Towards an Efficient Performance Evaluation of Communication Systems Described by Message Sequence Charts

Hesham Kamal Arafat Mohamed and Bruno Müller-Clostermann

Institute for Computer Science and Business Information Systems
University of Duisburg-Essen, 45127 Essen, Schützenbahn 70
{Hesham, bmc}@informatik.uni-essen.de
www.informatik.uni-essen.de

Abstract. A message sequence chart (MSC) is a high-level description of the message interaction between system components and their environment. Communication between distributed instances can be described by MSCs and these descriptions can be extended by notions for time consumption and resources and afterwards included in a system performance model. Such models can be evaluated by discrete event simulation or under reasonable assumptions alternatively with analytical queueing network algorithms. In this way steady state performance measures like resource utilizations and end-to-end delays can be calculated with low effort. The simulation uses the same input like the analytical formulas and allows for the investigation of dynamic performance behaviour or for the study of models including features which can not be handled by analytical formulas.

1 Introduction

The motivation for this work originates from the IPonAir project on architectures of future mobile communication systems [13]. The IPonAIR/MxRAN[1] project aims at a flexible radio access architecture that supports multi-band, multi-standard radio systems integration and the usage of existing and future IP-based protocols. A part of this project is the development of a discrete event simulation system which is to study the performance behaviour of different system designs. In [9 and 29] it is proposed to develop a simulation environment to analyze alternative network architectures and protocol stacks with respect to signalling performance. The authors describe a use case approach to construct a general event driven signalling protocol performance model. To this end Message Sequence Charts (MSCs) are employed as an input of use cases to a performance simulation tool.

Many approaches do exist to enhance formal description techniques by non-functional information on time and resources. In the field of SDL and MSC an overview on the role of performance aspects is given in [21, 22 and 23]. Examples for tools combining the SDL and/or MSC methodology and performance evaluation are QUEST, SPEET, and SPECS [11, 12, 18, 22 and 23]. Much work has also been especially done with respect to Timed MSCs [22 and 23] and Performance MSC [18].

[1] MxRAN stands for Multi-band, Multi-standard Radio Access Network

© IFIP International Federation for Information Processing 2003

In [18], the Performance Message Sequence Chart (PMSC) language extends MSC-96 by annotations to integrate performance aspects. Annotations have semantical meanings for performance evaluation tools as developped e.g. at the university of Erlangen-Nuremberg [10] but they are comments in the original language to allow standard tools to process the specification. PMSC is described in earlier versions in [6] and [7]. PMSC introduces a concept of time for an executed MSC by interpreting MSC events as actions that are executed by tasks which need some time to complete. Every task has a start and completion events that occur at some point in time. In PMSC a system model is used that has two separate sub-models, namely the load model and the machine model. The load model includes the MSC, which describes the functional dependencies between load units, the machine requirements, which are annotated with every load unit (action), and the traffic sources, that specify the intensity of the load. The machine model consists of queueing stations that model processors or channels between processors. To complete the system model a mapping from instances on modeled processors and communication paths on modeled channels must be obtained. To allow flexibility the concepts are separated in different documents.

There are some approaches to integrate time and performance into MSC. [28] extends MSC-92 (MSC-Real Time) by language constructs rather than by annotations. [30] introduces an extension of MSC-96, called Timed MSC, to support performance testing. Performance simulation on the basis of formalised use cases with a language similar to MSC-96 is reported in [4]. A tool that uses MSC-96 for deriving performance models in early phases of the object-oriented software engineering process is described in [31]. In [17] a formal timed semantical model based on term rewriting rules is introduced for MSC-92.

Most approaches to support specification based performance evaluation of systems in the SDL/MSC context extend SDL itself (e.g. the approach described in [5]). Since SDL and MSC are often combined in one project SDL- and MSC-based performance prediction should be integrated and share common documents to support consistency between both specifications.

Here we follow the ideas sketched above; in particular we will use MSCs notions which are extended by annotations to describe required resource consumptions. The instances are assumed to run on resources which have a certain processing speed. In this way a performance model is established which can be quantitatively evaluated, either by discrete event simulation or by queueing network algorithms. Here we mainly follow the latter approach to efficiently calculate mean values for end-to-end-delays and resource utilizations. Moreover a simulation tool has been developed which allows evaluating models which do not satisfy the necessary assumptions to obtain analytical solutions. Additionally simulation can be used to study the dynamic performance behaviour. This paper illustrates the basic ideas by a simplified real world example taken from a project on future wireless communication.

2 Extending MSCs by Time and Resources

2.1 Introduction

A message sequence chart (MSC) is a high-level description of the message interaction between system components and their environment. A major advantage of the MSC language is its clear and unambiguous graphical layout, which immediately gives an intuitive understanding of the described system behavior. The syntax and semantics of MSCs are standardized by ITU-T, as recommendation Z.120. Message Sequence Charts is a language to describe the interaction between a numbers of independent message-passing instances. The basic constituents of the Message Sequence Chart are instance, message, general ordering, condition, timer, action, instance creation and termination. For more details see the ITU standardization documents [25, 26].

2.2 An MSC Example

Here our focus is on MSCs consisting only of instances and messages. The most fundamental language constructs of MSCs, are *instances* (e.g., entities of SDL systems, blocks, processes and services). Instances are reactive entities whose communication behavior is described by the MSCs. The message exchange is the only mean of communication among instances.

Within the instance body the ordering events are specified. A message can be as simple as a signal or as complex as a sophisticated data packet. Each message is associated with a send and a receive event. To illustrate the basic ideas Fig. 1 shows a simple MSC-example.

