

MODELLING A SINGLE GSM/GPRS CELL WITH DELAY TOLERANT VOICE CALLS USING MOSEL-2

P. WÜCHNER^{1,2}, K. AL-BEGAIN¹, J. BARNER², G. BOLCH²

¹ *Research Group on Mobile Computing and Networking, School of Computing, University of Glamorgan, Pontypridd, CF37 1DL, Wales, UK*

² *Research Group on Analytical Modelling, Institute for Distributed Systems and Operating Systems, Department of Computer Science, University of Erlangen-Nürnberg, Martensstr. 1, D-91058, Erlangen, Germany*

Abstract: In this paper we introduce the new version of the MOdelling, Specification and Evaluation Language – now called MOSEL-2 – and prove its applicability for performance modelling of mobile networks with non-Markovian models. Like its predecessor MOSEL [Al-Begain et al, 2001], MOSEL-2 was developed at the Institute for Operating Systems at the University of Erlangen-Nürnberg. In contrast to many specification languages of existing performance modelling and evaluation tools, which often tend to be too verbose, most MOSEL-2 specifications are compact but anyhow easy to understand. Moreover, MOSEL-2 provides means by which many interesting performance or reliability measures and the graphical presentation of them can be specified straightforwardly. It is especially easy to evaluate a model with different sets of system parameters. The benefit of MOSEL-2 – especially for the practitioner from the industry – lies in its modelling environment: A MOSEL-2 model is automatically translated into various tool-specific system descriptions and then analysed or simulated by the appropriate tools. This exempts the modeller from the time-consuming task of learning different modelling languages. The following tools and evaluation methods are currently supported by MOSEL-2: MOSES [Bolch et al, 1994] provides several iterative and direct methods (Jacobi, Gauss Seidel, LPU, Crout and Grassmann) for analysing Markovian models, SPNP [Hirel et al, 2000] provides several numerical methods for solving Markovian models and discrete event simulation for evaluating non-Markovian models like extended stochastic Petri nets (ESPNS) and TimeNET [Zimmermann et al, 1999] provides numerical methods for solving Markovian and a restricted class of models with non-exponentially distributed transitions and moreover is able to simulate extended deterministic stochastic Petri nets (eDSPNs). Nevertheless, currently more endeavours are made to include even more evaluation tools and pre-processor concepts to enhance the modelling and evaluation power of MOSEL-2. This Paper aims to give an introduction to the MOSEL-2 language and environment and describe how easily the air interface of a GSM/GPRS cell can be modelled.

Keywords: MOSEL-2, system description language, GSM/GPRS, DeTVoC, non-Markovian distributions, IGL

1 INTRODUCTION

Performance and reliability evaluation and prediction play a major role in the design, development, testing and maintenance of systems in many application areas. These include computer systems, communication systems and networks, manufacturing systems and various others. In many cases, the following properties are characteristic for the real-world systems under investigation: Their dynamic evolution proceeds from one discrete state to another at arbitrary moments in time. Often they are composed of smaller subsystems which run in parallel and cooperate in order to fulfil a common task; they can be distributed and may react to stimuli which are triggered by the system's environment.

One of the most frequently used techniques to model and evaluate the performance of such a system at the *dynamic* level is to represent it by a stochastic process

which – roughly speaking – consists of all system states and all possible transitions between them. For an important class of stochastic processes, known as Continuous Time Markov Chains (CTMCs) [Bolch et al, 1998], standard numerical algorithms can be used to compute the state probabilities.

For systems whose dynamics can be described by only a few states, an experienced modeller is able to deduce the underlying stochastic process directly by inspection. Unfortunately, this is impossible for many interesting real world systems since they tend to possess a large number of states at the stochastic process level. In this case, the investigator has to resort to a formal description technique (FDT) which is based on some mathematical theory and allows expressing some or all of the system properties mentioned above at a *static* level. Moreover, these FDTs facilitate the specification of the measures of interest themselves.

