Control-theoretic Analysis of Admission Control Mechanisms for Web Server Systems

Kihl, Maria; Robertsson, Anders; Andersson, Mikael; Wittenmark, Björn

Published in:
World Wide Web

DOI:
10.1007/s11280-007-0030-0

2008

World Wide Web (2008) 11:93–116
DOI 10.1007/s11280-007-0030-0

Control-theoretic Analysis of Admission Control Mechanisms for Web Server Systems

M. Kihl · A. Robertsson · M. Andersson · B. Wittenmark

Received: 1 November 2005 / Revised: 10 February 2007 /
Accepted: 11 April 2007 / Published online: 15 August 2007

Abstract Web sites are exposed to high rates of incoming requests. The servers may become overloaded during temporary traffic peaks when more requests arrive than the server is designed for. An admission control mechanism rejects some requests whenever the arriving traffic is too high and thereby maintains an acceptable load in the system. This paper presents how admission control mechanisms can be designed with a combination of queueing theory and control theory. In this paper we model an Apache web server as a GI/G/1-system and then design a PI-controller, commonly used in automatic control, for the server. The controller has been implemented as a module inside the Apache source code. Measurements from the laboratory setup show how robust the implemented controller is, and how it corresponds to the results from the theoretical analysis.

Keywords system design · queueing theory · control theory · admission control

M. Kihl (✉) · M. Andersson
Department of Electrical and Information Technology,
Lund University, Box 118, 221 00, Lund, Sweden
e-mail: maria.kihl@eit.lth.se

M. Andersson
e-mail: mikael.andersson@eit.lth.se

A. Robertsson · B. Wittenmark
Department of Automatic Control,
Lund University, Box 118, 221 00, Lund, Sweden
e-mail: andersro@control.lth.se

B. Wittenmark
e-mail: bjorn@control.lth.se

1 Introduction

Today a web site can receive millions of hits per day and it may become overloaded as the arrival rate exceeds the server capacity. Capacity planning is therefore needed, which includes modelling and analysing the system with for example simulation tools. The need for good performance models of server systems has been discussed in, for example, [23]. Web servers can be modelled as queueing systems with one or more servers processing incoming requests at a certain rate. The web servers have a queue where requests wait for service.

One problem with web servers is that they are sensitive to overload. The servers may become overloaded during temporary traffic peaks when more requests arrive than the server is designed for. Because overload usually occurs rather seldom, it is not economical to overprovision the servers for these traffic peaks, instead admission control mechanisms can be implemented in the servers. Admission control may also be used to ensure that the system never crashes. An admission control mechanism rejects some requests when the traffic causes overload in the server.

Both capacity planning and the design of admission control mechanisms need performance models that are valid in the overloaded work region. Several attempts have been made to create performance models for web servers. In [22] servers were modelled as tandem queueing networks. In [31] web servers were modelled and analysed with colored Petri nets. In [7] a generalized processor sharing performance model for Internet access lines was proposed. The research concerning admission control has shown that the problem of optimally controlling the arrivals at a queueing system is a difficult task. The main problem comes from the fact that queueing 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 analyse different types of automatic control systems. Also, it contains a number of mathematical tools that may be used to analyse both the stability of a controlled system and to find good control schemes with respect to performance. One well-known controller in automatic control is the PID-controller, which enables a stable control for many types of systems (see, for example, [5]). The PID-controller uses three actions: one proportional, one integrating, and one derivative. Before designing the PID-controller, the system must be analyzed so that its dynamics during overload are known. This means that the system must be described with a control theoretic method. If the model is linear, it is easily analyzed with linear control theoretic methods. However, a queueing system is both nonlinear and stochastic. The main problem is that nonlinear models are much harder to analyse with control theoretic methods.

There are numerous early papers about admission control of server systems, in particular so called Stored Program Control (SPC) systems. An overview of this research field is given in [19]. One classical controller is the step-controller [13]. The objective of the control law is to keep the value of the control variable between an upper and a lower level. If the value of the variable is higher than the upper level, the admittance rate is decreased linearly. If the value is below the lower level, the admittance rate is increased.

Recent research on admission control mechanisms for server systems has mainly been focused on web servers. In [6] a queue length control with priorities was

developed. By optimizing a reward function, a static control was found in [10]. An on-off load control mechanism regulating the admittance of client sessions was developed in [12]. In [30] a control mechanism was proposed that combines a load control for the CPU with a queue length control for the network interface.

Few papers have investigated admission control mechanisms for web server systems with control theoretic methods, but it has gained a growing interest during the last years, see [14]. 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 [21]. In [9] 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. A queue length control with priorities was developed in [8]. Chen and Iyengar [11] investigated overload control schemes for distributed web sites.

In [16] and [24], we analyzed controllers for M/G/1-systems. The control objective in these papers was to keep the queue length at a certain level. For this rather simple system, it is possible to use fluid flow models that mimic the behavior of the queuing system. In the papers, we developed a nonlinear fluid flow model, based on the model in [3], and used this model for designing a PI-controller for the system. We demonstrated that linear models of this system are insufficient, 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 control mechanisms that will keep the CPU utilization on a certain level. The target system here is a web server system, which we model as a GI/G/1-system. For this system, the simple fluid flow models used in [16] and [24] are not applicable. We extend the investigations presented in the conference papers [17] and [18]. We develop and validate a control theoretic model of a GI/G/1-system that can be used for the design of CPU load controllers. We show that the model is valid for an Apache web server. Further, we design controller parameters for a PI-controller. We evaluate the controller both numerically and experimentally. Also, we perform a nonlinear stability analysis and determine a stability region for the PI-controller, see also [25]. Finally, the paper contains a discussion about the limitations with both linear control theoretic models of queueing systems and linear design methods.

2 Investigated system

The investigated system is shown in Figure 1. The server system is modelled as a GI/G/1-system with an admission control mechanism. New requests arrive according to a stochastic process with mean λ requests per second. If a request is admitted by the gate, it continues to an infinite queue. The service times have a statistical distribution with mean $E[X]$ seconds.

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 $[kh, kh + h]$ is denoted interval kh. The objective of the control mechanism is to keep the server utilization as close as possible to a reference value, ρ_{ref}.

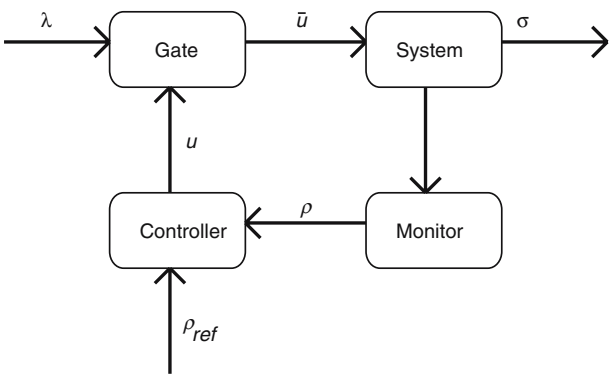

Figure 1 Queueing model.

The monitor measures the average value of the so called control variable, in this case the server utilization, ρ, during interval kh, $\rho(kh)$. At the end of interval kh, the controller calculates the desired number of admitted requests for interval $kh + h$, denoted $u(kh + h)$. The gate rejects those requests that cannot be admitted. The requests that are admitted proceed to the rest of the system. Since the number of admitted requests, $\bar{u}(kh)$, may never be larger than the number of arrived requests during interval kh, denoted $\alpha(kh)$, the actual number of admitted requests is $\bar{u}(kh) = min[u(kh), \alpha(kh)]$.