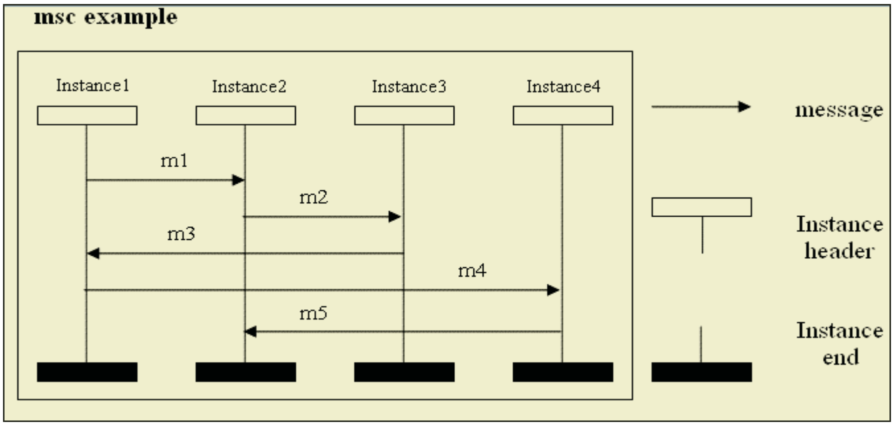


Fig. 1. An example of an MSC (Graphical notation)

2.3 High Level MSC

To define more complex scenarios, the High Level MSC (HMSC) provides a mean to graphically define how a set of MSC can be combined. A HMSC is a directed graph where different types of nodes can be found. An HMSC reference (a component) consists of a frame with rounded corners enclosing the name of the referenced HMSC. Every component has exactly one start node, indicated by an upside-down triangle. Also, it may contain a number of end nodes depicted by a triangle and several HMSC references.

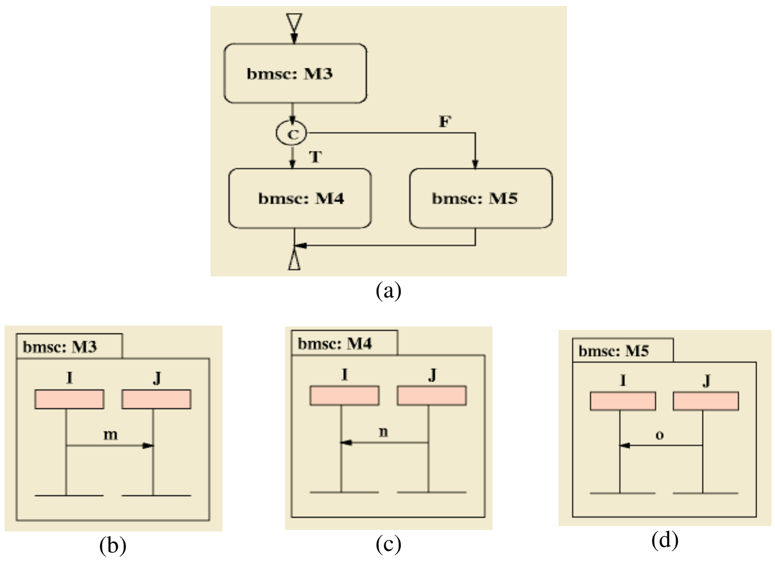


Fig. 2. Basic Graphical Syntax of HMSC (taken from [1])

MSCs can be composed via operators into high-level MSCs (HMSC). MSCs are identified in HMSCs by so called MSC-references. To gain flexibility these high-level MSCs can be MSC-references themself. The various compositional operators of HMSC are sketched below, cf. [16 and 19].

Sequencing: Whenever two MSCs M1 and M2 are sequenced or concatenated, it is interpreted to be *vertically composed*. A *strong sequencing* of Ml and M2 is interpreted to mean that transfer to M2 is possible only after the termination of all events in Ml whereas *weak sequencing* of Ml and M2 denotes the parallel execution of Ml and M2 with the restriction that an action from M2 can be executed only if that is permitted by Ml as defined by consistency requirements for the MSCs.

Selection: The selection operator is shown in Fig. 2(a). In Fig. 2(a), the branch *MSC:M4* will be taken only if the condition denoted by 'C' is true and branch *bMSC:M5* will be taken otherwise. Note that the feature introduces the notion of variables into MSCs and is useful in representing scenarios even though it adds complexity into the model.

Parallel composition: This is also called horizontal composition, and means that multiple MSCs "run" in parallel. There is no restriction among the MCSs.

Loops: A "Loop" is used to represent the possible execution of an MSC an arbitrary number of times with the possibility of termination. The loops are not explicit declared as an operation, but they can be constructed due the fact that the HMSC is a digraph.

Other HMSC operators are repetition, option, and exception. The operators option and exception are only abbreviations that can be encoded using (delayed) choice. Similarly finite repetition can be encoded using (delayed) choice and (weak) sequencing essentially by unfolding of the loop. Through the partial order of MSC events a set of (totally ordered) traces is specified by one plain MSC. An HMSC with only finite loops can be seen as the definition of a set of plain MSCs where the sequential composition glues MSCs together, choice is a set of all possible branches and parallel composition is a set of all possible combinations of free merges where the precedences between MSC events in each MSC is preserved.

2.4 Introducing Time and Resources

In order to construct quantitatively assessable models we extend MSCs by time and resources. This can be done in a rather straightforward way. Each message is associated with a service amount a_i to be executed at the receiving instance i. Each instance has a speed g_i, such that the service time is simply calculated by $s_i=a_i/g_i$. Of course we can group messages into classes and distinguish them, say by index j, j=1,.., m; hence we get the notion $s_{i,j} = a_{ij}/g_i$, describing the service time of a message of class j at station i. Furthermore we consider the instances to behave like queueing stations, i.e. messages arriving at a busy instance are stored in a queue and will have to wait for service.

