Application-aware Admission Control and Scheduling in Web Servers

Jakob Carlström, Raphael Rom

***Abstract*— This paper presents an architecture and algorithms for optimizing the performance of web services. For a given service, session-based admission control is combined with stage-wise request queuing, where the stages represent sub-tasks within sessions. The scheduling of requests is governed by generalized processor sharing. We present a performance model, relying on online estimation of parameters describing client-server interaction. A reward function corresponding to the service provider's objective is maximized using techniques for nonlinear optimization. In a case study, we model and optimize the resource sharing at a web server hosting an electronic store. The performance advantages of our approach are quantified numerically, and the robustness to parameter estimation errors is assessed by sensitivity analysis.**

***Index Terms*— Web server scheduling, admission control, QoS, reward maximization, generalized processor sharing.**

I. Introduction

For service providers on the Internet, high availability to the users is crucial. For a service to be available, it is not sufficient that the web server eventually responds to HTTP requests. The response time is also important. Selvridge et al. [1] found that long delays increase user frustration, and decrease task success and efficiency. In another study, Bhatti et al. [2] let users rate the Quality of Service (QoS) of an e-commerce application in experiments where the delay was varied. Among the conclusions were:

- Users tolerate different levels of delay for different tasks. For example, the tolerance was higher for viewing the content of the shopping cart than for viewing the product catalog. Further, the tolerance decreases with the length of the session.
- The tolerance for delay depends on how a page is loaded. Specifically, users accept longer delay if content is presented incrementally, as opposed to presenting all the page once its entire content has been loaded.
- With non-incremental loading, users rated the quality as "high" for delays ranging from 0 through 5 seconds, "average" in the interval 5 through 11 seconds and "low" for delays longer than 11 seconds.
- Users who experience long delays are likely to abort active sessions prematurely.

The above findings suggest that users' reactions to response delay should be integrated in the design of web services. This paper addresses the problem by proposing a scheme for scheduling of client requests and session-level admission control. The scheduling makes use of the generalized processor sharing (GPS) discipline. We focus on high-load situations, where the utilization of a bottleneck resource is close to the capacity limit. The key idea is to make the scheduling and admission control algorithms aware of the structure of the web service. After estimating parameters describing the dynamics of client-server interaction, request scheduling and session admission control can be optimized.

The paper is organized as follows: Section II provides background on web server control and GPS. Section III outlines the system architecture and provides our mathematical model of the web server and its load. In section IV, the rules for scheduling and admission control are derived. Section V describes the system configuration used in the numerical case studies conducted in section VI. Finally, section VII concludes the paper.

Jakob Carlström participated in this work while he was with the Electrical Engineering Department of the Technion, Haifa, Israel. He is currently with Xelerated, Stockholm, Sweden. Email: jakob.carlstrom@xelerated.com. Raphael Rom is with the Electrical Engineering Department of the Technion, and with Sun Microsystems, Inc., Santa Clara, CA. E-mail: raphael.rom@ieee.org

II. Background

A. Resource Control in Web Servers

A web server is forced to delay the response to a request when some resource necessary for processing the request is busy. In [3], three possible bottleneck resources were identified: HTTP protocol processing, reading data from disk and transmission of data on the network downlink. In the experimental setup of that study, the network was found to be the main bottleneck.

Web servers normally use either thread or process concurrency to handle concurrent requests. With process concurrency, session scheduling is done by the operating system, out of reach from the application. With thread concurrency, session scheduling can be done by the application. In [3], an experimental web server making use of one thread per connection was used to study effects of different scheduling policies. In this work, we assume that it is possible to implement the scheduling policy in the web server, for example based on thread concurrency.

Recent work on web QoS has proposed session-based admission control as a mechanism for preventing overload and limiting response delay [4], [5]. By rejecting requests if the measured load exceeds a threshold, admission control can prevent the server from entering a regime where delay is excessive or where dropped requests result in aborted sessions. Although the HTTP protocol is not session-oriented, sessions can be classified using cookies, client IP addresses or other mechanisms.

Cherkasova and Phaal [4] showed in simulations that an overloaded web server can experience a severe relative loss of throughput measured as the number of completed sessions per second, compared to the relative loss of requests per second.

They also found that web servers are unfair to long sessions, and proposed a measurement-based algorithm for session admission control that was shown to prevent overload and provide fairness with respect to session length.

Kanodia and Knightly [5] combined session-based admission control with service differentiation, devising a server architecture having one request queue per service class. In simulations, they showed that this enables their algorithm for admission control and scheduling to limit delay and achieve service separation

B. Generalized Processor Sharing

A GPS server serving K sessions is characterized by positive, real-valued weights $\{\phi_1, \ldots, \phi_K\}$ [6]. Let $W_i(t_1, t_2)$ be the amount of session-i work in the interval (t_1, t_2), and $W(t_1, t_2)$ be the total amount of service provided by the server during the same period. A *work-conserving* GPS server is defined as one for which

$$\frac{W_i(t_1, t_2)}{\phi_i} = \frac{W(t_1, t_2)}{\sum_{j \in B(t_2)} \phi_j}, \ \forall i \in B(t_2) \qquad (1)$$

holds for any interval $(t_1, t_2]$ during which $B(t_2)$, the set of backlogged sessions at time t_2 , does not change.

The fraction $\frac{W_i(t_1,t_2)}{\phi_i}$ is called the *normalized service* offered to session i during he interval (t_1, t_2). From equation (1) it follows that the GPS provides the same amount of normalized service to any session backlogged during the interval $(t_1, t_2]$.

Moreover, because the maximum number of sessions is limited by K, the minimum amount of service that a session receives is limited by

$$W_i(t_1, t_2) \geq \frac{\phi_i}{\sum_{j=1}^{K} \phi_j} W(t_1, t_2) \qquad (2)$$

for any time period (t_1, t_2) during which session i is continuously backlogged, regardless the behavior of other sessions. This property can be used to guarantee minimum service rates in computer and communications systems. In this work, however, we focus on the mean service rates.

GPS is an ideal server, assumed to be capable of serving all backlogged sessions simultaneously and requiring that the traffic is infinitely divisible. In more realistic systems, only one session can receive service at a time. There are several ways of emulating GPS service, for example [7], [8] and [9]. As this paper does not present any implementation, no particular algorithm for emulating GPS is chosen.

