Achieving Bounded Matching Delay and Maximized Throughput in Information Dissemination Management

Ming Chen, Xiaorui Wang, and Ben Taylor

***Abstract*—The demand for high performance information dissemination is increasing in many applications, such as e-commerce and security alerting systems. These applications usually require that the desired information be matched between numerous sources and sinks based on established subscriptions in a timely manner while a maximized system throughput be achieved to find more matched results. Existing work primarily focuses on only one of the two requirements, either timeliness or throughput. This can lead to an unnecessarily underutilized system or poor guarantees on matching delays. In this paper, we propose an integrated solution that controls both the matching delay and CPU utilization in information dissemination systems to achieve bounded matching delay for high-priority information and maximized system throughput in an example information dissemination system. In addition, we design an admission control scheme to meet the timeliness requirements for selected low-priority information. Our solution is based on optimal control theory for guaranteed control accuracy and system stability. Empirical results on a hardware testbed demonstrate that our controllers can meet the timeliness requirements while achieving maximized system throughput.**

***Index Terms*—real-time and embedded systems, Feedback control real-time scheduling, distributed systems, end-to-end task, quality of service, distributed model predictive control.**

I. Introduction

DURING the last decade, information dissemination has started to play a critical role in the design and development of a large class of applications. For example, buyers and sellers need to be matched based on their interests in e-commerce systems, and notified immediately when new business opportunities are identified. Likewise, in security alerting systems, threats detected by various sensors must be reported to the appropriate authorities within certain time constraint. In these applications, matches between numerous (*e.g.*, thousands of) sources and numerous sinks should be found accurately, efficiently, and more importantly, in a timely manner. These requirements have been generally described as *Valuable Information at the Right Time (VIRT)* [2]. This emphasizes that consumers of information should receive the accurate information that is of interest to them as soon as it is available or whenever it is requested. In the meantime, a maximal possible system throughput should be achieved to find more matched results between data sources and sinks.

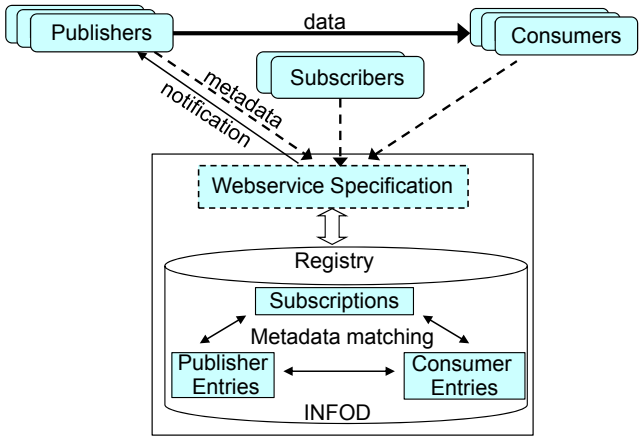


Fig. 1. INFOD: an example information dissemination system.

INFOD (INFOrmation Dissemination) [3] is an example information dissemination system that aims to support timely delivery of valuable information for a wide range of applications. As shown in Fig. 1, information sources and sinks are defined as *publishers* and *consumers*, respectively. *Subscriptions* are prescribed requests of information from publishers to subscribers and submitted by *subscribers* on behalf of consumers. Unlike other information dissemination systems that hold data itself, INFOD serves as an information broker and only allows publishers and consumers to advertise their attributes and constraints, which are generally referred to as *metadata*, in a database called *registry*. For example, a traffic sensor (publisher) may have its location and the abstract description of the traffic it monitors as metadata, while a driver (consumer) may have his location and name as metadata. An example subscription can be: all sensors (publishers) send their traffic information to all drivers (consumers) within 10 miles no later than 10 seconds after a jam occurs. Metadata can be updated periodically and aperiodically. When a metadata update arrives, INFOD needs to find new matched results between the metadata of publishers and consumers based on the subscriptions, which we refer to as *subscription reevaluation* or *metadata matching*. Based on the matched results, publishers are informed where to send filtered information. The information (data) is then sent to the matched consumers

Manuscript received December 16, 2009; revised June 3, 2010. The associate editor coordinating the review of this paper and approving it for publication was R. Stadler.

The authors are with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996 (e-mail: {mchen11, xwang, btaylo22}@utk.edu).

This paper is a significantly extended version of a conference paper [1].
Digital Object Identifier 10.1109/TNSM.2011.012111.00004

without passing through INFOD. The main function of the INFOD system is to find new matched results within a certain time constraint. Meanwhile, more matched results are desired by reevaluating the maximized number of subscriptions, which means that the registry server should be efficiently utilized.

To guarantee both the bounded matching delay and efficient system utilization in information dissemination systems faces several major challenges. *First*, when an update arrives, the system may need to reevaluate all subscriptions to find new matched results between the publishers and consumers. For example, a driver may constantly update his location attribute. As a result, all subscriptions in the registry need to be continuously reevaluated by rerunning metadata matching to ensure that the driver receives information from the right locations. However, given the large number of publishers and consumers, reevaluating all subscriptions may cause severe system overload and unacceptably long delays. Therefore, to guarantee bounded matching delay and avoid system overload, only a few of the subscriptions can be selected for reevaluation. *Second*, metadata updates may arrive at unpredictable intervals. Given a constant number of subscription reevaluations, the registry may suffer either over- or under-utilization, depending on whether the interval is short or long. Over-utilization may lead to unbounded matching delay, while under-utilization may lead to unnecessarily low system throughput, resulting in incomplete matched results. Both unbounded matching delays and unnecessarily under-utilized systems are undesirable. For example, a driver may fail in finding the fastest route, either due to a late notification of traffic information, or because the registry fails to find the existing matched traffic information. *Third*, subscriptions may have different priorities. For example, subscriptions of traffic information for police should have a higher priority than those for ordinary drivers. Therefore, reevaluation preference should be given to high-priority subscriptions.

To address all the challenges, we propose a mechanism by applying a batching window to group those updates at small interarrival intervals and release them all together. We differentiate subscriptions by two priorities: high and low. After each release, all high-priority subscriptions are reevaluated first, and then a certain number of low-priority subscriptions are selected for reevaluation. The low-priority subscriptions are reevaluated in a round-robin way. The number of subscriptions that are reevaluated after each batching window is defined as ***job budget***. Clearly, both the batching window size and job budget affect the matching delay and utilization of the registry server. Therefore, it is important to determine the batching window size and job budget in an integrated way such that the average matching delay of all high-priority subscriptions are guaranteed to meet the specified constraint, and the registry server can be efficiently utilized to achieve maximized system throughput.

In this paper, we present a novel solution to control both the matching delay of the high-priority subscriptions and CPU utilization of the registry server in an integrated manner. Under our control solution, the average matching delay of all the high-priority subscriptions can converge to the specified constraint, while the CPU utilization is controlled to a desired set point so that the system can reevaluate the maximized number of subscriptions. In addition, we present an admission controller to guarantee that all the low-priority subscriptions can be reevaluated at least once within a certain time constraint. Specifically, the contributions of our work are five-fold:

- We propose a novel batching mechanism to address the challenges of metadata matching in information dissemination systems;
- We model the average matching delay and CPU utilization of the registry server and validate the model with data measured on a hardware testbed.
- We design a Multiple-Input-Multiple-Output (MIMO) control solution based on the system model to guarantee the timeliness of all the high-priority subscriptions and system throughputs simultaneously and present a detailed analysis of system stability based on optimal control theory.
- We design an admission controller to guarantee the timeliness of all the low-priority subscriptions.
- We implement our control solution based on a real information dissemination system and present empirical results on a hardware testbed to demonstrate that our solution can guarantee the timeliness for all the subscriptions and achieve maximized system throughputs.

The rest of the paper is organized as follows. Section II discusses the related work. Section III introduces the overall architecture of the system. Section IV and Section V present the system modeling, controller design and analysis of the two controllers. Section VI describes the implementation details. Section VII presents the results of our experiments and Section VIII concludes the paper.