2.1 Gate

Several gates have been proposed in the literature, for example *Percent blocking*, *Token bucket* and *Dynamic window* mechanisms. In this paper we use the token bucket algorithm to reject those requests that cannot be admitted. New tokens are generated at a rate of $u(kh)$ tokens per second during time interval $[kh, kh + h]$. If there is an available token upon the arrival of a request, the request consumes the token and enters the web server. If there are no available tokens, the request is rejected. Rejected requests are assumed to leave the system without retrials.

2.2 Controllers

There are a variety of controllers to choose from when designing an admission control mechanism. Here follows a description of the controllers that are used in this paper.

2.2.1 Step controller

The step-controller is a classical controller in the telecommunication field. The objective of the control law is to keep the control variable between an upper and a lower level. If the value of the variable is higher than the upper level, the admittance

rate is decreased linearly. If the value is below the lower level, the admittance rate is increased. This means that the control law is as follows:

$$u(kh+h) = \begin{cases} u(kh) - s, & if\ \rho(kh) > \rho_{ref} + \epsilon \\ u(kh) + s, & if\ \rho(kh) < \rho_{ref} - \epsilon \\ u(kh), & else \end{cases} \quad (1)$$

where the value of s decides how much the rate is increased/decreased and the value of ϵ (size of dead-zone) decides how much the control variable may deviate from the reference value.

2.2.2 PI-controller

The PI-controller is a well-known controller in automatic control. It uses two actions: one proportional and one integrating. The discrete-time control law is given by:

$$u(kh+h) = K \cdot e(kh) + \frac{K}{T_i} \cdot \sum_{i=0}^{k-1} e(ih) \quad (2)$$

where $e(kh) = \rho_{ref} - \rho(kh)$ is the error between the reference and actual value of the control variable. The gain K and the integral time T_i are the controller parameters that are set so that the controlled system behaves as desired. For many systems a large value of K makes the controller faster, but weakens the stability. The integrating action eliminates stationary errors, but may also make the system less stable. This is however dependent on the system dynamics and needs to be formally analyzed.

2.2.3 RST-controller

For a more general controller structure, the polynomial methods proposed in [5] can be used. The controller will be a two-degree-of-freedom controller, which allows for separate feedback and prefiltering design (pole-placement and handling of reference values). The control law in discrete-time for a general so-called RST-controller is given by

$$R(q)u(kh) = T(q)\rho_{ref}(kh) - S(q)\rho(kh) \quad (3)$$

where $R(q)$, $T(q)$, and $S(q)$ are functions expressed in the forward-shift operator q (i.e., $qu(kh) = u(kh+h)$). R, S, and T are designed so that the closed loop system behaves as desired.

3 Control theoretic model

Control theory is a powerful tool for performance analysis of computer controlled systems. The system must be described in terms of transfer functions or differential (or difference) equations. A transfer function describes the relationship between the z-transforms (or Laplace transforms in case of continuous-time systems) of the

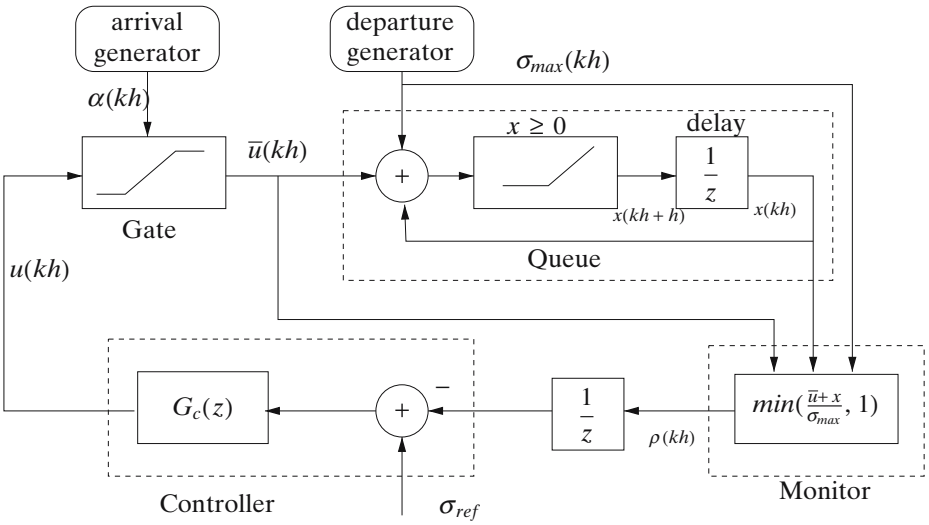

Figure 2 A control theoretic model of a GI/G/1-system with admission control.

input and the output of a system. In this case, the input to the system is the actual admittance rate, $\bar{u}$, whereas the output is the server utilization, denoted ρ.

We use the discrete-time control theoretic model shown in Figure 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. We assume that the sampling period, h, is chosen to guarantee that the quantization effects around the sampling times are small, see [5]. The system consists of an arrival generator, a departure generator, a controller, a queue and a monitor.

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 denoted $\alpha(kh)$. $\alpha(kh)$ is an integrated stochastic process over one sampling period with a distribution obtained from the underlying interarrival time distribution. If, for example, the arrival process is Poisson with mean λ, then $\alpha(kh)$ is Poisson distributed with mean λh. The *departure generator* decides the maximum number of departures during interval kh, denoted $\sigma_{max}(kh)$. $\sigma_{max}(kh)$ is also a stochastic process with a distribution given by the underlying service time distribution. If, for example, the service times are exponentially distributed with mean $1/\mu$, then $\sigma_{max}(kh)$ is Poisson distributed with mean μh. It is assumed that $\alpha(kh)$ and $\sigma_{max}(kh)$ are independent from between sampling instants and uncorrelated to each other.

The *gate* is constructed as a saturation block that limits the number of admitted requests during interval kh, $\bar{u}(kh)$, to be

$$\bar{u}(kh) = \begin{cases} 0 & u(kh) < 0 \\ u(kh) & 0 \leq u(kh) \leq \alpha(kh) \\ \alpha(kh) & u(kh) > \alpha(kh) \end{cases}$$

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, $\rho(kh)$, 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, $\rho(kh)$, and the reference value, ρ_{ref}. The control law is given by the transfer function, $G_c(z)$.

4 Stability analysis of closed loop system

In this section we will consider the stability properties of the controlled server system, when using a PI-controller for admission control. First, we will consider an approach based on a linear queue model and compare with the admission control parameters

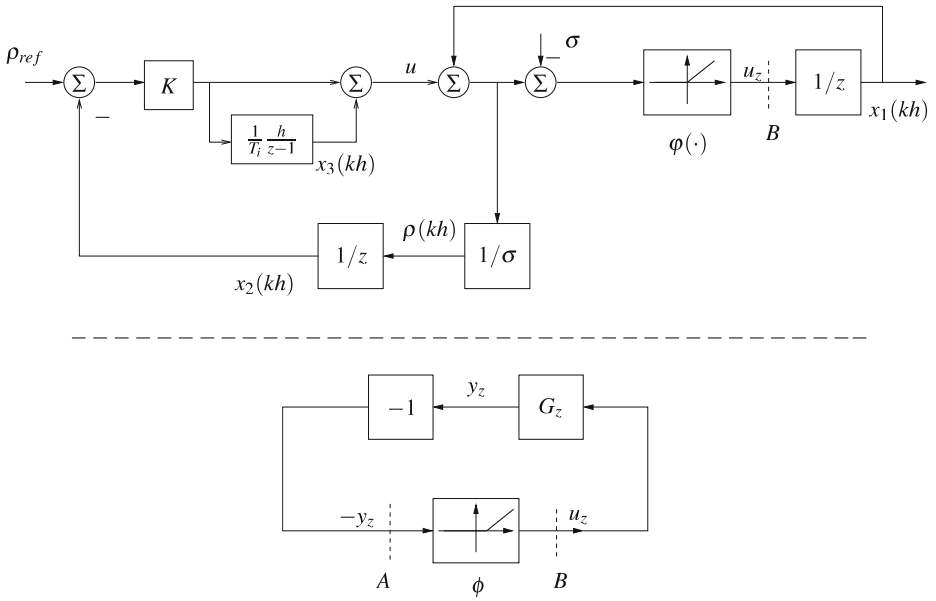

Figure 3 Decomposition of the upper original model into a linear block (G_z) and a nonlinear block (ϕ) under negative feedback.

derived from nonlinear analysis. The analysis is based on the Tsypkin/Jury-Lee stability criterion (discrete-time versions of the Popov criterion) [29].

