On the Gain Scheduling for Networked PI Controller Over IP Network

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

***Abstract*—The potential use of networks for real-time high-performance control and automation is enormous and appealing. Replacing a widely used proportional-integral (PI) controller by a new networked controller for networked control capability can be costly and time-consuming. This paper proposes a methodology based on gain scheduling with respect to real-time IP traffic conditions to enhance the existing PI controller so it can be used over IP networks with a general network protocol like Ethernet. This paper first describes the gain scheduling approach based on constant network delays using a rational function approach. The formulation is extended to random IP network round-trip time (RTT) delays by using the generalized exponential distribution model. Simulation results show that the PI controller with gain scheduling provides significantly better networked control system performance.**

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

I. Introduction

AN ADVANCING trend is to substitute specialized industrial networks with a general network such as Ethernet to control applications in distributed control, industrial electronics, and factory automation areas remotely over an Internet Protocol (IP) network [1]–[3]. A protocol like Ethernet has several advantages due to its well-developed infrastructure, widespread usage, and affordability, for IP connection. Of course, a controller can be connected to a plant to form a local closed-loop control system and the status of the system can be monitored via an IP network as well as simple on-off control. However, this configuration does not have good interaction with an operator or an automatic controller located on a remote side. When an anomaly occurs, the remote side can only notice it from reported messages and may not be able to respond in time. Researchers have proposed a scheme to have a controller connected to a plant via a network as a closed-loop networked control system [1] so that the remote controller could interact with the plant via the network more quickly. Several methodologies on this topic have been developed to handle the time-varying network delay effects. These methodologies are based on various techniques such as state augmentation [4], optimal stochastic control [5], nonlinear control [6], robust control [7], buffering and prediction [8], [9], Smith predictor [10], event-based control [11], sampling time scheduling [12], [13], optimal gain scheduling [14], scattering transformation [15], and fuzzy logic [16], [17]. Many of these techniques require a completely redesigned controller to handle network delays. Practical controllers that exist in industrial applications such as a proportional-integral (PI) controller have to be replaced. This replacement is time-consuming and costly.

Manuscript received January 20, 2004; revised May 4, 2004. This work was supported in part by the Royal Thai Government.

Y. Tipsuwan was with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911 USA. He is now with the Department of Computer Engineering, Kasetsart University, Bangkok, 10900 Thailand (e-mail: yyt@ku.ac.th).

M.-Y. Chow is with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911 USA (e-mail: chow@ncsu.edu).

Digital Object Identifier 10.1109/TMECH.2004.834645

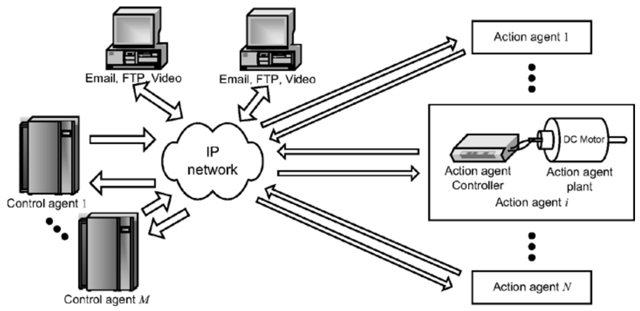

Fig. 1. Overall distributed networked control system over an IP network.

This paper proposes a methodology that enhances the widely used PI controller so it can be used over IP networks with a general computer protocol like Ethernet without requiring redesign and reinstallation of a new networked control system. The proposed approach is based on PI controller gain scheduling. The optimal PI controller gains are scheduled in real time with respect to the monitored IP network traffic condition in order to maintain the best possible system performance. Therefore, changes in IP network traffic conditions are always captured by the proposed approach.

II. System Description

A. System Configuration

In this paper, we consider a distributed networked control system configured over an IP network as shown in Fig. 1.

1) IP Network: The IP network under consideration links all networked control devices. In this paper, we assume that the networked control devices use the User Datagram Protocol (UDP) as the layer-4 protocol on the IP network to avoid additional delays from retransmissions.

