Performance Analysis of Concurrent PCOs in TTCN-3

Máté J. Csorba[1], Sándor Palugyai[1], Sarolta Dibuz[1], and Gyula Csopaki[2]

[1] Ericsson Hungary Ltd., Test Competence Center
H-1117 Budapest, Irinyi J. u. 4-20, Hungary
Phone: (36) 1-437 7489, Fax: (36) 1-437 7576
[2] Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Magyar tudósok körútja 2, Hungary
{Mate.Csorba, Sandor.Palugyai, Sarolta.Dibuz}@ericsson.com
Csopaki@tmit.bme.hu

Abstract. This paper deals with a study and a mathematical model of concurrent Points of Control and Observation (PCOs) realized in Testing and Test Control Notation version 3 (TTCN-3). We study test scenarios that are gaining importance as TTCN-3 is emerging as a notation suitable for conducting load tests too. We investigate communication between parallel test components (PTCs) and analyze race conditions between the queues underlying the implemented PCOs. This way, we build an analytic model to investigate behavior of PCOs under stress conditions and to assess possible latencies messages in a TTCN-3 based load test system might suffer. We present a discrete-time Quasi Birth-Death process to predict performance indices of test components and we propose to use the results to avoid indefinite postponement in the communication of PTCs. Also, we aim to use the model for calculating traffic intensity limits under which it is feasible to use TTCN-3 for load testing. Furthermore, we present the output of the model together with an example load test scenario that is vulnerable to that types of latencies.

1 Introduction

The subject of our investigation is the standardized test specifi cation language the Testing and Test control Notation version 3 (TTCN-3) [1]. TTCN-3 is widely used in different areas of testing including different fi elds of telecom and datacom and is gaining acceptance even in automotive systems testing. The language itself has a variety of applications including testing various systems for interoperability, robustness and conformance.

Recently, there has been a sore need to assess the capabilities of TTCN-3 as a testing solution not only for conformance and interoperability testing but for performance evaluation of telecommunication systems as well. Since TTCN-3 is a rather high level specifi cation language, concerns have arisen regarding its applicability in load tests that require a signifi cant amount of processing power, e.g. a high number of packets per second generated. On the other hand, tests and numerous pioneer projects show us that the language is capable of working at the edges of what the underlying hardware is capable of. Besides, emerging real-time extensions to the original notation also exist [2], [3], [4]. So, usage of TTCN-3 in load tests should not imply a bottleneck if tests are designed carefully.

Our work examines the event processing capabilities of test components that exchange messages via communication ports competing with each other for system resources. By analyzing race conditions between the queues underlying the implemented PCOs, we build an analytic model to investigate behavior of PCOs under stress conditions and to assess possible latencies messages in a TTCN-3 based load test system might suffer. A discrete-time Quasi Birth-Death process is presented to predict performance indices of test components. We aim to use the model for predicting traffic intensity limits under which it is feasible to use TTCN-3 for load testing purposes. The remainder of this paper is organized as follows.

In Section 2, we introduce briefly a few basic issues in TTCN-3, the operation of Points of Control and Observation (PCOs), and in Section 3, we present the mathematical formalism used in this paper. Section 4 presents a parametrical model of test components using multiple PCOs. In Section 5, analytic results of the model for an example test component are detailed, together with an *ns-2* [5] simulation that was used mainly for validating and fine tuning of the model. Finally, in Section 6, concluding remarks are given and future work is detailed.

2 Concurrent PCOs and Alternative Behavior in TTCN-3

This section outlines the basic structures in TTCN-3 we aim to model, together with examples of load test scenarios that might be vulnerable to certain types of latencies during execution. Moreover, we point out the performance indices we evaluate to predict the behavior of TTCN-3 test components.

In TTCN-3, configuration of the test system can be set dynamically. This means that a configuration may consist of several components participating in the test [6], either as a component that communicates directly with the System Under Test (SUT), or as a component having only registration or internal purposes, meaning that it communicates only with parts of the test system itself. Within every configuration there is only one designated component, the Main Test Component (MTC) that is created automatically. Other components can be created dynamically and are called Parallel Test Components (PTCs). PTCs do not have any hierarchical order among them. PTCs communicate with each other, and with the MTC also, via test ports. Similarly, communication towards the SUT is established via test ports too, as in Figure 1.

