# Response-Time Control of a Single Server Queue

Kjaer, Martin Ansbjerg; Kihl, Maria; Robertsson, Anders

*Published in:*
Proc. of the 46th IEEE Conference on Decision and Control

2007

Proceedings of the
46th IEEE Conference on Decision and Control
New Orleans, LA, USA, Dec. 12-14, 2007



# Response–Time Control of a Single Server Queue

Martin A. Kjær, Maria Kihl and Anders Robertsson

*Abstract*— **Feedback in server systems has during last years gained much interest in order to fulfill still increasing demands on performance and optimization regarding, for example, quality of service (QoS) requirements. In this paper we investigate a feedback–based prediction scheme for controlling a single server queue. This control structure has the benefit over other previously suggested control structures that no measurement of the required work of each job is needed. However, our solution maintains the same attractive properties, regarding average response time and variance.**

## I. Introduction

Server systems are used in most (tele)communication networks. The server system can process jobs from either other network entities or from clients. In the Internet, web servers send web pages to clients. In cellular networks, Home Location Registers handle subscriber information that may be requested by other nodes in the network.

Different control mechanisms may be used to control the server system. Admission control [1] prevents system overload by rejecting some jobs when needed. Load balancing [2] optimizes the load in a distributed system. Scheduling [3] can optimize the delays when jobs have different QoS requirements. Since a server system consists of CPU:s (the servers) and limited job queues, it has a non-linear behavior. A small increase in load can result in rapidly growing queues and delays. Therefore, the design of the control mechanisms is not a simple task if optimized performance is desired. In recent years this field has gained a large research interest from both academia and IT vendors [4], [5].

In this paper we focus on response-time control; an objective directly coupled to the end-user, but with close relations to *e.g.,* the more server oriented queue length control [6]. However, the connection between the queue length and the response time depends on numerous factors such as the nature of the arriving traffic and the service times.

Liu *et al.* [7] designed a response time control system based on admission control, where the admission probability was adjusted to obtain the desired response time. The control set–up included a queuing–theory based feed–forward where a measurement of the average required work was required. This was measured offline under low load, and then applied in on–line control. The offline identification of the average required work was also the procedure of the work by Lu *et al.* [8] and the work of Henriksson *et al.* [9] where feed–forward was used as part of a control set–ups with resource allocation actuators. This method can result in serious degradation of performance if the average required work changes, as might happen in real applications. The required work is not usually easy to define in real server systems that often operate by processor sharing, and it can be hard to give any qualified estimates of the this quantity.

M. A. Kjær and A. Robertsson are with the Department of Automatic Control, LTH, Lund University, Box 118, SE-221 00 Lund, Sweden {Martin.A.Kjaer,Anders.Robertsson}@control.lth.se.

M. Kihl is with the Department of Electrical and Information Technology, LTH, Lund University, Box 118, SE-221 00 Lund, Sweden Maria.Kihl@eit.lth.se.

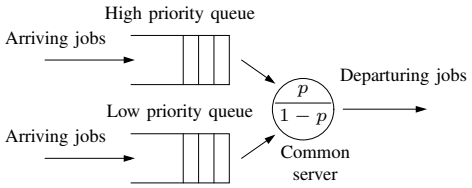


Fig. 1. Two queues with common server. The fraction $p$ of the server capacity is dedicated to the high–priority queue, and the fraction $1-p$ is dedicated to the low–priority queue.

In this paper, we expand existing prediction schemes by adding a feedback mechanism. The new prediction method is utilized for a new response time control scheme. This controller is shown, through simulations, to maintain the same desired robustness and transient properties as the previous published controllers, but here obtained without the measurement of the required work. The controller is compared to control strategies using the same or more measured variables.

