Adaptive Coherency Maintenance Techniques for Time-Varying Data

Ratul kr. Majumdar, *Kannan M. Moudgalya* †and *Krithi Ramamritham*

Laboratory for Intelligent Internet Research
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
{ratul, krithi}@cse.iitb.ac.in.

†Department of Chemical Engineering
Indian Institute of Technology Bombay
kannan@che.iitb.ac.in

Abstract

Often, data used in on-line decision making (for example, in determining how to react to changes in process behavior, traffic flow control, etc.) is dynamic in nature and hence the timeliness of the data delivered to the decision making process becomes very important. The delivered data must conform to certain time or value based application specific inconsistency bounds. A system designed to disseminate dynamic data can exploit user-specified coherency requirements by fetching and disseminating only those changes that are of interest to users and ignoring intermediate changes. But, the design of mechanisms for such data delivery is challenging given that dynamic data changes rapidly *and* unpredictably*, the latter making it very hard to use simple prediction techniques. In this paper, we address these challenges. Specifically, we develop mechanisms to obtain timely and consistency-preserving updates for dynamic data by* pull*ing data from the source at strategically chosen points in time. Motivated by the need for practical system design, but using formal analytical techniques, we offer a systematic approach based on control-theoretic principles. Our solution is also unique from a control-theoretic perspective due to the presence of inherent non-linear system components and the dependence between the sampled time and sampled value. A proportional controller with dynamically changing tuning criteria is used in this work as a means of deciding when to next refresh data from the source. Using real-world traces of real-time data we show the superior performance of our feedback-driven control-theoretic approach by comparing with (i) a previously proposed adaptive refresh technique and (ii) a new pattern matching technique.*

Keywords : Dynamic Data, Temporal Coherency, Data Dissemination, Push, Pull, Control-theory.

1 Introduction

An ever increasing fraction of the data delivered from today's data resources is time-varying (i.e., changes frequently). Examples of such data include sports information, news, financial information such as stock prices, and traffic data. An important issue in the dissemination of time varying data in such applications, is the maintenance of temporal coherency. The coherency requirements on a data item depends on the nature of the item and user tolerances. To illustrate, a user may be willing to receive sports and news information that may be out-of-sync by a few minutes with respect to the source, but may desire stronger coherency requirements on the status of patients' critical parameters.

In a typical data dissemination environment, maintaining consistency of the source and the user can be achieved in one of two ways: 1) In the `Source push` model, the source of the data *push*es data to the user (whenever data changes at the source). In this model, the source must maintain state information, in particular, it must keep track of user requirements and push the data to a user at the appropriate time. 2) In the `User Pull` model, the user *pulls* data from the source (whenever it suspects that data might have changed at the source). So, consistency of the data at the user depends on how often the user polls, i.e., pulls from the source. Too frequent polling may result in unnecessary overheads, while infrequent polling might mean stale data. Many real-world sources are designed to be polled e.g., simple sensors, web sources, sensor proxies[14]. Even if a source has push capability, pushing only when required leads to computational overheads and state-space overheads at the sources[3]. So, we look at the possible ways in which judicious pulling can be accomplished. We define *Time to Refresh (TTR)* as the time gap after which a user refreshes the data item from the source. For minimizing the incurred communication overheads, the value of TTR must be high. But a low TTR value may compromise temporal consistency. We must also dynamically update this TTR value using an algorithm that decides the value depending on the present and past rates of source changes, with the goal of keeping remote requests to a minimum while maintaining the needed temporal accuracy of the data.

1.1 Maintaining Temporal Coherency

We assume that a user specifies a temporal coherency requirement, c, for each cached item of interest. The value of c denotes the maximum permissible deviation of the value known to the user from the value at the source and thus constitutes the user-specified tolerance. Observe that c can be specified in units of *time* (e.g., the item should never be out-of-sync by more than 5 minutes) or *value* (e.g., the stock price should never be out-of-sync by more than a dollar).

In this paper, we only consider temporal coherency requirements specified in terms of the value of the data as maintaining temporal coherency specified in units of time is a simpler problem. Thus, we assume that a user specifies a temporal coherency requirement c, for each item of interest. The value of c specifies the maximum permissible deviation of the date item value seen by the user from the actual value at the source:

$$|data\ item\ value_{source} - data\ item\ value_{user}| \leq c \quad (1)$$

The degree to which users' coherency needs are met is measured in terms of the *fidelity* of the data seen by users. We define the fidelity F observed by a user to be the total length of time that the above inequality holds (normalized by the total length of the observations). In addition to specifying the coherency requirement, users can also specify their fidelity requirement F for each data item so that an algorithm that is capable of handling users' fidelity and temporal coherency requirements (c's) can adapt to users' needs. The problem is hence to devise an algorithm to generate the sequence of TTRs for which the temporal coherency requirement is satisfied at the lowest cost. Note that choosing large TTR values translates to low cost.

1.2 Periodic vs. Aperiodic Pull

The traditional approach is to use a constant refresh rate, i.e., periodic polls, in trying to meet the user specified coherency requirement. However this technique requires an *a priori* estimation of the data dynamics. Often a low conservative value is selected as the refresh rate in order to meet the tolerance requirement. This invariably leads to unnecessary client pulls resulting in an overhead cost of both communication and computational resources at the source.

An alternative to periodic polling is to adjust the refresh rate based on the observed dynamics of the data. This can be viewed as aperiodic polling. But, since dynamic data change independently and unpredictably, we cannot use a simple prediction algorithm for predicting the next TTR value to be assigned to the data object. In the adaptive method for assigning TTR values [7], the TTR value is varied based on the rate of change of the data item, given a user's coherency requirement: TTR decreases multiplicatively when a data item starts changing rapidly and increases additively when changes are infrequent. To achieve this objective, the *Adaptive TTR* approach incorporates (a) static bounds so that TTR values are not set too high or too low, and (b) accounts for the most rapid changes that have occurred so far as well as the most recent changes to the polled data. We describe the adaptive TTR algorithm in Section 2.

