Gain Scheduler Middleware: A Methodology to Enable Existing Controllers for Networked Control and Teleoperation—Part I: Networked Control

Yodyium Tipsuwan, *Member, IEEE,* and Mo-Yuen Chow, *Senior Member, IEEE*

***Abstract*—Conventionally, in order to control an application over a data network, a specific networked control or teleoperation algorithm to compensate network delay effects is usually required for controller design. Therefore, an existing controller has to be redesigned or replaced by a new controller system. This replacement process is usually costly, inconvenient, and time consuming. In this paper, a novel methodology to enable existing controllers for networked control and teleoperation by *middleware* is introduced. The proposed methodology uses middleware to modify the output of an existing controller based on a gain scheduling algorithm with respect to the current network traffic conditions. Since the existing controller can still be utilized, this approach could save much time and investment cost. Two examples of the middleware applied for networked control and teleoperation with IP network delays are given in these two companion papers. Part I of these two companion papers introduces the concept of the proposed middleware approach. Formulation, delay modeling, and optimal gain finding based on a cost function for a case study on DC motor speed control with a proportional–integral (PI) controller are also described. Simulation results of the PI controller shows that, with the existence of IP network delays, the middleware can effectively maintain the networked control system performance and stabilize the system. Part II of this paper will cover the use of the proposed middleware concept for a mobile robot teleoperation.**

***Index Terms*—Adaptive control, control systems, dc motors, distributed control, Internet, mobile robots, networks, real-time system, telerobotics.**

I. Introduction

DUE TO THE rapid advancement in data and communication network technologies, especially the Internet, real-time networked control applications including teleoperation and remote mobile robots have gained increasing attentions in industries such as factory automation and industrial electronics. By organizing wiring connections among control system devices via network resources, networked control applications can be conveniently and systematically maintained. Furthermore, this configuration also enables remote control operations or teleoperation. Nevertheless, the networked control system or teleoperation performances could be degraded and even become unstable by network-induced delays. The performance can be aggravated if the network delays are time varying and random (e.g., IP network delays) [1]. Several techniques have been developed to handle network delay effects, and some promising results have been reported. These control methodologies are based on different techniques such as optimal gain scheduling [2], buffering [3], [4], nonlinear and perturbation theory [5], optimal stochastic control [6], and sampling time scheduling [7]. Some control techniques are developed for a specific kind of applications such as robots. These techniques include robust gain scheduling [8], wave variables [9], [10], and event-based control [11]. However, applying and implementing these control techniques on existing systems that are extensively being used in industrial plants could be costly, inconvenient, and time consuming. The main reason is that all existing controllers may have to be redesigned, replaced, or reinstalled in order to be used over data networks.

Recently, there have been several efforts to apply middleware to assist networked control systems and teleoperation [12], [13]. Middleware is an implementation to seamlessly link applications and/or function calls together locally or over a network. In several implementations, middleware could also handle network resource allocation and reservation between two applications over a data network [14]–[17]. A networked control system or a teleoperation system can utilize middleware to achieve certain network conditions such as guaranteed bandwidth, delay bound, or loss rate, by negotiating with the network counterpart for resource reservation.

In these two companion papers, we introduce a novel methodology to apply middleware to enable an existing nonnetwork-based controller so it can be used for networked control and teleoperation. The proposed methodology uses middleware to modify the controller output with respect to the current network traffic conditions in addition to utilize the generic features of middleware as mentioned. Controller output modification is performed based on gain scheduling. Since the controller does not need to be replaced, reinstalled, or redesigned, the proposed approach can be cost effective, and conveniently applied on existing systems [18]. Part I of these two papers describes the fundamental concept of the middleware. A case study of applying the middleware to enable an existing PI controller for networked control is also given along with formulation, delay modeling, and optimal gain finding based on a cost function. The use of the middleware approach for a teleoperation application will be described in Part II [19].

Manuscript received September 10, 2003; revised May 31, 2004. Abstract published on the Internet September 10, 2004. This work was supported in part by the Royal Thai Government.

Y. Tipsuwan is with the Department of Computer Engineering, Kasetsart University, Bangkok 10900, Thailand (e-mail: yyt@ku.ac.th).

M.-Y. Chow is with the Advanced Diagnosis And Control (ADAC) Laboratory, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27606 USA (e-mail: chow@eos.ncsu.edu).

Digital Object Identifier 10.1109/TIE.2004.837866

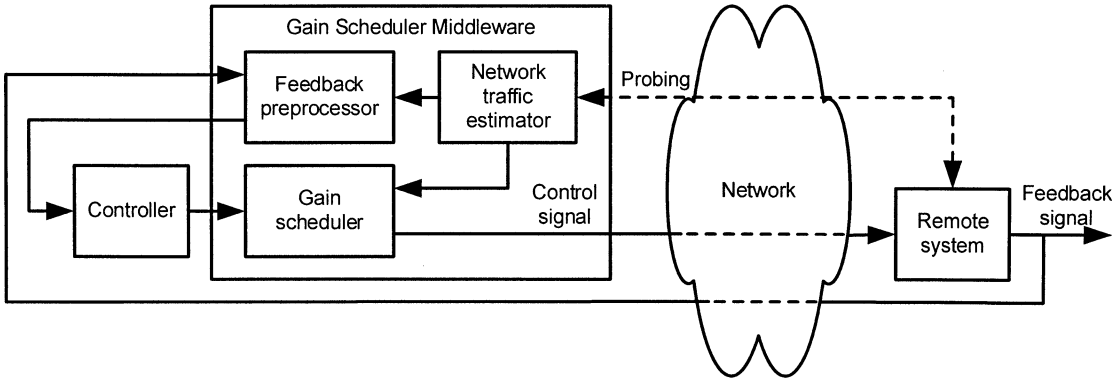

Fig. 1. Structure of GSM.

II. System Description

A. External Gain Scheduling

In order to enable an existing controller for networked control by using middleware, we will first describe the concept of external gain scheduling. The system dynamics of a remote system to be controlled can be described in a general form as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p_x}, \mathbf{q}), \tag{1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{p_x}) \tag{2}$$

and a general form of a control rule can be described by

$$\mathbf{u} = \mathbf{g}(\mathbf{y}, \boldsymbol{\gamma}\mathbf{p_u}) \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^n$ in this case is the state variable of the remote system, $\mathbf{y} \in \mathbb{R}^m$ is the remote system output, $\mathbf{u} \in \mathbb{R}^z$ is the controller output, $\mathbf{p_x} \in \mathbb{R}^w$ is the remote system parameters, $\mathbf{p_u} \in \mathbb{R}^r$ is the controller parameters, $\boldsymbol{\gamma} \in \mathbb{R}^r$ is a variable gain to adjust $\mathbf{p_u}$, and $\mathbf{q} \in \mathbb{R}^d$ is the network variable representing network traffic conditions. For example, network variables represented in $\mathbf{q}$ can be statistics of network delays such as mean delay, delay variance, loss rate, and other network quality-of-service (QoS) variables.

