Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Feedback Control of Internet Applications involving the Tracking of Dynamic Data

Shweta B. Shah, Kannan M. Moudgalya *
Krithi Ramamritham **

* *Dept. of Chemical Engineering, IIT Bombay, Powai, Mumbai 400076, India, (email: shwetash@andrew.cmu.edu, kannan@iitb.ac.in)*
** *Dept. of Computer Science and Engineering, IIT Bombay, Powai, Mumbai 400076, India, (e-mail: krithi@cse.iitb.ac.in)*

Abstract: Control of computing systems is different in many ways from the control of conventional systems. The quality of service problem in internet applications, that is, reducing the network overhead without affecting the quality, has been studied in detail. Network overhead can be measured in terms of the number of refreshes of the dynamic data. Use of randomly generated input vectors allows early use of feedback control, thereby lowering the network overhead. Identification using moving time windows helps obtain locally accurate models, thereby increasing the efficacy of the control scheme. Online tuning of the maximum permissible change in the input, namely maximum time till next refresh, has been shown to improve the control performance. Because this approach is not controller specific, it would be of use to all controllers. Two different ways of formulating the linear quadratic regulator objective function have been studied. Using the reciprocal of TTR (time to refresh) as the control input makes the control weight sensitive over a larger range, compared to using TTR itself as the input.

Keywords: feedback control; computing systems; quality of service; pull based approach; identification; moving window; online tuning; weight selection in LQR control.

1. INTRODUCTION

The problems in traditional areas of control involve plants that have concrete shapes and sizes. These plants are generally built on the principle of conservation of quantities, such as, mass, momentum, energy and voltage. In contrast to the plants in the traditional control areas, the "plants" in the Information Technology (IT) area are at best conceptual, as they typically involve only computational logic. The plants that arise in the IT sector also have to satisfy requirements, such as, stable performance and agile response in the presence of changing loads and uncertainties. Although the computing services are getting sophisticated, there are problems, such as, excessive load due to increased usage, inefficient memory allocation and CPU usage, delay in transfer of information and inefficient service provision.

The use of feedback control techniques in the IT industry is nascent. Although computing systems have been tuned traditionally using heuristics, control theoretic solutions are on the rise, mainly because of the underlying rigour and the ability to directly address issues such as, stability, short settling times, and accurate regulation [Diao et al., 2005].

Achieving service-level objectives, optimizing resources, regulating services and rejecting disturbances have been identified to be some difficult problems in the field of computing to which computing theory can be applied beneficially [Hellerstein, 2004]. Abdelzaher et al. [2002],

⋆ Shweta Shah is currently with Carnegie Mellon University

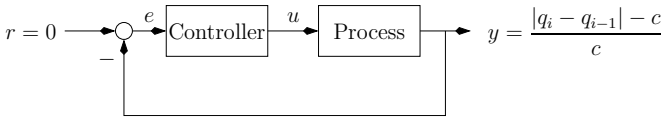

Fig. 1. Output feedback loop

Stankovic et al. [1999] and Stankovic et al. [2001] have used feedback control to achieve service-level objectives. Optimization of resources has been studied by Skadron et al. [2002], Kang et al. [2002], Diao et al. [2002], Gandhi et al. [2002], Majumdar et al. [2003, 2004] and Lu et al. [2005]. Regulation of services by minimizing the delays has been studied by Lu et al. [2003], Henriksson et al. [2004], Abdelzaher et al. [2002] and Lu et al. [2006]. Rejection of disturbance has been studied by Benmohamed and Meerkov [1993] and Stankovic et al. [1999].

2. PROBLEM STATEMENT

In this work, we study the system proposed by Majumdar et al. [2003, 2004]: The control objective is to poll a changing data source least often and yet ensure that the source is polled before the value changes by more than $\pm c$, see Fig. 1. The variable c is known as coherency. Satisfaction of coherency c is known as the quality of service in this context. The percentage of time during which the coherency requirement is violated is known as infidelity. If polling of data trace is done least number of times, the network overhead is said to be the minimum. Polling of data is also known as *refresh*.