2) Control Agent: Each control agent can be a high-performance computing unit to manage operations of action agents. In this paper, we consider one of the control agents that uses a PI controller to control each action agent in the low level. Periodically, the control agent converts the sensory signals in a packet sent across the IP network from each action agent to numerical

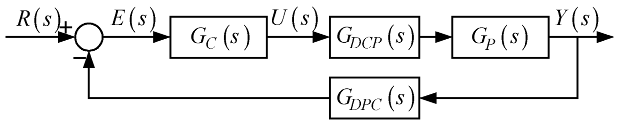

Fig. 2. Point-to-point networked control system formulation.

feedback data for a PI control response. The control signal from the PI controller is then sent back as a packet to each action agent via the IP network as well.

3) Action Agent: Each action agent contains an action agent controller and an action agent plant. The action agent controller is a simple hardware unit to periodically convert the control signal in a packet from the control agent to an actual signal to drive the action agent plant. The sensory output of the action agent plant such as motor speed is also monitored and sent back to the control agent.

B. Mathematical Formulation

In order to analyze how to schedule the PI controller gains on the control agent with respect to an IP network traffic condition, let us consider an action-agent–control-agent pair and 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 Laplace domain, respectively. The other action agents and control agents can be controlled using a similar formulation.

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

$$G_C(s) = \frac{K_P\left(s + (K_I/K_P)\right)}{s} = \frac{K_P(s + z_C)}{s} \tag{1}$$

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 action agent plant $G_P(s)$, and for sending the system output $Y(s)$ to the PI controller $G_C(s)$, are represented by $G_{DCP}(s)$ and $G_{DPC}(s)$, respectively, of which the analytical forms are

$$G_{DCP}(s) = e^{-\tau_{DCP}s} \tag{2}$$
$$G_{DPC}(s) = e^{-\tau_{DPC}s} \tag{3}$$

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

$$\frac{Y(s)}{R(s)} = \frac{G_C(s)G_{DCP}(s)G_P(s)}{1 + G_C(s)G_{DCP}(s)G_P(s)G_{DPC}(s)}. \tag{4}$$

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 [18]:

$$e^{-\tau s} \cong \left(1 + (\tau s/n)\right)^{-n} \tag{5}$$

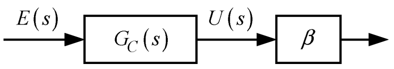

Fig. 3. Adaptation of PI controller gains at the controller output by β.

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 [18]. Moreover, (5) is suitable for real-time applications due to its computational simplicity.

C. Case Study

A dc motor speed control problem is used as a case study to demonstrate the proposed approach throughout this paper. The dc motor transfer function from [19] is used to represent the action agent plant dynamics, and is described by

$$G_P(s) = \frac{2029.826}{(s + 26.29)(s + 2.296)}. \tag{6}$$

Let us assume that the practical dc motor PI speed controller 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:

1) *Percentage overshoot (P.O.)*: $P.O. \leq 5\%$;
2) *Settling time* (t_s): $t_s \leq 0.309$ s;
3) *Rise time* (t_r): $t_r \leq 0.117$ s.

Using the root locus design approach without considering network delays, a feasible choice of $(K_P, K_I) = (K_P^0, K_I^0) \triangleq (0.1701, 0.378)$, and these gains satisfy all the design specifications as listed.

III. Parameterization for Gain Scheduling: Constant Network Delay

A. PI Controller Parameterization

With network delays in the control loop, the initial (K_P^0, K_I^0) may no longer satisfy the specifications. The system performance will degrade, and the system may become unstable. To maintain the best possible system performance with network delays, the controller gains need to be adapted with respect to the current network condition. In this section, we introduce the β gain to externally adapt (K_P^0, K_I^0) without completely redesigning the existing controller. The idea of β gain adaptation is adopted from [20]. 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:

$$\beta G_C(s) = \beta \frac{K_P^0(s + z_C)}{s}. \tag{7}$$

This parameterization enables PI gain scheduling to be tractable for real-time on-line analysis with existing theories such as root locus so that the control agent could quickly analyze the system to perform additional advanced control schemes such as fault detection and diagnosis. Adjusting K_P^0 and K_I^0 separately with no concern for z_C could maintain equivalent or better system performance than adjusting β. However, this approach usually 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 on-line analysis. In addition, by searching for β instead, the feasible region is only one dimensional, which makes it easier to search for the optimal value.