II. Related Work

Some related work has been done to guarantee the average response time and CPU utilization in computing systems. For example, Horvath et al. address End-to-End response time control in multi-tier web servers by using dynamic voltage scaling [4]. Wang et al. present a load balancing controller to control the relative response time among virtualized servers [5]. However, those solutions aim to control the average response time of web applications by conducting dynamic frequency and voltage scaling (DVFS). The key difference between their projects and our work is that they try to minimize system resource usage by running the CPU at the lowest possible DVFS level to achieve power savings, while our work tries to utilize the system resource to the maximum degree by evaluating as many low-priority subscriptions as possible. Diao et al. develop MIMO control algorithms to control the processor and memory utilization for Apache web servers [6]. Chen et al. also propose a hierarchical control architecture to guarantee the deadline of power grid computing tasks by controlling CPU utilization [7]. This paper is different because we control not only CPU utilization but also the average delay of metadata matching in information dissemination systems. Our experimental results presented in Section VII-C demonstrate that the proposed integrated control solution can achieve both bounded delay and maximized system throughput, while controlling CPU utilization or average

matching delay separately leads to either delay requirement violation or under-utilized systems.

Our previous work [8] presents an initial solution to control only the processing delay of metadata matching. This paper is significantly different because: 1) we design a batching mechanism that can handle unpredictable update interarrival intervals; 2) we simultaneously control matching delay (including waiting time and processing time) and CPU utilization, based on the optimal control theory, for better system throughputs; 3) we design an admission controller to guarantee the matching interval of all low-priority subscriptions to meet the specified constraint. We use our previous work as a baseline in our evaluation.

Information dissemination has recently received a lot of attention. For example, Bullet, a multi-receiver data dissemination mesh that aims for the high-bandwidth data dissemination for large-scale distributed systems, has been presented by Kostić et al. [9]. Voulgaris et al. propose a self-organizing content-based publisher/subscriber system for dynamic large-scale collaborative networks [10]. Iamnitchi et al. develop an interest-aware information dissemination system for small-world communities [11]. However, most related work primarily attempts to improve the efficiency and accuracy of information dissemination from the viewpoints of matching algorithm, system architecture, and security concern. In a complimentary way, this paper studies information dissemination systems from the view of the system level by focusing on a different, but equally important, problem, *i.e.*, guaranteeing bounded delay while achieving maximized system throughput.

Batch and priority-based algorithms have been applied in a number of other computing systems, *e.g.*, scheduling for clusters [12][13], memory controllers [14], and network switching [15][16]. However, all these algorithms are designed for specific scenarios with different goals, such as performance/engergy optimization, and decreased packet loss rate. To our best knowledge, those algorithms cannot be applied in information dissemination systems like INFOD due to different scenarios and goals. Mitzenmacher et al. utilize information freshness in servers to do load balancing [17]. In our paper, we consider information fresh as long as the average matching delay of the related subscription evaluations is shorter than the desired bound.

Some other previous work has been done on quality of service management in databases. For example, some on-demand updating algorithms have been developed to improve CPU utilization by skipping unnecessary updates [18][19][20][21]. However, those studies are based on the assumption that workload is either periodic sensor updates or aperiodic user transactions, and then adapt the number of updates for desired CPU utilization. All these methods cannot be directly applied to information dissemination systems.

Control theory has been applied in a number of other computing systems, *e.g.*, data services [22], power management [23][24], and Internet servers [25][26][27]. A survey of feedback performance control in various computing systems is presented in [28]. Most of these solutions focus on different problems, such as allocating computing resources to meet with application level performance goals. In this work, we apply control-based methodology to guarantee the timeliness of subscription reevaluation and system throughputs in information dissemination systems.

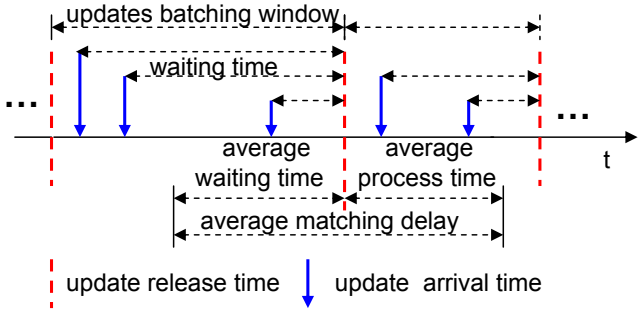


Fig. 2. Illustration of batching window.

III. System Overview

In this section, we first introduce the batching mechanism used in our solution, and then give a high-level description of the architecture that features two control loops. Finally we discuss the coordination between the two control loops.

A. Batching Mechanism

As shown in Fig. 2, we employ a *batching window* to accumulate all the updates that come within the window and release them in a batch. At the end of each batching window, defined as the *releasing time*, all the high-priority subscriptions are chosen for reevaluation first, and then some of the low-priority subscriptions are reevaluated. All the low-priority subscriptions are picked up in a round-robin way. Specifically, we refer to the length of time between two adjacent releasing times as the *batching window size*. We define the difference between the arrival time and the releasing time of an updates as the *waiting time* of that update. The difference between the releasing time and the time when a subscription is completed is defined as the *processing time* of that subscription. The *average matching delay* of all the high-priority subscriptions after a releasing time is calculated as the sum of the average waiting time of all the batched updates in a batching window and the following average processing time of all the reevaluated high-priority subscriptions after the batched updates are released.

In this paper, the number of all reevaluated subscriptions each second is defined as the ***overall throughput*** of the system. After each release, the job budget (*i.e.*, the number of all subscriptions to be reevaluated) should be maximized so that more low-priority subscriptions can be reevaluated after each batching window. The number of reevaluated low-priority subscriptions each second is defined as the ***throughput of low-priority subscriptions***. Hereinafter, we refer to both the overall throughput and the throughput of low-priority subscriptions as ***system throughputs***. In addition, the average time that all the low-priority subscriptions take to be reevaluated at least once is defined as the ***average matching interval***.

B. Control Architecture

Our control architecture consists of two control loops: the integrated control loop and the admission control loop. We now introduce them respectively.

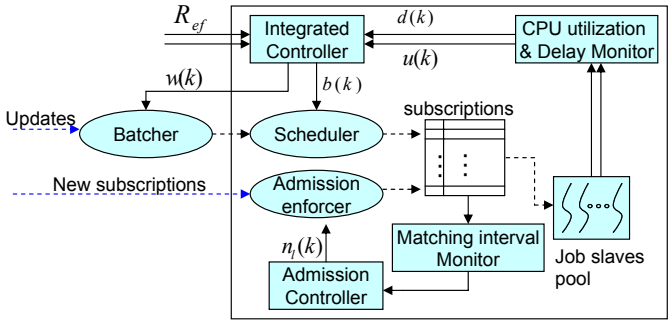


Fig. 3. Integrated control architecture.

As shown in Fig. 3, the key components in the integrated control loop include the integrated controller, delay and CPU utilization monitor, updates batcher, and scheduler. The control loop is invoked periodically at the end of every control period as follows: 1) The monitor measures the average matching delay and CPU utilization in the last control period, and sends the values to the controller. The average matching delay and CPU utilization are the *controlled variables* in the control loop; 2) The controller calculates the appropriate job budget and batching window size for the next control period based on the differences between the set points and measured controlled variables. The job budget and batching window size are the *manipulated variables* in the control loop; 3) Based on the calculated batching window size, all the updates that arrive within the window are released in a batch. At the releasing time, the batcher calculates the average waiting time of all updates in the window and notifies the scheduler in the database; 4) After receiving the notification from the batcher, the scheduler, based on the calculated job budget, schedules all the high-priority subscriptions and the desired number of low-priority subscriptions to be reevaluated in the system.

The key components in the admission control loop include the admission controller, the matching interval monitor, and the admission enforcer. The control loop is also invoked periodically. The *controlled variable* is the average matching interval and the *manipulated variable* is the number of low-priority subscriptions that the system is able to admit. The controller calculates the number of low-priority subscriptions that the system is able to admit for the next control period. Based on the calculated number of low-priority subscriptions that the system is able to admit, the admission enforcer admits all the new high-priority subscriptions and then, applies some possible admission control schemes for the arriving low-priority subscriptions. After a subscription is admitted, it is inserted into the subscription table from which the scheduler chooses subscriptions to be reevaluated.

C. Coordination of Control Loops

Clearly, without effective coordination, the two control loops (*i.e.*, the integrated control loop and admission control loop) may conflict with each other so that the system stability may be impaired. The job budget (*i.e.*, the number of all the subscriptions to be reevaluated) manipulated by the integrated control loop will have a direct impact on the average matching interval of all the low-priority subscriptions. The number of low-priority subscriptions admitted in the system may influence the average matching delay of all the high-priority subscriptions and CPU utilization of the registry server as well. In this subsection, we discuss the coordination of the two control loops.

