RC21952 (98681) 6 February 2001 **Computer Science**

IBM Research Report

Feedback Control of a Lotus Notes Server: Modeling and Control Design

S. Parekh, J. L. Hellerstein, N. Gandi, D. M. Tilbury
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Feedback Control of a Lotus Notes Server: Modeling and Control Design[1]

N. Gandhi and D. M. Tilbury
The University of Michigan
Mechanical Engineering Department
Ann Arbor, MI 48109-2125
{gandhin, tilbury}@umich.edu

S. Parekh and J. Hellerstein
IBM T. J. Watson Research Center
30 Sawmill Parkway
Hawthorne, NY
{sujay,hellers}@us.ibm.com

Abstract

This paper considers the modeling and feedback control of a mail server running Lotus Notes^{TM}. Computing systems such as this typically have two competing control goals: maximize throughput and minimize response time. To achieve these goals, a control input (tuning control) is used to limit the number of users allowed to connect to the mail server at any one time. The measured output is the server queue length—the number of requests that are waiting to be processed. Because response time is a client metric and thus difficult to measure at the server, we formulate the control problem as tracking a reference server queue length. A linear input-output model of the system is identified experimentally and used to design an integral controller. Losses in the queue length measurement due to the fact that requests are only logged after they are served are accounted for by another linear model. Experimental results are presented showing the effectiveness of a low-gain controller and the saturation problems experienced by a high-gain controller. The paper concludes with a discussion of future work.

1 Introduction

In this technology driven era, there has been an explosion in both the size and number of computing environments required to meet the needs of today's workplace. Administrators of these complex systems are faced with an important challenge: ensuring that end user service level metrics (such as response times) as well as throughput objectives are maintained at acceptable levels. Tuning controls are parameters that adjust the way resources are allocated on the target system; they thus have a direct impact on the system's performance characteristics.

A commonly used approach to the aformentioned problem is to take an existing system, the target system, and then insert a controller that has access to the metrics and tuning controls of the target system. The controller then manipulates these tuning parameters to achieve service level objectives. Trade-offs that must be considered include maximizing the number of users that are allowed to connect to the system while minimizing the delays that result from an overloaded server. Currently, either controllers are manually tuned or a rule-based methodology is implemented to ensure that an acceptable level of performance is achieved. While considerable attention has been focused on the software mechanisms needed to implement closed-loop systems (e.g., instrumentation and controls), much less attention has been paid to the effectiveness of the controller.

One potential application of this research is the allocation of resources. For example, it is often necessary for internet service providers (ISPs) to make guarantees to clients about the maximum performance they can expect especially during peak hours. In cases like this, it becomes important to allocate bandwidth in such a way as to maximize the number of clients an ISP can take on while ensuring that service agreements are met. Of course, this can be accomplished by buying enough servers to process requests by all clients during peak hours. This is not the most efficient method, because servers will be underutilized during off-peak hours. By using online control, ISPs can dynamically allocate resources between clients to ensure that all service agreements are met, while at the same time the utilization of resources is maximized.

The long term goal of this research is to develop a methodology for constructing effective controllers for computing systems. Herein, we address this question in a specific context—a Lotus Notes^{TM} email server. A feedback control scheme is used to set the tuning controls on-line, using real-time data about system parameters and user demands [4]. The results show that there is a good potential for achieving system performance goals with minimal administrator intervention.

The outline of the paper is as follows. In Section 2, the

[1]This research was supported in part by IBM.

/

modeling process is described. A brief overview of the system in question is also included. Section 3 discusses controller design and provides analysis of experimental results. Finally, Section 4 summarizes the findings and outlines future work.

2 Modeling and System Identification

The computing system considered in this paper is a Lotus Notes mail server. Many different users can connect to this system simultaneously to read, send, and manipulate electronic mail. A request to the server from the user generates a number of remote procedure calls, or RPCs. These RPCs are queued at the server until they can be served. In some instances, the server may deny service to a new user, depending on the number of other users already connected to the system and the load they are generating.

The Notes server has many available tuning controls (parameters that affect the different resources utilized by the system). Some of these tuning controls must be fixed at installation time; others can be changed on-line while the server is operating. The tuning control considered as the control input in this work is `SERVER_MAXUSERS` (hereafter abbreviated as `MAX_USERS`), which sets the maximum number of users allowed to be connected to the server at any one time.