1.3 Contributions of the Paper

The Adaptive TTR algorithm assumes linear changes to estimate TTR. Our new Adaptive Pattern Matching TTR algorithm dispenses with this assumption and makes estimates by matching current change patterns with those observed in the past. This algorithm is described in Section 3. We also propose a feedback driven control theoretic approach to analyze the problem in a systematic way. A proportional controller, tuned with Ziegler-Nichol's method [11] is shown to work well for low and medium speed traces. A detuning method that helps maintain the control effort within permitted bounds is then proposed as a way to deal with fast changing traces. Because a control theoretic approach has the capability to tune many parameters and respond very fast, this is both systematic and practical. This algorithm is described in Section 4. We describe detailed performance evaluation, and discuss the results in Section 5.

2 A Simple Heuristic Algorithm

To achieve the coherency requirement of a data item, a user computes a *Time To Refresh (TTR)* for the data item. The TTR denotes the next time that the user should poll the source so as to refresh the data item if it has changed in the interim. The *Adaptive TTR Algorithm* [7] allows the user to adaptively vary the TTR value based on the rate of change of the data item. The TTR decreases multiplicatively when a data item starts changing rapidly and increases additively when changes are smaller and slower. To achieve this objective, the Adaptive TTR approach takes into account

- static bounds so that TTR values are not set too high or too low,
- the most rapid changes that have occurred so far and
- most recent changes to the polled data.

In this approach the most recent changes to the data are used to estimate TTR, assuming linear changes of data. Although the assumption of *linear changes* simplifies the heuristic algorithm, we need to go for an approach which is not dependent upon this assumption, for better estimation of TTR. In the next section we describe an Adaptive Pattern Matching TTR algorithm which is based on remembering previous patterns and does not assume linear changes of dynamic data.

3 An Adaptive Pattern Matching TTR (APMT) Algorithm

We see that different dynamic data fluctuate at different rates and that a particular data may have different dynamic behaviour with the passage of time. This suggests that the TTR associated with each data item is likely to change with time. We see that in the adaptive TTR algorithm described in Section 2, the estimation of $TTR_{estimate}$ assumes linear increase or decrease over TTR_{latest} based on coherency requirement (*c*) and change of dynamic data. Motivated by the need for an algorithm which can capture non-linear changes of dynamic data items, we propose in this section an Adaptive Pattern Matching TTR (AMPT) algorithm. The basic principle behind APMT algorithm is to predict TTR based on past known values of TTR and the direction of change to the data.

Initially we use frequent Polling of the source to get samples of TTRs. We represent the sequences of known TTR values as a time series y = $\{y_1, y_2, y_3, ...y_n\}$, where y_n represents the current value. The simple method to predict the next value, y_{n+1}, may be based on the value closest to y_n in the past data, say y_j, and predicting y_{n+1} on the basis of y_{j+1}. The definition of the current state may be extended to include more than one sample value, e.g., the current state of size three may be defined as $\{y_{n-2}, y_{n-1}, y_n\}$.

A segment in the series may be defined as a difference vector δ = $\{\delta_1, \delta_2, ...\delta_{n-1}\}$, where δ_i = $y_{i+1} - y_i$, $\forall$ $1 \leq i \leq n-1$. A pattern contains one or more segments and may be visualized as a string of segments S = $\{\delta_1, \delta_2, ...\delta_h\}$, for given value of *h*. In order to define patterns mathematically, we choose to encode the time series as a vector of changes in direction (*b*). For this purpose, a value y_i is encoded as 1 if $y_{i-1} > y_i$, as 2 if $y_{i-1} < y_i$ and as 0 if y_{i-1} = y_i. So, b = $\{b_1, b_2, ...b_{n-1}\}$, where b_i is either 0, 1 or 2. We define a pattern by both δ and *b* of size of *k* samples. If *k* is 3, we define a pattern by both δ_n = $\{\delta_{n-3}, \delta_{n-2}, \delta_{n-1}\}$ and b_n = $\{b_{n-3}, b_{n-2}, b_{n-1}\}$ for predicting y_{n+1}.

The matching algorithm tries to find a segment δ which is similar to *b* and has the smallest value for *m*, where *m* = $\sum_{i=1}^{k} w_i(\delta_{n-i} - \delta_{j-i})$, for all j having similar *b* values, w_is are the weights associated with $\{\delta_{n-i} - \delta_{j-i}\}$ and *k* is the sample size. We choose that j value which gives minimum m.

If b_{j+1} = 1, then y_{n+1} = $y_n + \beta * \delta_{j+1}$. If b_{j+1} = 2, then y_{n+1} = $y_n - \beta * \delta_{j+1}$ and if b_{j+1} = 0, then y_{n+1} = y_n, where β = $(1/k) * \sum_{i=1}^{k} \delta_{n-i}$ / δ_{j-i} and *k* is sample size.

So, the predicted TTR is y_{n+1}. Suppose user should be informed when change of dynamic data exceeds *c*. The mathematical form of this requirement is $|\Delta q| - c \leq 0$, where *c* is the coherency requirement and $|\Delta q|$ is the difference of previous value of dynamic data and recently polled dynamic data. We can have three possibilities, i.e., $|\Delta q|$ is less than *c*, equal to *c* or greater than *c*.

If $|\Delta q|$ is equal to *c*, it implies user polls the dynamic data correctly. If $|\Delta q|$ is less than *c*, we increase the previous TTR linearly by a tuning parameter *l* and use this corrected TTR and previous three TTRs for predicting the next TTR, if we choose segment of size 3. We use a window of samples to predict the next TTR in order to account recency. This window keeps moving as new TTRs are added.

If $|\Delta q|$ is greater than *c*, we decrease the previous TTR multiplicatively by a tuning parameter *r* (less than one) and use this corrected TTR and other previous three TTRs for predicting the next TTR, as we choose segment of size 3 for predicting the next TTR.