If t_i and t_{i-1} are sampling instants, the sampling period $t_i - t_{i-1}$, also known as *time to refresh* (TTR), is the input or control effort, u. If q_i and q_{i-1} are the corresponding data values, the plant output is defined as

$$y = \frac{|q_i - q_{i-1}| - c}{c}, \tag{1}$$

which we would like to maintain at zero. While late polling results in loss of quality, early polling does not. Nevertheless, the latter results in increased network overhead. The objective of the control problem studied by Majumdar et al. [2003, 2004] is to obtain low network overheads, simultaneously with low infidelity levels. Achieving this objective at low coherencies is more difficult compared to high coherencies. The updating activity of a cache server so as to meet the requirements of many users is an example of this application.

A portfolio query is concerned with decision making based on a weighted sum of several data items. Because the data items could come from different sources, this is a multi input single output control problem. Multiple portfolio query involves polling several streams so as to satisfy different requirements, possibly from many users. This is a multi input multi output control problem [Majumdar et al., 2004]. It is clear that this class of problems offers a rich application area for control theroists.

The problem under study is different from conventional ones in many ways. The time till the next poll is the input variable and a function of the value of the data item at that time is the output variable. This is different from what we see in the usual control problems where time is an independent variable. This problem does not have an analog control equivalent but can be solved as a naturally occurring discrete time control problem [Moudgalya, 2007]. Because it is a computing system, the constraint on the control input could be based on performance (constraining maximum TTR), although it could depend on implementability: for example, negative TTR cannot be implemented, as it implies polling back in time. Being computing systems, it is often possible to define the variables and hence the resulting transfer functions and objective functions in more than one way, as we explain below. For more details on this problem, the reader is referred to Majumdar et al. [2003].

The above factors call for a detailed study so as to understand well the underlying processes in order to come up with effective control strategies. Because of the application focus, namely tracking dynamic data sources, in the work of Majumdar et al. [2003, 2004], some important questions that are of interest to the control field are not addressed.

Although Majumdar et al. [2003, 2004] have used a fixed model, a moving window based approach should give locally correct models, given the time varying nature of the underlying process. For identification of the process model, they have used a deterministic input, which is a repetition of the vector (1,2,3,4,5). We should expect a random input vector to work better. Both of these should allow us to identify the plant model more quickly and thus to apply the control based polling mechanism early, as compared to the approach of Majumdar et al. [2003, 2004]. While they adaptively tune the proportional controller gain, we demonstrate the benefits of tuning of TTR_{max}, which also happens to be controller independent. We also present two different ways of formulating the objective function in the LQR framework and compare the resulting performance.

Table 1. Data traces

Trace	Min	Max	Std. deviation	Type
IBM	106.64	108.35	0.3290	Fast
CSCO	18.95	19.82	0.2331	Medium
LU	6.30	6.41	0.0230	Slow

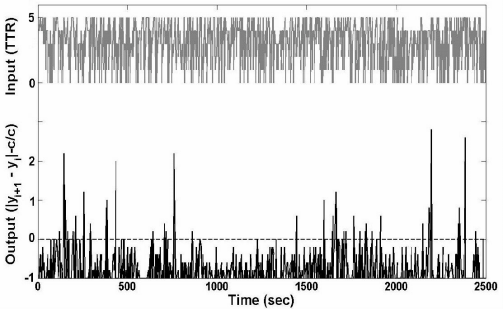

Fig. 2. A typical input-output plot, IBM trace, $c = 0.05$

We have used three traces of real time stock quotes, given in Table 1, to test our hypotheses. The fast trace is the most difficult of them. In view of this, we present mainly the results obtained for the fast trace.

3. MODEL IDENTIFICATION

As mentioned earlier, there are no first principles model for the system under study. In view of this, we build a model from input-output data. We identify the model with data refreshed initially. The minimum time interval between two refreshes has been taken as one second.

