long as the model is non-Markovian.
2. The second part only appears, if the SPNP-option `IOP_SIM_STD_REPORT` is not set to `VAL_NO`. This part only includes results that were specified using `pr_expected()` or `pr_cum_expected()`. In contrast to SPNP 6.1.2a, the output does not provide the mean values in addition to the confidence interval.

3.1.2 Changes to the CSPL-Generator

Enabling Simulation Output

In MOSEL-2 2.1 the SPNP-option `IOP_SIM_STD_REPORT` was set to `VAL_NO` by default because the additional output was not necessary and made the result parsing more complicated. To obtain simulation results, this option now has to be set to `VAL_YES` instead. *(cspl.c:747)*

Using `pr_expected()`

In simulation mode, `pr_expected()` or `pr_cum_expected()` have to be used instead of `expected()`, `cum_expected()` and `pr_value()`. For this, all terms that formerly contained `expected()`-calls, now have to be specified in their own reward function.

In other words:

- The following line (specified in MOSEL-2):

```
PRINT lambda1 := UTIL(p1) * mue1;
```

- was translated to CSPL for SPNP 6.1.2a like this (simplified):

```
double reward_func_0_ (void)
{
  return (mark ("p1") > 0 ? 1.0 : 0.0);
}

void ac_final (void)
{
solve (INFINITY);
  {
    double reward_0_ = expected (reward_func_0_);
    double lambda1_ = reward_0_ * mue1_;

    pr_value ("lambda1", lambda1_);
  }
}
```

- Now, using SPNP 6.1, it has to be translated like this (simplified)[1]:

```
double reward_func_0_ (void)
{
  return (mark ("p1") > 0 ? 1.0 : 0.0);
}

double sim_lambda1_(void)
{
   double ret_val = (reward_func_0_() * mue1_);
   return ret_val;
}

void ac_final (void)
{
    pr_expected ("R_1", sim_lambda1_);
}
```

This was achieved by changing or inserting the following lines of code in file cspl.c: 118-125, 157-165, 1365-1369, 1414-1416, 1436-1459, 1474-1478, 1507-1516, 1524-1527, 1530-1548

Unfortunately, this workaround only provides proper results for a certain class of simple result terms. The result term $f()$ using the marking dependent reward functions a, b, ... must accomplish the following property:

$$f(AVG(a), AVG(b), ...) = AVG(f(a, b, ...))$$

$AVG(z)$ denotes a function, that calculates the mean value of the marking dependent reward function z over all states (like `(pr_)(cum_)expected()`).

[1]Why the `solve()` call had to be removed (→ Chap. 3.1.4, p. 61) and the result name had to be changed (→ Chap. 3.1.2, p. 60) is described below.

Of course, this assumption is wrong in general. But the property does hold for several simple performance measures, e.g.:

- $f()$ is constant.
- $f()$ is a sum.
- $f()$ is a product and all reward functions a, b, ..., are constants except one.

This is better than nothing, but a better solution of this problem would be to evaluate only the basic reward functions (UTIL, MEAN, PROB) with the help of the tools and calculate the desired result terms within the MOSEL-2 result parser. This would also make it easier to include more tools. But, this requires a major rewrite of both the translators and the result parser, which could not be done in this thesis due to lack of time.

Result Symbol Encoding

The simulation output of SPNP 6.1 truncates result-names after 22 characters. If long node or result names are used in MOSEL-2 this could happen easily, especially if job-distribution measures are desired. In this case, the result measures may not be identified by the MOSEL-2 result parser correctly anymore. To solve this problems, in simulation mode every result measure now gets a unique ID. To keep this ID as short as possible but still readable and easy to parse, it is base62 encoded using the characters: '0'-'9', 'a'-'z' and 'A'-'Z'.

The base62 encoder is located in file basic.c (lines 258-278 and 303-317). A base62 decoder is also provided in file basic.c (lines 280-300 and 319-333) but currently unused. According entries are located in file basic.h (lines 101-120).

The encoder is used on the fly by the CSPL-generator cspl.c from line 1539 to line 1546 for normal result measures (prefixed with "R_"). For job distributions measures (preceded by "D_" and suffixed by "_" and the number of jobs) the base62 encoded node-id is used by the CSPL-generator (lines 1368-1372). The node-id is generated only once when saving the node to the MOSEL-2 internal model description. This is done in model.c (lines 422-436). The node structure is adapted to carry the ID in model.h (line 104).

To calculate the memory used by the base62 encoded IDs properly, the function `ceil()` is used (cspl.c:1539, model.c:425). This made it necessary to introduce the header file `math.h` to file cspl.c (line 20).

3.1.3 Changes to the Result-Parser

To make result parsing more flexible – not at least in the face of coming versions of SPNP – I decided to introduce support for Perl compatible regular expressions (pcre). For this, the corresponding library has to be linked to the MOSEL-2 code as specified in file Makefile.in (line 34) and the header file pcre.h has to be included in the result parser result.c (line 24).

To make the actual parsing more convenient, a helper function `parse_sim_output()` is provided in result.c (lines 1188-1222). Providing a search pattern and a line that should be parsed, this helper function can return up to three substring matches. The helper function is used by the simulation result parser (result.c: 1315-1419). To build the search pattern, result.c uses the base62 encoder again.

SPNP 6.1 simulation does not provide mean values for the results. Therefore, the average of the lower and upper bound of the confidence interval is calculated by the result parser (result.c: 1364-1365). To get more information about the accuracy of the simulation results, the result parser could be extended to show the confidence interval also in the MOSEL output.

3.1.4 Additional Changes

- The invocation of the numerical analysis before starting the simulation is done by the CSPL function `solve()`, that was generated by cspl.c. As the numerical analysis is very time- and memory-consuming and provides bogus results in most cases, the `solve()` call is now omitted. Unfortunately, we are not able to use (several) `solve()`-functions in one simulation run. Therefore, I had to disable transient analysis with multiple steps ("transient loop") when using SPNP simulation. Single-step transient analysis is still possible. As upper time limit, the variable `SIM_LENGTH` is set. If neither an upper time limit nor `SIM_LENGTH` is specified by the user manually (steady-state evaluation), MOSEL-2 now sets `SIM_LENGTH` to 10^6. Unfortunately, SPNP requires that `SIM_LENGTH` is specified by the user. Therefore, no real steady-state evaluation seems to be possible. (cmdline.c: 197-200; model.c: 1012-1016; main.c: 261-265; cspl.c: 682-745, 1567, 1586)

- There is no equivalent for the CSPL-function `pr_mtta()` in simulation mode. Therefore, duration methods will be ignored when using SPNP simulation. An adequate warning is generated (main.c: 266-269; result.c: 1414-1416).

- The order of the parameters of the hypoexponential distribution changed between SPNP versions 6.1.2a and 6.1. Now, the order is as described in [10]. The parameter check of MOSEL-2 had to be adapted in parser.y (lines 703-725).

- In SPNP 6.1 some distributions are available, that did not work with 6.1.2a: uniform, Weibull and negative binomial. These distributions are now available for MOSEL-2. The Weibull and negative binomial distributions where introduced similar to all other SPNP simulation distributions as described in [12] (scanner.l: 131-132; parser.y: 726-765, 1364, 1586-1589; model.h: 48-49, 145-146; model.c: 974-979, 1004-1007, 1214-1229; main.c: 137-140, 243-257, 324-327; cspl.c: 588-635, 956-993). For using the uniform distribution, that is already available for use with TimeNET, a distinction between uniform and discrete uniform distribution had to be implemented additionally (model.h: 29; model.c: 974-979; main.c: 141-142, 174-175, 340-343; timenet.c: 926), because only the former one is supported by SPNP simulation. The negative binomial distribution does only work, if we consider a typo ("negboval" instead of "negbval") in the SPNP 6.1 header file user.h (line 50). If this typo will be fixed in future SPNP versions, "negboval" has to be changed into "negbval" in file cspl.c (line 958).

Distribution	MOSEL-2 syntax
beta	BETA (a, b)
binomial	BINOM (a, b, c)
Cauchy	CAUCHY (a, b)
deterministic	AFTER a
Erlang	ERLANG (a, b)
exponential	RATE a
gamma	GAMMA (a, b)
geometrical	GEOM (a, b)
hyperexponential	HYPER (a, b, c)
hypoexponential	HYPO[1] (a, b, c, d)
lognormal	LOGN (a, b)
negative binomial	NEGB (a, b, c)
normal	NORM (a, b)
Pareto	PARETO (a, b)
Poisson	POIS (a, b)
uniform	AFTER a .. b
Weibull	WEIB (a, b)

Table 3.1: Distributions available for MOSEL-2 2.10 that can be evaluated using SPNP 6.1 simulation.

- The function CEIL is available to MOSEL-2 now. CEIL is comparable to FLOOR, but rounds the argument to the next greater integer. The appropriate changes were done in scanner.l (line 134), parser.y (lines 1368, 1778-1781), expr.h (line 38), expr.c (lines 165, 647-653, 772, 831-832, 943, 1022) and cspl.c (lines 51, 85, 187). Similar to FLOOR, SIN and SQRT [2], CEIL has some restrictions:
  - Using MOSES, CEIL may not be used in state-dependent expressions.
  - Using TimeNET, CEIL is only allowed in a state-dependent expression if its argument is constant.

3.1.5 Summary

- MOSEL-2 is now able to work with the discrete event simulation of SPNP 6.1
- The available distributions for SPNP 6.1 simulation are listed in Table 3.1. The parameters (a-d) of the different distributions are described in [12] and [10].
- The operator CEIL can be used in MOSEL-2 (with restrictions similar to FLOOR, SIN or SQRT).
- Duration measures are not allowed if using SPNP 6.1 simulation.
- Transient analysis is allowed for SPNP 6.1 simulation but may not contain several steps (transient loops).

[1]The order of parameters is as described in [10], not as described for SPNP 6.1.2a in [12]: a = number of stages (2 or 3); b-d = rates for stages 1 to 3.

SPNP Simulation Option	transient[2] at time T	"steady-state"
IOP_SIMULATION	VAL_YES	VAL_YES
FOP_SIM_ERROR	0.1	0.1
FOP_SIM_LENGTH	T	10^6
FOP_SIM_STD_REPORT	VAL_YES	VAL_YES

Table 3.2: SPNP 6.1 simulation defaults.

- If using SPNP 6.1 simulation, the result measures may only be sums of constants, or elements that are linearly dependent from one marking dependent result function (MEAN, UTIL, PROB), e.g.:

```
PRINT example_a := CEIL(SQRT(param_1 * const_1))^2; /* valid: constant                */
PRINT example_b := MEAN(place1);                    /* valid: lin. dependent from MEAN */
PRINT example_c := example_a * example_b;           /* valid: lin. dependent from MEAN */
PRINT example_d := UTIL(place1) + PROB(place2==10); /* valid: sum of reward functions  */
PRINT example_e := example_a * example_d;           /* valid: sum of lin. dependencies */
PRINT example_f := UTIL(place1) * PROB(place2==10); /* invalid: contains UTIL * PROB   */
PRINT example_g := example_b * example_c;           /* invalid: contains MEAN * MEAN   */
```

- The default SPNP simulation options chosen by MOSEL-2 changed from MOSEL-2 version 2.1 to version 2.10. The defaults are shown in Table 3.2. A description of these and other options is given in [12] and [10].