## II. Queuing Model

The target system in this paper is a single-server system that processes jobs coming from clients, as illustrated in Fig. 1. Clients are considered satisfied if the response time $T$ (the time between the arrival of a job and the time where is has been fully served, also often denoted the system time [10]) for a job is less than a certain time limit, $T_{ref}$, in average. Also, there are low-priority jobs in the server systems, for example, maintenance and monitoring jobs. The low-priority jobs have no strict real-time requirements, but they should be processed. We assume that jobs coming from clients have high priority, but since we need to assure that also low-priority jobs can be served, the high-priority jobs only have a share, $p$, of the server capacity (where $0 \leq p \leq 1$). $p$ should be set so that the average response time, $T$, for a high-priority job is kept at the reference value, $T_{ref}$ with low variance.

We consider a system with two classes of requests, high-priority jobs and low-priority jobs, each with its own job queue. It is assumed that there are always low-priority jobs waiting for service, which means that the low-priority job

1-4244-1498-9/07/$25.00 ©2007 IEEE.

class uses all its available server capacity.

Jobs have a required work $w$ that describes which processing time the job would have if it was given the full server capacity. This gives the relationship $x = w/p$, where $x$ is the processing time.

## III. Control system

A block–diagram of the server system is illustrated in Fig. 2. The inputs are those variables that are directly accessible by the controller, and in the target system, the only input is $p$. The disturbances are the inputs that are not affectable by the controller. They might be measurable, but in many cases they are not. In the target system, the inter–arrival times of new jobs, denoted $a$, and required work, $w$, can be regarded as disturbances. Outputs are measurable quantities that depend on the system under investigation, and the previous behavior of the inputs. In the target system, the response time for a job, denoted $T$, and the job queue length, denoted $N$, are the outputs.

In this paper, the objectives of the control system are the following. First, to maintain an average response time $E\{T\} = T_{ref}$ for high-priority jobs with $p$ as control variable. Second, to reduce the average allocated server capacity $E\{p\}$ while minimizing the response–time variance $E\{T^2\} - E\{T\}^2$. Only the queue length ($N$) and the response time ($T$,) the time of arrivals ($a$), and, if necessary, the required work ($w$) should be used by the control system. Also, effects of varying arrival rates should be kept minimal.

The dynamics of a queuing system can change dramatically depending of the load and the length of the queue. If the queue holds a lot of jobs, the influence of the statistics will be reduced because of the averaging effect. Since the response time for a job can be measured when the job leaves the system, it is natural to adjust $p$ at departure instants. However, this control action will not only affect the response time of the job now being served, but it will also affect the response time of all the other jobs waiting in the queue. Likewise, the response time of the job being served will have been affected by the control signals set while the job was waiting in the queue. This means that if the queue on average holds many jobs, a significant, and non–constant, time–delay will arise, which can reduce the control performance. If, on the other hand, the queue is almost empty at all time, the stochastic nature of both the inter–arrival times and the processing times will have large affect on the obtained response times, thus complicating the control actions. Furthermore, a short queue is often obtained on the cost of a higher control signal on average.

## IV. Controllers

Due to the robustness properties inherited in every closed–loop control strategies, all the following control design methods will be based on feedback.

### A. Classical feedback controllers

The classical closed–loop setup contains a measurement of the objective variable, compared to the desired reference.

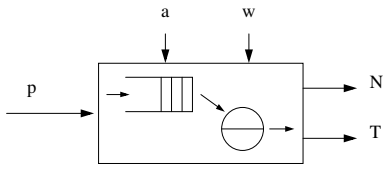


Fig. 2. The high–priority queue seen from a control perspective. The arrival times ($a$) and required work ($w$) are unaffectable quantities and are treated as disturbances. Dynamic variables of the queuing system are the queue length ($N$), the response time ($T$), which are treated as outputs. The fraction of the server capacity denoted to the high–priority queue ($p$) is a user–defined parameter, and is therefore treated as input.

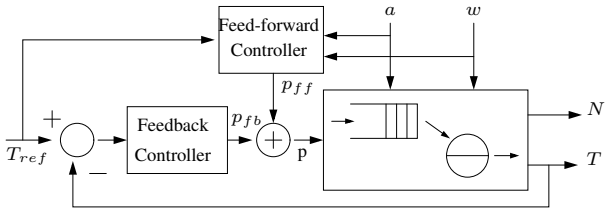


Fig. 3. The high–priority queue with a classical feedback controller. The setup can also hold a feed–forward mechanism in order to achieve better performance.