In information dissemination systems, when new metadata updates continuously arrive, high-priority subscriptions are usually required to be reevaluated more frequently than low-priority subscriptions so that high-priority information can be disseminated in a more timely manner. Therefore, the desired average matching interval of all the low-priority subscriptions is usually greater than the desired average matching delay of all the high-priority subscriptions. As a result, the admission control loop is configured with a control period that is multiple of the control period of the integrated control loop. Without loss of generality, the admission control loop can be designed based on a system model in which the integrated control loop is working in its steady state. The impact of the steady state variation of the integrated control loop on the admission control loop can be modeled as a model variation. A larger control period of the admission control loop, compared with that of the integrated control loop, implies a smaller model variation but a slower reaction speed. In Section V-C, we have proven that the closed-loop system of the admission control loop is stable only if the variation is within a certain range. On the other side, the admission controller changes the number of the low-priority subscriptions from which the scheduler may choose subscriptions for reevaluation. If the total number of subscriptions admitted to the system is extremely small so that it is smaller than the job budget, the average matching delay and CPU utilization may be smaller than their set points. Therefore, the admission control loop does not affect the integrated control loop provided that the number of subscriptions admitted to the system is greater than the job budget.

Since the core of each control loop is the controller, we introduce the design and analysis of the two controllers in the next two sections, respectively. The implementation details of other components are given in Section VI.

IV. Integrated Controller

In this section, we present the problem formulation, modeling, design, and analysis of the integrated controller.

A. Problem Formulation

We first introduce the following notation.

- $\mathbf{R_{ef}}$: The reference vector, $\mathbf{R_{ef}} = \begin{bmatrix} R_d & R_u \end{bmatrix}^T$, where R_d and R_u are the reference values of the average matching delay and CPU utilization, respectively.
- T: The control period of the integrated controller.
- $w(k)$, $p(k)$, and $d(k)$: The average waiting time of all the batched update, the average processing time of the high-priority subscriptions, and the average matching delay in the k^{th} control period, respectively. Specifically, $d(k) = p(k) + w(k)$.
- $u(k)$, $s(k)$, and $b(k)$: The average CPU utilization, batching window size, and job budget, respectively, in the k^{th} control period.

- $\overline{d}$, $\overline{u}$, $\overline{s}$ and $\overline{b}$: The operating points of $d(k)$, $u(k)$, $s(k)$, and $b(k)$.

The goal of the integrated control is to simultaneously control both the average matching delay of the high-priority subscriptions and CPU utilization to their respective set points, R_d and R_u. The purpose of controlling the average matching delay is to guarantee that all the matched results of high-priority subscriptions can be disseminated within a time constraint after a metadata update arrives. The purpose of controlling the CPU utilization is to achieve maximized system throughputs (*i.e.*, the overall throughput and throughput for low-priority subscriptions) so that more subscriptions can be reevaluated to find more matched results between data sources and sinks. The selection of the set point of the CPU utilization is a compromise between the control accuracy of the average matching delay and system throughputs. The higher the CPU utilization, the larger the system throughputs, but the more difficult it is to control the average matching delay accurately, which may result in undesired large oscillations.

In this paper, the integrated controller is formulated as an optimal control problem to find the optimal batching window size and maximize the job budget (*i.e.*, the number of subscriptions to be reevaluated) while controlling matching delay and CPU utilization to their set points.

B. System Modeling

In order to have an effective controller design, it is important to model the dynamics of the controlled system, namely the relationship between the controlled variables (*i.e.*, $d(k)$ and $u(k)$) and the manipulated variables (*i.e.*, $b(k)$ and $s(k)$). However, a well-established physical equation is usually unavailable for computing systems. Instead of trying to build a physical equation between the manipulated variables and controlled variables, we infer their relationship by collecting data in experiments and establish a statistical model based on the measured data. Therefore, we apply a standard approach called system identification [29] to this problem.

We use the following difference equation to model the controlled system [30]:

$$\mathbf{y(k)} = \sum_{i=1}^{n_a} \mathbf{A_i y(k-i)} + \sum_{i=1}^{n_b} \mathbf{B_i v(k-i)}, \tag{1}$$

where $\mathbf{y(k)} = \begin{bmatrix} d(k)-\overline{d} & u(k)-\overline{u} \end{bmatrix}^T$ and $\mathbf{v(k)} = \begin{bmatrix} b(k)-\overline{b} & s(k)-\overline{s} \end{bmatrix}^{\mathbf{T}}$, the input and output vector, respectively. n_a and n_b are the orders of the control outputs and control inputs. $\mathbf{A_i}$ and $\mathbf{B_i}$ are the model parameters whose values need to be determined by system identification.

To have an accurate model, we need to identify the operating points of the controlled system [31]. The operating points of the outputs (*i.e.*, $\overline{d}$ and $\overline{u}$) are typically chosen to lie close to the reference values (*i.e.*, R_d and R_u). Meanwhile, the operating points of the inputs (*i.e.*, $\overline{b}$ and $\overline{s}$) are identified by preliminary experiments such that the operating points of the outputs can be reached.

For system identification, we need to first determine the right orders for the system, *i.e.*, n_a and n_b, in the difference equation (1). The order values are normally a compromise between model simplicity and model accuracy. To stimulate the system by using pseudo-random digital white noise is a standard way used in system identification [30]. In this paper, we stimulate the system by mapping a sequence of white noise to the system inputs (*i.e.*, $b(k)$ and $s(k)$) around their operating points and then measure the control outputs (*i.e.*, $d(k)$ and $u(k)$) in each control period. Our experiments are conducted on the testbed introduced in Section VI. Based on the collected data, we use the *Least Squares Method (LSM)* to iteratively estimate the values of parameters $\mathbf{A_i}$ and $\mathbf{B_i}$. In this paper, we choose $n_a = 1$ and $n_b = 1$ for a trade-off between model complexity and model accuracy.

We then generate another sequence of white noise to validate the results of system identification. Figure 4 shows the measured outputs from the open-loop system and the predicted outputs from the model. The errors measured in *Root Mean Squared Error (RMSE)* are sufficiently small (*i.e.*, 0.113 and 0.031 for the average matching delay and the CPU utilization, respectively), which measures the root of the average square of the difference between the measured and the predicted outputs. The errors have also been calculated as R^2, which measures how well the system outputs can be predicted by the model [30]. The accuracy measured in R^2 is around 73% and 80% for the average matching delay and the CPU utilization, respectively. We can see that the predicted outputs of the selected model are sufficiently close to the actual system outputs. Therefore, the resultant system model from our system identification is:

$$\mathbf{y(k)} = \mathbf{A_1 y(k-1)} + \mathbf{B_1 v(k-1)}, \tag{2}$$

where A_1 and B_1 are 2×2 constant matrices whose values are determined by system identification.

C. Controller Design

We apply the *Linear Quadratic Regulator (LQR)* control theory [32] to design the controller based on the system model (2). LQR is an optimal control technique that can deal with coupled MIMO control problems and has small computational overhead at runtime. To design the controller, we first convert our system model to a state space model. The state variables are defined as follows:

$$\mathbf{x(k)} = \begin{bmatrix} \mathbf{e}(k) \\ \mathbf{e_I}(k) \end{bmatrix}, \tag{3}$$

where $\mathbf{e}(k) = \mathbf{R_{ef}} - \begin{bmatrix} d(k) & u(k) \end{bmatrix}^T$ and $\mathbf{e_I}(k) = \mathbf{e_I}(k-1) + \mathbf{e}(k)$ are the control error vector and the accumulated control error vector, respectively. After some transformation, our final state space model of the controlled system is:

$$\begin{bmatrix} \mathbf{e}(k+1) \\ \mathbf{e_I}(k+1) \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{e}(k) \\ \mathbf{e_I}(k) \end{bmatrix} + \mathbf{B v}(k) + \begin{bmatrix} \mathbf{I} - \mathbf{A_1} \\ \mathbf{0} \end{bmatrix} \mathbf{r}, \tag{4}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_1} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -\mathbf{B_1} \\ \mathbf{0} \end{bmatrix},$$
$$\mathbf{r} = \mathbf{R_{ef}} - \begin{bmatrix} \overline{d} & \overline{u} \end{bmatrix}^T.$$

The detailed derivation can be found in [31] and is omitted here. Based on the state space model (4), we then design the LQR controller by choosing gains to minimize the following quadratic cost function:

$$\mathbf{J} = \sum_{k=1}^{\infty} \begin{bmatrix} \mathbf{e(k)^T} & \mathbf{e_I(k)^T} \end{bmatrix} \mathbf{Q} \begin{bmatrix} \mathbf{e}(k) \\ \mathbf{e_I}(k) \end{bmatrix} + \sum_{k=1}^{\infty} \mathbf{v^T(k) R v(k)}. \tag{5}$$