The original model in Figure 2 contains three nonlinearities in the gate, the queue and the monitor, respectively, where investigations showed that the queue-limitation is the dominating nonlinearity. Therefore, in this analysis only the queue-limitation will be considered. The saturation due to limited arrival rate can be handled with a standard implementation of an anti-reset windup scheme, see [24].

The new model is shown in the upper part of Figure 3. Here, σ is the average value of σ_{max}. We introduce three states $\{x_1, x_2, x_3\}$ corresponding to the queue length, the (delayed) server utilization ρ and the integrator state in the PI-controller, respectively. A standard PI-controller is used, which has the transfer function $G_c(z) = K(1 + \frac{1}{T_i} \cdot \frac{h}{z-1})$. The queue limitation is denoted φ.

To analyse the stability of the closed loop system where we take the queue nonlinearity into account, we partition the system into a linear part in feedback connection with the nonlinearity, ϕ, as shown in the lower part of Figure 3.

4.1 Linear design (neglecting saturations)

Neglecting the nonlinearities in Figure 2, by assuming that $\varphi(z) = z$ (i.e., linear and no saturation) will result in the closed loop dynamics

$$\begin{aligned} G_c &= \frac{G_c(1+G_q)G_m}{1+G_c(1+G_q)G_m} \\ &= \frac{z \cdot K/\sigma\,(z-1+h/T_i)}{z \cdot (z^2+(K/\sigma-2)z+(1-K/\sigma+Kh/(\sigma T_i))} \end{aligned} \tag{4}$$

where G_q and G_m represent the queue and monitor dynamics, respectively.

To match the characteristic polynomial

$$z \cdot (z^2+(K/\sigma-2)z+(1-K/\sigma+Kh/(\sigma T_i)) \tag{5}$$

with a desired characteristic polynomial

$$z \cdot (z^2+a_1 z+a_2) \tag{6}$$

we get the control parameters

$$K = (2+a_1)\,\sigma, \quad T_i = h\,(2+a_1)/(1+a_1+a_2)$$

Using the parameters of the PI-controller it is thus possible to make an arbitrary pole-placement, except for the pole $z = 0$, which corresponds to a time delay. A simplified linear analysis will thus predict stability for the closed loop for all coefficients $\{a_1,\ a_2\}$ belonging to the stability triangle

$$\{\ a_2 < 1, \quad a_2 > -1+a_1, \quad a_2 > -1-a_1\ \}, \tag{7}$$

see [5] for more details.

4.2 Model with queue limitation

Consider the upper part of Figure 3. The state space model will be

$$\begin{aligned} x_1(kh+h) &= \varphi\left(u + x_1(kh) - \sigma\right) \\ x_2(kh+h) &= \frac{1}{\sigma}(u + x_1(kh)) \\ x_3(kh+h) &= Kh/T_i(\rho_{ref} - x_2(kh)) + x_3(kh) \end{aligned} \tag{8}$$

where $u = K(\rho_{ref} - x_2) + x_3$ and $\varphi(\cdot)$ is the saturation function in Figure 3. By introducing the *forward shift operator* and leaving out the time arguments, we get

$$q\,x_1 = \varphi\left(K(\rho_{ref} - x_2) + x_3 + x_1 - \sigma\right) \tag{9}$$

$$q\,x_2 = \frac{1}{\sigma}\left(K(\rho_{ref} - x_2) + x_3 + x_1\right) \tag{10}$$

$$q\,x_3 = Kh/T_i\,(\rho_{ref} - x_2) + x_3 \tag{11}$$

The equilibrium for the system (9–11) satisfies $qx = x$.

From (11) we get

$$x_3 = Kh/T_i\,(\rho_{ref} - x_2) + x_3 \quad \Rightarrow \quad x_2^o = \rho_{ref} \tag{12}$$

Inserting this in (9) and (10) we get

$$\begin{aligned} x_1^o &= \varphi(x_3^o + x_1^o - \sigma) \\ x_2^o &= \rho_{ref} = \frac{1}{\sigma}\left(x_3^o + x_1^o\right) \\ &\Rightarrow \\ x_1^o &= \varphi\left(\sigma(\rho_{ref} - 1)\right) \end{aligned} \tag{13}$$

As $\rho_{ref} \in [0, 1]$ and using the fact that $\varphi(z) = 0,\ \forall z \leq 0$ we get

$$\begin{cases} x_1^o = 0 \\ x_2^o = \rho_{ref} \\ x_3^o = \sigma x_2^o = \sigma\rho_{ref} \end{cases} \tag{14}$$

By introducing the change of variables

$$\begin{cases} z_1 = x_1 - 0 \\ z_2 = x_2 - \rho_{ref} \\ z_3 = x_3 - \sigma\rho_{ref} \end{cases} \quad \text{or} \quad \begin{aligned} x_1 &= z_1 \\ x_2 &= z_2 + \rho_{ref} \\ x_3 &= z_3 + \sigma\rho_{ref} \end{aligned}$$

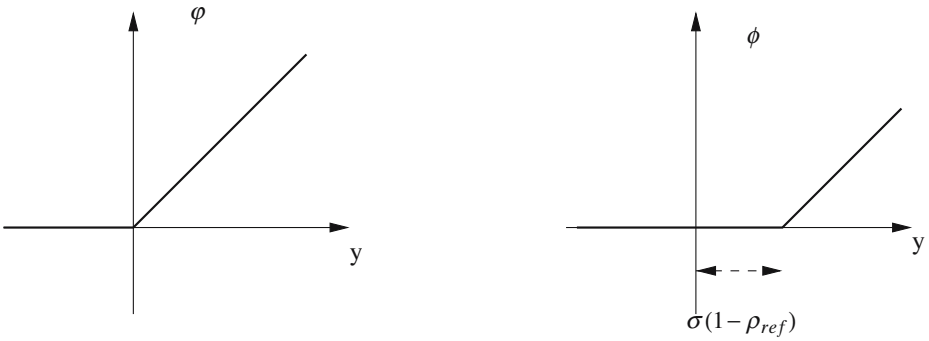

Figure 4 $\phi(y) = \varphi(y - \sigma(1 - \rho_{ref}))$ where $\sigma > 0$ and $\rho_{ref} \in [0, 1]$.

we get

$$\begin{aligned} q\,z_1 &= q\,x_1 - 0 &&= \varphi\left(-Kz_2 + z_3 + z_1 - \sigma\right) \\ q\,z_2 &= q\,x_2 - \rho_{ref} &&= \frac{1}{\sigma}\left(-Kz_2 + z_3 + \sigma\rho_{ref} + z_1\right) - \rho_{ref} \\ q\,z_3 &= q\,x_3 - \sigma\rho_{ref} &&= -\,Kh/\,T_i\,z_2 + z_3 + \sigma\rho_{ref} - \sigma\rho_{ref} \end{aligned}$$

The equation system above can be rewritten as a linear system in negative feedback with the nonlinear function $\phi : y \to \varphi(y - \sigma(1 - \rho_{ref}))$, as shown in the lower part of Figure 3. The system is given by

$$\begin{aligned} qz &= A_z z + B_z u_z = A_z z + B_z \phi(-y) \\ y &= C_z z \end{aligned} \tag{15}$$

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}, \quad A_z = \begin{bmatrix} 0 & 0 & 0 \\ 1/\sigma & -K/\sigma & 1/\sigma \\ 0 & -Kh/\,T_i & 1 \end{bmatrix}, \quad B_z = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad C_z = \begin{bmatrix} -1 & K & -1 \end{bmatrix}$$

Note that for $\rho_{ref} \in [0, 1]$ the function $\phi(\cdot)$ will belong to the same cone as $\varphi(\cdot)$, namely $[\alpha, \beta] = [0, 1]$, see Figure 4. The incremental variation will also have the same maximal value (=1).