Each test port that connects either two test components (internal port) or a test component and the interface towards the SUT is modeled as a FIFO queue for the incoming/outgoing messages. Each component can access messages in its correspondent queue one by one. Properties of the FIFO queues assigned to a test port are dependent on the actual implementation of the TTCN-3 compiler. The queues can be infinite in principle, as long as the system memory lasts, but might overflow indeed. More importantly, in a load test system response time must be considerably short. This means that it is inexpedient to implement a virtually infinite buffer for a PCO and forget about

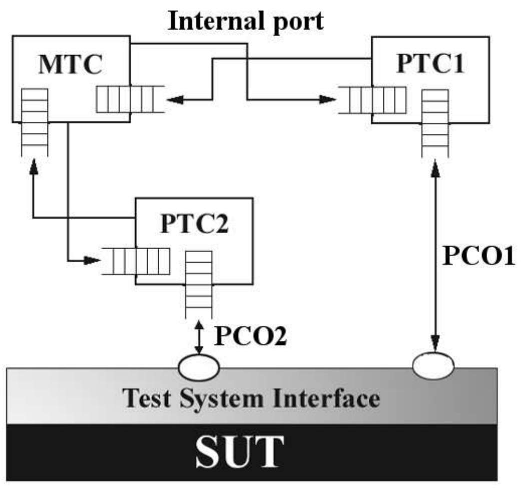

Fig. 1. Test Components and Test Ports

message loss at all. Although a sufficiently long buffer might eliminate message loss, response time increases significantly at the same time. Accordingly, we investigate message loss in relatively short queues.

The actual behavior of a test case is defined by dynamic behavioral statements in a test component that communicates over certain test ports. Usually, sequences of statements can be expressed as a tree of execution paths, that is alternatives. In TTCN-3, the *alt* statement is used to handle events possible in a particular state of the test component. These events include reception of messages, timer events and termination of other PTCs.

The *alt* statement uses a so-called snapshot logic [7]. This means, that before evaluating the actual alternatives in the *alt* a snapshot of the test component containing any information that is relevant (e.g. status of the test ports involved in the *alt*, running timers) is taken. Branches of the *alt* might have a Boolean guard expression assigned to them that is evaluated before the branch is examined. The guard expression might be based on the snapshot as well.

Different types of branches exist in an *alt* statement (e.g. timeout, receive). Each time a receiving branch is found during execution a matching operation is done first. In case the incoming message, that is first in the corresponding PCO's FIFO, matches the criteria the *alt* branch will be executed and the message will be removed from the top of the queue. Otherwise, the execution continues and the next branch will be examined. However, execution does not stop after a snapshot was taken, so the state of the test component and the queues assigned to it might change in between. However, these events do not change the actual snapshot, until the *alt* statement is not executed again.

So, generally the two most significant factors we consider, while evaluating the performance of a test component are the matching mechanism and the queuing at the test ports. See for example the following scenario (Figure 2).

