periodica polytechnica

Transportation Engineering
38/2 (2010) 113–118
doi: 10.3311/pp.tr.2010-2.09
web: http://www.pp.bme.hu/tr

RESEARCH ARTICLE

The control of fleet management systems' server model

Szilárd Aradi / *Tamás* Bécsi

Received 2009-10-07

Abstract

Our article deals with the load controlling of server systems which can be represented as M/M/1 queuing models. It introduces the results of a service structure's state space based control, which has also been realized in practice, the system model, and the control which guarantees the availability of the system.

Keywords

server control · queuing model

Acknowledgement

This work is connected to the scientific program of the " Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" project. This project is supported by the New Hungary Development Plan (Project ID: TÁMOP-4.2.1/B-09/1/KMR-2010-0002).

Szilárd Aradi

Department of Control and Transport Automation, BME, H-1111 Budapest Bertalan L. u. 2., Hungary

e-mail: szilard.aradi@auto.bme.hu

Tamás Bécsi

Department of Control and Transport Automation, BME, H-1111 Budapest Bertalan L. u. 2., Hungary

e-mail: becsi.tamas@mail.bme.hu

1 Goals

The aim of the article is to examine the question of the availability of a communication server [3,4] which is part of the fleet management system, and to answer questions regarding the control of the system. Systems of this kind are handled as the combination of an M/M/1 queuing [2, 5] and a load model. The queuing model operates by modelling the length of the queue between the incoming internal and external requests, and their service i.e. it determines how many unsolved tasks the software realizing the system still has. The load model, on the other hand, determines how large the amount of the system's resources has to be in order to be able to execute these tasks [1, 7]. In case of a server application, the load means the utilization of hardware-based resources, which can be processor load, memory usage, I/O operations of the computer storage, and, in a given case, use of network resources.

Thus, a system based on a theory of this kind has to satisfy two basically opposite conditions:

1 It has to provide service to incoming requests as fast as possible;

2 No problem should occur while operating, which could mean a real danger if the resources were overloaded.

So, the expected load has to be estimated during the construction of the system, which means that during the realization, a classical measuring problem has to be realized in which the number of the server's clients, the cardinality of messages sent and to be received by them, and the resource need of these "services" has to be determined as accurately as possible. This planning results, like in the case of every other technological problem, in an oversize, which also takes costs (investment and maintenance) into consideration. As a result of which as well as by proper operation but also in case of small demand growths with the application of sufficient reserve resources, the system can operate successfully. Which means that it can serve every demand in a minimal amount of time.

After this planning, it can be a logical presupposition that the system should serve every request as fast as possible. However, should any incident of vis major nature occur e.g. an intended

shutdown, the system had to face such a large number of requests that their execution could result in an overload, or even the breakdown of the system.

A system, which operates without control, cannot respond to a challenge of this sort, so the installation of a service model is necessary which can take control of the load of the resources. This can result in a longer service time but it also improves the availability and ensures the stabile operation of the system.

2 Model build-up

To make the design of the system's service model possible, the internal dynamics of the system have to be mapped first. It seems to be effective to suppose that the system is dynamic, time-invariant, deterministic and discrete time.

It is dynamic, as the queuing model does not contain only present-time information but past-time as well. It is time-invariant i.e. it can be supposed that the answers of the system are independent of time. Discrete time i.e. the examination of the system's states are not done continuously but according to operating cycles. It is deterministic i.e. it can be supposed that the operation of the system is not influenced by any random variable.

The system can of course lose some features while being constructed but it is definitely practical to formulate these presuppositions.

During the construction of the model, every parameter has to be set which together can influence the operation of the system. It is important to know from the point of view of the resources how many clients the system serves at the same time. The client number (x_{klsz}) is a simple integrating state variable whose value in step (n+1) depends on the client number of the previous cycle (n), and on the number of incoming and outgoing clients (d_{klb}incoming clients; d_{klk}outgoing clients) during the cycle:

$$x_{klsz}(n+1) = x_{klsz}(n) - d_{klk}(n) + d_{klb}(n) \tag{1}$$

The incoming requests, be they requests from the side of the client or from that of the server depend on the actual client number but it is practical to define these as external inputs as the density of occurrence of these requests can vary in time. The queuing model of the requests waiting for service in the given cycle can be written up from this (x_{sh}), which can be written up from the value taken up by it in the previous cycle, and the incoming (d_{ib}) and outgoing (x_{ksz}) requests:

$$x_{sh}(n+1) = x_{sh}(n) - x_{ksz}(n) + d_{ib}(n) \tag{2}$$

The number of requests which are to be served in the next cycle are interpreted in the given cycle as the system's input i.e. the state variable depends on the number of requests which are to be served (u_{ksz}):

$$x_{ksz}(n+1) = u_{ksz}(n) \tag{3}$$

In order for all the information to be available, the state variable of the resource load has to be determined as well. What the resources regards, the demand for system memory can be regarded as an insignificant variable because this resource, as it is also shown by the examination of the sample system provided with the suitable data structure, becomes practically inexhaustible. The network and the computer storage I/O operations, as the CPU is waiting for the realization of these, can be converted unambiguously into CPU load values. Every universal built-in algorithm i.e. searches, maintenance of lists, control of state machines, etc. occupy the CPU's time as well. According to these considerations, it is suitable to model the resource need with CPU time or with CPU load [%]. This value depends on more parameters:

- the load independent from other variables (t_{sb} Stand by [%]);
- the universal functions applied to certain clients, which is a function of the client number (x_{klsz} multiplying c_{ekp} passive clients [%/client]);
- the CPU load created during the login and logout of the clients (($d_{klk+}d_{klb}$) multiplying c_{klkb} clients in/out load [%/client]);
- the load of the manifestation of requests (d_{ib} multiplying c_{ib} request receiving load [%/client]);
- and the load of requests served in the given cycle (x_{ksz} [db] multiplying c_{ksz} service load [%/client]) :

$$\begin{aligned} x_{pt}(n+1) = t_{sb} + c_{ekp}x_{klsz}(n) + c_{ksz}x_{ksz}(n) + \\ c_{klkb}(d_{klk}(n) + d_{klb}(n)) + c_{ib}d_{ib}(n) \end{aligned} \tag{4}$$

This last state variable is taken for the model's output i.e.:

$$y(n) = x_{pt}(n) \tag{5}$$

The Eqs. (1)-(5) define a linear system with the state space representation of these Linear equations as follows:

$$\begin{aligned} x &= \begin{pmatrix} x_{proc} \\ x_{ksz} \\ x_{sh} \\ x_{klsz} \end{pmatrix}, u = \begin{pmatrix} u_{ksz} \\ d_{ib} \\ d_{klk} \\ d_{klb} \end{pmatrix}, \\ A &= \begin{pmatrix} 0 & c_{ksz} & 0 & c_{ekp} \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ B &= \begin{pmatrix} 0 & c_{ib} & c_{klkb} & c_{klkb} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \\ D &= \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix} \\ C &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{6}$$

$$\begin{aligned} x(n+1) &= Ax(n) + Bu(n) \\ y(n) &= Cx(n) + Du(n) \end{aligned} \tag{7}$$

2.1 Linear model with certain restrictions

2.1.1 Restrictions

The value which is represented in the linear system as client number can under the effect of appropriate inputs take up negative values as well. This is not possible in a real system on the one hand, while on the other it would lead to negative CPU load in the dynamics, which is also not very realistic. Thus, client number has a minimum:

$$x_{klsz}^{real} = \max\left(x_{klsz}^{\text{theoretical}}; 0\right) \tag{8}$$

The same goes for queue length, which also cannot take up negative values:

$$x_{sh}^{real} = \max\left(x_{sh}^{\text{theoretical}}; 0\right) \tag{9}$$

Besides, the request number which is actually to be served in the given cycle does not only equal the theoretical service number i.e. that which is defined as input, but it is also limited by the number of servable entities in the queue:

$$x_{ksz}^{real} = \min\left(x_{ksz}^{\text{theoretical}}; x_{sh}^{real}\right) \tag{10}$$

2.1.2 Stochastic variables

The discrete time linear system which is taken up according to Eq. (6) has four inputs, d_{klk}, d_{klb},d_{ib} and u_{ksz}. However, it is only the last one of these, the service number, which can be used for controlling purposes, as the system has no impact upon the login or logout of the clients. Therefore, the effect of these inputs has to be interpreted as disturbance.