The first item in (5) represents the control errors and accumulated control errors. By minimizing the first item, the closed-loop system can converge to the desired set points. The second item in (5) represents the control efforts. Minimizing the second item ensures that the controller will minimize the changes in the control inputs, *i.e.*, the changes of the job budget and batching window size. $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices that determine the trade-off between the control errors and control efforts. A general rule to determine the values of $\mathbf{Q}$ and $\mathbf{R}$ is that a larger $\mathbf{Q}$ leads to faster response to workload variations, while a larger $\mathbf{R}$ makes the system less sensitive to system noise. In our paper, the control accuracy of the average matching delay $d(k)$ is more important than the CPU utilization $u(k)$. Therefore, we give a higher weight to $d(k)$ than that to $u(k)$ in $\mathbf{Q}$. The LQR controller is designed by using the Matlab command *dlqry* to solve the optimization problem (5). The controller gain matrix $\mathbf{K}$ is as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K_P} & \mathbf{K_I} \end{bmatrix}, \tag{6}$$

where $\mathbf{K_P}$, $\mathbf{K_I}$ are constant controller parameters for the error vector $\mathbf{e}(k)$ and the accumulating error vector $\mathbf{e_I}(k)$, respectively, so that the cost function (5) is minimized. Consequently, the control inputs in the k^{th} control period, *i.e.*, the job budget and the batching window size, are computed as:

$$\begin{bmatrix} b(k) - \overline{b} \\ s(k) - \overline{s} \end{bmatrix} = -\mathbf{K} \begin{bmatrix} \mathbf{e}(k) \\ \mathbf{e_I}(k) \end{bmatrix} \tag{7}$$

D. Control Analysis for Model Variations

A fundamental benefit of the control-theoretic approach is that it gives us theoretical confidence in system stability, even when the controller is used under different working conditions. In this subsection, we analyze the system stability when the integrated controller is applied to a system whose model is different from the nominal model described by (2). One of the major model variations is due to the varying arrival time of updates within the batching window. In this subsection, we give the stability analysis for varying arrival time of updates.

In system identification, we assume that all the updates arrive at the middle of a batching window and the average waiting time of updates is equal to half of the batching window size, *i.e.*, $w(k) = 0.5s(k)$. However, in real systems, updates may arrive at any time within a batching window and the average waiting time of updates is unknown a priori. To verify that the closed-loop system is stable when the average waiting time is any proportion of the batching window size, we model the variation of arrival time as $w(k) = g \cdot s(k)$, where $0 \leq g \leq 1$. We define g as the *waiting-time factor*. Without loss of generality, we outline the general steps to analyze the stability as follows.

1) We model the variation of the average arrival time as g in the matrix B_1 in the system model (2) as follows:

$$\mathbf{B_1'} = \begin{bmatrix} b_{11} & b_{12} - 0.5 + g \\ b_{21} & b_{22} \end{bmatrix}, \tag{8}$$

where $\mathbf{B_1'}$ is the varied matrix of $\mathbf{B_1}$ at runtime. b_{12} models the relationship between the batching window size and matching delay;

2) Derive the closed-loop system model by substituting the derived control inputs $\mathbf{v}(k)$ into the system model (2) by replacing $\mathbf{B_1}$ with $\mathbf{B_1'}$. The closed-loop system model is in the following form:

$$\begin{bmatrix} \mathbf{e}(k) \\ \mathbf{e_I}(k) \end{bmatrix} = (\mathbf{A} - \mathbf{B'K}) \begin{bmatrix} \mathbf{e}(k-1) \\ \mathbf{e_I}(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{I - A_1} \\ \mathbf{0} \end{bmatrix} \mathbf{r}, \tag{9}$$

where $\mathbf{B'} = \begin{bmatrix} -\mathbf{B_1'} & \mathbf{0} \end{bmatrix}$, and $\mathbf{r} = \mathbf{R_{ef}} - \begin{bmatrix} \overline{d} & \overline{u} \end{bmatrix}^T$.

3) Derive the stability condition of the closed-loop system described by (9). According to control theory, the closed-loop system is stable if all eigenvalues of the matrix $(\mathbf{A} - \mathbf{B'K})$ are located inside the unit circle.

Following the steps above, we have proven that the closed-loop system is stable when g varies from 0 to 1 at runtime, which means that the system can be guaranteed to be stable no matter when the updates arrive.

V. Admission Controller

In this section, we present the problem formulation, modeling, and design of the admission controller.

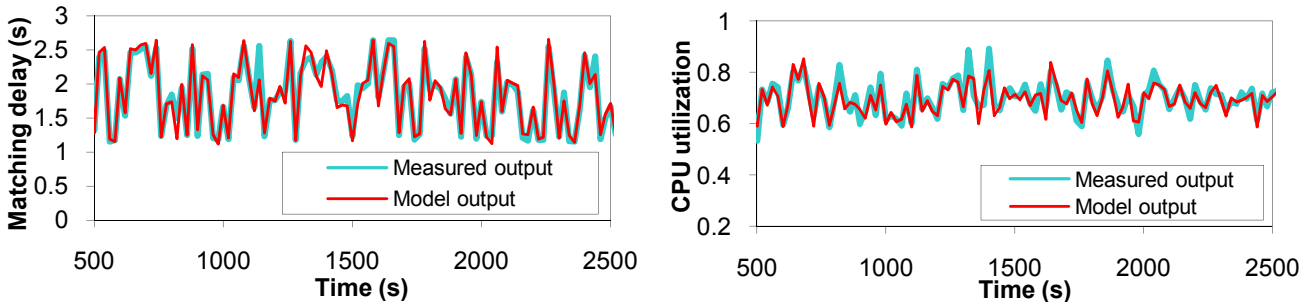
A. Problem Formulation

We first introduce some notations. T_{ref} is the set point of the average matching interval. T_a is the control period of the admission controller. $t(k)$ is the measured average matching interval in the k^{th} admission control period. $\overline{n}(k)$ and $\overline{win}(k)$ are the average job budget and the average batching window size in the k^{th} admission control period, respectively. $n_l(k)$ and $n_h(k)$ are the number of low-priority and high-priority subscriptions, respectively, admitted in the registry.

The goal of the admission controller is to guarantee that all the low-priority subscriptions are reevaluated at least once within a time constraint so that the matching interval of all the low-priority subscriptions is bounded. The decision on how many low-priority subscriptions should be admitted into the system is affected by many factors, such as the job budget, the batching window size, and the number of high-priority subscription in the registry. In this paper, we present a controller to calculate the maximum number of low-priority subscriptions periodically.

B. System Modeling

Unlike the integrated controller, we model the system in an analytical way. Since the system reevaluates all the high-priority subscriptions first and then reevaluates some low-priority subscriptions in a round-robin way, there are $(\overline{n}(k) - n_h(k))$ low-priority subscriptions reevaluated after each batching window. Note that we assume that the job budget is always larger than the number of high-priority



(a) Matching delay

(b) CPU utilization

Fig. 4. System model validation.

subscriptions in the system, *i.e.*, $(\overline{n}(k) > n_h(k))$. This is a valid assumption in that if $\overline{n}(k) < n_h(k)$, the system is unable to guarantee the average matching delay of all the high-priority subscriptions. Therefore, the average matching interval of all the low-priority subscriptions is calculated as follows:

$$t(k) = \frac{n_l(k-1)}{\overline{n}(k-1) - n_h(k-1)} \overline{win}(k-1). \tag{10}$$

The variation of $\overline{n}(k)$ and $\overline{win}(k)$ is very small during the steady state of the integrated controller. We also assume that the number of high-priority subscriptions $n_h(k)$ in the system has small variations within a certain period of time. The assumption is valid in real information dissemination systems since only most important subscriptions will be set as high-priority. As a result, we can take $\overline{n}(k) = \overline{n}$, $\overline{win}(k) = \overline{win}$ and $n_h(k) = n_h$, where $\overline{n}$, $\overline{win}$ and n_h are constants. Hence, we get the system model as follows:

$$t(k) = p n_l(k-1), \tag{11}$$

where $p = \frac{\overline{win}}{\overline{n} - n_h}$.