The input vector for identification is selected as $(u_{k_1}, u_{k_2}, \ldots, u_{k_N})$, where, $u_{k_i} = t_{k_i} - t_{k_{i-1}}$ and u_{k_j} is a random integer in the interval 1 to 5. Suppose that we obtain the following specific vector, $(3, 2, 4, 3, 1, 5, \ldots)$. Then, the data acquired at time instants of $(t_k, t_{k+3}, t_{k+5}, t_{k+9}, t_{k+12}, t_{k+13}, t_{k+18}, \ldots)$ are used to create the input data vector. Twenty randomly selected integers in the range of 1 to 5 are strung together to form the input vector in the current work. The mean of u_{k_i} is 3 and hence, we use approximately the data from the first sixty seconds for identification. The corresponding output vector is calculated using (1). A typical data set is shown in Fig. 2.

It is important to note that the inputs u_{k_1}, u_{k_2}, etc. correspond to non overlapping intervals. This mode of selection provides sequential information. This selection also provides a random nature to the input. In contrast, Majumdar et al. [2003, 2004] have used a deterministic pattern of a repetition of the vector $(1, 2, 3, 4, 5)$. Random inputs have a better chance of exciting the modes of the system and hence are more suitable for identification [Ljung, 1999]. Moreover, random selection can also reduce the number of data points required.

Auto Regressive models with eXogeneous inputs (ARX models) have been used to identify the plant model, using the input-output data. Matlab Identification Toolbox [Ljung, 1999] has been used for this purpose. Because uniformly spaced sample number is the independent varible

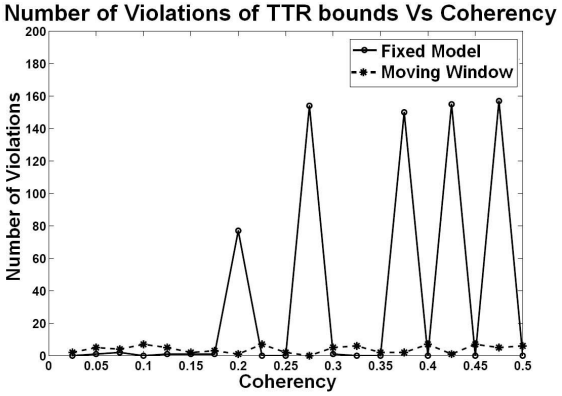

Fig. 3. TTR violations in fixed and moving window approach

(and not time), it is possible to use these algorithms. We have fitted the data with second order models. Larger order models did not result in any appreciable improvement. Moreover, the uncertainty in the model parameters increased with order.

All the traces used in this work are of 10,000 seconds long. As mentioned earlier, twenty data points, polled during the first about sixty seconds, are used for identifying the model. In contrast, Majumdar et al. [2003, 2004] have used the data refreshed during the first 1,500 seconds for identification. The advantages in the current scheme are:

(1) Need a smaller number of points to build the model.
(2) Because a smaller number of points is used for initial identification and for future updates, (a) the local behaviour of the model is well brought out. (b) one can apply feedback control based refreshing techniques earlier, resulting in a smaller network overhead, compared to that of Majumdar et al. [2003, 2004].

4. MOVING WINDOW APPROACH

In this section, we evaluate the efficacy in using a moving window approach.

Using the model identified with the first twenty polls in the feedback loop, we have made subsequent polls, using which, the initial model is updated. This is clearly a case of closed loop identification, in which, the data obtained in the presence of a feedback controller is used for modelling Ljung [1999]. We have used a window length of 50 polls for identification of a new model. There are times when we could not determine a model in a new window, as it can happen in closed loop identification. We have used the previous model for the current window as well, under these conditions.

As expected, the moving window approach has performed a lot better than the fixed model. A comparison of the number of times the TTR limits is exceeded by the fixed model, and that for the moving window approach, is given in Fig. 3. In the moving window approach, the TTR limits are exceeded only occasionally. In contrast, the fixed model performs poorly: as a matter of fact, in some experiments, almost half of the calculated TTRs exceed the limits. This is clearly unacceptable and as a result, we use the fixed model in the rest of this paper, except in Section 6.