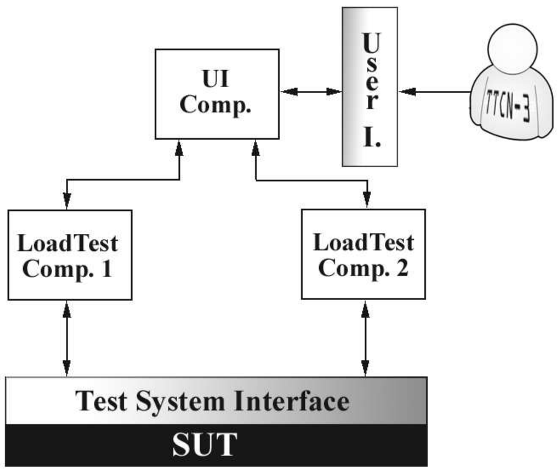

Fig. 2. Load Testing Example Scenario

In this simple example we have two load test components that connect to the SUT via two different PCOs and handle the actual protocol behavior. One test component is receiving commands from the user via a user interface (UI). Besides, this component can supply the user with additional data regarding the test case execution by querying the load test components periodically. Let us say that in this setup LoadTest Comp.1 stimulates the SUT, which is for example an ATM switching center, and due to this stimulation the SUT forwards a high number of calls towards LoadTest Comp.2. In this case, LoadTest Comp.2 should be able to handle and examine a very high amount of calls per second coming from the switch. The actual race condition results from the different functionalities of LoadTest Comp.2. Firstly, it receives a high amount of incoming calls from the SUT and secondly it must answer status request messages, coming from UI. Comp. on an internal port periodically. Although, status messages might be relatively infrequent compared to the messages participating in the load test, they also need to be handled by LoadTest Comp.2. most probably in the same *alt* structure. This setup leads to a race condition between the two separate FIFO queues assigned to the two test ports of the component. Namely, in the test component branches of the *alt* referring to the PCO towards the SUT receive signifi cantly more hits in a unit of time than branches referring to the internal port do, this way increasing the risk of an indefi nite postponement of the status messages.

In our modeling approach, we build an analytic model of test components that use alternatives to handle internal messages and messages coming from the SUT. We describe concurrent queues underlying the test ports of a component with a stochastic process and calculate the steady-state solution. After solving the analytical model, we predict the probability that one of the queues contain a message that is postponed indefi - nitely, because of the fact that race conditions arise between the queues. The probability of an indefi nite postponement is calculated as a function of arrival intensities at the corresponding queues and of other parameters relevant to the implementation of the actual test component.

3 Discrete-time Quasi Birth-Death Processes

Our method uses a mathematical model that can be evaluated and performance indices of the described components can be derived by existing solver techniques. The mathematical formalism behind our evaluation method can be identifi ed as discrete-time Quasi Birth-Death processes (QBDs) [8]. A simpler M/G/n type solution would not allow us to use fi nite queues and to calculate state distributions [9].

The mathematical analysis in turn, is based on matrix-geometric solution techniques and matrix analytic methods [10]. In order to become acquainted with QBD processes let us consider processes $N(t)$ and $J(t)$, where { $N(t), J(t)$ } is a DTMC (Discrete-Time Markov Chain). The two processes have the following meaning: $N(t)$ is the level process and $J(t)$ is the phase process. { $N(t)$, $J(t)$ } is a QBD if the transitions between levels are restricted to one level up or down, or inside the actual level. The structure of the transition probability matrix P of a simple QBD is the following:

$$\mathbf{P} = \begin{bmatrix} \underline{\underline{A_1}}^* & \underline{\underline{A_0}} & \underline{\underline{0}} & \underline{\underline{0}} & \cdots & & \\ \underline{\underline{A_2}} & \underline{\underline{A_1}} & \underline{\underline{A_0}} & \underline{\underline{0}} & \cdots & & \\ \underline{\underline{0}} & \underline{\underline{A_2}} & \underline{\underline{A_1}} & \underline{\underline{A_0}} & \underline{\underline{0}} & & \\ & \underline{\underline{0}} & \underline{\underline{A_2}} & \underline{\underline{A_1}} & \underline{\underline{A_0}} & & \\ & & & \ddots & \ddots & \ddots & \end{bmatrix} \qquad (1)$$

Matrix A_0 describes the arrivals (transitions one level up), matrix A_1 describes transitions inside each level and matrix A_2 describes departures (transitions one level down). Matrix A_1^* is an irregular transition matrix at level 0. The row sum of P is equal to 1 (discrete-time model). The tangible meaning of levels and phases can be seen in Figure 3.

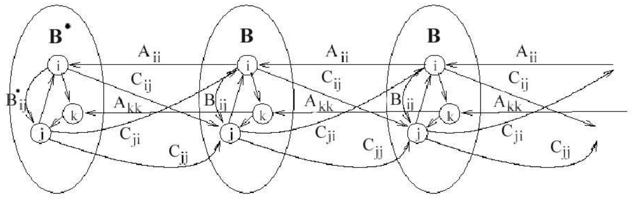

Fig. 3. Logical representation of a QBD process

The two-dimensional property of a QBD is utilized by our method signifi cantly. On one hand, we map the sequential branch examination and matching mechanisms of the *alt* structures in a given test case we model, into the internal phases inside each level of the QBD. On the other hand, we use the levels to describe queuing at the test port.

After the system's behavior is described by the QBD model, with the aid of matrix analytic methods the following properties of the model can be calculated: steady state solution, queue length in steady state, phase distribution, mean value of time to transition back to level zero (meaning that the buffer is empty). As a result, a variety of performance indices can be derived from the QBD model, such as packet loss ratios, delays, and available throughput under stress conditions.

