ADMISSION CONTROL WITH SERVICE LEVEL AGREEMENTS FOR A WEB SERVER

Mikael Andersson, Jianhua Cao, Maria Kihl and Christian Nyberg
Department of Communication Systems
Lund Institute of Technology
email: mike,jcao,maria,cn@telecom.lth.se

ABSTRACT
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. This paper investigates two overload control strategies with performance bounds for a web server. In service level agreements, we bound average response times and throughputs for all service classes. Each request is sorted into a class, where each class is assigned a weight representing the income for the web site owner. Then a linear optimization algorithm is applied so that the total revenue for the web site during overload is maximized.

KEY WORDS
Electronic commerce, Optimization Techniques, Protection and security

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 queue. 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. The admission control mechanism rejects some requests whenever the arriving traffic is too high and thereby maintains an acceptable load in the system. In this paper we study two admission control strategies based on percent blocking where we assume that the traffic rate are known.

Other papers have been presented in this area of research. Chen et al. describes in [1] an admission control scheme that divides requests into classes and then tries to guarantee a maximum response time for prioritized classes. Lee et al. describe a similar admission control scheme in [2]. Zhang et al. [3] develop a profit-aware QoS policy for web servers, where each request generates a certain profit to the site owner depending on the response time. Kanodia and Knightly propose an admission control scheme without profit optimization where requests are given priorities and response time limits called Latency-Targeted Multiclass Admission Control (LMAC) [4]. Also, control theoretic methods have been applied to web servers, see e.g. [5].

In this paper we use a web server model that consists of a processor sharing node with a queue attached to it. A more thorough investigation of the model can be found in previous works, [6] and [7], by Cao et al. From the studies by Kihl and Widell [8] and Menasce et al. [9], we have introduced a set of classes that each request to a typical E-commerce site can be sorted into. The classes have different attributes such as revenue, throughput, service time requirements. Together with the class definition, we set up a service level agreement that the E-commerce site should uphold. The service level agreement regulate the throughput for each class as well as the maximum allowed average response time.

Requests can be rejected in numerous ways, but since the requests generate different revenues for the site owner, it is a good idea to optimize the total profit during overload. Two different control strategies are therefore investigated that optimizes the total profit. One controller acknowledges the request's class attribute whereas the other one disregards the class attribute. The latter one is introduced as a comparison to the class dependent controller. Their performance regarding the ability to hold the service level agreement and the generated profit during overload is studied. One important thing to consider is that a rejection requires processing. It is not enough to simply disconnect the client, instead some ”rejection page” should be sent. The rejection action therefore costs about the same amount of work as a small static web page. This is included in our model and the simulations show that this affects the total profit in an overloaded server. Also, the connection setup processing is considered. Before any rejections can be performed, the web server must set up a connection to see what kind of request that is coming. This connection setup processing has to be performed for all requests, not only the admitted ones.

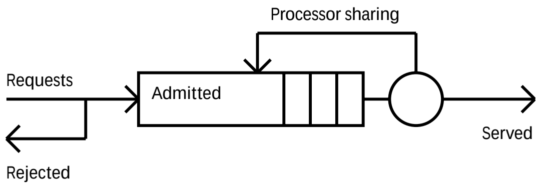

Figure 1. The web server model.

The rest of the paper is organized as follows; Section 2 describes the web server model we use and explains the concept of classes and the service level agreement. Section 4 defines the admission control problem that can be formulated as two alternate linear programming problems. Section 5 shows simulations that compare the two methods investigated in this paper. Section 6 discusses the results while the last section concludes the work.

2 Preliminaries

We describe the web server model that we use and we also define classes in a commercial web site context.

2.1 Classes

It is natural to define a set of request classes when it comes to a server system like the web server. The type of classes that are considered depends on the web site. In this work, as will be shown, we have chosen to adopt and extend the request types found in the works of Kihl and Widell [8] and Menasce et al. [9]. In [8] they investigate admission control strategies for commercial web sites using different types of requests, for example Buy, Browse and Pay requests. The request types correspond to the different stages that a visitor to the site goes through in a typical session.

2.2 Model description

In [6] and [7] we show how a web server can be modeled as a single server queue with a processor sharing discipline. The queue length is restricted to a certain number of jobs. The model used here is similar (Figure 1).

