Control theoretic modelling and design of admission control mechanisms for server systems*

Maria Kihl[a], Anders Robertsson[b], and Björn Wittenmark[b]
[a]Department of Communication Systems, [b]Department of Automatic Control
Lund University, BOX 118, 221 00 Lund, Sweden
email: maria@telecom.lth.se, fax: +46 46 14 58 23, tel.no: +46 46 222 9010

Abstract. The admission control mechanism is an important part of many communication systems. In this paper we investigate load control mechanisms for server systems, that is systems that may be modelled as queueing systems. We show how control theory can be used when designing controllers for a *G/G/1*-system. We design a PI-controller for the system and compare the steady-state and transient behavior of this controller with the behavior of a static controller.

1. Introduction

One problem with all server systems, for example web servers or application servers in 3G networks, is that they are sensitive to overload. Therefore, admission control mechanisms can be implemented in the systems. The mechanism can either be static or dynamic. A static mechanism admits a predefined rate of calls whereas a dynamic mechanism contains a controller that, with periodic time intervals, calculates a new admission rate depending on some control objective.

The research concerning admission control has shown that the problem of optimally controlling the arrivals at a server system is a difficult task. The main problem comes from the fact that server systems usually are analyzed with queueing theory. However, there are no queueing theoretic methods that can be used when developing and designing controllers for the systems. Another solution is, therefore, to use control theory. Control theory has since long been used to analyze different types of automatic control systems. One well-known controller in automatic control is the PID-controller, which enables a stable control for many types of system (see, for example, [9]). The PID-controller uses three actions: one proportional, one integrating, and one derivative.

Very few papers have investigated admission control mechanisms for server systems with control theoretic methods. In [1] and [2] a web server was modelled as a static gain to find controller parameters for a PI-controller. A scheduling algorithm for an Apache web server was designed using system identification methods and linear control theory in [7]. In [4] a PI-controller is used in an admission control mechanism for a web server. However, no analysis is presented on how to design the controller parameters.

In [5] and [8], we analyzed queue length controllers for M/G/1-system. We developed a nonlinear fluid flow model and used this model when designing a PI-controller for the system. We demonstrated that linear models of this system are insufficient,

*. The work in this paper has partially been supported by the Swedish Research Council through the Multi Project Grant 621-2001-3020 and contract 621-2001-3053.

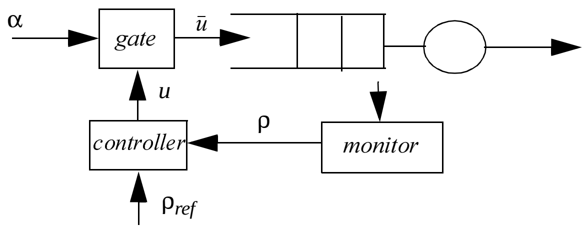

Figure 1. Investigated system.

since the nonlinearities in the gate and queue introduce system dynamics that must be considered in the design process.

In this paper we instead analyze load control mechanisms. In [6], we developed and validated a control theoretic model of a G/G/1-system that can be used for the design of load control mechanisms. In [3] we show that the model is valid for an Apache web server. In this paper, we design and analyse a PI-controller.

2. System model

The system model is shown in Fig. 1. We assume that the system may be modelled as a G/G/1-system with an admission control mechanism. The admission control mechanism consists of three parts: a *gate*, a *controller*, and a *monitor*. Continuous control is not possible in computer systems. Instead, time is divided into control intervals of length h seconds. Time interval [kh-h, kh] is denoted interval kh.

The monitor measures the *control variable*, in this case the average server utilization during interval kh, $\rho(kh)$. At the end of interval kh, the controller calculates the desired admittance rate for interval $kh+h$, denoted $u(kh+h)$, from the measured average server utilization during interval kh, and the reference value, ρ_{ref}. The objective is to keep the server utilization as close as possible to the reference value. The gate rejects those requests that cannot be admitted. The variable representing the number of arrivals during control interval kh is denoted $\alpha(kh)$. Since the admittance rate may never be larger than the arrival rate, the actual admittance rate, $\bar{u}=\min[u, \alpha]$.