[2]When the user wants to start transient evaluation, he has to specify the evaluation time point T.

3.2 Introducing general Distributions – The Rule Pre-Processor

As we have seen, only a few non-Markovian models, i.e. DSPNs, can be solved by numerical analysis using MOSEL-2 2.1. Usually, non-Markovian models have to be evaluated using discrete event simulation, which is a very time-consuming task. We will now enable approximative numerical analysis for a subclass of general distributions whose first two moments are known: their mean and variance. We will call this subclass *empirical distributions* (*EMP*), because the mean and the variance are values that can be obtained quite easily from measurements. For example, consider X_i with $i = 1 \dots k$ as measured values (samples). Then the first two moments of the values' distribution are [14]:

$$\overline{X} = \frac{1}{k} \sum_{i=1}^{k} X_i \tag{3.1}$$

$$\overline{X^2} = \frac{1}{k} \sum_{i=1}^{k} X_i^2 \tag{3.2}$$

The mean is equal to the first moment. The variance can be calculated as:

$$\sigma^2 = \overline{(X - \overline{X})^2} = \overline{X^2} - \overline{X}^2 \tag{3.3}$$

3.2.1 Concept

The analyst should be able to use an empirical distribution directly in MOSEL-2 by applying

```
EMP (mean, var)
```

to the according rule, whereas `mean` or $\bar{t}$ is the mean of the firing time and `var` or σ^2 is the variance of the firing time's distribution. This rule then gets substituted by one or several sub-rules depending on the *square coefficient of variation* (*SCV*) c^2:

$$c^2 = \frac{\sigma^2}{\bar{t}^2} \tag{3.4}$$

If c^2 equals one, the empirical distribution is equivalent to an exponential distribution. If it is greater than one, the distribution can be approximated by a general exponential distribution. If c^2 is smaller than one, the distribution can be approximated by a hypoexponential distribution.

Consider the following MOSEL-2 rule:

```
FROM p1 TO p2 EMP (mean, var);
```

This rule can be represented by the Petri net shown in Figure 3.1

The dotted arcs to the environment represent arbitrary input, output or inhibitor arcs (or, more general, guard functions) to the adjacent parts of the model.

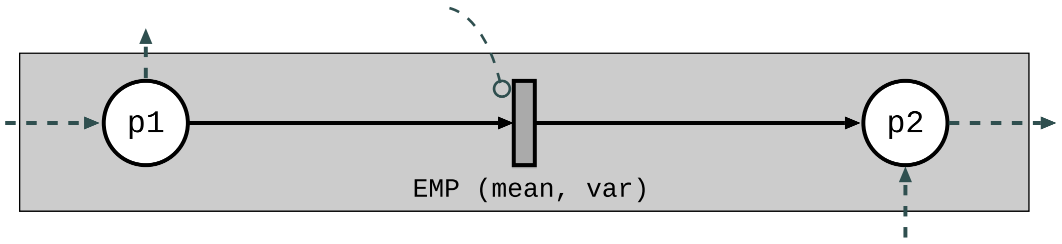

Figure 3.1: Example for transition with empirical distributed firing time.

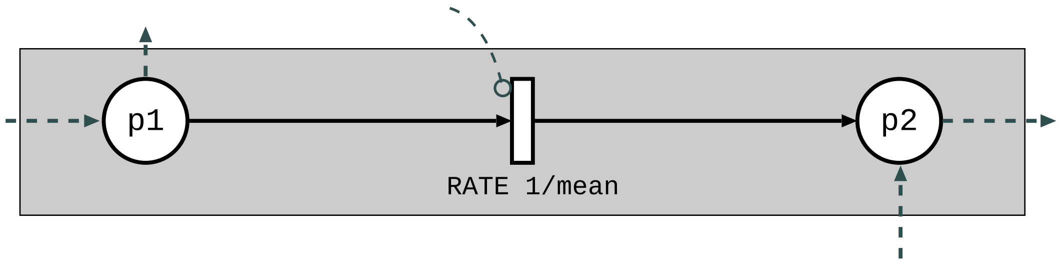

Figure 3.2: Transition with exponentially distributed firing rate.

Exponential Substitution ($c^2 = 1$)

Now consider the parameters `mean > 0` and `var = SQRT(mean)` for Figure 3.1 The SCV resulting from these parameters is:

$$c^2 = \frac{\sigma^2}{\bar{t}^2} = \frac{mean}{mean} = 1 \tag{3.5}$$

A c^2 of one is equivalent to an exponential distribution. The empirical distribution in 3.1 can then be substituted by the exponential distribution of Figure 3.2.

General Exponential Substitution ($c^2 > 1$)

If c^2 is greater than 1, we choose the general exponential (GE) distribution [48, 52, 53, 54, 55] as substitution. The GE distribution can be modelled as shown in Figure 3.3. The advantage of the GE distribution is, that the size of the equivalent model does not change with the parameters – in contrast to, for example, the Cox distribution.

The transition probability p can be calculated as:

$$p = \frac{2}{c^2 + 1} \tag{3.6}$$

We can use weights instead of transition probabilities in MOSEL-2. This allows us to define the weight of the exponential path as one. The weight of the immediate path can than be calculated as $\frac{1}{c^2-1}$.

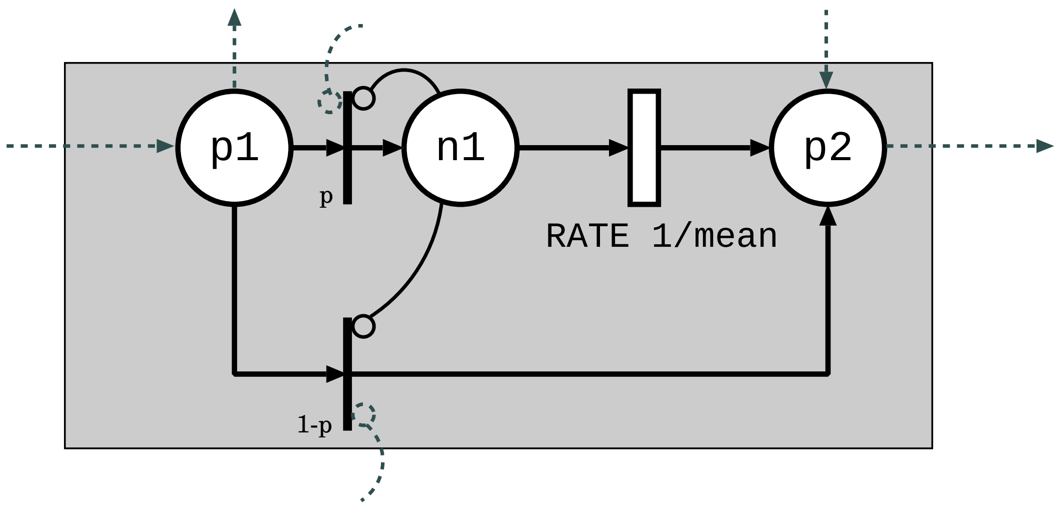

Figure 3.3: Equivalent model for GE distributed transition.

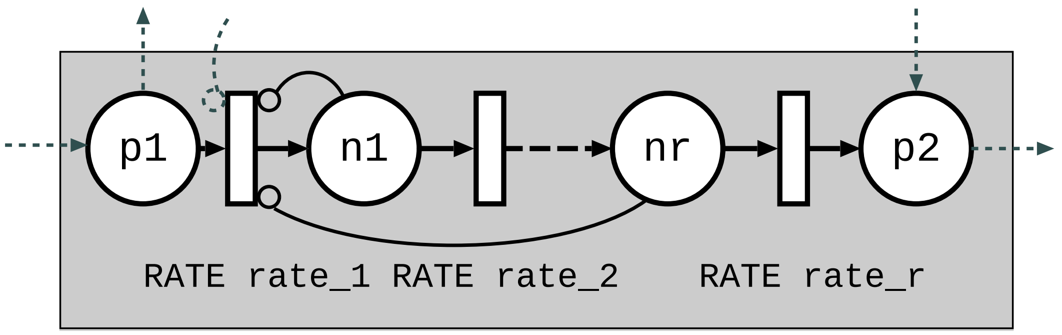

Figure 3.4: Equivalent model for PH distributed transition.

Phase Type Substitution ($c^2 < 1$)

If c^2 is less than 1, then we use a subclass of phase type (PH) distributions, the hypoexponential distribution, as substitution [56]. The equivalent model is shown in Figure 3.4.

The number r of exponential distributed stages needed can be calculated using:

$$r = \lceil \frac{1}{c^2} \rceil \tag{3.7}$$

The calculation of the firing rates `rate_1` (μ_1) to `rate_n` (μ_n) of the exponential stages $1 \ldots r$ can be done using the following recursive algorithm:

- Step 1: We calculate the SCV c^2 using Equation 3.4.
- Step 2: We determine r using Equation 3.7.

- Step 3: If $r = 2$, then we finish using:

$$\mu_{a,b} = \frac{2}{\bar{t}}\left(\frac{1}{1 \pm \sqrt{2c^2 - 1}}\right) \tag{3.8}$$

- Step 4a: r is even: We split the further investigation into two identical parts (a and b) and restart the algorithm with Step 0 twice using the parameters $\overline{t_a}$, σ_a^2 and $\overline{t_b}$, σ_b^2, whereas:

$$\overline{t_a} = \overline{t_b} = \frac{\bar{t}}{2} \tag{3.9}$$

  and:

$$\sigma_a^2 = \sigma_b^2 = \frac{\sigma^2}{2} \tag{3.10}$$

- Step 4b: r is odd: We determine the rate of a single stage using:

$$\mu_a = \frac{1}{\overline{t_a}} = \frac{r}{\bar{t}} \tag{3.11}$$

  We then restart the algorithm with an even number of remaining stages using the parameters:

$$\overline{t_b} = \bar{t} - \overline{t_a} = \bar{t}(1 - \frac{1}{r}) \tag{3.12}$$

  and

$$\sigma_b^2 = \sigma^2 - \sigma_a^2 = c^2 \cdot \bar{t}^2 - 1 \cdot \overline{t_a}^2 = c^2 \cdot \bar{t}^2 - (\frac{\bar{t}}{r})^2 = \bar{t}^2(c^2 - \frac{1}{r^2}) \tag{3.13}$$

3.2.2 Implementation

The substitutions for approximating the empirical distribution should be generated automatically by a rule *pre-processor* within MOSEL-2. A first step will be to examine the MOSEL-2 architecture and to decide, where the pre-processor should be attached.

Architecture of MOSEL-2

The MOSEL-2 evaluation environment can be divided into six main parts:

1. Command line parser: The command-line parser checks and evaluates the options given when starting MOSEL-2. It is implemented in file cspl.c.
2. Input file scanner/parser: The input file scanner (scanner.l) does the lexical analysis of the input file, i.e. it splits the source file into tokens ignoring whitespace (blanks, tabs, newlines) and comments. Furthermore, the syntactical scanner pre-processes and expands the MOSEL-2 loops. The parser (parser.y) instead does the syntactical analysis. It finds the hierarchical structure of the scanned input file, checks the semantics of the input (e.g. the parameters of distributions) and stores the MOSEL-2 model into the internal model description.