In the last decades, several modelling formalisms have been developed and used for performance and reliability modelling in various problem domains. Among these, Stochastic Process Algebras (SPAs) [Hermanns et al, 2000] and Stochastic Petri Nets (SPNs) with their numerous derivatives [Ciardo, 1993] such as General Stochastic Petri Nets (GSPNs), Deterministic and Stochastic Petri Nets (DSPNs) and Extended Stochastic Petri Nets (ESPNs), that permit arbitrary distributions of firing times of transitions, turned out to be especially useful as they allow a combined description of the functional and temporal system behaviour including the specification of phenomena like priorities, synchronisation and preemption. Queuing Network Systems [Mehdi, 2003] is another widely used modelling formalism which does not reach the expressiveness of ESPNs and SPAs but instead supports solution algorithms by which the desired performance measures can be calculated extremely fast.

ESPNs and SPAs are high-level modelling formalisms in which the modeller gives a static description of the system structure and behaviour using a textual (SPAs, ESPNs) or graphical (ESPNs) syntax. In order to evaluate the functional and temporal behaviour the system has to be either simulated or analysed depending on the ability to generate the whole state space.

For analysis, the set of dynamic system states – the state space – has to be generated out of the static model. This can only be done for Markovian models and is accomplished in GSPN and SPA formalisms by a set of semantic rules which is used to derive the complete state space from a start state given in the high-level description. Once the state space has been generated it can be automatically transformed into a stochastic process, which in most cases forms a CTMC. The underlying stochastic process can then be generated automatically from the static system description. The analysis of the system is then completed by solving the system of linear equations which is given by the generator matrix of the CTMC.

Fortunately, there exists a variety of (freeware and commercial) tools which are based on one of the high-level modelling formalisms mentioned above: SPNP [Hirel et al, 2000], TimeNET [Zimmermann et al, 1999], GreatSPN [Chiola et al, 1995] and WebSPN [Bobbio et al, 1998] are based on Petri Nets, whereas TIPP [Goetz et al, 1993] and PEPA [Hillston, 1993] are built around the SPA modelling formalism. Other packages, for example PEPSY [Kirschnick, 1994] and QNAP2 [Veran and Potier, 1985] favor the queueing theoretical approach, whereas SHARPE [Sahner et al, 1996] and MOSES [Bolch et al, 1994] use a mixture of methods towards the generation and solution of the underlying CTMC.

All these packages usually have their own textual or graphical specification language which depends largely on the underlying modelling formalism. The different syntax of the tool-specific modelling languages implies that once a tool has been chosen it will be difficult to switch to another one as the model has to be rewritten using a different syntax.

Starting from these observations the development of MOSEL [Herold et al, 2001] – and now MOSEL-2 [Beutel, 2003] – is based on the following idea: Instead of creating another tool with all the components needed for system description, state space generation, stochastic process derivation, and numerical solution or simulation, we focus on the formal system description part and exploit the power of various existing and well tested packages for the subsequent stages.

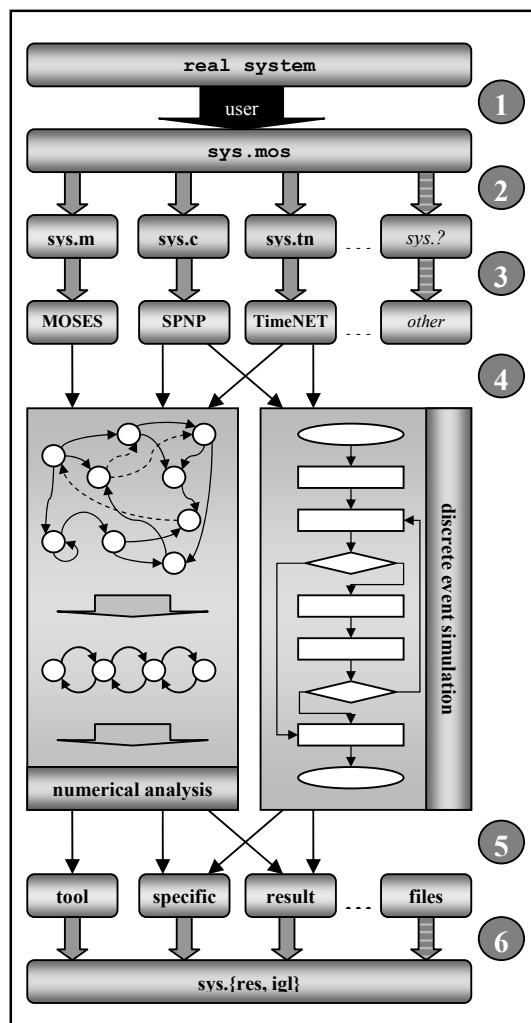


Figure 1: The modelling and analysis process in the MOSEL-2 environment.

