Citation for the published paper:
M. Andersson, M. Kihl and A. Robertsson, 2003,
"Modelling and Design of Admission Control Mechanisms for Web
Servers using Non-linear Control Theory",
Performance and control of next-generation communication networks : [ITCom's Conference on Performance and Control of Next-Generation Communication Networks] ; 9 - 10 September 2003, Orlando, Florida, USA (SPIE proceedings series ; vol. 5244).
ISBN: 0-8194-5127-4. Publisher: The International Society for Optical
Engineering (SPIE).
http://dx.doi.org/10.1117/12.509281

Modelling and Design of Admission Control Mechanisms for Web Servers using Non-linear Control Theory

Mikael Andersson[a]*, Maria Kihl[a], Anders Robertsson[b]
[a)] Department of Communication Systems, Lund Institute of Technology
[b)] Department of Automatic Control, Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden

ABSTRACT

Web sites are exposed to high rates of incoming requests. Since web sites are sensitive to overload, admission control mechanisms are often implemented. The purpose of such a mechanism is to prevent requests from entering the web server during high loads. This paper presents how admission control mechanisms can be designed and implemented with a combination of queueing theory and control theory. Since web servers behave non-linear and stochastic, queueing theory can be used for web server modelling. However, there are no mathematical tools in queueing theory to use when designing admission control mechanisms. Instead, control theory contains the needed mathematical tools. By analysing queueing systems with control theoretic methods, good admission control mechanisms can be designed for web server systems. In this paper we model an Apache web server as a GI/G/1-system. Then, we use control theory to design a PI-controller, commonly used in automatic control, for the web server. In the paper we describe the design of the controller and also how it can be implemented in a real system. The controller has been implemented and tested together with the Apache web server. The server was placed in a laboratory network together with a traffic generator which was used to represent client requests. Measurements in the laboratory setup show how robust the implemented controller is, and how it correspond to the results from the theoretical analysis.

Keywords: Web servers, Apache, overload control, admission control, control theory, queueing theory

1. INTRODUCTION

Web sites on the Internet can be seen as server systems with one or more web servers processing incoming requests at a certain rate. The web servers have a waiting-queue where requests are queued while waiting for service. Therefore, a web server can be modelled as a queueing system including a server with finite or infinite queues.

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 occur rather seldom, it is not economical to overprovision the servers for these traffic peaks, instead admission control mechanisms can be implemented in the servers. The admission control mechanism rejects some requests whenever the arriving traffic is too high and thereby maintains an acceptable load in the system. The mechanism can either be static or dynamic. A static mechanism admits a predefined rate of requests whereas a dynamic mechanism contains a controller that, with periodic time intervals, calculates a new admission rate depending on some control objective.

The controller bases its decision from measurements of some control variable, for example the queue length, processor occupancy, or processing delays. The control objective is usually that the value of the control variable should be kept at a reference value. The choice of control variable is an important issue when developing an admission control scheme. First, the control variables must be easy to measure. Second, the control variable must in some way relate to the QoS demands that the users may have on the system. Traditionally, server utilization or queue lengths have been the variables mostly used in admission control schemes. For web servers, the main objective of the control scheme is to protect it from overload. As long as the average server utilization or queue length is below a certain level, the response times are low.

* mike@telecom.lth.se; http://www.telecom.lth.se/panda; phone +46462224962; fax+4646145823

One well-known controller in automatic control is the PID-controller, which enables a stable control for many types of systems (see, for example Åström[1]). The PID-controller uses three actions: one proportional, one integrating, and one derivative. In order to get the system to behave well it is necessary to decide proper control parameters. Therefore, before designing the PID-controller, the system must be analysed 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 analysed with linear control theoretic methods. However, a queueing system is both non-linear and stochastic. The main problem is that non-linear models are much harder to analyse with control theoretic methods.