C. Controller Design and Analysis

Proportional-Integral(PI) control [29] can provide robust control performance despite considerable modeling errors. Based on the system model (11), we design a PI controller as follows:

$$n_l(k) = n_l(k-1) + K_1 e(k) - K_1 K_2 e(k-1), \tag{12}$$

where $e(k) = T_{ref} - t(k)$ is the control error. Using the Root-Locus method [29], we can choose our control parameters as $K_1 = 1/p$ and $K_2 = 0$ such that our closed-loop transfer function is:

$$G(z) = z^{-1}. \tag{13}$$

Now we reevaluate the control performance when the system (11) changes due to the variation of $\overline{n}$, $\overline{win}$ and n_h. Without of loss of generality, we model the overall variation as g_p and the real system at runtime can be modeled as:

$$t(k) = p' n_l(k-1), \tag{14}$$

where $p' = g_p \cdot p$. We apply the PI controller (12) on the real system model (14) to get the closed-loop transfer function at runtime as:

$$G(z) = \frac{g_p}{z - (1 - g_p)}. \tag{15}$$

Based on control theory, in order for the system to be stable, the poles of the closed-loop system at runtime (15) must be within the unit circle, that is, $0 < g_p < 2$. This analysis shows that the closed-loop system at runtime (15) can be guaranteed to be stable despite significant variations of $\overline{n}$, $\overline{win}$ and n_h, as long as those variations do not result in a p' that is greater than twice the nominal value of p. Since the average job budget $\overline{n}(k)$ increases proportionally to the increment of the average batching window size $\overline{win}(k)$ and the variations are very small in the steady state of the integrated controller, it is very rare that p' is greater than twice the value of p.

To handle models with p' that are outside the established stability range, an online model estimator, implemented in our previous work [33], can be adopted to dynamically correct the models based on the relationship between the controlled variable and manipulated variable such that system stability can be guaranteed despite significant variations.

We now analyze the steady state error of the closed-loop system at runtime (*i.e.*, $g \neq 1$). Based on the final value theorem [29], we have

$$\lim_{z \to 1}(z-1) G(z) T_{ref} \frac{z}{z-1} = \lim_{z \to 1}\left(\frac{g_p z}{z - (1 - g_p)} T_{ref}\right) = T_{ref}. \tag{16}$$

This analysis shows that the admission controller (12) can efficiently control the average matching interval of the low-priority subscriptions to the desired the reference value T_{ref} as long as the system is stable.

VI. System Implementation

In this section, we introduce the testbed we use for the experiments and the implementation details of the control loops we introduced in Section III.

A. Testbed and Application

We use the INFOD system introduced in Section I as a representative information dissemination system to test our control solution. In INFOD, while publishers, consumers, and subscribers are distributed in nature, metadata matching is a separate process from the information dissemination process itself and runs in a database system called registry to completely find all the right matching results in a centralized way. The INFOD registry is implemented in Oracle Database 11.1g on a server, which is equipped with Intel Core 2 Duo Xeon 5160 3.0GHz. Since our objective is to evaluate the performance of the control loops running in the registry, all

the distributed INFOD publishers, consumers, and subscribers are implemented on another server to generate workloads for the registry for simplified experimental setup. The workload server is equipped with AMD Athlon 64x2 4200+ 1.0GHz. The two servers are connected via a network switch.

The objective of the INFOD project is to develop a prototype information dissemination system that can find matched information among numerous sources and sinks in a timely manner. To evaluate the functionality of the prototype system, several use cases, including an emergency response system and a cyber-attack detection system, have been implemented and tested. For example, the emergency response use case introduces unique requirements for information dissemination management. In emergency situations, it is best to know the resources and entities that can be requested prior to any incident. Most pub-sub systems evaluate policies and constraints after an incident occurs. With a lot of policies defined for every possible incident, it requires an officer to be familiar with all these policies and act based on them, which is impractical. In the INFOD model, the entities are matched before an event occurs, and only the information being sent to individual entities changes based on the incident. For instance, in an industrial area, it would be beneficial if the local authorities know what resources are required and available, in order to respond to a specific industrial accident. After an accident occurs, only the decision to request these resources needs to be made. In this use case, the E911 center creates subscriptions which characterize events and the necessary actions to be taken once an event happens. The event types, the actions to be initiated, and the specific consumers to be alerted are all defined in the subscription.

In this paper, as an example, all the metadata matching algorithms, metadata updates, subscriptions, publishers, subscribers, and consumers are directly adopted from the emergency response system, which has been used in a real-world application. Since the example information dissemination system is configured for only one application, each metadata update may involve all subscriptions registered in the registry.

Metadata matching is implemented as expression filters [34] and configured as event-based tasks with a priority of 3 out of the five priorities (1 to 5, with 1 as the highest) provided by the Oracle database scheduler to let the controllers have the highest priority to run. Due to the complicate constraints involved and high running overhead of expression filters, metadata matching is the major workload in the registry compared with other workloads such as metadata updates, notification sending. The execution time of each subscription reevaluation ranges from $4ms$ to $20ms$ based on the complexity of its constraints.

According to queuing theory, the arrival of requests at a server can often be realistically characterized as a Poisson process [35]. Without loss of generality, we use updates whose arrival pattern follows the Poisson process with specific values of λ (*i.e.*, the average number of arriving updates every second) in this paper. We refer to λ in Poisson process as the *update arrival rate*. The updates in our testbed have some typical arrival rates, ranging from 2 to 10. Note that our control algorithm is not limited to Poisson process. As proven in Section IV-D, our control algorithm is insensitive to arrival patterns of updates.

B. Integrated Control Loop

We now introduce the implementation details of each component in the integrated control loop.

Updates Batcher: The updates batcher is implemented as a two-threaded daemon in Java. One thread listens on a TCP/IP socket to receive updates from the clients and records the arrival time of each update. The other thread serves as one of the actuators in the integrated control loop and is invoked from time to time based on the batching window size calculated by the integrated controller. Upon each invocation (*i.e.*, the releasing time), it calculates the average waiting time of all the updates within the batching window and invokes the scheduler in the registry. The updates batcher communicates with the registry through JDBC (Java DataBase Connectivity), a Java package enabling Java programs to execute SQL statements in databases [36].

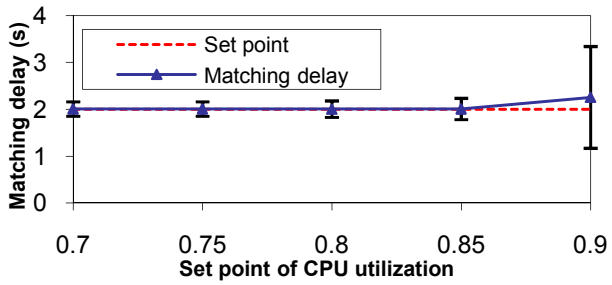
Scheduler: The scheduler is an application-level Oracle PL/SQL procedure implemented on top of the internal scheduler of the Oracle database. Oracle provides a package called DBMS_SCHEDULER from which we can call a collection of scheduling functions and procedures to manage jobs (*e.g.*, create, configure and schedule) [37]. The application-level scheduler is invoked by the updates batcher and serves as the other actuator for controlling the average matching delay and CPU utilization.

Delay and CPU utilization Monitor: The monitor is implemented as a PL/SQL procedure. The monitor calculates the average matching delay as the sum of the average waiting time of each update and average processing time of all high-priority subscriptions reevaluated in this control period, and the CPU utilization in this control period based on the statistical information in the view of V$OSSTAT in the database [38].

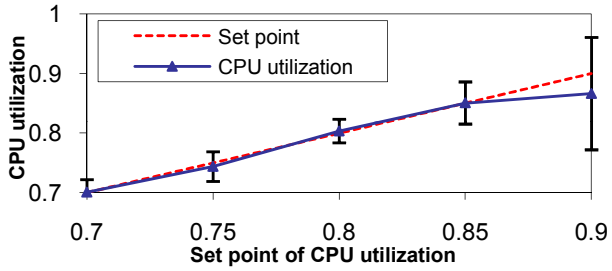
Integrated Controller: The integrated controller is a periodic job in the database. The controller is assigned the highest priority (*i.e.*, 1) such that it can be guaranteed to run periodically even when the system becomes overloaded. The period of the integrated controller is selected based on a trade-off between the sensitivity to system noise and reaction speed of the controller. On one hand, each control period should be long enough to include multiple batching windows such that the influence of the system disturbance and noise incurred to the controlled variables can be reduced. On the other hand, a longer control period leads to slower reaction to workload variations. In this paper, the batching window size calculated by the integrated controller is within the range of $(1s, 4s)$ (*i.e.*, the operating region) based on our preliminary experiments. Therefore, the control period is set as $20s$ to include at least 5 batching windows. The measured average overhead of running the whole control loop is about $100ms$, roughly 0.5% of a control period. This amount of overhead should be accpetable to most systems.