The CPU load, if we leave some other features out of consideration, is mostly made up of the load of served requests. The service time of one request (c_{ksz}) however depends on many external factors e.g. response time of the database and the occupation of other hardware resources. However, this system parameter thus cannot be considered to be constant, it is only its theoretical minimum that can be defined.

3 Controllability analysis

3.1 Control goals

The planning of the system's control has to be performed mainly by taking the above mentioned two basic aspects into consideration, so that the condition of keeping the positive value of input u_{ksz} has to be continuously satisfied.

The CPU load clearly determines the length of the server's processing cycle. Should the system serve too many requests in a cycle, the time of the given cycle becomes too long, and, because of the service, the system cannot take care of other tasks e.g. login or communication of the clients, or the processing of requests. Thus, it is practical to determine a cycle time in which in the given cycle every task can be executed in the case of every parameter and input corresponding to regular operation. This is a simple measuring task, and the hardware need of the system should be determined according to this parameter. If the system is overladen, the amount of requests that are to be served in one cycle needs to be determined so, that the CPU load should not be higher than the value of the determined cycle time. The same goes for the reverse case, should the CPU load be much lower than the cycle time, the system load is not optimal, the queue of pending requests does not decrease in the highest possible degree. However, exceedance of the cycle time is to a small extent and on a single occasion is allowed. Different surveys touch upon various fields of control theory while controlling systems of this kind, so both solutions of PI-type [6] and LPV-based ones can be found [8],[9].

3.2 Controllability, observability

The states and inputs of the system are measurable, as it is an information technological application, so the application of an observer is not necessary. As the system can only be controlled with input u_{ksz}, controllability analysis has to be performed on this input. The rank of the controllability matrix that is projected to the system's input u_{ksz} is three, which is less than the dimension of the system. As the system is not complete state controllable, the stability of the system cannot be ensured with input u_{ksz} under the present modelling paradigm. The unreachable state is the client number (x_{klsz}), and therefore, the CPU load is not completely controllable either.

3.3 Planning controllability

It is practical to leave the client number, as unreachable state, and thus its effect on CPU load out of the system that is to be taken into consideration during the construction of the controller, or to take it for a disturbance which appears at the output of the system. This can be easily kept in hand if we know the nature of the disturbance. In the present case, there is even something more that can be achieved, as we can determine this disturbing sign, which has an effect on the CPU load, with approximate estimating.

So, the state space of the system is defined by the following matrices:

$$\begin{gathered}\tilde{x} = \begin{pmatrix} x_{proc} \\ x_{ksz} \\ x_{sh} \end{pmatrix}, \tilde{u} = (u_{ksz}), \\ \tilde{A} = \begin{pmatrix} 0 & c_{ksz} & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 1 \end{pmatrix} \quad \tilde{B} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \\ \tilde{C} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \qquad \tilde{D} = (0)\end{gathered} \tag{11}$$

3.3.1 Control planning

The rank of the (11) system's controllability matrix is 3, which is equal to the system's dimension, so, the system becomes controllable. The system's poles, the own values of matrix $\tilde{A}$, (12) are not stable, so it is necessary to use state feedback in order to stabilize them.

$$\overline{P}(\tilde{A}) = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \tag{12}$$

A solution to the pole-replacing task is given by Ackermann's formula:

$$K = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} M_c^{-1}(\tilde{A}, \tilde{B})\phi_c(\tilde{A}) \tag{13}$$

$$K = \Bigg(\frac{\alpha_1\alpha_2\alpha_3}{c_{ksz}}, 1 - \alpha_1 - \alpha_3 - \alpha_2, -1 + \alpha_1 + \alpha_3 + \alpha_2 - \alpha_1\alpha_3 - \alpha_1\alpha_2 - \alpha_2\alpha_3 + \alpha_1\alpha_2\alpha_3\Bigg) \tag{14}$$