Fig. 3 displays the execution of a timed MSC; each message has to spend some wait time at arrival at an instance (including the case of zero wait time) followed by a service time which depends on speed of the instance and the required service amount.

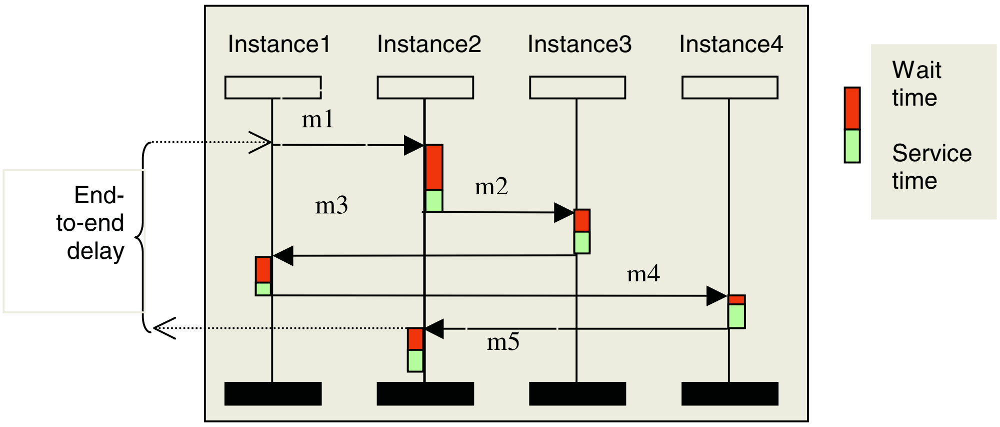


Fig. 3. Wait and service times during execution of a timed MSC

Moreover we consider MSCs to be “open“, i.e. the start of an MSC is triggered from the environment according to some interarrival distribution. Since we will employ analytical mean value formulas based on queueing network theory the interarrival distribution is assumed to be negative exponential. The same assumption is made for the service amounts.

By combining MSCs using the HMSC operators of composition more complex traces can be defined. As a consequence we can define end-to-end delays also for HMSCs; this is done in a later section.

3 Mapping of MSC-Scenarios to Queueing Network Models

Here we describe the derivation of a model which can be quantitatively assessed by means of analytical or simulative techniques. Since the instances are queueing stations and the messages can be considered to be customers or jobs we obtain a queueing network. Each queueing station consists of a wait queue and a server. Messages are generated according to an arrival rate λ, are served at the stations 1 through 4, and finally leave into a sink.

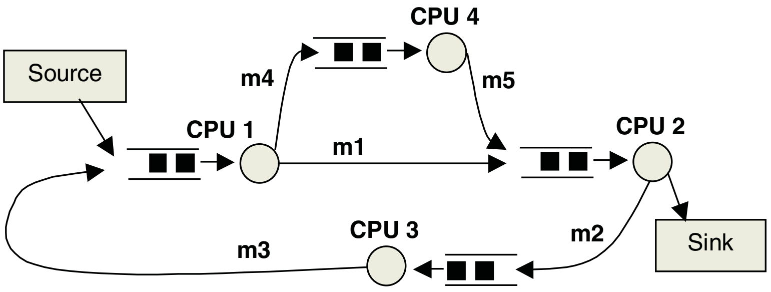


Fig. 4. The example MSC transformed into a queueing network model

Depending on the interarrival distribution of messages, the service time distribution of the messages and the service disciplines of the stations such a network has the so called product form property and can be solved analytically, i.e. performance measures, like utilization of stations or response time can be derived very fast. Theory and algorithms are well established; indeed in this scenario we have a queueing network of Jackson type [14, 20].

Note that the numbering of messages defines their order of execution, here source → m1→ m2 → m3 → m4 → m5 → sink. On the other hand, the queueing network formula[2] to be applied here does neglect the correct order of visits! What really matters when deriving mean performance measures is the number of visits (not their order) and the amount of requested service at the stations.

Here we assume that each station is of type -/M/1-FCFS and MSC arrivals occur according to a Poisson stream and the service times of the messages are also negative exponentially distributed.

In general we assume that a system consists of n stations and m different MSCs classes which arrive with an overall MSC arrival rate λ [MSCs/sec]. Each MSC consists of a certain number of messages which are to be served by the different stations. The arrival rate at station i is $P_i\,\lambda$, [messages/sec] where P_i is the number of messages to be served by station i, i = 1, 2, …, n for all MSCs of the classes j, j=1,.2, …, m.

Let c_{ij} be the number of messages of an MSC of class j which are served at station i, then we can define P_i, the total number of messages received and served by station i, as follows:

[2] In case any of these assumptions is not valid, we have to use approximation algorithms or discrete event simulation.

$$P_i = \sum_{j=1}^{m} c_{ij}, \quad i = 1, 2, \ldots, n, \quad \mathit{for\ all\ stations\ i} \tag{1}$$

The same way, we can define the message arrival rates $\lambda_{i,j}$ [messages/sec] at station i for all messages of MSC class *j* as:

$$\lambda_{ij} = c_{ij}\lambda, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m \tag{2}$$

Hence, the overall message arrival rate λ_i at station i for all messages of all MSC classes is:

$$\lambda_i = P_i\ \lambda, \quad i = 1, 2, \ldots, n \tag{3}$$

Let μ_{ij} be the service rate (messages of MSC class *j* / sec) at station i, i = 1, 2, …, n. According to our definition of service amount a_{ij} and speed of stations g_i we have the service time $s_{ij} = a_{ij}/g_i$ and hence these service rates are defined by $\mu_{ij}=1/s_{ij}$ [messages/sec].