C. Admission Control Loop

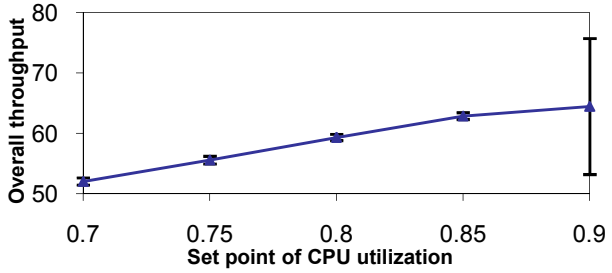
Now we introduce the implementation details of the admission control loop. The matching interval monitor is im-



(a) Matching delay



(b) CPU utilization



(c) Overall throughput

Fig. 5. Selection of set points for CPU utilization control.

plemented as a PL/SQL procedure. The average matching interval is calculated by dividing the admission control period by the number of runs for each low-priority subscription in the last control period and then taking the average. Similar to the integrated controller, the admission controller is also a periodic job in the database with the highest priority. The period of the admission controller is selected to be multiple of both the specified constraint of the matching interval and the control period of the integrated controller. In our experiments, all the low-priority subscriptions should be reevaluated at least once within a time constraint of $10s$. Therefore, the control period of the admission controller is set as $60s$ to include at least 6 reevaluations and 3 periods of the integrated controller, which is a compromise between the system noise and timely reaction of the admission controller.

VII. Experimentation

In this section, we first test different set points of CPU utilization to justify the selection of the set point for CPU utilization in our experiments. We then evaluate the integrated controller without the admission controller and compare it with two baselines. Finally, we examine the closed-loop system with both of the two controllers.

In all of the experiments, we use $2s$ as the set point of the average matching delay and 0.8 as the set point of the CPU utilization. For the admission controller, we use $10s$ as the set point of the average matching interval for all low-priority subscriptions. The steady states of both the average matching delay and CPU utilization are defined as $\pm 10\%$ of the set point.

A. Set Point Selection for CPU Utilization Control

As explained in Section IV-A, the selection of the set point of CPU utilization is a trade-off between the control accuracy of the matching delay and overall throughput. In this subsection, we test the effect of set points of CPU utilization on the control accuracy and overall throughput (*i.e.*, the number of reevaluated subscriptions per second) with experiments.

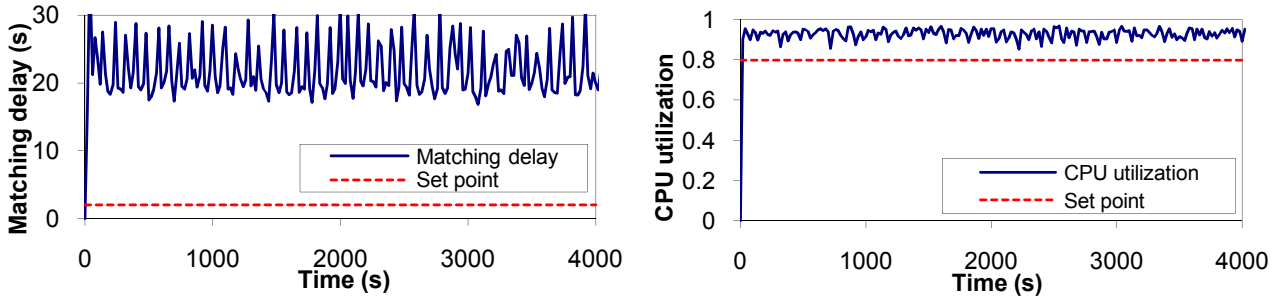
We first run the integrated controller with different set points of CPU utilization, from 0.70, 0.75, 0.80, 0.85 to 0.90, while keeping the set point of the average matching delay constantly at $2s$. The reason why we do not vary the set point of the matching delay is that the constraint of matching delay is usually specified by customers. We then plot the mean and standard deviation of 50 outputs in the steady state for the average matching delay, CPU utilization, and the overall throughput in Fig. 5. As shown in Figs. 5(a) and (b), both the average matching delay and the CPU utilization have mean values approximately equal to their set points with only small deviations when the set point of the CPU utilization is smaller than 0.9. However, as the set point increases to 0.9, both the average matching delay and CPU utilization have mean values deviating from the set point with significant oscillations. This is because the system becomes overloaded and the system model is nonlinear when the CPU utilization approaches 0.9. As shown in Fig. 5(c), the overall throughput steadily improves as the set point increases from 0.7 to 0.85 and reaches the maximal value at the CPU utilization of 0.85 with small oscillations. However, as the set point increases to 0.90, the overall throughput begins to oscillate significantly due to the large oscillation of the CPU utilization.

To ensure that the controller works safely in the operating region, we choose 0.80 as the set point to allow some leeway for the nonlinear region. This experiment gives us a good reference to choose the set point for CPU utilization.

B. Control Performance of the Integrated Controller

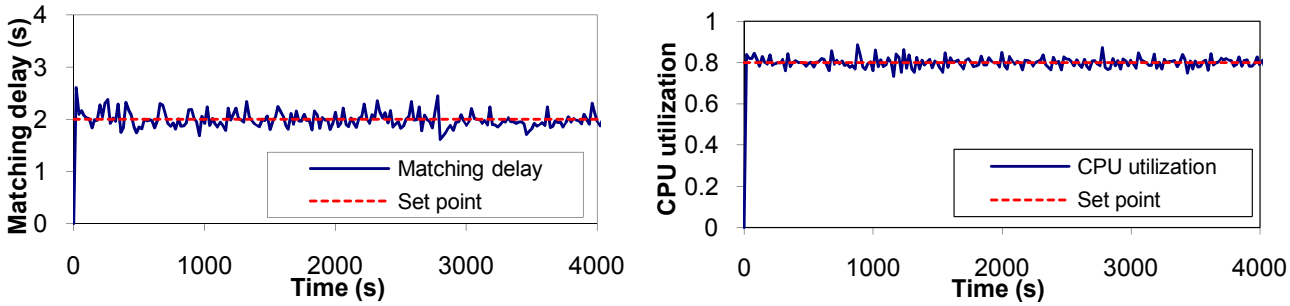
In this subsection, we first demonstrate the performance of the open-loop system to show the importance of controlling the matching delay and CPU utilization. We then evaluate the performance of the integrated controller with different update arrival rates.

The open-loop system is the system without any matching delay or CPU utilization management, in which all the subscriptions are reevaluated upon the arrival of each update. In this experiment, we use an update arrival rate of 4 (*i.e.*, $\lambda = 4$), which is a typical arrival rate in real systems. Figure 6 shows the average matching delay and CPU utilization of the open-loop system with 400 subscriptions. We can see that reevaluating all subscriptions upon each update causes severe system overload (*i.e.*, the CPU utilization is almost 0.93 with some CPU time for I/O waiting.) and unacceptably long delays (*i.e.*, the delay is greater than $30s$ at times). This



(a) Matching delay (b) CPU utilization

Fig. 6. A typical run of the open-loop system.



(a) Matching delay (b) CPU utilization

Fig. 7. A typical run of the integrated controller.

experiment shows that unbounded matching delay and system overutilization will occur without matching delay or CPU utilization management in information dissemination systems. Figure 7 shows that the integrated controller successfully achieves the desired matching delay and CPU utilization with only small oscillation.