In this approach we see two important parameters: 1) window size (*n*) and 2) segment size (*p*). In order to have better performance of this approach we need to evaluate optimal window size and segment size across different types of the traces. We evaluate the sensitivity of these two parameters in section 5.3. Other tuning parameters are *r* and *l*, which we choose heuristically. In the next section, we propose a feed-back driven control theoretic approach to systematically address the problem described so far.

4 Control Theoretic Approach

4.1 Challenges in Providing a Control Theoretic Formulation

The control objective is to decide the time of pull so that a change of *c* is not missed. This immediately suggests that we use the time of pulling as the input variable and the value of the data item at that time as the output variable. Note that this input is quite different from what we see in the usual control problems where *time* is an independent variable.

This choice, however, will result in a non-stationary signal since the time of pull is a monotonically increasing function. The value of the data item could also increase or decrease endlessly during the observation period. The usual method of overcoming this difficulty is to take the first order difference between successive data values and use these values instead for the input and output variables. Although this differencing procedure is common in forecasting, it is unusual in control problems.

The process that we deal with is also unusual in that we are required to find the absolute difference between the data item values at two successive samples, say q_i and q_{i-1}. Let $\Delta q_i = q_i - q_{i-1}$ be the difference between the data item values at the two successive samples. In order to decide a strategy for providing a high fidelity together with a high average TTR, we are interested in knowing whether the absolute value of the difference Δq_i is equal to c. That is, we require that at all times,

$$|\Delta q_i| - c \leq 0 \qquad (2)$$

If we take $(|\Delta q_i| - c)/c$ as the output y_i of the system, then our fidelity requirement can be stated as demanding y_i to be equal to 0. This hence takes the form of a regulation problem. Note that the absolute function is a process model and is also unusual.

4.2 Problem Formulation

As discussed above, we define the input u_i to be,

$$u_i = t_i - t_{i-1} \qquad (3)$$

where t_i and t_{i-1} are respectively the times at which the i^{th} and $(i-1)^{th}$ pulls are effected. We define the corresponding output, y_i to be,

$$y_i = \frac{|q_i - q_{i-1}| - c}{c} \qquad (4)$$

where q_i and q_{i-1} are the data item values at t_i and t_{i-1} respectively. The independent variable i is the sample number. As is done in control problems, we define the input and output variables to be deviation variables from some reference values.

$$\Delta u_i = u_i - U_r \qquad (5)$$

and

$$\Delta y_i = y_i - Y_{ss} \qquad (6)$$

where U_r is some reference value of u (in traditional control applications, this is taken as the steady state value). U_r has been arbitrarily chosen as the mean of TTR_{min} and TTR_{max}. It has been found that U_r is not a sensitive parameter. Y_{ss} is the steady state value of the output. As the desired value of Y_{ss} is 0, we have,

$$\Delta y_i = y_i \qquad (7)$$

The open loop system depicting our process can be diagrammatically represented as in Figure 1(a).

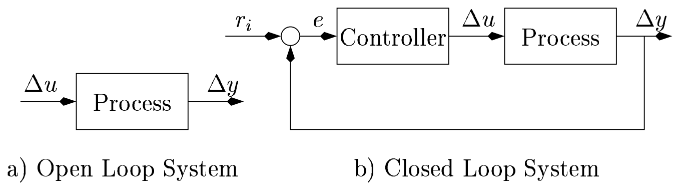


Figure 1. Open loop and Closed Loop Systems

4.3 Logic of Closed Loop Control

We next explore whether putting this process in a feedback loop with a controller will help achieve the required regulation: maintaining y_i at 0 for all i. Consider the behavior of the closed loop system in Figure 1(b). Here r_i denotes the setpoint for y_i. In our case, r_i is 0. In this figure, the output $\Delta y_i = y_i$ is compared with 0 and the difference, $e_i = -y_i$ is sent as input to the controller. One can easily observe that a negative error, $e_i < 0$, corresponds to $y_i > 0$ or equivalently to $|q_i - q_{i-1}| > c$. It follows that the user should have polled earlier. Hence, when $e_i < 0$, we want Δu_i to be negative so that u_i is decreased. Similarly, $e_i > 0$ implies that Δu_i should be increased. This indicates that a positive control action is required.

4.4 The Process Model

Let us suppose that the output Δy_i and input Δu_i of the process in Figure 1(a) be represented by an ARX[1] model [10] of the form,

$$\begin{aligned} &\Delta y_k + a_1 \Delta y_{k-1} + a_2 \Delta y_{k-2} + \cdots + a_n \Delta y_{k-n} \\ &= b_1 \Delta u_{k-1} + b_2 \Delta u_{k-2} + \cdots + b_n \Delta u_{k-n} + e_k \end{aligned} \qquad (8)$$

where e_k is an error term. The parameters n, a_1, a_2, ..., a_n and b_1, b_2, ..., b_n are determined using the observed input-output data as follows [12]:

We write equation (8) as,

$$e_k = \Delta y_k - \phi_k^T \theta \qquad (9)$$

where

$$\phi_k^T = \begin{bmatrix} -\Delta y_{k-1} & \cdots & -\Delta y_{k-n} & \Delta u_{k-1} & \cdots & \Delta u_{k-n} \end{bmatrix},$$

$$\theta = \begin{bmatrix} a_1 & \cdots & a_n & b_1 & \cdots & b_n \end{bmatrix}$$

We stack the values of the errors for the observed input and output sequences $\{\Delta u_1, \Delta u_2, ..., \Delta u_N\}$ and

[1]Auto-Regressive eXogenous

$\{\Delta y_1, \Delta y_2, ..., \Delta y_N\}$ respectively, thereby obtaining a tall system of the form,

$$\begin{bmatrix} e_{n+1} \\ e_{n+2} \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} \Delta y_{n+1} \\ \Delta y_{n+2} \\ \vdots \\ \Delta y_N \end{bmatrix} - \begin{bmatrix} \phi_{n+1}^T \\ \phi_{n+2}^T \\ \vdots \\ \phi_N^T \end{bmatrix} \theta, \qquad (12)$$

where N is a large number such that the matrix inverse exists in equation (13).