As stated earlier, the goal of applying feedback control to this system is to achieve certain service level objectives with minimal administrator intervention. These objectives may include limiting response times, maximizing throughput, or a combination of these conflicting goals. Hence, in order to implement control, a system output must be chosen that adequately captures whether or not these objectives have been met. For the purposes of the Notes server, queue length, the number of remote procedure calls waiting to be served, will be used to characterize the performance of the system. This metric provides a way to manage trade-offs between response times and throughput. In addition, queue length is easily measured from information recorded in the server log. Measuring response time and throughput often requires information from the end user, which may not be available to the server during regular operation.

A common approach in computer systems analysis is to form a system model using first principles [1, 5, 6]. However, for complex systems, considerable insight is required to develop a first principles model. In addition, first principles models that employ detailed characteristics of the target system may be quite sensitive to changes in the target system, such as those that occur with software releases. Thus rather than proceeding from first principles, this research uses an empirical approach. Obviously, the system in question is highly discrete and nonlinear. However, for the purposes of control, a linear system model was desired. This model is an approximation of the real system at best, but for control, approximate models often suffice. Data was logged from server runs to aid in the creation of a linear system model.

The experimental testbed includes a workload generator and product-level Notes server. The workload generator simulates the activity of multiple users by running copies of an identical script that sends commands to the server that generate RPCs. The Notes server, in turn, processes and logs the RPCs that are generated. In order to determine the effect of the tuning control `MAX_USERS`, the number of users allowed to connect to the server during a run was increased over fixed intervals of time. The server logs various information about each RPC at its departure including arrival time, response time, RPC type, and user number. From the information provided in the log, the queue length of each server run can be calculated. The offered load in the case of homogeneous users is the number of users trying to connect to the system. It is important that this that this parameter be distinguished from the carried load, which is the number of users *allowed* to connect to the server. When the offered load is greater than `MAX_USERS`, the carried load is the value of `MAX_USERS`. Key parameters for each run include the offered load, run length, and the values used for the tuning control.

As mentioned above, the tuning control utilized for this work was `MAX_USERS`. This parameter controls the number of users allowed to connect to the server; however, it does not limit the number of RPCs that can be generated by a single customer and hence does not directly affect the queue length. This presents an interesting problem if a linear system model is desired. If fewer users are allowed to connect to the system, it seems reasonable to expect that fewer RPCs would be generated and thus result in a smaller queue length. But does a linear model adequately capture the relationship between the tuning control, `MAX_USERS`, and the output of the system, queue length? The answer to this question was not clear at the onset. As a result, different models were created to try to capture the effect of `MAX_USERS` on queue length. Another issue is that the tuning control saturates. If the offered load is less than the control input, `MAX_USERS`, the control is inactive. Thus, for a fixed offered load, the control input will saturate at the level of the offered load. Increasing the control input above this level has no effect on the system. As a result, only the region where the control was active (where the offered load was greater than `MAX_USERS`) was considered for the purposes of system identification.

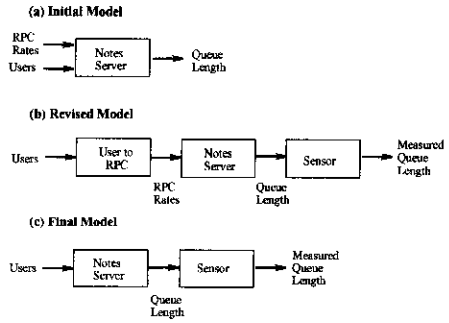

Figure 1: Block diagrams of different model structures.

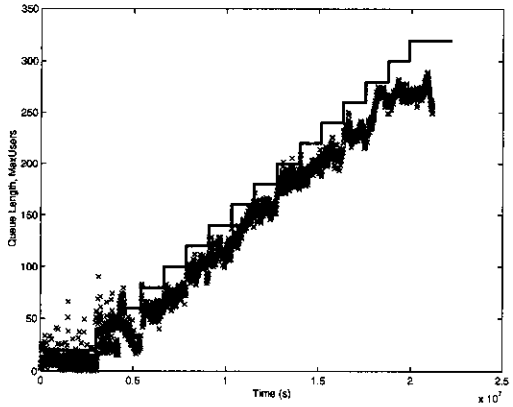

Figure 2: Response of queue length to step changes in MAX_USERS.

