Application of dynamic priorities for controlling the characteristics of a queuing system

S V Polyanskiy[1], **Yu Ya Katsman**[1]

[1] Tomsk Polytechnic University, 30, Lenina Ave., Tomsk, 634050, Russia

E-mail: Katsman@tpu.ru

Keywords: model, queuing system, priority-free discipline of expectation and service, relative priority, absolute priority, queue, FIFO, dynamic priority.

Abstract. This paper considers the development and modification of an imitation model of a queuing system. The initial model uses the laws of control (discipline of expectation and service) with mixed priorities. The work investigates the model with three types of entities (absolute priority, relative priority and priority-free ones) in the regime of overload, i.e. a system with losses. Verification and validation of the created imitation model confirmed its adequateness and accuracy of received results. The application of dynamic priorities for changing the laws of model control substantially alters certain system characteristics. The creation of the model in MatLab Simulink environment with the use of SimEvents and Stateflow library modules allowed creating a fairly complex queuing system and obtain new interesting results.

1. Introduction

The work presents the developed and partially modified imitation model of a queuing system (QS). The model was developed in the MatLab Simulink environment [1] using the SimEvents library [2], which allows modeling discrete states of a QS, control the advancing of entities in queues, etc. The library has blocks (subroutines) intended for analyzing the model performance characteristics, which allow specifying or changing the priorities of entities, collecting and analyzing statistics of input and output flows for each type of entities. The model was modified [1] with the use of Stateflow library [3], which helps the researcher to model combinatory and sequentional logic of decision-making. The presented work used Stateflow blocks for dynamic change of the imitation model functioning laws (logic).

The following parameter were set for the QS model:

– Three sources of entities: priority-free, priority (relative priority), high-priority (absolute priority).
– Departure of non-served entities after queue filling.
– One server.
– Entity arrival time has a uniform distribution law.
– Entity service time has an exponential distribution law – the higher the entity priority, the lower the serving time in the server:

- priority-free $T_{average}$ = 10;

- medium-priority $T_{average}$ = 5;
- absolute priority $T_{average}$ = 3;

– Modeling time is 1000 units.

2. Results and discussion

A. QS Structure

The QS was studied under the overload regime; if the entity queue reached maximum, the entity is rejected. The serving discipline, according to which priority-free entities get into the server only in the absence of entities with high priorities in the queue, enables the situation when the fraction of served low-priority entities becomes negligibly small. This questions the reasonability of their presence in the system. If the exclusion of a flow of priority-free entities from the service is impermissible, this problem can be solved using a dynamic increase of priority, if the flow (relative flow) of priority-free entities is lower than the threshold value.

The developed QS consists of the following main blocks: a generator of entities and a parameter setter, departure of entities in case of a full queue (rejection), a queue, a priority increase, a server, return of partially served entities, departure of served entities. The structural scheme of the QS is presented in Fig. 1.

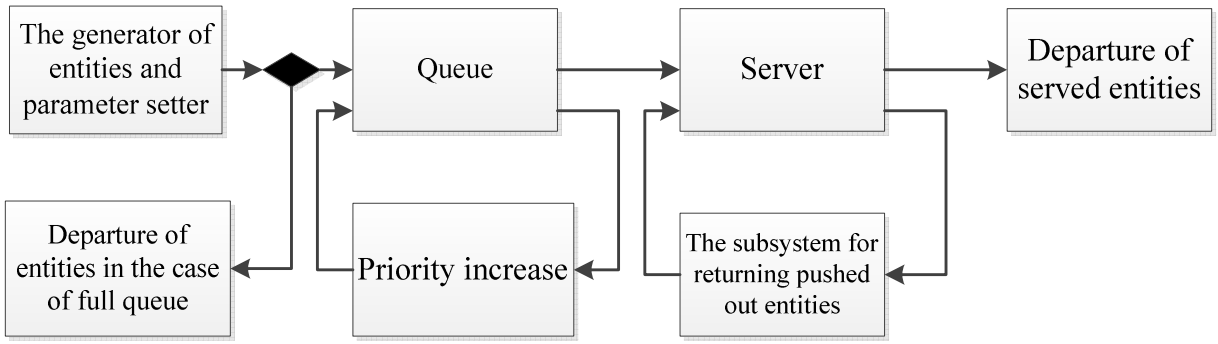

Figure 1. The scheme of QS functioning

The generator of entities and a parameter setter provides the entry of an entity into the system at the time corresponding to the entity type and its distribution density. In this block, the attributes are also set.