A method to compensate network delay effects so that the overall networked remote control system and teleoperation performances can be maintained is to adapt the controller gain $\mathbf{p_u}$ by $\boldsymbol{\gamma} \in \mathbb{R}^r$ using a gain scheduling approach. The network conditions characterized by $\mathbf{q}$ can be used as parameters to adjust $\mathbf{p_u}$ [2]. In our papers, we propose the external gain scheduling approach to enable existing controllers for networked control and teleoperation. The main concept of this approach is to find a $\boldsymbol{\beta} \in \mathbb{R}^z$ gain such that

$$\boldsymbol{\beta}\mathbf{u} = \boldsymbol{\beta}\mathbf{g}(\mathbf{y}, \mathbf{p_u}) \cong \mathbf{g}(\mathbf{y}, \boldsymbol{\gamma}\mathbf{p_u}). \tag{4}$$

The $\boldsymbol{\beta}$ gain adjusts the controller output $\mathbf{u}$. This adjustment is equivalent to adjusting the parameter $\mathbf{p_u}$ by the variable gain $\boldsymbol{\gamma}$. If the relationship between $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ can be found, we can compensate network delays by updating $\boldsymbol{\gamma}$ externally via the $\boldsymbol{\beta}$ gain. This $\boldsymbol{\beta}$ gain can be determined from different optimal objectives, and be applied on $\mathbf{u}$ to compensate network delays with respect to $\mathbf{x}$, $\mathbf{u}$, $\mathbf{p_u}$, and $\mathbf{q}$. This approach allows a convenient upgrade on an existing controller because the controller output $\mathbf{u}$ is modified externally without alternating the original control algorithm. In some systems such as in [18], the relationship between $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are linear so that the optimal $\boldsymbol{\beta}$ based on a certain objective can be obtained easily. Unfortunately, several systems such as a mobile robot path tracking control system [20] have a highly nonlinear relationship or the relationship may be too complicated to find. An alternative way to represent the relationship in this case is to obtain simulation or experimental data, and apply a lookup table or a soft-computing technique such as fuzzy logic or neural networks. In this case, a heuristic gain scheduling approach may also be able to provide satisfactory control performances.

B. Gain Scheduler Middleware (GSM)

External gain scheduling can be implemented via middleware. The middleware in this paper is defined as the *Gain Scheduler Middleware (GSM)*. We assume that the GSM handles all network connections between the controller and the remote system to be controlled over a network. These include typical network operations such as sending and receiving packets, and other general middleware operations such as negotiation and resource reservation. The structure of the GSM is shown in Fig. 1.

The basic components of the GSM shown in Fig. 1 are as follows.

1) Network Traffic Estimator: The function of the network traffic estimator unit is to estimate the current network traffic conditions, which can be characterized into $\mathbf{q}$. The network variable $\mathbf{q}$ is then utilized by feedback preprocessor and gain scheduler, depending on the control algorithm used. After initialization at the beginning of the networked control or teleoperation process, the network traffic estimator will periodically monitor the network conditions by sending a probing packet to the remote system with sampling time T_p. The estimator then characterizes the network conditions with the updated network variable $\mathbf{q}$ based on the monitored probing packet roundtrip measurement.

2) Feedback Preprocessor: The feedback preprocessor unit is used to preprocess the feedback data such as motor speed and current from the remote system before forwarding the signal to the controller. Preprocessing in this case can be, for example, filtering noises in the feedback data, or prediction of remote

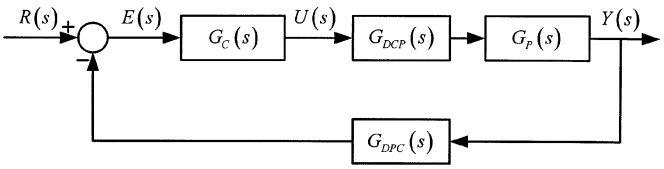

Fig. 2. A point-to-point networked control system formulation.

system states. Necessity of these operations depends on the gain scheduling algorithm used in the gain scheduler.

3) Gain Scheduler: By using an external gain scheduling algorithm, the gain scheduler unit modifies the controller output with respect to the current network conditions (characterized by $\mathbf{q}$). The algorithm to modify the controller output depends on the overall system configuration of the controller and the remote system.

The overall GSM operations for networked control and teleoperation can be summarized as follows.

1) The feedback preprocessor waits for feedback data from the remote system. Once the feedback data arrives, the preprocessor processes the data using the current values of network variables and passes the preprocessed data to the controller.
2) The controller computes the control signals and sends them to the gain scheduler.
3) The gain scheduler modifies the controller output based on the current values of network variables and sends the updated control signals to the remote system.

III. Case Study: GSM for PI DC Motor Speed Controller

A. Problem Formulation

In this section, a case study of the GSM to enable an existing PI controller for networked control is described. An application to illustrate the use of the GSM is a DC motor speed control system over IP network delays. In order to analyze how to apply the GSM for external gain scheduling on a PI controller with respect to IP network traffic conditions, let us formulate the problem mathematically in a continuous-time approach by first assuming IP network delays are constant. A typical single networked control system is formulated as shown in Fig. 2, where $R(s)$, $U(s)$, $Y(s)$, and $E(s) = R(s) - Y(s)$ are the reference, control, output, and error signals in the Laplace domain, respectively.

The plant dynamics is expressed as a transfer function $G_P(s)$, where the PI controller $G_C(s)$ is described by

$$G_C(s) = \frac{K_P\left(s + \left(\frac{K_I}{K_P}\right)\right)}{s} = \frac{K_P(s + z_C)}{s} \tag{5}$$

where K_P and K_I are the proportional gain and integral gain, respectively, and $z_C = K_I/K_P$ is a constant. The network delays for sending the control $U(s)$ to the plant $G_P(s)$, and for sending the system output $Y(s)$ to the PI controller $G_C(s)$, are represented by $G_{\mathrm{DCP}}(s)$ and $G_{\mathrm{DPC}}(s)$, respectively, of which the analytical forms are

$$G_{\mathrm{DCP}}(s) = e^{-\tau_{\mathrm{DCP}}s} \tag{6}$$

$$G_{\mathrm{DPC}}(s) = e^{-\tau_{\mathrm{DPC}}s} \tag{7}$$

where τ_{DCP} and τ_{DPC} are the delay from the controller to the plant, and the delay from the plant to the controller in time domain, respectively. The closed-loop transfer function including the network delays becomes

$$\frac{Y(s)}{R(s)} = \frac{G_C(s)\,G_{\mathrm{DCP}}(s)\,G_P(s)}{1 + G_C(s)\,G_{\mathrm{DCP}}(s)\,G_P(s)\,G_{\mathrm{DPC}}(s)}. \tag{8}$$

In order to analyze the closed-loop control system with network delay effects, a typical approach is to use a rational function with the numerator degree zero to approximate the delays as follows [21]:

$$e^{-\tau s} \cong \left(1 + \left(\frac{\tau s}{n}\right)\right)^{-n} \tag{9}$$

where τ can be τ_{DCP} or τ_{DPC}. This approximation is adequate for many practical applications because the primary branches of the system root locus usually contain the dominant eigenvalues of the system [21]. Moreover, (9) is suitable for real-time applications due to its computational simplicity.