The difference is that in this work, there is no maximum number of threads in the web server. The server serves N classes of requests. The arrival processes of all classes are assumed to be Poisson. The arrival rate for the customers of class i is λ_i. The mean service requirement for the customers of class i is v_i besides the connection setup time, denoted as v_{init}. The connection setup time is the same for all classes. For each received request a TCP connection has to be set up. To be able to determine what class the request belongs to, the HTTP header must be parsed in the HTTP layer. The total arrival rate of all classes is therefore

$$\Lambda = \sum_{i=1}^{N} \lambda_i \tag{1}$$

Since the service discipline is processor sharing, the actual service time distribution can be neglected. Let the customer of class $(N+1)$ represent the ”rejection service”. In some papers dealing with admission control, the rejection service is neglected. Instead, some sort of message should be sent to the rejected visitor that notifies about the rejection. The rejection service required must be less than the originally requested service to be of any practical use. We assume that a rejection requires v_{rej} amount of service and

$$v_{rej} \leq min_i(v_i), (i = 1..N) \tag{2}$$

For requests of class i, the probability that the request will be served normally, that is without rejection is denoted x_i. If the request is rejected, it will become a request of class $(N+1)$. This gives

$$\lambda_{N+1} = \sum_{i=1}^{N} (1 - x_i)\lambda_i \tag{3}$$

3 Admission Control

We will investigate two admission controllers based on percent blocking:

CAC-CI, Contract-based Admission Control - Class Independent: The customers are accepted with probability $x_i = x$, disregarding their class identity.

CAC-CD, Contract-based Admission Control - Class Dependent: The customers of class i are accepted based on their class identity, with probability x_i. For the CAC-CI controller the server utilization is

$$\rho = \sum_{i=1}^{N} \lambda_i \cdot (v_{init} + x \cdot v_i + (1 - x) \cdot v_{rej}) \tag{4}$$

and for the CAC-CD controller

$$\rho = \sum_{i=1}^{N} \lambda_i \cdot (v_{init} + x \cdot v_i + (1 - x_i) \cdot v_{rej}) \tag{5}$$

It follows from the model that the average response time for served customers of class i is

$$w_i = \frac{v_{init} + v_i}{1 - \rho} \tag{6}$$

where ρ is the server utilization and thus depends on the type of admission control in question.

The purpose of admission control is to guarantee that the served customers enjoy reasonable service times. Let

τ_i be the upper bound of the average response time for customers of class i. We want

$$w_i \leq \tau_i, \ \forall i = 1, .., N. \tag{7}$$

We require that for customers of class i, the minimum acceptance probability must be α_i. For the CAC-CI controller, this means

$$max \ \alpha_i \leq x \leq 1 \tag{8}$$

and for type CAC-CD admission control

$$\alpha_i \leq x_i \leq 1 \tag{9}$$

Now, given $\tau_1, \tau_2, ...\tau_N$ and $\alpha_1, \alpha_2, ...\alpha_N$ we define the so called *service level agreement* to be

$$S = (\{\tau_i\}_{i=1}^N, \{\alpha_i\}_{i=1}^N). \tag{10}$$

The service level agreement is considered broken if one or more of its constraints are violated. For example, if the response time for a certain class i exceeds τ_i or less than α_i requests gets served, the service level agreement is broken.

Usually there are infinitely many admission control policies that satisfies the service level agreement.

Let γ_i be the revenue (potential or real) for serving a class i customer. We can therefore restrict our attention to those policies that maximize the reward. Since the CAC-CI controller accepts customers regardless of class identity, this is equivalent to maximize the throughput i.e.

$$\max x \cdot \sum_{i=1}^N \gamma_i \cdot \lambda_i \tag{11}$$

For type CAC-CD, it is slightly more complicated but still, the objective function is linear,

$$\max \sum_{i=1}^N \gamma_i \cdot \lambda_i \cdot x_i \tag{12}$$

To summarize we give a list of parameters and variables in Table 1.

4 Linear programming formulations

Since both of the objective functions for the controllers are linear, it is now feasible to set up linear programming formulations. For the CAC-CI and CAC-CD controllers they can be formulated as follows:

Table 1. Parameter list

Variable	Description
N	number of customer classes
i,j=1..N	indices of the customer class
λ_i	arrival rate for class i
v_i	average service requirement for class i
γ_i	revenue for request of class i
$v_{init,rej}$	service requirements for setup and rejection
α_i	acceptance rate guarantee
τ_i	mean service time guarantee
ρ	server utilization