3. MOSEL-2's internal model description is implemented in model.c. It takes the model information from the parser and stores it in several data structures. The internal model description can be dumped using the command line option -d to check, if the model was parsed as expected.

4. Model translators: There exist model translators for MOSES (moslang.c), SPNP (cspl.c) and TimeNET (timenet.c). These translators take their information from the internal model description and generate the tool-specific input file out of it.

5. Tool invocation and result parsing: result.c starts the appropriate tool with the specified options and parses the output produced by the tool to provide the tool independent MOSEL-2 result files.

6. Main routine: The main routine (main.c) coordinates all parts mentioned above.

Placing the Pre-Processor within the Architecture of MOSEL-2

The *pre-processor* – as its name implies – should operate before the MOSEL-2 internal model description is build. This has several advantages:

- The current and future tool specific translators do not have to be changed or complicated significantly.
- The output of the pre-processor can be checked using the model dump option.

Furthermore, the pre-processor . . .

- . . . will have to pre-process MOSEL-2 rules that contain a special distribution (e.g. EMP). Therefore, the pre-processor will have to work after the input file scanner and parser, where single rules are available as such.
- . . . should have as less impact on "classical" MOSEL-2 rules as possible.
- . . . should be implemented as modular as possible to avoid changes to the former code and to allow easier debugging, improvement and expansion.

Putting all requirements together, we derive the conceptual architecture shown in Figure 3.5.

The parser passes all model information, that does not have to be pre-processed, directly to the internal model description as before. Only rules, that have to be pre-processed, for example because they have empirical distribution, are passed to the pre-processor instead. In this case, the pre-processor substitutes the original rule by one or several new rules and passes these to the internal model description.

The main functions of the rule pre-processor are located in rule_pp.c.

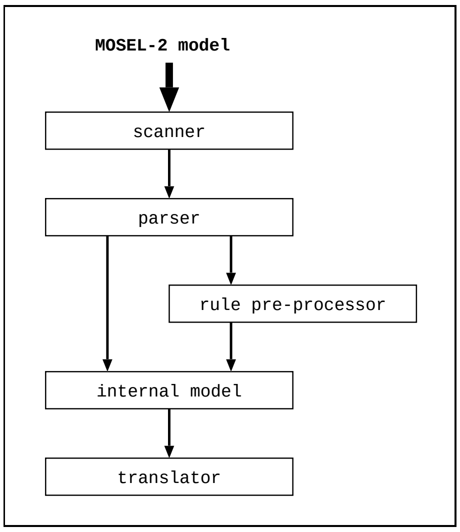

Figure 3.5: Placing the rule pre-processor within the MOSEL-2 architecture.

Detailled Description of the Pre-Processor

Due to lack of time and space, only the main concepts of implementation can be shown here. Not all functions are presented, e.g. the numerous functions for handling the data structures and the memory.

At first, we have do introduce the new keyword "EMP" for the new empirical distribution (scanner.l: 134; parser.y: 1367). If the parser recognises this keyword (parser.y: 1597), it calls function `set_distrib` (parser.y: 1599) and sets the variable `pp_state` to `PP_USED` (parser.y: 1600), which is set to `PP_UNUSED` by default (parser.y: 1499).

The function `set_distrib` (parser.y: 505-778) checks the distribution's parameters (parser.y: 513-769) and saves the distribution and its parameters to the parser internal rule structure (parser.y: 771-777) as it is done for all other distributions as well. This structure saves all data until the whole rule is parsed. Then, the data collected in the structure is transferred to the internal model description using the function `generate_rule` (parser.y: 1508, 901-1001). If the rule does not have to be pre-processed, the function `generate_rule` passes the rule information directly to the internal model description using the function `define_rule` (parser.y: 993-998) which is provided by model.c (lines 545-627). If instead the rule has to be pre-processed, i.e. the variable `pp_state` is set to `PP_USED`, then the rule information is passed to the pre-processor using the function `pp_rule` (parser.y: 983-988), which is provided by rule_pp.c (lines 1854-1935). The return value of `pp_rule` is the new number of temporary nodes including the nodes, that might have been generated by the pre-processor.

When the rule pre-processor is called, it first checks, if any error occurred while parsing the input file so far. If not, it copies the given rule information to a local data structure to work on (rule_pp.c: 1889-1895) and then decides which constructs the rule has to be substituted with on the basis of the used distribution (rule_pp.c: 1897-1915). Currently only empirical distributions (EMP) are supported. A few hints on how to implement other pre-processor distributions (e.g. Erlang) are given in Appendix A.

In case of EMP, function pp_emp is invoked (rule_pp.c: 1907). The function pp_emp (rule_pp.c: 1687-1850) calculates the SCV using the parameters mean and variance and then decides based on the SCV, which constructs (exponential, GE or phase type substitution) will be generated by the subroutines pp_dist_exp, pp_dist_ge or pp_dist_ph.

Due to the fact, that the SCV may take several values because of different parameter settings, and that some settings may produce constructs with terms that cannot be calculated in other settings (e.g. division by zero or negative square root) we have to introduce a new concept of conditions for rules. We call this conditions "rule allowance conditions" or just "allowance" to distinguish them from IF or enabling conditions that can be specified in MOSEL-2 or that can be specified by MOSEL-2 in the tool specific input files to enforce model constraints like capacities of nodes. In contrast to the latter conditions, the allowances are specified by the pre-processor and are used by the translators to determine, if the rule is applicable in the current setting of parameters or if it should be ignored. Nodes, that are only used in ignored rules in the current parameter setting, are ignored as well.

We will now have a closer look at the three types of substitutions of the EMP distribution:

Exponential Substitution: The exponential substitution is the easiest case. It can be substituted exactly as shown in Figure 3.2. No temporary nodes have to be generated. Only the distribution type has to be changed and the rate has to be calculated. The allowance of the generated rules does not change with the parameters as long the SCV is equal to one. Therefore, we have to specify the rule allowance only once, what is done in function pp_emp (rule_pp.c: 1764-1790). The generation of the final rule is then done in function pp_dist_exp (rule_pp.c: 863-902). The function pp_dist_exp manipulates the working rule only by setting the distribution type to exponential (rule_pp.c: 882), deleting the second parameter, the variance, which is no longer needed (rule_pp.c: 883-884), and sets the first parameter, now the rate, to a newly generated MOSEL-2 expression that calculates $\frac{1}{mean}$. Fortunately, this minor change does not affect other elements of the rule, e.g. arc multiplicities. The manipulated rule is then saved to the internal model description by using the function pp_save_rule (rule_pp.c: 891), which is provided by rule_pp.c as well (lines 164-197) and uses function define_rule which is provided by model.c.

GE Substitution: Even though the GE substitution – similar to the exponential – only needs one allowance (SCV > 1) for all rules that is specified in function pp_emp (rule_pp.c: 1805-1821), this construct is much more complicated. According to Figure 3.3 we would have to insert a new node (n1) between the source node p1 and the destination node p2. This would make it necessary to take care of input and output arc multiplicities that may also have effect on the temporary nodes capacity or – speaking in terms of MOSEL-2 – we would have to consider node arities or node states for the temporary nodes.

Therefore, we will not substitute the EMP distribution by the construct presented in Figure 3.3 but by a relay circuit construct as shown in Figure 3.6. This will keep the original rule except changing the distribution type and adding an input node. The relay circuit is then added using new and simple rules and basic nodes with capacity one.

The triangles in Figure 3.6 depict sources and sinks for tokens and only serve clarity. The dependence arcs were introduced before (→ Fig. 1.7, p. 13).

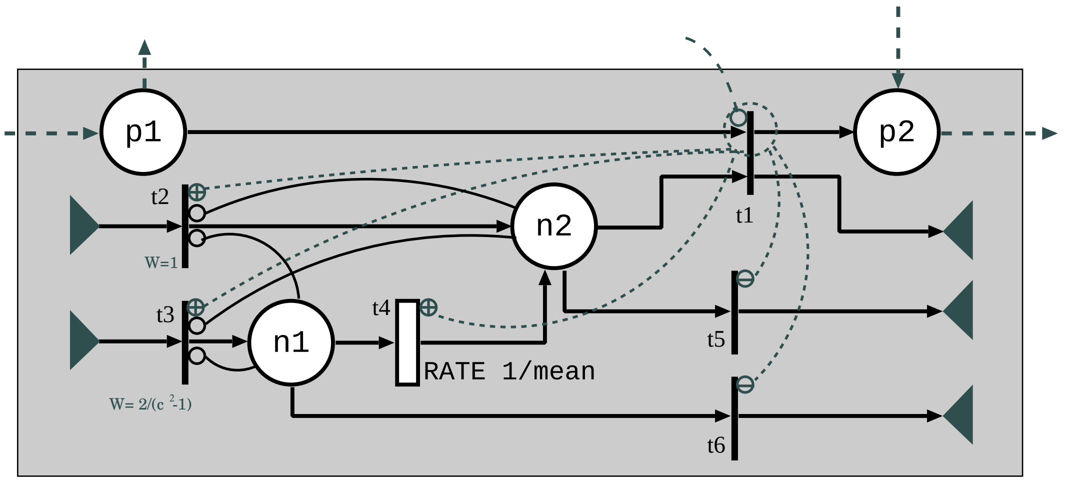

Figure 3.6: Construct of GE Substitution.

Example: A token enters place **p1**. This enables the upper part of the master transition **t1** and therefore transitions **t2** and **t3**. Nevertheless, only one of both transitions can fire according to their weight. If **t3** fires, a single token is generated in place **n1**. The token enables transition **t4** which has exponentially distributed firing time with firing rate $\frac{1}{mean}$. After the firing of **t4**, the token is transferred to place **n2**, which enables the main transition **t1**. The transitions **t5** and **t6** are only generated by the pre-processor, if the policy *preemptive repeat different* is chosen for the current rule. They clear the places **n1** and **n2** if the master transition gets disabled. If the policy *preemptive resume* is chosen, the token remains in **n1** or **n2** if the master transition gets disabled. Therefore, the relay circuit resumes its process when the master transition gets re-enabled. The policy *preemptive repeat identical* cannot be handled by the pre-processor. It is replaced with *preemptive repeat different.*

The GE substitution is build by the function `pp_dist_ge` (rule_pp.c: 906-1175).

PH Substitution: The phase type substitution comprises a relay circuit as well. The construct is shown in Figure 3.7. Unfortunately, the number of stages r depends on the SCV, i.e. we may have to generate the rule allowance dynamically for each settings (rule_pp.c: 1488-1497, 1547-1577).

The construct is built by function `pp_dist_ph` (rule_pp.c: 1374-1683). This function uses – among others – the function `pp_compute_ph_rates` (rule_pp.c: 1179-1368), which is the implementation of the algorithm that was introduced in Section 3.2.1 on page 66.