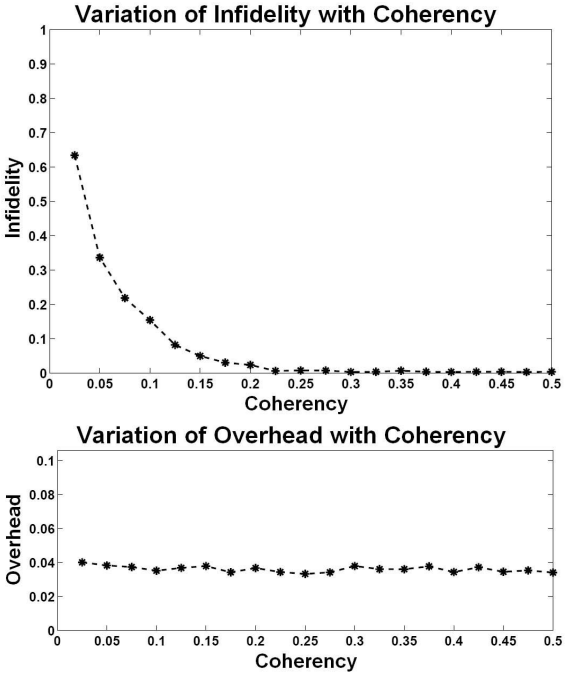

Fig. 4. Infidelity and overhead as a function of coherency (IBM)

The performance of the moving window approach is brought out by the infidelity and the overhead plots of Fig. 4. In the top plot of Fig. 4, we show infidelity as a function of coherency. In the bottom plot, we show overheads as a function of coherency c. The overheads are a ratio of actual number of polls to the maximum possible number. In these plots, we have included the initial polls required for identification as well. The plot shows that a total of 4% of total number of points of 10,000 have been polled, irrespective of the coherency c. We have not used the detuning approach of Majumdar et al. [2003] in these simulations. Whenever the TTR limits are exceeded, TTR is chosen as the corresponding limiting value, however.

From these plots we see that for small coherencies, the infidelity is high. Small coherency is equivalent to a tight control requirement, which results in a large number of violations. In contrast, the violations are minimal for large coherency values, as can be expected.

To improve the performance at low coherencies, we explore the detuning approach of Majumdar et al. [2003]. This involves reducing the proportional controller gain obtained with a Ziegler Nichols heuristics [Moudgalya, 2007] by a factor of ρ, $0 < \rho < 1$, after every M consecutive violations of the TTR limits. Infidelity and overhead plots, obtained with $M = 10$ and $\rho = 0.9$ are shown in Fig. 5. Infidelity for the smallest value of c in Fig. 5 is smaller than that in Fig. 4. Nevertheless, it is still high, of the order of 45%, even after detuning the controller. Because of this shortcoming, we have not used the above procedure to detune proportional controller gains in the rest of this paper.

5. CONSTRAINING THE MAXIMUM TTR

The reason why the infidelity is high for low coherency values in Fig. 5 is that TTR values, although within the

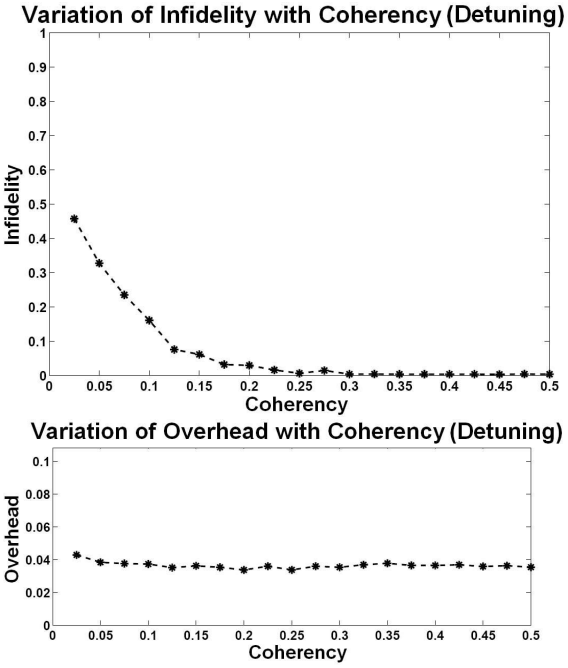

Fig. 5. Infidelity and overhead as a function of coherency with detuning of controller (IBM)