Departure of entities in case of a full queue is the subsystem that calculates the number of entities unserved due to a full queue.

The *Queue* block is for saving entities, if the server is busy; if the server is free, then the entity goes into the service according to its priority. First, the entities with absolute priority are served, then – the entities with relative priority, and lastly – priority-free entities.

Priority increase is a subsystem used when the percentage of served priority-free entities is low, and demand in the increased quantity of served priority-free entities arises. Here, priority (relative one) is assigned to a priority-free entity, but the time of its processing in the server remains the same.

Server is a block that has a system for interruption of the entity service with any priority in case an entity with absolute priority enters the system. A partially served entity is pushed from the server to the queue for afterservice, where all its attributes and time necessary for its final service are stored.

The subsystem for returning pushed out entities sends the entity from the queue for afterservice into the server immediately after serving an entity with absolute priority.

Departure of served entities is the subsystem that terminates the entity passage in the model; it calculates the number of served entities according to their priorities.

B. The subsystem of the queue and dynamic priority increase

A number of works study queues with dynamic priorities [4]. However, the practical use of

analytical results received for primitive QS is problematic. Work [5] implemented the increase of the priority of a priority-free entity, if its queuing time exceeded the allowed time. The estimation of this time involved preliminary test calculations, which is not always practical. In this connection, the subsystem of the queue and priority increase of this model was substantially reworked. The structural scheme of the queue is depicted in Fig. 2.

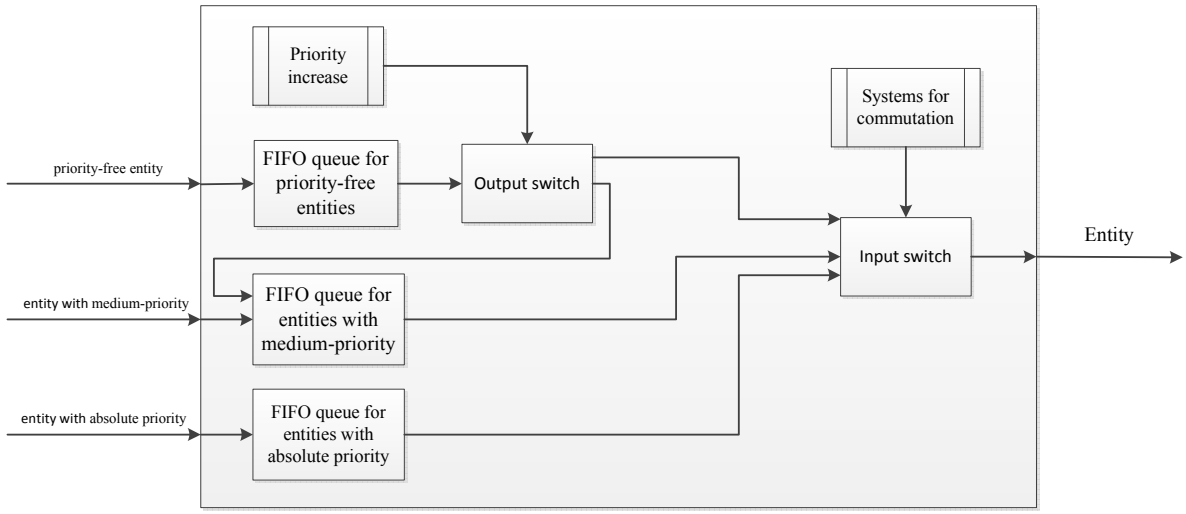

Figure 2. The structural scheme of the queue

The subsystem of the queue (Fig. 2) consists of the Input Switch block, the Output Switch block, systems for the commutation and priority increase, and three FIFO queue blocks, one block for entities of each type.

After the entity was generated, it gets into a corresponding queue. The Input Switch block in conjunction with the commutation system provides the operation of the whole block as a priority queue. The commutation system checks the presence of entities in queues starting from the highest priority, and if this queue is not empty, it commutates it to the exit from the subsystem. Thus, the system first checks the presence of entities in the FIFO queue for entities with absolute priority, and if the queue is not empty, then the Input Switch block commutates this queue with the exit; if it is empty, then the FIFO queue with relative priority is checked, and so on.