The transfer function $G_z = G_{u_z \to y_z}(z)$ from cut B to cut A in Figure 3 will be

$$\begin{aligned} G_z &= C_z(zI - A_z)^{-1} B_z \\ &= \frac{-z \cdot (z - 1)}{z \cdot \left(z^2 + (-1 + K/\sigma)\,z + K\,(h - T_i)\,/(\sigma\,T_i)\right)} \end{aligned} \tag{16}$$

4.3 Stability analysis for the discrete-time nonlinear system

In the forthcoming stability analysis we determine for which controller parameters the linear subsystem G_z is stable. The poles of (16) are stable, i.e., inside the unit circle ($|z| < 1$), for the area depicted in Figure 5 for the normalized parameters K/σ and $h/\,T_i$.

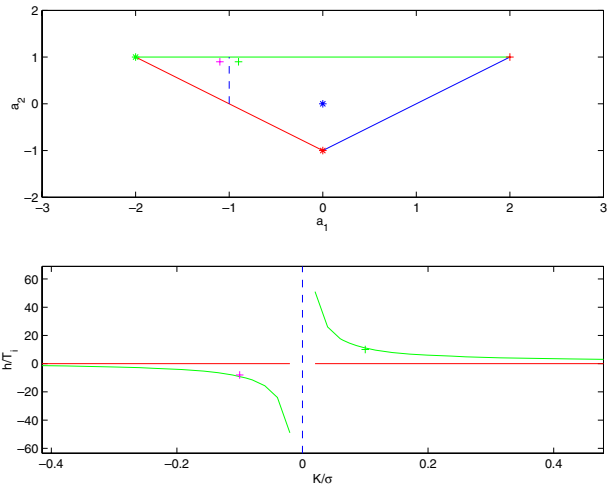

Figure 5 *(Upper:)* The internal of the *triangle* corresponds to the stability area for the characteristic polynomial $z \cdot (z^2 + a_1 z + a_2)$ of G_z. *(Lower:)* Corresponding stabilizing control parameters $\{K/\sigma,\ h/T_i\}$.

To determine the stability for the nonlinear system in (15) we can use the Tsypkin criterion or the Jury-Lee criterion which are the discrete-time counterparts of the Popov criterion for continuous time systems [20, 29]. Sufficient conditions for stability are that G_z has all its poles within the unit circle and that there exists a (positive) constant η such that

$$\mathrm{Re}\left[(1+\eta(1-z^{-1}))G_z(z)\right] + \frac{1}{k} \geq 0 \quad \text{for} \quad z = e^{i\omega}, \quad \omega \geq 0 \tag{17}$$

where the nonlinearity ϕ belongs to the cone $[0, k = 1]$.

In the upper left plot of Figure 6 we have *the stability triangle* for the characteristic polynomial of (5). By choosing coefficients for the characteristic polynomial (5) in

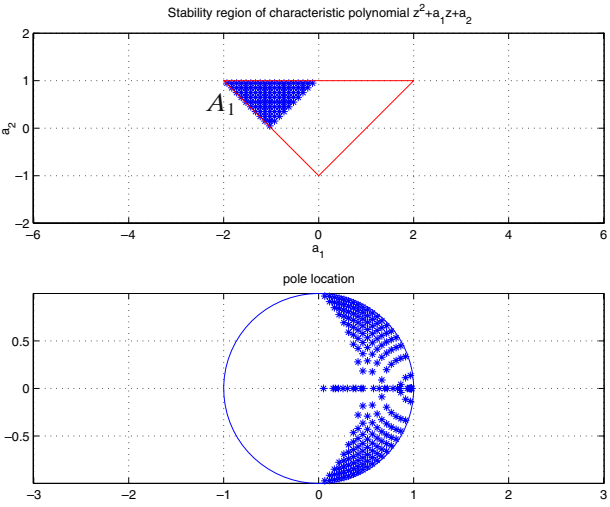

Figure 6 *(Upper:)* The *large triangle* is the stability area a linear model would predict. However, from this parameter set, nonlinear analysis guarantees only stability for parameters $\{a_1,\ a_2\}$ in the *upper left triangle* $(A_1, '*')$. *(Lower:)* Pole location corresponding to $\{a_1,\ a_2\} \in A_1$.

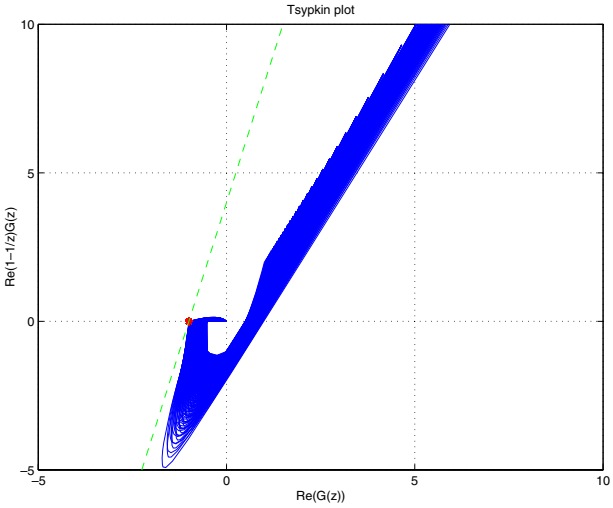

Figure 7 Set of Tsypkin plots which all satisfy the frequency condition (17) for $G_z = G_z(K,\ T_i)$, where $(K,\ T_i)$ correspond to pole locations to the *left upper stability triangle* A_1 i Figure 6.

the upper left triangle (A_1) we will get controller parameters $\{K,\ T_i\}$ which also will give a stable transfer function G_z. The corresponding poles are plotted in the lower diagram of Figure 6. Figure 7 shows a graphical representation of the Tsypkin condition (17) for this set of control parameters. The dashed non-intersecting line in Figure 7 corresponds to the existence of a positive parameter η satisfying (17). Thus, absolute stability for the nonlinear system also is guaranteed for this choice of parameters.

Remark: The Tsypkin criterion guarantees stability for any cone bounded nonlinearity in $[0, 1]$ and we can thus expect to have some robustness in addition to stability in our case.

5 Numerical investigations

In the numerical investigations, the queueing model was represented by a discrete-event simulation program implemented in C. Also, the control theoretic model was implemented with the Matlab Simulink package. The objectives of the numerical investigations were to validate the control theoretic model and thereafter design suitable controllers for an example system.

During all investigations, the reference load, ρ_{ref}, was set to 0.8, and the average arrival rate, λ, was 150 jobs per second. Two systems were used in the numerical investigations, one M/M/1-system and one M/H_2/1-system. In both cases the average service time, $E[X]$ = 0.02 s. The parameters for the H_2-distribution was μ_1 =20, μ_2 =600, and p_1 =0.38 which gives a squared coefficient of variance of 3.74. 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. However, the system becomes more difficult to control if there is more variation in the system. This will be shown in the results.

5.1 Performance metrics

The admission control mechanism has two control objectives. First, it should keep the control variable at a reference value, i.e., the control error, $e = \rho_{ref} - \rho$, should be as small as possible. Second, it should react rapidly to changes in the system, i.e., the so-called settling time should be short.