4 Computing Performance Measures by Queueing Network Analysis

Under the definitions given by equations 1, 2 and 3 we can easily compute mean values for stationary performance measures like station utilizations and response times. Of course the response times for each MSC class and the overall end-to-end delay (E2E) for the execution of a all MSCs can also be computed.

4.1 MSC Response Times

Note that we calculate performance measures for the execution of MSCs, i.e. response time refers to the mean execution time of one MSC including wait times as well as service times. Link delays are only included if links are explicitly modeled as instances. Performance measures for single messages are not considered; moreover all messages belonging to the same MSC are not distinguished. An extension to distinguish between messages would lead to a three-indexed service amount, say a_{ijk}. This can be done, but is not in the scope of this paper.

The utilization ρ_{ij} of station i with respect to MSC class *j* is defined as

$$\rho_{ij} = \frac{\lambda_{ij}}{\mu_{ij}}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m \tag{4}$$

Of course the overall utilization ρ_i for station i with respect to all MSC classes is

$$\rho_i = \sum_{j=1}^{m} \rho_{ij} = \sum_{j=1}^{m} \frac{\lambda_{ij}}{\mu_{ij}} = \lambda \sum_{j=1}^{m} \frac{c_{ij}}{\mu_{ij}}, \quad i = 1, 2, \ldots, n \tag{5}$$

Let R_{ij} denote the response time for an MSC of class *j* that is served by station i, then we can apply the M/M/1 formula to get the response times R_{ij} as follows:

$$R_{ij} = \frac{1}{\lambda} \cdot \frac{\rho_{ij}}{1-\rho_i} \tag{6}$$

Let R_j denotes the overall response time for MSC of class *j* that is served by all stations then we can apply the M/M/1 formula to get the R_j as follows:

$$R_j = \sum_{i=1}^{n} R_{ij} = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{\rho_{ij}}{1-\rho_i} \tag{7}$$

4.2 End-to-End Delay for Strong Sequencing Composition in HMSC

In the previous section, we assumed that R_j denotes the overall response time for MSC of class *j* that is served by all stations and the corresponding formula is given by equation 7. In this section we will derive a formula for calculating the end-to-end delay E2E. We can define the *end-to-end delay* to be the time between the arrival of the first message in the first MSC of the HMSC and the departure of the last message in the last MSC of that HMSC, i.e. the time needed to run a complete HMSC. We assume that we have n MSCs that are strongly sequenced and that the calculated response times are R_j, $j = 1, 2, \ldots, n$. In this case the average end-to-end delay E2E gives the average time duration of one single execution of all MSCs by:

$$E2E = \sum_{j=1}^{m} R_j = \sum_{j=1}^{m} \frac{1}{\lambda} \sum_{i=1}^{n} \frac{\rho_{ij}}{1-\rho_i} = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{\sum_{j=1}^{m} \rho_{ij}}{1-\rho_i} = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{\rho_i}{1-\rho_i} \tag{8}$$

Equation 8 gives a formula to calculate the end-to-end delay which can be applied in cases similar to the example, in which we have two MSCs that are vertically composed (see Fig. 7). According to the definition of the sequencing, the HMSC in the example represents a strong sequencing case. Note that we assumed that there is no time gap between the MSCs.

The analytical computation of average end-to-end delay for other cases of HMSC compositions can be done under assumptions which must be defined more precisely. For the simulation we just determine the time between the start of the first and the end of the last message of all simulated HMSCs and calculate some statistic measures, like mean values, confidence intervals, minimum and maximum.

5 An Example from Mobile Communications

We study an example which is closely related to current projects on third and fourth wireless initiatives, like the one pursued in the IPonAir-project. We consider the communication interaction between User Equipment (UE, these are users with mobile terminals), one or several NodeBs (which are the UMTS equivalent to the Base Transceiver Stations, in the context of GSM known as BTS), a Radio Network Controller (RNC) and the Core Network (CN). This communication needs dozens of system functions which are structured into many MSCs with many hundreds of messages.

Here we consider a scenario following system functions named "RRC connection setup" and "RRC connection release" as shown in Fig. 5 and Fig. 6.

These two MSCs may grouped together to build a High Level MSC as displayed in Fig. 7. We consider a strongly sequenced pair of setup and release functions as a workload model. To build a scenario for performance evaluation we have to consider the execution of this HMSC assuming a certain traffic model, e.g. assuming a Poisson stream of HMSCs according a certain arrival rate; also a certain system configuration must be described.

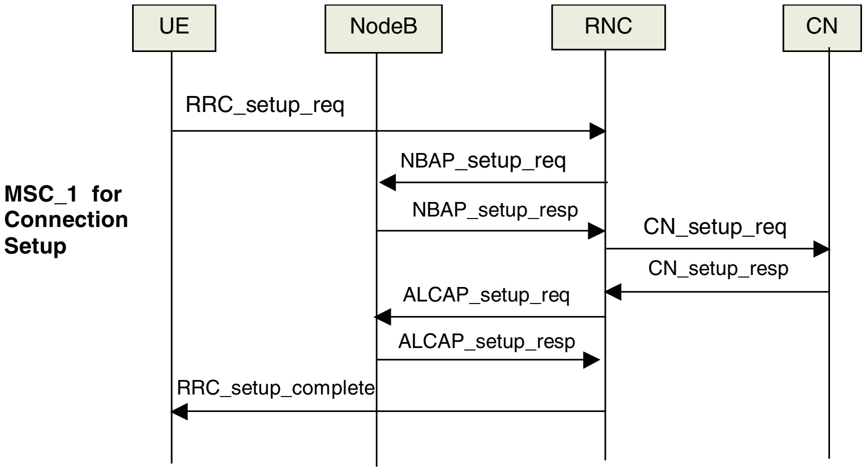


Fig. 5. MSC_1 for connection setup (partially fictitious)