Very few papers have investigated admission control mechanisms for server systems with control theoretic methods. Abdelzaher[2,3] modelled the web server as a static gain to find optimal controller parameters for a PI-controller. A scheduling algorithm for an Apache web server was designed using system identification methods and linear control theory by Lu et al[4]. Bhatti[5] developed a queue length control with priorities. By optimizing a reward function, a static control was found by Carlström[6]. An on-off load control mechanism regulating the admittance of client sessions was developed by Cherkasova[7]. Voigt[8] proposed a control mechanism that combines a load control for the CPU with a queue length control for the network interface. Bhoj[9] used an PI-controller in an admission control mechanism for a web server. However, no analysis is presented on how to design the controller parameters. Papers analysing queueing systems with control theoretic methods usually describe the system with linear deterministic models. Stidham Jr[10]. argues that deterministic models cannot be used when analysing queueing systems. Until now, no papers have designed admission control mechanisms for server systems using non-linear control theory.

In this paper we implement an admission control mechanism for the Apache[11] web server. Measurements in the laboratory setup show how robust the implemented controller is, and that it corresponds to the results from the theoretical analysis.

Section 2 describes a general admission control mechanism. Section 3 shows how this can be applied on a web server. In section 4, we describe a non-linear control theoretic model of an admission control mechanism for a web server. We describe the controller design in section 5, where examples of good and bad parameters are given. The control theoretic model is used to design and implement an admission control mechanism for the Apache web server. The measurements are shown in section 6, and section 7 concludes the work.

2. ADMISSION CONTROL MECHANISM

A good admission control mechanism improves the performance of a server system during overload by only admitting a certain amount of requests at a time into the system. Admission control mechanisms for server systems usually have the same structure and are based on the same type of rejection mechanisms.

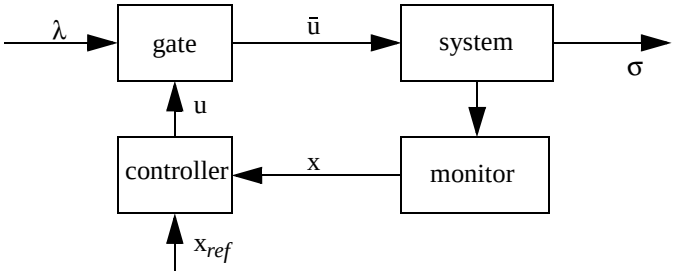

Figure 1. An admission control mechanism.

Figure 1 shows a general admission control mechanism that consists of three parts: a *gate*, a *controller*, and a *monitor*. The monitor measures a so called control variable, x. Using the control variable, the controller decides the rate, u, at which requests can be admitted to the system. The objective is to keep the value of the control variable as close as possible to a reference value, x_{ref}. The gate rejects those requests that cannot be admitted. The requests that are admitted proceed to the rest of the system. Since the admittance rate may never be larger than the arrival rate, λ, the actual admittance rate is

$\bar{u}$ =min[u,λ] requests per second. A survey of different admission control mechanisms for communication systems is given by Kihl[12].

2.1 Gate

Several gates have been proposed in the literature. One example is *Percent blocking*. In this mechanism, a certain fraction of the requests is admitted. Another example is *Token bucket*. Here, tokens are generated at a certain rate. An arriving request is admitted if there is a token available. The gate can also use a *Dynamic window* mechanism, that sets an upper limit to the number of requests that may be processed or waiting in the system the same time. The window size may be increased or decreased if the traffic conditions change.

2.2 Controllers

There are a variety of controllers to choose from when designing an admission control mechanism. Some of the most common controllers are the *Static controller*, the *Step controller*, and the *PID-controller*.

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. In this case, u_{fix} is given by

$$u_{fix} = \frac{\rho_{ref}}{\bar{x}}$$

Step controller. 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:

$$u(t+1) = \begin{cases} u(t) - s & y(t) > y_{ref} + \varepsilon \\ u(t) + s & y(t) > y_{ref} - \varepsilon \end{cases}$$

where the value of s decides how much the rate is increased/decreased and the value of ε decides how much the control variable may deviate from the reference value.