Based on the error (and possible past errors), the control signal is adjusted in a manner intended to reduce the error. A simple method is to adjust the control signal with a fixed amount as long as the error is above some threshold. This strategy will in the future be denoted *step control*. Another method is to use variants of the PID-controller [11], [12].

### B. Feed–forward based prediction

The control strategies above are based solely on feedback. This means that information is taken from the outputs alone, and sent back to the controller, thus for the controller to react, an error must be present. It is often possible to predict, based on measurements of e.g. the disturbances, that an error is about to arise if no immediate action is taken. If such information is available, it can be utilized in the control mechanism in what is often called *feed–forward*.

The combination of feedback and feed–forward for the single–server queue can be seen in Fig. 3. Here the arrival times and the required work are used for feed–forward. If the feed–forward block is removed, the control is based exclusively on feedback. Removing the feedback and relay only on the feed–forward results in an open–loop scheme, which means that the robustness properties of the closed loop are lost.

Measurements of disturbances can be included in many ways. In control of queuing systems, queue–theoretic approaches are often taken, such as in [9], [13]. In [9] a prediction for the response time is derived where the prediction is calculated at the departure of a job from the server (at time $t_{now}$). Assuming that there are jobs in the queue, the predicted response time of the remaining jobs is given by the average time they have spent in the queue plus a prediction of how long time it will take to serve them. A

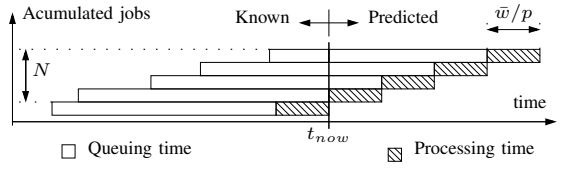


Fig. 4. Server and queuing diagram. The average time of the known part is given by $\frac{1}{N}\sum_i(t_{now}-a_i)$. The average time of the predicted area is given by $\frac{(N+1)\bar{w}}{2p}$.

slightly modified version of the predictor is given by

$$\hat{T} = \frac{1}{N}\sum_i(t_{now}-a_i) + \frac{(N+1)\bar{w}}{2p} \quad (1)$$

where $\bar{w}$ is the expected required work for all the remaining jobs.

A graphical interpretation of the predictor is given in Fig. 4. See also [9].

Using this predictor, a feed–forward signal can be calculated by

$$p_{ff} = \frac{(N+1)}{2(T_{ref} - \frac{1}{N}\sum_i(t_{now}-a_i))}\,\bar{w} \quad (2)$$

Notice that an estimate of the required work is needed in order to use the predictor. Also note, that the feed–forward in (2) does not calculate the predicted response time explicitly, and therefore, there is no mechanism to ensure that the prediction is correct.

### C. Proposed feedback-based prediction and control

In this section we propose a redesign of the response–time prediction in order to improve the prediction based on knowledge of earlier predictions. This is performed by imposing feedback into the prediction scheme. This is a method often used in linear state estimation [12]. The control scheme presented here contains the robustness properties of feedback as well as the predictive actions from the feed–forward.

Instead of using the information from the measurements of the required work in (1), a virtual parameter $z$ is used instead. This parameter can be interpreted as the compensated averaged required work. The predictor is now given by

$$\hat{T} = \frac{1}{N}\sum_i(t_{now}-a_i) + \frac{(N+1)}{2p}\,z \quad (3)$$

The estimated service time is now seen as a new output and $z$ as an input. The purpose of this control problem is to make the response–time estimate $\hat{T}$ follow the real response time $T$. The control error is therefore given by $e_T = T - \hat{T}$. Using a PI controller with $e_T$ as input and $z$ as output will, at least in steady state, assure that $E\{\hat{T}\} = E\{T\}$. The variable $z$ can now be used to calculate the feed–forward control signal for the single server queue:

$$p_{ff} = \frac{(N+1)}{2(T_{ref} - \frac{1}{N}\sum_i(t_{now}-a_i))}\,z \quad (4)$$