B. DC Motor Model

The dc motor transfer function from [22] is used to represent the action agent plant dynamics, and is described by

$$G_P(s) = \frac{2029.826}{(s + 26.29)(s + 2.296)}. \tag{10}$$

Let us assume that the practical PI speed controller for the DC motor is designed and tuned without concerning for the network delays to have the relative damping ratio of 0.707 and to satisfy the following specifications with the step response:

- percentage overshoot (*P.O.*): $\mathrm{P.O.} \leq 5\%$;
- settling time (t_s): $t_s \leq 0.309$ s;
- rise time (t_r): $t_r \leq 0.117$ s.

Using the root locus design approach without considering network delays, a feasible choice of $(K_P, K_I) = \left(K_P^0, K_I^0\right) \triangleq (0.1701, 0.378)$, and these gains satisfy all the design specifications as listed. We will use the nominal gain $\left(K_P^0, K_I^0\right)$ as a baseline reference to compare with the proposed GSM when network delay effects are considered.

C. Parameterization for Gain Scheduling: Constant Network Delay

PI Controller Parameterization: With the existence of network delays in the control loop, the initial $\left(K_P^0, K_I^0\right)$ may no longer satisfy the design specifications. The system performance will also degrade, and the system may become unstable. To remain the best possible system performance with network delays, the controller gains need to be adapted with respect to the current network conditions. In this section, we apply the β gain, where $\beta \geq 0$, as a multiplicative factor to externally

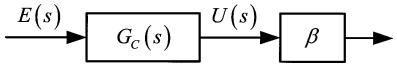

Fig. 3. Adaptation of PI controller gains at the controller output by β.

adapt $\left(K_P^0, K_I^0\right)$ without completely redesigning the existing controller. The idea of β gain adaptation is adopted from [23]. The β gain has to be greater than zero to avoid positive feedback. The β gain is placed in front of the initial PI controller as depicted in Fig. 3.

Analytically, β adjusts both K_P^0 and K_I^0, while keeping the ratio between both gains at z_C as follows:

$$\beta G_C(s) = \beta \frac{K_P^0 (s + z_C)}{s}. \tag{11}$$

This parameterization enables PI gain scheduling to be tractable for real-time online analysis with existing theories such as root locus so that the controller or the middleware could quickly analyze the system to perform additional advanced control schemes such as fault detection and diagnosis. Adjusting K_P and K_I separately with no concern about the ratio z_C could maintain equivalent or better system performance than adjusting β. However, separately adjusting K_P^0 and K_I^0 requires a more complicated approach like root contour, which is quite tedious and time-consuming. Thus, this approach may not be suitable for real-time online analysis. In addition, without β parameterization, (K_P, K_I) has to be chosen for adjustment from a two dimensional feasible region. By searching for β instead, the feasible region is only single dimensional, which is easier to search for the optimal value.

Optimizing β: In order to evaluate the best possible system performance with respect to β under different IP network conditions, we minimize the following cost function to find the optimal β:

$$J = w_1 J_1 + w_2 J_2 + w_3 J_3 \tag{12}$$

$$J_1 = \begin{cases} (\mathrm{MSE} - \mathrm{MSE}_0)^2, & \mathrm{MSE} > \mathrm{MSE}_0 \\ 0, & \mathrm{MSE} \le \mathrm{MSE}_0 \end{cases} \tag{13}$$

$$J_2 = \begin{cases} (\mathrm{P.O.} - \mathrm{P.O.}_0)^2, & \mathrm{P.O.} > \mathrm{P.O.}_0 \\ 0, & \mathrm{P.O.} \le \mathrm{P.O.}_0 \end{cases} \tag{14}$$

$$J_3 = \begin{cases} (t_r - t_{r0})^2, & t_r > t_{r0} \\ 0, & t_r \le t_{r0} \end{cases} \tag{15}$$

where

$$\mathrm{MSE} = \frac{1}{N} \sum_{k=0}^{N} e^2(k) \tag{16}$$

is the mean-squared error, MSE_0 is the nominal mean-squared error, $\mathrm{P.O.}_0$ is the nominal percentage overshoot, and t_{r0} is the nominal rise time. The weights w_1, w_2, and w_3 are used to specify the relative significance of J_1, J_2, and J_3, respectively, on the overall system performance. The error $e(k) = y(k) - r(k)$ is computed by sampling $y(t)$ at $t = kT$, where T is the sampling period, and k is the time index. The costs J_1, J_2, and J_3 are mainly used to provide the penalty when the system performance degrades from the nominal system performance. In this case, the nominal performance can be adopted from the design specifications mentioned earlier such that $\mathrm{P.O.}_0 = 5\%$, $t_{r0} = 0.117$, whereas MSE_0 has to be determined from a simulation or an experiment. In this paper, we use $\mathrm{MSE}_0 = 0.00595$. The cost J_1 gives the penalty on poorer response time and convergence, while the cost J_3 provides an extra penalty on the slower response. The cost J_2 gives the particular penalty on the higher value of the percentage overshoot. These costs will increase if the networked control system performs worst than the nominal condition. If the system performs equally to the nominal condition or better, the costs are zero. Therefore, when $\beta = 1$ without network delays in the system, $J = 0$.

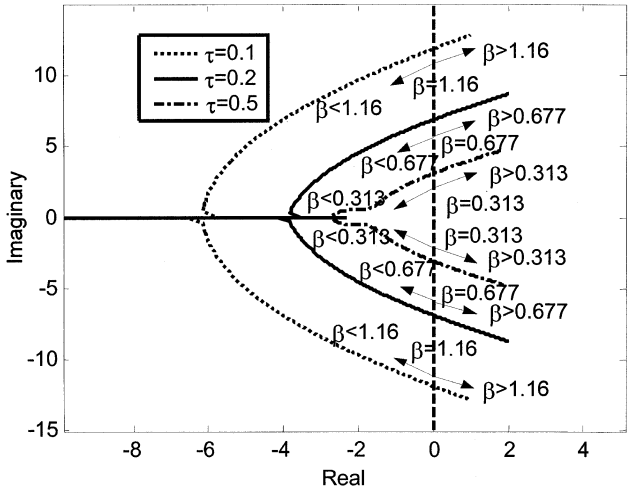

Fig. 4. Primary branches of the root locus of the networked dc motor PI speed control system using (9) to approximate network delays.