Therefore, we compare the controllers in two ways. First, we show the steady-state distribution of the server utilization, by plotting the estimated distribution function, i.e., $P(\rho \leq \rho_0)$ where $0 \leq \rho_0 \leq 1$. The distribution function shows how well the controller meets the first control objective. Second, we plot the step response when starting with an empty system. The step responses show the transient behavior of the controllers.

5.2 Validation of the control theoretic model

We have numerically validated that the open system, that is without control feedback, is accurate in terms of the server utilization distribution. The distribution functions of the server utilization for varying arrival rates are shown in Figures 8 and 9. The distribution function was estimated from 5,000 measurements of the server utilization for a specific parameter setting. As can be seen, the control theoretic model and the queueing model behave similar. There is a slight deviation for low loads, however this is not important since the concerned region for our investigations is when the load is high. Also, we have validated the closed loop system for various controllers and traffic cases. The results showed that the control theoretic model is accurate enough to be used when designing controllers for the "real" system, that is the queueing system.

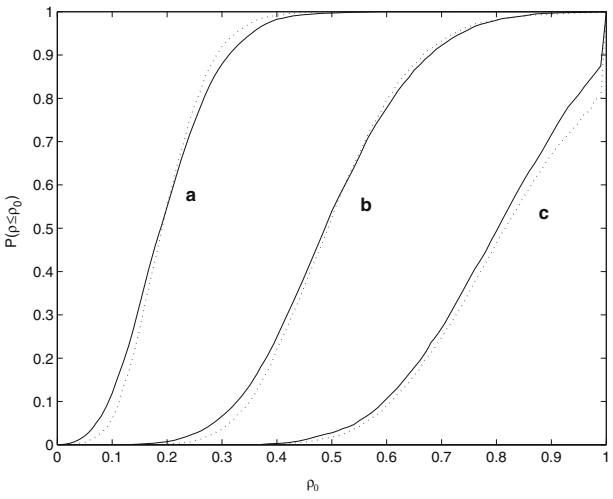

Figure 8 Server utilization distributions for the open $M/M/1$-system. **a** l=10, **b** l=25, **c** l=40 *Dotted line*: control theoretic model, *solid line*: simulation.

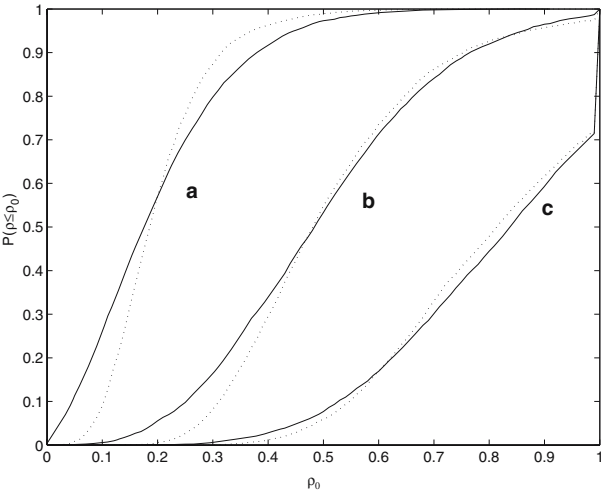

Figure 9 Server utilization distributions for the open $M/H_2/1$-system. **a** l=10, **b** l=25, **c** l=40 *Dotted line*: control theoretic model, *solid line*: simulation.

5.3 Controller design

In this section four controllers are designed for the system. We use a static controller as a benchmark and then compare this controller with a step controller, a PI-controller, and an RST-controller.

5.3.1 Static 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 given by

$$u_{fix} = \rho_{ref} E[X]$$

which in this case is equal to 40 jobs per second.

5.3.2 Step controller

The parameters for the step-controller are the step size, s, the marginal, ϵ, and the sampling period, h. These parameters are usually chosen by ad hoc methods, that is, with "trial and error" methods. Our investigations demonstrated a problem with the step-controller. If the sampling period is too short, the controller will overestimate the load of the system, and thereby create a stationary error in the controlled load. For our system, a sampling period of 2 s was necessary to eliminate the stationary error. Also, we selected a step size of 5, and a marginal of 0.05. With these parameters the steady-state behavior of the controlled system became close to the benchmark controller. It was not possible to find a step-controller that had a better steady-state behavior than the static controller.

5.3.3 PI-controller

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 s (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 (uncanceled) roots of the characteristic polynomial in (5) will be $0.4 \pm 0.2i$. These poles are inside the stability area A_1 of Figure 6.

5.3.4 RST-controller

For a pole placement design the controller polynomials (R, S, and T) are found by solving a Diophantine equation,

$$A(q)R(q) + B(q)S(q) = A_m(q)A_o(q) \tag{18}$$

describing the relationship of the process polynomials (A and B), the controller polynomials and the desired closed loop polynomials (A_m and A_o). The linear system has

$$\begin{aligned} A(q) &= q^2 - q \\ B(q) &= \frac{1}{\sigma} \cdot q \end{aligned} \tag{19}$$

A_m and A_0 are chosen so that the poles of the closed loop system are placed as desired. Note that there is a stable pole-zero cancellation at z=0 in the transfer function from the control signal u to the load. If the sampling interval is set to h=0.2 seconds and the desired closed loop poles are chosen as {0.4, 0.2} we get

$$A_m A_o = q^2 - 0.6q + 0.08 \tag{20}$$

If we impose the controller to have integral part, the controller polynomials are given by

$$\begin{aligned} R(q) &= q - 1 \\ S(q) &= 14q - 9.2 \\ T(q) &= 6q - 1.2 \end{aligned} \tag{21}$$

The polynomial control structure used in this section could also allow for minimum variance and LQ/LQG control criteria design, see [5].

5.4 Controller comparisons

This section contains a comparison of the controllers that were designed above.

5.4.1 Distribution function

The distribution functions for the two systems are shown in Figures 10 and 11. The optimal distribution function is zero for $0 \leq \rho \leq 0.8$ and one for $0.8 \leq \rho \leq 1$.

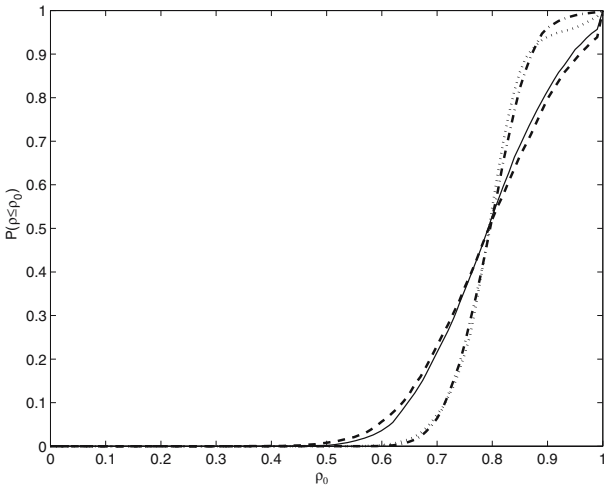

Figure 10 Distribution functions for the M/M/1-system. *solid line*: static controller, *dashed line*: step-controller, *dotted line*: PI-controller, *dash-dotted line*: RST-controller.

The systems with a step-controller behaves similar to the systems with a static controller. We can conclude that the $M/H_2/1$-system is more difficult to control than the $M/M/1$-system, since the distribution functions are remoter from the optimum function. However, as can be seen, the systems with a PI-controller and an RST-controller actually behave better than the systems with a static controller. This phenomenon is due to that those controllers 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 controllers behave as the static controller.