PID-controller. The PID-controller uses three actions: one proportional, one integrating, and one derivative. The control law in continuous time is as follows:

$$u(t) = K \cdot e(t) + \frac{K}{T_i} \cdot \int_0^t e(v)dv + K \cdot T_d \cdot \frac{d}{dt} \cdot e(t) \quad (1)$$

where $e(t)$ is the error between the control variable and the reference value, that is $e(t) = y_{ref} - y(t)$. The gain K, the integral time T_i, and the derivative time T_d are the controller parameters that are set so that the controlled system behaves as desired. 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. The derivative action improves the stability, however, in a system with a bursty arrival process the derivative action may cause problems. Therefore, the derivative action is usually either deleted (i.e. $T_d = 0$) or low pass filtered to remove the high frequencies.

3. INVESTIGATED SYSTEM

The system we have investigated in this work, is a web server with an admission control mechanism. The web server is Apache, described below. The web server is connected to the admission control according to Fig. 2, where the admission control runs as a stand-alone application, independent of Apache. Not all admission control mechanisms in the literature

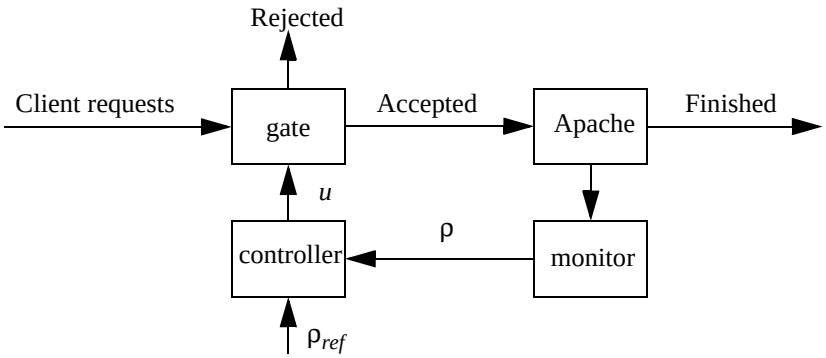

Figure 2. The Apache web server with admission control

are implemented like this. It is for example possible to have admission control within the kernel or within the Apache code. This architecture however, makes the system independent of web server.

3.1 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 script or Perl files. The communication between clients and server is based on HTTP[13]. A 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 a HTTP GET message to the server. The server then replies with a HTTP GET REPLY message. Finally, the TCP connection is closed by sending 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. Other types of web servers e.g. event-driven ones also exist (Voigt[14]).

3.2 Admission control

Since continuous control is not possible in computer systems, time is divided into control intervals of length h seconds. At the end of interval k, that is when the time is kh, the controller calculates the desired admittance rate for interval $[kh, kh+h]$, denoted $u(kh)$, from the measured average server utilization during the interval, $\rho(kh)$, and the reference value ρ_{ref}. There are three main parts in our admission control architecture:

Gate. The Gate thread runs a loop that accepts or rejects incoming requests on its own TCP port. It copies the requests to Apache's TCP socket, and then sends back the responses to the clients as the web server replies. 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. When a request is rejected, the TCP socket to the client is closed. Rejected requests are assumed to leave the system without retrials.

Monitor. The Monitor thread constantly 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 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.

Controller. The Controller is a PI-controller. The Controller is possible to turn on/off in order to be able to measure on an un-controlled system. The Controller's output is forwarded to the Gate thread. The Controller design is discussed more extensively in section 5.

4. 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) of the imput and the output of a system, see Fig. 3. In this case, the input to the system is the actual admittance rate, $\bar{u}$, whereas the output is either the server utilization, denoted ρ, or the number of jobs in the server, here denoted x.

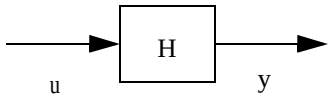

Figure 3. A system with transfer function *H*.

We use a discrete-time control theoretic model of web server developed by Kihl et al.[15]. We assume that the system may be modelled as a GI/G/1-system with an admission control mechanism. Kihl et al. showed that the queueing model is a good model for admission control purposes. The input to the system is the actual admittance rate, $\bar{u}$, whereas the output is the server utilization, ρ. 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 sufficiently long to guarantee that the "quantization" effects around the sampling times are small. The model is shown in Fig. 4. 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 $\bar{u}(kh)$ to be

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

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 $x(kh + h) \ge 0$. 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