Although we would like the left hand side in equation (12) to be zero, we can only hope for a least squares solution as it is a system with more number of equations than unknowns. The least squares estimate using the non-recursive technique [12] is then given by,

$$\hat{\theta}_N = (\Phi_N^T \times \Phi_N)^{-1} \times \Phi_N^T \times Y_N \qquad (13)$$

where,

$$\Phi_N = \begin{bmatrix} \phi_{n+1} & \phi_{n+2} & \cdots & \phi_N \end{bmatrix}^T, \qquad (14)$$

$$Y_N = \begin{bmatrix} \Delta y_{n+1} & \Delta y_{n+2} & \cdots \Delta y_N \end{bmatrix}^T \qquad (15)$$

Initially, 1500 equally spaced samples are chosen. As the input u_i is the time difference $t_i - t_{1-1}$, u_i will be equal to the sampling interval, a constant value, for all i. Since the input is constant, the model cannot be identified as the persistent excitation condition [12] is violated. We have chosen the input vector to be repetitions of the vector (1,2,3,4,5) and the corresponding data values as the output. This, combined with the fact that 1500 samples are used to calculate four parameters of the model helps identify the model easily. Once the control procedure starts, the samples are at arbitrary time intervals and this strengthens the persistent excitation requirement. In the next section, we describe a model for the controller.

4.5 The Controller

Since proportional controllers are commonly used in systems where it is desirable to speed up the response, we would like to determine if they are sufficient to achieve our objective of making $y_i \rightarrow 0$. A proportional control is of the following form ,

$$\Delta u_{n+1} = \Delta u_n + K e_n \qquad (16)$$

where K is the proportional gain. A popular way to select the parameter K is the Ziegler-Nichol's method [11]. In this technique, the controller in Figure 1(b) is chosen as a proportional controller with a gain K. The value of K is gradually increased until the output, y_i starts oscillating. The value of K when this happens is taken as the critical gain, K_u. In order to determine the critical gain K_u, we take $\Delta \widehat{y_i}$ to be the output of the prediction model of Equation (8). If $\widehat{Y}(z)$ and $U(z)$ denote the z-transforms of $\Delta \widehat{y_i}$ and Δu_i respectively, then,

$$\begin{aligned} \left(1 + a_1 z^{-1} + ... + a_n z^{-n}\right) \widehat{Y}(z) = \\ \left(b_1 z^{-1} + ... + b_n z^{-n}\right) U(z) \end{aligned} \qquad (17)$$

We hence obtain the transfer function of the process as,

$$H = \frac{\widehat{Y}(z)}{U(z)} = \frac{b_1 z^{-1} + b_2 z^{-2} + ... + a_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + ... + b_n z^{-n}} \qquad (18)$$

Let the numerator and denominator polynomials of H be N and D respectively. The closed loop gain of the system using proportional controller with a gain K is then given as,

$$G_{proportional} = \frac{KH}{1+H} = \frac{KN}{D+KN} \qquad (19)$$

Simple algebraic manipulation shows that

$$D + KN = z^n + \Sigma_{i=1}^{n} (a_i + K b_i) z^{n-i} \qquad (20)$$

Since the stability of a system requires that all the poles of the z-transform of the transfer function lie within the unit circle, the problem of finding the critical gain reduces to that of finding the value of K for which there exists a root of the polynomial in equation (20) whose absolute value is equal to one. Call the corresponding gain as K_u. The Ziegler-Nichol's setting for proportional controller is $K = 0.5K_u$.

4.6 Controller Tuning

The controller designed so far does not take into account the constraints on the inputs, namely, $TTR_{max} \geq \Delta u \geq TTR_{min}$. Sometimes, the predicted TTR may lie outside this bound. This happens mainly in the fast moving traces. One way to handle this problem is to pose control design as a constrained optimization problem. As this is relatively a complicated problem, we have chosen an alternate approach, which involves detuning the Ziegler-Nichol's controller so that TTR will come within the required bounds. This detuning strategy is intuitive, simple and easy to implement. Moreover, computational overheads in implementing this strategy are minimal:

- We place bounds on Ziegler-Nichol's tuning parameter with upper bound as 0.5 and a pre-defined lower bound l_b.
- We set initial value of Ziegler-Nichol's parameter as 0.5 and effect the polls. Let TTR calculated by the controller be *actual TTR*. This may be out of bounds and hence may be unusable. It is then constrained to come within bounds. Let this TTR be called *TTR with bounds*.

- If the TTR exceeds TTR_{max} or goes below TTR_{min} more than a threshold t number of times, we decrease Ziegler-Nichol's tuning parameter by a factor λ.

In the next section we describe the results of experimental evaluation of this control-theoretic TTR adjustment and compare the overall performance with the Adaptive TTR and the APMT approaches.

5 Performance Evaluation

In this section we will discuss the simulation environment, metrics used for the experiment, and present the results.

5.1 Simulation environment

The algorithm was evaluated using a prototype source that replayed traces of dynamic data. The experiments assume that the network latency in polling and fetching dynamic data items from the source is fixed and is negligible.

The performance of the algorithm was evaluated using real-world traces. The presented results are based on stock price traces (i.e., history of stock prices) of a few companies obtained from *http://finance.yahoo.com*. The traces were collected at a rate of 2-3 stock traces per second. Since the rate of change of any stock quote is much greater than even one change per second, the traces can be considered to be real time traces. All the experiments were done on the local intranet. We have categorized the traces as fast, medium and slow based on a statistical measure *standard deviation* of the data item. We have taken 1000 traces, each of length 10,000, for categorization. The top one-third of the traces, which show rapid changes (i.e., *large standard deviation*) have been considered as fast changing trace, the middle one-third and the rest have been considered as medium and slow changing traces respectively. Table 1 shows examples of different types of traces and their statistical measure standard deviation.