Initially, a multi-input model was considered as shown in Figure 1a. In addition to the control input, RPC rates were considered as additional uncontrollable but measurable (disturbance) inputs. It is clear that this model is not linear, because the RPC rates are dependent on the number of users connected to the system. This model was revised as shown in Figure 1b to include more transfer functions in the hopes of discerning a clear relationship between the control input and the system output, queue length. In this model, users were translated into RPC rates which in turn were translated into queue length. This model became increasing complex requiring the identification of over 50 transfer functions. After considering data from higher levels of offered load, a simplified model was considered as shown in Figure 1c. The motivation for this model can be seen in Figure 2, which plots the queue length of the Notes server where the offered load is 300 users and MAX_USERS is increased by twenty users every twenty minutes. The impact of MAX_USERS on the queue length is clear, suggesting that a linear model between the control input and the queue length will suffice.

A first order ARX model was fit to server data using a linear least squares method, where the input to the system was the tuning control parameter, MAX_USERS, and the output was the performance metric, queue length [2]. The resulting transfer function for the Notes engine denoted by $N(z)$ is given in (1). The R^2 value for the model is 97.6 indicating a very good model fit.

Table 1: Models coefficients and fits for $M(z)$

Delay	R^2	a_1	b_0	b_1
0	75.5	0.6371	0.1692	-0.1057
1	83.7	0.7991	0.7182	-0.6564
2	91.2	0.9237	0.9388	-0.9092

$$N(z) = \frac{QL(z)}{U(z)} = \frac{0.4709}{z - 0.4261} \quad (1)$$

Because of the way data is logged by the server, an additional model was required to account for measurement losses. RPCs are logged when they leave the server. This means that when a queue length measurement is taken by reading the server log, all the RPCs that are currently waiting to be served will not be included in the queue length. At first glance, this loss may seem insignificant. However, as load on the server is increased, RPCs spend more time being processed and hence there is a greater chance that they will be in service when measurements are taken. This presents an interesting dilemma. If the the measurement is delayed so that longer running RPCs can complete their execution, then more RPCs will be counted in the queue length measurement resulting in more accurate model fits. However, this delay will also change the closed loop dynamics of the system and probably limit the effectiveness of the control. Data from actual server runs was used to derive a linear relationship between the actual and measured queue lengths using different delays. The resulting "measure" transfer function denoted by $M(z)$ is given in (2). The coefficients of the transfer function for various values of the delay are given in Table 1.

$$M(z) = \frac{QL_{meas}(z)}{QL(z)} = \frac{b_0 z + b_1}{(z - a_1)z^d} \quad (2)$$

3 Control Design

As noted before, the tuning control saturates once MAX_USERS exceeds the offered load. In addition, the control saturates at zero because it does not make intuitive sense to have a negative number of users allowed to connect to the system. To help prevent saturation from occurring at the bottom, an integral control scheme was chosen. Hence, the controller is of the form $K_i * D(z)$ where K_i is the control gain and $D(z)$ equals $\frac{z}{z-1}$ [3]. Figure 3 shows the block diagram of the closed loop system.

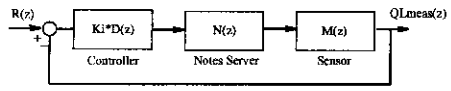

Figure 3: Block diagram of closed loop system with integral control.

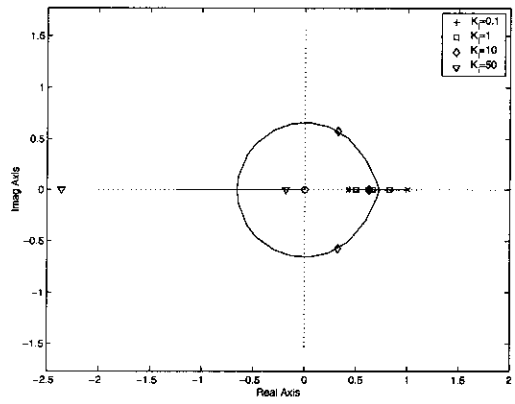

Figure 4: Root locus with integral control ($d = 0$).