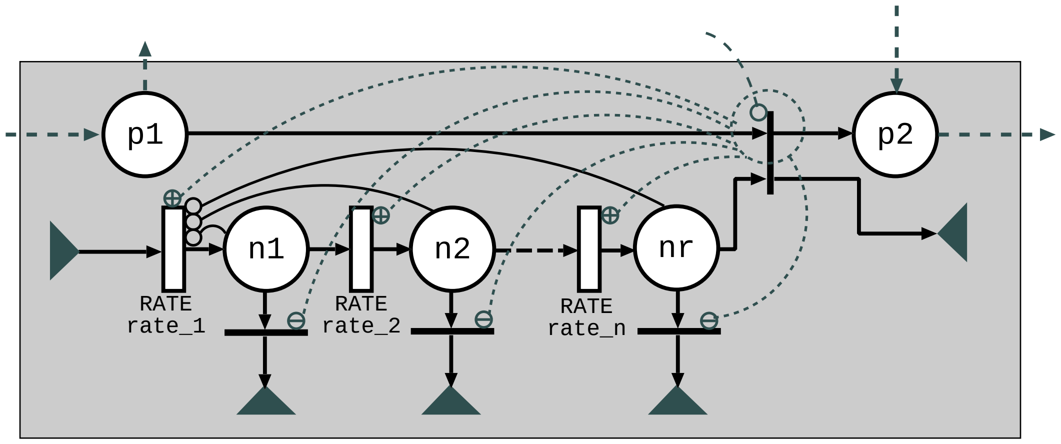

Figure 3.7: Construct of PH Substitution.

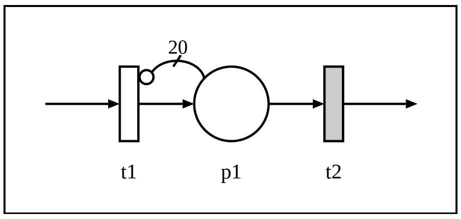

Figure 3.8: Conceptual Model for Testing the EMP distribution.

3.2.3 Example

Consider the ESPN shown in Figure 3.8. Let **t1** be a transition with exponentially distributed firing rate, **p1** a place with capacity 20 and **t2** a transition with empirical distribution.

The MOSEL-2 model: The following MOSEL-2 model can be derived:

```
PARAMETER mue_1 := 1 .. 10;              /* service rate of t1 */
CONST     t_2   := 5;                    /* mean firing time of t2 */
PARAMETER scv_2 := 0.1, 0.4, 1, 2, 10;   /* scv of t2 */
CONST     var_2 := scv_2 * (t_2 * t_2); /* variance of t2 */

NODE p1[20] := 0;                        /* p1 */

FROM EXTERN TO p1 RATE mue_1;            /* t1 */
FROM p1 TO EXTERN EMP (t_2, var_2) PRD; /* t2 */

PRINT avg_p1 = MEAN(p1);

PICTURE "average number of tokens in p1"
PARAMETER mue_1
CURVE avg_p1
```

The internal model description: We generate an output of the MOSEL-2 internal model description after parsing and pre-processing the model above using the MOSEL-2 option "-d":

```
Parameters:
  mue_1 := 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  scv_2 := 0.1, 0.4, 1, 2, 10

Symbols:
  t_2 := 1 / 8 [constant]
  var_2 := scv_2 * (t_2 * t_2) [constant]
  avg_p1 := MEAN (p1) [result, print]

Enums:

Nodes:
  p1[20] := 0
  temp_0_[1]
  temp_1_[1]
  temp_2_[1]
  temp_3_[1]
  temp_4_[1]
  temp_5_[1]
  temp_6_[1]
  temp_7_[1]
  temp_8_[1]
  temp_9_[1]
  temp_10_[1]
  temp_11_[1]
  temp_12_[1]
  temp_13_[1]

Assertions:

Rules:
 Source rule 1:
  0: TO p1 RATE mue_1
 Source rule 2:
  1: [FLOOR (var_2 / (t_2 * t_2) * 1e+10 + 0.5) / 1e+10 = 1]
     FROM p1 RATE 1 / t_2
  2: [var_2 / (t_2 * t_2) > 1]
     FROM p1 FROM temp_1_ PRIO 0
  3: [var_2 / (t_2 * t_2) > 1]
     IF temp_0_ < 1, temp_1_ < 1, p1 > 0 TO temp_1_ PRIO 0 WEIGHT 1
  4: [var_2 / (t_2 * t_2) > 1]
     IF temp_0_ < 1, temp_1_ < 1, p1 > 0 TO temp_0_ PRIO 0 WEIGHT 2 / (var_2 / (t_2 * t_2) - 1)
  5: [var_2 / (t_2 * t_2) > 1]
     FROM temp_0_ TO temp_1_ RATE 1 / t_2 PRIO 0
  6: [var_2 / (t_2 * t_2) > 1]
     IF NOT p1 > 0 FROM temp_0_ PRIO 0
  7: [var_2 / (t_2 * t_2) > 1]
     IF NOT p1 > 0 FROM temp_1_ PRIO 0
  8: [var_2 / (t_2 * t_2) < 1]
     FROM p1 FROM temp_2_ PRIO 0
  9: [CEIL (1 / (var_2 / (t_2 * t_2))) = 10 AND var_2 / (t_2 * t_2) < 1]
     IF NOT p1 > 0 FROM temp_3_ PRIO 0
[...]
  27: [CEIL (1 / (var_2 / (t_2 * t_2))) = 10 AND var_2 / (t_2 * t_2) < 1]
     IF temp_3_ < 1, temp_4_ < 1, temp_5_ < 1, temp_6_ < 1, temp_7_ < 1, temp_8_ < 1, temp_9_ < 1, temp_10_ < 1, temp_11_ < 1
     TO temp_11_ RATE 2 / (t_2 / 2 * 0.8 / 2 * (1 - SQRT (2 * (1.5625 * (var_2 / (t_2 * t_2) * 2 - 0.04) * 2) - 1)))
  28: [CEIL (1 / (var_2 / (t_2 * t_2))) = 3 AND var_2 / (t_2 * t_2) < 1]
     IF NOT p1 > 0 FROM temp_12_ PRIO 0
  29: [CEIL (1 / (var_2 / (t_2 * t_2))) = 3 AND var_2 / (t_2 * t_2) < 1]
     FROM temp_12_ TO temp_2_ RATE 3 / t_2
  30: [CEIL (1 / (var_2 / (t_2 * t_2))) = 3 AND var_2 / (t_2 * t_2) < 1]
     IF NOT p1 > 0 FROM temp_13_ PRIO 0
  31: [CEIL (1 / (var_2 / (t_2 * t_2))) = 3 AND var_2 / (t_2 * t_2) < 1]
     FROM temp_13_ TO temp_12_ RATE 2 / (t_2 * 0.666667 * (1 + SQRT (2 * (2.25 * (var_2 / (t_2 * t_2) - 0.111111)) - 1)))
  32: [CEIL (1 / (var_2 / (t_2 * t_2))) = 3 AND var_2 / (t_2 * t_2) < 1]
     IF temp_12_ < 1, temp_13_ < 1 TO temp_13_ RATE 2 / (t_2 * 0.666667 * (1 - SQRT (2 * (2.25 * (var_2 / (t_2 * t_2) -
     0.111111)) - 1)))

Pictures:
  Picture "average number of tokens in p1":
    Parameter: mue_1
    Curve: avg_p1
    X Axis Label: "arrival rate"
    Y Axis Label: "average number of tokens in p1"

Rewards:
  MEAN (p1)
```

Description: In lines 14 to 27 we can see, that 14 nodes are generated by the pre-processor for the relay circuits needed. The nodes `temp_0_` and `temp_1_` are used for the GE substitution nodes **n2** and **n1** (→ Fig. 3.6, p. 71). The node `temp_2_` is equivalent to node **nr** (→ Fig. 3.7, p. 72) and is used by both phase type substitutions that will be generated: one with 10 stages (SCV = 0.1) and one with 3 stages (SCV = 0.4). The remaining 9 nodes for SCV=0.1 are `temp_3_` (= **n9**) to `temp_11_` (= **n1**). The remaining 2 nodes for SCV=0.4 are `temp_12_` (= **n2**) and `temp_13_` (= **n1**).

Source rule 1 (lines 32-33) is equivalent to the exponential transition **t1** in Figure 3.8. It is not touched by the pre-processor. It is allowed in every parameter setting and therefore does not need an allowance condition.

Source rule 2, which substitutes the empirical distribution **t2** (→ Fig. 3.8, p. 72), now consists of 32 different sub-rules. In the model dump, each sub-rule is preceded by its allowance condition (in square brackets "[]"). For example, sub-rule 1 (lines 35 and 36) describes an exponential transition which is only used if the SCV is equal to one. For the use in the allowance condition (e.g. line 35), the SCV ($= \frac{\sigma^2}{\bar{t}^2}$) is rounded to 10 digits after the decimal point to exclude errors that arise because of the internal representation of floating point values. The sub-rules 2 to 7 (lines 37-48) are only used to build the GE substitution of Figure 3.6 if the SCV is greater than one. Sub-rule 8 (lines 49 and 50) is used for both PH substitutions (10 and 3 stages) if the SCV is smaller than one. The 10-stage PH substitution is built by the sub-rules 9 to 27 (lines 51-88) and the 3-stage PH substitution is built by the sub-rules 28 to 32 (lines 89-98).

The tool-specific input file: MOSEL-2 generates a single tool-specific input file for each setting of parameters. According to the allowance conditions, in each input file only elements are used, that are needed for the current setting.

The results: The results of the above originally non-Markovian model were obtained within 9.270s/1.960s/11.180s (user/system/real) using MOSEL-2 with numerical analysis provided by SPNP 6.1 and are presented in Figure 3.9.

In Figure 3.10 the results for a different set of parameters is shown. The values of the empirical distribution's SCV are now chosen closer to one. The numerical analysis with MOSEL-2 and SPNP took 4.120s/1.080s/5.140s (user/system/real).

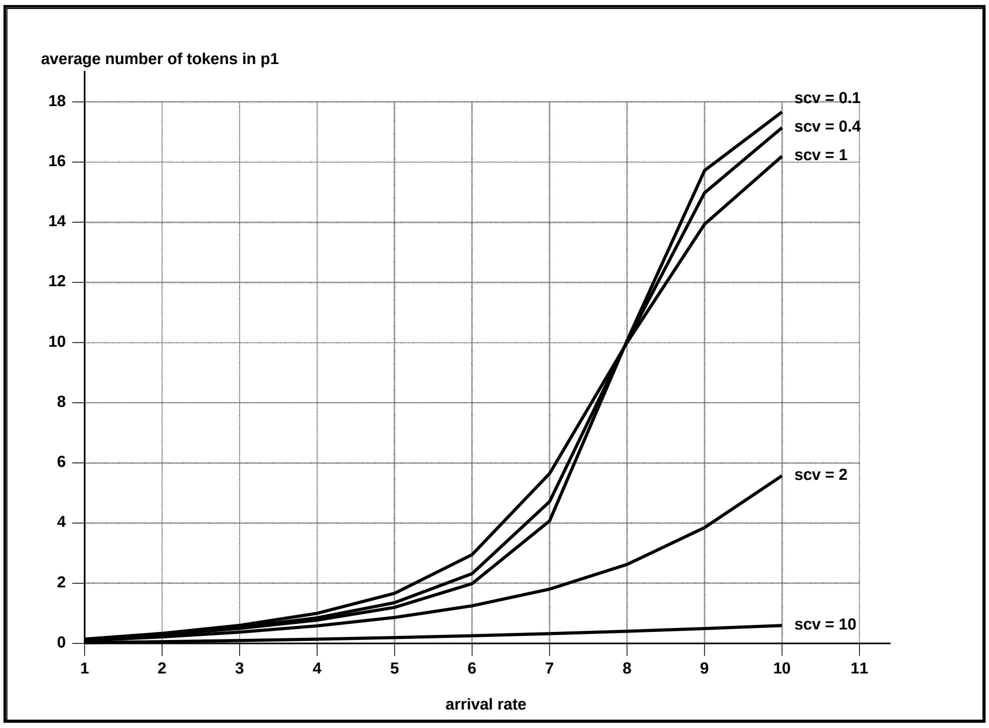

Figure 3.9: Results of example with EMP distribution.