If we leave the feedback of x_{ksz} out of consideration, the second element of vector K can be 0. In this case, the following rule has to be observed while setting the poles:

$$\alpha_2 = 1 - (\alpha_1 + \alpha_3) \tag{15}$$

In this case, K's value changes as follows:

$$K = \Bigg(-\frac{\alpha_1\left(-1 + \alpha_1 + \alpha_3\right)\alpha_3}{c_{ksz}}, 0, 2\alpha_1\alpha_3 - \alpha_1 + \alpha_1^2 - \alpha_3 + \alpha_3^2 - \alpha_1^2\alpha_3 - \alpha_1\alpha_3^2\Bigg) \tag{16}$$

By choosing an appropriate pole triplet:

$$\alpha_1 = \alpha_3 = 0.2; \alpha_2 = 0.6 \tag{17}$$

The vector of the state feedback will be as a function of service response time (c_{ksz}) as follows:

$$K = \begin{pmatrix} \frac{0.024}{c_{ksz}} & 0 & \text{-0.256} \end{pmatrix} \tag{18}$$

3.3.2 Tracking control

The controlling mentioned in section 3.3.1 can adequately control the system into a stabile state if it functions properly when, according to system scaling, less requests appear than the system constantly can serve. However, should queue length become too high because of a rapid growth of request density or that of specific service time, the controller would "push" the queue length state variable over the 0 value, besides, the cycle time of the system would grow as well. The reason why the first problem significant is, is because the system – because of its positive feature – would get out of the linear domain this way and, therefore, stabile controlling cannot be ensured with it. The second problem, the lengthening of the cycle time can hinder the execution of the system's other tasks and by doing this, it blocks proper functioning. To prevent an overload of this sort, it is practical to use an alternative controlling loop, which puts emphasis other aspects. As it can be supposed that because of the measuring the above mentioned anomalies are only temporary i.e. the service gradually catches up with the appearance of requests, so the queue of pending requests decreases after the solution of the problem to normal value, it is practical to design a control which can ensure the availability of the system in the period of overload. In this second control, the cycle time of the system has to be kept in hand. As the product of queue length and service time is in this case higher than the expected cycle time, it is necessary to construct a fixed value control which can ensure this. This can be done by installing an integral term and by using output feedback. A further benefit of integral control is that as the output disturbance appears on the output, it can also take care of its suppression.

As there is no task in this model for the stabilization of the queue length variable and it can be excluded from the point of view of control, there are only two state variables in the new model: x_{proc}, and x_{ksz}. The integral term, however, has to be integrated into the system model:

$$x_i(n+1) = x_i(n) + y_i(n) = x_i(n) + Cx(n) \tag{19}$$

The state matrices take up after the introduction of expanded state variable $\hat{x} = \begin{pmatrix} x_{proc} & x_{ksz} & x_i \end{pmatrix}^T$ the following form:

$$\hat{x} = \begin{pmatrix} x_{proc} \\ x_{ksz} \\ x_i \end{pmatrix}, \hat{u} = (u_{ksz}), \quad \hat{A} = \begin{pmatrix} 0 & c_{ksz} & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad \hat{B} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \hat{C} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \quad \hat{D} = (0) \tag{20}$$

The poles of this new system are also not stabile, so construction of control is necessary.

The new system's state feedback vector can be derived again from Ackermann's formula (13), the new feedback vector is:

$$\hat{K} = \begin{pmatrix} \frac{1-\alpha_1-\alpha_3-\alpha_2+\alpha_1\alpha_3+\alpha_1\alpha_2+\alpha_2\alpha_3}{c_{ksz}} \\ 1 - \alpha_1 - \alpha_3 - \alpha_2 \\ \frac{1-\alpha_1-\alpha_3-\alpha_2+\alpha_1\alpha_3+\alpha_1\alpha_2+\alpha_2\alpha_3-\alpha_1\alpha_2\alpha_3}{c_{ksz}} \end{pmatrix}^T \tag{21}$$

Here, the feedback of x_{ksz} can again be excluded by the observation of the following equation.