B. 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:

$$J = w_1 J_1 + w_2 J_2 + w_3 J_3, \tag{8}$$

$$J_1 = \begin{cases} (MSE - MSE_0)^2, & MSE > MSE_0 \\ 0, & MSE \leq MSE_0 \end{cases} \tag{9}$$

$$J_2 = \begin{cases} (P.O. - P.O._0)^2, & P.O. > P.O._0 \\ 0, & P.O. \leq P.O._0 \end{cases} \tag{10}$$

$$J_3 = \begin{cases} (t_r - t_{r0})^2, & t_r > t_{r0} \\ 0, & t_r \leq t_{r0} \end{cases} \tag{11}$$

where

$$MSE = \frac{1}{N} \sum_{k=0}^{N} e^2(k) \tag{12}$$

is the mean-squared error, MSE_0 is the nominal mean-squared error, $P.O._0$ 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 $P.O._0 = 5\%$, $t_{r0} = 0.117$, whereas MSE_0 has to be determined from a simulation or an experiment. In this paper, we use $MSE_0 = 0.00595$. Therefore, when $\beta = 1$ without network delays in the system, $J = 0$.

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 (5) with the following approximated characteristic equation:

$$\begin{aligned} 1 &+ \beta G_C(s) G_{DCP}(s) G_P(s) G_{DPC}(s) \\ &= e^{-\tau s} \frac{2029.826 \beta K_P^0 (s + z_C)}{s(s + 26.29)(s + 2.296)} \\ &\cong \frac{2029.826 \beta K_P^0 n^n (s + z_C)}{\tau^n s (s + 26.29)(s + 2.296)\left(s + \left(\frac{n}{\tau}\right)\right)^n} \end{aligned} \tag{13}$$

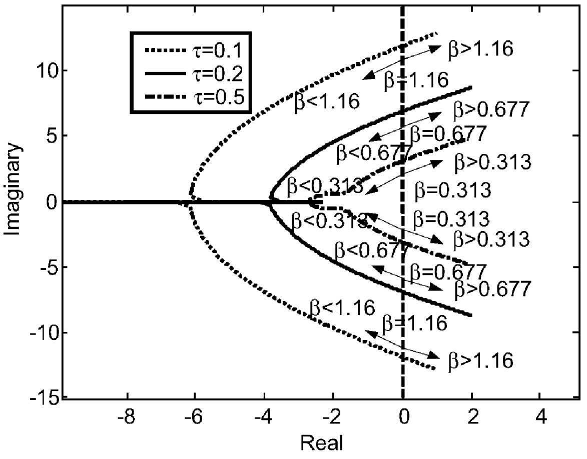

Fig. 4. Primary branches of the root locus of the networked dc motor PI speed control system using the approximation in (5) to approximate delays.

where τ is defined as $\tau = \tau_{CP} + \tau_{PC}$. For example, setting $n = 4$, the root locus of (13) 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 [18], the primary branches are sufficient to be used for stability region approximation.

In a stable system, β can range from 0 to the value at 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 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. Various optimization methods could be used for β. In this paper, for example, we optimize J by a lookup table approach using (2) and (3) as delays for $U(s)$ and $Y(s)$, respectively, in simulations with $w_1 = 1.64902$, $w_2 = 0.00833$, $w_3 = 0.01395$, $\tau = 0.1, 0.2, 0.6$ s, 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.

IV. Parameterization for Gain Scheduling: Actual IP Network Delay

A. IP Network Delay Characteristics

Actual IP network delays are not constant, but stochastic in nature. In addition, because packets are sent in discrete time according to the applications and protocols used, the network delays are not necessarily continuous. To illustrate actual IP network delay characteristics, roundtrip time (RTT) delays are measured from an Ethernet network in the Advanced Diagnosis And Control (ADAC) Laboratory, North Carolina State University (NCSU), to the destinations listed in Table I for 24 h

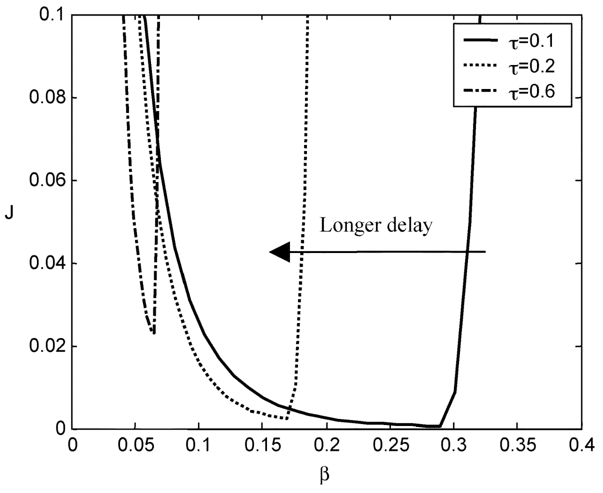

Fig. 5. Cost J from optimization using (2) and (3) to delay $U(s)$ and $Y(s)$, respectively, with $w_1 = 1.64902$, $w_2 = 0.00833$, $w_3 = 0.01395$, $t_f = 10$ s, $\tau = 0.1, 0.2, 0.6$ s.

