Using Extended Petri Nets for Modeling and Simulation of Queuing Systems with Priorities

Mehmet Karay
Management Information Systems Department
Girne American University
Kyrenia, Cyprus
mehmetkaray@gau.edu.tr

Alexander Kostin
Management Information Systems Department
Girne American University
Kyrenia, Cyprus
alexanderkostin@gau.edu.tr

Abstract— **Theoretical study of queuing systems with priorities has been done previously for some specific probability distributions of arrivals of customers, mainly for Poisson input flow. But in practice, there is often a need to use queuing systems with priorities for arbitrary laws of input flows of arriving customers and arbitrary probability distributions of service time. The study of such general queuing systems can be done most easily with their simulation modeling. This paper presents creation and study of simulation models of queuing systems with relative and absolute priorities with the use of a class of extended Petri nets. A brief summary of such Petri nets, with a language to describe the models and simulation tool, is given. Then models of two queuing systems with relative and absolute priorities are described. Characteristics of these models (time of a customer in system and waiting time) were investigated in extensive simulation experiments. Results of simulation are compared with results of analytical modeling.**

Keywords; queuing systems with priorities; extended Petri nets; simulation.

I. INTRODUCTION

Analytical study of queuing systems with priorities of customer or request classes has been undertaken in a few previous works for some specific probability distribution of arrival of customers, mainly for Poisson input flow [1], [2], [4]. Theoretical study of such queuing systems for arbitrary probability distribution of input flow of customers is a challenging problem. Recently, work [3] has been published, in which the input flow of arriving customers is assumed to be hyper exponential.

However, in different areas of practical application of queuing systems, such as telecommunication, computer networks, distributed systems, traffic control etc., there is often a need to use queuing systems with priorities for arbitrary probability laws of arriving customers and arbitrary probability distributions of service time. The study of such general queuing systems can be done most easily with their simulation modeling.

There are a number of simulation tools suitable for simulation modeling of queuing systems with priorities. Probably one of the oldest simulation tools oriented to simulation of queuing systems is FPSS, with the current version known as GPSS world [5].

Although this simulation tool provides a good user interface and is easy to use, it is limited in its capability to express, in a clear form, parallel events and processes as functions of time in models. In addition, GPSS is poor of debugging facilities and models, developed with GPSS, are slow, which is especially notable for large models.

In this paper, we propose to use a class of extended Petri nets for modeling and simulation of queuing systems with priorities. In general, Petri nets very clearly and naturally represent parallelism of events and processes in all types of information systems. They are very convenient also for representation of different types of process synchronization and interaction. Oriented mainly to analysis of structural properties of the discrete systems, Petri nets have well developed formal methods to determine such important properties of systems as reachability, deadlocks and safety.

General (or classic) Petri nets are described in detail in [6]. Unfortunately, general Petri nets are not suitable for simulation, since they do not represent an algorithmic system. They are used only for structural analysis of discrete systems.

In contrast with general Petri nets, extended Petri nets, applied in this paper, represent a complete algorithmic system and are appropriate for simulation modeling. The detailed description of this class of extended Petri nets, with numerous simulation models and their solutions, is given in [9]. We will show that the use of extended Petri nets gives a possibility to design very simple simulation models that clearly represent all aspects of their behaviour in time and produce necessary performance metrics in a convenient form. It is important to note that, in contrast with the existing theory, developed for restricted probability laws of input flow of requests and their servicing, the proposed models can be used for arbitrary cases.

The rest of the paper is organized as follows. In Section 2, theoretical aspects of queuing systems with priorities are outlined. Section 3 surveys extended Petri nets and gives a brief information about the simulation language MDL, used to create simulation models in terms of extended Petri nets. In Section 4, the model of a queuing system with relative priorities is described and investigated. Section 5 describes the model of queuing system with absolute priorities and presents results of its simulated study. Finally, Section 6 concludes the paper.