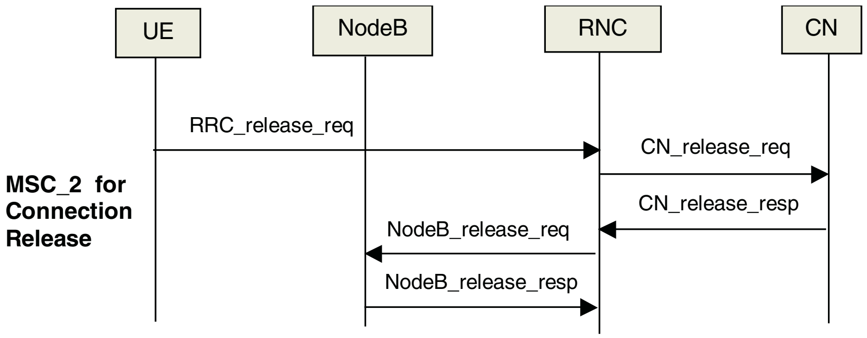


Fig. 6. MSC_2 for connection release (partially fictitious)

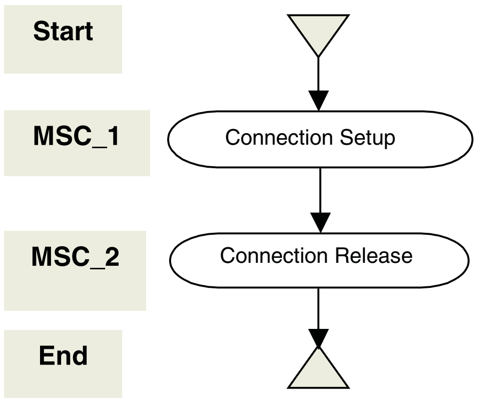


Fig. 7. High Level MSC consisting of MSC1 and MSC2

We consider four queueing stations which are to execute (or do represent) the four instances and the HMSC combining the connection setup (MSC_1) and the connection release (MSC_2) respectively as displayed in Fig. 7.

We assume speeds of 1000, 2000, and 5000 messages/sec for the stations, named UE, NodeB and CN. We run a series of model evaluations for three different RNC-configurations by setting the speed of the station RNC to 3000, 4000 and 5000 messages/sec. The service amounts of the messages are set to 2.0 for service at NodeB, to 5.0 at the RNC, and to 10.0 at the CN-station; in this way we account roughly for the fact that a single message may represent multiple messages.

We considered for each configuration the HMSC arrival rates λ =10, 20, …, 130 HMSC/sec, i.e. the number of Setup/ Release-Pairs described by HMSC are varied in the range from 10 to 130 per second. Please note, that we do just consider a part of the signalling traffic; neither other signalling traffic nor user traffic is included in this example.

Note that the kind of queueing stations used in these calculations can be differently chosen to better describe real system properties, e.g. the UE-cluster should be modelled as an infinite-server station and the Radio Network Controler (RNC) and Core Network (CN) as multi-server stations. This would still result in analytically solvable models implying minor modifications of the analytical formulas.

The evaluation of the formulas yields the end-to-end delays for the connection setup. As expected the evaluations provide exponentially increasing curves as displayed in Fig. 8.

Such curves are useful to investigate in early design stages which amount of traffic can be carried by the planned configuration, or the other way round, what kind of resources are needed to carry the traffic under specified service levels.

Although the necessary assumptions with respect to the independence of messages and negative exponential service normally do not hold exactly, these evaluations may be extremely helpful for a system developer. Such analytical results show the scope of possible parameter settings and allow a better planning of simulation scenarios which include more details and are closer to reality.