MOSEL-2 is a modelling environment which comprises a high-level modelling language that provides a very simple way for system description. In order to reuse existing tools for the system analysis, the envi-

ronment is equipped with a set of translators which transform the MOSEL-2 model specification into various tool-specific system descriptions.

Figure 1 gives an overview of performance and reliability modelling and evaluation using the MOSEL-2 environment:

1. The modeller inspects the real-world system and generates a high-level system description using the MOSEL-2 specification language. He also specifies the desired performance and reliability measures using the syntax provided by MOSEL-2. He passes the model to the environment which then performs all following steps without user interaction.
2. The MOSEL-2 environment automatically translates the MOSEL-2 model into a tool-specific system description, for example a CSPL-file (C based Stochastic Petri net Language) suitable to serve as input for SPNP.
3. The appropriate tool (i.e. SPNP) is invoked by the MOSEL-2 environment.
4. The appropriate tool processes its input file in one of the two following ways:
 - a. Numerical analysis: Out of the static model description the whole state space of the model is generated by the tool according to the semantic rules of its modelling formalism. This semantic model is mapped onto a stochastic process. The stochastic process is solved by one of the standard numerical solution algorithms which are part of the tool.
 - b. Simulation: The model is evaluated by the tool without building the whole state space using discrete event simulation.
5. The results of the numerical analysis or discrete event simulation are saved in a file with a tool specific structure.
6. The MOSEL-2 environment parses the tool specific output and generates a textual result file (*sys.res*) containing the performance and reliability measures which the user specified in the MOSEL-2 system description. If the modeller requested graphical representation of the results, a second file (*sys.igl*) is generated by MOSEL-2.

At present the MOSEL-2 environment is able to translate models into system descriptions for the SPN-based packages SPNP and TimeNET and for the tool MOSES, whose model description language MOSLANG is a predecessor of MOSEL and MOSEL-2. The connection of other performance modelling packages to the MOSEL-2 environment is projected.

During the recent integration of TimeNET [Beutel, 2003] we revised and enhanced the MOSEL syntax – introducing MOSEL-2 – to support a set of non-exponential distribution types for the system state transitions. MOSEL-2 models containing non-Markovian

transitions can now be evaluated by the Discrete Event Simulation components of either SPNP [Wüchner, 2003] or TimeNET. Moreover TimeNET offers numerical solution methods for a restricted class of systems with non-exponentially distributed transitions.

The paper is organized as follows. In section 2 we give a brief introduction to the MOSEL-2 language by means of a simple example. In section 3, we model a single GSM/GPRS cell using non-markovian distributions. In section 4, we give a brief outlook onto future work enhancing the modelling and evaluation power of MOSEL-2.

2 MOSEL-2

In this section we explain the structure of the MOSEL-2 model specification by demonstrating the application of the language constructs and the evaluation environment using a simple queuing network model.

2.1 The System

As a simple example for a complete model specification in MOSEL-2, we consider an open tandem queuing network with two M/M/1-FCFS nodes:

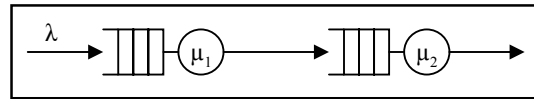


Figure 2: Open tandem queuing network

2.2 The MOSEL-2 Model

```

01 // Parameter declaration part
02
03 PARAMETER K := 1 .. 10;
04 CONST lambda := 0.25;
05 CONST mue1 := 0.28;
06 CONST mue2 := 0.22;
07
08 // Component definition part
09
10 NODE N1[K] := 0;
11 NODE N2[K] := 0;
12
13 // Transition definition part
14
15 FROM EXTERN TO N1, num RATE lambda;
16 FROM N1 TO N2 RATE mue1;
17
18 // Result part
19
20 PRINT rho1 := UTIL (N1);
21 PRINT rho2 := UTIL (N2);
22 PRINT throughput = rho2 * mue2;
23 PRINT WIP = MEAN (N1) + MEAN (N2);
24
25 // Picture part
26
27 PICTURE "utilisation"
28 PARAMETER K
29 CURVE rho1
30 CURVE rho2;

```

Every MOSEL-2 model consists of five parts. Each part is dedicated to the description of a different aspect of the performance and reliability model by means of appropriate language constructs [Al-Begain et al, 2001; Beutel, 2003]:

1. Parameter declaration part (optional):	01-06
2. Component definition part:	08-11
3. Transition definition part:	13-16
4. Result part:	18-23
5. Picture part (optional):	25-30

Parameter Declaration Part: In this part constants and variable system parameters can be declared. The ENUM construct allows the definition of a set of constants. The possibility of declaring a list of values in the PARAMETER construct is a powerful feature of the language, since this makes experimenting with the model very easy. The MOSEL-2 environment automatically analyses the specified system for any combination of system parameters. In the example above three constants are defined: the arrival rate lambda (line 04) and the two service rates mue1 (line 05) and mue2 (line 06). The system is analysed assuming that the maximum capacity of queues K varies from 1 to 10 (line 03). This implies that during the performance analysis the MOSEL-2 environment automatically generates a sequence of ten high-level system descriptions (e.g. CSPL-files) and as well invokes the appropriate tool (i.e. SPNP) for each variant. The results of the experiment (the set of analysis runs performed by the tool) are collected in a *single* result file by the MOSEL-2 environment. The same applies to the graphical representation of the results.

Component Definition Part: This part consists of the NODE constructs and the optional ASSERT statement:

The NODE constructs specify the components (nodes) of the model. Each node can hold an integer-valued number of jobs (or tokens) up to its capacity. For every component in the model a name and the capacity has to be given. In our example the two nodes N1 (line 10) and N2 (line 11) representing the queues are defined. Each node is capable of holding up to K jobs.

The ASSERT statement can be used to specify the prohibited system states. Most frequently, this is used to express the seclusiveness of the modelled system. For a closed system, we have to assure that the total number of tokens in the nodes which are used to model the job-flow is always constant. We do not need an ASSERT statement for our open tandem network example.

Transition Definition Part: In this part the transitions that determine the behaviour of the system are specified. In MOSEL-2 the transitions are also called rules. In the example above the transition part defines the possible transitions of jobs between the nodes using the rates specified in the parameter declaration part. The special node EXTERN (line 15) is used to model arrivals and departures of jobs from and to the environment.

Result Part: In this section of the system specification the modeller can define the performance and reliability measures which he is interested in. Several state dependent measures can be specified, e.g. the mean value MEAN, the utilization UTIL, the distribution of jobs DIST or the probability that a component of the system is in a certain state PROB. In our result part the utilisations (probability that the queue is not empty) of the nodes N1 (rho1) and N2 (rho2) are defined using the keyword UTIL (lines 20, 21). Moreover, the throughput of the second server is specified as the product of rho2 and the service rate mue2 (line 22). Finally the mean total number of jobs in the system (WIP) is requested (line 23). The keyword PRINT at the beginning of each line of the result part indicates that all specified results are going to be saved into the MOSEL-2 result file. Intermediate results, which often are used to compute complex measures and which need not be stored in the result file, can be specified by using the keyword RESULT instead of PRINT.

Picture Part: In this optional part the user can determine which values should be plotted. With special keywords the appearance of the curves can be changed and improved. We define a plot showing the utilisations rho1 (line 29) and rho2 (line 30) as a function of the varying system parameter K (line 28).