III. System Architecture and Model

This section describes the architecture of the admission control and scheduling blocks of an application-aware reward-maximizing web server, and provides some implementation aspects. Further, we introduce the mathematical notation and the model of the dynamics of the web server.

A. System architecture

When the initial request of a new session is processed by the server, an admission control algorithm either accepts or rejects the session. The admission controller is rate-based, limiting the intensity by which new sessions are admitted. If the initial request is admitted, all subsequent requests within the same session will also be admitted. A session classifier, which for example can be based on cookies or client IP addresses, maintains a table of active sessions. The termination of sessions can be detected using time-outs, or from users logging out.

Once established, a session resides in a fixed set of stages. An example of such a set in an e-commerce application is {`Welcome`, `Browsing-Empty_Cart`, `Browsing-Articles_in_Cart`, `Checkout`}. The clustering of requests into stages is further discussed below. After a request has passed admission control, a stage classification step follows.

Following stage classification, the request may have to wait in a queue before receiving service. We propose *stage-wise* queueing; that is, requests are classified with respect to the requested stage, and enter a stage-specific FIFO queue. When the critical resource is freed, a scheduling algorithm dequeues and grants service to one of the requests that is at the head of its queue. The scheduling algorithm is assumed to emulate GPS. Figure 1 summarizes the system architecture.

Making the web server aware of application-layer stages means that the server must be reconfigured with respect to the specific application. However, this procedure can be done semi-automatically, and needs to be invoked only when a service is launched or significantly modified.

For use by the optimization algorithm, the server must collect statistics about client-server interaction. Such statistics include page sizes, arrival intensity of requests for new sessions, counts of the number of requests per stage, and counts of how many times a stage was requested following a request for another stage. Tools for this are commonly available.

B. Server dynamics

After a page (belonging to a stage) has been downloaded, an idle period follows while the user processes the page. Then the user may make a new request. After a possible delay, the server produces and returns the new web page. At this point, with a probability that depends on the delay in server response, the user terminates the session. If not, a new (or the same) stage is entered, and so forth.

In e-commerce applications, the user may pay at transitions from certain stages to other. For services not associated with any payment, rewards that reflect the objective of the server provider may be associated with transitions between stages. For instance, the web-based help-desk of a computer manufacturer's may have a unity reward associated with every downloaded help document.

It is not our goal to model the details of the internal processing of a web server. Instead, we observe that some bottleneck resource limits the amount of data that can be delivered within a fixed time slot. As stated in [14], any resource of a web server (CPU, physical memory, disk, network) may become the bottleneck, depending on the kind of workload it is experiencing.

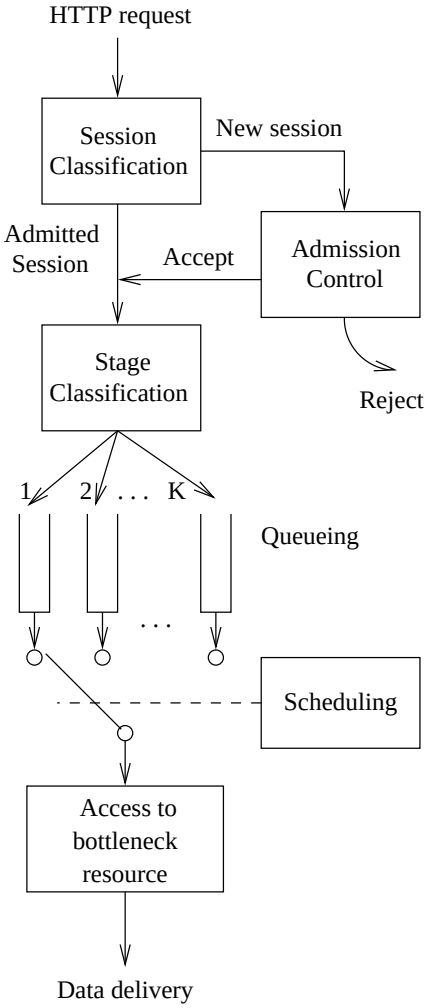

Fig. 1. Admission control and scheduling of HTTP requests

In this paper, we assume that the service rate of this bottleneck resource is a constant ν, and that the time t required to process a web page of size d is $t = d/\nu$. If desired, other delay models could be used instead within our framework.

The probability that a user does not terminate a session before completion of the processing of a request is represented by a *patience function*. Denote such a function by $w(\tau)$, where $\tau \geq 0$ is the time from when a request is received by the server (excluding time spent in the listen queue) until the processing of the request is completed. Thus, $w(\tau)$ is monotonically decreasing. Since τ does not include network delay, the end-to-end delay of a web server response can be longer. Due to impatience caused by this additional delay and due to the positive probability of a user terminating a session even if the response to a request is immediate, $w(0)$ can be expected to be less than 1.

Let $\mathcal{I} = \{1, \ldots, K\}$ be the set of stages, where an arbitrary stage $i \in \mathcal{I}$ be characterized by the following:

- distribution $C_i(\cdot)$ of reward obtained if the client successfully downloads a page at stage i, having mean $\bar{c}_i$
- distribution $D_i(\cdot)$ of the number of work units needed to process a request for this stage, having mean $\bar{d}_i$
- probability that a session is initiated at this stage, q_i
- transition probabilities, p_{ji}, $j \in \mathcal{I}$ — the probability that the next requested stage is j, given that the session is not interrupted
- patience function, $w_i(\tau)$

Moreover, denote the full matrix of stage transition probabilities by $\mathbf{P}$. In section IV, further stage-specific variables are introduced, associated with the choice of optimization algorithms.

For fixed response delay, the model of stage transitions within a session is a Markov chain, where a stage corresponds to a Markovian state. The reason for aggregating requests into stages (and not to treat each possible type request as an individual stage) is to limit the number of states in this Markov chain. This has the benefit of limiting the number of requests needed to observe before reliable estimates of the stage-specific parameters can be made, and it speeds up the convergence of the algorithm for optimization of scheduler parameters.

The properties of pages making up a stage should be similar, in the sense that the parameters listed above should be roughly equal. Menascé et al. [11] proposed an automatic process for stage classification and estimation of per-stage parameters, where a clustering algorithm such as k-means delineates the stages. The clustering problem closely resembles the state aggregation problem in the dynamic programming literature. The ideas outlined by Bertsekas [12] for reducing the number of states in Markov decision processes could be applicable.

Future development of the model may include other relations between the size of a web page and the processing time. For example, Chen et al. use a model where delay is the sum of a constant and a term that is proportional to the page size [10].