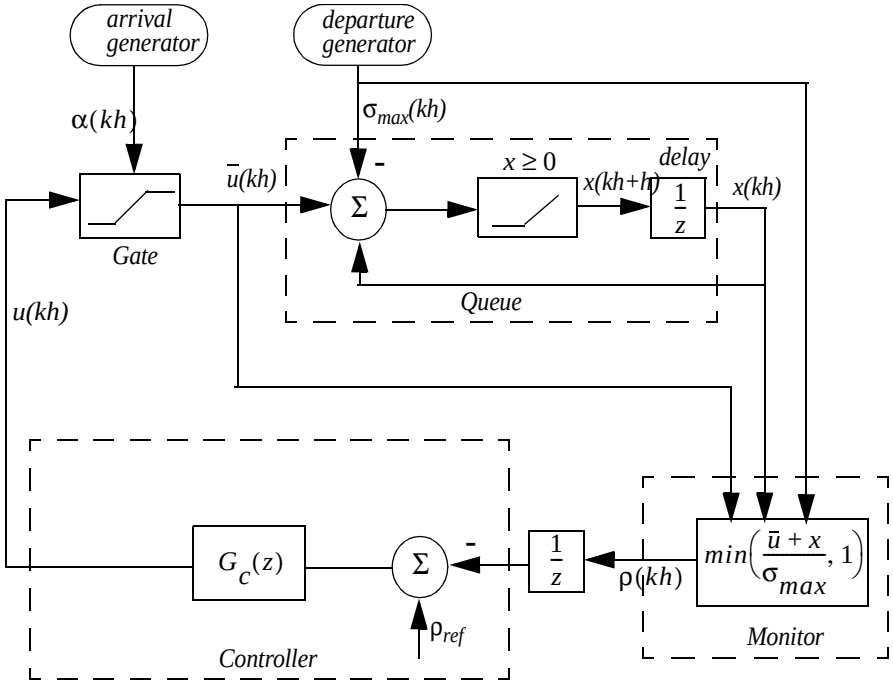

Figure 4. A control theoretic model of a GI/G/1-system with admission control.

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

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, $G_c(z)$.

5. CONTROLLER DESIGN

The PI-controller is commonly used in automatic control. The controller uses two actions: one proportional, and one integrating by using the following discrete-time control law (see Åström[1] for more details):

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

where $e(kh) = \rho_{ref} - \rho(kh)$. The gain, K, and the integral time, T_i, are the controller parameters that are set so that the controlled system behaves as desired. Since the controller is discrete, the controller parameter for the integration action, T_i, is given by $T_i = T_{si}/h$ where T_{si} would be the integral time in continuous-time. Note that the control signal, *u(kh)*, is allowed to become negative. A negative control signal will be treated as a zero signal in the gate.

Design and analysis. The control law for the PI-controller expressed in z-transform is given by

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

In this paper we use a linear design method, which means that we during the design analysis consider a deterministic system with no active saturations. However, it is important to note that we can take into account the saturations in the system during the design, since the underlying model is non-linear. This is performed by choosing the controller parameters carefully, since some controller parameters may cause oscillations in a system with saturations. This is the main difference between our work and the previous published work about control theoretic analysis of queueing systems. In those papers, the underlying system models are linear and deterministic, which means that the non-linearitities and stochastic processes in the real systems are ignored.

The linear transfer function from the desired utilization, ρ_{ref}, to the (delayed) output, $\rho(kh)$, will be

$$G_{cl} = \frac{G_c(1+G_q)G_m}{1+G_c(1+G_q)G_m} = \frac{(z+1)K(T_i z - T_i + h)}{z^3\sigma T_i + (KT_i - \sigma T_i)z^2 + Kzh + (-KT_i + Kh)} \quad (2)$$

where $G_c = K\left(1 + \frac{1}{T_i}\frac{h}{z-1}\right)$, $G_q = \frac{1}{z-1}$, $G_m = \frac{1}{\sigma}\cdot\frac{1}{z}$ are the transfer functions for the controller, for the queue, and for the monitor, respectively. σ is the average value of σ_{max}. The characteristic polynomial for the linear closed loop system will be

$$z\cdot\left(z^2 + \frac{K-2\sigma}{\sigma}z + \frac{-KT_i + Kh + \sigma T_i}{\sigma T_i}\right) \quad (3)$$

where 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

$$z\left(z^2 + a_1 z + a_2\right) = 0$$

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.