2.3 Analysing the Open Tandem Network using the MOSEL-2 Environment

Suppose that we have saved our MOSEL-2 model in a file named tandem.mos. We are now ready to invoke the MOSEL-2 environment on the command line via:

```
mosel2 -cs tandem.mos
```

The option *-c* denotes that we request that our MOSEL-2 specification should be translated into a set of CSPL files (C based Stochastic Petri net Language), which serve as input for the tool SPNP. Because we also use the *s* option (“start appropriate tool automatically”) the MOSEL-2 environment will now perform the rest of the analysis and result post-processing steps automatically using the SPNP package for state space generation, derivation of the underlying CTMC and numerical solution (cf. Figure 1).

Upon completion of the analysis, the MOSEL-2 environment creates two files named tandem.res and tandem.igl which contain the requested performance measures of the system in textual and graphical form. The textual result file is very helpful if we are interested in exact values of the performance measures for a specific value of the variable system parameter which was given in the model specification. The graphical representation of the results provides insight about how the performance measures depend on a variable system parameter over an interval. The

graphical result file can be viewed and post-processed with the IGL-utility, which is part of the MOSEL-2 environment. Please see section 3.4 and 3.5 for result file examples.

3 MODELLING A GSM/GPRS CELL WITH MOSEL-2

In this section we will use MOSEL-2 to evaluate a non-Markovian model of a single GSM/GPRS cell with Delay Tolerant Voice Calls (DeTVoC).

3.1 Real System

We consider the air interface of a single GSM/GPRS cell with a maximum of 100 users and a single carrier frequency. This carrier frequency is split up into eight TDMA channels. Every single channel can be used for a single circuit switched GSM voice or data over GSM connection or for one or several packet switched GPRS data connections. A single TDMA channel (without EDGE technology) can serve a gross data rate up to 21.4 Kbit/s [Ermel et al, 2002]. A single GPRS user can use a maximum of eight TDMA channels simultaneously and therefore can achieve a maximum gross data rate of 171.2 Kbit/s. Every GPRS data connection is identified by a temporary flow identity (TFI) of 5 bit length which limits the number of simultaneous GPRS connections to 32. We assume that one TDMA channel is reserved for GPRS traffic. All other TDMA channels can only be used by the GPRS system if they are not used by higher priority GSM voice (or data) connections. If a voice call cannot be served, the connection attempt is not rejected immediately but the user will wait up to a maximum of 10 seconds before giving up. We call this behaviour “*Delay Tolerant Voice Call*” (DeTVoC). If a data burst cannot be served, it gets rejected immediately and the transmission needs to be tried again later. We assume that voice calls arrive following a Poisson process with a rate of 1/100 per second and finish with a rate of 1/80 per second. For data traffic we assume an exponentially distributed burst rate of 0.1 to 2.5 bursts per second and a geometrically distributed burst size with a mean of 10 KB.

3.2 Conceptual Model

Figure 3 shows a Petri Net (DSPN) that describes the real system in a more formal way.

Description (times and rates refer to seconds):

- t1: Poisson voice call arrival process: exponential distributed transition with firing rate:
 $\lambda_v = 1/100$
- t2: Poisson data burst arrival process: exponential distributed transition with firing rate:
 $\lambda_d = 0.1 \dots 2.5$

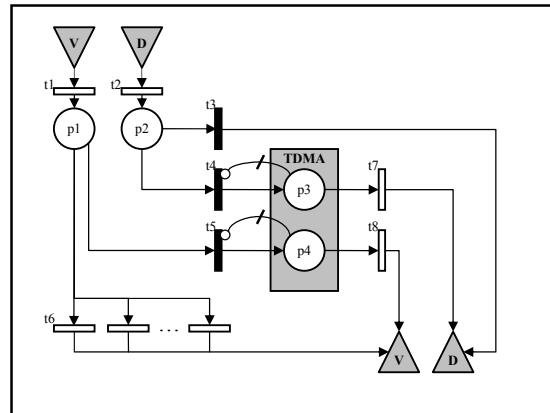


Figure 3: DSPN model of GSM/GPRS air interface.