C. Objective

The goal of this paper is to develop algorithms that keep delay low for sessions expected to generate high reward, to prevent these sessions from terminating prematurely. If the server operates in an overload regime, sessions in stages less likely to generate reward pay the price for this by longer delays. Session admission control may help to avoid excessive delay.

Denote by λ_i the request arrival rate at stage $i \in \mathcal{I}$. Let $\bar{w}_i$ be the expectation of the patience function at this stage, $\bar{w}_i = E\{w_i(\tau)\}$, where τ is the response delay. The objective of our web server control is to maximize the average reward rate, ρ, defined as the rate of completed jobs (in response to client requests), multiplied by their associated expected rewards:

$$\rho = \sum_{i \in \mathcal{I}} \lambda_i \bar{w}_i \bar{c}_i \ . \tag{3}$$

Based on stage-wise queueing, heuristic algorithms could be constructed by prioritizing sessions in stages from which stages associated with high expected reward can be entered. However, it is not obvious how to design such a priority scheme in an optimal way. Instead, the approach taken here is to apply an optimization algorithm that computes the weights for a GPS scheduler, such that the reward rate is maximized.

IV. Reward Maximization

In this section, we develop equations and algorithms for computing reward rate and per-stage delays for given session arrival rates and scheduling weights. Once these relations are established, algorithms for for constrained nonlinear optimization can be applied to perform the reward maximization. The result is an open-loop controller, based on equilibrium assumption. The server-specific variables used in this section were previously introduced in section III-B.

A. Arrivals and controls

For a typical web service, the intensity of session arrivals varies according to a distinct pattern with a 24 hour period [10]. This can be modeled by a modulated Poisson process with time-dependent intensity. We denote the session arrival rate during a period of constant intensity by γ.

As stated in section III-A, the session admission controller puts an upper limit on γ. Denote the maximum admitted session arrival rate by γ_{max}.

Recalling that every request is classified into the set of stages $\mathcal{I}$, denote the rate of request arrivals at any stage $i \in \mathcal{I}$ by λ_i. Further, let ϕ_i be the scheduling weight assigned to the request queue at this stage, denote by $\bar{m}_i$ the mean number of requests in this queue, and let $\bar{\tau}_i$ be the mean time from entering the queue until service is completed.

B. Effective service rates

Only if all queues are backlogged, the service rate of an arbitrary backlogged queue n will equal the minimum rate implied by equation (2). Otherwise, the minimum service rates of the empty queues will be shared among the non-empty ones, according to equation (1). Denote by $\mu_n(t)$ the instantaneous service rate at time t that an ideal GPS server grants to a session n in the instantaneous set of backlogged queues, $n \in B(t)$.

$$\mu_n(t) = \frac{\nu \phi_n}{\sum_{j \in B(t)} \phi_j} \tag{4}$$

Define the *effective service rate* obtained by a queue n as the expectation of the instantaneous service rate, $\mu_n = E\{\mu_n(t)\}$. For the modeling of delay and reward rate, μ_n needs to be quantified for all $n \in \mathcal{I}$.

We observe that when obtaining service, n itself is always among the backlogged queues. While the status of n thus is known, each of the other $K-1$ queues can be either backlogged or not backlogged, resulting in 2^{K-1} possible sets of backlogged queues (all including n). Let these sets be denoted $B_1^n, B_2^n, \ldots, B_{2^{K-1}}^n$.

The probability of a queue $n \in \mathcal{I}$ being backlogged at arbitrary time t is equal to the utilization $0 \leq u_n < 1$, which is the ratio between the arriving work rate and the effective service rate:

$$u_n = P(n \in B(t)) = \frac{\lambda_n \bar{d}_n}{\mu_n} \,. \tag{5}$$

In the following analysis, we assume that backlogged time intervals for the different queues are independent. This is an approximation, since a traffic burst arriving at one stage will spread to subsequent stages. The benefit of this approximation is that the probability of a set of backlogged queues $B_k^n, k \in \{1, \ldots, 2^{K-1}\}$ to occur (given that n is backlogged) can be written as the product

$$P\{B(t) = B_k^n \mid n \in B(t)\} = \prod_{i \in B_k^n \setminus \{n\}} u_i \prod_{j \in \mathcal{I} \setminus B_k^n} (1 - u_j) \,. \tag{6}$$

Combining equations (4), (5) and (6), the effective service rate of stage n is the sum of instantaneous service rates for all backlogged sets B_k^n, weighted by their probability of occurring. This results in a nonlinear system of K equations in μ. For all $n \in \mathcal{I}$,

$$\begin{aligned} \mu_n \quad = \quad & \nu \sum_{k=1}^{2^{K-1}} \left[\prod_{i \in B_k^n \setminus \{n\}} \frac{\lambda_i \bar{d}_i}{\mu_i} \times \right. \\ & \left. \times \prod_{j \in \mathcal{I} \setminus B_k^n} \left(1 - \frac{\lambda_j \bar{d}_j}{\mu_j}\right) \times \frac{\phi_n}{\sum_{l \in B_k^n} \phi_l} \right] \end{aligned} \tag{7}$$

For general K, the closed-form solution to equation (7) is non-trivial. For use in the optimization algorithm, we choose to solve the equation numerically by fix-point iteration. After guessing initial values of the components of μ, the right hand side of the equations is applied iteratively as an update rule until convergence.

C. Constraints and response delay

Recall from section II-B that the scheduling weights are always positive. This constraint must be enforced during optimization.

$$\phi_i > 0 \quad \forall i \in \mathcal{I} \tag{8}$$

Further, it is possible to add QoS constraints. These can be chosen to make the offered web service comply with guarantees given to the customers, or reflect knowledge of customer psychology. A commonly discussed QoS constraint for web services is maximum response delay. However, it is also possible to define and implement other types of constraints, such as delay fairness.

As our proposed optimization method assumes equilibrium, a delay constraint that is well-suited for integration in the model is a limit on the mean delay:

$$\bar{\tau}_i \leq \bar{\tau}_i^{max} \quad \forall i \in \mathcal{I}, \tag{9}$$

where $\bar{\tau}_i^{max}$ is an upper QoS bound.