4 A QBD Model for PCOs

For the performance evaluation of PTCs that use alternatives to handle messages we build the following novel QBD model. For the sake of simplicity consider a test component using two test ports, each of them with a separate FIFO queue (e.g. in Section 2). The model will macroscopically look like a simple QBD that is infinite in one dimension similar to the example in Figure 3. So, in the first dimension the model is an infinite QBD. This dimension represents the PCO that drives the system into a race condition, call it PCO1. This might be the PCO serving a significant load coming inwards from the SUT. But at a lower level another embedded QBD is to be found, representing the PCO (call it PCO2) that is suppressed by the heavily loaded PCO1. Levels of the QBD describe queue sizes in this case, accordingly the second dimension of the model is in direct connection with the size of the queue assigned to PCO2. This dimension will be finite with a parametrical size, in order to assure matrix-analytic solutions to work [11], [12] and to allow the model to predict infinite postponement in the PCO, which is underprivileged because of the race conditions between the concurrent PCOs (Figure 4).

Example 1 (Example alt structure).

```
alt {
/*Group 1*/
[] PCO1.receive(template1) { Statements... }
[] PCO1.receive(template2) { Statements... }
[] PCO1.receive     { // Trash unmatched messages
                     repeat }
/*End of Group 1*/
/*Group 2*/
[] PCO2.receive(template3) { Statements... }
[] PCO2.receive(template4) { Statements... }
[] PCO2.receive     { // Trash unmatched messages
                      repeat }
/*End of Group 2*/
}
```

The vertical dimension, that is the number of distinct groups of states in Figure 4 is restricted by the size of the FIFO queue of PCO2, which can be set as a model parameter. Also, relatively short buffers are considered for the investigated PCO in order to keep response time, that is crucial in a load test system, under a considerably low level. If we look at this model as a simple QBD, transitions upwards and downwards are described by matrices *C* and *A* respectively. Whereas, matrix *B* describes internal phase transitions. Irregularities, denoted by *, occur at the first level, where there is no possibility of a transition downwards. Each group of phases (within ellipses) corresponds

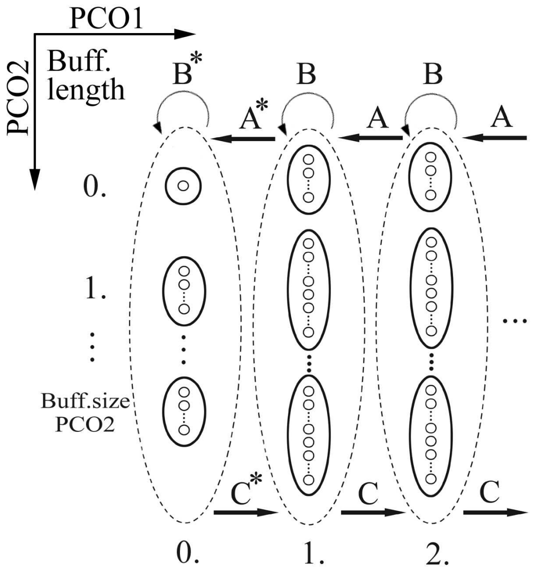

Fig. 4. Two dimensional model for two concurrent PCOs

to a group of *alt* branches that use the same PCO. Group to group (vertical) transitions describe queuing and service of messages at PCO2. For example, see the excerpt in *Example 1*.

In this example branches that refer to the same PCO are grouped together. Two groups are formed that correspond to the separate groups of phases in Figure 4. In effect, only branches of PCO2 are active in the first horizontal level of the model as horizontal levels represent the queue of PCO1. Being at the first level means the buffer of PCO1 is empty, so in this case queueing at the FIFO queue of PCO2 is considered only, in addition to the matching mechanism. Similarly, when the queue of PCO2 is empty but PCO1 is active the model moves on the horizontal axis across the first groups of phases representing queueing at the FIFO of PCO1 and the matching mechanism on the correspondent group of branches. Of course, the other phases describe mixed states, when both queues contain messages.