II. THEORETICAL ASPECTS OF QUEUING SYSTEMS WITH PRIORITIES

The theory of general queuing systems is studied quite well. But exact theoretical expressions for characteristics of queuing systems are derived mainly for queuing systems with the random Poisson arrival rate of requests or customers. For such a type of arrival rate, interarrival intervals are exponentially distributed, so that the corresponding systems are considered as Markovian models [7].

Queuing systems with multiple classes or multiple priorities of requests are usually not described in sufficient detail in general books on queuing theory. Probably, one of the first books on theoretical study of priority queues is [1]. Again, in this book and in some subsequent published works the input flow is assumed to be Poisson one.

There are two basic types of priorities of requests – relative priorities and absolute priorities. With relative priorities, servicing of the current request is not interrupted, even if new requests of higher priority arrive. It is said that such a queuing system implements non-pre-emptive service. Only after complete servicing of the current request, a new request with the highest priority among the arrived requests will be chosen for servicing. Such a queuing system is usually implemented with separate input queues for requests of different priorities.

With absolute priorities, servicing of the current request is interrupted, if a new request with a higher priority arrives. It will be chosen for the immediate servicing. The corresponding system in this case implements pre-emptive service.

The interrupted request can be treated with the use of a few disciplines, such as pre-emptive resume, pre-emptive repeat identical, pre-emptive repeat different and pre-emptive loss [8]. The most often used discipline is pre-emptive resume, according to which servicing of the interrupted request will be resumed as soon as all requests of higher priorities are completely serviced.

To be capable of comparing results of simulation with theoretical results, we will give expressions for mean time of a request in system and mean time of waiting by a request [4]. It is assumed that the queuing system is of type *M*/*M*/*n*. That is, the input flow of requests is Poisson, service time is distributed exponentially for each of *n* identical servers working in parallel.

For the queuing system with relative priorities (non pre-emptive servicing), mean time of a request in system and mean waiting time are as follows:

$$S_i = \frac{1}{AB_{i-1}B_i} + \frac{1}{\mu}, \quad (1)$$

$$W_i = \frac{1}{AB_{i-1}B_i}, \quad (2)$$

where $i = 1, 2, \ldots, k$ is the class number, with class 1 having the highest priority, k is the number of classes(with class k of the lowest priority), and

$$A = D\mu + \frac{D!(D\mu - \lambda)}{\rho^D} \sum_{j=1}^{D-1} \frac{\rho^j}{j!}, \quad (3)$$

$$B_i = 1 - \frac{1}{D\mu} \sum_{j=1}^{i} \lambda_j, \ i = 1,2,\ldots,k, \quad (4)$$

with $B_0 = 1$.

In expression (3) and (4), D is the number of identical servers, $\rho = \frac{\lambda}{\mu}$ is the load servers, λ is the total arrival rate of requests. $\frac{1}{\mu}$ is mean servicing time of a server and λ_j is arrival rate of requests of class j. Note that if D>1, then load of each server is $\frac{\lambda}{D}$. With D=1, we have $A = \frac{\mu}{\rho}$.

For the queuing system with absolute priorities (pre-emptive servicing), mean time of a request in system and mean waiting time are following:

$$S_i = \frac{1}{\mu B_{i-1} B_i}, \quad (5)$$

$$W_i = S_i - \frac{1}{\mu}, \quad (6)$$

where $i = 1, 2, \ldots, k$ are priority class numbers, k is the number of classes, with class 1 having the highest priority and $B_0 = 1$. Expression for B_i is given in (4).

III. A SURVEY OF A CLASS OF EXTENDED PETRI NETS

As was stated in the introduction, general Petri nets, oriented from works of K. Petri in 1962, are not suitable for simulation of discrete-event systems. We give here a brief survey of a class of extended Petri nets that represent a complete algorithmic system and were used in this paper for simulation modeling. These nets are timed Petri nets, with attributed tokens and the abilities to control transfer of information, carried with token attributes, and to process this information during transition firing.

The structural elements of extended Petri nets are places, transitions and directed arcs. There are two types of places in these nets-simple and queue places. A simple place or S-place (represented graphically as a circle) can hold one token at most, while a queue place or Q-place (depicted in the form of an oval) can accumulate an arbitrary number of tokens.