The analysis of service time distribution in GPS systems is generally very difficult, as the instantaneous service rate $\mu_i(t)$ at any moment t depends on the contents of all queues in the system [13]. In the following, we assume that the job size distribution $D_i(\cdot)$ is exponential, and make the approximation of treating the effective service rates μ as constant rates. This enables us to model every request queue as an M/M/1 queue. The

mean number of requests in the queue or being serviced at any stage $i \in \mathcal{I}$ is thus given by

$$\bar{m}_i = \frac{u_i}{1 - u_i}, \tag{10}$$

where u_i is given by equation (5). An expression for the expected time from entering the queue until service is completed is obtained from Little's formula, and can be simplified using equations (5) and (10).

$$\bar{\tau}_i = \frac{\bar{m}_i}{\lambda_i} = \frac{\bar{d}_i}{\mu_i - \lambda_i \bar{d}_i} \tag{11}$$

The delay model may be extended to non-exponentially distributed service time. A flexible way of doing this which requires small changes to the current model is to use a weighted sum of exponential distributions.

D. Algorithm

Recall that the probability of a session not being interrupted by the user before completion of the processing of a request at stage i is represented by a patience function, $w_i(\tau)$, where τ is the server delay. We assume that this function is well-modeled by an exponential function $w_i(\tau) = w_{i0} e^{-\theta_i \tau}$, where $0 \geq w_{i0} \geq 1$ is the probability that the session is not aborted in case of zero delay at the server.

In case an exponential model of user patience is found unsatisfactory, it is possible to use a weighted sum of exponential functions, analogously to the above discussion of service time distributions.

The expected *survival probability* $\bar{w}_i$ of a session at stage i is computed by integrating the patience function, weighted by the density function for the waiting time in an M/M/1 queue.

$$\begin{aligned} \bar{w}_i &= \int_0^\infty w_{i0} e^{-\theta_i t} (\mu_i / \bar{d}_i - \lambda_i) e^{-(\mu_i / \bar{d}_i - \lambda_i) t} dt \\ &= \frac{w_{i0}(\mu_i - \bar{d}_i \lambda_i)}{\mu_i + \bar{d}_i(\theta_i - \lambda_i)} \end{aligned} \tag{12}$$

There is no risk of the numerator or denominator in equation (12) becoming negative, as long as the workload arriving to the initial stages of all sessions does not exceed the server rate,

$$\sum_{i \in \mathcal{I}} q_i \gamma \bar{d}_i < \nu \ . \tag{13}$$

The reason for this is that the closer the left hand side is to the right hand side, the lower the session survival probability, and the lower the load resulting from sessions making a subsequent request. Thus, the server is effectively self-regulating.

Denote by h_{ji} the probability of a session entering stage j after leaving stage i, where $i, j \in \mathcal{I}$.

$$h_{ji} = \bar{w}_i p_{ji} \tag{14}$$

The flow of sessions into a stage $j \in \mathcal{I}$ has two components. One is $q_j \gamma$, the fraction of session arrivals that commence at this stage. The other is $\sum_{i \in \mathcal{I}} h_{ji} \lambda_i$, the sessions coming from other stages. For the incoming and outgoing flows at a stage to balance, the following equality must be satisfied for all $j \in \mathcal{I}$:

$$\lambda_j = \sum_{i \in \mathcal{I}} h_{ji} \lambda_i + q_j \gamma \tag{15}$$

For fixed stage transition probabilities, this is a system of linear equations. In effect, the web server at equilibrium is modeled as a Jackson network.

We now have all components required to compute ρ and $\bar{\tau}$ as functions of γ and ϕ for given system parameters. The following pseudo algorithm, based on fix-point iteration, summarizes the computations. Iteration indices k on the variables clarify the update order.

EVALUATE_SYSTEM(γ, ϕ)
- Initialize $\lambda(0)$ arbitrarily
- $k \leftarrow 0$
- Repeat until convergence
  - Solve (7) for $\mu(k)$
  - Compute $u_i(k)$ for all $i \in \mathcal{I}$ using (5)
  - Compute $\bar{w}_i(k)$ for all $i \in \mathcal{I}$ using (12)
  - Compute $h_{ji}(k)$ for all $i \in \mathcal{I}$ using (14)
  - Solve (15), denoting the solution $\lambda'(k)$
  - $\lambda_i(k+1) \leftarrow \alpha(k)\lambda'_i(k) + (1 - \alpha(k))\lambda_i(k)$ for all $i \in \mathcal{I}$
  - $k \leftarrow k + 1$
- Compute ρ using (3)
- Compute $\bar{\tau}_i$ for all $i \in \mathcal{I}$ using (11)

To avoid problems with numerical stability during convergence, $\mu_i - \bar{d}_i \lambda_i$ in equation (12) is truncated so as not to be less than a small, positive constant. Further, the forgetting factor $0 < \alpha(k) < 1$ depends on iteration time, and is tuned to be small enough to avoid oscillation, yet large enough to ensure convergence.

E. On-line operation

To optimize the weights of the scheduler, the parameters describing the system and customer behavior need to be estimated. These parameters are ν, $\bar{c}_i$, $\bar{d}_i$, w_0, θ, q and $\mathbf{P}$. Based on statistics collected during operation, estimates can be obtained using linear regression. Once confident estimates are available, EVALUATE_SYSTEM() will yield ρ and $\bar{\tau}$ as functions of ϕ.

We chose the simplex search optimization method [15], as implemented in [16]. Alternatively, state-of-the-art methods for nonlinear optimization, such as Sequential Quadratic Programming, could be used. Such methods normally need fewer iterations to reach optimum, since they employ first and second order derivatives of the objective function to guide the search. However, this also make them less robust to noise than the simplex method. The robustness of simplex search is a clear advantage in our case, since a very large number of iterations may be needed in EVALUATE_SYSTEM() to secure sufficient accuracy for finite-difference approximations of derivatives.

The weight positivity constraint (8) is enforced by truncating negative ϕ_i to a small, positive constant. In case the delay constraint (9) is employed, it is enforced by adding a penalty g to the objective function:

$$g = -\psi \sum_{i \in \mathcal{I}} 1_{\{\bar{\tau}_i > \tau_i^{max}\}} \lambda_i (\bar{\tau}_i - \tau_i^{max})^2 \ . \qquad (16)$$

Here, $1_{\{\bar{\tau}_i > \tau_i^{max}\}}$ is an indicator function, which takes the value of 1 if the constraint is violated and 0 otherwise. The constant λ_i scales the penalty such that every request subject to certain delay should result in the same punishment, independent of stage. The positive constant ψ makes sure that the magnitude of the punishment matches that of the original objective function.