Figure 4 shows a plot of the root locus of the system combined with the integral controller, $D(z)N(z)M(z)$, when $M(z)$ included no measurement delay. When $K_i = 50$ the poles are outside the unit circle causing the closed loop system to be unstable. In fact, when K_i is greater than 37 the closed loop system is unstable. Figure 5 shows a plot of the root locus of the system combined with the integral controller when $M(z)$ has a measurement delay of 2. When K_i is greater than 0.8, the closed loop system is unstable. As expected, adding the measurement delay adversely affects closed loop performance. The viable region for K_i is radically reduced.

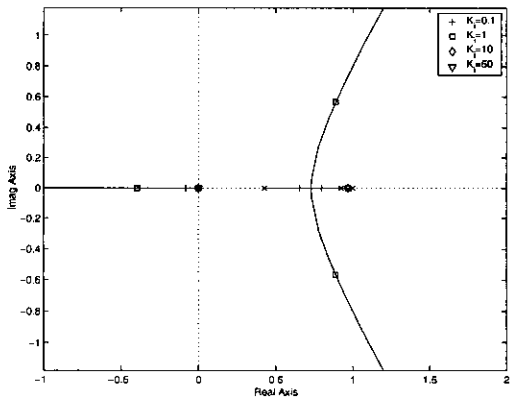

Figure 5: Root locus with integral control ($d = 2$).

4 Experimental Results

In addition to the workload generator and product level Notes server used for system identification, a sensor (running on the Notes server) and controller (running on a third machine) were added to the testbed to study the effects of feedback control on a real system with a synthetic workload. In all runs, the offered load was maintained at 200 users and a sample time of 60 seconds was used for control. Although smaller sample times might result in better control performance, the server must run a script each sample time to measure the queue length from the server log file. This increases the load on the server and negatively impacts its performance. This trade-off between sample time and server load should be investigated further.

Various gains were tested to capture a broad spectrum of responses, ranging from slow to fast. In addition, the effect of the measurement delay on closed loop response was observed.

Figures 6 through 10 show server runs for two different control gains. In each figure, the top plot shows the queue length on the server with respect to time and the bottom plot shows the tuning control value over the same span of time. In the top plot, the solid line is the reference queue length and the line marked with x's is the actual queue length.

Figure 6 shows a run where the control gain, K_i, was set to 0.1. The plot of the queue length shows two clear regions, one for each value of the reference. In both these regions, the queue length fluctuates around the reference value. However, there is a large initial transient due to the slow response. It takes nearly 2000 seconds for the queue length to start "following" the reference. Figure 7 shows a run where the control gain, K_i, was set to 1. There is a marked difference in the response. The queue length still fluctuates around the reference value. However, the initial transient time is reduced by almost 75 percent. Comparing the plots of the control values for the two gains, it is clear that the larger gain induces some oscillations. However, there is no significant increase in the variability of the queue length so these oscillations do not pose a problem.

Another interesting phenomena is the effect of the delay incorporated into the queue length measurement on control performance. Studying the root locus of the two systems, it is obvious that instability will occur for certain values of K_i when a delay is added to the system, whereas the system with no delay will be stable using the same gain. In this saturated system, there is a limit on how high the queue length can get and hence there is no physical means to observe closed loop instability. However, it is interesting to compare the closed loop re-

sponse of a system with measurement delay to one with no delay. Figure 8 shows a run where the control gain, K_i, was set to 0.1 and a measurement delay of two sample periods was added to the system. When $K_i = 0.1$, the closed loop system with the delay is still stable. Comparing Figure 6 and Figure 8, it is obvious that the added accuracy of the model fit does not increase the effectiveness of the controller. The initial transient of the system with the delay is slightly smaller, probably because it has faster closed loop poles. Figure 9 shows a run where the control gain, K_i, was set to 1 and a measurement delay of two sample periods was added to the system. When $K_i = 1$, the closed loop system with the delay is unstable. Comparing Figure 7 and Figure 9, it is clear that the added delay has hindered the performance of the controller. Oscillations dominate the queue length in Figure 8, hinting at the instability that is caused by adding a delay in the system.