The gate uses a token bucket algorithm to reject those requests that cannot be admitted. Rejected requests are assumed to leave the system without retrials. An arriving request is only admitted if there is an available token. New tokens are generated at a rate of $u(kh)$ tokens per second during control interval kh.

3. Control theoretic model

We use the discrete-time control theoretic model shown in Fig. 2. This model has been validated in [6] for the single server queue in Section 2. The model is a flow or liquid model in discrete-time. The model is an averaging model in the sense that we are not considering the specific timing of different events, arrivals, or departures from the queue.

There are two stochastic traffic generators in the model. The *arrival generator* feeds the system with new requests. The number of new requests during interval kh is

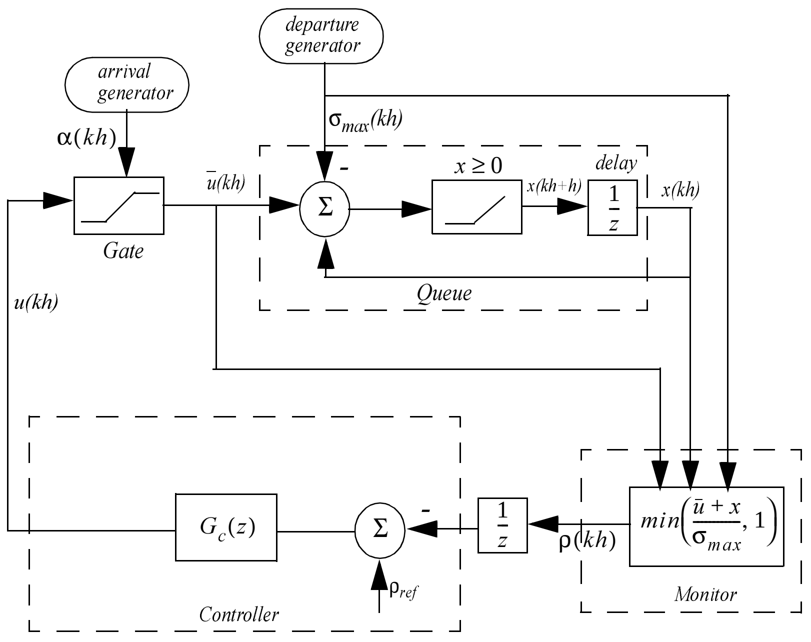

Figure 2. A control theoretic model of a G/G/1-system with admission control.

denoted α*(kh)*. α*(kh)* is an integrated stochastic process over one sampling period with a distribution obtained from the underlying interarrival time distribution. The *departure generator* decides the *maximum* number of departures during interval *kh*, denoted σ_{max}*(kh)*. σ_{max}*(kh)* is also a stochastic process with a distribution given by the underlying service time distribution.

The *gate* is constructed as a saturation block that limits the number of admitted requests during interval *kh*, $\bar{u}(kh)$, to be zero when $u(kh) < 0$, *u(kh)* when $0 \leq u(kh) \leq \alpha(kh)$, and α*(kh)* when $u(kh) > \alpha(kh)$.

The *queue* is represented by its state *x(kh)*, which corresponds to the number of requests in the system at the end of interval *kh*. The difference equation for the queue is given by $x(kh+h) = f(x(kh) + \bar{u}(kh) - \sigma_{max}(kh))$

where the limit function, *f(w)*, equals zero if *w*<0 and *w* otherwise. The limit function assures that $x(kh+h) \geq 0$. When the limit function is disregarded then the queue is a discrete-time integrator.