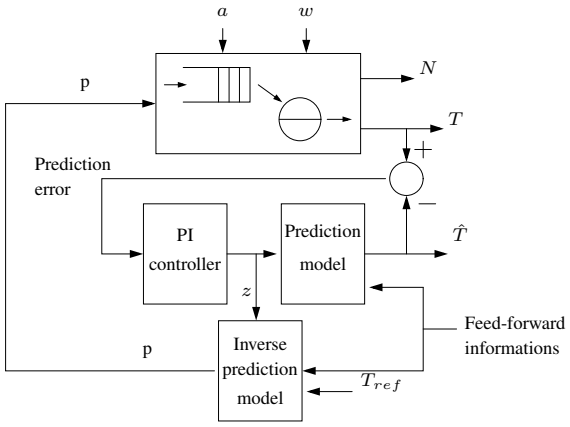


Fig. 5. The response–time prediction is updated by a feedback mechanism. Using the internal state $z$, the inverse queue model gives the control signal $p$ required to obtain the desired average response time.

The control setup using this feedback based prediction scheme is illustrated in Fig. 5. It is noted that the structure has significant similarities to the state feedback control structure, which is often used in linear control design [12]. Because of this similarity, the purposed control scheme will be denoted *state feedback control* in the remainder of this paper.

Alternatively, the PI–controller could be expanded with a differential part (PID) which has been successfully used in many other applications to achieve faster responses. The differential part is sensitive to noise, and since the system under investigation is extremely noise, the differential part has been discarded at an early stage.

## V. Investigations

We investigated the controllers with simulations based on a discrete-event simulation program written in Java. The Java-program included classes for the traffic generator, the queue, the observer and the controller. In this section all details regarding our investigations will be described.

### A. Controller Design

Five controllers were evaluated: A step controller, two ordinary PI-controllers, one PI controller combined with feed–forward, and one state–feedback controller. One PI-controller was designed to obtain fast transients, in the following called the *fast PI* controller. The other PI-controller, called *slow PI*, was designed to avoid oscillations. All controllers were event based, and was triggered by when a job departs from the server.

To reduce the variance of the control signal, a low–pass filtered response time was used. The filter, which also was event based, was given by

$$T_f(k) = 0.9\,T_f(k-1) + 0.1\,T(k)\ . \quad (5)$$

The control signal was truncated to be in the interval $p_{min} = 0.001$ to $p_{max} = 1$ before applied to the server.

**The step controller** was given by

$$p(k) = \begin{cases} p(k-1)+d & for \quad e(k) > h\,,\, p(k-1) < p_{max} \\ p(k-1)-d & for \quad e(k) < h\,,\, p(k-1) > p_{min} \\ p(k-1) & else \end{cases} \tag{6}$$

where $h$ was the threshold and chosen to $h = 0.005$. The step size was $d = 0.001$. The filtered response time was used to calculate the control error $e(k) = T_{ref} - T_f(k)$.

**The slow PI controller** was designed with an anti–windup scheme as in [14]

$$\begin{aligned} I(k) &= I(k-1) + g(1-r)\cdot e(k) && (7)\\ &\quad + \frac{g(1-r)}{10}\big(p_{fb}(k-1) - p(k-1)\big) && (8)\\ p_{fb} &= g\cdot r\cdot\, e(k) + I(k) && (9) \end{aligned}$$

The variable $I$ represent the integral part of the PI controller. The controller constants $g$ and $r$ were chosen to $0.0005$ and $0.01$, respectively. The controller was based on the filtered response time $e(k) = T_f(k) - T_{ref}$.

Alternatively, the integral part of the PI controller could be implemented with varying amplification to take the aperiodically nature of the event–based control update into account, as in [15].