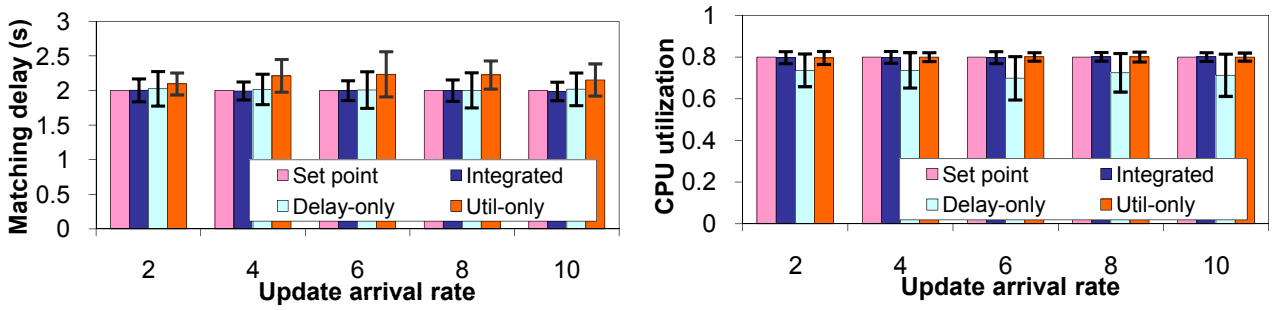
In real information dissemination systems, the update arrival rate may vary due to different factors, such as working hours. To evaluate the integrated controller under varying arrival rates, we run experiments and plot the mean and standard deviation of 50 outputs in the steady state when the arrival rates vary from 2, 4, 6, 8 to 10. The standard deviation indicates the intensity of oscillation of the outputs in the steady state. From Fig. 8, we can see that the mean of the average matching delay and CPU utilization are all around the set points with only small standard deviations (the largest is $0.18s$ and 0.03, respectively). This experiment shows that the integrated controller can, precisely, achieve the desired average matching delay and CPU utilization under different update arrival rates.

C. Comparison with Baselines

One of the advantages of the integrated controller is that it can simultaneously control both the matching delay and CPU utilization to achieve maximized system throughputs so that more valuable information can be timely disseminated. To highlight this advantage, we compare our integrated controller with two baselines: *Delay-only* and *Util-only*, in terms of control accuracy, overall throughput, and throughput for low-priority subscriptions.

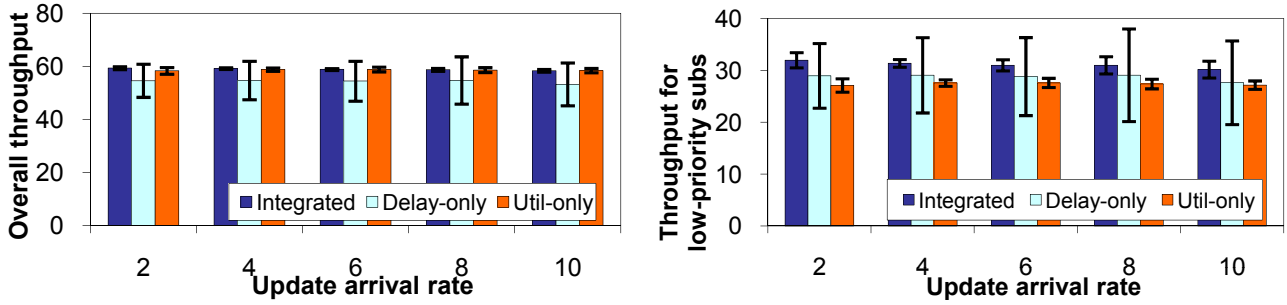
Comparison with Delay-only: Delay-only is a Single-Input-Single-Output (SISO) controller that is proposed in [8]. It has a fixed batching window size and controls only the average matching delay of the high-priority subscriptions by manipulating the job budget. We tune it with different batching window sizes to make it have the best overall throughput while controlling the average matching delay with an accuracy close to that of the integrated controller. We find that a window size of $1.95s$ produces the best results. We test Delay-only with different update arrival rates and plot the mean and standard deviation of 50 outputs in the steady state for average matching delay, CPU utilization, overall throughput, and throughput for low-priority subscriptions. As shown in Figs. 8(a), the average matching delay of Delay-only with different arrival rates is all approximately equal to the set point with small oscillations (with the largest standard deviation as $0.25s$.). However, as shown in Fig. 8(b), Delay-only has a poor CPU utilization because it only controls matching delay. As a result, Delay-only has a poor overall throughput and a poor throughput for low-priority subscriptions, as shown in Figs. 9(a) and (b). Based on this experiment, we can see that only controlling the average matching delay without controlling the CPU utilization cannot guarantee maximized system throughput for low-priority subscriptions.

Comparison with Util-only: Util-only is another SISO controller that only controls CPU utilization by manipulating the job budget with a fixed batching window size. We first fix the batching window size for Util-only at the mean of the batching window size of the integrated controller in the steady state (*i.e.*, $1.8s$). We plot the mean and standard deviation of 50 outputs in the steady state for average matching delay and CPU utilization under different update arrival rates, as shown in Figs. 8. We can see that Util-only violates the specified delay constraint though its CPU utilization converges to the set point with small oscillations. To have a fair comparison, we then tune Util-only with different batching window sizes



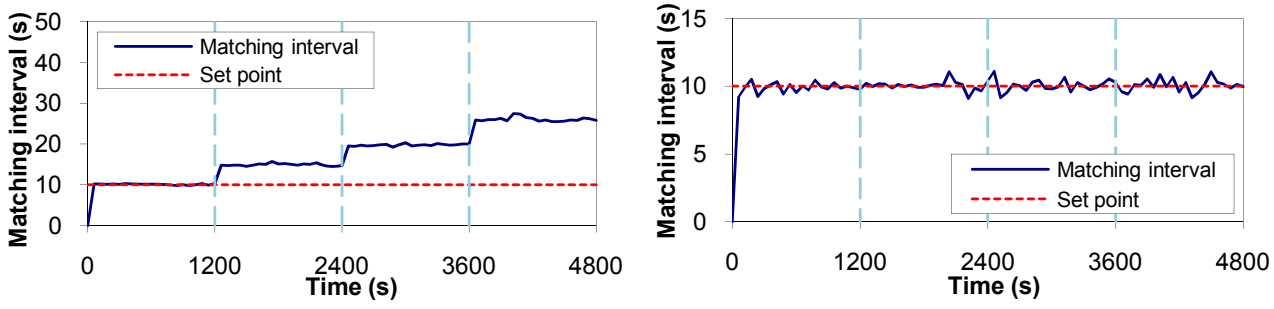
(a) Matching delay (b) CPU utilization

Fig. 8. Control accuracy comparison among Delay-only, Util-only, and the integrated controller.



(a) Overall throughput (b) Throughput for low-priority subscriptions

Fig. 9. Throughput comparison among Delay-only, Util-only, and the integrated controller.



(a) Without the admission controller (b) The admission controller

Fig. 10. Performance of the admission controller.

so that it has an average matching delay close to the set point of $2s$. By iterative tuning and testing, we find that a batching window size of $1.6s$ gives us the best results. We then compare Util-only with the integrated controller in terms of overall throughput and throughput for low-priority subscriptions. As shown in Figs. 9(a), since both Util-only and the integrated controller control the CPU utilization to the set point of 0.8, they have almost the same overall throughput. However, due to a smaller batching window size, Util-only has a smaller job budget, which results in a lower throughput for low-priority subscriptions, as shown in Fig. 9(b).

The two experiments show that only controlling either matching delay or CPU utilization results in poor system throughputs or violates the specified delay requirement. In contrast, the integrated controller, by controlling both matching delay and CPU utilization, can meet the specified constraint for matching delay and achieve maximized possible system throughputs.

D. Control Performance of the Admission Controller

In this subsection, we evaluate the system with both the admission controller and the integrated controller. This experiment emulates a scenario common to many information dissemination systems, in which new low-priority subscriptions arrive at runtime. Figure 10(a) shows the performance of the system without the admission controller (only running the integrated controller). The admission enforcer manually calculates the maximum number of low-priority subscriptions that the system is able to admit without violating the time constraint of $10s$ at the beginning. As shown in Fig. 10(a), the registry indeed achieves the desired average matching interval when the number of low-priority subscriptions remains the same. However, new low-priority subscriptions arrive at runtime and the registry admits all of the new low-priority subscriptions without any admission control. As a result, the total number of low-priority subscriptions in the registry increases from 340 to 600, 800, and 1000 at time $1200s$, $2400s$, and $3600s$, respectively. Consequently, the average matching interval in-

creases and violates the desired time constraint. Figure 10(b) shows the performance of the admission controller running together with the integrated controller in the same scenario. With the admission controller, the average matching interval is controlled to approach the set point with small oscillation. This is because that the admission enforcer denies new low-priority subscription arrivals based on the maximum number of low-priority subscriptions calculated by the admission controller. This experiment shows that the admission controller can effectively guarantee that the average matching interval of all low-priority subscriptions meets the specified time constraint.