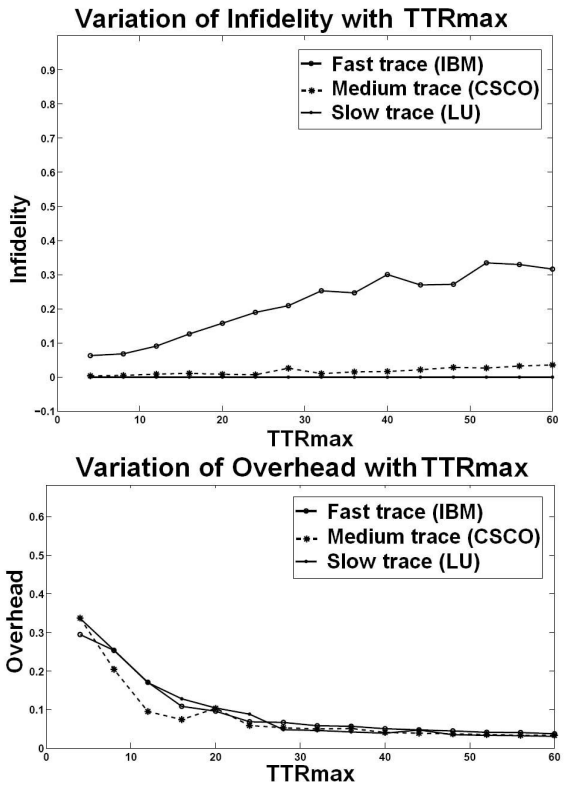

Fig. 6. Infidelity and overhead as functions of TTR_{max} for $c = 0.05$

allowable limits, are still high at times. To check this hypothesis, we have carried out experiments with different TTR_{max} values. The coherency c has been kept at 0.05 in all these simulations. The resulting plots for all three traces are shown in Fig. 6. From this figure, it is clear that we have achieved low infidelity, even for fast traces. Of course, it is inefficient to keep TTR_{max} constant at a small value

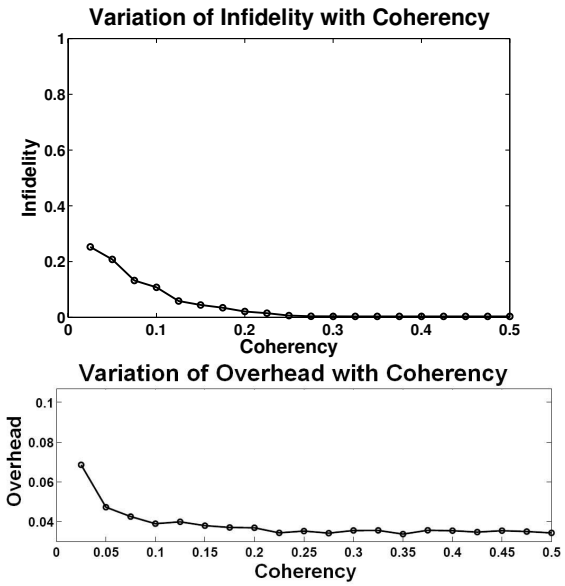

Fig. 7. Infidelity and overhead for tuning of TTR_{max} (IBM)

throughout, as this would result in larger overheads than required. We are interested in finding a control law that will help realize the best aspects of both approaches: want low infidelities at low c values while keeping the overheads also low.

We propose an adaptive detuning and retuning of the maximum allowable TTR: If the infidelity in the last P polls is 1, the maximum allowable TTR be chosen as 1. If on the other hand, if the infidelity is 0, the maximum allowable TTR be set as 60. For intermediate values, the following linear interpolation formula is used:

$$TTR_{max} = -59 \times \text{infidelity} + 60$$

The results of simulation with $P = 50$, same as the length of the moving window used earlier, are shown in Fig. 7.