The minimal building blocks of extended Petri nets are elementary nets. Each elementary net consists of only one transition and a few incident input and output places. In the used extended Petri nets, there are five basic types of elementary nets, denoted by T, X, Y, I and G. Each type has its own structure and functionality. By connecting elementary

nets with each other, one can create Petri nets of arbitrary size and complexity.

For the purpose of this paper, only elementary nets of types T, Y, X and I were used. The following paragraphs contain brief descriptions of these types [10].

Elementary net of type T is used for time delay and data processing. When its transition fires, the net merges tokens from all input places and creates new tokens in all output places. The transition can fire only if each input place contains a token and all simple output places (S-places) are empty. At the end of transition firing, the default data processing function, associated with this net, copies attributes of the token of the first input place and assigns the values of these attributes to each new token in all output places. If necessary, any desired data processing function can be performed on attributes of new tokens in output places. The net of this type actually implements logical operation AND on input places. Note that, in general, any non-zero time delay can be associated with the transition. This is true also for all other types of elementary nets. In addition, this elementary net, without input places, can be used as a token generator.

Elementary net of type Y logically implements the conditional multiplexing operation by selecting one token from one of the input places when the transition fires. As a result of transition firing, one token will be created in each output place. Values of attributes of the token, selected from an input place, are copied and assigned to corresponding attributes of new tokens created in all output places. On default, the token from the first non – empty input place is selected. But with an explicitly assigned input control function, the selection can be done from any other non – empty input place. Note that tokens in the remaining input places are not affected. Without input places, this elementary net behaves like a net of type T without input places.

Elementary net of type X routes a token from the first input place to one of the output places. For firing of the transition, it is necessary that each input place holds a token and the selected output place is empty. When the transition fires then, on default, attributes of the first place are copied and their values are assigned to attributes of a new token created in one of the output places.

Elementary net of type I, shown in Figure 1, provides a possibility to interrupt the firing interval of transition by an external event. As shown in the figure, this net has two input places x_1 and x_2 and two output places y_1 and y_2. Place x_1 is the main input, while place x_2 serves as an interrupting place. Correspondingly, y_1 and y_2 are outputs for the main input and for the interrupting input, respectively. The arrow symbol on the transition bar is directed from the interrupting input to the main input.

There are two main scenarios in the operation of this elementary net. First, if there is a token in place x_1, but place x_2 is empty, the net works as usual net of type T with two incident places x_1 and y_1. Second, if, during started firing of transition, a token appears in place x_2, then the ongoing firing instantly stops, the token from place x_1 goes to place y_1 with its attributes, and the interrupting token from place x_2 goes, with its attributes and zero delay, to place y_2. After the interruption, the tokens in place y_1 and y_2 can be treated separately in subsequent parts of the model. Usually, the interrupting token will initiate some interruption procedure by this same net, while the interrupted token will wait until the net completes processing the interruption event.

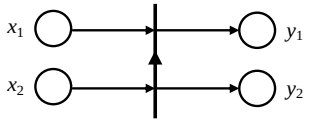

Figure 1. Elementary net of type I.

Detailed description of the elementary nets can be found in [9].

For the description of models in terms of extended Petri nets there is the Model Description Language (MDL). The MDL is an extension of object Pascal, with added features for the description of elementary nets, token attributes and initialized data. There is also Modelling Control Language (MCL), that is used to parameterize the compiled models and control their runs on the computer. All these capabilities are implemented in the simulation system Winsim. Its software, user guide and many examples of solved models are available on the CD attachment of [9].

IV. MODEL OF A QUEUING SYSTEM WITH RELATIVE PRIORITIES

In this section, the model of a queuing system *M/M/*1 with relative priorities of input requests (non pre-emptive servicing) is described and investigated. For validation, performance of the model is compared with its theoretical performance. Corresponding theoretical aspects of this type of a queuing system are given in Section 2.