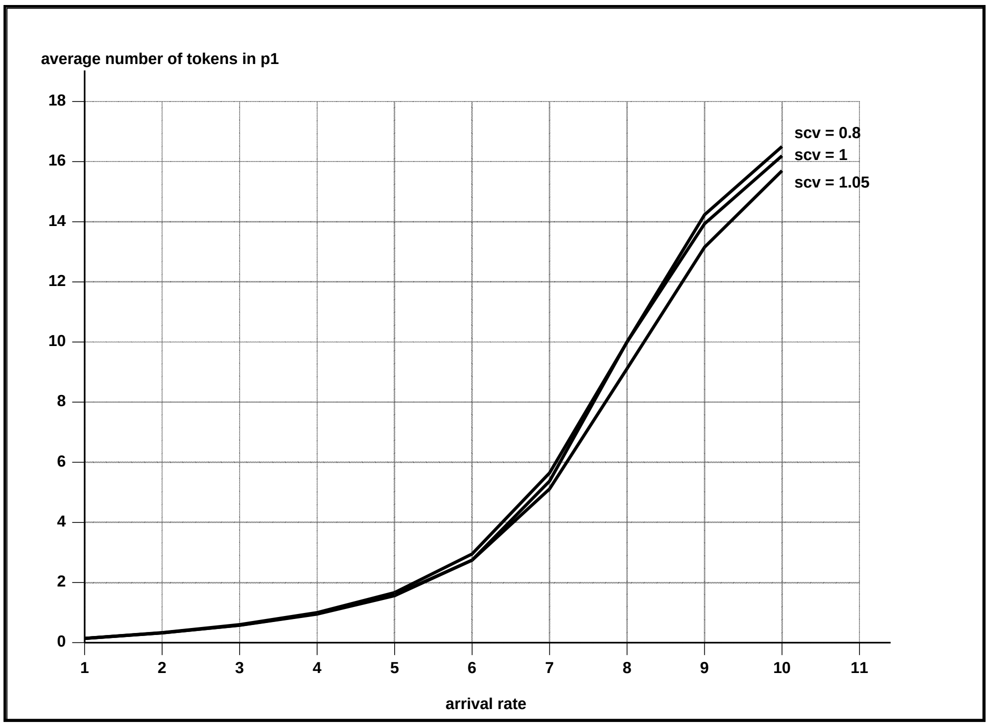

Figure 3.10: Results of example with EMP distribution.

3.2.4 Summary

MOSEL-2 is now able to pre-process rules that contain empirical distributions, that are defined by their mean and their variance. Empirical distributions are approximated automatically by substituting them using exponential, general exponential or phase type constructs. All these constructs only contain Markovian distributions and therefore can be solved using numerical analysis.

The pre-processor can be extended for generating further phase-type constructs, for example to enable numerical analysis of Cox, hyperexponential, hypoexponential or Erlang distributions, without forcing the user to specify these constructs manually. Some useful hints for the implementation are given in Appendix A.

The pre-processor allows the usage of arbitrary positive parameters for the mean and the variance. Nevertheless, the smaller the SCV is, the more stages are needed for the phase-type substitution. Therefore, the model size increases significantly. This is no big problem for MOSEL-2, but for the underlying tools.

For example, a simple MOSEL-2 model with a SCV of 0,0001 was pre-processed and translated to CSPL by MOSEL-2 within about 45 minutes using about 700 MB of RAM. The resulting CSPL file had a size of about 2 MB. It contained over 10000 nodes and as many transitions.

Unfortunately, SPNP cannot handle such large models. Tests showed, that SPNP is able to solve a simple model containing a single EMP distribution with SCVs down to 0.004675 (214 generated nodes) and that TimeNET can solve the same model with SCVs down to 0.017245 (58 generated nodes).

MOSEL-2 itself does not restrict the parameters, because the smallest working SCV strongly depends on the complexity of the entire model and on the evaluation tool chosen. A possible check of the SCV could be introduced in line 1739 of rule_pp.c. As a rule-of-thumb we can say, that we should avoid SCVs smaller than 0.05, but a fixed lower limit with this value would be too restrictive.

Chapter 4

Using improved MOSEL-2

We will now use the new MOSEL-2 version 2.10 to finally evaluate non-Markovian models of our GSM/GPRS/DOVE system with DeTVoC.

4.1 Simulating non-Markovian GSM/GPRS Model

At first, we will simulate the non-Markovian model introduced in Chapter 2.2, now using MOSEL-2 with SPNP 6.1 simulation. The results are obtained using 100 simulation runs with a simulation length of 20000 each within 85.600s/0.160s/85.740s (user/system/real).

Unfortunately, this results show very strange behaviour for the voice call loss probability, which takes negative values:

```
Simulation of "thesis_2_2_4.mos" by SPNP
(length: 20000, runs: 100)

Parameters:
  v_ar = 0.005
Results:
  v_loss = -0.59998
  d_loss = 0.24
  p3_full = 0
========================================
Parameters:
  v_ar = 0.007
Results:
  v_loss = -0.285713
  d_loss = 0.59
  p3_full = 0
========================================
Parameters:
  v_ar = 0.009
Results:
  v_loss = -0.66667
  d_loss = 0.89
  p3_full = 0
========================================
[...]
```

Evaluation			Model Configuration		Result
Tool	Method	steady-st./trans.	DOVE	DeTVoC	MEAN(p1)
SPNP	DES (20000/100)[1]	transient (20000)[2]	20×`AFTER 7`	20×`AFTER 5..10`	0.79999
SPNP	DES (1000/100)	transient (1000)	20×`AFTER 7`	20×`AFTER 5..10`	0.800015
SPNP	DES (1000/1000)	transient (1000)	20×`AFTER 7`	20×`AFTER 5..10`	0.686
SPNP	DES (10000/10000)	transient (10000)	20×`AFTER 7`	20×`AFTER 5..10`	0.7142
SPNP	num. analysis	steady-state	20×`RATE 1/7`	20×`RATE 1/7.5`	0.499932
SPNP	num. analysis	transient (1000)	20×`RATE 1/7`	20×`RATE 1/7.5`	0.499909
SPNP	DES (1000/1000)	transient (1000)	20×`RATE 1/7`	20×`RATE 1/7.5`	0.715
SPNP	num. analysis	steady-state	`RATE 1/7*p3`	`RATE 1/7.5*p3`	0.499932
SPNP	num. analysis	transient (1000)	`RATE 1/7*p3`	`RATE 1/7.5*p3`	0.499909
SPNP	DES (1000/1000)	transient (1000)	`RATE 1/7*p3`	`RATE 1/7.5*p3`	0.715
SPNP	DES (20000/20000)	transient (20000)	`RATE 1/7*p3`	`RATE 1/7.5*p3`	0.7045
TimeNET	num. analysis	steady-state	`RATE 1/7*p3`	`RATE 1/7.5*p3`	0.499927
TimeNET	DES	steady-state[3]	`RATE 1/7*p3`	`RATE 1/7.5*p3`	0.499952
TimeNET	DES	steady-state[3]	20×`RATE 1/7`	20×`RATE 1/7.5`	0.498936
TimeNET	DES	steady-state[3]	20×`AFTER 7`	20×`RATE 1/7.5`	N/A[4]

Table 4.1: Investigation of MEAN(p1).

Remember, that the `v_loss` was calculated using:

```
v_loss := 1 - ((v_sr * MEAN(p1)) / v_ar);
```

This term can only be negative, if `(v_sr * MEAN(p1)) / v_ar`, i.e. the probability that a voice connection request is accepted, is greater than one. Then the mean arrival rate `v_ar` is lower than the mean service rate `v_sr` – which theoretically can not happen. Because `v_sr` and `v_ar` are values specified by the user, the problem has to be the simulation result of `MEAN(p1)`. `MEAN(p1)` should always be less than $\frac{v_a r}{v_s r}$.

To get an idea, why this error happens, we will now check the behaviour of `MEAN(p1)` for `v_sr=0.01`, `v_sr=0.005`, several evaluation methods and model configurations. Theoretically, `MEAN(p1)` should always be less than `0.5`. The results of this investigation are shown in Table 4.1.

Apparently, SPNP 6.1 simulation has problems to provide correct results in this configuration. Unfortunately, neither a reasonable cause nor an adequate workaround for this faulty behaviour could be found in time.

[1](`SIM_LENGTH/SIM_RUNS`)
[2](time point of transient evaluation)
[3]but upper real-time limit of 10 minutes
[4]same error as in Chapter 2.2.5

4.2 Using Empirical Distributions

The failure of the simulation methods stress the importance of numerical analysis. We will now try to obtain results, if we substitute the deterministic (DOVE, **t7**) and uniform (DeTVoC, **t6**) distributions in our last GSM/GPRS model (→ Chap. 2.2, p. 51) by the new empirical distribution. For the mean value, i.e. the first parameter of the empirical distribution, we will choose the deterministic firing time or the mean of the lower and upper bound of the uniformly distributed firing time, respectively. The second parameter, i.e. the SCV will be chosen smaller than one, because the deterministic and uniform distribution show less variability than an exponential distribution:

The MOSEL-2 model description is changed as follows:

```
/** CONSTANT AND PARAMETER PART ***********************************************/
// GSM user behaviour
PARAMETER v_ar := 0.005 .. 0.015 STEP 0.002;  /* mean voice call arrival rate */
CONST     v_ht := 100;                        /* mean voice call holding time */
CONST     v_sr := 1/v_ht;                     /* mean voice call service rate */
CONST     v_dt_min := 5;                   /* minimum voice call waiting time */
CONST     v_dt_max := 10;                  /* maximum voice call waiting time */
CONST     v_dt_avg := (v_dt_min+v_dt_max)/2;  /* mean voice call waiting time */
CONST     v_dt_scv := 0.1;      /* low variability of voice call waiting time */
CONST     v_dt_var := v_dt_scv * v_dt_avg * v_dt_avg;             /* variance */

// GPRS data behaviour
CONST     d_ar := 0.7;                        /* mean data burst arrival rate */
CONST     d_sz := 10 * 8;                      /* mean data burst size (kbit) */

// network configuration
CONST freq_c      := 1;                        /* carrier frequencies in cell */
CONST chans_f     := 8;                        /* physical channels per freq. */
CONST chans       := freq_c*chans_f;             /* overall numb. of channels */
CONST chans_sig   := 2;                       /* channels used for signalling */
CONST chans_use   := chans-chans_sig;         /* channels available for users */
CONST chans_gpr   := 0;                         /* channels reserved for GPRS */
CONST chans_gsm   := chans_use - chans_gpr;   /* max. numb. of voice channels */
CONST gprs_max    := 32 * freq_c;             /* maximum number of GPRS users */
CONST pdch_dr     := 21.4;                     /* data rate per PDCH (kbit/s) */
CONST burst_sr    := pdch_dr/d_sz;             /* burst service rate per PDCH */
CONST dove_dt     := 7;                                /* delay time for DOVE */
CONST dove_dt_scv := 0.01;         /* very low variability of DOVE delay time */
CONST dove_dt_var := dove_dt_scv * dove_dt * dove_dt;             /* variance */
CONST dove_res    := 1;   /* number of PDCHs saved from immediate pre-emption */
CONST dove_vch    := chans_use - dove_res;      /* always available for voice */

/** NODE PART *****************************************************************/
NODE p1[chans_gsm]      := 0;                                          /* p1 */
NODE p2[gprs_max]       := 0;                                          /* p2 */
NODE p3[20]             := 0;                                          /* p3 */

/** RULE PART *****************************************************************/
FROM EXTERN TO p3 RATE v_ar;                                           /* t1 */
FROM p1 TO EXTERN RATE v_sr * p1;                                      /* t2 */
FROM EXTERN TO p2 RATE d_ar;                                           /* t3 */
FROM p2 TO EXTERN RATE burst_sr * (chans_use - p1);                    /* t4 */
IF (p1 < dove_vch) FROM p3 TO p1;                                      /* t5 */

@<1..20>{                                                              /* t6 */
 IF p3 >= # FROM p3 TO EXTERN EMP (v_dt_avg, v_dt_var);                /* t6 */
}                                                                      /* t6 */

@<1..20>{                                                              /* t7 */
 IF p3 >= # FROM p3 TO p1 EMP (dove_dt, dove_dt_var);                  /* t7 */
}                                                                      /* t7 */

/** RESULT PART ***************************************************************/
PRINT v_loss  := 1 - ((v_sr * MEAN(p1)) / v_ar);     /* voice call loss prob. */
PRINT d_loss  := PROB(p2==gprs_max) + PROB(p1==chans_use); /* data burst loss */
PRINT p3_full := PROB(p3==20);

/** PICTURE PART **************************************************************/
PICTURE "call and burst loss probability"
PARAMETER v_ar
CURVE d_loss
CURVE v_loss
XLABEL "voice call arrival rate"
YLABEL "call and burst loss probability"
```

The use of empirical distributions with low SCV within loop constructs is very dangerous because the final model will be very large.

For example, the translation of this model description to CSPL by MOSEL-2 takes 42.290s/0.070s/42.350s (user/system/real). The six CSPL files have a size of about 1 MB each. To compile one CSPL file, it takes SPNP more than 20 minutes (real). SPNP is neither able to solve the compiled model:

```
SPNP Version 6.1
The analysis is starting.
EXIT: hash table overflow, too many symbols
make: *** [all] Error 1
```

As a workaround, we could think of exploiting the quasi-Markov property of the empirical distribution's substitution construct. As we know, the PH substitution uses the reciprocal of the mean firing time as parameter for exponential distributions. Therefore, we can divide the mean firing time by the number of tokens in place **p3** to mimic the infinite server transitions **t6** and **t7** instead of using the loop constructs:

```
[...]
FROM p3 TO EXTERN EMP (v_dt_avg/p3, v_dt_var/(p3*p3));              /* t6 */
FROM p3 TO p1 EMP (dove_dt/p3, dove_var/(p3*p3));                   /* t7 */
[...]
```

Note, that the variance is $\sigma^2 = c^2 \cdot \bar{t}^2$. Therefore, if we divide the mean by `p3`, we have to divide the variance by $\texttt{p3}^2$.

But unfortunately, the use of state dependent expressions as parameter for the empirical distribution is prohibited, because in general this would result into a state dependent SCV and therefore would require state dependent substitutions for the empirical distribution, which is not implementable with adequate effort.

Therefore, we will restrict ourselves to the investigation of the model with empirical distributed DeTVoC. The DOVE-delay will be modelled as exponential distribution again:

```
[...]
@<1..3>{                                                             /* t6 */
 IF p3 >= # FROM p3 TO EXTERN EMP (v_dt_avg, v_dt_var);              /* t6 */
}                                                                    /* t6 */
FROM p3 TO p1 RATE 1/dove_dt * p3;                                   /* t7 */
[...]
```

The results will prove, that a capacity of 3 for place **p3** still is enough.

The steady-state results are obtained within 1948.860s/5.340s/32min33.960s (user/system/real) by using MOSEL-2 with SPNP's numerical analysis and are shown in Figure 4.1.

In the textual output file we see, that the probability that place **p3** is full, is very low ($< 10^{-6}$). Thus, a capacity of 3 is enough.

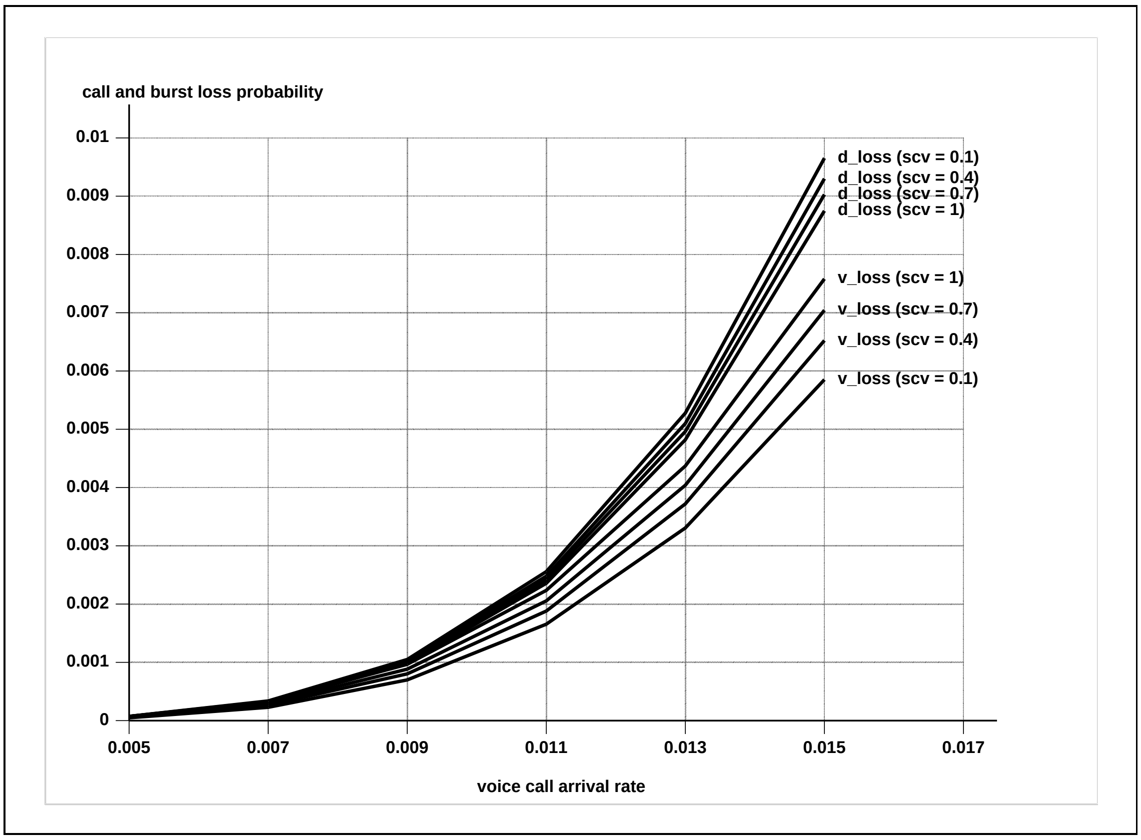

Figure 4.1: GPRS/DeTVoC with DOVE: voice call and data loss probability.

The graphical output shows, that if we aim at balancing the probability for voice and data loss, the assumption, that the user waiting time (DeTVoC) is exponentially distributed (SCV=1), is too optimistic. The variability has a fair effect on the loss probabilities and therefore should definitively be considered. The assumption, that exponential distributions can be generally used for appropriate modelling of firing times is certainly not applicable.

Chapter 5

Conclusion and Future Work

The final chapter of this thesis will give a brief summary of what has been achieved and which lessons have been learned. Furthermore, motivation for future work is given.

5.1 What has been achieved?

In this thesis, MOSEL-2 2.1 was used to model the air-interface of a single GSM/GPRS cell under consideration of delay tolerant voice calls (DeTVoC). The investigations aimed on optimising the call admission control (CAC) algorithm to assure user contentment despite increasing utilisation of the system.

Unfortunately, only relative investigations could be done, because – as all too often – real measured data was not available. A closer co-operation with mobile technology companies and/or access to equipment for measuring real mobile traffic would be preferable. Then the practical usefulness of this and related work would be more noticeable.

Speaking of practical work: A lot of time was invested in coding improvements to MOSEL-2, like the handling of SPNP 6.1 Simulation and the totally new rule pre-processor. An overall number of more than 3000 lines of C-code were developed or changed – not counting unfruitful approaches. Successful operating outcomes often belie the actual time exposure. The extensive search for self-inflicted errors was very time-consuming.

But also third party faults can be very time-consuming, for example by investigating why some evaluation tool would not produce reasonable or not even any results for a special setting of parameters. Several simulation runs took hours, just to find out, that the results cannot be used.

Nevertheless, it could be proven, that the MOSEL-2 language is definitively capable for modelling the behaviour of complex systems. These models can also be translated by the MOSEL-2 environment to the tool specific input files without greater difficulties. The evaluation power of MOSEL-2 instead heavily depends on the abilities and on the reliability of the used evaluation tool.

5.2 What has still to be done?

The MOSEL-2 models developed in this thesis could be quite easily extended for evaluating the GPRS data rate, or for considering handovers, several frequencies or retrials. These aspects could not be investigated in this thesis due to lack of time.

Numerical analysis methods always seemed to be more reliable and definitively faster than discrete event simulation. In future computer systems more and more memory will be available for the state space generation and more and more algorithms are developed to allow the numerical analysis of non-Markovian models. Therefore, it will be desirable, to make the arising tools available to MOSEL-2 by providing appropriate translators.

The numerical analysis of non-Markovian distributions is also the goal of the rule pre-processor presented in this thesis. The pre-processor can be extended quite easily to handle further phase-type distributions like the Erlang (→ Chap. A, p. 85), hyperexponential, hypoexponential or Cox distribution.

A minor task would be to enhance the result parser to parse the simulation confidence intervals and provide this information in the result file.