$$\alpha_2 = 1 - (\alpha_1 + \alpha_3) \tag{22}$$

So, the simplified vector **K** is the following:

$$\hat{K} = \Bigg(-\frac{\alpha_1\alpha_3 - \alpha_1 + \alpha_1^2 - \alpha_3 + \alpha_3^2}{c_{ksz}} 0 \quad \frac{-2\alpha_1\alpha_3 + \alpha_1 - \alpha_1^2 + \alpha_3 - \alpha_3^2 + \alpha_1^2\alpha_3 + \alpha_1\alpha_3^2}{c_{ksz}}\Bigg) \tag{23}$$

By choosing an appropriate stabile pole triplet:

$$\alpha_1 = \alpha_3 = 0.8; \alpha_2 = -0.6 \tag{24}$$

Vector **K** of the integral control becomes the following:

$$\hat{K} = \begin{pmatrix} -\frac{0.32}{c_{ksz}} & 0 & \frac{0.064}{c_{ksz}} \end{pmatrix} \tag{25}$$

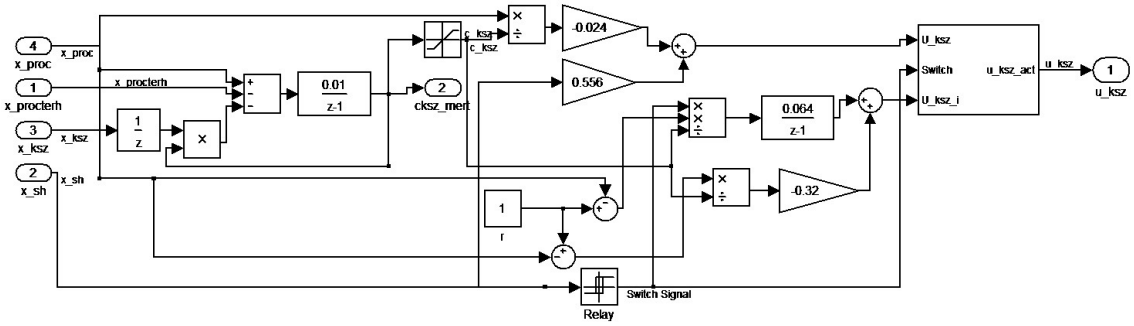

Fig. 1. Control synthesis

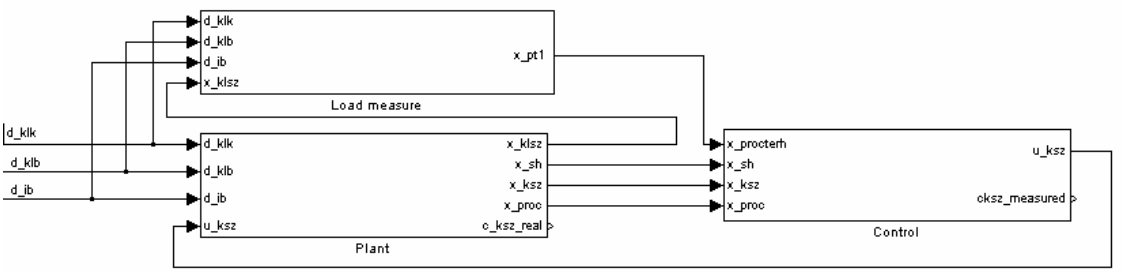

Fig. 2. The controlled system

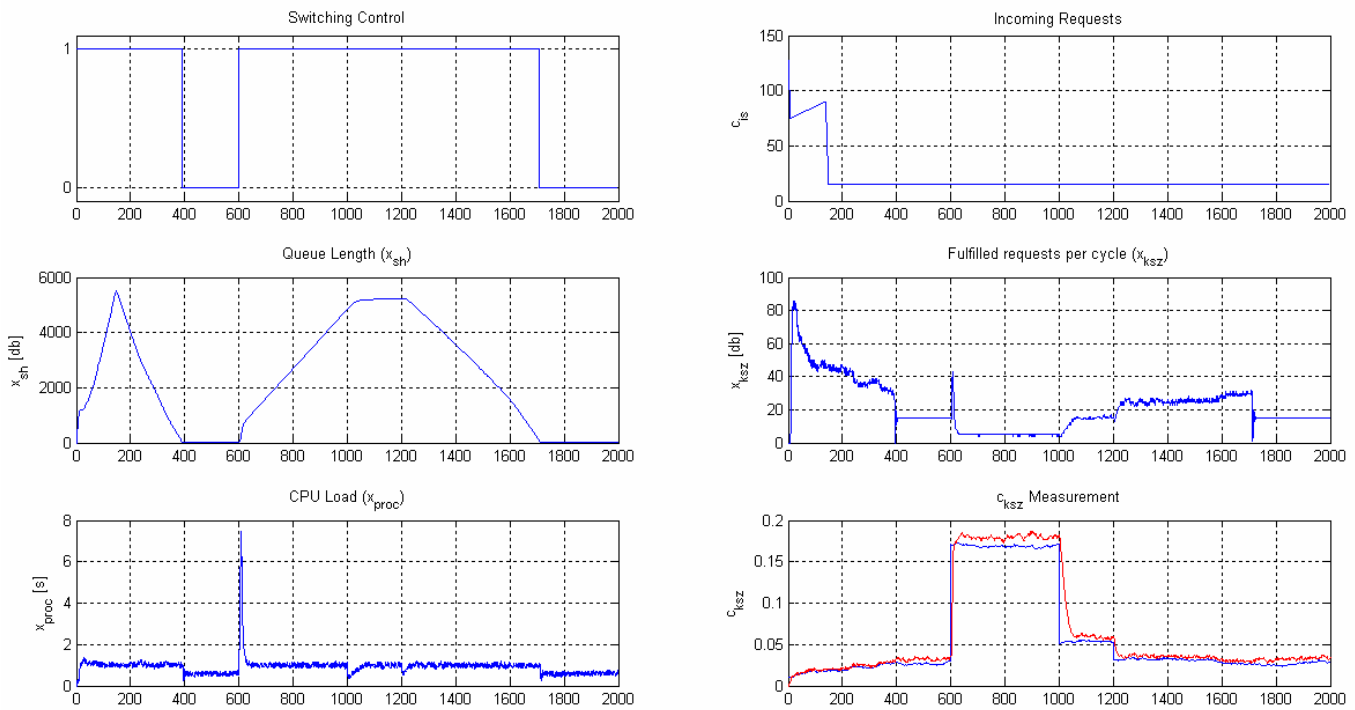

Fig. 3. Simulation Results

3.3.3 Installing the controls

In the two previous sections, two controls of different philosophical background have been elaborated for two different operation conditions. If the system operates properly, stabilization is easy, while, in the case of overload, a fixed value control has been constructed in order to limit cycle time.

The situation becomes even more complicated because an inconstant parameter, which by its nature has an influence on cycle time, can be found in the system, whose development has to be taken into consideration. Thus, in order to achieve the full synthesis of control, the following two basic tasks have to be performed: a subsystem is needed which is able to provide an adequate estimation or measuring about service time parameter c_{ksz}, and a block which realizes a shift between control architectures.