**The fast PI controller** was equivalent to the slow PI controller, except for the control parameter $g = 0.01$.

**The slow PI–FF controller** used the same feedback–loop as the slow PI controller, but a feed–forward mechanism was also included. A inverse response–time predictor was given by

$$\begin{aligned} Q(k) &= \frac{1}{N}\sum_i (t_{now} - a_i(k)) && (10)\\ p_{ff}(k) &= \begin{cases} \frac{N(k)+1}{2\,(T_{ref}-Q(k))}\, w_f(k) & for \quad T_{ref} > Q(k) \\ p_{max} & else \end{cases} && (11) \end{aligned}$$

where $a_i(k)$ are the arrival times of the jobs in the server system at departure $k$ and $t_{now}$ is the current time. The variable $w_f$ is a filtered version of the required work $w$, given by $w_f(k) = 0.00\, w_f(k-1) + 0.01\, w(k)$. It is assumed that $w$ can be measured. The control signal was given by a summation of $p_{ff}$ the output of the slow PI controller in (9). Note that this feed–forward strategy requires a measurement of the required work $w$.

**The state feedback controller** was implemented with a feedback–corrected prediction scheme as

$$\begin{aligned} Q(k) &= \frac{1}{N}\sum_i (t_{now} - a_i(k)) && (12)\\ \hat{T}(k) &= Q + \frac{N+1}{2\,p(k-1)}\, z(k-1) && (13)\\ \hat{T}_f(k) &= 0.9\,\hat{T}_f(k-1) + 0.1\,\hat{T}(k) && (14)\\ e(k) &= T_f(k) - \hat{T}_f(k) && (15)\\ z(k) &= z(k-1) + g\,\big(e(k) - r\,e(k-1)\big) && (16)\\ p(k) &= \begin{cases} \frac{N(k)+1}{2\,(T_{ref}-Q(k)}\, z(k) & for \quad T_{ref} > Q(k) \\ p_{max} & else \end{cases} && (17) \end{aligned}$$

The control parameters were given by $g = 0.0001$ and $r = 0.01$. Note that this strategy does not require a measurement of the required work $w$.

### *B. Simulations*

Both steady-state and transient simulations were performed. All steady–state results were evaluated after all transients had been removed. Transient behavior was investigated after the queuing system had converged to steady state, and was allowed to run for sufficiently long time to make sure that all transient behavior was observed. In all simulations, we used a Poisson process for modeling the arrival process of new jobs. Also, the required work of a job was exponentially distributed.

In the steady state simulations, the mean arrival rate of new jobs, $\lambda$, was first varied between 10 and 90 jobs per second. The mean required work for a job was 0.01 seconds. If $p = 1$ (i.e., an uncontrolled system) and $\lambda = 50\,/s$, the average response time would become 0.02 seconds with a mean queue length of 0.5 jobs. The reference value for the response time, $T_{ref}$, was set to $0.6$ seconds, which correspond to a queue length of about 30 jobs for $\lambda = 50\,/s$ (Little's law [16]).

Then, the response time reference $T_{ref}$ was varied from $0.05\,s$ to $1.7\,s$ while the arrival rate was kept constant at $\lambda = 50$. All other parameters was as with the previous steady–state experiment.

In the transient simulations, the required work distribution was kept constant as an exponential distribution with mean 0.01 seconds. However, the average arrival rate of new jobs changes from 20 to 80 jobs per second at time $t = 8000$ seconds. This is a considerable change in arrival rate, which brings the queuing system from operating with low load to operating in the high end of the medium–load range. All initial transient behavior has died out and the system has reached steady state before the change of arrival rate.

## VI. Results and Discussion

In this section we will present and discuss the results from the investigations.

### *A. Steady state simulations*

Fig. 6 shows some performance metrics from the simulations. The top of the figure shows how the controllers track the reference as the traffic changes. It is noticed from the figure that both the fast PI controller and the step controller are not capable of maintaining the desired response time in steady state. Also the proposed state feedback controller suffers from poor steady state reference following at both high and low arrival rates, whereas it performs satisfactory at medium arrival rates in the range of $20 - 80\,/s$. The bottom of Fig. 6 shows the variance of the response time. Here the benefit of using feed–forward becomes obvious. The combined PI and feed–forward controller and the state feedback controller damp the stochastic variations of the queuing system far better than any of the controllers solely based on feedback.