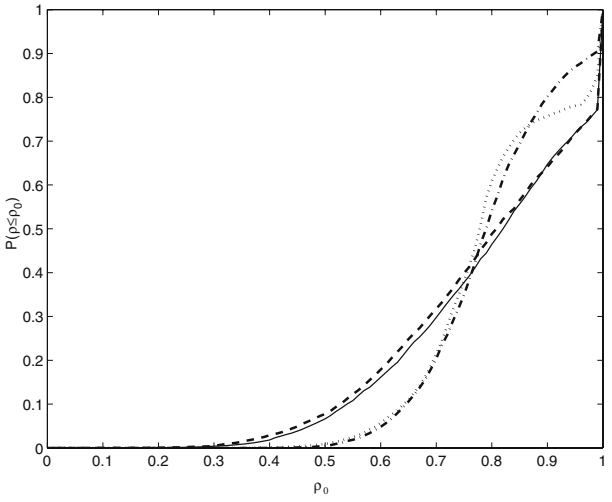

Figure 11 Distribution functions for the M/H/1-system. *solid line*: static controller, *dashed line*: step-controller, *dotted line*: PI-controller, *dash-dotted line*: RST-controller.

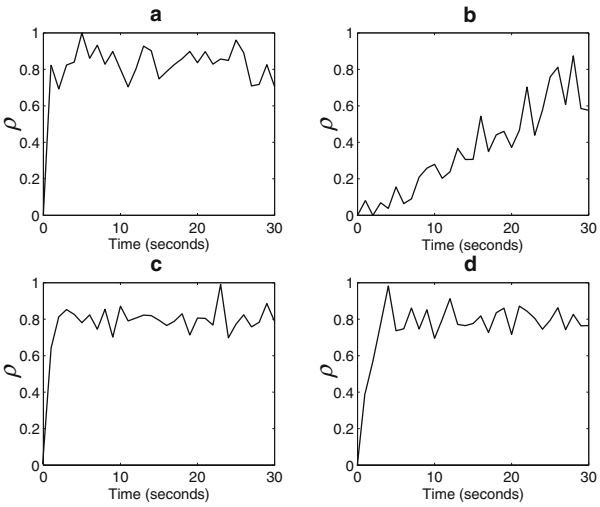

Figure 12 Step responses for the load controlled M/M/1-system. **a** static controller, **b** step-controller, **c** PI-controller, **d** RST-controller.

5.4.2 Average step response

The step responses for the load controlled $M/M/1$-system are shown in Figure 12. The step responses for the $M/H_2/1$-system are similar. As can be seen, the controllers have very divergent step responses. 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, whereas the RST-controller is slightly slower. The step-controller is very slow. Even after 30 s it has not managed to find a correct admittance rate. This is the main problem with the step-controller. In order to achieve a good steady-state behavior, the step size must be small and the sampling period must be long. However, this means that it takes a long time for the controller to adapt to changes in the system.

5.4.3 Robustness

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. Therefore, we investigated the robustness of the controllers by changing the average service time in the system. In the first case, the average service time was increased with 30% ($E[X] = 0.026$ s), and in the second case the average service time was decreased with 30% ($E[X] = 0.014$ s).

All controllers were tested, however we only show the results for the PI-controller in Figure 13. As can be seen, the PI-controller works well even when the average service time is changed. The step-controller has a similar result, The RST-controller is very robust, since the distribution function is the same even when the average service time is changed with 30%. The static controller is, of course, dependent on a

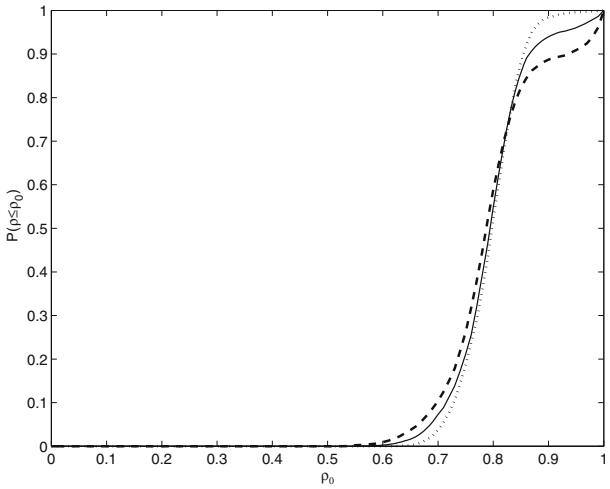

Figure 13 Robustness of PI-controller: Server utilization distribution function. *Solid line*: $\bar{x} = 0.020$ s, *dotted line*: $\bar{x} = 0.014$ s, *dashed line* = $\bar{x} = 0.026$ s.

correct service time, which means that it cannot operate properly when the service times change.

5.5 Discussion

The analysis in Section 4.3 gives sufficient conditions and a region for control parameters which guarantee stability of the nonlinear closed loop as well as for the simplified linear model. We are of course not restricted to choose parameters from only this region as the main objective is that the nonlinear system should be stable. However, we can conclude that

- Pole-placement based on a linear model is OK in a restricted area (region A_1 in Figure 6).
- There are choices of parameters that gives stable closed loop poles, but where the linear analysis would indicate an unstable closed loop systems.

During simulation studies the dominant nonlinear effect has come from the queue nonlinearity φ. The saturation due to limited arrival rate can be handled with a standard implementation of an anti-reset windup scheme, see [24].

6 Experimental investigations

The models and analysis in this paper are based on a web server system that can be modelled as a queuing system. In the experimental evaluation of the controllers we have used an Apache web server [4], which is further described below.

6.1 Web servers

A web server like Apache, contains software that offers access to documents stored on the server. Clients can browse the documents in a web browser. The documents

can be for example static Hypertext Markup Language (HTML) files, image files or various script files, such as Common Gateway Interface (CGI), Java scripts or Perl files. The communication between clients and server is based on HTTP [28]. An HTTP transaction consists of three steps: TCP connection setup, HTTP layer processing and network processing. The TCP connection setup is performed through a threeway handshake, where the client and the server exchange TCP SYN, TCP SYN/ACK and TCP ACK messages. Once the connection has been established, a document request can be issued with an HTTP GET message to the server. The server then replies with an HTTP GET REPLY message. Finally, the TCP connection is closed by TCP FIN and TCP ACK messages in both directions.

Apache, which is a well-known web server and widely used, is multi-threaded. This means that a request is handled by its own thread or process throughout the life cycle of the request.

6.2 Experimental system

Our experimental investigations used one server computer and one computer representing the clients connected through a 100 Mbits/s Ethernet switch. The server was a PC Pentium III 1,700 MHz with 512 MB RAM running Windows 2000 as operating system. The computer representing the clients was a PC Pentium II 400 MHz with 256 MB RAM running RedHat Linux 7.3. Apache 2.0.45 was installed in the server. We used the default configuration of Apache. The client computer was installed with an HTTP load generator, which was a modified version of S-Client[6]. We modified the S-Client code to use Poissonian arrivals instead of the original deterministic ones. The client program was programmed to request dynamically generated HTML files from the server. The CGI script was written in Perl. It generates a number of random numbers, adds them together and returns the summation. The average request rate was set to 100 requests per second in all experiments except for the measurements in Figure 14. In all experiments, the sampling period, h, was set to 1 s.

6.3 Admission control

The admission control mechanism was implemented in the Apache web server. Apache is made up of a core package and several modules that handle different operations, such as Common Gateway Interface (CGI) execution, logging, caching etc. A new module was created that contains the admission control mechanisms. The new module was then hooked into the core of Apache, so that it was called every time a request was made to the web server. The module could then either reject or admit the request according to the control mechanism. The admission control mechanism was written in C and tested on a Windows platform.