CAC-CI: maximize

$$x \cdot \sum_{i=1}^N \gamma_i \cdot \lambda_i$$

subject to (for all i = 1..N)

$$v_{init} + v_i \leq \tau_i(1 - \sum_{j=1}^N \lambda_j(v_{init} + xv_j + (1-x)v_{rej}) \tag{13}$$

$$\max_i \ \alpha_i \leq x \leq 1 \tag{14}$$

CAC-CD: maximize

$$\sum_{i=1}^N \gamma_i \cdot \lambda_i \cdot x_i$$

subject to (for all i = 1..N)

$$v_{init} + v_i \leq \tau_i(1 - \sum_{j=1}^N \lambda_j(v_{init} + x_j v_j + (1-x)v_{rej}) \tag{15}$$

$$\max_i \ \alpha_i \leq x \leq 1 \tag{16}$$

CAC-CI. The CAC-CI problem can be solved explicitly as follows:

From 13, we have

$$x \leq \frac{1 - (v_{init} + v_{rej}) \sum_{j=1}^N \lambda_j - \frac{v_{init}+v_i}{\tau_i}}{\sum_{j=1}^N \lambda_j(v_j - v_{rej})} = l_i$$

Let K:= arg $max_j \frac{v_{init}+v_j}{\tau_j}$. Clearly the CAC-CI problem has a solution

$$\max x_i \ \leq l_K.$$

If the condition above is satisfied the optimal solution for the CAC-CI problem is

$$x = l_K.$$

CAC-CD. The CAC-CD problem can be solved by any linear programming solver, e.g. CPLEX [10] quite easily.

We will examine the necessary and sufficient conditions for the existence of a solution.

By the definition of w_i and 7, we have

$$\rho \leq 1 - \frac{v_{init} + v_i}{\tau_i} \ \forall i = 1, .., N$$

On the other hand, we can achieve the smallest server utilization when we reject all customers and still fulfill the service level agreement:

Let K:= arg $max_j \frac{v_{init}+v_j}{\tau_j}$. Hence the problem has a solution

$$\sum_{j=1}^{N} \lambda_j (v_{init} + \alpha_j + (1-\alpha_j) v_{rej}) \leq 1 - \frac{v_{init} + v_K}{\tau_K}$$

Let $\rho_i := \lambda_i / \Lambda$. The condition above implies

$$\Lambda \leq \frac{1 - \frac{v_{init}+v_j}{\tau_j}}{\sum_{j=1}^{N} \rho_j (v_{init} + \alpha_j + (1-\alpha_j) v_{rej})}$$

The similar condition for the CAC-CI controller is

$$\Lambda \leq \frac{1 - \frac{v_{init}+v_j}{\tau_j}}{\sum_{j=1}^{N} \rho_j (v_{init} + \alpha_s + (1-\alpha_j) v_{rej})}$$

where $\alpha_s = max_i \ \alpha_i$.

5 Experiments

In all simulations we set a total arrival rate, Λ. The arrival rates for the different classes were then determined by the request type distribution derived from [8] where the ratio of ”leaving customers” were ignored. The distribution originally comes from the work of Menasce et al. [9], where the occasional buyer on a web site is studied. The buyer's requests are categorized into the classes shown in Table 2 and then the rates of each class are determined. The service times for each class has been taken from the simulation values in [8].

In all experiments, the work required in the connection setup phase was set to 0.005 seconds. The rejection work was also set to 0.005 seconds. The queueing model and the admission control algorithms were implemented as a discrete event simulation program in Java. Two sets of simulations were performed; one set where the CAC-CI controller was used, and one where the CAC-CD controller was used. Both methods were evaluated for their ability to enforce the service level agreement. In all simulations, the total arrival rate was increased from 10 to 60 requests per second, in steps of 5 requests per second. Table 2 shows the simulation configuration for both controllers with required service times v_i, distribution d_i, request revenue γ_i, acceptance rate guarantee α_i and mean service time guarantee τ_i for classes 1 to 5. For the CAC-CD controller optimization of the linear programming formulation was performed by using the Java version of lpsolver [11].

Table 2. Class Parameters

	Description	v_i	d_i	γ_i	α_i	τ_i
1	Browse	0.015	0.41	1	0.2	1.5
2	Search	0.030	0.40	1	0.4	3.0
3	Select	0.015	0.17	1	0.6	1.5
4	Add	0.015	0.014	5	0.8	1.5
5	Pay	0.035	0.006	10	1.0	3.0

The simulations evaluated the controllers in terms of response times, throughput counted as admitted requests and the total profit generated at each arrival rate.

6 Results and Discussion

Figure 2 shows the response time for each class as a function of the total arrival rate. It can be seen that the CAC-CD is capable of keeping the agreed response time limits whereas the CAC-CI controller cannot at higher arrival rates.