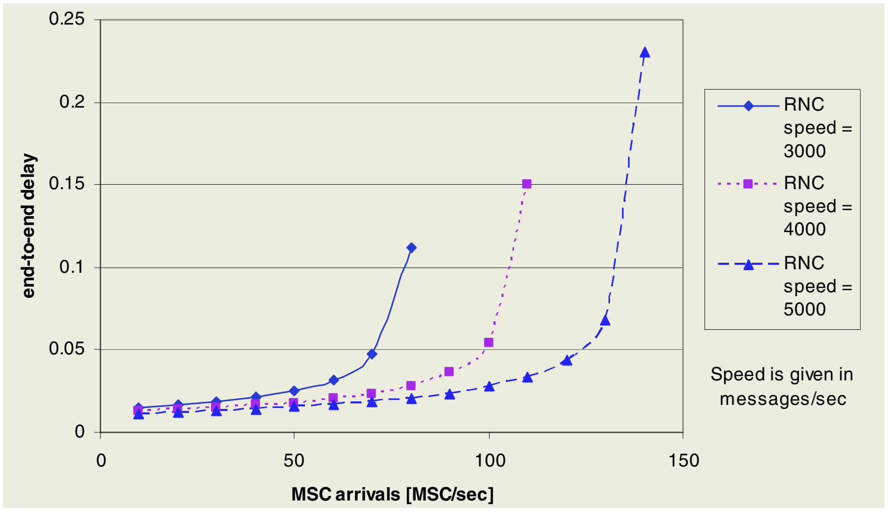


Fig. 8. End-to-end delay of MSC connection setup

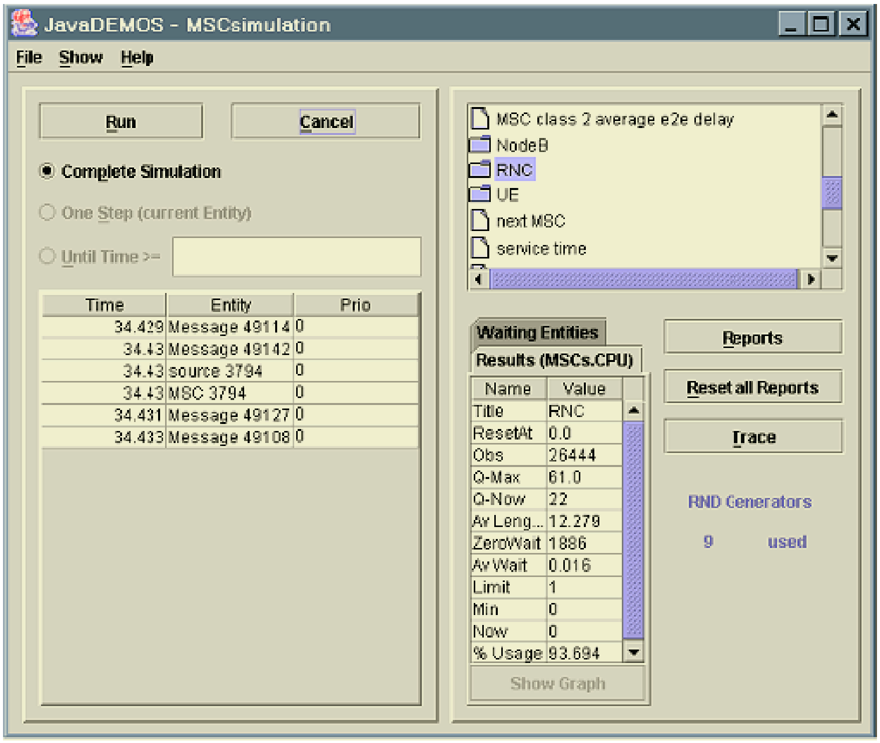
6 Simulation of MSC Based Models

Additional to the analytical formulas a simulation system has been developed, which uses the same input as the analytical model and delivers the same results, i. e. mean values and confidence intervals. Of course the simulator can be used to evaluate models which do not satisfy the conditions necessary for analytical evaluations; important examples for model features which violate these conditions are non-exponential distributed service times (e.g. low service time variations or even deterministic service), non-Poisson arrivals (e.g. bursty sources) and priority scheduling.

This simulator has been written in Java using the JavaDEMOS package [8, 15 and 24]. JavaDemos is a Java library for discrete event simulation, which was inspired by the DEMOS system written by G. M. Birtwistle [2 and 3]. JavaDemos is based on an implementation of the DEMOS features in Java. The syntax of the procedures is as close as possible to DEMOS, in order to simplify the translation of DEMOS programs to JavaDemos. In addition, JavaDemos consists of a graphical front-end which permits the visualization of a simulation run and which allows basic interactions with the simulation system.

The basic concept of JavaDEMOS include process-like objects which do implement behaviour patterns, which may acquire and release resources, may wait until certain conditions are fulfilled, are able to interact with each other in a master/slave mode and can of course be scheduled in the event list.

Fig. 9 has taken from the simulation of our example described in a previous section. In the simulation the messages are entities and the visited stations (UE, NodeB, …) are modeled as resources. Fig. 9 (on the left) shows the entities which are scheduled in the event list according to their event times. Moreover, the state of all objects can be inspected at any time; Fig. 9 (on the right) shows the current values of the state variables of the station RNC, in particular the maximum queue length Q-



[✓ Complete Simulation] One Step (current Entity) Until Time >=

- MSC class 2 average e2e delay
- NodeB
- RNC
- UE
- next MSC
- service time

Time	Entity	Prio
34.429	Message 49114	0
34.43	Message 49142	0
34.43	source 3794	0
34.43	MSC 3794	0
34.431	Message 49127	0
34.433	Message 49108	0

Name	Value
Title	RNC
ResetAt	0.0
Obs	26444
Q-Max	61.0
Q-Now	22
Av Leng...	12.279
ZeroWait	1886
Av Wait	0.016
Limit	1
Min	0
Now	0
% Usage	93.694

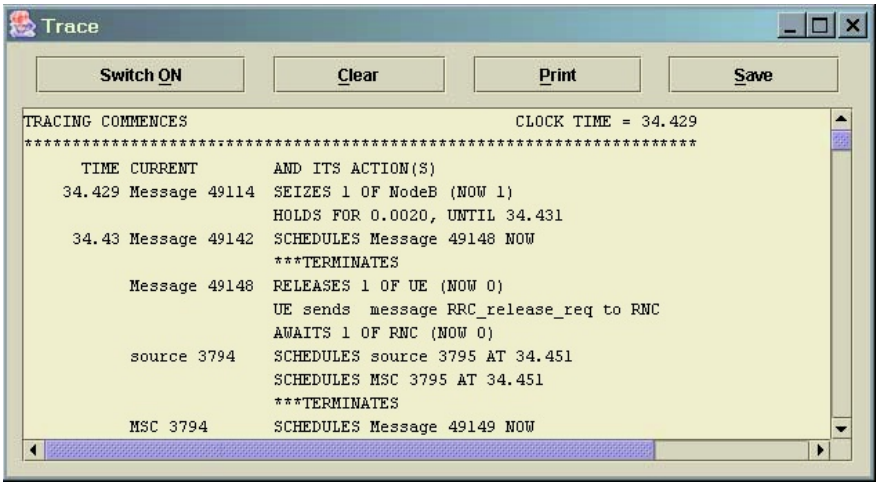
Fig. 9. The graphical user interface of JavaDEMOS

MAX, the current queue length Q-NOW, the average queue length and the average wait time.