With network delays, $\beta = 1$ may no longer be optimal. Thus, the optimal gain has to be obtained by evaluating J with concern for current network delays. Unfortunately, J usually does not have a closed-form relationship with β. Therefore, a feasible approach to search for the optimal β is to rely on a simulation according to the feasible set of β. We define $\mathcal{F}$ as the feasible set containing all β that do not cause system instability. The feasible set $\mathcal{F}$ can be estimated by the root locus analysis and the approximation in (9) with the following approximated characteristic equation:

$$\begin{aligned} 1 + \beta G_C(s) G_{\mathrm{DCP}}(s) G_P(s) G_{\mathrm{DPC}}(s) &= e^{-\tau s} \frac{2029.826 \beta K_P^0 (s + z_C)}{s(s + 26.29)(s + 2.296)}, \\ &\cong \frac{2029.826 \beta K_P^0 n^n (s + z_C)}{\tau^n s (s + 26.29)(s + 2.296)\left(s + \frac{n}{\tau}\right)^n} \end{aligned} \tag{17}$$

where τ is defined as $\tau = \tau_{CP} + \tau_{PC}$. For example, by setting $n = 4$, the root locus of (17) with respect to $\tau = 0.1$,0.2, and 0.5 are shown in Fig. 4. Only the primary branches of the root locus are shown in Fig. 4 to determine the stability of the closed-loop system. As discussed in [21], the primary branches are sufficient to be used for stability region approximation.

In a stable system, β can range from 0 to the value of which the root locus crosses the imaginary axis. Thus, $\mathcal{F}$ can be defined as $\mathcal{F} = \{\beta \,|\, \beta \in [0, \beta_{\max}(\tau))\}$, where $\beta_{\max}(\tau)$ is the β gain at the point that the root locus crosses the imaginary axis with respect to τ. In fact, $\beta_{\max}(\tau)$ does not need to be very precise since the optimal β is unlikely to be close to $\beta_{\max}(\tau)$. If β

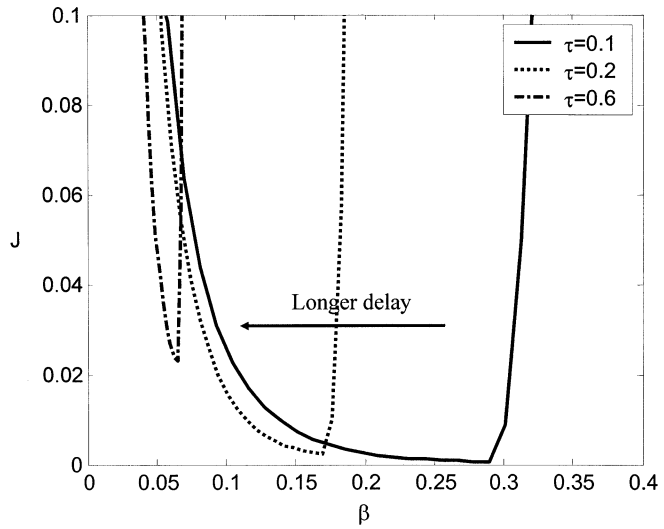

Fig. 5. Cost J from optimization using (6) and (7) to delay $U(s)$ and $Y(s)$, respectively, with $w_1 = 1.64902$, $w_2 = 0.00833$, $w_3 = 0.01395$, $t_f = 10$ s, $\tau = 0.1$, 0.2, and 0.6 s.

TABLE I
STATISTICAL MEASURES (MINIMUM, MEDIAN, MEAN, AND MAXIMUM) OF RTT DELAYS MEASURED FROM ADAC LABORATORY AT NCSU TO www.lib.ncsu.edu, www.visitnc.com, www.utexas.edu, AND www.ku.ac.th.

Destination host	$\tau_{\min}$ (sec)	τ_{median} (sec)	τ_{mean} (sec)	$\tau_{\max}$ (sec)
www.lib.ncsu.edu	0.000435	0.000471	0.000580	0.0862
www.visitnc.com	0.0166	0.0232	0.0326	0.7562
www.utexas.edu	0.0622	0.0627	0.0629	0.1187
www.ku.ac.th	0.0045	0.3150	0.3730	227.7095

is closed to $\beta_{\max}(\tau)$, J_1 and J_2 can be very large because the closed-loop system is nearly unstable. As indicated in Fig. 4, $\beta_{\max}(\tau)$ becomes smaller and less sensitive to τ when τ increases. This implies that a longer delay τ gives a smaller feasible set $\mathcal{F}$. Simple search for the optimal β for a specific τ can be easily accomplished by iteratively running simulations with various β in the feasible region, and comparing the cost J to optimize for β. For example, we optimize J using (6) and (7) as delays for $U(s)$ and $Y(s)$, respectively, in simulations with $w_1 = 1.64902$, $w_2 = 0.00833$, $w_3 = 0.01395$, $\tau = 0.1$,0.2,0.6 sec, where the final time for simulations $t_f = 10$ s. The cost J from this optimization is shown in Fig. 5.

As shown in Fig. 5, when the delay τ is longer, the optimal β shifts to the left, and J becomes more sensitive to β.

D. Parameterization for Gain Scheduling: Actual IP Network Delay

IP Network Delay Characteristics: Actual IP network delays are not constant, but stochastic in nature. In addition, the network delays are not necessarily continuous. To illustrate actual IP network delay characteristics, round-trip time (RTT) delays are measured from an Ethernet network in the Advanced Diagnosis And Control (ADAC) Laboratory at North Carolina State University (NCSU), Raleigh, to the destinations listed in Table I for 24 h (00:00–24:00). Statistical measures of the RTT delays are also shown in Table I. The corresponding histograms of the RTT delays to approximate probability densities are shown in Fig. 6.

Parameterization of IP Network Delay Characteristics for Gain Scheduling: Fig. 6 shows that the histograms skew to the left. These shapes of the histograms indicate the higher probability to have RTT delay that is shorter than the median and mean. RTT delay can be much longer than the median and mean, but with much lower probability. In Part I of this paper, we will focus more on well-regulated traffic IP networks with a small number of hops that have the probability density of RTT delay similar to Fig. 6(a) and (b). To investigate how this stochastic behavior can affect the optimality of β, RTT delay is modeled by a random probability distribution. The random distribution model should be simple in order to estimate in real-time for gain scheduling, while providing reasonable accuracy in representing different IP network conditions. Based on these reasons, we propose to use the generalized exponential distribution to describe IP network delays as follows:

$$P[\tau] = \begin{cases} \frac{1}{\phi} e^{-(\tau-\eta)/\phi}, & \tau \geq \eta \\ 0, & \tau < \eta \end{cases} \tag{18}$$

where the expected value of the RTT delay $E[\tau] = \phi + \eta$, and variance $\sigma^2 = \phi^2$. If η is known, ϕ can be easily approximated from η, and an experimental value of $E[\tau]$ or the mean μ. Some researchers have also used the exponential distribution to approximate IP network delays for control over IP networks [24]. The variables η and ϕ can be thought of as network variables in the form of $\mathbf{q} = [\eta \quad \phi]^T$. An important concern is how to select η in real-time. Both η and ϕ parameters have to be updated frequently with respect to the current IP traffic condition. This concern relates to how to generalize the approach based on a constant τ in the previous section. RTT delay can be treated as the delay τ except that RTT delay is stochastic and happens as discrete events. In this paper, we treat the IP network stochastic behavior as a parameter variation of the system transfer function. Therefore, η should be an appropriate value to serve as a base for the parameter variation described as $\tau = \eta + \Delta\tau$, where $\Delta\tau$ is the delay parameter variation. Also, based on (18), $P[\tau = \eta]$ should be the peak of the probability density function. A feasible choice of η is the median of RTT delay. The median can be easily computed in real-time and is representative for a majority of RTT delay. For example, the medians in Table I are very representative because they locate closely to the peaks of the histograms in Fig. 6, which are the majority of RTT delays. We ignore the case $\Delta\tau < 0$ (i.e., $\tau < \eta$) since this variation is relatively small. In addition, $P[\tau = \eta]$ could be used as the worst case RTT delay distribution for $\tau < \eta$. Based on (18) and the RTT delay statistics in Table I, we also assume that the ideal RTT delays have $\mu > \eta$.