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

Destination host	$\tau_{\min}$ (s)	τ_{median} (s)	τ_{mean} (s)	$\tau_{\max}$ (s)
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

(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.

B. 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 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:

$$P[\tau] = \begin{cases} \frac{1}{\phi} e^{-\frac{(\tau-\eta)}{\phi}}, & \tau \geq \eta \\ 0, & \tau < \eta \end{cases} \tag{14}$$

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 [21]. An important concern is how to select η in real time. Both η

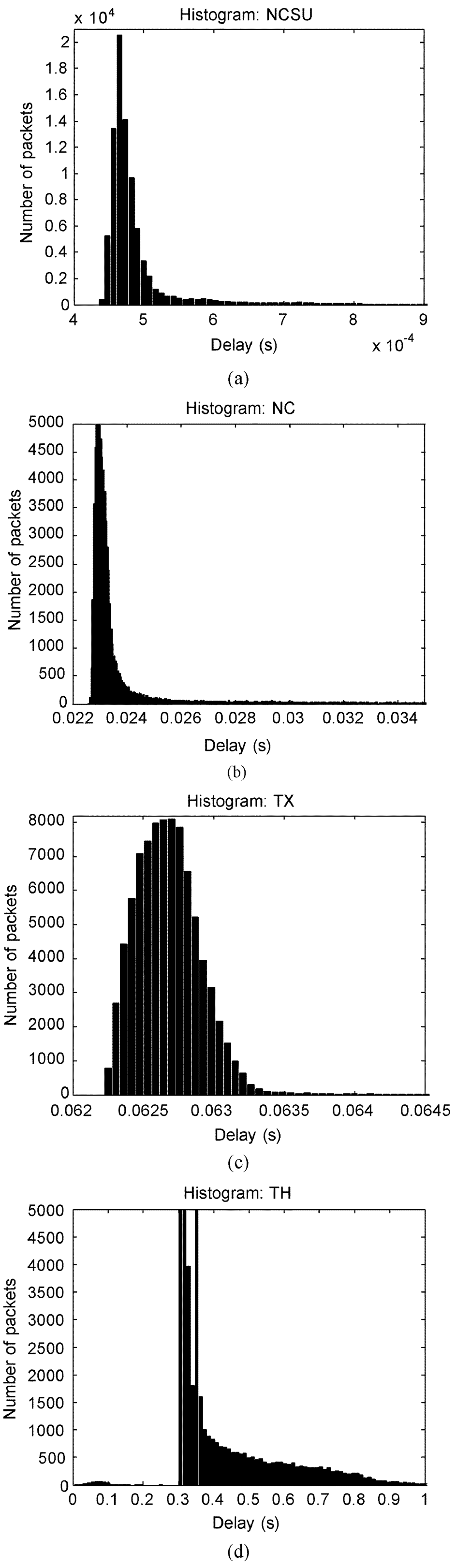

Fig. 6. Histograms of RTT delays measured from ADAC Laboratory at NCSU to: (a) www.lib.ncsu.edu; (b) www.visitnc.com; (c) www.utexas.edu; (d) www.ku.ac.th.

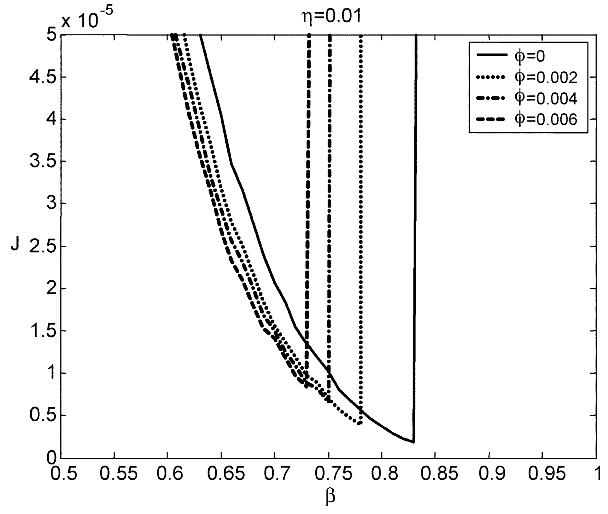

Fig. 7. Typical effect of various ϕ on the optimal β selecting while holding η constant at $\eta = 0.01$.