Since the effective service rates μ depend on the ratios between scheduling weights ϕ rather than their absolute values, the number of degrees of freedom of the server control problem is $K - 1$. This is encoded by keeping one of the scheduling weights ϕ_i constant.

A possible mode of operation is to compute optimal ϕ for a range of λ values, and storing such (λ, ϕ) pairs in a lookup table. By estimating λ on-line and loading corresponding ϕ into the scheduler, load-dependent, near-optimal control is achieved.

Moreover, the optimal reward rates ρ corresponding to the values of ϕ may be stored in the lookup table. This makes it possible to analyze the reward rate curve for optima. If the curve has a maximum for a certain arrival rate, the maximum reward rate γ_{max} allowed by the admission controller may be set to this rate.

V. Case Study

This section presents a case study that demonstrates our methodology. The example is an electronic store, where users can enter the store, browse the product catalog, add items to an electronic shopping cart, remove items, and eventually check out and pay.

A. Reward maximization in the e-store

Our model has four stages, shown in Table I. As discussed in section III-B, each stage may comprise requests for a large number of different web pages. In our example, all requests for a page in the product catalog of the e-store are aggregated into one stage, given that the shopping cart is empty. As it is likely that the probability of checking out is higher if there are items in the shopping cart, another stage encodes requests made with a non-empty cart.

TABLE I
STAGES IN THE E-STORE EXAMPLE

Stage	Description
1	Welcome
2	Browsing, empty cart
3	Browsing, articles in cart
4	Checkout

The parameters describing the model are listed in table II. We assume that all users enter through the `Welcome` stage. Unity reward is paid on checkout. The average amount of work needed to process a request is identical at all stages, and the session survival probabilities for zero delay are all 0.9. The θ parameter of the patience function is set such that the probability of session survival is halved for $\tau = 15$ s at all stages.

TABLE II
PARAMETERS IN THE E-STORE EXAMPLE

Parameter	Value
K	4
ν	5000
q	$(\ 1.0 \quad 0.0 \quad 0.0 \quad 0.0\ )$
$\mathbf{P}$	$\begin{pmatrix} 0.0 & 0.2 & 0.1 & 0.0 \\ 1.0 & 0.5 & 0.2 & 0.1 \\ 0.0 & 0.3 & 0.5 & 0.2 \\ 0.0 & 0.0 & 0.2 & 0.2 \end{pmatrix}$
$\bar{c}$	$(\ 0.0 \quad 0.0 \quad 0.0 \quad 1.0\ )$
d	$(\ 100 \quad 100 \quad 100 \quad 100\ )$
$\bar{w}_0$	$(\ 0.9 \quad 0.9 \quad 0.9 \quad 0.9\ )$
θ	$(\ 0.0462 \quad 0.0462 \quad 0.0462 \quad 0.0462\ )$

B. E-store with delay constraints

This example is identical to the one above, with one exception: the mean delay QoS constraint (9) is enforced, with the maximum mean delay set to 4 seconds at all stages.

VI. Numerical Results

In the numerical case study, the scheduling weight of stage 4 was fixed to $\phi_4 = 100$. To ensure positivity and avoid numerical problems, an additional constraint, $\phi_i \geq 10^{-12}\ \forall i \in \mathcal{I}$, was added. The constant ψ in the penalty function (16) was set to 0.2. The e-store was modeled and optimized in Matlab, using the optimization toolbox [16].

The optimizer was run for consecutive values of γ, from left to right in the interval $[0.2, 10]$ with step 0.2. The first run was initialized by setting ϕ_1, ϕ_2 and ϕ_3 to 10^{-12}. As γ was successively increased, the weights found optimal for one value of γ served as starting point for the next value of γ.

To quantify the approximation errors of the equilibrium model, we also evaluated performance using an event-driven simulation of individual session arrivals, stage transitions and departures. In this model, we had to specify the distribution of the time between service completion and the next request for service. We chose an exponential distribution having mean 30 s.

The simulations comprised two scheduling disciplines: GPS, using the weights resulting from optimization, and first-come-first-server (FCFS), which is the discipline commonly employed in web servers. In case nothing else is stated, the simulation data presented below result from averaging the results of ten simulation runs, made up of 10^6 new session arrivals each (after simulation warm-up).

A. Performance evaluation

Figures 2, 3 and 4 display optimal ϕ, ρ and $\bar{\tau}$ resulting from unconstrained optimization, respectively. Figure 2 shows ϕ_1, ϕ_2 and ϕ_3 (recall that $\phi_4 \equiv 100$) for session arrival rates $\gamma \in [5, 10]$. For lower γ, the low load means that the weights have very little effect on delay $\bar{\tau}$ and reward rate ρ, since all the effective service rates μ_n are close to the total service rate ν.

Figure 3 shows that the reward rate grows approximately linearly with γ until the service rate of the web server begins to limit ρ. The modeled GPS curve has an optimum at $\gamma \approx 6.8$, whereas the simulated GPS curve (plotted with 95% confidence interval) shows that the true optimum is located at $\gamma \approx 7.5$, indicating the maximum session arrival rate γ_{max} that should be used by the admission controller to maximize the reward rate.

For $\gamma \in [5, 8]$ the discrepancy between the simulated and modeled curves is non-negligible. The main reason is correlation between per-stage backlogs. For $\gamma < 5$, the correlations are low, because the probability is low that the bottleneck service is occupied. For $\gamma > 8$, the high load means that the bottleneck resource is almost constantly occupied. This also reduces the correlation between backlogs. However, under medium load, the stage transition probabilities $\mathbf{P}$ make it probable that certain groups of stages are simultaneously backlogged, making the queueing delay longer than predicted at stages having low scheduling weights.

For comparison, figure 3 also displays ρ under FCFS scheduling. The performance advantage of GPS to FCFS is twofold. First, the greatest reachable reward rate is about 8% higher. Second, the performance degradation for session arrival rates greater than optimum is significantly faster with FCFS than with GPS.

The diagram in figure 4 shows simulated delays (plotted with lines and points) paired with modeled delays (plotted with lines only). Logarithmic scale is used for the delay axis to make all curves clearly visible, although the delays at stages 2 through 4 are so short that they hardly influence the survival probabilities. The diagram shows that the optimized controller favors long delay at stage 1, in order to keep the delay low at all other stages. This results in the termination of a large portion of admitted sessions. In effect, this policy implements session admission control, although the terminated sessions still receive service at stage 1. The low delay at subsequent stages for sessions that survive stage 1 make the probability high that these sessions continue all the way to checkout, thus contributing to the reward rate.