6. EXPERIMENTS

The admission control mechanism was implemented on a real web server. We tested the system by running tests on it and collecting performance metrics such as the server utilization distribution and step responses. The admission control mechanism was written in Java and tested on the Windows platform. We also compared the measurements with simulations. The queueing model was represented by a discrete-event simulation program implemented in C, and the control theoretic models were implemented with the Matlab Simulink package. The traffic generators in the discrete-time model were built as Matlab programs. They generate arrivals and departures according to the given statistical distributions.

6.1 Setup

Our measurements 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 1700 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 a HTTP load generator, which was a modified version of S-Client[16]. The S-Client is able to generate high request rates even with few client computers by aborting TCP connection attempts that take too long time. The original version of S-Client uses deterministic waiting times between requests. We modified the code to use Poissonian arrivals instead. The client program was programmed to request dynamically generated HTML files from the server. The CGI

script was written in Perl. It generates a random 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 Fig. 5. Apache was installed on the server and set to listen to port 8080. The Gate thread listened to port 80, the normal web server port, and then copied the admitted requests to port 8080. The admission control mechanism and the web server ran on the same computer. In all experiments, the control interval was set to one second.

6.2 Validation of the model

We have validated that the open system, that is without control feedback, is accurate in terms of average server utilization. The average server utilisation for varying arrival rates are shown in Fig. 5.

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 Fig. 5 gives an estimation of the average service time in the web server, $1/\mu = 0.0255$.

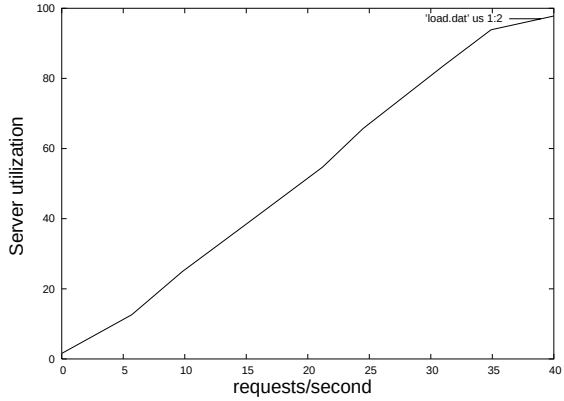

Figure 5. Average server utilization for the open system.

6.3 Controller parameters

Root locus arguments show that reasonable parameters give a stable closed loop system. Fig. 6 shows the root locus diagram when Ti =2.8 and the gain K varies between [0,40].

"Good" controller parameters. By choosing $\{K, T_i\}=\{20, 2.8\}$ the roots of the characteristic polynomial in eq. (3) will be rather well damped and the transients from the pole on the real axis will decay fast. This controller design can, therefore, be seen as a "good" design.

"Bad" controller parameters. If the controller parameters are chosen badly, the system will not behave well. One example is $\{K, T_i\}=\{20, 0.1\}$, which for the linear systems gives unstable poles placed outside the unit circle.

6.4 Performance metrics

An admission control mechanism have two control objectives. First, it should keep the control variable at a reference value, i.e. the error, $e=y_{ref}-y$, 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 test the mechanism in two ways. First, we show the steady-state distribution of the control variable, by plottting the estimated distribution function. The distribution function is estimated from measurements during 1000 seconds with the specific parameter setting. The distribution function shows how well the control mechanism meets the first control objective. Second, we plot the step response during 60 seconds when starting with an empty system. The step response shows the settling time for the control mechanism.