The Monitor was implemented as a thread that samples the server utilization every control interval. The server utilization is calculated as one minus the fraction of time an idle process has been able to run during the last control interval. The idle process' priority level is set to the lowest possible, which means that it only runs whenever there is no request requiring CPU work. This way of measuring the load on the CPU results in a quantization effect in server utilization. The reason to this is that the operating system where the admission control mechanism runs has a certain time resolution in function calls regarding process uptimes. This means that the control

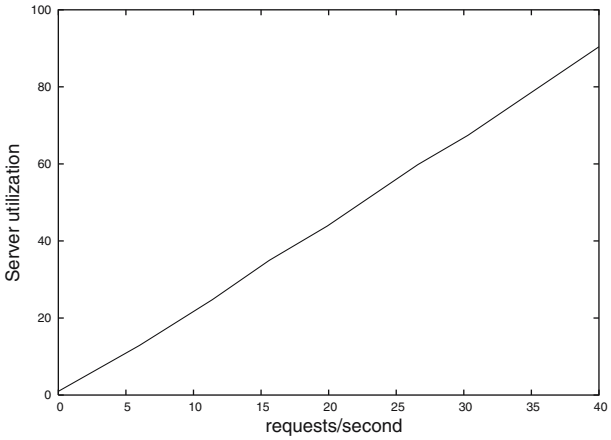

Figure 14 Average server utilization for the open system.

interval cannot be chosen arbitrary. It has to be long enough not to be affected by the time resolution effects, and short enough so that the controller responds quickly.

The controller tries to minimize the error between the server utilization and the reference value, ρ_{ref}. The controller's output is forwarded to the gate module. In this paper we have implemented and evaluated a PI-controller.

6.4 Validation of the model

We have validated that the queueing model (GI/G/1) is an accurate representation of the experimental system, in terms of average server utilization. The average server utilization for varying arrival rates are shown in Figure 14. For a single-server queue, the server utilization is proportional to the arrival rate, and the slope of the server utilization curve is given by the average service time. The measurements in Figure 14 gives an estimation of the average service time in the web server, $E[X]$=0.0225 s. Further, we have in a number of experiments validated that the GI/G/1-model

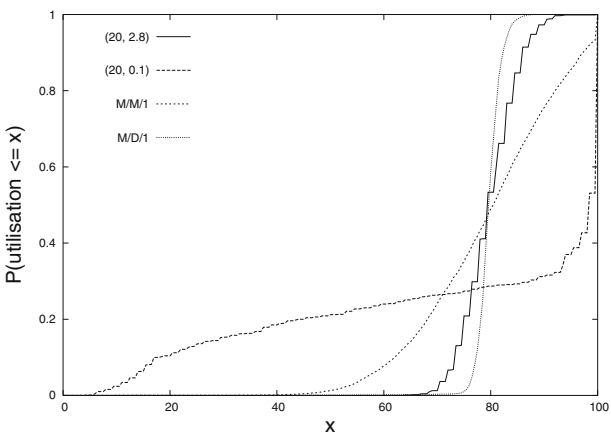

Figure 15 Server utilization distribution of measurements from the real system together with simulations of the M/M/1 and the M/D/1 system.

can be used when designing admission control mechanisms for an Apache web server system.

6.5 Controller design

Control parameters for the PI-controller are chosen from the stability area A_1 in Figure 6. In the simulations and experiments below we use $\{K, T_i\}=\{20, 2.8\}$. The parameter setting is also compared to $\{K, T_i\}=\{20, 0.1\}$, found outside the stability area.

6.6 Distribution function

The distribution function was estimated from measurements during 1,000 s with the specific parameter setting. Figure 15 shows the estimated distribution function for the PI-controller. Both good and bad parameter settings were used. An ideal admission control mechanism would show a distribution function that is zero until the wanted load, and that is one thereafter. In this case, the load was kept at 0.8,

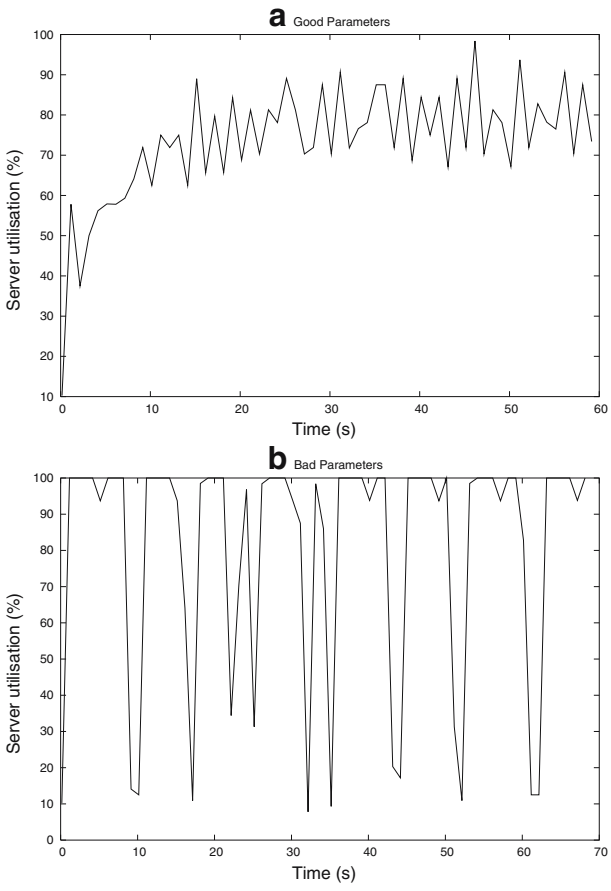

Figure 16 **a** Example of a realisation with good parameters. **b** Example of a realisation with bad parameters.

and the parameter setting, {K, Ti}={20, 2.8}, chosen from results in a controller that behaves very well in this sense. The parameter setting, $\{K, T_i\}$={20, 0.1}, as can be seen, perform worse. Also, as comparison, results from simulations of the M/D/1 system and the M/M/1 system are given in Figure 15, when using $\{K, Ti\}$={20, 2.8}.

6.7 Step response

Figure 16 shows the behaviour of the web server during the transient period. The measurements were made on an empty system that was exposed to 100 requests per second. The good parameter setting, $\{K, Ti\}$={20, 2.8}, exhibits a short settling time with a relatively steady server utilization. The bad parameter setting, $\{K, Ti\}$= {20, 0.1} has its poles outside the unit circle and behaves badly with large load oscillations.

6.8 Bursty arrival traffic

As stated before, the design of controller parameters is independent of the arrival process. To show this, we also performed experiments using a more bursty arrival process. The arrival process was a two-state Markov Modulated Poisson Process (MMPP-2). An MMPP is a doubly stochastic Poisson process where the rate process is determined by a continuous-time Markov chain. An MMPP-2 means that the Markov chain consists of two different states, S_1 and S_2. The Markov chain changes state from S_1 to S_2 with intensity $r_1\ s^{-1}$, and transits back with intensity $r_2\ s^{-1}$. When the MMPP is in state S_1, the arrival process is a Poisson process with rate λ_1 requests per second, and when the MMPP is in state S_2, the arrival rate is λ_2 requests per second. We used $r_1 = 0.05$, $r_2 = 0.95$, $\lambda_1 = 75$, and $\lambda_2 = 475$.

MMPPs have been commonly used to model bursty arrival processes, for example to web servers. In [27] web traffic was modeled as MMPPs. In [15], an HTTP generator was constructed from analyzed traffic samples. The chosen traffic model was MMPP. Yoshihara et al. [32] and Salvador et al. [26] showed how traffic that exhibit self-similar properties can be modeled as MMPPs.