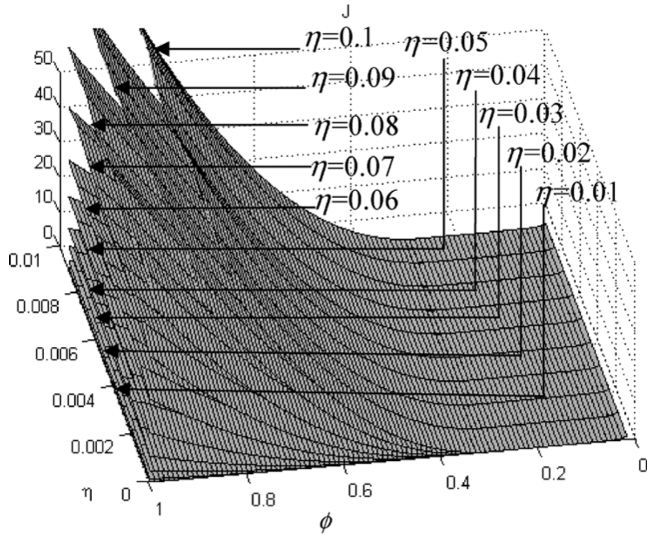

Fig. 8. Typical cost surfaces of J with respect to various η and ϕ without constraint.

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 (14), $P[\tau = \eta]$ 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 (14) and the RTT delay statistics, we also assume that RTT delays have $\mu > \eta$.

C. Optimizing β With Actual IP Network Delay Concern

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 (14) 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 $\phi = 0, 0.002, 0.004, 0.006$, while holding $\eta = 0.01$. In addition, the surfaces of J with respect to various

TABLE II
OPTIMAL β WITH RESPECT TO $\eta = 0.01, 0.02, 0.03, 0.04$, AND $\phi = 0, 0.001, 0.002, \ldots, 0.005$ OBTAINED WITH $w_1 = 1.64902$, $w_2 = 0.00833$, $w_3 = 0.01395$, WITHOUT CONSTRAINT

η \ ϕ	**0**	**0.001**	**0.002**	**0.003**	**0.004**	**0.005**
0.01	0.83	0.68	0.58	0.51	0.45	0.40
0.02	0.79	0.65	0.56	0.49	0.44	0.39
0.03	0.78	0.65	0.55	0.48	0.43	0.39
0.04	0.76	0.64	0.55	0.48	0.43	0.39

η and ϕ obtained from the same condition are shown in Fig. 8. Some of the optimal β found from J are shown in Table II. As shown in Figs. 7 and 8 and Table II, 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.

V. GAIN SCHEDULING ALGORITHM

A possible solution to handle the variable traffic condition on actual IP networks is to adjust (K_P^0, K_I^0) by scheduling β with respect to the current traffic condition characterized by η and ϕ. We assume that the gain scheduling approach is implemented as a hardware or software part of the control agent called the β scheduler middleware, which is physically (in hardware) or virtually (in software) attached in front of the original PI controller, respectively. The gain scheduling procedure is depicted in Fig. 9 with steps 1, 2, 3, 4, and 5, described as follows.

Step 1) The β gain scheduler middleware initializes the packet index defined as $i = 0, \ldots, N$, to $i = 0$, the summation of RTT delay 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.

Step 2) To send $u(t)$ out to an action agent according to the original controller operation, the β scheduler middleware 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

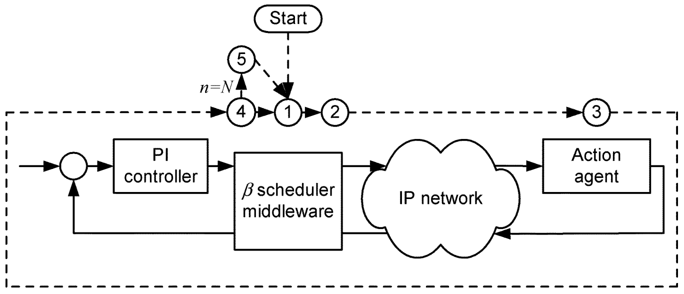

Fig. 9. β gain scheduler middleware operation.