Description of the transitions are realized by matrices *A*, *B*, *C*, A^*, B^* and C^* that consist of several submatrices, for example matrix *A* and its irregular counterpart is shown here (2).

$$\mathbf{A} = \begin{bmatrix} \underline{\underline{\widetilde{A_1}}} & \underline{\underline{\widetilde{A_0}}} & \underline{\underline{0}} & \underline{\underline{0}} & \cdots & \\ \underline{\underline{\widetilde{A_2}}} & \underline{\underline{A_1}} & \underline{\underline{A_0}} & \underline{\underline{0}} & & \\ \underline{\underline{0}} & \underline{\underline{A_2}} & \underline{\underline{A_1}} & \underline{\underline{A_0}} & \underline{\underline{0}} & \\ & & \ddots & \ddots & \ddots & \\ & & & & \underline{\underline{A_2}} & \underline{\underline{\widehat{A_1}}} \end{bmatrix} ; \mathbf{A}^* = \begin{bmatrix} \underline{\underline{\widetilde{A_1^*}}} & \underline{\underline{\widetilde{A_0^*}}} & \underline{\underline{0}} & \underline{\underline{0}} & \cdots & \\ \underline{\underline{\widetilde{A_2^*}}} & \underline{\underline{A_1^*}} & \underline{\underline{A_0^*}} & \underline{\underline{0}} & & \\ \underline{\underline{0}} & \underline{\underline{A_2^*}} & \underline{\underline{A_1^*}} & \underline{\underline{A_0^*}} & \underline{\underline{0}} & \\ & & \ddots & \ddots & \ddots & \\ & & & & \underline{\underline{A_2^*}} & \underline{\underline{\widehat{A_1^*}}} \end{bmatrix} \qquad (2)$$

Submatrices A_0 and A_0^* describe a step forward (that is downwards vertically) in the embedded QBD. This means, a new message has arrived into the queue of PCO2 (step downwards) while a message has been served from the queue of PCO1 (one step left). In a similar manner, matrices A_1, A_1^*, $\widehat{A_1}$, $\widehat{A_1^*}$, $\widetilde{A_1}$, and $\widetilde{A_1^*}$ contain transition probabilities confi ne to the case that there has been no new arrival on PCO2 (no vertical movement), but exactly one message has undergone service from PCO1's message queue (one step left). Furthermore, matrices A_2, A_2^*, $\widetilde{A_2}$ and $\widetilde{A_2^*}$ are all equal to the zero matrix, because only one message can be under service by the TTCN-3 executor at a time, naturally the meaning of a transition left and upwards at the same time would mean that. In Figure 5, transitions between group of states and the assigned matrices are depicted, restricted to the case that a message is under service (*A* matrices only). Probabilities of the actual phase (states of the embedded QBD) transitions are described by the specifi c elements of the matrices.

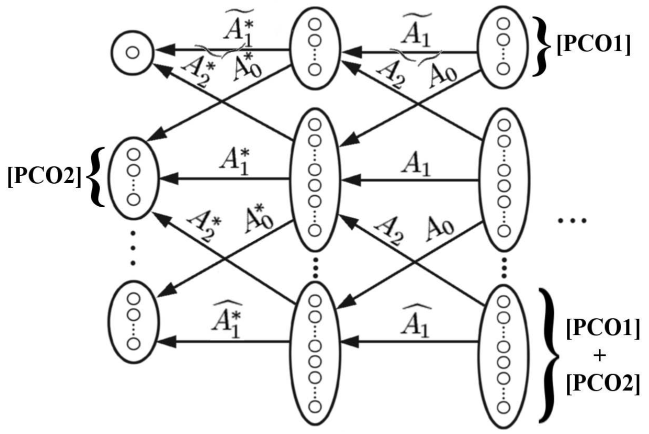

Fig. 5. Possible transitions of the model while a message is under service in PCO1

Important cornerstones of the matrices that build up the generator matrix of the model are the matching probabilities of branches the *alt* contains. Consider these values as predicted hit rate counters for each branch stored in vectors (p), one for each PCO, e.g. in case of two PCOs we apply $p^{'}$ for PCO1 and $p^{''}$ for PCO2. Naturally, length of the *alt* structures (groups of different PCOs) is equal to the length of the hit rate vectors and determines the size of the submatrices among others in Equation 2. Elements of the vectors (e.g. p_i) contain the probability that a message under service does not match the template at branch number *i*. In this case, it is checked against the template at the next branch.