- t3: Packet loss: immediate transition with priority 1 (low)
- t4: Packet admission: immediate transition with priority 2 (high) and inhibitor arc (multiplicity 32, because of TFI)
- t5: Call admission: immediate transition with inhibitor arc (multiplicity 7, because one channel is reserved for packet service) and priority 2 (high)
- t6: DeTVoC: IS (infinite server) transitions with deterministic firing time of 10, marking dependent enabling functions and priority 1 (low)
- t7: Packet service: PS (processor sharing) using exponential distributed transition with firing rate:
 $\mu_{e_d} = \frac{21.4}{8 \cdot 10} \cdot (8 - p4)$
- t8: Voice service: IS using exponential distributed transition with firing rate:
 $\mu_{e_v} = \frac{1}{80} \cdot p4$
- p1: place with capacity 100 for incoming (and waiting) voice connection requests
- p2: place with capacity 1 for incoming data connection requests
- p3: place with capacity 32 for bursts in service
- p4: place with capacity 7 for voice connections in service

3.2 MOSEL-2 Model

```

01 /* PARAMETER AND CONSTANTS *****/
02 PARAMETER lambda_d := 0.1, 0.5 .. 2.5 STEP 0.25;
03 CONST lambda_v := 1/100;
04 CONST mue_single_v := 1/80;
05 CONST max_users_v := 100;
06 CONST chan_res_d := 1;
07 CONST waiting_time := 10;
08 CONST mue_single_d := (21.4/8) / 10;
09
10 /* NODES *****/
11 NODE p1[max_users_v] := 0;
12 NODE p2[1] := 0;
13 NODE p3[32] := 0;
14 NODE p4[8-chan_res_d] := 0;
15
16 /* TRANSITIONS *****/
17 FROM EXTERN TO p1 RATE lambda_v; //t1
18 FROM EXTERN TO p2 RATE lambda_d; //t2
19 FROM p2 TO EXTERN PRIO 1; //t3
20 FROM p2 TO p3 PRIO 2; //t4
21 FROM p1 TO p4 PRIO 2; //t5

```

```

22 @<1..max_users_v>{ //t6
23 IF p1 >= # FROM p1
24     TO EXTERN
25     AFTER waiting_time
26     Prio 1;
27 }
28 FROM p3 TO EXTERN RATE mue_single_d * (8-p4); //t7
29 FROM p4 TO EXTERN RATE mue_single_v * p4; //t8
30
31 /* RESULTS *****/
32 // TEXTUAL (.res)
33 PRINT d_loss      := PROB (p3 == 32);
34 PRINT v_block     := UTIL (p1);
35
36 // GRAPHICAL (.igl)
37 PICTURE "prob. of data loss and voice blocking"
38 PARAMETER lambda_d
39 XLABEL "incoming burst rate"
40 YLABEL "PROB"
41 CURVE d_loss
42 CURVE v_block

```

Explanations: The lines 22 to 27 show the powerful loop construct of MOSEL-2. This pre-processor expression is introduced by the special character “@” that is followed by the range list(s) enclosed in angle brackets (“<>”) and by the body in curly brackets (“{}”). The body may contain the special character “#” that will be exchanged with the current range list item. Loops can be used anywhere in a MOSEL-2 model description.

The loop above is constructing the infinite server transitions of t6 and will be expanded like this:

```

IF p1 >= 1 FROM p1 TO EXTERN AFTER waiting_time Prio 1;
IF p1 >= 2 FROM p1 TO EXTERN AFTER waiting_time Prio 1;
[...]
IF p1 >= 100 FROM p1 TO EXTERN AFTER waiting_time Prio 1;

```

As we can see, loops are very useful to combine several code parts with similar structure. This keeps the model description concise and saves the user from excessive typing.

The IF parts of this rules describe the state dependent enabling function of each rule, i.e. for every single token in p1 a deterministic transition gets enabled.