Figures 5, 6 and 7 display scheduling weights, reward rate and delay, respectively, with the delay constraint (9) activated, setting $\bar{\tau}^{max} = 4\ s$. The plot styles are analogous to figures 2, 3 and 4. In figure 6, the reward curve resulting from simulation using the weights from unconstrained optimization is plotted for comparison.

Comparing to the solution of the unconstrained problem, the weights ϕ diverge at $\gamma = 8.2$, where $\bar{\tau}$ reaches the constraint. Consequently, the delay constraint is met, at the expense of a lower reward rate. For example, comparing the curves for constrained and unconstrained optimization in figure 6 shows that constraining the delay decreases the reward rate by 6% for $\gamma = 10$ sessions/s.

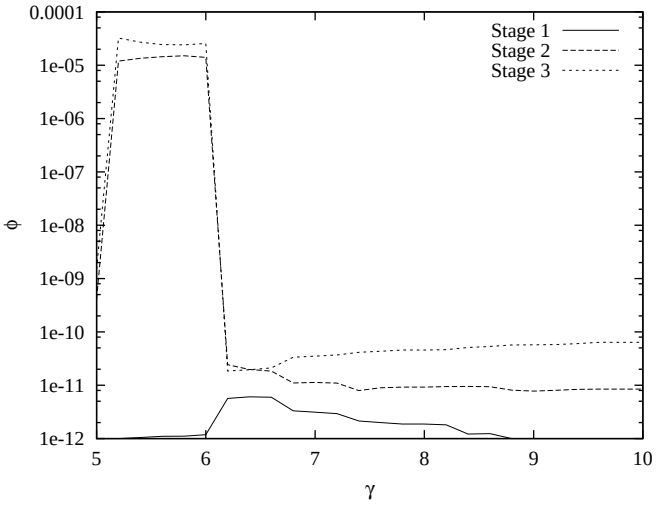

Fig. 2. Optimized weights

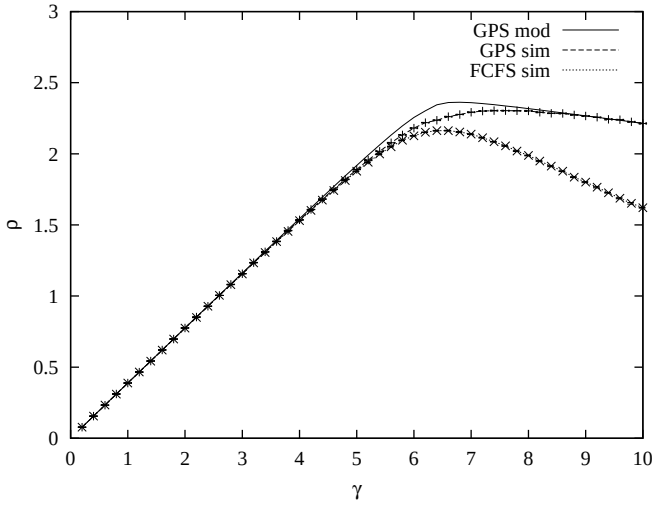

Fig. 3. Reward rate with GPS (model and simulated) and with FCFS

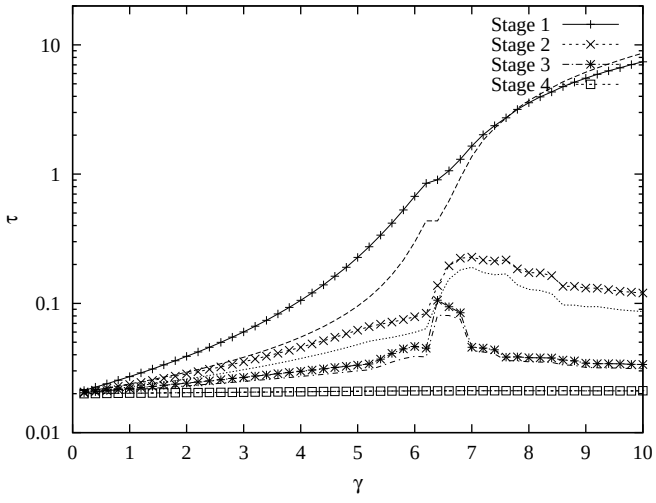

Fig. 4. Per-stage mean delay after optimization without delay constraints

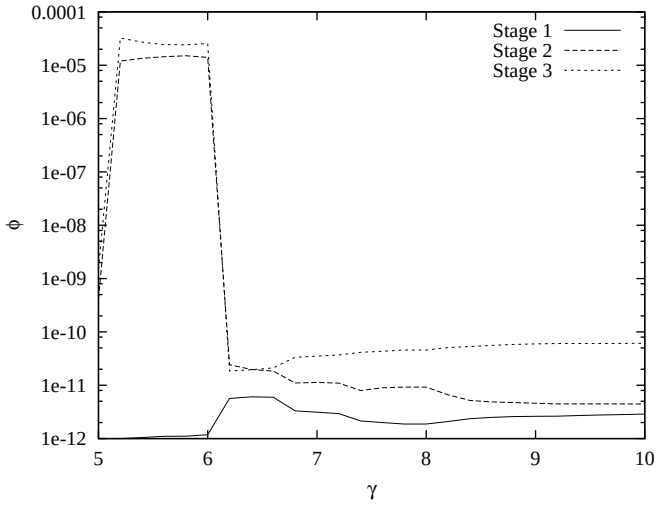

Fig. 5. Weights after optimization with delay constraints

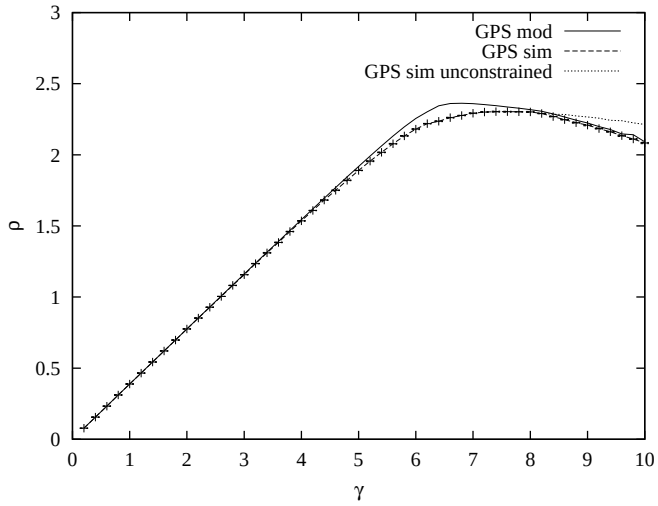

Fig. 6. Reward rate after constrained optimization

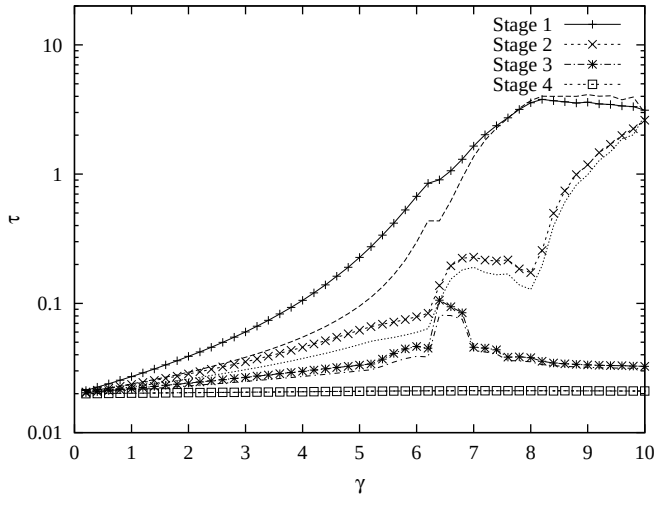

Fig. 7. Per-stage mean delay after constrained optimization

B. Sensitivity analysis

Since our proposed algorithms rely on estimating the parameters governing the dynamic behavior of the web server with traffic, it is desirable to quantify the effects of errors in these estimates. This section provides such analysis of our e-store case study without delay constraints.

We analyzed the sensitivity of the reward rate ρ and per-stage mean delay $\bar{\tau}$ to small changes in the model parameters in Table I (except K), using the equilibrium model. This was done by estimating partial derivatives by finite differencing for varying session arrival rate γ. Figures 8 and 9 display the sensitivity to variation of the stage transition probabilities $\mathbf{P}$, which were found to cause especially large variations in ρ and $\bar{\tau}$.

Unlike the sensitivity of $\bar{\tau}$, the sensitivity of ρ is bounded. For session arrival rates $\gamma < 6$, the sensitivity of $\bar{\tau}$ is low with respect to disturbances in any of the model parameters. However, as γ grows, $\bar{\tau}$ becomes more and more sensitive. This also holds for other model parameters. For example, $\Delta\bar{\tau}_1/\Delta\theta_1$ (not shown) decreases approximately linearly from zero to -100 when γ increases from 6 to 10. Moreover, note that some components of $\Delta\bar{\tau}/\Delta P$ change signs with changing γ. This is yet an indication of the complex dynamics of the web server.