Figure 3 shows the throughput as a percentage, of completed requests for each class. The solid lines in the diagrams represent the agreed minimum service level. When it comes to throughput, each class receives the contracted amount of throughput with the CAC-CD controller. The CAC-CI breaks the contract at higher rates for classes 3, 4 and 5.

The two methods were compared from a profit perspective in Figure 4. The figure shows the profit per second versus arrival total rate. As can be seen for the CAC-CD controller, requests from classes 4 and 5 are more likely to be admitted at the expense of requests in class 1, 2 and 3 in higher arrival rates. The reason is that higher individual request revenue is generated in class 4 and 5. It may seem strange that the total profit decreases after its peak at $\lambda = 40$. The decline of total profit when the aggregrated traffic rate reaches a certain limit is mainly due to that the server is busy with rejection most of the time in that traffic rate region. This implies that the server should be properly dimensioned in order to achieve the best performance, i.e. maximum profit, when the service of rejection cannot be neglected. For CAC-CI however, the total profit is lower at higher arrival rates. The controllers behave the same, profit-wise, up until $\lambda = 35$, after which the CAC-CD yields more total profit.

7 Conclusions

We have presented and compared two admission control strategies for a web server. The CAC-CI controller disregards the request's class property resulting in inferior per-

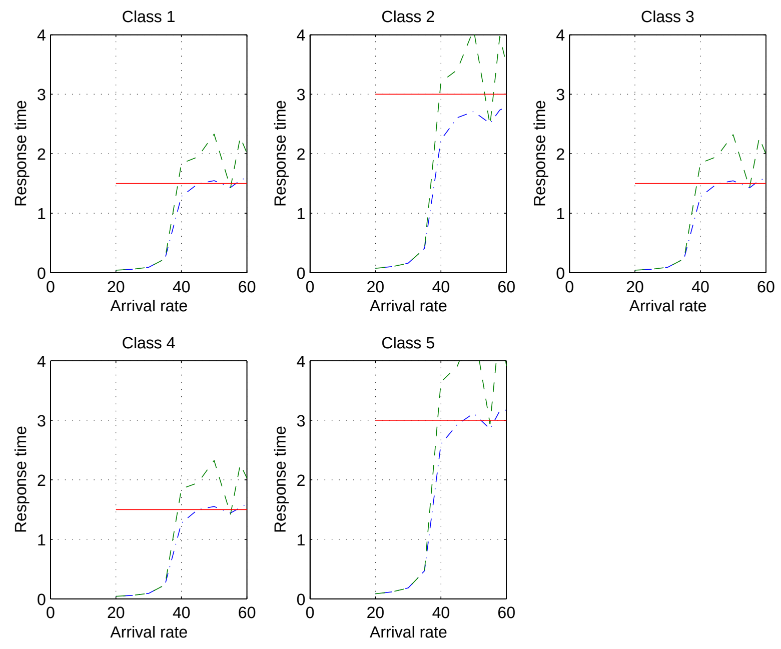

Figure 2. Response times per class. Dashed lines are CAC-CI, dash-dotted are CAC-CD. Straight lines represent maximum response times.

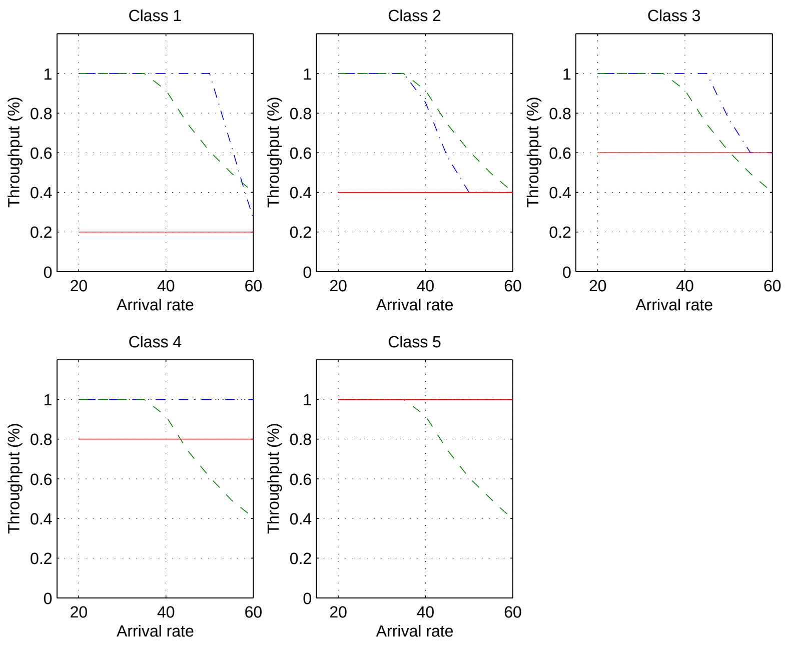

Figure 3. Throughput per class. Dashed lines are CAC-CI, dash-dotted are CAC-CD. Straight lines represent minimum throughput.