3.3 Invoking the MOSEL-2 Environment

We invoke the MOSEL-2 environment from command line using the option +simulation to evaluate the non-Markovian model with SPNP’s discrete event simulation:

```

mosel2 -cso +simulation gsmgprs.mos

```

3.4 Textual Results

Below we show a compressed form of the result file gsmgprs.res:

```

Simulation of "gsmgprs.mos" by SPNP
(length: 1000, runs: 1000)
Parameters:      | Parameters:
  lambda_d = 0.1 | lambda_d = 0.5
Results:        | Results:
  loss_prob_d = 0 | loss_prob_d = 0.001
  block_prob_v = 0 | block_prob_v = 0.00200001
[...]
Parameters:      | Parameters:
  lambda_d = 2.25 | lambda_d = 2.5
Results:        | Results:
  loss_prob_d = 0.186 | loss_prob_d = 0.303
  block_prob_v = 0.001 | block_prob_v = 0.00200001

```

3.5 Graphical Results

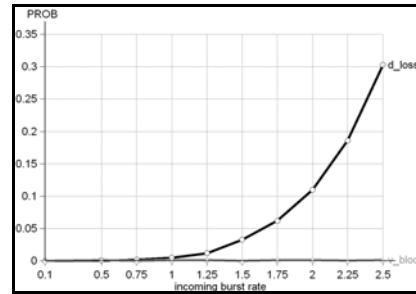


Figure 4: Voice-blocking and data-loss over burst rate

In Figure 4 – an IGL-generated plot of the voice-blocking and data-loss probabilities – we can see that the data loss probability increases fast with rising data arrival rate whereupon the probability of blocked calls of course is constant zero. Assuming, that a data loss of 5% can be tolerated, we can determine the maximum arrival rate of data bursts as 1.65 data bursts per second.

Now we will keep the arrival rate of data bursts constant at 1.5 burst per second and raise the arrival rate of voice calls from 0.001 to 0.1 calls per second. The results are shown in Figure 5.

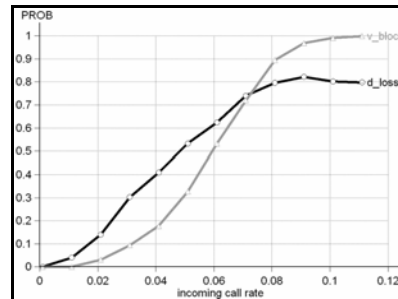


Figure 5: Voice-blocking and data-loss over call rate

The voice call blocking probability is rising fast and is allocating more and more TDMA channels. Therefore, the data-loss probability is rising too.

Assuming that a blocking probability for voice calls of 1% is still convenient, we see that a reservation of further TDMA-channels for data transmission would not be useful in this configuration, because the limit of the voice call blocking is exceeded earlier than the limit of the data loss. It would be even better to restrict the data traffic by using the reserved data channel for voice. This could be achieved with the DOVE concept [Mahdavi et al, 2001].

4 CONCLUSION AND FUTURE WORK

In this paper we have shown that on one hand it is very easy getting used to specify models in the MOSEL-2 language and that on the other hand the language is capable of describing rather complex systems. The evaluation power depends of course on the

evaluation tools that are used by the MOSEL-2 environment. Currently MOSES, SPNP and TimeNET provide the possibility of evaluating systems containing the following distributions of firing times with MOSEL-2:

- immediate, exponential
- deterministic (SPNP Simulation or TimeNET only)
- uniform (TimeNET Simulation only)
- beta, Cauchy, gamma, geometric, hyper-exponential, hypoexponential, lognormal, normal, Pareto, Poisson (SPNP Simulation only)

To enable numerical analysis and provide more tool independence while evaluating non-Markovian nets, we are currently implementing new pre-processor constructs to MOSEL-2 that will approximate general distributions (which are defined by their first two moments: mean and variance) by using general exponential and phase-type distributions [Al-Begain, 1993] which can be both described by using only sets of immediate and exponential distributions.