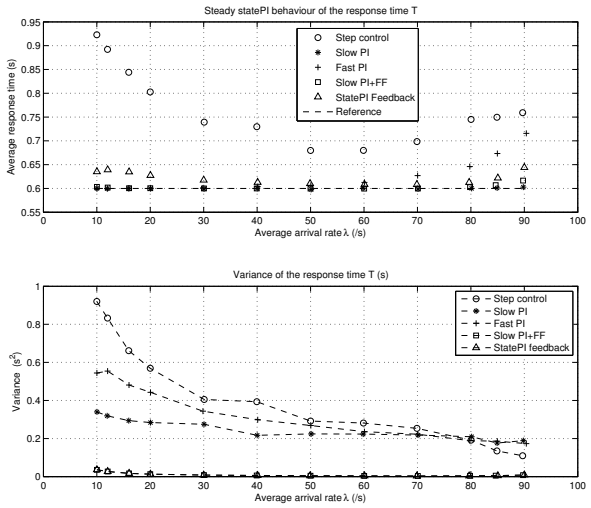


Fig. 6. Performance metrics for the controllers. Top: steady state reference following of the response time. Bottom: Steady state variance of the response time. Notice that the slow PI controller, the slow PI controller with feed–forward, and the state feedback controller coincide in some parts in the top figure. So does the slow PI controller with feed–forward and the state feedback–controller in the lower plot.

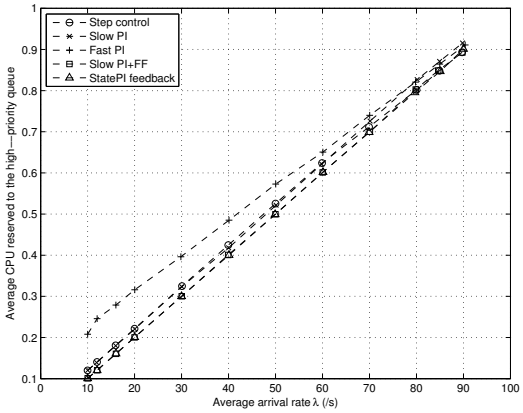


Fig. 7. Server capacity allocation as function of arrival rate in steady state.

Fig. 7 shows the averaged allocated CPU as function of the arrival rate. The figure shows a general trend; the need for server resources increases with the traffic. The figure even indicates that this relationship is linear. By close inspection it can be seen that the slow PI controller demands an insignificantly smaller amount of server resources than both the combined PI and feed–forward controller and the state feedback controller for arrival rates above $20\,/s$. Below this, the step controller and the slow PI controller seem to perform slightly better. In this region the stochastics become extremely dominating, and predictive control, which is based on averaging, becomes useless. The slow reacting feedback–based mechanisms therefore perform better. However, in the dominating arrival rate span, the feed–forward based controllers show superior performance.

A common way to represent simulation and experiment data in the telecommunication community is the distribution as illustrated in Fig. 8. The dashed line represents the result

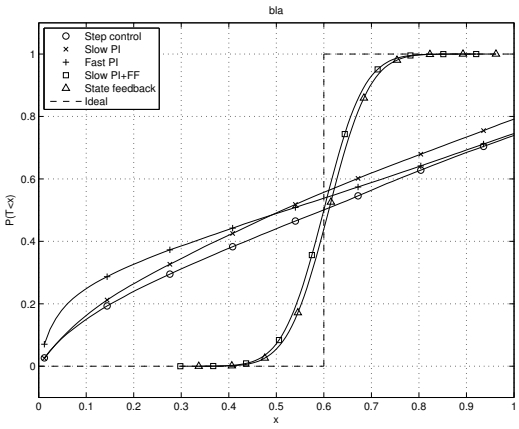


Fig. 8. Response time distributions. Notice that the the slow PI controller with feed–forward and the state feedback controller coincide in most parts of the figure.