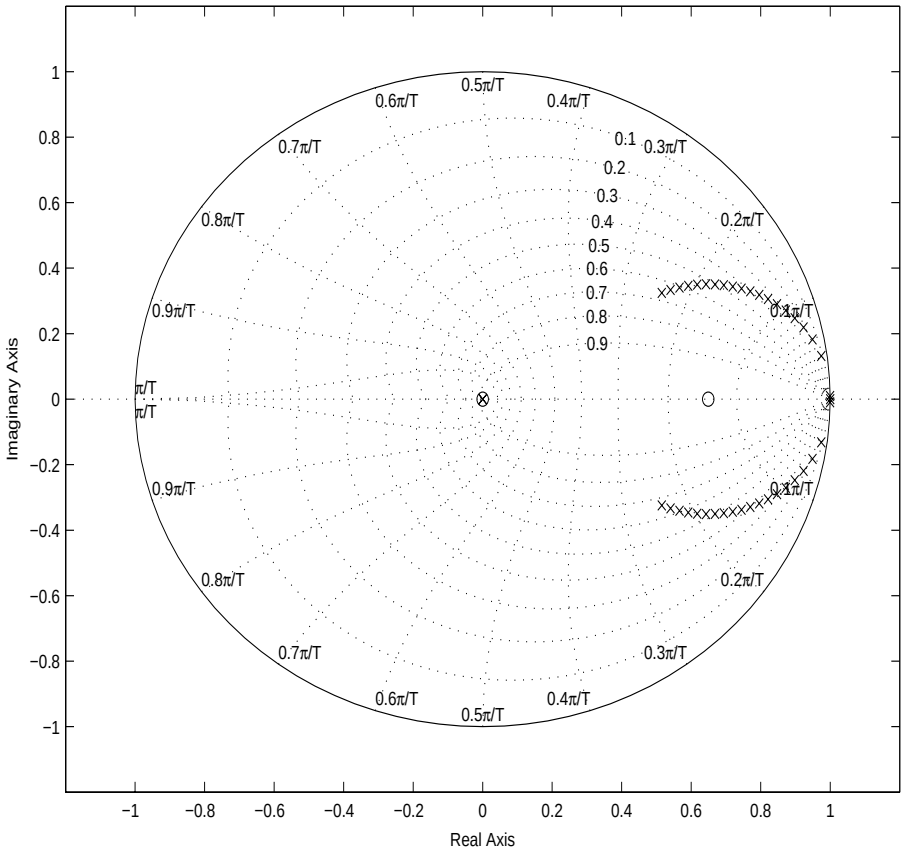

Figure 6. Root locus diagram for T_i=2.8. The gain K varies between [0,40].

6.5 Distribution function

Fig. 7 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 is one thereafter. In this case, the load was kept at 0.8, and the parameter setting, $\{K, T_i\}=\{20, 2.8\}$, 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/1system and the M/M/1system are given in Fig. 7, when using $\{K, T_i\}=\{20, 2.8\}$. They show that the system behaves as expected.

6.6 Step response

Fig. 8 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, $\{K, T_i\}=\{20, 2.8\}$, exhibits a short settling time with a relatively steady server utilization. The bad parameter setting, $\{K, T_i\}=\{20, 0.1\}$ has its poles outside the unit circle and behaves badly, the load oscillates and is never stable. Comparisons to M/D/1 and M/M/1 simulations, also in Fig. 8, show that the model is accurate.

6.7 Limitations with linear design

All papers published earlier about control theoretic analysis of queueing systems have only used linear deterministic models that can be analyzed with linear design methods. However, it is important to know that a linear model of a non-linear system is not accurate. The strength with our work is that we can try the controllers in the non-linear model, and thereby see how the real system behaves. For example, the controller parameters {5, 0.185} gives unstable poles outside the unit