From this figure, it is clear that the infidelity value for the lowest coherency has come down quite a lot, compared to Fig. 5: from 45% to 25%. The network overhead also comes down dramatically, compared to Fig. 6. From Fig. 6, we see that for IBM trace, infidelity of 0.2 corresponds to a network overhead of 0.1. In contrast, from Fig. 7, we see that for $c = 0.05$, infidelity and network overheads are 0.2 and 0.045, respectively. In other words, the detuning approach of TTR_{max} brings down the network overhead from 0.1 to 0.045 with everything else remaining identical.

Selection of P is not straight forward, however. Supposing that we choose it to be a small number, say 10. Suppose that the last ten polls occur at every second, which is the minimum allowable time interval. As per the above formula, the next allowable TTR_{max} would become 60! If on the other hand, if we choose P to be large, say 200, the infidelity would not be as small as that with a fixed, small, TTR_{max}: we have to wait for infidelity violations in the last 200 polls to detune TTR_{max}. As mentioned earlier, we have chosen P to be 50, the same as the size of the moving window and have obtained good performance.

This approach uses the infidelity observed only from the past refreshes. Note that it does not cost much to send

the intermediate updates, once the client connects to the server. As a result, whenever a poll is made, it is possible for the server to send the current value as well as all the values not polled by the client since the previous poll. Using these intermediate data, we can calculate the past infidelity more accurately.

6. COMPARISON OF LQR STRUCTURES

We conclude this article with a comparison of two different weighting schemes in linear quadratic regulator (LQR) design. The LQR is obtained by minimizing the following objective function J in (2).

$$J = \sum_{k=0}^{N} e(k)^T Q e(k) + u(k)^T R u(k) \qquad (2)$$

Application of this procedure to the problem at hand is not straight forward, because, the control variable TTR is the reciprocal of the control effort: as TTR increases, work overhead decreases. One way to address this difficulty is to choose $u = 1/TTR$. There could be two difficulties with this choice. Firstly, this will make the plant transfer function even more nonlinear - the transfer function has to be estimated using the reciprocal of the measured value TTR. It is also not clear whether the choice of $1/TTR$ is extendible to multivariable systems, the main motivation for using LQR. An alternate approach is to choose the control variable as $u = TTR$. If the communication overheads are high, we would increase u, for which, we would reduce the control weight R.

In order to explore the pros and cons of these approaches, we carried out simulations for both choices of u. The detuning approach of the proportional controller was not used in these simulations: whenever there is a violation of the condition that TTR should be in the range of $[1, TTR_{max}]$, the corresponding limit is used. While there is a clear tuning guideline in the case of proportional controllers, there is no such help in choosing Q and R. We selected Q to be an identity matrix and explored a large range of values for R.

The fast trace of IBM has been chosen for these simulation studies with the coherency as $c = 0.05$. The simulation has been carried out for both types on inputs, $u = TTR$ and $u = 1/TTR$. The infidelity and overhead have been observed while varying R.

The results of a set of experiments with $u = 1/TTR$ are shown in Fig. 8, where we have plotted infidelity, which is a measure of x and overhead, which is a measure of TTR, as functions of $\log R$. In order to accommodate a large range of R, log scale has been used.

In Fig. 8, for small values of R, u becomes large, resulting in small TTR. For extremely small values of R, TTR gets selected close to 1, which is equivalent to polling most of the data. This results in infidelity being close to zero, simultaneously with overheads close to 1. The horizontal lines obtained for small values of R illustrate this idea. If R is chosen extremely large, small u, large TTR and hence, small overheads and large infidelity are attained. This situation is once again depicted by the horizontal lines at the right hand of the plots in the figure. Of course, the extreme values of R are not of interest to us. Only the