The *monitor* must estimate the server utilization since this is not directly measurable in the model. The server utilization during interval *kh*, ρ*(kh)*, is estimated as

$$\rho(kh) = min\left(\frac{\bar{u}(kh) + x(kh)}{\sigma_{max}(kh)}, 1\right)$$

The objective of the *controller* is to minimize the difference between the server utilization during interval *kh*, ρ*(kh)*, and the reference value, ρ_{ref}. The control law is given by the transfer function, G$_c$*(z)*.

4. Controller design

The system we investigated had an average service time of 0.02 seconds and the reference load, ρ_{ref}, was set to 0.8. We will use linear control design methods for finding parameters for a PI-controller. This means that we during the design consider a deterministic system with no active saturations. However, the real queueing system will for instance only allow positive queue lengths.

4.1 Static controller

We used a static controller as a benchmark controller. A static controller uses a fixed acceptance rate, u_{fix}, that is set so that the average value of the control variable should be equal to the reference value. u_{fix} is in this case equal to 40 jobs per second.

4.2 PI-controller

The PI-controller is a well-known controller in automatic control. It uses two actions: one proportional and one integrating. The control law for the PI-controller expressed in z-transform is given by

$$G_c(z) = K\left(1 + \frac{1}{T_i} \cdot \frac{h}{z-1}\right)$$

where the gain K and the integral time T_i are the controller parameters that are set so that the controlled system behaves as desired. The characteristic polynomial for the linear closed loop system will be

$$z \cdot \left(z^2 + \frac{K - 2\sigma}{\sigma} z + \frac{-KT_i + Kh + \sigma T_i}{\sigma T_i}\right) \qquad \text{(EQ 1)}$$

where σ is the average value of σ_{max} and the pole at z=0 is cancelled in the transfer function from the input (the load reference) to the desired output (the load). Assume that the desired characteristic equation is

$$z(z^2 + a_1 z + a_2) = 0$$

The values of the controller parameters that gives this are

$$K = 2\sigma + a_1\sigma \qquad T_i = h \cdot \frac{2 + a_1}{1 + a_1 + a_2}$$

The controller parameters K and T_i influence the closed loop response for the system and need to be determined with respect to stability and robustness. The behavior of the PI-controller becomes better when the sampling period is short (should match desired dynamics). Therefore, for these investigations we used a sampling period of 0.2 seconds (h=0.2). This means that $\sigma = 10$, since σ is the average maximum number of departured jobs during a control interval. Choosing $\{K, T_i\}=\{12, 0.6\}$ the roots of the characteristic polynomial in eq. (1) will be 0.4 ± 0.2. This set of controller parameters can, therefore, be seen as a “good” choice.

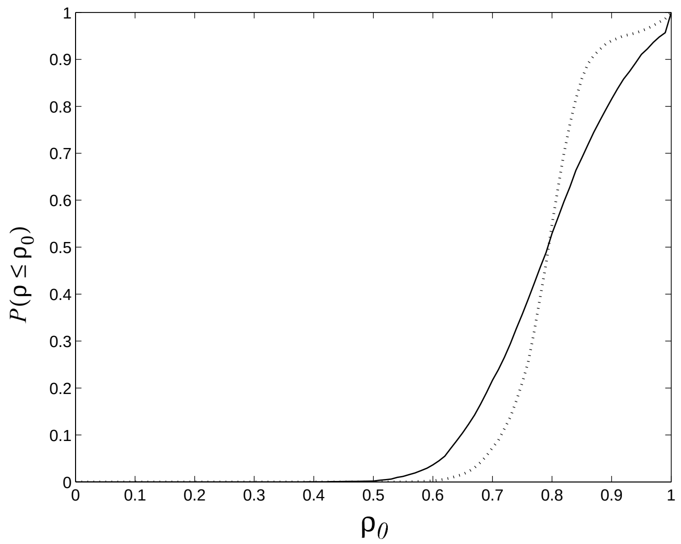

Figure 3. Distribution function for an M/M/1-system,
solid line: static controller , dotted line: PI-controller.