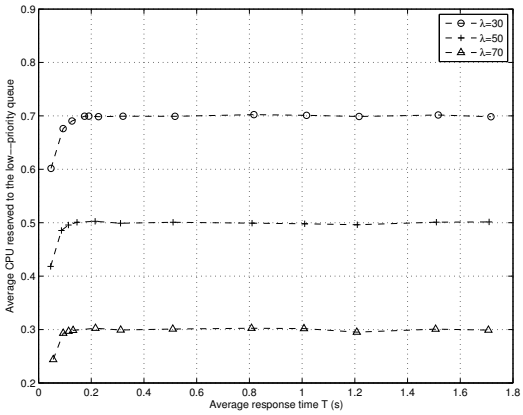


Fig. 9. Steady state simulation results for the state feedback controller. The resources available for the low priority queue, as function of response time $T$, plotted for different values of arrival rate $\lambda$.

of an ideal response. The figure indicates that none of the controllers are ideal, but the two controllers with feed–forward (the slow PI+FF controller and the state feedback controller) performs significantly better. The Step controller and the two pure–PI controllers are skewed and show to result in high possibility for long response times.

Fig. 9 shows what is earned by using control, since the capacity not reserved for the high-priority queue is available for the low-priority queue. The figure shows that a lot is gained when accepting slightly longer response times. However, as longer response times are accepted, the increase in capacity for the low-priority queue diminishes.

### *B. Transient simulations*

One strong argument to use feedback in the control is the robustness towards changes in the environment. The above simulations were all performed with steady state environments, which means that the both the required work and the arrival rate had constant distributions. It is of high relevance to investigate the behavior of the controlled queuing system under changes in the environment. Fig. 10 illustrates the

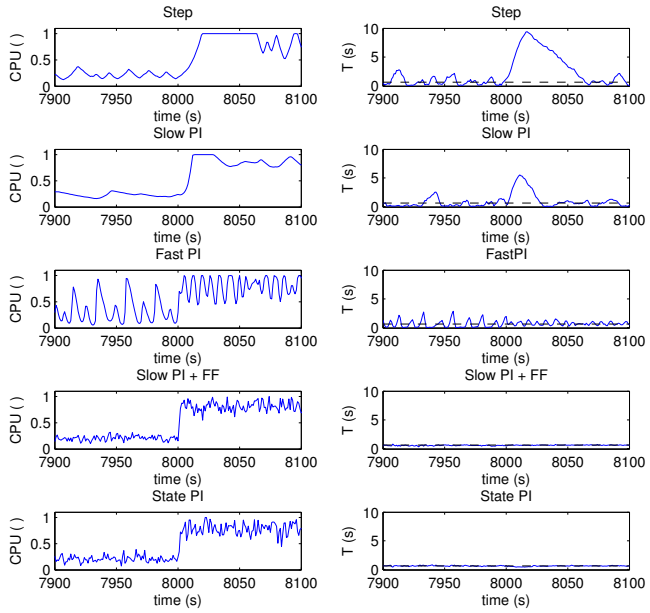


Fig. 10. Transient behavior of the single server queuing system. An increase in the arrival rate from $20\,/s$ to $80\,/s$ occurs at time $t = 8000\,s$. The left column shows the control signal and the right column shows the response times. Rows one to five show results for the step controller, the slow PI controller, the fast PI controller, the combined PI and feed–forward controller, and the state controller, in the mentioned order.

transient behavior of the single serve queue with the different controllers.

The transient behaviors show that the slow designed controllers based only on feedback result in a large transient at the arrival rate change. First the change must be detected in the response time measurements, and then the controllers must build up the control signal to compensate. During this time, the queue has built up, resulting in large response times. The fast PI controller manages to react faster to the disturbance and thereby maintaining a shorter queue. The transient behavior can be seen in the figure, which yields a oscillating behavior that is undesirable. It should be mentioned that the fast PI–controlled system becomes unstable (ever increasing queue length) if the reference is increased too much. The two controllers based on both feedback and feed–forward seems to manage the change well, which was also the reason to utilize the feed–forward. The feed–forward measures the change of arrival–rate and starts to compensate for it even before it has affected the response time. There is no significant difference between the transient performance of the two controllers.