We have seen, that the result measures of some evaluation tools might not provide support for complex terms. Perhaps it would be suggestive, to rewrite MOSEL-2's tool specific input file generators (i.e. model translators) and especially the result parser in such a manner, that only the basic result measures (MEAN, PROB and UTIL) are requested from the evaluation tool. MOSEL-2's result parser could then use the results of this basic measures to compute the complex results that were desired by the user. This would also make the introduction of new evaluation tools to MOSEL-2 easier.

Finally, we can only hope, that the MOSEL project will not die when it can no longer be maintained by the ANA Group at the University of Erlangen.

Appendix A

How to insert new Distributions to the Rule Pre-Processor

This chapter will give some hints on how to insert a new pre-processor distribution. One of the easiest phase-type distributions, the Erlang distribution, will be taken as example. The given source code line numbers are suggestions, where the according changes should be done or where similar constructs are already implemented, and refer to MOSEL-2 version 2.10.

1. At first, we have to choose a keyword for the new distribution. We could take `ERL` for instance, because `ERLANG` is already used by a non-pre-processed distribution that can be evaluated using SPNP 6.1 simulation.
2. Then we have to consider the parameters, this distribution will need. In our case it will be the rate μ of each state and the number of states k.
3. We prefer "`ERL (`μ`, `k`)`" as the according MOSEL-2 syntax for the new distribution.
4. The new keyword has to be introduced to the scanner (scanner.l:134) and to the parser (parser.y:1367).
5. We have to insert the new distribution type `DIST_PP_ERL` to model.h (line 150).
6. When parsing a rule containing the ERL distribution, we have to call function `set_distrib` with parameter `DIST_PP_ERL` and set the variable `pp_state` to `PP_USED` as it is done for the EMP distribution (parser.y:1597-1601).
7. Similar to `DIST_PP_EMP` (parser.y:758-765) we have to check the parameters of our new distribution. μ has to be positive and k has to be a positive integer. We can use the functions `expr_is_positive` (parser.y:172-185) and `expr_is_cardinal` (parser.y: 152-168) for this purpose.
8. If a rule now contains an "`ERL (`μ`, `k`)`"-part, the rule pre-processor is invoked.
9. Hint: for easier debugging, the rule pre-processor rule_pp.c provides different debug levels with different debug outputs. The debug output can be activated in line 34 and should be deactivated again before checking the source into the CVS repository.

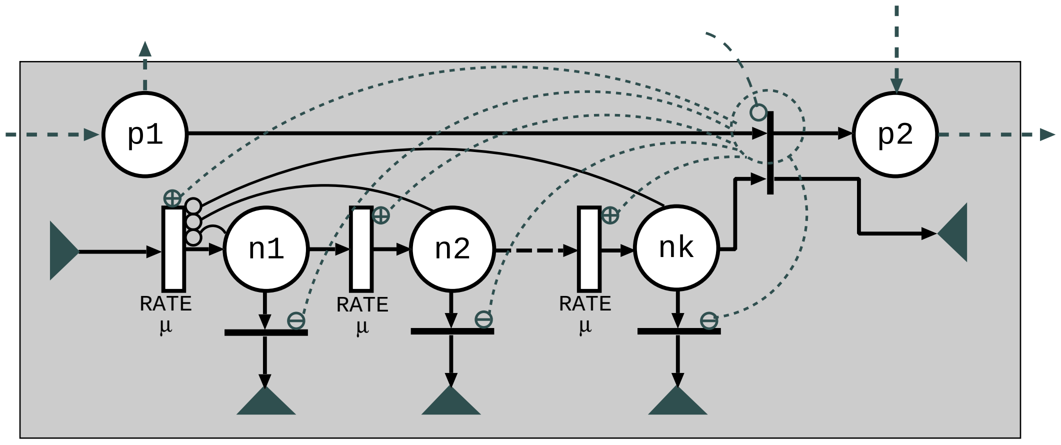

Figure A.1: Conceptual Model for Erlang Substitution.

10. The pre-processor has to recognise the new distribution type at first. This should be done similar to `DIST_PP_EMP` (rule_pp.c: 1900-1912). Instead of function `pp_emp` we will call the function `pp_erl`, that will be developed below.

11. The function `pp_erl` will be responsible for pre-processing the ERL distribution.

12. Similar to function `pp_emp` (rule_pp.c: 1687-1850) the function `pp_erl` will accept the three parameters `(pp_rule_t *pprule, int source_rule_index, int temp_node_count)` and will return the modified variable `temp_node_count`.

13. Now we will check all parameter settings for different values of k using a for-loop and a list for storing these values (cp. rule_pp.c: 1422-1449).

14. If the list contains more than one element, we have to provide each sub-rule with an appropriate allowance condition, which has to be constructed as MOSEL-2 expression by using the functions `new_bin_expr()`, `new_mon_expr()` and/or `new_value_expr()`, that are provided by expr.c.

15. Now we have to consider the architecture of the relay circuit that will mimic the Erlang distributed firing time by using only exponential distributions. This architecture is shown in Figure A.1.

16. The architecture of the Erlang Substitution is similar to the architecture of the EMP distribution's PH substitution (→ Fig. 3.7, p. 72) except that all rates are equal to μ. Therefore, the implementation is quite similar to the function `pp_dist_ph` (rule_pp.c: 1374-1683).

Appendix B

Abbreviations

1G	1st Generation of mobile (telephone) systems
2.5G	Generation 2.5 of mobile (telephone) systems
2G	2nd Generation of mobile (telephone) systems
3G	3rd Generation of mobile (telephone) systems
8PSK	8-Phase Shift Keying
AMPS	Advanced Mobile Phone System
ARQ	Automatic Repeat reQuest
BCMP	Baskett, Chandy, Muntz and Palacios
BTS	Base Transceiver Station
CAC	Call Admission Control
CB	Citizen Band
CDF	Cumulative Distribution Function
CDMA	Code Division Multiple Access
CDM	Code Division Multiplexing
CPU	Central Processing Unit
CS	Coding Scheme
CSPL	C-based Stochastic Petri net Language
CTMC	Continuous Time Markov Chain
D-AMPS	Digital Advanced Mobile Phone System
DCS	Digital Cellular System

DECT	Digital Enhanced Cordless Telecommunications
DES	Discrete Event Simulation
DeTVoC	Delay Tolerant Voice Call
DOVE	Delay Of Voice End-user
DSPN	Deterministic and Stochastic Petri Net
DTMC	Discrete Time Markov Chain
ECSD	Enhanced Circuit-Switched Data
EDGE	Enhanced Data (rates) for GSM/Global Evolution
EGPRS	Enhanced General Packet Radio Service
ESPN	Extended Stochastic Petri Net
ETSI	European Telecommunications Standards Institute
FCFS	First Come First Served
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FDM	Frequency Division Multiplexing
FEC	Forward Error Correction
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GSM	Global System for Global communication
GSMP	Generalised Semi Markov Process
GSPN	Generalised Stochastic Petri Net
HSCSD	High-Speed Circuit-Switched Data
ID	IDentification
IMT	International Mobile Telecommunications
IMTS	Improved Mobile Telephone System
IS	Infinite Server
ITU	International Telecommunication Union
LAN	Local Area Network
LCFS	Last Come First Served

MAN	Metropolitan Area Network
MCS	Modulated Coding Scheme
MOSEL	MOdelling, Specification and Evaluation Language
MTTF	Mean Time To Failure
PAN	Personal Area Network
PDCH	Packet Data Channel
PDC	Personal Digital Cellular
PDP	Packet Data Protocol
PMF	Probability Mass Function
PRD	Preemptive Repeat Different
PRI	Preemptive Repeat Identical
PR	Preemptive Resume
PRS	Preemptive ReSume
PS	Processor Sharing
QoS	Quality of Service
RAM	Random Access Memory
RFT	Remaining Firing Time
RR	Round Robin
SDMA	Space Division Multiple Access
SDM	Space Division Multiplexing
SDU	Service Data Unit
SIRO	Service In Random Order
SMP	Semi Markov Process
SOR	Successive Over-Relaxation
SPNP	Stochastic Petri Net Package
TACS	Total Access Communication System
TBF	Temporary Block Flow
TDD	Time Division Duplex
TDMA	Time Division Multiple Access

TDM	Time Division Multiplexing
TFI	Temporary Flow Identity
UMTS	Universal Mobile Telecommunication System
UTRAN	UMTS Terrestrial Radio Access Network
WAN	Wide Area Network
W-CDMA	Wide-band Code Division Multiple Access

Bibliography

[1] K. Begain, G. Bolch, H. Herold,
Practical Performance Modeling. Application of the MOSEL Language,
Kluwer Academic Publishers, 2001,
ISBN 0-7923-7951-9

[2] B. Beutel,
Integration of the Petri Net Analysator TimeNET into the Model Analysis Environment MOSEL,
University of Erlangen-Nürnberg, 2003,
http://www.linguistik.uni-erlangen.de/~bjoern/mosel-2.pdf (26.10.03)

[3] L. Wolf, J. Schiller, M. Zitterbart,
Vorlesung Mobilkommunikation,
Technische Universtiät Braunschweig, 2003,
http://www.ibr.cs.tu-bs.de/lehre/ss03/mk/ (27.10.03)

[4] J.J. O'Connor, E.F. Robertson,
Andrei Andreyevich Markov,
University of St Andrews, 1996,
http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Markov.html (20.04.04)

[5] ***Markov chain***,
in: *Wikipedia, the free encyclopedia*,
2004,
http://en.wikipedia.org/wiki/Markov_chain (20.04.04)

[6] C.G. Cassandras, S. Lafortune,
Introduction to Discrete Event Systems, Kluwer Academic Publishers, 1999, ISBN 0-7923-8609-4

[7] G. Bolch, S. Greiner, H. de Meer, K.S. Trivedi,
Queueing networks and Markov chains,
Wiley, 1998,
ISBN: 0-471-19366-6

[8] V.I. Romanovski,
Discrete Markov Chains,
Wolters-Noordhoff Publishing, 1970,
ISBN: 90-01-76185-2

[9] G. Bolch,
Leistungsbewertung von Rechensystemen mittels analytischer Warteschlangenmodelle,
B.G.Teubner, 1989,
ISBN: 3-519-02279-6

[10] K.S. Trivedi,
SPNP User's Manual. Version 6.0,
Duke University, Durham, USA, 1999,
http://www.ee.duke.edu/~chirel/MANUAL/manual.pdf (23.04.04)

[11] C.A. Petri,
Kommunikation mit Automaten,
in: *Schriften des Institutes für instrumentelle Mathematik*,
Bonn, 1962

[12] P. Wüchner,
Extending the Interface between the Modeling Languages MOSEL and CSPL by Adding Simulation Constructs,
University of Erlangen-Nürnberg, 2003,
http://www.xn–wchner-3ya.de/pub/MOSEL2-2_PW.pdf (25.04.04)

[13] I.R. Chen,
Stochastic Petri Net Models,
in: *Lecture Slides on Modeling and Evaluation of Computer Systems*, pp. 150-174,
Virginia Polytechnic Institute and State University, 2002
http://people.cs.vt.edu/~irchen/5214/pdf/p150-174.pdf (25.04.04)