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 β scheduler middleware will increase the packet index by $i = i + 1$.

Step 3) The action agent will return the output $y(t)$, i, and $t_S(i)$, as a packet back to the β scheduler middleware 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 action agent has the same queueing mechanism to handle outgoing packets.

Step 4) When the β scheduler middleware receives a packet containing $y(t, i)$ from the action agent at time t during a sampling period, the β scheduler middleware will compute

$$rtt(i) = t_A(i) - t_S(i) \tag{15}$$

$$m = m + rtt(i) \tag{16}$$

$$n = n + 1 \tag{17}$$

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 β scheduler middleware 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 β scheduler middleware and an action agent 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 PI controller and the action agent will opt to use the most updated data to compute $u(t, i)$ and $y(t, i)$, respectively. Out-of-order packets will be ignored. 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.

Step 5) Once $n = N$, the β scheduler middleware will calculate

$$\mu = \frac{m}{N} \tag{18}$$

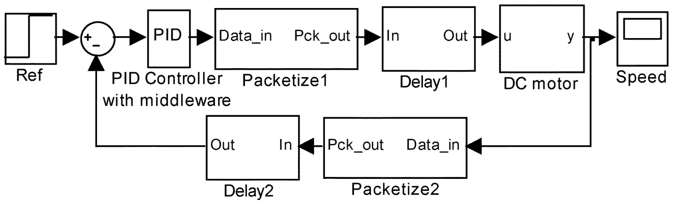

Fig. 10. β gain scheduler middleware simulation.

$$\eta = \begin{cases} \left(RTT\left[\left(\frac{N}{2}\right)\right] + RTT\left[\left(\frac{N}{2}\right)+1\right]\right)/2, & N \text{ is even} \\ RTT\left[\left\lceil\frac{N}{2}\right\rceil\right], & N \text{ is odd} \end{cases} \tag{19}$$

$$\phi = \mu - \eta \tag{20}$$

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 (14). The β scheduler middleware then updates β by picking the optimal β from the table with respect to η and ϕ. Steps 1–5 will be repeated for the next iteration.

VI. Simulation Result

The performance of the proposed β scheduler middleware approach on the networked dc motor PI speed control system is verified by simulations implemented on Matlab/Simulink 6.1 as shown in Fig. 10. The following environment is used to illustrate the effectiveness of the proposed β scheduler middleware 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 β scheduler middleware, and the action agent controller: $T = 1$ ms.
- The number of packets to evaluate the characteristic of RTT delay: $N = 100$.

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

1) The dc motor is controlled over IP networks by the PI controller with the nominal gains (K_P^0, K_I^0).
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 β gain scheduler middleware using η and ϕ from real-time measurements as described in the previous section. In this paper, we assume that the delay from the β scheduler middleware to the dc motor (τ_{CP}) and the delay from the dc motor to the β scheduler middleware (τ_{PC}) have similar characteristics. Therefore, we prepare these delays for the simulations by dividing