The priority increase system works in conjunction with the Output Switch block. When the model is launched, a threshold value is specified, a minimum percentage of priority-free entities should be maintained by the system. The system (Fig. 3) checks the percentage of served priority-free entities; if it is lower than the threshold value, then this system commutates the queue of priority-free entities with the queue of entities with relative priority by throwing two entities from the priority-free queue into the queue with relative priority. After that, the system again checks the percentage of served entities, and after 10 units of time accounts the inertia of the QS.

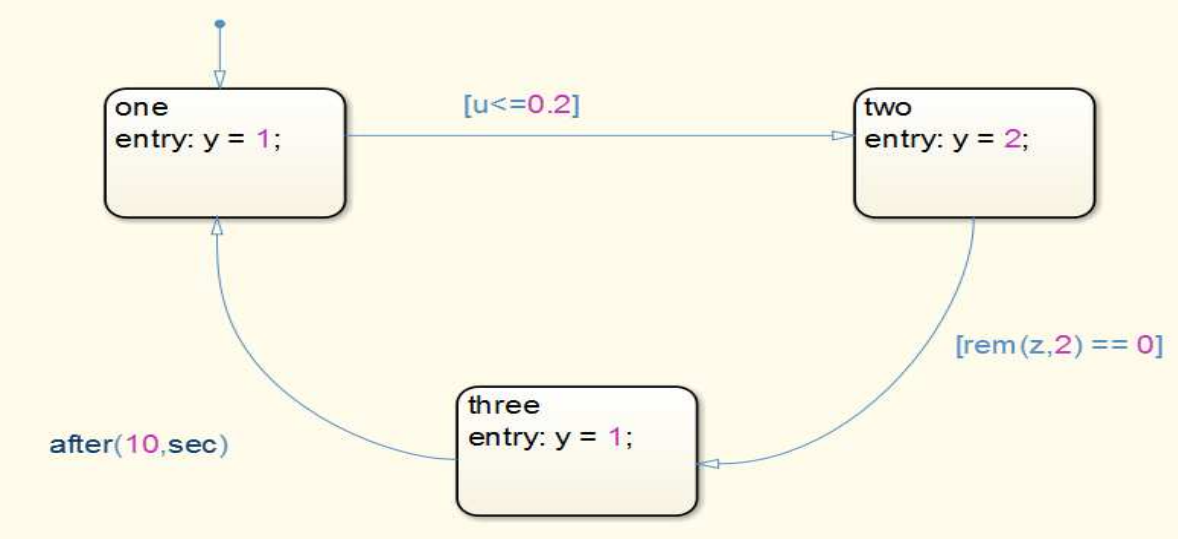

Figure 3. The priority increase system

If the percentage of served priority-free entities is sufficiently high, and there is no necessity in increasing the probability of serving a priority-free entity, then this block commutates with the Input Switch block.

C. QS with dynamic priorities

After the modification, the QS model looks as follows.