In order to check whether our approach is applicable for other domains, such as the process industry, we also used a set of temperature data obtained every six seconds from a polymer reactor.

5.2 Metrics

The algorithm was evaluated using the following metrics:

1. Network Overhead in (%), which is the number of polls normalized by the length of the trace and multiplied by 100. So, 5% Network Overhead means five polls over a trace of length 100.
2. Loss of Fidelity, f_l, which can be measured based on the total time duration for which the client was oblivious to changes exceeding user specified *c* value.

$$f_l = \frac{Total\ out\ of\ sync\ time}{Total\ trace\ duration}$$

Total out of sync time is the time duration for which there were *false negatives*.

5.3 Effect of Parameters on the AMPT approach

Figure 2 shows the effect of variation of window size, a parameter in APMT approach, for fast, medium and slow changing traces. From Figure 2(a) we see the variation of network overheads (%) for different window sizess. For window sizes from 50 to 550, we see some variation of network overheads. For window size $>$ 550, however, the plots get flattened across all types of traces. Figure 2(b) shows variation of loss of fidelity for different window sizes. We see from the figure that for window size $>$ 500, we get almost the same fidelity across the categorized traces. So, we conclude that window size $>$ 550 can be selected as an optimal window size.

Figure 3 shows the effect of variation of segment size (*p*), another parameter in the APMT approach, for fast, medium and slow changing traces. From Figure 3(a) we see the variation of network overheads (%) for different segment sizes. We see from the figure that for segment size = 3, we get the least network overheads. From Figure 3(b) we see the variation of loss of fidelity for different segment sizes. If we consider both the Figures 3(a) and 3(b) to choose segment size, we can set segment size to be 3.

For segment size 3 we get better performance in terms of network overheads (%) and loss of fidelity in fast changing trace compared to other segment sizes. In medium and slow changing traces, we can also use segment size of 3 as the segment size.

5.4 Performance with Modified Ziegler-Nichol's Settings

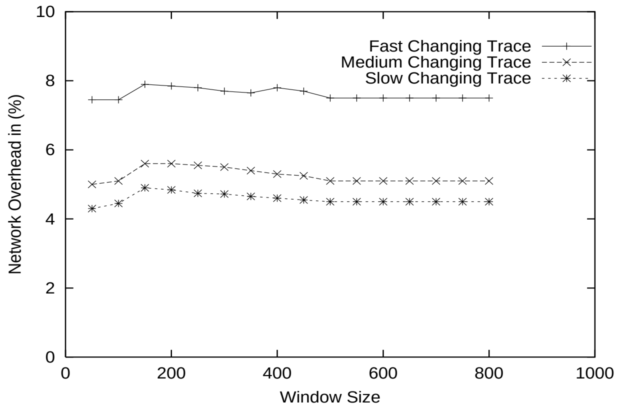
Table 2 shows the values of different parameters used in the experiment. We see in Figures 4(a), 4(b), and 4(c), the effect on TTR values of variation of Ziegler-Nichol's tuning parameter in fast, medium and slow changing traces. Figure 4(a) shows a plot of *actual TTR* (predicted TTR without considering TTR_{max} and TTR_{min}) and *TTR with bounds* with a standard Ziegler-Nichol's setting as 0.5 for fast changing traces. Figure 4(b) shows a plot for Ziegler-Nichol's setting as 0.1 and Figure 4(c) shows a plot with Ziegler-Nichol's setting modified using the technique discussed in Section 4.6. These three figures show the effect of variation of Ziegler-Nichol's setting and placing bounds

Type of trace	Company	Max value	Min Value	Standard Deviation
Fast Changing	Veritas	135.75	131.50	0.0100121
Medium Changing	INTC	134.5	132.5	.00587
Slow Changing	IBM	86.48	86.16	.000335

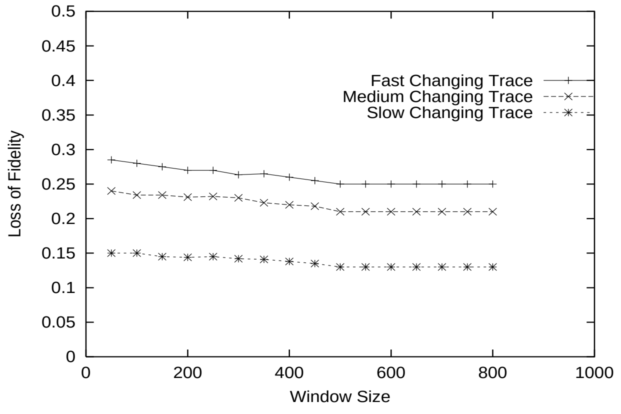
Table 1. Type of Traces used for the Experiment

parameter	Definition	Value
p	segment size in AMPT approach	3
n	window size in AMPT approach	600
l	a tuning parameter in AMPT approach	8-10
r	a tuning parameter in AMPT approach	0.4-0.6
l_b	lower bound of modified Ziegler's setting	0.1
λ	tuning parameter of modified Ziegler-Nichol's setting	0.1-0.2
t	threshould used in modified Ziegler-Nichol's setting	8-10

Table 2. Parameters used in Experiment

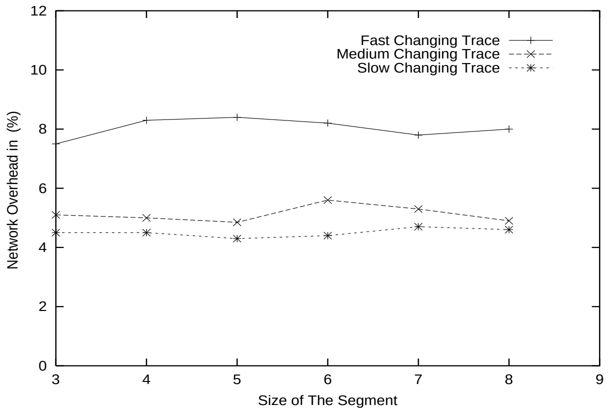


(a) Network Overhead in (%)