5. Numerical investigations

The numerical investigations contain a comparison of the controllers described in the previous section. The queueing model was represented by a discrete-event simulation program implemented in C, and the control theoretic model was implemented with the Matlab Simulink package. During all investigations, the reference load was set to 0.8, and the average arrival rate was 150 jobs per second. We investigated an M/M/1-system, with average service time 0.02 seconds. Note that the controller design is independent of the type of arrival process and the service time distribution, since the system dynamics only depend on the average service time.

In Fig. 3, we show the *steady-state distribution* of the server utilization, by plotting the estimated distribution function, i.e. $P(\rho \le \rho_0)$ where $0 \le \rho_0 \le 1$. The distribution function was estimated from 5000 measurements of the server utilization for a specific parameter setting. The optimal distribution function is zero for $0 \le \rho \le 0.8$ and one for $0.8 \le \rho \le 1$. Each measurement is the average server utilization during one second. The distribution function shows how well the controller meets the first control objective. As can be seen, the systems with a PI-controller behave better than the system with a static controller. This phenomenon is due to that this controller can adapt to the stochastic variations in the system. This behavior requires a short sampling period. With a longer sampling period, for example one second, the PI-controller behave as the static controller.

The *step responses* for the load controlled M/M/1-system were also investigated. The step responses show the transient behavior of the controllers. The fastest controller is of course the static controller, since it already from start is set to an accurate admittance rate. However, the PI-controller has found a correct admittance rate only after a few seconds, which means that it can be regarded as fast enough.

Finally, we have investigated the *robustness* for the system. A good controller should maintain a good performance even when the system parameters change, that is the controller should be robust to modelling errors. In a real system, it is likely that the average service time will change slowly with time, for example due to changes in the user behavior. The results showed that the PI-controller was very robust to changes in the average service time. The static controller is, of course, dependent on a correct service time, which means that it cannot operate properly when the service times change.

6. Conclusions

In this paper, we have designed load control mechanisms for a G/G/1-system with control theoretic methods. We have designed a PI-controller. We have shown that, when considering transient and stationary behavior, the PI-controller behave well. One conclusion of this paper is that it is possible to use control theoretic methods when designing admission control mechanisms for server systems. The designs have been verified with simulations for discrete-event systems based on queuing theory.

References

[1] T.F. Abdelzaher and C. Lu, “Modeling and performance control of Internet servers”, Proc. of the 39th IEEE Conference on Decision and Control, 2000, pp 2234-2239.

[2] T.F. Abdelzaher, K.G. Shin and N. Bhatti, “Performance guarantees for web server end-systems: a control theoretic approach”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 1, Jan 2002, pp 80-96.

[3] M. Andersson, M. Kihl, and A. Robertsson, “Modelling and design of admission control mechanisms for web servers using non-linear control theory”, Proc. of SPIE ITCom, 2003.

[4] P. Bhoj, S. Ramanathan, and S. Singhal, “Web2K: Bringing QoS to web servers”, HP Labs Technical report, HPL-2000-61, 2000.

[5] M. Kihl, A. Robertsson, and B. Wittenmark, “Analysis of admission control mechanisms using non-linear control theory”, Proc. of IEEE International Symposium on Computer Communcations, 2003.

[6] M. Kihl, A. Robertsson, and B. Wittenmark, “Performance Modelling and Control of Server Systems using Non-linear Control Theory”, Proc. of 18th International Teletraffic Congress, 2003.

[7] C. Lu, T.F. Abdelzaher, J.A. Stankovic and S.H. Son, “A feedback control approach for guaranteeing relative delays in web servers”, Proc. of the 7th IEEE Real-Time Technology and Applications Symposium, 2001, pp 51-62.

[8] A. Robertsson, B. Wittenmark, and M. Kihl, “Analysis and design of admission control in web-server systems”, American Control Conference, 2003.

[9] K.J. Åström and B. Wittenmark, *Computer-controlled systems, theory and design*, Prentice Hall International Editions, 3rd Edition, 1997.