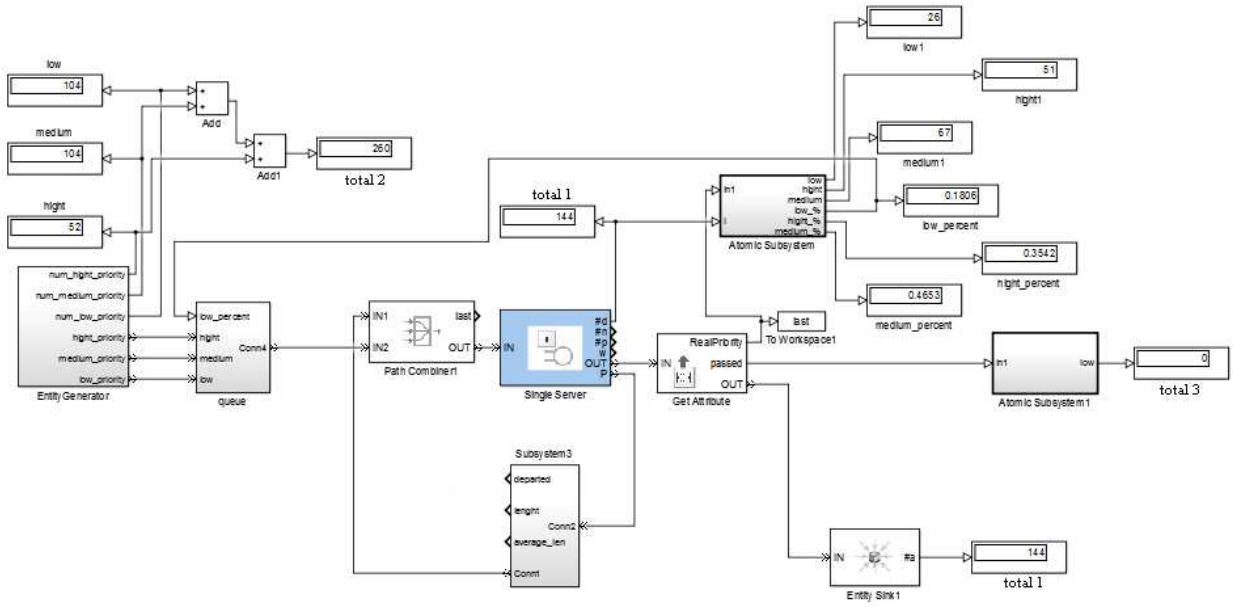

Figure 4. The QS model with dynamic priorities

During the development of an imitation model, one of the most important stages is the testing of the system [6, 7]. Taking into account these requirements, a number of tests were performed for the QS (Fig. 4). The received results are given in Table 1.

Table 1. QS characteristics with dynamic priorities

No.	Priority increase system	N_1	N_2	N_3	N	% 1	% 2	% 3	K	P_{serv}
1	-	31	97	51	179	17.31	54.18	28.49	98.28	0.6884
	+	35	79	51	165	21.21	47.87	30.9	98.28	0.6346
2	-	22	126	36	184	11.95	68.47	19.56	99.14	0.6666
	+	35	103	36	174	20.11	59.19	20.68	99.14	0.6304
3	-	28	94	43	165	16.96	56.96	26.06	98.28	0.7674
	+	33	84	43	160	20.62	52.5	26.87	98.28	0.7441
4	-	63	50	27	140	45	35.71	19.28	97.74	0.7608
	+	63	50	27	140	45	35.71	19.28	97.74	0.7608
5	-	56	50	46	152	36.84	32.89	30.26	97.28	0.8539
	+	56	50	46	152	36.84	32.89	30.26	97.28	0.8539

Legends: +/- is on/off of the priority increase system;
N is the total number of served entities;
N_i is the number of served entities with priority i;
% i is the percentage of served entities with priority i;
K is a server load coefficient;
P_{serv} is probability of the entity service by the server.

3. Conclusions

The received results testify that there are no losses of high-priority entities, except of the cases, when a high-priority entity is in the server at the moment of modeling end.

The use of dynamic priorities increases the priority of a priority-free entity only once; in this case, the probability of serving a flow of such entities increases. At the same time, the number of served entities with relative priority decreases. Moreover, such entity with priority preserves the average time of the priority-free entity service. And since priority-free entities are served for a longer time, the number of entities served by the system will decrease. This property is observed with dynamic priorities turned on (Table 1, P_{serv}). In experiments 4 and 5, the initiation of the priority increase system did not change the QS characteristics, since the flow of served priority-free entities exceeded the threshold value and dynamic priorities did not turn on (Table 1).

References

[1] Katsman J.J., Apachidi X.N. 2014 Algorithm Simulation of Resource Allocation of the Queueing Systems, Based on the Priorities. *Proceeding of 2014 International Conference on Mechanical Engineering, Automation and Control Systems.* pp 1- 6.
[2] SimEvents. Model and simulate diskrete-event system. http://www.mathworks.com/products/simevents/
[3] Stateflow. Model and simulate decision logic using state machines and flow. http://www.mathworks.com/products/stateflow/
[4] Stephen S. Fratini. Analysis of a dynamic priority queue, Communications in Statistics. *Stochastic Models* **6(3)** 415-444
[5] Apachidi X. N., Katsman Yu. Ya. Development of a queuing system with dynamic priorities. *Key Engineering Materials* **685** 934-938

[6] Law A. M. 2008 How to build valid and credible simulation models. *Proc. of the Winter Simulation Conf. Miami (USA)* pp 39-47
[7] Sargent R. G. 2008 Verification and validation of simulation models. *Proc. of the Winter Simulation Conf. Miami (USA)* pp 39-47