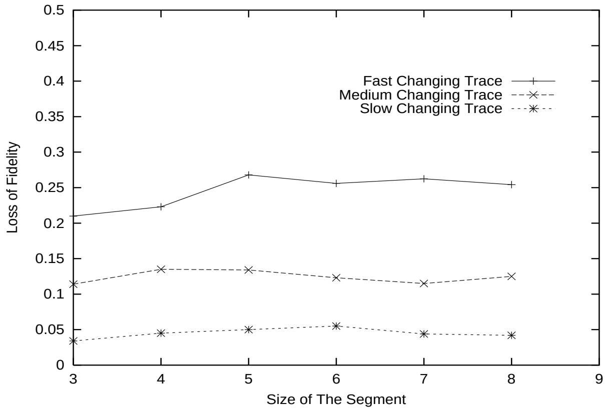


(b) Loss of Fidelity

Figure 2. Effect of Variation of Window (*n*) Size on Networks Overheads and Loss of Fidelity

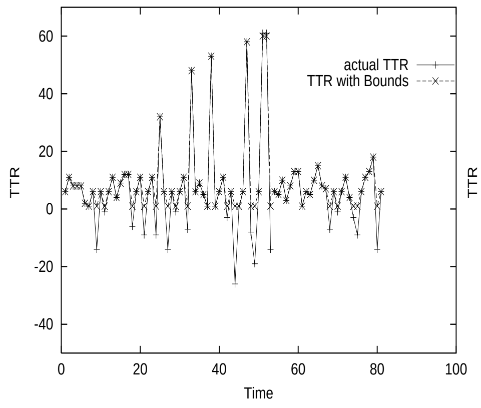


(a) Network Overhead in (%)

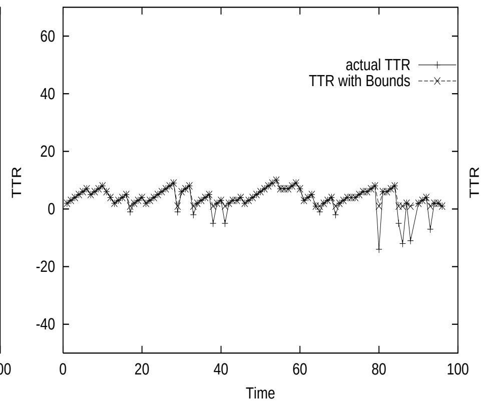


(b) Loss of Fidelity

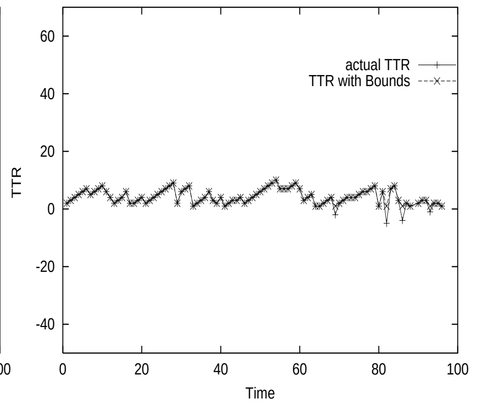
Figure 3. Effect of Variation of Segment Size (*p*) on Network Overheads and Loss of Fidelity