Optimizing β With Actual IP Network Delay Concern: The controller used in the real IP network environment has to be a discrete-time PI controller to support actual IP packet transmissions. The PI controller can be discretized by several approximation methods such as forward Euler, backward Euler, or Tustin approximations. In this paper, we use a discrete-time PI controller approximated by forward Euler approximation as an example. With the forward Euler approximation, a stable continuous-time system could be possibly mapped in an unstable discrete-time system [25]. The optimal β obtained by the procedure for a continuous-time PI controller may cause instability when

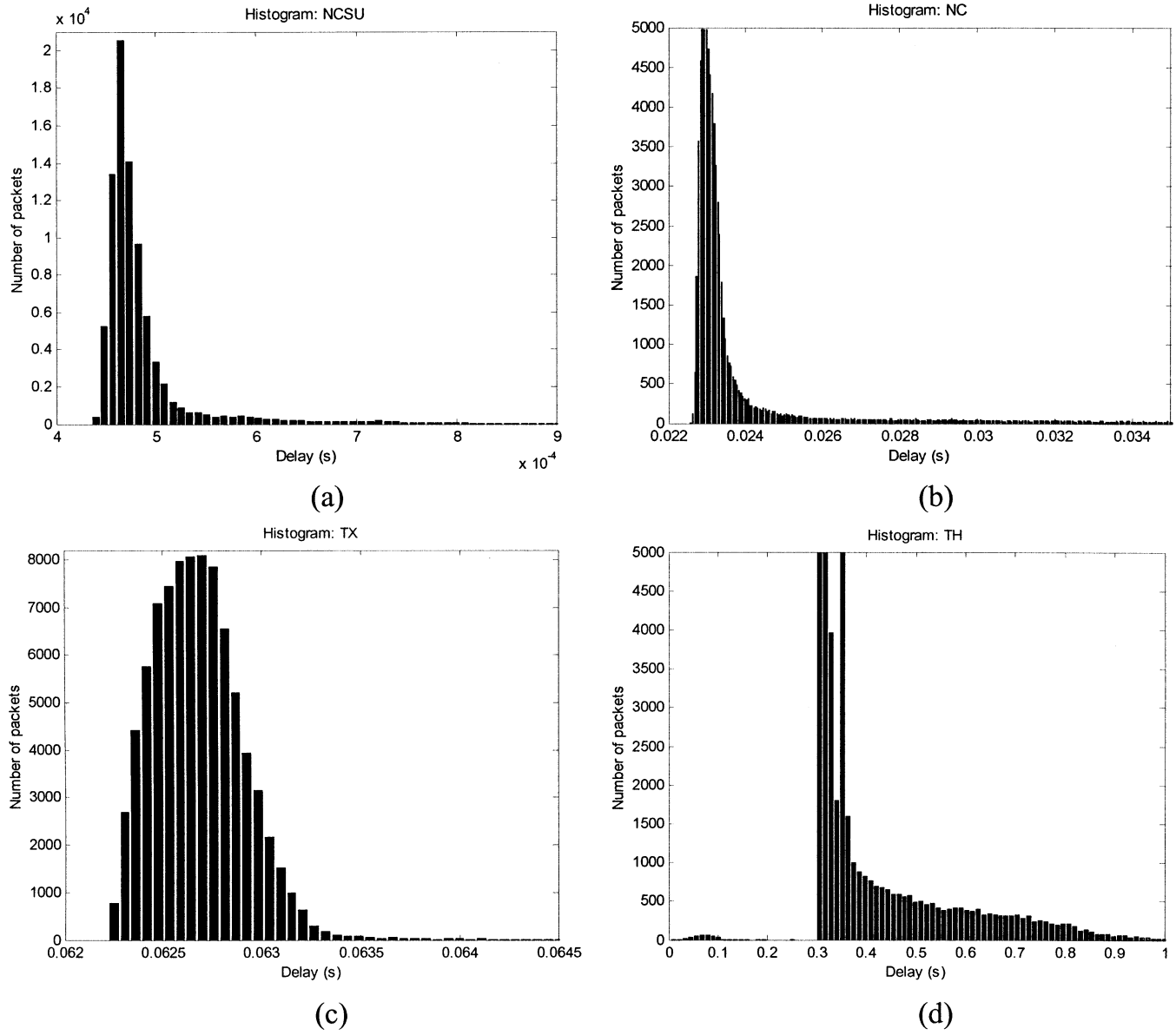

Fig. 6. Histograms of RTT delays measured from ADAC Laboratory at NCSU to (a) www.lib.ncsu.edu, (b) www.visitnc.com, (c) www.utexas.edu, and (d) www.ku.ac.th.

the corresponding discrete-time PI controller is used. Therefore, the optimal β for a discrete-time PI controller should be obtained by using the discrete-time controller in the gain searching directly, where $\beta_{\max}(\tau)$ obtained from the root locus in the continuous-time domain can still be used. The sampling time of the controller in this paper is defined as $T = 1$ ms so that the discrete-time PI controller behaves closely to the continuous-time controller at the nominal condition without violating the performance specifications as mentioned earlier.

If RTT delay is constant, ϕ will be zero, and the optimal β for $\tau = \eta$ can be immediately applied. With the stochastic behavior of actual IP networks, ϕ could affect the optimal setting of the PI controller as the delay transfer functions in the control loop are changed. Thus, the optimal β under different actual IP network traffic conditions has to be evaluated with respect to the updated η and ϕ. To find the optimal β with respect to the updated η and ϕ, the same iterative simulation approach as for constant delays can be used except that the delay τ has to vary by the generalized exponential distribution in (18) with given η and ϕ. In addition, the PI controller in simulations has to be a discrete-time controller to support actual IP packet transmission, but $\beta_{\max}(\tau)$ obtained from the root locus in the continuous-time domain can still be used to approximate feasible regions. Fig. 7 shows the cost J with respect to ϕ =0, 0.002, 0.004, and 0.006, while holding $\eta = 0.01$.

As shown in Fig. 7, the optimal β shifts to the left and decreases in value when there is more delay variation indicated by a higher value of ϕ. A higher η also lowers the optimal β.