Figure 2 shows the Petri nets scheme of the model. It is assumed, that arriving requests (or customers) belong to 3 priorities 1, 2 and 3, with the highest priority of 1. Requests for servicing are generated by transition T1. Transition X1 routes each request to one of input queues represented by places Q1, Q2 and Q3, depending on the priority of the request. In particular, requests of the highest priority 1 are inserted into queue Q1, while requests of the lowest priority 3 are put into queue Q3.

The server is represented by transition T2. In general, each of queues Q1, Q2 and Q3 can contain requests. When the server becomes idle at the end of the firing of transition T2, then transition Y1 fires and selects the next request for servicing from queue Q1 or Q2 or Q3, in this order. That is, the next request for servicing will have the highest priority from all requests in queues Q1, Q2 and Q3.

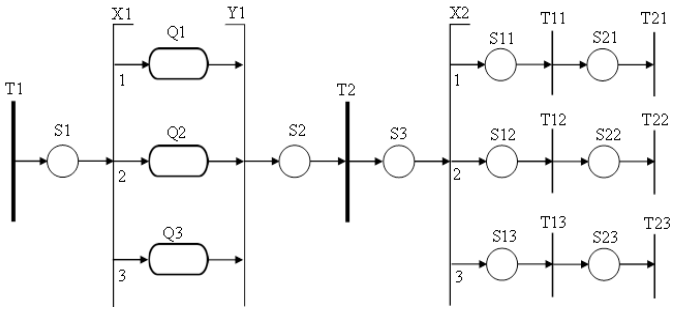

Figure 2. Petri net scheme of a queuing system with relative priorities of servicing.

Each serviced request will be routed by transition X2 to one of three outputs 1,2 and 3 according to its priority. Transition T11, T12 and T13 tabulate time in system for each seviced request, and transitions T21, T22 and T23 absorb the requests.

In the figure, transitions T1 and T2 are shown in bold to indicate that time delay (duration of transition firing) associated with these two transitions are not zero. In particular, time delay of transition T1 represents interarrival time of requests, while time delay of transition T2 is equal to servicing time of requests. Both delays are random and, according to *M/M/*1 queue, correspond to exponential probability distribution. Note that transition T1 randomly assigns one of the three priorities and current simulation time to each generated request. The attribute of the current simulation time is necessary for measurement and tabulation of time in system for each request.

Although the model tabulates only time in system for each request, other characteristics can be obtained from raw results of simulation. In particular, average token time in queues Q1, Q2 and Q3 is average waiting time of a request for each of the three priorities. Further, the number of firings of transition T1 is the total numbers of firings of transitions T11, T12 and T13 are the numbers of completed requests of the three priorities.

To give an idea of the Model Description Language (MDL), used to input the model into the simulation system, in Figure 3, we give the complete source text of the model. Due to space limitation, description of this text is omitted, but the reader can easily relate the text to the scheme of Figure 2.