REFERENCE

- Al-Begain K. 1993, "Using Subclass of PH Distributions For The Modeling of Empirical Activities". In *Proc. ICSCS 93, 18th International Conference on Statistics and Computer Sciences*, Cairo, Egypt. Pp141-152.
- Al-Begain K., Bolch G. and Herold H. 2001, "Practical Performance Modeling – Application of the MOSEL Language". Kluwer Academic Publishers, Norwell, MA, USA. ISBN 0-7923-7951-9
- Chiola G., Franceschinis G., Gaeta R. Ribaldo 1995, "GreatSPN 1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets." In *Performance Evaluation*, 24(1-2). Pp137–159.
- Beutel B. 2003, "Integration of the Petri Net tool TimeNET into the MOSEL modelling environment". Diploma Thesis DA-14-2002-17, Department of Computer Science, University of Erlangen, Germany.
- Bobbio A., Puliafito A., Scarpa M. and Telek M. 1998, "WebSPN: Non-Markovian Stochastic Petri Net tool". In *Int. Conf. on WEB-based Modeling and Simulation*. (San Diego, USA, January).
- Bolch G., Greiner S., Jung H. and Zimmer R. 1994, "The Markov Analyzer MOSES". Technical Report TR-14-10-94, Department of Computer Science, University of Erlangen, Germany.
- Bolch G., Greiner S., de Meer H. and Trivedi K.S. 1998, "Queueing Networks and Markov Chains". John Wiley & Sons, New York, NY, USA. ISBN 0-471-19366-6
- Ciardo G., German R., and Lindemann C. 1994, "A characterization of the stochastic process underlying a stochastic Petri net". In *IEEE Trans. Softw. Eng.*, 20(7). Pp506-515.
- Ermel M., Müller T., Schüler J., Schweigel M. and Begain K. 2002, "Performance of GSM networks with general packet radio services". In *Performance Evaluation*, 48. Pp285-310.
- Götz N., Herzog U., Rettelbach M 1993, "TIPP – Introduction and application to protocol performance analysis". In König H. (editor) *Formale Beschreibungstechniken für verteilte Systeme* (GI/ITG-Fachgespräch, Magdeburg, 10.-11. Juni 1992). K. G. Saur Verlag, Germany. ISBN 3-598-22409-5
- Hermanns H, Herzog U. and Katoen J.P. 2000, "Process algebra for performance evaluation". In *Theoretical Computer Science*, 24(1-2). Pp43–87.
- Hillston J. 1993, "PEPA: Performance Enhanced Process Algebra". Technical Report CSR-24-93, University of Edinburgh, UK.
- Hirel C., Tuffin B., Trivedi K.S. 2000, "SPNP: Stochastic Petri Nets. Version 6.0". Duke University, Durham, NC, USA.
- Kirschnick M. 1994, "The Performance Evaluation and Prediction System for Queueing Networks (PEPSY-QNS)". Technical Report TR-14-18-94, Department of Computer Science, University of Erlangen, Germany.
- Mahdavi M., Edwards R.M., Cvetkovic S.R. 1993, "Policy for Enhancement of Traffic in TDMA Hybrid Switched Integrated Voice/Data Cellular Mobile Communications Systems". In *IEEE Communication Letters*, 5(6). Pp242-244.
- Mehdi J. 2002, "Stochastic Models in Queueing Theory". Academic Press, 2nd edition (November 2002), New York, NY, USA. ISBN 0-1248-7462-2
- Sahner R., Trivedi K. and Puliafito A. 1996, "Performance and reliability Analysis of Computer Systems. An example-based approach using the SHARPE software Package". Kluwer Academic Publishers, Norwell, MA, USA. ISBN 0-7923-9650-2
- Veran M. and Potier D. 1985, "QNAS 2: A portable environment for queueing system modelling". In *Proc. Int. Conf. on Modelling Techniques and Tools for Performance Analysis*. Pp25-32.
- Wüchner P. 2003, "Extending the Interface between the Modeling Languages MOSEL and CSPL by Adding Simulation Constructs". Semester Thesis SA-14-2003-06, Department of Computer Science, University of Erlangen, Germany.
- Zimmermann A., Freiheit J., German R., Hommel G. 1999, "Petri Net Modelling and Performability Evaluation with TimeNET 3.0". In *11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'2000)*, (Schaumburg, Illinois, USA, March 2000). Pp188-202.