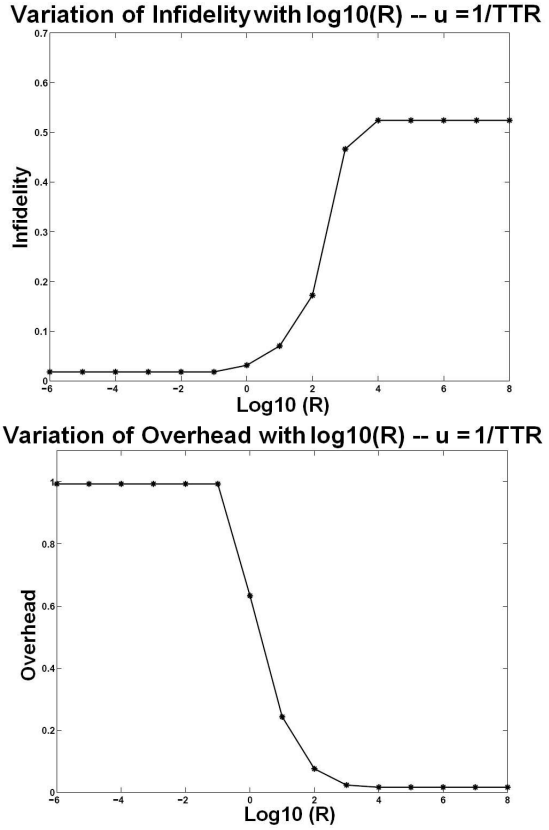

Fig. 8. Changes with R for $u = \frac{1}{TTR}$ (IBM, $c = 0.05$)

intermediate values, where R is a sensitive parameter, are useful to us.

Next, we repeat the calculations for the case of $u = TTR$ and plot the results in Fig. 9. As expected, the results of low and high values of R are reversed, as compared to the case of $1/TTR$. Here also, only the intermediate values of R are of interest.

From control implementation point of view, there is an important difference between the two selections of u: R corresponding to $1/TTR$ is sensitive over a larger range, as compared to that for TTR: When $u = TTR$, the range of suitable R is small (10^{-2},10^{-1}). When $u = 1/TTR$, the range of suitable R is larger (10^{0},10^{4}), resulting in a larger working range of R. In view of this, the use of $1/TTR$ as the control variable seems to have additional benefits. Similar results have been obtained for other traces as well.

7. CONCLUSION

In this paper, we have addressed the problem of efficient tracking of changes in dynamic data through the internet using pull based algorithms and a control theoretic approach based on ARX models.

It has been shown that a random selection of data points help identify the model with less number of points, thereby increasing the efficiency in dissemination. A moving window approach has been shown to be better than that with a fixed window.

It has been shown that using a constant TTR_{max} is not a good idea: while small values result in large network

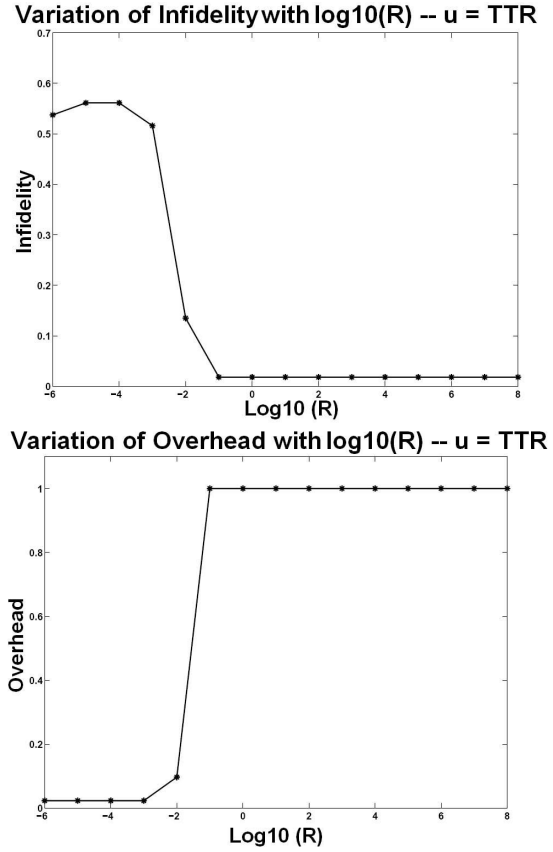

Fig. 9. Changes with R for $u = TTR$ (IBM, $c = 0.05$)

overheads, large values give rise to large infidelity. A simple procedure to tune TTR_{max} is proposed and is found to yield good results.