[14] G. Bolch,
Modellierung und Leistungsbewertung von Rechensystemen,
University of Erlangen-Nürnberg, 2002,
http://www4.informatik.uni-erlangen.de/Lehre/WS02/V_MLR/Skript/ (26.04.04)

[15] I.N. Bronstein, K.A. Semendjajew, G. Musiol, H. Mühlig,
Taschenbuch der Mathematik,
Harri Deutsch, 1997,
ISBN: 3-8171-2003-6

[16] K. Park, W. Willinger,
Self-Similar Network Traffic: An Overview,
Purdue University, AT&T Labs-Research,
http://netlab.cs.tsinghua.edu.cn/~xuke/paperlist/intro-ss-chap.ps (28.04.04)

[17] J. Banks, J.S. Carson, B.L. Nelson, D.M. Nicol,
Discrete-Event System Simulation,
Prentice-Hall, 2001,
ISBN: 0-13-088702-1

[18] ***On the Simulation of Stochastic Petri Nets***,
Computer Science Department, College of William and Mary,
http://www.cs.wm.edu/~gli/Wpapers/PetriNet.ps (06.02.03)

[19] I.M. Sobol,
Die Monte-Carlo-Methode,
VEB Deutscher Verlag der Wissenschaften, 1971

[20] N.P. Buslenko, D.I. Golenko, Yu.A. Shreider, I.M. Sobol, V.G. Sragovich,
The Monte Carlo Method,
Pergamon Press, 1966

[21] B. Tuffin, K. S. Trivedi,
Implementation of Importance Splitting Techniques in Stochastic Petri Net Package,
Duke University, Campus universitaire de Beaulieu,
ftp://ftp.eos.ncsu.edu/pub/ccsp/papers/ppr0003.PS (05.05.04)

[22] P. E. Heegaard,
Rare Event Provoking Simulation Techniques,
The Norwegian Institute of Technology, 1995,
http://www.item.ntnu.no/ poulh/publications/its.pdf (05.05.04)

[23] P. Wüchner, K. Al-Begain, J. Barner, G. Bolch,
Modelling a single GSM/GPRS Cell with Delay Tolerant Voice Calls using MOSEL-2,
in: D. Al-Dabass (Ed.), *Proceedings of the UK Simulation Conference*, pp. 88-94,
Oxford, March 2004,
ISBN: 1-84233-099-3,
http://www.xn--wchner-3ya.de/pub/UKSim04_PW.pdf (05.05.04)

[24] A. Zimmermann,
TimeNET 3.0 - User Manual,
TU Berlin, 2001,
http://pdv.cs.tu-berlin.de/˜timenet/TimeNET-UserManual30.ps.gz (05.05.04)

[25] G. Bolch, S. Greiner, H. Jung, R. Zimmer,
The Markov Analyzer MOSES,
Universität Erlangen, 1994
http://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-94-10.pdf (05.05.04)

[26] A.S. Tanenbaum,
Computernetzwerke,
Pearson Studium, 2003,
ISBN: 3-8273-7046-9

[27] A.S. Tanenbaum,
Computer Networks,
Prentice-Hall, 1996,
ISBN: 0-13-394228-1

[28] M. Ermel, T. Müller, J. Schüler, M. Schweigel, K. Begain,
Performance of GSM networks with general packet radio service,
in: *Performance Evaluation*, 48, pp. 285-310,
Elsevier Science, 2002

[29] B. Mielke,
Dafu - Datenfunk in Deutschland,
http://www.dafu.de (07.05.04)

[30] B. Ghribi, L. Logrippo,
Understanding GPRS: The GSM Packet Radio Service,
in: *Computer Networks*, 34, pp. 763-779,
2000,
http://lotos.site.uottawa.ca/ftp/pub/Lotos/Papers/GPRS_Tutorial.pdf (07.05.04)

[31] K. Begain, M. Ermel, T. Müller, J. Schüler, M. Schweigel,
Analytical Call Level Model of a GSM/GPRS Network,
in: *Proceedings of SPECTS'2000*, Vancover, Canada,
July 2000,
http://141.30.128.24/ schweige/Paper/spects2000.pdf (08.05.04)

[32] ***EDGE – Introduction of high-speed data in GSM/GPRS networks***,
Ericsson AB, 2003,
http://www.ericsson.com/products/white_papers_pdf/edge_wp_technical.pdf (08.05.04)

[33] ***Nokia 6220 – Funktionen***,
Nokia, 2004,
http://www.nokia.de/de/mobiltelefone/modelluebersicht/6220/funktionen/41082.html (09.05.04)

[34] ***Nokia 6610i – Funktionen***,
Nokia, 2004,
http://www.nokia.de/de/mobiltelefone/modelluebersicht/6610i/funktionen/105702.html (09.05.04)

[35] M. Mahdavi, R.M. Edwards, S.R. Cvetkovic,
Policy for Enhancement of Traffic in TDMA Hybrid Switched Integrated Voice/Data Cellular Mobile Communications Systems,
in: *IEEE Communications Letter*, pp. 242-2440,
June 2001,
http://www.shef.ac.uk/eee/ecs/opnet/Mehdi_IEEE_Comm_letter.ps (10.05.04)

[36] T. Okuda, Y. Ando, T. Ideguchi, X. Tian,
A Simplified Evaluation for Delay of Voice End-User in TDMA Integrated Voice/Data Cellular Mobile Communications Systems,
in: *GLOBECOM 2002 - IEEE Global Telecommunications Conference*, 1, pp. 900-904
November 2002,
http://www.aichi-pu.ac.jp/ist/Staff/Chiiki/okuda200208/Pdf/lab/published/globecom2002.pdf (10.05.04)

[37] ***Overview of 3GPP Release 99***,
ETSI MCC, 2004,
http://www.3gpp.org/Releases/Rel99-Features-Draft.pdf (10.05.04)

[38] ***About 3GPP***,
3GPP, 2004,
http://www.3gpp.org/About/about.htm (10.05.04)

[39] ***3GPP Specification - Releases (and phases and stages)***,
3GPP, 2004,
http://www.3gpp.org/specs/releases.htm (10.05.04)

[40] ***3GPP Specification - Release contents and functionality***,
3GPP, 2004,
http://www.3gpp.org/specs/releases-contents.htm (10.05.04)

[41] K. Begain, G. Bolch, M. Telek,
Scalable Schemes for Mobile Networks with Multiple Service,
Universität Erlangen, 1998
http://www4.informatik.uni-erlangen.de/TR/pdf/TR-I4-98-04.pdf (13.05.04)

[42] J. Roszik, C.S. Kim, J. Sztrik,
Retrial Queues in the Performance Modeling of Cellular Mobile Networks using MOSEL,
in: *International Journal of Simulation: Systems, Science and Technology*,
(submitted 2003)

[43] J. Färber, S. Bodamer, J. Charzinski,
Statistical evaluation and modelling of Internet dial-up traffic,
in: *Proceedings of SPIE Photonics East '99 Conference on Performance and Control of Network Systems III*, pp. 112-121,
Boston, 1999,
http://www.ind.uni-stuttgart.de/Content/Publications/Archive/Fa_spie_30631.pdf (18.05.04)

[45] I. Adan,
Basic concepts from probability theory,
Technische Universiteit Eindhoven, 2003,
http://www.win.tue.nl/ iadan/sdp/h1.pdf (18.05.04)

[46] F. Barcelò, J. Jordàn,
Channel Holding Time Distribution in Public Cellular Systems,
in: *Proceedings of 16 th International Teletraffic Congress*, pp. 107-116,
Edinburgh, 1999,
http://citeseer.ist.psu.edu/barcel99channel.html (18.05.04)

[47] M. Garetto, D. Towsley,
Modeling, Simulation and Measurements of Queuing Delay under Long-tail Internet Traffic,
in: *SIGMETRICS'03*,
San Diego, California, June 2003,
http://www.telematics.polito.it/garetto/papers/p004-garetto.ps.gz (19.05.04)

[48] Y. Li, K. Al-Begain, I. Awan,
Performance Modelling of GSM/GPRS Mobile System with MOSEL,
in: *4th PGNet*, pp. 245-250,
Liverpool, June 2003,
http://www.cms.livjm.ac.uk/pgnet2003/submissions/Paper-30.pdf (19.05.04)

[49] V. Paxson, S. Floyd,
Wide-Area Traffic: The Failure of Poisson Modeling,
in: *IEEE/ACM Transactions on Networking*, 2(3), pp. 226 - 224,
June 1995,
http://www.maths.tcd.ie/~eoin/index/papers/paxson.floyd_failure95.pdf (19.05.04)

[50] S. Bodamer, J. Charzinki, K. Dolzer,
Dienstgütemetriken für elastischen Internetverkehr,
in: *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, 25(3), pp. 82 - 89,
2002,
http://www.ind.uni-stuttgart.de/Content/Publications/Archive/Bo_PIK2002_34653.pdf (19.05.04)

[51] A. Horvàth, M. Telek,
Approximating heavy tailed behaviour with Phase type distributions,
in: G. Latouche, P. Taylor (Eds.), *Advances in Algorithmic Methods for Stochastic Models, Notable Publications*, pp. 191 - 213,
2000,
http://www.di.unito.it/~horvath/publications/HoTe00.html (19.05.04)

[52] S.K. Bose,
QNAT – A Graphical Queueing Network Analysis Tool for General Open and Closed Queuing Networks,
2000,
http://home.iitk.ac.in/~skb/qbook/QNAT_Slides.PDF (20.02.04)

[53] R.D. Gupta, D. Kundu,
Generalized Exponential Distribution: Statistical Interference,
Technical Report, The University of New Brunswick, Saint John, 1999,
http://home.iitk.ac.in/~kundu/pap.pdf (20.02.04)

[54] R.D. Gupta, D. Kundu,
Generalized Exponential Distribution: Different Method of Estimations,
in: *Journal of Statistical Computation and Simulation*, 2001a, Vol. 69, pp. 315-338
http://home.iitk.ac.in/~kundu/paper62.pdf (20.02.04)

[55] A. Kouvatsos, N. Tabet-Aouel,
An ME-based approximation for multi-server queues with preemptive priority,
in: *European Journal of Operational Research* 77, pp 496-515,
Elsevier, 1994

[56] K. Al-Begain,
Using Subclass of PH Distributions for the Modeling of Empirical Activities,
in: *Proc. ICSCS 93, 18th International Conference on Statistics and Computer Sciences*,
pp. 141-152,
Cairo, 1993

[57] H. de Meer,
Transiente Leistungsbewertung und Optimierung Rekonfigurierbarer Fehlertoleranter Rechensysteme,
in: *Arbeitsberichte des Instituts für mathematische Maschinen und Datenbanken* 25 (10), Universität Erlangen, 1992

[58] G. Ciardo,
Toward A Definition of Modeling Power for Stochastic Petri Net Models,
in: *Proc. of the Int. Workshop on Petri Nets and Performance Models*,
Madison, August 1987
http://www.cs.wm.edu/~ciardo/pubs/1987PNPM-ModelingPower.pdf (28.05.04)

Index