The simulation has been done for the server loads of 0.1, 0.2,…, 0.9, with mean (assigned) service time of 750 ms per request and varying interarrival time. The simulation interval for each run has been set to 10^7 ms to obtain sufficiently large volume of statistics data. As was mentioned, only time in system per request was tabulated for each of three priorities. For definitions, it was assumed that for each priority requests come with the same arrival rate. Thus, if the server load $\rho = 0.9,$ then the total arrival rate $\lambda = \frac{750}{\rho} = 833.33$ ms, and arrival rates of requests of three priorities are $\frac{\lambda}{3}$ = 277.78 ms. Correspondingly, with total server load $\rho,$ loads caused by requests of each of three priorities are $\rho_1 = \rho_2 = \rho_3 = \frac{\rho}{3}$. In particular, for $\rho = 0.9, \rho_1 = \rho_2 = \rho_3 = 0.3.$ Equal arrival rate of requests of each of the three priorities is achieved by using the same probability of generation of requests for each priority. This is done in statement TRANS T1 in the model text.

```
SEGMENT RELPRT, TICK=msec;
ATTRIBUTES
  PRTY : INTEGER;                  (* Priority of request   *)
  TS   : REAL;                     (* Service mean time     *)
  TIM  : REAL;       (* Time of arrival or time in system  *)

 DATA
  TGEN /0.0/ : REAL;          (* Mean interarrival time  *)
  TSRV /0.0/ : REAL;            (* Mean service time     *)
  PROB /0.0/ : REAL;             (* Probability value    *)

(*   Generation of arrivals of customers and assigning    *)
(*                 priorities to each of them             *)

NET T1 : / S1;
TIME T1 : % DELAY := EXPON(1, %TGEN);
TRANS T1 : % S1.TS  := EXPON(1, %TSRV);
     % S1.TIM := clock(1);
     % PROB := FRANDOM(1);
     % S1.PRTY := 1;                  (* Default priority *)
  if % PROB <= 0.333 then % S1.PRTY := 2;
  if % PROB > 0.666 then   % S1.PRTY := 3;

NET X1 : S1 / Q1, Q2, Q3;
CONTR X1 : %OUT :=  % S1.PRTY;
NET Y1 : Q1, Q2, Q3 / S2;
NET T2 : S2 / S3;
TIME T2 : % DELAY := %S2.TS;
TRANS T2 : % S3.TIM := CLOCK(1)-%S2.TIM;
NET X2 : S3 / S11, S12, S13;
CONTR X2 : % OUT := %S3.PRTY;

(*     Register time in system for different priorities    *)
(*                  and absorb a token                    *)
(*                     Absorbers                          *)

NET T11 : S11 / S21;
NET T21 : S21;
NET T12 : S12 / S22;
NET T22 : S22;
NET T13 : S13 / S23;
NET T23 : S23;
SEGEND.
```

Figure 3. Source text of the model of a queuing system (in MDL).

Figure 4 shows obtained graphs of time in system for each of three priorities versus server load. Each server load in the figure is the total load caused by requests of all priorities.

For comparison with the theory, the figure also contains graphs, calculated with the use of expression (1). As one can see, simulated and theoretical results are very close to each other. Actually, for priorities 2 and 3, the simulated and theoretical graphs are almost indistinguishable, and only for highest priority 1 and server load of 0.9 there is a noticeable difference, that is caused by the instability of simulation results for high server loads.

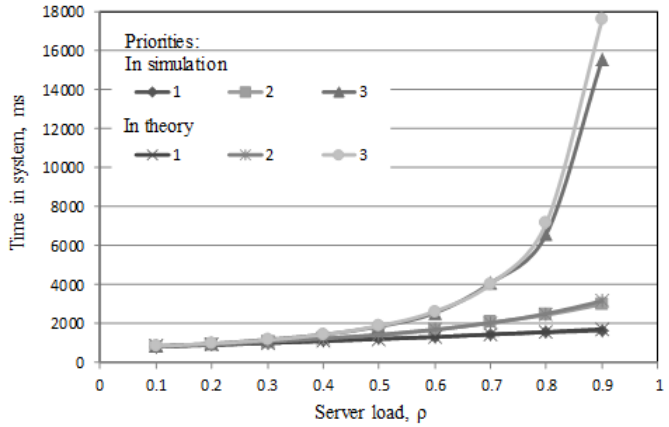

Figure 4. Graphs of time in system vs. server load for the queuing system with relative priority.

V. MODEL OF A QUEUING SYSTEM WITH ABSOLUTE PRIORITIES

Theoretical characteristics of *M/M/*1 queuing system with absolute priorities (pre-emptive servicing) are presented in Section 2. Recall that in such a system, a request that is being serviced will be interrupted by a new arriving request with higher priority. However, requests of the highest priorities are never interrupted, even by new requests of the same priority.

Figure 5 shows a scenario of servicing of requests with three priorities or classes 1, 2 and 3, with the highest priority of class 1. It is assumed, in this scenario, that up to time t_1 there are no requests for servicing. At time t_1 a request of class 3 arrives and is being serviced up to time t_2, when a request of class 1 arrives, interrupts the servicing of request of class 3, is serviced up to time t_3 and leaves the system. Since time t_3 servicing of request of class 3 continues up to time t_4, when a request of class 2 arrives. This request interrupts the servicing of request of class 3 and is being serviced up to t_5, when it is interrupted by a new request of class 1. As shown, the servicing of request of class 2 and 3 terminates at times t_7 and t_8, respectively.