A further important input parameter of the model is the time slot (*T*) needed for one elementary action in the TTCN-3 executor, e.g. the average time needed for the examination of one *alt* branch, the average time in which a template matching operation is finished. Samples of messages traversing the PCOs the test component needs to deal with are taken in discrete *T* time intervals (DTMC).

Furthermore, we describe the arrival intensities of the PCOs involved with a λ assigned to each of them. λ values are relative to the selected elementary *T*, e.g. $\lambda_1 = 1000$ and $T = 10^{-6}$ (1 μsec) means that one message arrives approximately every millisecond ($\lambda_1 \cdot T = 10^{-3}$) on PCO1. Accordingly, we use four different variables to include arrival intensities into the model. D_{00} denotes no arrival at a time slot. D_{10} and D_{01} indicate that one message has arrived on PCO1 or on PCO2 respectively. The meaning of D_{11} is that a message has arrived on both PCOs at the same time concurrently, but this is disallowed by the model, since the TTCN-3 executor handles only one message at a time. This leads to the following scheme (3).

$$D_{00} = 1 - (\lambda_1 + \lambda_2) \cdot T; D_{01} = \lambda_2 \cdot T; D_{10} = \lambda_1 \cdot T; D_{11} = 0; \tag{3}$$

The transition submatrices are constructed using the *D* matrices and the matching probability vectors until every state transition of the state space is covered. For the sake of clarity we present only one submatrix as it is defined (4). Matrix $\widehat{A_1}$ has elements only in the first column and only until the s_1*th* row, where s_1 is the number of branches referring to PCO1 (that is the size of $\underline{p}'$). $(D_{00} + D_{01})$ represents the probability that either no message has arrived or exactly one arrival happened on PCO2.

$$\widehat{A_1} = \begin{bmatrix} (1 - \underline{p}'_1) \cdot \ (D_{00} + D_{01}) & \cdots & 0 \\ \vdots & & \vdots \\ (1 - \underline{p}'_{s_1}) \cdot \ (D_{00} + D_{01}) & \cdots & 0 \\ \vdots & & \vdots \\ 0 & & 0 \end{bmatrix} \tag{4}$$

When the matrices are built up we then calculate the ratio of time spent at the *nth* and *n+1th* level before returning to level *n*, in matrix *R*. *R* is calculated using a logarithmic reduction algorithm described in [8]. The complete distribution of states in the QBD model is then calculated using *R* (5).

$$\underline{P}_1 = \underline{P}_0 \cdot \underline{\underline{R}}; \underline{P}_2 = \underline{P}_1 \cdot \underline{\underline{R}}; \ldots \underline{P}_n = \underline{P}_0 \cdot \underline{\underline{R}}^n \tag{5}$$

Where vector P_i contains distribution of the phases at level *i* and it is not to be confused with p' or with p'' that contain match probabilities, which are architectural constants of the system. After the state distribution is available we can derive performance indices of the system. In this case, we calculate the probability that a message suffers indefinite postponement while in the buffer of PCO2, because PCO1 is under a significantly heavier load. This means that we sum state probabilities at every (horizontal) level, but only those at the vertically last group of states (subsets *l* and *m*). This gives us the following equation, where matrix I is the identity matrix. Practically, we need state distributions only at the first two levels and matrix R for the evaluation.

$$Pr(PCO2loss) = \sum_i P_{oi} + \sum_{j=1}^{\infty} (\sum_k P_{j_k}) = \ldots \underline{P_0}_l + \sum_{j=1}^{\infty} (\underline{P_j}_m) =$$

$$= \underline{P_0}_l + \sum_{j=1}^{\infty}(\underline{P_1}_m \cdot \underline{\underline{R}}^{j-1}) = \underline{P_0}_l + \underline{P_1}_m \cdot \sum_{k=0}^{\infty}(\underline{\underline{R}}^k)$$

$$= \underline{P_0}_l + \underline{P_1}_m \cdot (\underline{\underline{I}} - \underline{\underline{R}})^{-1} \qquad (6)$$

This way, after obtaining steady state distributions of the QBD process, we can predict the probability that messages are postponed in the queue of the PCO that is in lack of system resources because another PCO of the same test component used them up. In the stochastic model this means that we have a new demand incoming in the queue of that PCO that is already at the last level possible, so further messages cannot be received or they will be lost. This analysis allows us to evaluate load test components implemented differentially by setting the parameters of the model accordingly. Also, analysis can be set up to examine a test component with different load requirements as the arrival intensities are parametrical.