Observing these parameters dynamic performance behaviour can be inspected. As an example consider the values displayed in Fig. 9 showing a snapshot of the simulation at a certain point of time (here after approximately 34.43 seconds); since the utilization of the system is rather large the simulation is still in its transient phase and the average queue length is 12.27 (26444 observed messages, RNC utilization 93.6%,) in contrast to an average queue length of 15.4 after more than 773000 observed messages and 1000 sec model time (and we have still slightly growing utilization of 93.95 %), cf. Fig. 11.

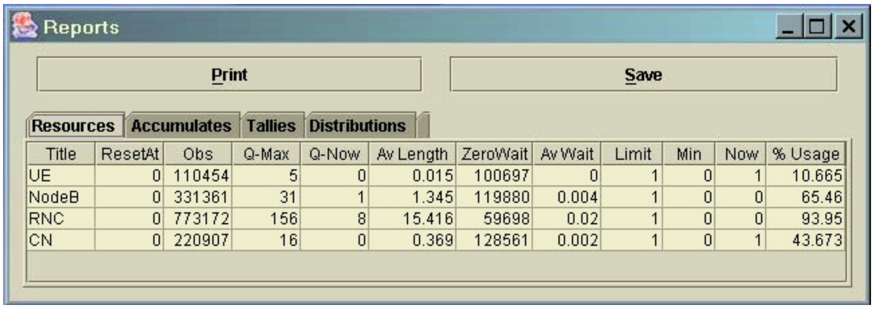
Also the maximum observed queue lengths of 61 (at time point 34.43) and 156 (at time point 1000, at the end of the simulation) respectively, illustrate some aspects of the dynamic system behaviour. The true stationary values (obtained by the analytical formulas) are still significantly higher than those obtained by simulation, namely 96.25 % for utilization and 26.66 for the average queue length.

In Fig. 10 we can see the trace-window in which we can see step by step how the simulation proceeds. It shows the model times, the messages and their actions with respect to resource acquirements and resource releases.



```
TRACING COMMENCES                                          CLOCK TIME = 34.429
**********************************************************************
      TIME CURRENT        AND ITS ACTION(S)
    34.429 Message 49114  SEIZES 1 OF NodeB (NOW 1)
                          HOLDS FOR 0.0020, UNTIL 34.431
     34.43 Message 49142  SCHEDULES Message 49148 NOW
                          ***TERMINATES
           Message 49148  RELEASES 1 OF UE (NOW 0)
                          UE sends  message RRC_release_req to RNC
                          AWAITS 1 OF RNC (NOW 0)
           source 3794    SCHEDULES source 3795 AT 34.451
                          SCHEDULES MSC 3795 AT 34.451
                          ***TERMINATES
           MSC 3794       SCHEDULES Message 49149 NOW
```

Fig. 10. Trace window showing a running MSC simulation



Title	ResetAt	Obs	Q-Max	Q-Now	Av Length	ZeroWait	Av Wait	Limit	Min	Now	% Usage
UE	0	110454	5	0	0.015	100697	0	1	0	1	10.665
NodeB	0	331361	31	1	1.345	119880	0.004	1	0	0	65.46
RNC	0	773172	156	8	15.416	59698	0.02	1	0	0	93.95
CN	0	220907	16	0	0.369	128561	0.002	1	0	1	43.673

Fig. 11. Report-window for the MSC simulation

A complete report can be obtained as shown in Fig. 11. We see in Fig. 11 the resources report which shows the observation number (the number of messages that visited each station), the maximum queue length for each station, the average queue length and the average wait time for entities in the queue for each station and finally the % usage (utilization) for each station.

Other performance measures like the end-to-end-delay for a system function defined by one or several MSCs is not part of JavaDEMOS and has been programmed additionally including a batch means procedure for the estimation of confidence intervals.

Conclusion and Outlook

Currently the tool is in a state that demonstrates the feasibility of the approach. In particular the automatic inclusion of complex system functions consisting of many procedures described as MSCs and HMSCs will allow to build up and evaluate performance models with rather low effort. The analytical results can be used to get a fast overview on the overall stationary performance or potential bottlenecks. Simulation can alternatively used to study the dynamic behaviour or to show the impact of features which can not be represented sufficiently in analytical models. Future work will be directed towards the application of these methods and the associated tool to real world projects. Hence, the mapping of systems specified by MSC and HMSC to queueing networks must be conceptually enhanced and implemented. This includes the extension of the analytical formulas to a richer model world; for this purpose the field of queueing network theory provides vast resources.

Acknowledgements

This work has been inspired by related work done in the IPonAir/MxRAN project. The authors acknowledge the contributions of Peter Schefczik, Michael Söllner, Wilfried Speltacker (all with Lucent Technologies, Nuremberg), Andreas Mitschele-Thiel (Technical University of Ilmenau), Georgios Nikolaidis (University of Athens) and Anja Wiedemann (University of Duisburg-Essen).