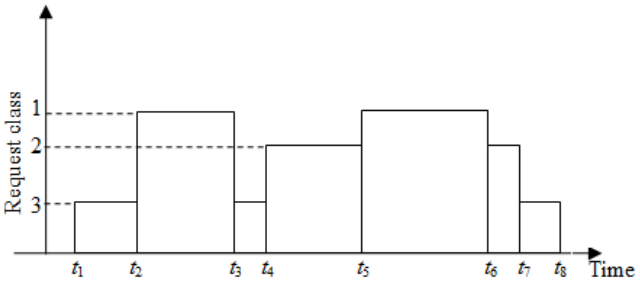

Figure 5. A scenario of servicing of requests of three classes (priorities).

Figure 6 presents the scheme of this queuing systems for three classes of requests. Ariving requests are modeled by tokens appering in place S1000 as a result of their generation by transition T1000. Elementary net X10 routes each request into place S2 or, if the priority of the arrived request is higher than the priorty of the current serviced request, into place S5. From place S2 the incoming request will be added (by transition X2) into one of queues Q1, Q2 or Q3 according to its priority. It is assumed here (as in the model with rerlative priorities) that priority 1 is the highest, and priority 3 is the lowest.

Servicing of each request is modeled by transition I4 of interruption type. In this transition, S4 is the main place, and S5 is the interrupting place.

If a token comes to place S5, the servicing of the current request will be interrupted (postponed). In this case the token, representing the interrupting request, goes from place S5 into place S3 and the token representing the interrupted request goes from place S4 to place S6. From place S3 the interrupting token, through transition Y3, will immediately activate transition I4 to model start of servicing of the interrupting request. The token, representing the interrupted request, goes, through place S6, transition X6, place S8 and transition X8 into the head of its corresponding queue. Note that the request of the highest priority is not interrupted, so that there will never be a token in place S10 (but only in places S20 and S30, representing heads of queues Q2 and Q3).

In any case, when servicing of a request is finished, its token will go through place S6, transition X6 and place S7 to activate transition T7. This transition puts a token into place S1 to set the server idle and into place S9 for the final processing. This is done by transition X9, that determines the time of each request in the system and routes the processed request into one of places S100, S200 or S300 for absorbing by transition T100, T200 or T300, respectively.

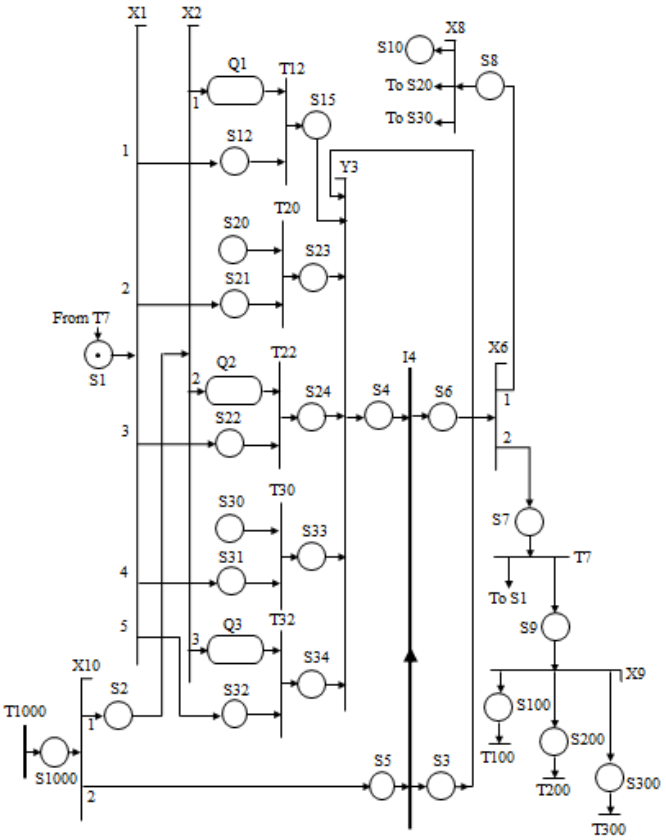

Figure 6. Petri net scheme of a queuing system with absolute priorities (for three priorities).