Determining parameter c_{ksz} – as it will be shown later – does not demand full precision but it is also not negligible. For this, the measurable actual CPU load and the x_{ksz} number of the services can be used. Moving window regression would seem to be the simplest solution but would occur too frequently during

the operation of the system that parameter x_{ksz}, which is used to determine CPU load, does not occur deviated enough, so putting a curve on it would lead to an inaccurate result. The other possibility is model-based the estimation of CPU load. By using the original system's state Eqs. (6) the – from service independent – CPU load becomes the following:

$$d_{proc}(n+1) = c_{klkb}(d_{klk}(n)+ d_{klb}(n)) + c_{ib}d_{ib}(n) + c_{ekp}x_{klsz}(n) \quad (26)$$

The parameters in the equation (c_{klkb}, c_{ib}, c_{ekp}) can be considered to be nearly constant. According to the system's characteristics, it is true that:

$$x_{proc}(n+1) = d_{proc}(n+1) + c_{ksz}^{real}x_{ksz}(n) \quad (27)$$

So, the subsystem has to be built up so, that it has to be true for parameter c_{ksz}, which is a variable in it, that:

$$x_{proc}(n+1) - d_{proc}(n+1) - c_{ksz}x_{ksz}(n) \to 0 \quad (28)$$

For the minimization of difference, it is suitable to use approaching of c_{ksz} with slow integrator, and creating the difference function by feeding back the output. Although, this is not the most exact solution possible yet it approaches parameter c_{ksz}, which changes not too frequently or with a slope of relatively small value, adequately.