Two different ways to form the objective function in LQR framework is proposed. The choice of $u = 1/TTR$ results in a conventional formulation, simultaneously with a large range over which the control weight is sensitive. This choice, however, would result in the transfer function being nonlineard. The choice of $u = TTR$ results in exactly opposite results.

The problem solved in this work is effective tracking of dynamic data with the use of feedback control. This is quite different from the prediction of profiles, for example, prediction of future stock prices. It is the feedback control that makes this problem tractable.

A simple model structure has been used in this study, as the focus of this work is to understand the identification related issues from an application domain. Future work should focus on model orders and also closed loop identification methodologies.

REFERENCES

T.F. Abdelzaher, K.G. Shin, and N. Bhatti. Performance guarantees for a web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):80 –96, Jan 2002.

L. Benmohamed and S.M. Meerkov. Feedback control of congestion in packet switching networks: The case of a single congested node. *IEEE/ACM Transactions on Networking*, 1(6):693 –708, December 1993.

Y. Diao, N. Gandhi, J.L. Hellerstein, S. Parekh, and D.M. Tilbury. Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server. *Network Operations and Management Symposium*, pages 219–234, April 2002.

Y. Diao, J.L. Hellerstein, S. Parekh, R. Griffith, G.Kaiser, and D. Phung. Self-managing systems: A control theory foundation. *International Conference and Workshops on Engineering of Computer-Based Systems*, pages 441 –448, April 2005.

N. Gandhi, D.M. Tilbury, Y. Diao, J. Hellerstein, and S. Parekh. Mimo control of an apache web server: Modeling and controller design. *American Control Conference*, 6:4922 –4927, 2002.

J.L. Hellerstein. Challenges in control engineering of computing systems. *American Control Conference*, 3: 1970 –1979, 2004.

D. Henriksson, Y. Lu, and T. Abdelzaher. Improved prediction for web server delay control. *16th Euromicro Conference on Real-Time Systems*, pages 61 –68, 2004.

K.D. Kang, S.H. Son, J.A. Stankovic, and T.F. Abdelzaher. A qos-sensitive approach for timeliness and freshness guarantees in real-time databases. *14th Euromicro Conference on Real-Time Systems*, pages 203–212, 2002.

L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Inc., Upper Saddle River, NJ, 2nd edition, 1999.

C. Lu, Y. Lu, T.F. Abdelzaher, J.A. Stankovic, and Sang Hyuk Son. Feedback control architecture and design methodology for service delay guarantees in web servers. *IEEE Transactions on Parallel and Distributed Systems*, 17(9):1014 –1027, 2006.

C. Lu, X. Wang, and X. Koutsoukos. Feedback utilization control in distributed real-time systems with end-to-end tasks. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):550 –561, June 2005.

Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Li. Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers. *Real-Time and Embedded Technology and Applications Symposium*, pages 208–217, May 2003.

R.K. Majumdar, K.M. Moudgalya, and K. Ramamritham. Adaptive coherency maintenance techniques for time-varying data. *RTSS*, pages 98–107, December 2003.

R.K. Majumdar, K. Ramamritham, R.N. Banavar, and K. Moudgalya. Disseminating dynamic data with qos guarantees in wide area network: A practical control theoretic approach. *Real-Time and Embedded Technology and Applications Symposium*, pages 510–517, May 2004.

K. M. Moudgalya. *Digital control*. John Wiley & Sons, Chichester, 2007.

K. Skadron, T. Abdelzaher, and M. Stan. Control-theoretic techniquesand thermal rc modeling for accurate and localized dynamic thermal mangement. *International Symposium on High Performance Computer Architecture*, pages 17 –28, Feb 2002.

J.A. Stankovic, Tian He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback control scheduling in distributed real-time systems. *RTSS*, pages 59 –70, December 2001.

J.A. Stankovic, C. Lu, S.H. Son, and G. Tao. The case for feedback control real-time scheduling. *11th Euromicro Conference on Real-Time Systems*, pages 11 –20, 1999.