References

1. Belachev, M., Shyamasundar, R. K.: MSC+: From Requirement to Prototyped Systems. 13th Euromicro Conference on Real-Time Systems, Technical University of Delft, Delft, The Netherlands, June 13th - 15th, (2001), 117-124.
2. Birtwistle G. M.: DEMOS a System for Discrete Event Modelling on Simula. Macmillan, http://www.cosc.canterbury.ac.nz/teaching/classes/cosc327/
3. Birtwistle G. M.: DEMOS Reference Manual, (1985) http://www.informatik.uni-essen.de/Lehre/Material/DiskreteSim/DemosRefMan.txt
4. Braga L., Manione, R., and Renditore, P.: A Formal Description Language for the Modelling and Simulation of Timed Interaction Diagrams. In: Gotzhein and Bredereke (eds.) [11], (1996) 245-260.
5. Diefenbruch, M., Hintelmann, J., and Müller-Clostermann, B.: The QUEST-Approach for the Performance Evaluation of SDL-Systems. In: Gotzhein and Bredereke (eds.) [11], (1996) 229-244.
6. Faltin, N., Lambert, L., Mitschele-Thiel, A., and Slomka, F.: An Annotational Extension of Message Sequence Charts to Support Performance Engineering. In: SDL'97: Time for Testing - SDL, MSC and Trends, Evry, France, Eighth SDL Forum, North-Holland, (1997) 307-322
7. Faltin, N., Lambert, L., Mitschele-Thiel, A., and Slomka F.: PMSC - Performance Message Sequence Chart. Technical Report 10/97, Universität Erlangen-Nürnberg, IMMD VII, Erlangen (1997)
8. Flüs, C., Mohamed, H., Müller-Clostermann, B.: JavaDEMOS: Java-based Discrete Event Simulation. In: MMB-Mitteilungen Nr. 41, (2002), cf. http://www.informatik.unibw-muenchen.de/mmb/mmb41/inhalt.htm
9. Frangiadakis, N., Nikolaidis, G., Schefczik, P., Wiedemann, A.: MxRAN Functional Architecture Performance Modeling. OPNET Conference, (2002) cf. http://www.iponair.de/publications.shtml
10. Friedrich-Alexander-Universität Erlangen-Nürnberg, Department of Computer Science 7 Homepage: http://www7.informatik.uni-erlangen.de

11. Gotzhein, R. and Bredereke, J. (eds.): Formal Description Techniques IX: Theory, application and tools, IFIP TC6 WG6.1 International Conference on Formal Description Techniques IX / Protocol Specification, Testing and Verification XVI, Kaiserslautern, Germany (1996)
12. Herzog, U.: Formal Methods in Performance Evaluation. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.): Lectures on Formal Methods and Performance Analysis; LNCS 2090, Springer (2001)
13. IPonAir Homepage : http://www.iponair.de
14. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, Inc., New York (1991)
15. JavaDEMOS software on the JavaDEMOS home page: http://www.informatik.uni-essen.de/SysMod/JavaDEMOS/
16. Jonsson, B., Padilla, G.: An Execution Semantics for MSC2000. 10th International SDL Forum Copenhagen, Denmark, Springer Verlag LNCS 2078 (2001)
17. Kosiuczenko, P.: Time in Message Sequence Chart. In Lengauer C., Griebl M., and Gorlatch, S. (eds.), Euro-Par'97 Parallel Processing, Springer LNCS 1300, 562-566, (1997)
18. Lambert, L.: PMSC for Performance Evaluation. in: Mitschele-Thiel, A.; Müller-Clostermann, B.; Reed, R. (eds.); Workshop on Performance and Time in SDL and MSC; February 17-19, 1998, University of Erlangen-Nürnberg
19. Mauw, S. and Reniers, M.A.: High-Level Message Sequence Charts. In Cavalli and Sarma [6], (1997) 291–306.
20. Menascé, D., Almeida, V. and Dowdy, L.: Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems. Prentice-Hall (1994)
21. Mitschele-Thiel, A.: Integrating Performance and Time in SDL and MSC – Current State and Vision. In [22]
22. Mitschele-Thiel, A., Müller-Clostermann, B. and Reed, R.(eds.): Proc. of Workshop on Performance and Time in SDL and MSC. February 17-19, 1998, University of Erlangen-Nürnberg
23. Mitschele-Thiel, A.: Systems Engineering with SDL – Developing Performance Critical Applications. Wiley (2000)
24. Mohamed, H.: Discrete Event Simulation Using JavaDEMOS. Technical Report, University of Essen, (2001) http://www.informatik.uni-essen.de/SysMod/publikationen/index.html,
25. MSC Standard, ITU-T. Recommendation Z.120, Message Sequence Charts. Geneva, (1999)
26. MSC-2000 MESSAGE SEQUENCE CHART (MSC), (revised in 2001), SDL Forum Version of Z.120 (11/99) rev. 1(14/11/01)
27. Padilla, G.: An Execution Semantics for MSC2000. August 2000, http://citeseer.nj.nec.com/padilla00execution.html
28. Schaffer. C.: MSC/RT: A Real-Time Extension to Message Sequence Charts (MSCs). Technical Report TR140-96, Johannes Kepler Universitat Linz, Institut für Systemwissenschaften (1996)
29. Schefczik, P., Mitschele-Thiel, A., Soellner, M. and Speltacker, W.: On MSC-Based Performance Simulation. the Third International Workshop on Software and Performance (WOSP 2002), Rome (2002)
30. Schieferdecker, I., Rennoch A., and Mertens, O.: Timed MSCs - an Extension to MSC96. In Wolisz et al. [32], (1997) 165-174.
31. Smith, C.U. and Williams, L.G.: Performance Engineering Evaluation of Object-Oriented Systems with SPEED. In R. Marie, B. Plateau, M. Calzarossa, and G. Rubino, (eds.), Computer performance evaluation, Springer LNCS 1245 (1997)
32. Wolisz, A., Schieferdecker, I. and Rennoch, A. (eds.): Formale Beschreibungstechniken für verteilte Systeme. GMD-Studie Nr. 315, Berlin, Germany, GI/ITG, GMD-Forschungszentrum (1997)