In order for us to be able to choose between the two control theory, the queue length parameter has to be examined. It is important that the shift between the controls should not be bound to a concrete value (barrier of overload) but it is necessary to introduce hysteresis to a certain degree. Fig. 1 shows the schema of built-in control.

Fig. 2 shows the block schema of the realized system. The three main blocks are well separated. The first one shows the system which has to be controlled and which has been taken up restrictedly, the second one the control that has been outlined above, and the third one the measuring of the output noise.

3.4 Results

The article outlines the construction of a "switching control" structure that performs the control of a server architecture represented by a queue growing and a service model according to the following aspects: The primary aspect of construction is to ensure the system's availability by controlling cycle time with service number per cycles, the second aspect is the shortening of the queue, the serving of incoming requests.

For the examination of control, a theoretical load model has been used that includes more situations. The model's operation results can be seen on Fig. 3. The development of c_{ksz} introduces two suddenly rising loads into the system. The density of the requests' arrival at the starting of the system presupposes that the clients waiting for connection have collected more requests that, as a result of this, arrive in a large quantity at the same time. It can be seen clearly that at the lengthening of the queue, when the product of the number of servable requests and of the service time exceeds the requested cycle time, the control shifts to value-reserving control, which is able to ensure appropriate cycle time, and when the system gets load from proper functioning, and the queue length has shortened, it uses the "normal" stabilizing control.

The figure shows clearly that because of the constant change of parameter c_{ksz}, and other state characteristics, service is not constant in the particular service stages but it ensures appropriate CPU load by adapting itself to the new conditions.

Finally, it is worth taking a look at CPU load, and so examining cycle time. It can be seen clearly that, under proper functioning, the system does not reach the required 1s cycle time, which corresponds to the measuring. The system responds well to overload caused by incoming requests but, logically, the growth of service time in the given cycle is still very rapid but then we get a quickly descending CPU load.

References

1 **Abdelzaher T, Stankovic J. A, Chenyang Lu, Zhang R, Ying Lu**, *Feedback Performance Control in Software Services*, IEEE Control Systems Magazine **23** (2003), 74-90, DOI 10.1109/MCS.2003.1200252.

2 **Abdelzaher T, Ying Lu, Ronghua Zhang, Henriksson D**, *Practical application of control theory to Web services*, American Control Conference, 2004. Proceedings of the 2004, 2004, pp. 1992-1997.

3 **Aradi Sz, Bécsi T**, *Flottamenedzsment rendszerek adatátviteli módszerei*, A jövő járműve **3** (2008), no. (3-4), 39-43.

4 **Bécsi T, Aradi Sz**, *Reliability of Data Transfer and Handling in Railway Telemonitoring Systems*, Proc. of Symposium FORMS/FORMAT 2008, 2008, pp. 185-192.

5 **Hellerstein J L, Yixin Diao, Parekh S**, *A first-principles approach to constructing transfer functions for admission control in computing systems*, Decision and Control, 2002, Proceedings of the 41st IEEE Conference, 2002, pp. 2906-2912, DOI 10.1109/CDC.2002.1184291.

6 **Lui S, Xue L, Ying L, Abdelzaher T**, *Queueing model based network server performance control*, Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE, 2002, pp. 81-90, DOI 10.1109/REAL.2002.1181564, (to appear in print).

7 **Robertsson A, Wittenmark B, Kihl M, Andersson M**, *Design and evaluation of load control in Web server systems*, American Control Conference, 2004. Proceedings of the 2004, 2004, pp. 1980-1985.

8 **Wubi Qin, Qian Wang**, *Modeling and Control Design for Performance Management of Web Servers Via an LPV Approach*, Control Systems Technology, IEEE Transactions on **15** (2007), no. 2, 259-275.

9 **Wubi Q, Qian W, Sivasubramiam A**, *An α-Stable Model-Based Linear-Parameter-Varying Control for Managing Server Performance Under Self-Similar Workloads*, Control Systems Technology, IEEE Transactions on **17** (2009), no. 1, 123-134.