(a) Ziegler-Nichol's parameter = 0.5

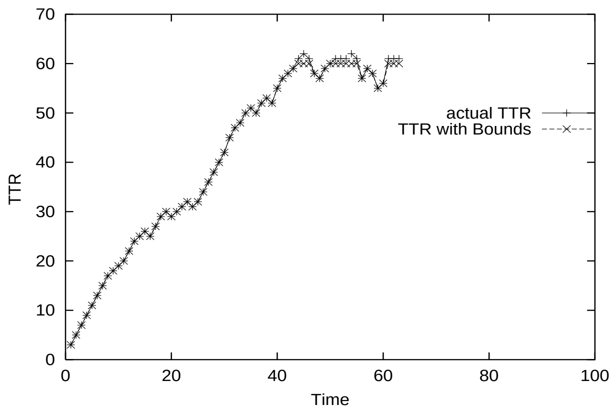


(b) Ziegler-Nichol's Parameter =0.1

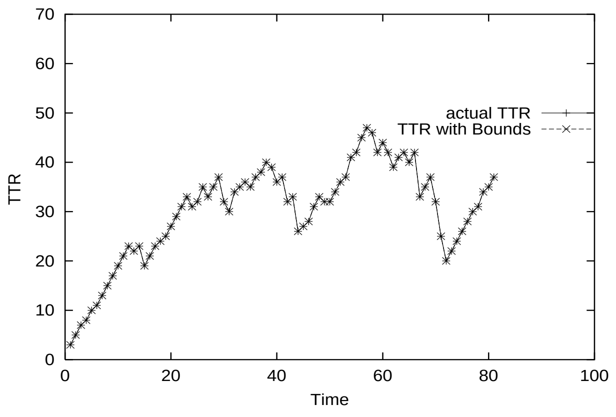


(c) Modifed Ziegler-Nichol's Settings

Figure 4. Effect of Variation of Ziegler-Nichol's setting in Fast Changing Trace

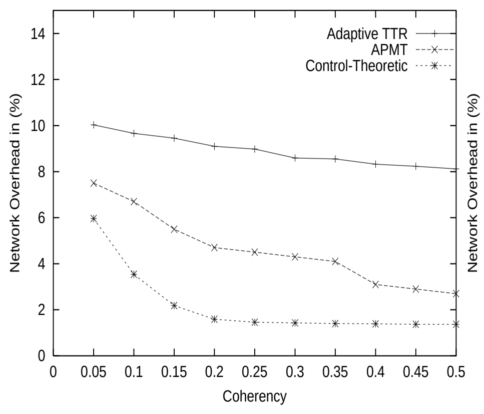


(a) Medium Changing Trace

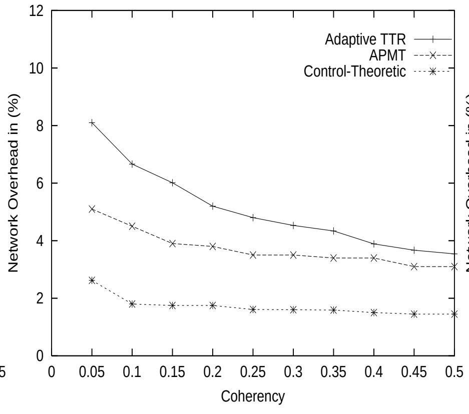


(b) Slow Changing Trace

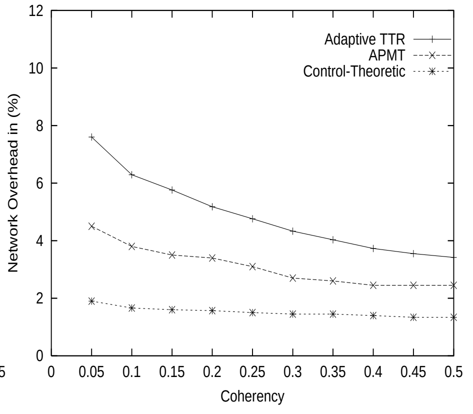
Figure 5. Standard Ziegler-Nichol's setting



(a) Fast Changing Trace



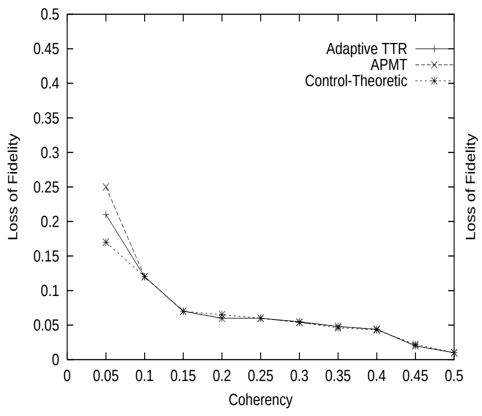
(b) Medium Changing Trace



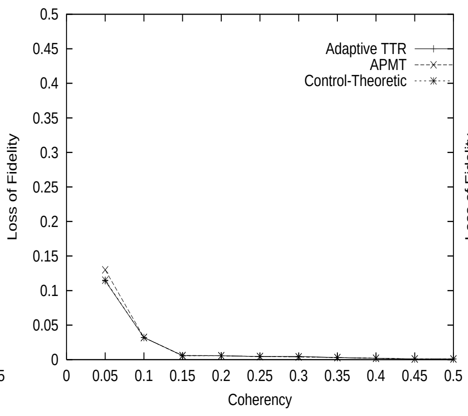
(c) Slow Changing Trace

Figure 6. Network Overheads (%) vs. Coherency

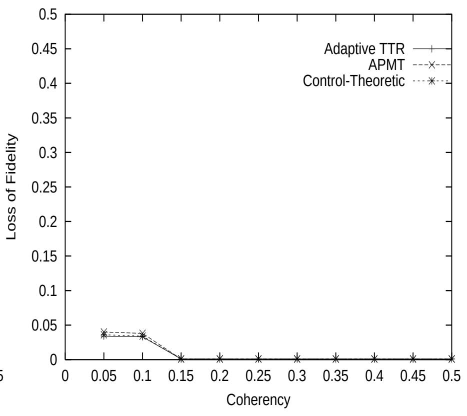
on controller's performance. We see from Figure 4(b) that we get less difference between *actual TTR* and *TTR with bounds* as compared to Figure 4(a). In other words, we can get better performance if controller responds less ag-



(a) Fast Changing Trace

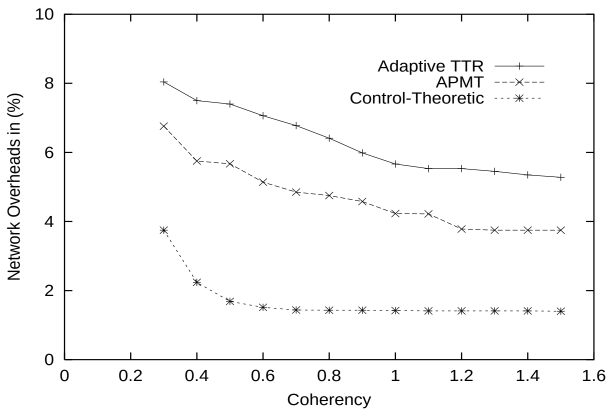


(b) Medium Changing Trace

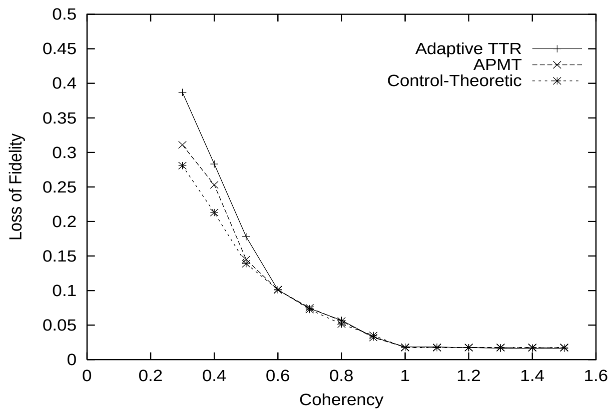


(c) Slow Changing Trace

Figure 7. Loss of Fidelity vs. Coherency



(a) Network Overheads vs. Coherency



(b) Loss of Fidelity vs. Coherency

Figure 8. Sensor Data: Air Temperature variation

gressively for rapid changes in dynamic data with a reduced Ziegler-Nichol's setting. We see from Figure 4(c) that we get less difference between *actual TTR* and *TTR with bounds* as compared to Figures 4(a) and 4(b), as in this case we use modified Ziegler-nichol's setting, which makes controller respond adaptively to dynamics of data. We see from figure 5(a) that we get very less difference between *actual TTR* and *TTR with bounds* in medium changing trace even if we use standard Ziegler-Nichol's setting. We also see from figure 5(b) that there is no difference between *actual TTR* and *TTR with bounds* for slow changing traces. So, we see that modified Ziegler-Nichol's setting makes the controller to respond adaptively and more accurately where needed - when data changes rapidly.