### C. Discussion

Both the steady state simulations and the transient simulations show that the capability to maintain the reference and to reduce the response time reference is improved significantly by imposing feed–forward. The two feed–forward methods show similar properties in most of the tested cases. However, our proposed state feedback controller does not require a measurement of the required work, which can be considered as a considerable advantage.

## VII. Conclusions

Performance related control of server systems is a wide research area with numerous applications. In this paper, we have considered a single-server queue in which high-priority jobs should receive as much server capacity as needed to experience a certain response time. However, the server capacity should be optimized in order to allow for the execution of low-priority jobs.

We have proposed a state feedback controller that uses a prediction of the response time. We compare the proposed controller with other controllers found in the literature. The proposed controller is shown to have a similar desirable behavior for the target system than the best of other controllers despite that no measurement of the required work is required in our solution.

## VIII. Acknowledgments

This work has been founded by the Swedish Research Council, project 621-2006-5522.

## References

[1] M. Kihl, "Overload control strategies for distributed communication networks," Ph.D. dissertation, Dep. of Communication Systems, Lund University, Sweden, 1999.

[2] O. Kremien and J. Kramer, "Methodical analysis of adaptive load sharing algrorithms," *IEEE Trans. on Parallell and Distributed Systems*, vol. 3, no. 6, 1992.

[3] J. S. C. Lu, G. Tao, and S. Son, "Feedback control real-time scheduling: Framework, modelling and algorithms," *Real-Time Systems Journal*, vol. 23, no. 1/2, 2002.

[4] J. Hellerstein, Y. Diao, S. Parekh, and D. Tilbury, "Control engineering for computing systems," *IEEE Control Systems Magazine*, vol. 25, no. 6, pp. 56–68, 2005.

[5] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. Wiley-Interscience, 2004.

[6] M. Kihl, A. Robertsson, and B. Wittenmark, "Analysis of admission control mechanisms using non-linear control theory," in *Proc. IEEE Int Symp on Computer Communications (ICSS 2003)*, Kemer-Antalya, Turkey, June 2003.

[7] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multi-tiered web application using queueing predictor," in *Proc. 10th IEEE/IFIP Network Operations and Management Symp. (NOMS 2006)*, Vancouver, Canada, 2006.

[8] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback control with queueing-theoretic prediction for relative delay guarantees inweb servers," in *Proc. IEEE Real–Time and Embedded Technology and Application Sympositum (RTAS'03)*, Toronto, Canada, 2003, pp. 208–217.

[9] D. Henriksson, Y. Lu, and T. Abdelzaher, "Improved prediction for web server delay control," in *Proc. 16th Euromicro Conferance on Real–Time Ststems (ECRTS'04)*, Catania, Italy, June 2004.

[10] L. Kleinrock, *Queuing Systems*. New York: John Wiley & Sons, Inc, 1975.

[11] K. J. Åström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NC: ISA - The Instrumentation, Systems, and Automation Society, 2005.

[12] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Boston, MA: Addison-Wesley, 1994.

[13] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu, "Feedback control with queuing–theoretic prediction for relative delay guarantees in web servers," in *Proc. 9th IEEE Real–Time and Embedded Technology and Application Symp. (RTAS'03)*, Toronto, Canada, May 2003.

[14] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*. Upper Saddle River, NJ: Prentice Hall, 1997.

[15] K.-E. Årzén, "A simple event-based PID controller," in *Preprints 14th World Congress of IFAC*, Beijing, P.R. China, Jan. 1999.

[16] I. Mitrani, *Modelling of Computer and Communication Systems*. Cambridge Computer Science Texts 24, Cambridge University Press, 1987.