VIII. Conclusions

Existing information dissemination systems control metadata matching delay and resource utilization separately. However, both bounded matching delay and maximized system throughputs are critical for information dissemination systems to find the maximum number of matched results and then disseminate valuable information within application-specified time constraints. In this paper, we have proposed an integrated controller to control both the matching delay of all the high-priority subscriptions and the CPU utilization of the registry server in an example information dissemination system. In addition, we have designed an admission controller to efficiently guarantee the average matching constraint for all low-priority subscriptions as well. Our empirical results on a physical testbed demonstrate that our solution can guarantee the timeliness requirements while achieving maximized system throughputs.

Acknowledgements

This work was supported, in part, by NSF under a CAREER Award CNS-0845390 and two CSR grants CNS-0720663 and CNS-0915959, by ONR under grant N00014-09-1-0750, and by a subcontract from the Department of Homeland Security-sponsored Southeast Region Research Initiative (SERRI) at the National Nuclear Security Administration's Y-12 National Security Complex.

References

[1] M. Chen, X. Wang, and B. Taylor, "Integrated control of matching delay and CPU utilization in information dissemination systems," in *17th IEEE International Workshop Quality-of-Service*, July 2009.

[2] F. Hayes-Roth, "Model-based communication networks and VIRT: orders of magnitude better for information superiority," *Military Commun. Conf.*, Oct. 2006.

[3] INFOD-WG, *Information Dissemination in the Grid Environment - Base Specifications*, Open Grid Forum (2004-2007), May 2007.

[4] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, "Dynamic voltage scaling in multi-tier web servers with end-to-end delay control," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 444-458, 2007.

[5] Y. Wang, X. Wang, M. Chen, and X. Zhu, "Power-efficient response time guarantees for virtualized enterprise servers," in *30th IEEE Real-Time Syst. Symp.*, Dec. 2008.

[6] 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," in *8th IEEE/IFIP Netw. Operations Management Symp.*, 2002.

[7] M. Chen, C. Nolan, X. Wang, S. Adhikari, F. Li, and H. Qi, "Hierarchical utilization control for real-time and resilient power grid," in *21st Euromicro Conf. Real-Time Syst.*, July 2009.

[8] M. Chen, X. Wang, R. Gunasekaran, H. Qi, and M. Shankar, "Control-based real-time metadata matching for information dissemination," in *15th IEEE International Conf. Embedded Real-Time Comput. Syst. Applicat.*, Aug. 2008.

[9] D. Kostić, A. C. Snoeren, A. Vahdat, R. Braud, C. Killian, J. W. Anderson, J. Albrecht, A. Rodriguez, and E. Vandekieft, "High-bandwidth data dissemination for large-scale distributed systems," *ACM Trans. Comput. Syst.*, vol. 26, no. 1, pp. 1-61, 2008.

[10] S. Voulgaris, E. Riviére, A.-M. Kermarrec, and M. van Steen, "Sub-2-sub: self-organizing content-based publish subscribe for dynamic large scale collaborative networks," in *5th International Workshop on Peer-to-Peer Syst.*, Feb. 2006.

[11] A. Iamnitchi and I. Foster, "Interest-aware information dissemination in small-world communities," in *Proc. 4th High Performance Distributed Comput.*, Apr. 2005.

[12] G. S. Choi, J.-H. Kim, D. Ersoz, A. B. Yoo, and C. R. Das, "Coscheduling in clusters: is it a viable alternative?" in *Super Computing*, 2004.

[13] E. Frachtenberg, D. G. Feitelson, F. Petrini, and J. Fernandez, "Adaptive parallel job scheduling with flexible coscheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 11, 2005.

[14] O. Mutlu and T. Moscibroda, "Parallelism-aware batch scheduling: enhancing both performance and fairness of shared DRAM systems," in *35th International Symp. Comput. Architecture*, 2008.

[15] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, 2003.

[16] X. Cao, J. James, J. Li, and C. Xin, "Group schedule serialized traffic in optical burst switching networks," in *4th IEEE International Conf. Broadband Commun., Netw., Syst.*, Sept. 2007.

[17] M. Mitzenmacher, "How useful is old information?" *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 1, pp. 6-20, 2000.

[18] T. Gustafsson and J.Hansson, "Data management in realtime systems: a case of on-demand updates in vehicle control systems," in *10th IEEE Real-Time Embedded Technol. Applicat. Symp.*, May 2004.

[19] K. Kang, S. Son, and J. Stankovic, "Managing deadline miss ratio and sensor data freshness in real-time databases," *IEEE Trans. Knowledge Data Eng.*, vol. 16, no. 10, Oct. 2004.

[20] M. Amirijoo, J. Hansson, and S.Son, "Specification and management of QoS in real-time databases supporting imprecise computations," *IEEE Trans. Comput.*, vol. 55, no. 3, pp. 304-319, Mar. 2006.

[21] J. R. Haritsa, M. J. Carey, and M. Livny, "Value-based scheduling in real-time database systems," *VLDB J.: Very Large Data Bases*, vol. 2, no. 2, pp. 117-152, 1993.

[22] M. Amirijoo, N. Chaufette, J. Hansson, S. H. Son, and S. Gunnarsson, "Generalized performance management of multi-class real-time imprecise data services," in *26th IEEE Real-Time Syst. Symp.*, 2005.

[23] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *4th IEEE International Conf. Autonomic Comput.*, 2007.

[24] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management," in *Proc. 8th International Symp. High-Performance Comput. Architecture*, 2002.

[25] Y. Diao, J. L. Hellerstein, S. Parekh, H. Shaikh, and M. Surendra, "Controlling quality of service in multi-tier web applications," in *26th International Conf. Distributed Comput. Syst.*, 2006.

[26] R. Zhang, C. Lu, T. F. Abdelzaher, and J. A. Stankovic, "ControlWare: a middleware architecture for feedback control of software performance," in *22th International Conf. Distributed Comput. Syst.*, July 2002.

[27] S. Parekh, J. Hellerstein, T. S. Jayram, N. Gandhi, D. Tilbury, and J. Bigus, "Using control theory to achieve service level objectives in performance management," *Real-Time Syst.*, vol. 23, no. 1/2, pp. 127-141, 2002.

[28] T. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback performance control in software services," *IEEE Control Syst.*, vol. 23, no. 3, June 2003.

[29] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd edition. Prentice Hall, 1997.

[30] M. Verhaegen and V. Verdult, *Filtering and System Identification, A Least Squares Approach*. Cambridge University Press, 2007.

[31] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.

[32] P. Dorato, C. T. Abdallah, and V. Cerone, *Linear Quadratic Control: An Introduction*. Krieger Publishing Co., 2000.

[33] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online with online model estimation," in *36th International Symp. Comput. Architecture*, 2009.

[34] *Rules Manager and Expression Filter Developer's Guide 11g Release 1 (11.1)*, Oracle, Aug. 2008.

[35] U. N. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications*. Birkhäuser, 2008.
[36] *Oracle Database JDBC Developer's Guide and Reference 11g Release 1 (11.1)*, Oracle, July 2008.
[37] D. Raphaely, *Oracle Database PL/SQL Packages and Types Reference 11g Release 1 (11.1.)*, Oracle, Sep. 2007.
[38] *Oracle Database Reference 11g Release 1 (11.1)*, Oracle, Aug. 2008.

Ming Chen received the BEng and MEng degrees in electrical engineering from Northwestern Polytechnic University, Xian, China, in 2002 and 2005, respectively. He is currently a Ph.D. candidate in computer engineering in the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville. From 2005 to 2006, he worked as a software engineer at ZTE Corp., China, developing embedded software. His research interests include real-time systems and power-aware computing.

Xiaorui Wang is an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. He received the Ph.D. degree from Washington University in St. Louis in 2006, and the B.S. degree from Southeast University, China, in 1995, both in Computer Science. He is the recipient of the NSF CAREER Award in January 2009, the Chancellor's Award for Professional Promise and the College of Engineering Research Fellow Award from the University of Tennessee in 2009 and 2010, respectively, the Power-Aware Computing Award from Microsoft Research in 2008, and the IBM Real-Time Innovation Award in 2007. He also received the Best Paper Award from the 29th IEEE Real-Time Systems Symposium (RTSS) in 2008. He is an author or coauthor of more than 50 refereed publications. In 2005, he worked at the IBM Austin Research Laboratory, designing power control algorithms for high-performance computing servers. From 1998 to 2001, he was a senior software engineer and then a project manager at Huawei Technologies Co. Ltd, China, developing distributed management systems for optical networks. His research interests include real-time embedded systems, power-aware computer systems, and cyber-physical systems. He is a member of the IEEE and the IEEE Computer Society.

Ben Taylor is currently a Master student in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. He received his B.S. degree from the same department in 2008.