5 Model Results

After the transition matrices are defi ned according to the architectural constants, the matching probability vectors and the arrival intensities for each PCO, the model has been built up. Together with the arrival intensities the elementary time slot *T* has to be set also. Besides, one more important input parameter exists, the size of the queue at the PCO we investigate for possible latencies.

Accordingly, considering the simple example in Section 4 we can get the following results on the probability that a message suffers indefi nite postponement in the queue of a PCO. In this example the test component uses two separate PCOs. Say PCO1 is a load generator with considerably high amount of packets generated, while PCO2 handles lower priority operations, such as communicating with the user, or receiving maintenance messages from the main test component. In the analytical model (Figure 6), message loss is calculated according to (6). Different sizes of queues underlying the analyzed PCO are represented by each curve. In Figure 6 and in Table 1, the arrival intensity in PCO1 is considered to be constant, while the arrival rate in PCO2, that is the port we investigate, is variable from 0 to 0.5. An arrival rate ($\lambda \cdot T$) of 0.5 means, in case we have a *T* = 1 μsecond, that approximately 500 messages arrive each millisecond to PCO2.

Simulations in *ns-2*, that has been used for validating the model, show similar behavior. The simulated scenario consists of two PCOs implemented as FIFOs with variable buffer length. Among others, each simulated PCO has a variable length delay loop sequence to simulate the matching mechanism too. Table 1 shows simulated results together with the model results for the same parameters. On one hand we use this simulation to develop the analytic model for more precise results, on the other hand an analytic model is necessary to analyze more complex test scenarios with more than two PCOs involved.

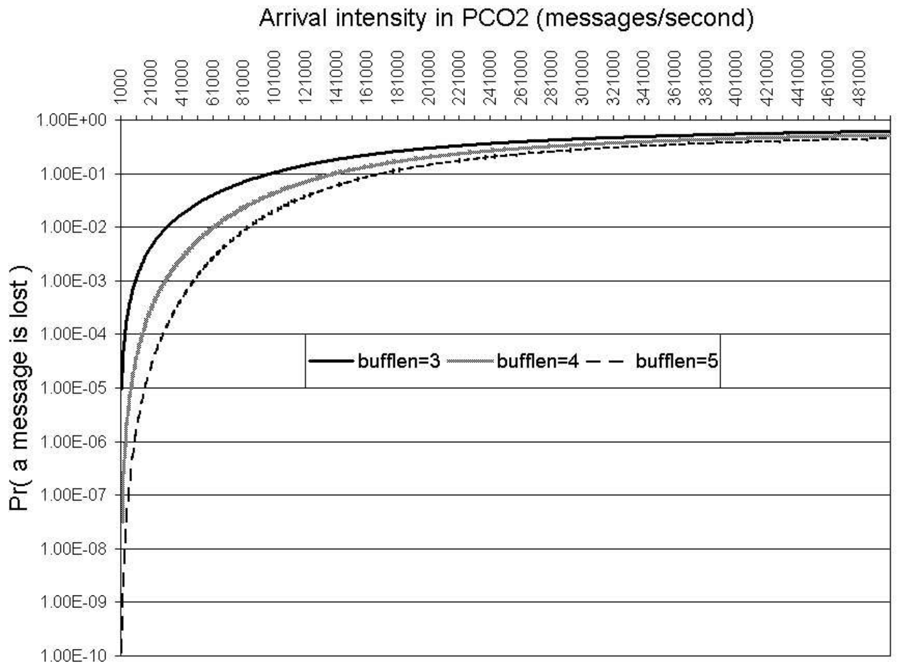

Fig. 6. Model results for different PCO queue sizes

6 Conclusions

Generally, load testing is not an easy task, neither with TTCN-3, nor with any other test environment [13], [14]. Tests that work somehow are not suffi cient, but load tests must also work well. Careful test design and test optimization is a must in these kinds of tests, meaning that tests have to be designed to work effi ciently on the execution platform that accommodates them. Effi ciently, that is e.g. with low CPU load, even using stock hardware elements. To evaluate a load test component and to verify it meets the identifi ed requirements, the complete path traversed by the corresponding messages has to be taken into consideration.