Finally, the effect of saturation on control performance is investigated. Figure 10 shows a run where the control gain, K_i, was set to 50. This gain renders the closed loop system unstable. In this saturated system, there is no mechanism for the queue length to grow without bound. However, using this gain, the effect of saturation can be more closely examined. Inspecting the plots, it is clear that the tuning control value is saturated for the entire run. As mentioned earlier, saturation occurs for two reasons: (1) `MAX_USERS` cannot take on negative values and (2) when `MAX_USERS` is greater than the offered load, it has no effect. The queue length never converges to the reference. In fact, the step change in the reference has no effect on the queue length. Instead, the variability of the queue length is greatly increased and can be entirely attributed to the saturation of the control. Obviously, this saturation effect is undesirable and hence lower values for the integral gain would be preferred.

5 Conclusions and Future Work

The prevalence of IT services in today's world has created great interest in devising automated techniques to achieve service level objectives in computing systems. As mentioned in the introduction, the long term goal of this research is to develop a methodology for constructing effective controllers for computing systems. One possible approach is to apply the principles of feedback control, using available tuning controls and measurable metrics that are representative of system performance. In this paper, this approach was attempted on a specific system—a Lotus Notes^{TM} email server. A system model was developed using a statistical approach to system identification. Using this model, an automatic feedback control scheme was created to dynamically change the number of users allowed to connect to

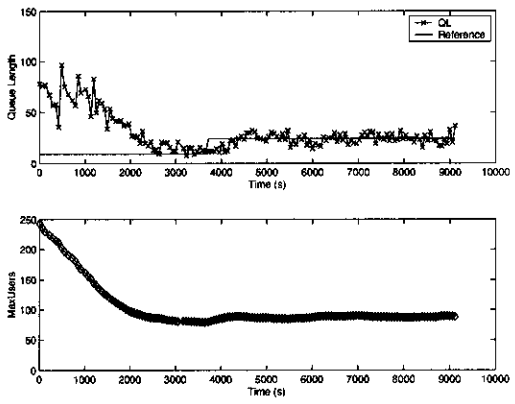

Figure 6: Comparing the reference to actual queue length for $K_i = 0.1, d = 0$, and an offered load of 200 users. Note the long initial transient region.

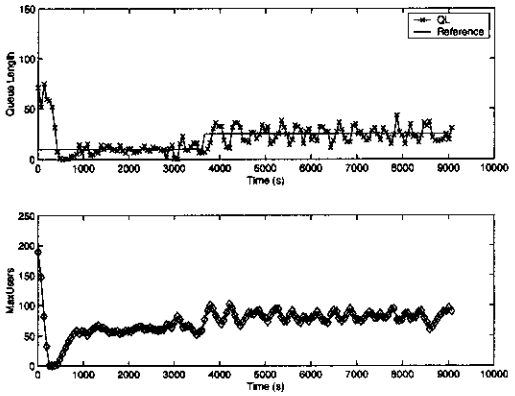

Figure 7: Comparing the reference to actual queue length for $K_i = 1, d = 0$, and an offered load of 200 users.

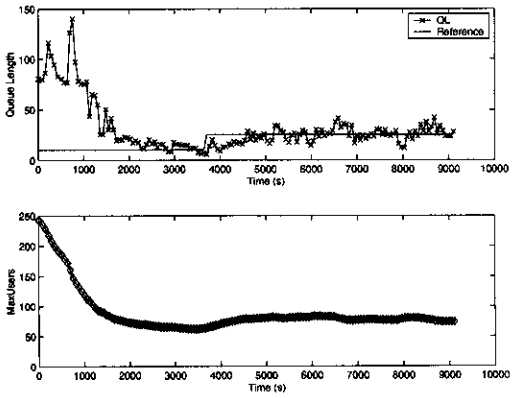

Figure 8: Comparing the reference to actual queue length for $K_i = 0.1, d = 2$, and an offered load of 200 users. The performance is similar to the case with no delay.