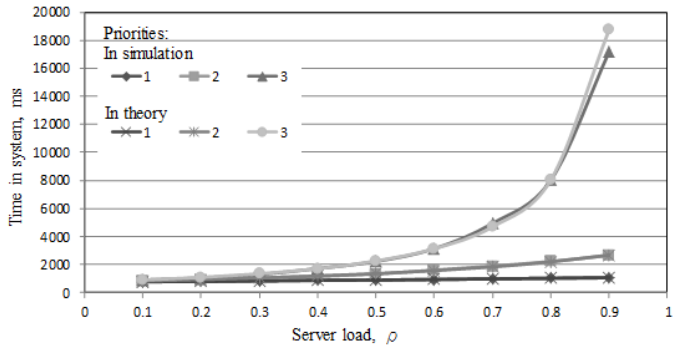

Figure 7. Graphs of time in system vs server load for the queuing system with absolute priorities.

Figure 7 shows graphs of request time in system versus server load. As for queuing system with relative priorities, the simulation results are quite close to theoretical results. Again, for low server loads, average time of request in system is almost the same for different priorities. Since, for low server loads, service interruptions are rare, the queuing system with absolute priorities behaves like the system with relative priorities. However, for high server loads, the graphs of Figure 7 are noticeable different from graphs of Figure 4. Indeed, in the system with absolute priorities the time of a request in system is smaller for the highest priority and is considerably larger for requests of the lowest priority. This is in complete agreement with the logics of operation of the queuing system with absolute priorities.

VI. CONCLUSION

The use of extended Petri nets for modeling and simulation has been explained and demonstrated for investigation of two types of queuing systems. The developed models correspond to *M/M/*1 queuing system. However, the models can be easily updated for arbitrary probabilistic laws of servicing time and interarrival time just by choosing desired random varieties functions from those that are available in the simulation system. This will require only the replacement of functions EXPON() in the model text.

The model can be easily extended to simulate any desirable distribution of arrival rates among the priorities and any number of different priorities.

Finally, the modeler may also increase the number of servers working in parallel just by adding respective transitions.

Together with the formalism of extended Petri nets, all these possibilities make the developed models a useful tool for the experimental study of queuing systems.

REFERENCES

[1] N.K. Jaiswal, Priority queues, Academic Press, 1968.
[2] L. Kleinrock, Queuing systems vol. 2: Computer applications, Wiley,1976.
[3] A. V. Ushakov and V.G. Ushakov, "Queue length in an absolute priority system with hyper – exponential input flow", Moscow University: Computational Mathematics and Cybernetics, vol. 36, no. 1, pp28 – 35, 2012.
[4] S. L. Baver, Queuing system Theory Cookbook, 2006.
[5] H. Rathbauer, Angewandte Simulation unit GPSS world für Windows, Logos Verlag Berlin, 2003.
[6] T. Murata, Petri nets: Properties, Analysis and Applications, Proc. IEEE, vol. 77, no. 4, pp. 541 – 580, 1989.
[7] J. Banks, J. S. Carson, B.L. Nelson, and D. M. Nicol, Discrete – Event system simulation, 3rd ed., Printice Hall, 2001.
[8] H. Takagi, Queuing Analysis 1: A foundation of Performance Evaluation: Vacation and Priority Systems, North – Holland, 1991.
[9] A. Kostin and Ilushechkina L, "Modeling and simulation of distributed systems", *World Scientific Publ. Co.*, 2010.
[10] Y. Fanaeian and A. Kostin, "Simulated Study of an Anycast – Based Routing Method for Wireless Sensor Networks with the use of Petri nets", International Journal of Science and Advanced Technology, vol. 3, no. 4, pp.18 – 27, 2013.