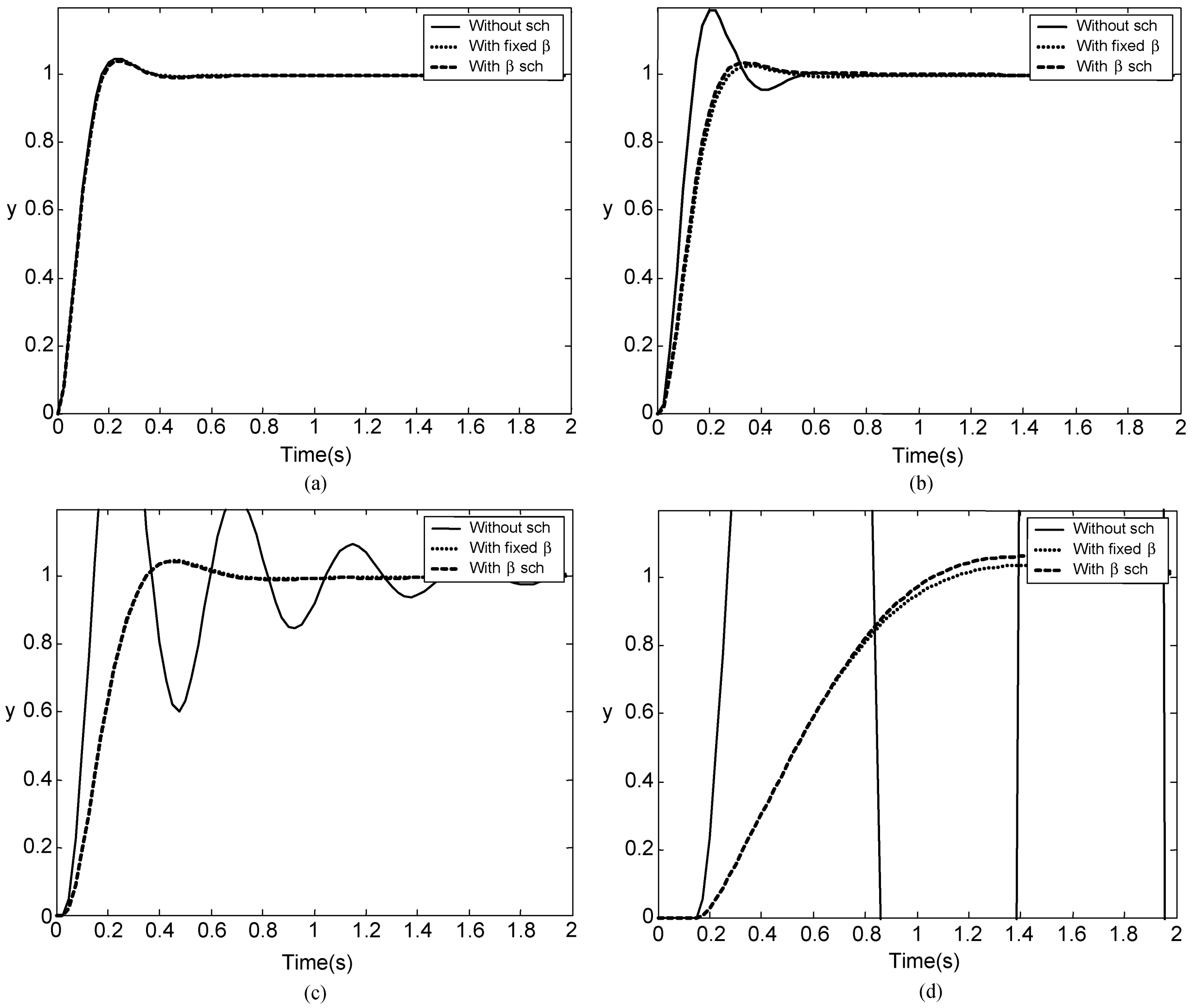

Fig. 11. Step responses from networked dc motor PI speed control simulations using RTT delays from ADAC Laboratory at NCSU to: (a) www.lib.ncsu.edu; (b) www.visitnc.com; (c) www.utexas.edu; (d) www.ku.ac.th.

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 III, whereas the step responses from the simulations are illustrated in Fig. 11.

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 III and Fig. 11. The networked PI control system using fixed β and β scheduler middleware could maintain the system performance much better. As shown in Table III and Fig. 11(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 III and Fig. 11(b) and (c). Nevertheless, the fixed β and β scheduler middleware schemes can still maintain the system performance satisfactorily. The performance is much better than the system with the nominal (K_P^0, K_I^0) since the PI controller gains in this case are adapted to be more suitable for the network traffic conditions. However, as shown in Table III and Fig. 11(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.

TABLE III
COSTS J FROM NETWORKED DC MOTOR PI SPEED CONTROL SIMULATIONS USING RTT DELAYS FROM ADAC LABORATORY AT NCSU TO www.lib.ncsu.edu, www.visitnc.com, www.utexas.edu, AND www.ku.ac.th

Control scheme / Destination host	Nominal gains	Fixed β	β gain scheduling
www.lib.ncsu.edu	0	0	0
www.visitnc.com	1.7634	4.9965e-005	3.6111e-005
www.utexas.edu	28.0574	2.0517e-004	2.0147e-004
www.ku.ac.th	7.1413e+033	0.0062	0.0208

In addition, the PI controllers using the pre-computed optimal β and β gain scheduler middleware 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 Laboratory 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 β gain scheduling middleware to adjust the PI controller gains could provide an acceptable and more flexible control in real IP environment.