E. Gain Scheduler Middleware for PI Controller

A possible solution to handle the variable traffic condition on actual IP networks is to adjust $\left(K_P^0, K_I^0\right)$ via GSM by scheduling β with respect to the current traffic conditions characterized by η and ϕ. We assume that the GSM is implemented as a hardware or software part, which is physically (in hardware) or

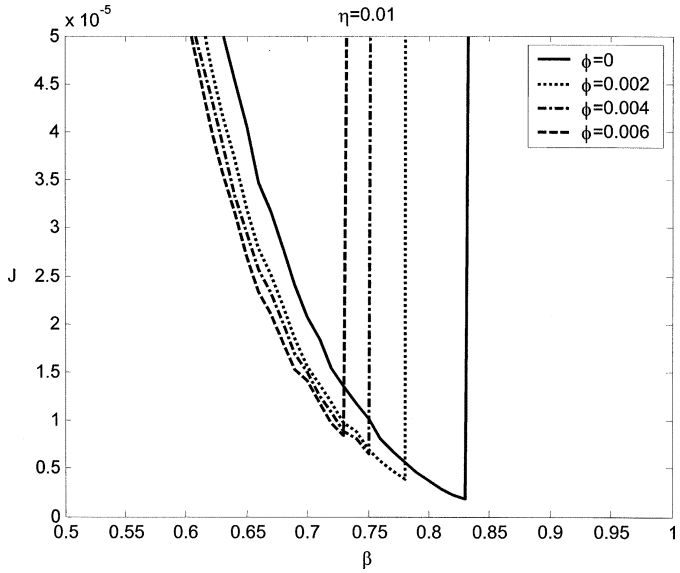

Fig. 7. Typical effect of various ϕ on the optimal β selecting while holding η constant at $\eta = 0.01$.

virtually (in software) attached in front of the original PI controller, respectively. For PI control system formulation, the feedback preprocessor is not necessary. Thus, we will describe the GSM of the PI controller for DC motor speed control by focusing on the gain scheduler and network traffic estimator parts. The operations of the GSM to enable an existing PI controller for networked control with IP network delays are described as follows.

1) The network traffic estimator initializes the packet index defined as $i = 0, \ldots, N$, to $i = 0$, the summation of RTT delays defined as m to $m = 0$, and the number of successful packet roundtrips defined as n to $n = 0$ to be used in later steps.
2) The gain scheduler captures and puts $u(t)$ in a UDP packet at every sampling time T with i, and the current time defined as the sending time $t_S(i)$. The control $u(t)$ in the packet i is defined as $u(t,i)$ for future reference. The packet is sent out immediately if possible. However, the network may not be always available for a transmission. Thus, the packet i may have to be stored in the output queue to wait for sending at the instant that the IP network is ready. Once the packet i is pushed in the queue, or sent out immediately without being stored, the gain scheduler will increase the packet index by $i = i+1$.
3) The plant will return the output $y(t)$, i, and $t_S(i)$, as a packet back to the GSM once it receives and processes $u(t,i)$ periodically using the sampling period T. This corresponding feedback is defined as $y(t,i)$. Likewise, we assume the plant has the same queuing mechanism to handle outgoing packets.
4) When the GSM receives a packet containing $y(t,i)$ from the plant at time t during a sampling period, the network traffic estimator will compute

$$rtt(i) = t_A(i) - t_S(i) \tag{19}$$

$$m = m + rtt(i) \tag{20}$$

$$n = n + 1 \tag{21}$$

where $rtt(i)$ is the RTT delay of the packet roundtrip i, and $t_A(i)$ is the arrival time of the corresponding feedback packet i. The summation of m is used to later compute the mean μ. The network traffic estimator will store $rtt(i)$ in memory along with other RTT denoted as $rtt(j), \forall j \in \mathbb{N} < i$ that are previously computed. The RTT delay $rtt(i)$ is placed in the memory, at which $rtt(a) < rtt(i) \leq rtt(b), \forall a, \forall b \in \mathbb{N} \leq i$ for sorting RTT delays in the memory to later compute η. For future reference, the RTT delay stored at the position l in the memory is defined as $RTT[l]$.

Packets transmitted between the GSM and the plant may be lost because of several reasons such as IP network congestion and a router's packet dropping policy. Therefore, there would be some unsuccessful packet roundtrips. In this case, the GSM and the plant will opt to use the most updated data to compute $u(t,i)$ and $y(t,i)$, respectively. In this paper, we focus on the effect of IP network delay and variation, and assume that the number of unsuccessful packet roundtrips is small such that it does not significantly affect the control performance.

5) Once $n = N$, the GSM will calculate

$$\mu = \frac{m}{N} \tag{22}$$

$$\eta = \begin{cases} \dfrac{\left(RTT\left[\left(\frac{N}{2}\right)\right] + RTT\left[\left(\frac{N}{2}\right)+1\right]\right)}{2}, & N \text{ is even} \\ RTT\left[\left\lceil \frac{N}{2} \right\rceil\right], & N \text{ is odd} \end{cases} \tag{23}$$

$$\phi = \mu - \eta \tag{24}$$

where N is the number of packets used to approximate the characteristic of RTT delay. When $\mu \leq \eta$, $\phi = 0$ and η becomes the representative worst-case delay to avoid a negative ϕ, which violates the shape of (18). The network traffic estimator then updates β by picking the optimal β from the lookup table with respect to η and ϕ. Steps 1)–5) will be repeated for the next iteration.

IV. Simulation Results

The performance of the proposed GSM is verified by simulations implemented on Matlab/Simulink 6.1. The following environment is used to illustrate the effectiveness of the GSM control scheme.

- The steady-state reference value $c = 1$.
- The final simulation time $t_f = 10$ s.
- The size of the optimal β lookup table $\eta \times \phi$ is 14×26, where $\eta = 0.01, 0.02, 0.03, \ldots, 0.09, 0.1, 0.2, 0.3, \ldots, 0.5$, and $\phi = 0, 0.001, 0.002, \ldots, 0.019, 0.02, 0.03, 0.04, \ldots, 0.07$. The optimal β from η and ϕ that is not in the table is obtained by the linear-interpolation technique.
- The sampling time of the PI controller, the GSM, and the plant $T = 1$ ms.
- The number of packets to evaluate the characteristic of RTT delays $N = 100$.

To investigate the effectiveness of the GSM on actual RTT delays, the three scenarios are simulated by the RTT delay data sets measured from the ADAC Laboratory at NCSU to the destinations in Table I.