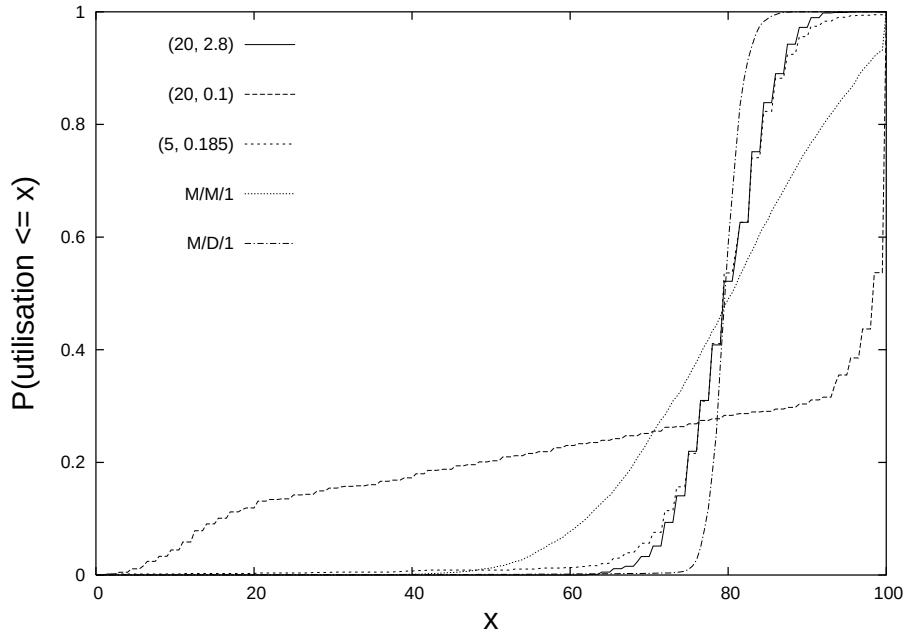

Figure 7. Server utilization distribution of measurements from the real system together with simulations of the M/M/1 and the M/D/1 system.

circle. Therefore, they should not be used. However, the non-linear system behaves very well, as shown in Fig. 7. We have found that this is due to the non-linearity in the queue, since the queue length can never be negative.

7. CONCLUSIONS

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. The main problem here is that queueing systems are non-linear and stochastic, which means that they are difficult to model and analyse with control theoretic methods.

The main objective with our work has been to find models and methods from control theory that can be used when designing admission control mechanisms for server systems. In this paper, we have designed admission control mechanisms for this system with control theoretic methods. We have shown that the model is accurate enough for this purpose, and have also shown an example of how to perform the controller design. We have used the PI-controller, commonly used in automatic control. The admission control mechanism has been implemented on a real web server, running the well-known Apache web server software. Measurements were made to examine the performance of the system. The experiments show that the control mechanism behaves as expected.

The main conclusion of this paper is that it is possible to use control theoretic methods when designing admission control mechanisms for server systems. However, linear deterministic models, as have been used in previous papers, are not enough for this purpose. A server system is both non-linear and stochastic, and if this fact is ignored during modelling