A consequence of the high sensitivity of delay in case of high session arrival rates is that an admission-controlled web server, with γ_{max} set to maximize the reward rate, can offer QoS delay bounds that are more robust to parameter estimation errors.

We also studied the sensitivity of the reward rate ρ to variations in the scheduling weights ϕ. Figure 10 displays simulated ρ values for $\gamma \in [5, 10]$, and ϕ_2 multiplied by a factor κ ranging from 10^{-3} to 10^3. Since each data point results from a single simulation run, the plot is not as smooth as the simulated GPS curve in 3.

As can be seen in figure 2, the variation of κ means that ϕ_2, which is smaller than ϕ_3 and greater than ϕ_1 at optimum (marked by a thick line in figure 10), will range from being smaller than both ϕ_1 and ϕ_3 to being greater than both of them. The graph shows a significant decrease in ρ for $\phi_2 < \phi_1$, whereas the change of reward rate caused by $\phi_2 > \phi_3$ is very small.

To summarize the sensitivity analysis, confident estimates of the system parameters, especially those of the patience functions, are important for making reliable predictions of delay and reward rate. Unless the session arrival rate is very high, the sensitivities are not extreme, and should be possible to accommodate. Regarding sensitivity to weight changes, the ordering of the weights is more significant than exact values of the weights. This may suggest strict priority queueing as a cheap alternative to GPS. However, the possibility of tuning mean delays to meet QoS constraints would be lost with strict priority queueing.

VII. Conclusion

This paper presented an architecture and algorithms for request scheduling and session-admission control in web servers. The key idea is to break down sessions into stages with specific service requirements and transition probabilities, and make the web server aware of this structure. By controlling the resource sharing between stages, an application-specific reward function is maximized.

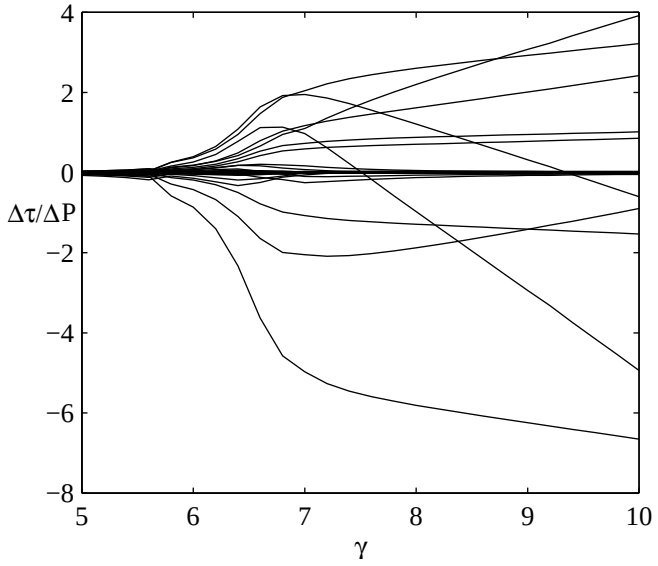

Fig. 8. Sensitivity analysis: $\{\frac{\Delta \bar{\tau}_k}{\Delta p_{ij}}\}$ for all $i, j, k \in \mathcal{I}$