TABLE II
COSTS J FROM NETWORK DC MOTOR PI SPEED CONTROL SIMULATIONS USING RTT DELAYS FROM ADAC LABORATORY AT NCSU TO (a) www.lib.ncsu.edu, (b) www.visitnc.com, (c) www.utexas.edu, AND (d) www.ku.ac.th

Control scheme	Destination host			
	www.lib.ncsu.edu	www.visitnc.com	www.utexas.edu	www.ku.ac.th
Nominal gains	0	1.763	28.057	7.141e+033
Fixed β	0	4.997e-005	2.052e-004	0.006
GSM	0	3.611e-005	2.015e-004	0.006

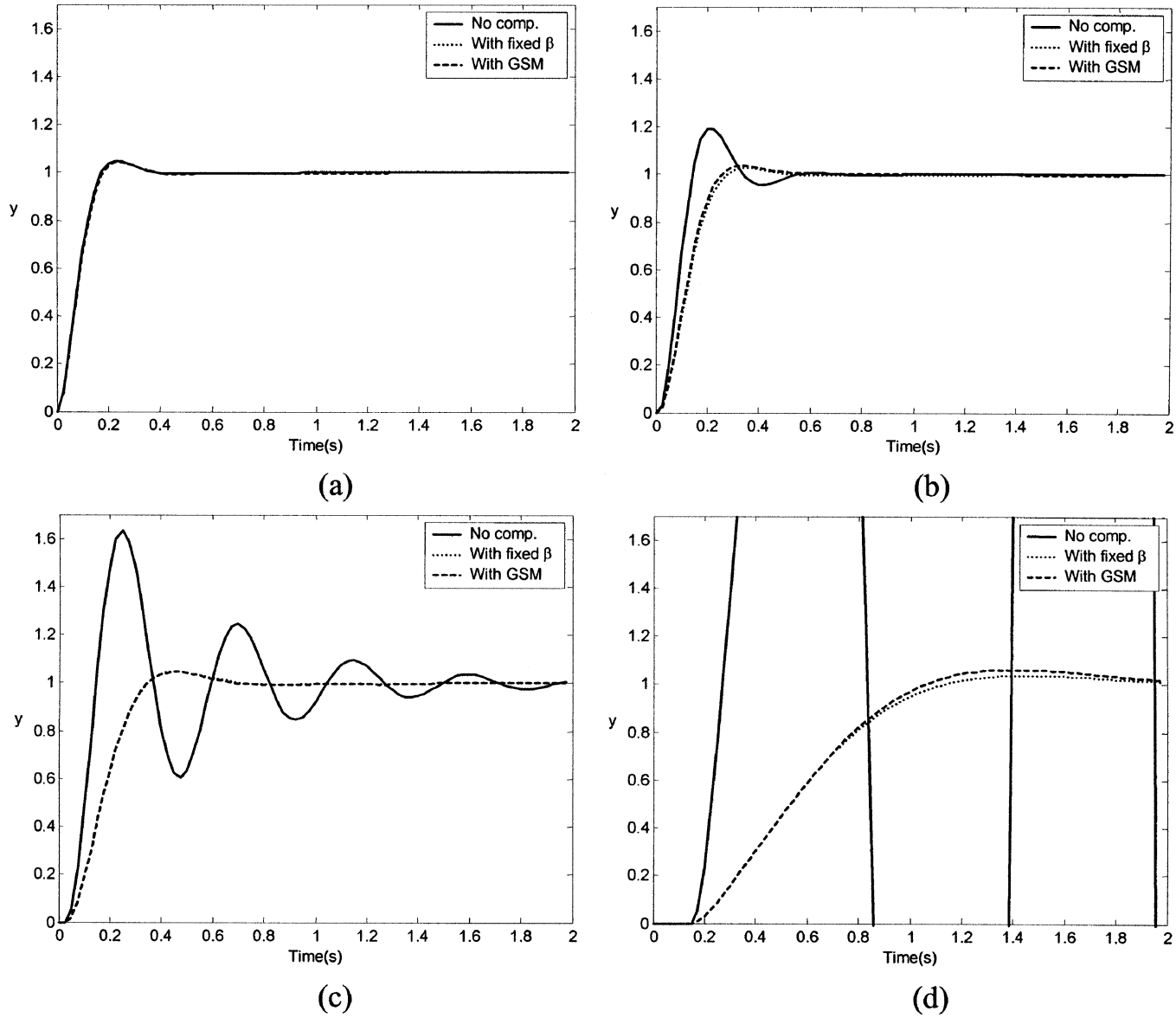

Fig. 8. Step responses from the networked dc motor PI speed control simulation using RTT delays from ADAC Laboratory at NCSU to (a) www.lib.ncsu.edu, (b) www.visitnc.com, (c) www.utexas.edu, and (d) www.ku.ac.th.

1) The dc motor is controlled over IP networks by the PI controller with the nominal gains $\left(K_P^0, K_I^0\right)$ and no delay compensation.
2) The dc motor is controlled over IP networks by the PI controller with a fixed β. The fixed β is obtained by pre-estimating η and ϕ from an RTT delay data set.
3) The dc motor is controlled over IP networks by the PI controller with the GSM using η and ϕ from real-time measurements. In this paper, we assume that the delay from the GSM to the DC motor (τ_{CP}) and the delay from the DC motor to the GSM (τ_{PC}) have similar characteristics. Therefore, we prepare these delays for the simulations by dividing all data points in an RTT delay data set by two, and selecting some values from this set to apply as τ_{CP}, and τ_{PC}. Each value of τ_{CP} chosen from data points in the RTT data set is different from the data points used for τ_{PC}. The costs J from the simulations with actual RTT delays on the three scenarios are shown in Table II, whereas the step responses from the simulations are illustrated in Fig. 8.

The performance of the PI controller using the nominal gains is significantly lower when RTT delay is longer and its variation is larger as shown in Table II and Fig. 8. The networked PI control system using fixed β and the GSM could maintain the system performance much better. As shown in Table II and Fig. 8(a), all control schemes can satisfy the performance requirements by resulting in $J = 0$ because RTT delay and its variation are relatively low. With longer RTT delay and more delay variation, the requirements cannot be satisfied as shown in Table II, and Fig. 8(b) and (c). Nevertheless, the fixed β and the GSM schemes can still maintain the system performance satisfactorily. The performance is much better than the system with the nominal $\left(K_P^0, K_I^0\right)$ since the PI controller gains in this case are adapted to be more suitable for the network traffic conditions. However, as shown in Table II and Fig. 8(d), both schemes cannot perfectly maintain the system performance to meet the specifications with very long RTT delay and very high variation, but can still reasonably stabilize the networked control system. In addition, the PI controllers using the pre-computed optimal β and the GSM with respect to real-time RTT delay characteristics have similar performance. The only exception in these examples is that when RTT delays are very long with high variation (e.g., RTT delays from ADAC Lab to www.ku.ac.th). In actual IP networks, the probability density of RTT delay may vary, and sometimes the variation can be large. Therefore, the fixed optimal β gain approach may no longer be suitable in some situations such as IP network congestion. Using the GSM to dynamically adjust the PI controller gains could provide an acceptable and more flexible control in a real IP environment.

V. Conclusion

Part I of this paper has proposed the concept of external gain scheduling via the GSM. The GSM is used to adjust the controller gains externally at the controller output with respect to the current network traffic conditions without interrupting the internal design or structure of the existing controller. An example of the GSM to enable an existing controller for networked control is given in Part I by using a PI controller for DC motor speed control with IP network delays as a case study. The PI control system is initially formulated with constant network delays, which are approximated by a rational function. The concepts and methods for constant delays can be extended to enable a PI controller to operate over actual IP network delays based on RTT delay measurements and the generalized exponential distribution model. The concept of finding the optimal β to schedule the PI controller externally at the controller output is extended to the distribution model by considering the median of RTT delays η as the constant delay and letting the delay variation be parameterized by ϕ. The GSM iteratively estimates the parameters η and ϕ of the distribution model from a number of RTT delay measurements in real time. Under reasonably long random IP network delays, the GSM can adapt the controller gain suitably for the current traffic conditions and maintain the system performance in a satisfactory level much better than the controller with the nominal gains. Part II of these companion papers [19] will cover the use of the GSM for a mobile robot teleoperation.