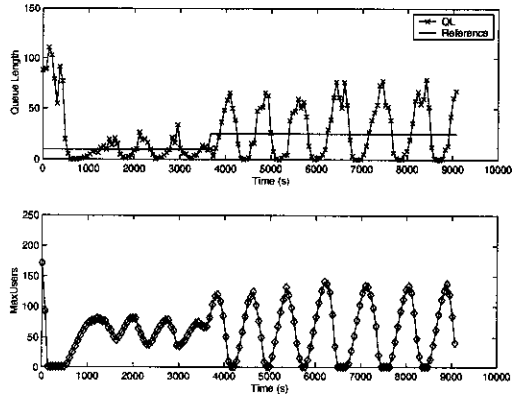

Figure 9: Comparing the reference to actual queue length for $K_i = 1, d = 2$, and an offered load of 200 users.

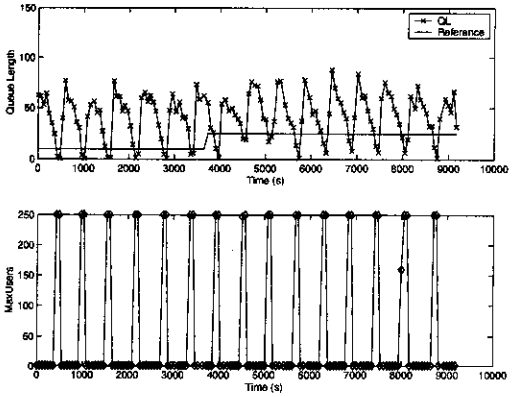

Figure 10: Comparing the reference to actual queue length for $K_i = 50, d = 0$, and an offered load of 200 users. Note the significant variation in the queue length due to saturation of the control input.

the server, using real-time queue length measurements. The results of the preliminary integral control scheme show that there is a good potential for achieving system performance goals.

The injection of an integral controller into the Notes system enabled the queue length to reasonably follow a given reference trajectory consisting of step inputs. Because of the stochastic nature of the system in question, it is not expected that the queue length will track the reference perfectly. However, it would be undesirable if the closed loop system exhibits more variability or oscillations than the open loop system. This appears to be the case when large values are used for the integral gain because the controller saturates.

There are a number of issues that are yet to be addressed. The model constructed is valid only in the region when `MAX_USERS` is less than the offered load (otherwise the tuning control is inactive). As a result, saturation occurs when the control input exceeds the offered load. Furthermore, there is saturation due to constraints intrinsic to the `MAX_USERS` parameter (it cannot be negative). This saturation effect limits the value of the controller gain and thus limits the closed loop response. The trade-off between sample time and server load should be investigated, and other tuning controls should be considered. Future work will focus on resolving these outstanding issues, as well as evaluating more complex control schemes.

Again, it is desired that the methodology used for the Notes server can be extended to other computing systems such as web servers, database servers, and eventually, distributed heterogenous multi-server systems. Because the system model was created using an empirical approach, it is believed that it more generally applicable then the conventional first principle approach. However, because computing systems are highly discrete and nonlinear, the question becomes will a linear model suffice for purposes of control? Another direction of future work will be to apply the methodology utilized in this paper to other service level management situations. This will help elucidate whether a linear model can be used to describe and control computing systems in a broader context.

References

[1] B. Li and K. Nahrstedt. Control-based middleware framework for quality of service applications. *IEEE Journal on Selected Areas in Communication*, 17(9):1632–1650, September 1999.

[2] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1999.

[3] K. Ogata. *Modern Control Engineering*. Prentice Hall, 3rd edition, 1997.

[4] S. Parekh, N. Gandhi, J. Hellerstein, D.M. Tilbury, J. Bigus, and T.S. Jayram. Using control theory to achieve service level objectives in performance management. In *Proceedings of International Conference on Integrated Network Management (IM 2001)*, May 2001.

[5] M. H. Shor, K. Li, J. Walpole, D. C. Steere, and C. Pu. Application of control theory to modeling and analysis of computer systems. In *Proceedings of the Japan-USA-Vietnam RESCCE Workshop*, June 2000.

[6] D. Steere, M. H. Shor, A. Goel, J. Walpole, and C. Pu. Control and modeling issues in computer operating systems: Resource management for real-rate computer applications. In *Proceedings of the IEEE Conference on Decision and Control*, Sydney, December 2000.