5.5 Comparison of Adaptive TTR, AMPT and Control Theoretic Approaches

We take 25 traces of each type for conducting experiments and the plots are based on average of result obtained from these traces. Figure 6 shows Network overheads of Adaptive TTR (Heuristic), APMT and Control-theoretic approaches. From Figure 6(a) we see that for fast changing traces we can reduce considerable network overheads, as much as 70% using control-theoretic approach. Similarly from Figure 6(b) and Figure 6(c) we see that even for medium and slow changing traces more than 60% and 50% improvement can be obtained.

Figures 7(a), 7(b) and 7(c) show comparison of loss of fidelity for fast, medium and slow changing traces respectively. We see from figure 7(a) that control-theoretic approach achieves less loss of fidelity as compared to adaptive TTR and AMPT approaches for lower coherency requirements. We also see from Figures 7(b) and 7(c) that loss of fidelity is almost the same for all three approaches for medium and slow changing traces. As mentioned earlier, we also conducted experiments on sensor data, specifically, air temperatures. Figures 8(a) and 8(b) show that Control-theoretic approach achieves less network overheads and high fidelity. In summary, from the experimental results we see that control-theoretic approach performs better than Adaptive TTR and APMT approaches in terms of achieving higher fidelity and lowering network traffic overheads.

6 Related Work

The design of coherency mechanisms for dynamic data has received significant attention recently. Proposed techniques include strong and weak consistency and the leases approach [5, 8]. Our contributions in this area lie in the definition of temporal coherency in combination with the fidelity requirements of users. *User polling*, is discussed in [4], where users periodically poll the server to check if the objects have been modified. In the Alex protocol, the user adopts an adaptive Time-To-Live (TTL) expiration time which is expressed as a percentage of the data's age.

Several research groups have designed adaptive techniques for web workloads [1, 2, 9, 6]. Whereas these efforts focus on reacting to network loads and/or failures as well dynamic routing of requests to nearby proxies, our effort focuses on adapting the dissemination protocol to changing system conditions.

A new approach, More-less principle, is proposed in [13] to derive deadlines and periods in update transactions in real-time databases. This approach provides better schedulability and reduces update transaction workload while guarantees data validity constraints, but it does not deal with unpredictable dynamics. We focus on adapting time-to-refresh for time-varing data, reducing network overheads.

7 Conclusions and Future Work

One of the attractive features of the novel approaches discussed in this paper is that these do not require sources to push changes, thus avoiding the computational overheads and state space overheads at the source. In this paper, we explored the possibility of using a proportional controller to address the problem of intelligent polling. We show superior performance of control theoretic approach over both the existing Adaptive TTR and the new Pattern Matching based approach in terms of communication overheads and fidelity given to the users. Our experimental results show that we can save substantial network overheads, as much as 70% in fast chaging traces, 63% in medium changing traces and 50% in slow chaging traces. We also showed that we can obtain better fidelity using control-theoretic approach compared to other two approaches.

Finally, the control theory based solutions have the potential to scale to multivariable systems, as these are backed by formal methods.

References

[1] FreeFlow Product Details, Akamai Inc., *http://www.akamai.com/service/freeflow.html, 1999.*

[2] P. Cao and S. Irani, Cost-Aware WWW Proxy Caching Algorithms., *Proceedings of the USENIX Symposium on Internet Technologies and Systems, December 1997, Anaheim, California.*

[3] M.Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, Adaptive Push-Pull: Disseminating Dynamic Web Data *IEEE Transactions on Computers special issue on Quality of Service*, June 2002, Vol 51, pp 652-668.

[4] A. Dingle and T. Partl. Web cache coherence. In *Proc Fifth International WWW Conference*, May 1996, Paris, France.

[5] V. Duvvuri, P. Shenoy and R. Tewari, *Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web. IEEE InfoCom March 2000, Tel-Aviv, Israel.*

[6] C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for Cached Approximate Values. In *Proceedings of the ACM SIGMOD Conference*, May 2001, Santa Barbara, USA.

[7] R. Srinivasan, C. Liang, and K. Ramamritham. Maintaining temporal coherency of virtual warehouses. In *Proceedings of of the 19th IEEE Real-Time Systems Symposium*, December 1998, Madrid, Spain.

[8] J. Yin, L. Alvisi, M. Dahlin and C. Lin, Hierarchical Cache consistency in a WAN., *Proceedings of the USENIX Symposium on Internet Technologies and Systems, October 1999, Colorado, USA.*

[9] A. Fox, Y. Chawate, S. D. Gribble and E. A. Brewer, Adapting to Network and Client Variations Using Active Proxies: Lessons and Perspectives., *IEEE Personal Communications, August 1998.*

[10] P. P. J. Van-den Bosch and A. C. Van-der Klauw. *Modeling, identification and simulation of dynamical systems*. CRC Press, Boca Raton, 1994.

[11] W. L. Luyben. *Process Modeling, Simulation and Control for Chemical Engineers*. McGraw-Hill International Editions.

[12] K. J. Astrom and B. Wittenmark. *Computer-Controlled Systems - Theory and Design*, chapter 13, pages 416–437.

[13] M. Xiong and K. Ramamritham, *Deriving Deadlines and Periods for Update Transactions in Real-Time Databases*, 20th IEEE Real-Time Systems Symphosium, 1999, Phoenix, USA

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, W. Hong, *TAG: A Tiny Aggregation Service for Ad-hoc Sensor Networks*, OSDI Conference, December 2002, Boston, MA.