Our analytical model is designed to evaluate load testing TTCN-3 parallel test com-

Table 1. Message loss probability, model and simulation results

Arrival intensity (λ)	Messages per second	Buffer size	Model result	Simulated value
0.08	80000	5	0.00831	0.01252
0.10	100000	4	0.04333	0.04333
0.11	110000	3	0.12333	0.59024
0.12	120000	4	0.06971	0.07333
0.14	140000	4	0.10102	0.14333
0.19	190000	5	0.12718	0.13333

ponents to satisfy user requirements using discrete-time QBDs. In order to be able to design test components that simulate behavior of real nodes in a telecom test network and that are even interchangeable with a real node in terms of performance, modeling of the components performance is nearly inevitable.

This modeling approach makes it possible to evaluate existing test components and also to give a feedback to designers of load test scenarios. Also, the model can aid distribution of load test traffi c among the test components participating in test. The aim of successful traffi c mix composition is to simulate behavior of real nodes, or even to produce an equivalent counterpart of the real node in TTCN-3.

Our future work on this topic includes extending the model to be capable of describing PTCs containing more than two concurrent PCOs at a time. Besides, further simulations and measurements of real components are ongoing and observations are gathered to improve our model beyond the current level.

References

1. ETSI ES 201 873-1 (V3.1.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language", 2005.
2. H. Neukirchen, Z. Ru Dai, J. Grabowski. Communication Patterns for Expressing Real-Time Requirements Using MSC and their Application to Testing. R. Groz, R. M. Hierons (Eds.) TestCom 2004, LNCS 2978, pp. 144-159, 2004.
3. Z. Ru Dai, J. Grabowski, H. Neukirchen. Timed TTCN-3 Based Graphical Real-Time Test Specification. D. Hogrefe and A. Wiles (Eds.): TestCom 2003, LNCS 2644, pp. 110-127, 2003.
4. Z. Ru Dai, J. Grabowski, H. Neukirchen. Timed TTCN-3 - A Real-time Extension for TTCN-3. I. Schieferdecker and H. König and A. Wolisz (Eds.) Proceedings of the IFIP 14th International Conference on Testing Communicating Systems - TestCom 2002, pp. 407-424, 2002.
5. S. McCanne, S. Floyd. ns Network Simulator. http://www.isi.edu/nsnam/ns/
6. I. Schieferdecker, T. Vassiliou-Gioles. Realizing Distributed TTCN-3 Test Systems with TCI. D. Hogrefe and A. Wiles (Eds.) TestCom 2003, LNCS 2644, pp. 95-109, 2003.
7. ETSI ES 201 873-4 (V3.1.1): "Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 4: TTCN-3 Operational Semantics", 2005.
8. G. Latouche, V. Ramaswami. Introduction to Matrix Analytic Methods in Stochastic Modeling. by the American Statistical Association and the Society for Industrial and Applied Mathematics, 1999, pp. 83-99, pp. 221-237.
9. S. Palugyai, M. J. Csorba. Performance Modeling of Rule-Based Architectures as Discrete-Time Quasi Birth-Death Processes. Proceedings of the 14th IEEE Workshop on Local and Metropolitan Area Networks, 2005, Chania, Greece.
10. M. F. Neuts. Matrix-Geometric Solutions in Stochastic Models. Johns Hopkins University Press, 1981, pp. 81-107, pp. 63-70, pp. 112-114.
11. T. Osogami, A. Wierman, M. Harchol-Balter, and A. Scheller-Wolf. A recursive analysis technique for multi-dimensionally infinite Markov chains. Performance Evaluation Review, 32(2):3-5 (2004).
12. T. Osogami. Analysis of a QBD Process that Depends on Background QBD Processes. Technical Report CMU-CS-04-163 (2004).
13. G. Rößler, T. Steinert. Traffic Generator for Testing PABX and Call Center Performance. I. Schieferdecker and H. König and A. Wolisz (Eds.) Proceedings of the IFIP 14th International Conference on Testing Communicating Systems - TestCom 2002, pp. 139-151, 2002.

14. I. Acharya, H. Kumar Singh. Testing of 3G 1xEV-DV Stack - A Case Study. D. Hogrefe and A. Wiles (Eds.): TestCom 2003, LNCS 2644, pp. 20-32, 2003.