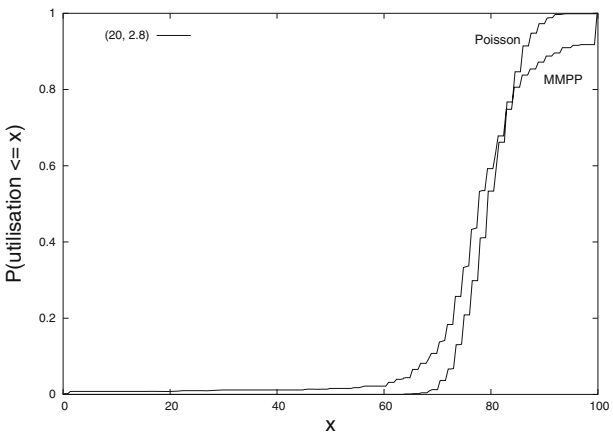

Figure 17 Server utilization distribution of measurements from the real system. M = Poisson process.

Figure 17 shows the steady state server utilization distribution for the experimental system, when using a PI-controller with $\{K, Ti\}$=$\{20, 2.8\}$. As comparison, the results for the Poisson process is also shown in the figure. As can be seen, the system behaves as predicted, even though the bursty arrival traffic makes the system harder to control.

7 Conclusion

Admission control mechanisms have since long been developed for various server systems. Traditionally, queueing theory has been used when investigating server systems, since they usually can be modelled as queueing systems. However, there are no mathematical tools in queueing theory that can be used when designing admission control mechanisms. Therefore, these mechanisms have mostly been developed with empirical methods. Control theory contains many mathematical tools that can be used when designing admission control mechanisms.

In this paper, we have designed load control mechanisms for a $GI/G/1$-system with control theoretic methods. We have compared a PI-controller and an RST-controller, both commonly used in automatic control, with a static controller and a step controller, both commonly used in telecommunication systems. We have shown that, when considering transient and stationary behavior, and robustness, both the PI-controller and the RST-controller behave better than the other controllers.

Also, we perform a nonlinear stability analysis for a PI-controlled system. We show that linear design is sufficient for stability of the nonlinear system. However, there are some limitations with linear design, which should be considered. 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. and with experiments on an Apache web server. We have shown that the control theoretic model can be used when designing admission control mechanisms for the Apache web server.

References

1. Abdelzaher, T., Lu, C.: Modeling and performance control of internet servers. In: Proceedings of the 39th IEEE Conference on Decision and Control, pp. 2234–2239 (2000)
2. Abdelzaher, T., Shin, K., Bhatti, N.: Performance guarantees for web server end-systems: a control theoretic approach. IEEE Trans. Parallel Distrib. Syst. **13**(1), 80–96 (2002)
3. Agnew, C.: Dynamic modeling and control of congestion-prone systems. Oper. Res. **24**(3), 400–419 (1976)
4. Apache Web server, home page: http://www.apache.org, as of 2007
5. Åström, K., Wittenmark, B.: Computer-controlled Systems. Theory and Design, 3rd edn. Prentice-Hall, Englewood Cliffs, NJ (1997)
6. Banga, G., Druschel, P.: Measuring the capacity of a web server under realistic loads. In: World Wide Web Journal, vol. 2, pp. 69–83. Springer, Berlin Heidelberg New York (1999)
7. Beckers, J., Hendrawan, I., Kooij, R.E., van der Mei, R.: Generalized processor sharing performance model for internet access lines. In: 9th IFIP Conference on Performance Modelling and Evaluation of ATM and IP Networks, Budapest (2001)
8. Bhatti, N., Friedrich, R.: Web server support for tiered services. IEEE Netw. 64–71. Sept/Oct (1999)

9. Bhoj, P., Ramanathan, S., Singhal, S.: Web2K: bringing QoS to web servers. HP Labs Technical report, HPL-2000-61 (2000)
10. Carlström, R.R.J.: Application-aware admission control and scheduling in web servers. In: Proceedings of Infocom (2002)
11. Chen, H., Iyengar, A.: A tiered system for serving differentiated content. In: World Wide Web Journal, vol. 6, pp. 331–352. Springer, Berlin Heidelberg New York (2003)
12. Cherkasova, L., Phaal, P.: Predictive admission control strategy for overloaded commerical web servers. In: Proceedings of the 8th International IEEE Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 500–507 (2000)
13. Farel, R., Gawande, M.: Design and analysis of overload control strategies for transaction network databases. In: Proceedings of the 13th International Teletraffic Congress, pp. 115–120 (1991)
14. Hellerstein, J.L., Yixin Diao, Parekh, S., Tilbury, D.M.: Control engineering for computing systems. IEEE Control Syst. Mag. **25**(6), 56–68 (2005)
15. Hryń, G., Jerzak, Z., Chydziński, A.: MMPP-based HTTP traffic generation with multiple emulated sources. Arch. Inform. Teor. Stosow. **16**, 321–335 (2004)
16. Kihl, M., Robertsson, A., Wittenmark, B.: Analysis of admission control mechanisms using non-linear control theory. In: Proceedings of IEEE International Symposium on Computer Communications (2003)
17. Kihl, M., Robertsson, A., Wittenmark, B.: Performance modelling and control of server systems using non-linear control theory. In: Proceedings of the 18th International Teletraffic Congress (2003)
18. Kihl, M., Robertsson, A., Wittenmark, B.: Control theretic modelling and design of admission control mechanisms for server systems. In: Proceedings of IFIP Networking (2004)
19. Körner, U., Nyberg, C.: Overload control in communication networks. In: Proceedings of Globecom'91, pp. 1331–1335 (1991)
20. Larsen, M., Kokotovic, P.V.: A brief look at the Tsypkin criterion: from analysis to design. Int. J. Adapt. Control Signal Process. **15**(2), 121–128 (2001)
21. Lu, C., Abdelzaher, T., Stankovic, J., So, S.: A feedback control approach for guaranteeing relative delays in web servers. In: Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium, pp. 51–62 (2001)
22. Mei, R.D.V.D., Hariharan, R., Reeser, P.K.: Web server performance modeling. Telecommun. Syst. **16**(3–4), 361–378 (2001)
23. Robb, D.: Up to capacity. In: Computerworld, no. Aug 29, pp. 24–26 (2005)
24. Robertsson, A., Kihl, M., Wittenmark, B.: Analysis and design of admission control in web-server systems. In Proceedings of American Control Conference (2003)
25. Robertsson, A., Wittenmark, B., Kihl, M., Andersson, M.: Admission control for web server systems—design and experimental evaluation. In: Proceedings of IEEE Conference on Decision and Control (CDC2004), pp. 531–536. Paradise Island, Bahamas (2004)
26. Salvador, P., Valadas, R., Pacheco, A.: Multiscale fitting procedure using Markov modulated poisson processes. Telecommun. Syst. **23**(1–2), 123–148 (2003)
27. Scott, S.L., Smyth, P.: The Markov modulated Poisson process and Markov Poisson cascade with applications to Web traffic modeling. In: Bayesian Statistics, vol. 7. Oxford University Press, London, UK (2003)
28. Stallings, W.: Data and Computer Communications, 6th edn. Prentice-Hall, Englewood Cliffs, NJ (2000)
29. Tsypkin, Y.Z.: Frequency criteria for the absolute stability of nonlinear sampled-data systems. Autom. Remote Control **25**(3), 261–267 (1964)
30. Voigt, T., Gunningberg, P.: Adaptive resource-based web server admission control. In: Proceedings of the 7th International Symposium on Computers and Communications (2002)
31. Wells, L., Christensen, S., Kristensen, L.M., Mortensen, K.H.: Simulation based performance analysis of web servers. In: Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM 2001), pp. 59–68. IEEE Computer Society, Los Alamitos, CA (2001)
32. Yoshihara, T., Kasahara, S., Takahashi, Y.: Practical time-scale fitting of self-similar traffic with Markov-modulated poisson process. Telecommun. Syst. **17**(1–2), 185–211 (2001)