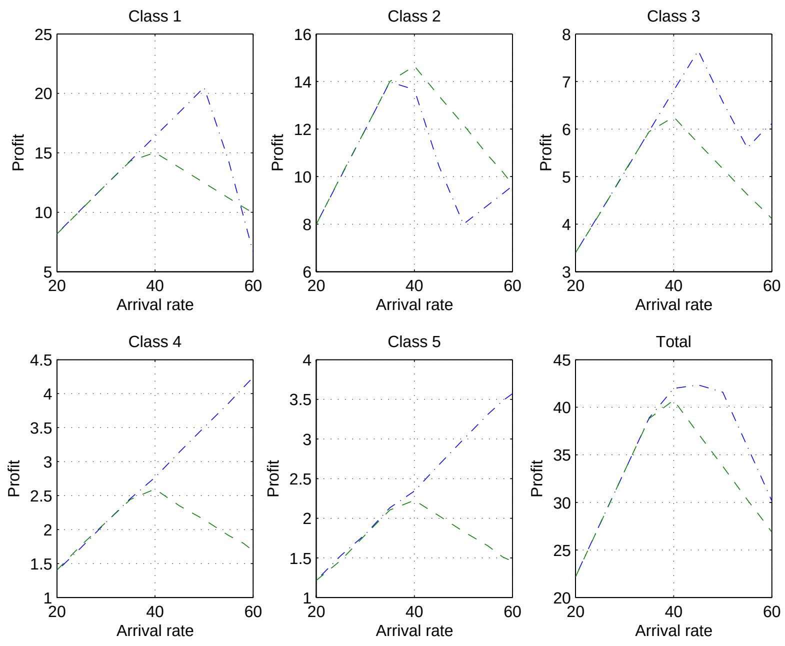

Figure 4. Profit per class. Dashed lines are CAC-CI, dash-dotted are CAC-CD.

formance compared to the CAC-CD controller that does regard class property. Both controllers optimize the total profit given the constraints given in the service level agreement concerning response times and throughput. The fact that each request is associated with an initialization work (even for rejected requests) and that rejections also cost in terms of processing power is considered.

Acknowledgments

The work has been supported by The Swedish Research Council under contract no. 621-2001-3053 and also by the Swedish Emergency Management Agency.

References

[1] X. Chen, H. Chen, and P. Mohapatra, "Aces: An efficient admission control scheme for qos-aware web servers," *Computer Communications*, no. 26, p. 1581, 2003.

[2] S. C. Lee, J. C. Lui, and D. K. Yau, "A proportional-delay diffserv-enabled web server: Admission control and dynamic adaptation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, 2004.

[3] Q. Zhang, E. Smirni, and G. Ciardo, "Profit-driven service differentiation in transient environments," in *In Proceedings of MASCOTS*, 2003, p. 230.

[4] V. Kanodia and E. W. Knightly, "Ensuring latency targets in multiclass web servers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 1, 2003.

[5] M. Andersson, "Performance modeling and control of web servers," Department of Communication Systems, Lund Institute of Technology, Tech. Rep. 160, 2004, lic. Thesis.

[6] J. Cao, M. Andersson, C. Nyberg, and M. Kihl, "Web server performance modeling using an m/g/1/k*ps queue," in *In Proceedings of International Conference on Telecommunication (ICT)*, 2003.

[7] M. Andersson, J. Cao, M. Kihl, and C. Nyberg, "Performance modeling of an apache web server with bursty arrival traffic," in *In Proceedings of International Conference on Internet Computing (IC)*, 2003.

[8] N. Widell and M. Kihl, "Admission control schemes guaranteeing customer qos in commercial web sites," in *In Proceedings of IFIP and IEEE Conference on Network Control and Engineering (NETCON)*, 2002.

[9] D. Menasce, V. Almeida, R. Fonseca, and M. Mendes, "Business-oriented resource management policies for e-commerce servers," *Performance Evaluation*, vol. 42, 2000.

[10] "Cplex," http://www.ilog.com/products/cplex/.

[11] "Linear programming solver, washington university in saint louis," http://www.cs.wustl.edu/ javagrp/help/LinearProgramming.html.