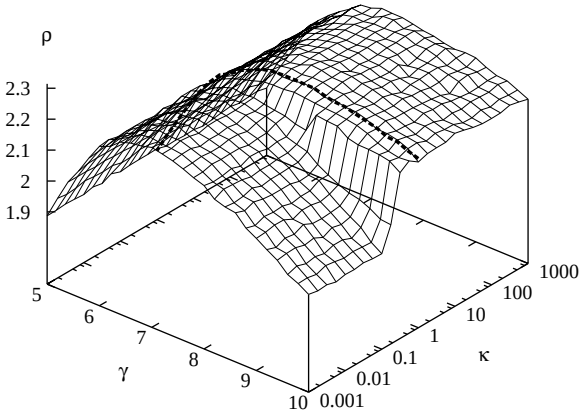

Fig. 10. Reward rate resulting from multiplying weight of stage 2 by values in $[0.001, 1000]$

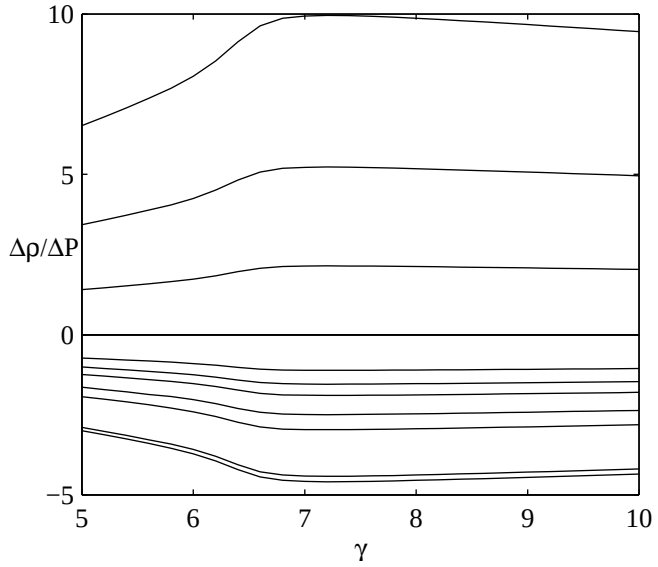

Fig. 9. Sensitivity analysis: $\{\frac{\Delta \rho}{\Delta p_{ij}}\}$ for all $i, j \in \mathcal{I}$

We proposed a fix-point algorithm for computing reward rate and per-stage delays for given system parameters, controls and load. Using this algorithm, the weights of a scheduler emulating generalized processor sharing (GPS) can be optimized by means of simplex search. Before optimization, system parameters are estimated from measurements during client-server interaction. Optionally, session-based admission control may support the reward rate maximization, by preventing the session arrival rate from exceeding the value for which the reward rate reaches maximum.

In a numerical case study, a fictitious e-store with four stages was modeled. Our optimized GPS scheduler reached up to 8% higher reward rate than a server using the first-come-first-server discipline. GPS was also shown to yield lower performance degradation for session arrival rates above the maximum point. Consequently, scheduling of HTTP requests has larger impact on the reward rate if a web server is allowed to be overloaded than if it is not.

Further, we showed how to include bounds on the mean delay per stage as optimization constraints. The integration of delay bounds was also demonstrated numerically. With delay constraints activated, the reward rate decreased compared to the unconstrained case.

Sensitivity analysis of the e-store example showed that delay is more sensitive to modeling errors than is the reward rate. The sensitivity of delay increases with increasing load. We also found that the ordering of scheduling weights has much larger impact on the reward rate than the exact values of the weights, suggesting strict priority queueing as cheap alternative to GPS scheduling; however, without the possibility of tuning mean delays to meet QoS constraints.

REFERENCES

[1] P.R. Selvridge, B. Chaparro and G.T. Bender, "The world wide wait: Effects of delays on user performance", in *Proceedings of the IEA 2000/HFES 2000 Congress*, 2000.

[2] N. Bhatti, A. Bouch and A. Kuchinsky, "Integrating user-perceived quality into web server design", in *Proceedings of the 9th International World-Wide Web Conference*, pp. 1-16. Elsevier, May 2000.

[3] M.E. Crovella, R. Frangioso and M. Harchol-Balter, "Connection scheduling in web servers", in *Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems* (USITS'99), Boulder, CO, Oct 1999.

[4] L. Cherkasova and P. Phaal, "Session based admission control: a mechanism for improving performance of commercial web sites", in *Proceedings of the International Workshop on Quality of Service*, London, UK, June 1999.

[5] V. Kanodia and E. Knightly, "Multi-class latency-bounded web services", in *Proceedings of the International Workshop on Quality of Service 2000* (IWQoS 2000), Pittsburgh, PA, June 2000.

[6] D. Clark, S. Shenker, L. Zhang, "Supporting real-time applications in an integrated service packet network: architecture and mechanism", in *Proceedings of ACM SIGCOMM'92*, pp. 14-26, 1992.

[7] A. Golestani, "A self-clocked fair queueing scheme for broadband applications", in *Proceedings of IEEE INFOCOM'94*, Apr. 1994, pp. 636–646.

[8] J.C.R. Bennett and H. Zhang, "WF^2Q: Worst-case fair weighted fair queueing", in *Proceedings of IEEE INFOCOM'96*, March 1996, pp. 120–128.

[9] D. Stiliadis and A. Varma, “Efficient fair queueing algorithms for packet-switched networks”, *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp 175–185, Apr. 1998.

[10] X. Chen, P. Mohapatra, H. Chen, “An admission control scheme for predictable server response time for web accesses”, in *Proceedings of The Tenth International World Wide Web Conference (WWW10)*”, Hong Kong, `http://www10.org`, May 2001.

[11] D.A. Menascé, V.A.F. Almeida, R. Fonseca and M.A. Mendes, “A methodology for workload characterization of e-commerce sites”, in *Proceedings 1999 ACM Conference on Electronic Commerce*, Denver, CO, Nov., 1999, pp. 119–128.

[12] D.P. Bertsekas, *Dynamic programming and optimal control*, vol. 2, Athena Scientific, Belmont, MA, 1995.

[13] Z.L. Zhang, D.F. Towsley and J.F. Kurose, “Statistical analysis of generalized processor sharing scheduling discipline”, in *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp 1071–1080, Aug. 1995.

[14] L. Eggert and J. Heidemann, “Application-level differentiated services for web servers”, in *World Wide Web Journal*, vol. 3, no. 2, pp. 133–142, Sep. 1999.

[15] J.C. Lagarias, J.A. Reeds, M.H. Right and P.E. Wright, “Convergence properties of the Nelder-Mead simplex method in low dimensions”, *SIAM Journal of Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[16] *Optimization toolbox for use with Matlab*, User’s Guide version 2, The MathWorks, `http://www.mathworks.com`, 2000.