VII. Conclusion

A gain scheduling approach with respect to the current network traffic condition is introduced in this paper to enhance widely used and existing PI controllers for using over IP networks. Simulation results have shown the promising chances to apply the proposed gain scheduling scheme for a networked PI controller on actual IP network environment with reasonably long RTT delays and relatively low variations. There are still several issues to be investigated in order to improve and strengthen the gain scheduling approach such as packet loss effects and the performance of the gain scheduling on actual networked control systems over real IP networks under various IP quality-of-service (QoS) protocols. These additional studies could support the gain scheduling approach for practical uses in the future.

References

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

[2] G. Kaplan, “Ethernet’s winning ways,” *IEEE Spectrum*, vol. 38, pp. 113–115, Jan. 2001.

[3] F.-L. Lian, J. R. Moyne, and D. M. Tilbury, “Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet,” *IEEE Control Syst. Mag.*, vol. 21, pp. 66–83, Feb. 2001.

[4] Y. Halevi and A. Ray, “Integrated communication and control systems: Part I—Analysis,” *J. Dynamic Syst., Meas., Contr.*, vol. 110, pp. 367–373, 1988.

[5] J. Nilsson, “Real-time control systems with delays,” Ph.D. dissertation, Lund Inst. Technol., Lund, Sweden, 1998.

[6] 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.

[7] F. Goktas, “Distributed control of systems over communication networks,” Ph.D. dissertation, Univ. Pennsylvania, Philadelphia, PA, 2000.

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

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

[10] M. L. Sichitiu, “Control of data networks: Models, stability and controllers,” Ph.D. dissertation, Univ. Notre Dame, Notre Dame, IN, 2001.

[11] T.-J. Tarn and N. Xi, “Planning and control of Internet-based teleoperation,” in *Proc. SPIE: Telemanipulator and Telepresence Technologies V*, vol. 3524, Boston, MA, 1998, pp. 189–193.

[12] 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.

[13] Y. H. Kim, H. S. Park, and W. H. Kwon, “Stability and a scheduling method for network-based control systems,” in *IEEE Int. Conf. Industrial Electronics, Control, and Instrumentation (IECON’96)*, vol. 2, Taipei, Taiwan, 1996, pp. 934–939.

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

[15] N. Chopra, M. W. Spong, S. Hirche, and M. Buss, “Bilateral teleoperation over the Internet: the time varying delay problem,” in *IEEE Automatic Control Conf. (ACC’03)*, vol. 1, Denver, CO, 2003, pp. 155–160.

[16] K. C. Lee, S. Lee, and M. H. Lee, “Remote fuzzy logic control of networked control system via Profibus-DP,” *IEEE Trans. Industr. Electron.*, vol. 50, pp. 784–792, Aug. 2003.

[17] N. B. Almutairi, M.-Y. Chow, and Y. Tipsuwan, “Network-based controlled DC motor with fuzzy compensation,” in *IEEE Int. Conf. Industrial Electronics, Control, and Instrumentation (IECON’01)*, vol. 3, Denver, CO, 2001, pp. 1844–1849.

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

[19] Y. Tipsuwan and M.-Y. Chow, “Fuzzy logic microcontroller implementation for DC motor speed control,” in *IEEE Int. Conf. Industrial Electronics, Control, and Instrumentation (IECON’99)*, vol. 3, San Jose, CA, 1999, pp. 1271–1276.

[20] N. B. Almutairi and M.-Y. Chow, “A modified PI control action with a robust adaptive fuzzy controller applied to DC motor,” in *Proc. IEEE/INNS Int. Joint Conf. Neural Networks (IJCNN’01)*, vol. 1, Washington, DC, 2001, pp. 503–508.

[21] J. W. Park and J. M. Lee, “Transmission modeling and simulation for Internet-based control,” in *IEEE Int. Conf. Industrial Electronics, Control, and Instrumentation (IECON’01)*, vol. 1, Denver, CO, 2001, pp. 165–169.

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.

After receiving 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 receiving 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. 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 motors, process control, power systems, and communication systems. He has established the Advanced Diagnosis and Control (ADAC) Laboratory at North Carolina State University. He has authored one book, several book chapters, and over 90 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.