Acknowledgment

The authors would like to acknowledge Maxon Precision Motor, Inc. for the motor donation.

References

[1] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control Eng. Practice*, vol. 11, no. 10, pp. 1099–1111, 2003.

[2] M.-Y. Chow and Y. Tipsuwan, "Gain adaptation of networked DC motor controllers based on QoS variations," *IEEE Trans. Ind. Electron.*, vol. 50, pp. 936–943, Oct. 2003.

[3] R. Luck and A. Ray, "An observer-based compensator for distributed delays," *Automatica*, vol. 26, no. 5, pp. 903–908, 1990.

[4] H. Chan and Ü. Özgüner, "Closed-loop control of systems over a communication network with queues," *Int. J. Control*, vol. 62, no. 3, pp. 493–510, 1995.

[5] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Trans. Contr. Syst. Technol.*, vol. 10, pp. 438–446, May 2002.

[6] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," *Automatica*, vol. 34, no. 1, pp. 57–64, 1998.

[7] S. H. Hong, "Scheduling algorithm of data sampling times in the integrated communication and control systems," *IEEE Trans. Contr. Syst. Technol.*, vol. 3, pp. 225–230, June 1995.

[8] A. Sano, H. Fujimoto, and M. Tanaka, "Gain-scheduled compensation for time delay of bilateral teleoperation systems," in *Proc. IEEE ICRA'98*, Leuven, Belgium, 1998, pp. 1916–1923.

[9] S. Munir and W. J. Book, "Internet based teleoperation using wave variables with prediction," *IEEE/ASME Trans. Mechatron.*, vol. 7, pp. 124–133, June 2002.

[10] C. Benedetti, M. Franchini, and P. Fiorini, "Stable tracking in variable time-delay teleoperation," in *Proc. IEEE/RSJ IROS'01*, Maui, HI, 2001, pp. 2252–2257.

[11] K. Brady and T.-J. Tarn, "Internet-based teleoperation," in *Proc. IEEE ICRA'01*, Seoul, Korea, 2001, pp. 644–649.

[12] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro-middleware for mobile robot applications," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 493–497, Aug. 2002.

[13] D. Brugali and M. E. Fayad, "Distributed computing in robotics and automation," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 409–420, Aug. 2002.

[14] D. C. Schmidt, "Middleware techniques and optimizations for real-time, embedded systems," in *Proc. Int. Symp. System Synthesis*, San Jose, CA, 1999, pp. 12–16.

[15] B. Li and K. Nahrstedt, "A control-based middleware framework for Quality-of-Service adaptations," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1632–1650, Sept. 1999.

[16] S. Song, J. Huang, P. Kappler, R. Freimark, and T. Kozlik, "Fault-tolerant Ethernet middleware for IP-based process control networks," in *Proc. IEEE Conf. Local Computer Networks*, Tampa, FL, 2000, pp. 116–125.

[17] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS negotiation in real-time systems and its application to automated flight control," *IEEE Trans. Comput.*, vol. 49, pp. 1170–1183, Nov. 2000.

[18] Y. Tipsuwan and M.-Y. Chow, "On the gain scheduling for networked PI controller over IP network," in *Proc. IEEE/ASME AIM'03*, Kobe, Japan, 2003, pp. 640–645.

[19] Y. Tipsuwan and M.-Y. Chow, "Gain scheduler middleware: A methodology to enable existing controllers for networked control and teleoperation—Part II: Teleoperation," *IEEE Trans. Ind. Electron.*, vol. 51, pp. 1228–1237, Dec. 2004.

[20] ——, "Gain adaptation of networked mobile robot to compensate QoS deterioration," in *Proc. IEEE IECON'02*, Seville, Spain, 2002, pp. 3146–3151.

[21] B. C. Kuo, *Automatic Control Systems*, 5 ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[22] Y. Tipsuwan and M.-Y. Chow, "Fuzzy logic microcontroller implementation for DC motor speed control," in *Proc. IEEE IECON '99*, San Jose, CA, 1999, pp. 1271–1276.

[23] N. B. Almutairi and M.-Y. Chow, "PI parameterization using adaptive fuzzy modulation (AFM) for networked control systems-Part I: Partial adaptation," in *Proc. IEEE IECON'02*, Seville, Spain, 2002, pp. 3152–3157.

[24] J. W. Park and J. M. Lee, "Transmission modeling and simulation for internet-based control," in *Proc. IEEE IECON'01*, Denver, CO, 2001, pp. 165–169.

[25] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 2 ed. Englewood Cliffs, NJ: Prentice-Hall, 1990.

Yodyium Tipsuwan (S'99–M'04) was born in Chiangmai, Thailand. He received the B.Eng. degree (with honors) in computer engineering from Kasetsart University, Bangkok, Thailand, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from North Carolina State University, Raleigh, in 1999 and 2003, respectively.

Following receipt of the Ph.D. degree, he joined the Department of Computer Engineering, Kasetsart University, as an Instructor. His main interests are distributed networked control systems, mobile robotics, and computational intelligence. His current research topic is networked control over IP networks.

Dr. Tipsuwan was awarded the Royal Thai Scholarship for his graduate studies.

Mo-Yuen Chow (S'81–M'82–SM'93) received the B.S. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 1982, and the M.Eng. and Ph.D. degrees from Cornell University, Ithaca, NY, in 1983 and 1987, respectively.

Upon completion of the Ph.D. degree, he joined the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, as an Assistant Professor. He became an Associate Professor in 1993 and a Professor in 1999. He was a Senior Research Scientist with the U.S. Army TACOM TARDEC Division during the summer of 2003. He spent his sabbatical leave in 1995 as a Visiting Scientist in the ABB Automated Distribution Division. He has also been a consultant to Duke Power Company, Otis Elevator Company, Taiwan Power Company, J. W. Harley Company and a faculty intern at Duke Power Company. His core technology is diagnosis and control, artificial neural networks, and fuzzy logic. Since 1987, he has been applying his core technology to areas including motor systems, power distribution systems, network-based distributed control systems, and unmanned vehicles. He has served as a Principal Investigator in several projects supported by the National Science Foundation, Center for Advanced Computing and Communication, Nortel Company, Electric Power Research Institute, Duke Power Company, ABB Company, Electric Power Research Center, NASA, and the U.S. Army. He established the Advanced Diagnosis and Control (ADAC) Laboratory at North Carolina State University. He has authored one book, several book chapters, and over 100 journal and conference articles related to his research work.

Prof. Chow is an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS and an AdCom Member of the IEEE Industrial Electronics Society (IES). He served as the IES Vice President of Member Activities during 2000–2001.