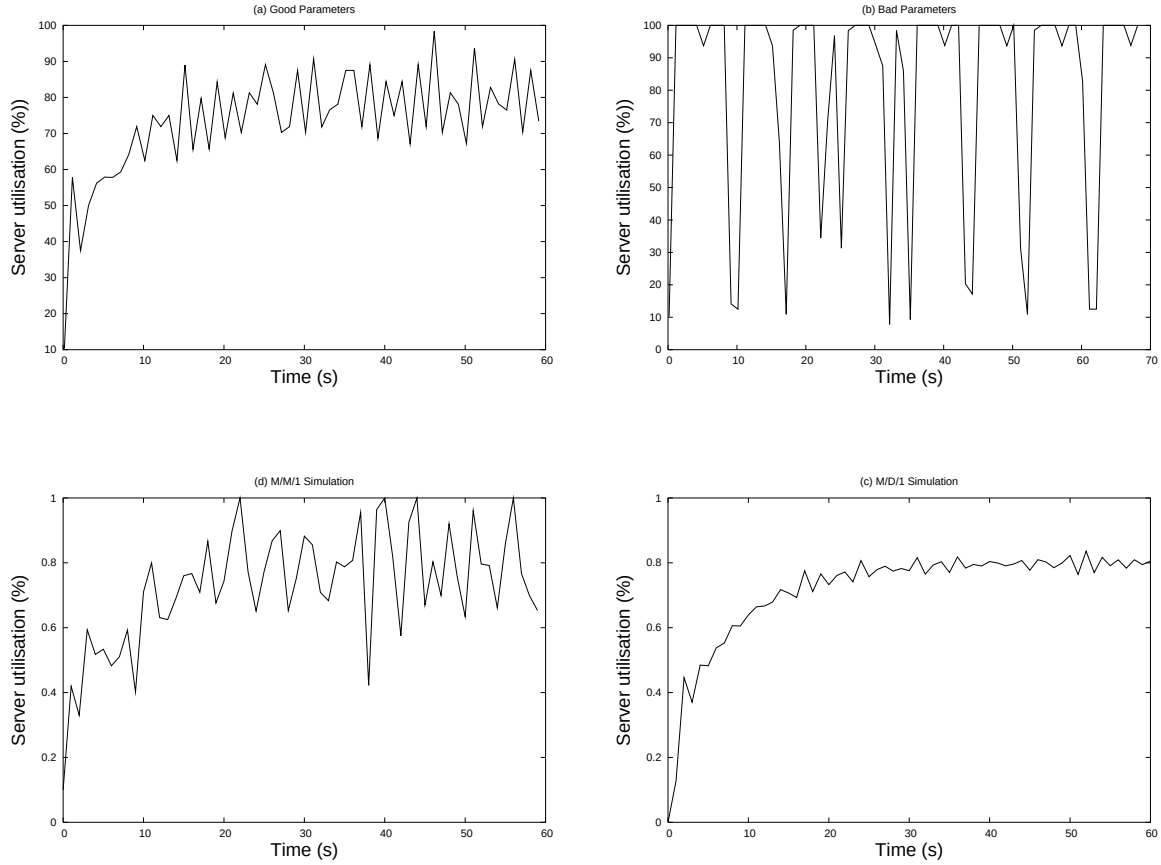

Figure 8. (a) Example of a realisation with good parameters. (b) Example of a realisation with bad parameters. (c) Simulation of M/D/1-system with good parameters. (d) Simulation of M/M/1-system with good parameters.

and analysis, the behaviour of the real system may not be as expected. It is obvious that more research is needed in this field. We will, therefore, continue investigating non-linear, stochastic control theoretic models of queueing systems.

ACKNOWLEDGMENTS

This work has partially been supported by the Swedish Research Council through the Multi Project Grant 621-2001-3020 and contract 621-2001-3053.

REFERENCES

1. K.J. Åström and B. Wittenmark, *Computer-controlled systems, theory and design*, Prentice Hall International Editions, 3rd Edition, 1997.
2. 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.
3. 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.
4. C. Lu, T.F. Abdelzaher, J.A. Stankovic and S.H. So, “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.

5. N. Bhatti, R. Friedrich, Web server support for tiered services, in; IEEE Network, Sept/Oct 1999, pp. 64-71.
6. J. Carlström, R. Rom, Application-aware admission control and scheduling in web servers, in; Proc. Infocom, 2002.
7. L. Cherkasova, P. Phaal, Predictive admission control strategy for overloaded commerical web servers, in; Proc. 8th International IEEE Symposium on modeling, analysis and simulation of computer and telecommunication systems, 2000, pp. 500-507.
8. T. Voigt, P. Gunningberg, Adaptive resource-based web server admission control, in; Proc. 7th International Symposium on Computers and Communications, IEEE Computer Society, 2002.
9. P. Bhoj, S. Ramanathan and S. Singhal, "Web2K: Bringing QoS to web servers", HP Labs Technical report, HPL-2000-61, 2000.
10. S. Stidham Jr., "Optimal control of admission to a queueing system", *IEEE Transactions on Automatic Control*, **Vol. 30**, No. 8, Aug 1985, pp 705-713.
11. "Apache web server", http://www.apache.org
12. M. Kihl, "Overload control strategies for distributed communication networks", Ph.D thesis, Dep. of Communication Systems, Lund Institute of Technology, Sweden, 1999.
13. W. Stallings, *Data & Computer Communications*, Prentice Hall, 2000, Sixth Edition.
14. T. Voigt, "Overload behaviour and protection of event-driven web servers", In Proceedings of the International Workshop on Web Engineering, May 2002, Pisa, Italy.
15. M. Kihl, A. Robertsson, B. Wittenmark, Performance Modelling and Control of Server Systems using Non-linear Control Theory, 18th International Teletraffic Congress, 2003.
16. G. Banga and P. Druschel, "Measuring the capacity of a web server", in USENIX Symposium on Internet Technologies and Systems, December 1997, pp. 61-71.