International Journal of Control, Automation, and Systems (2012) 10(5):1013-1022 ISSN:1598-6446 eISSN:2005-4092
DOI 10.1007/s12555-012-0519-x http://www.springer.com/12555

Markov-Chain-Based Output Feedback Control for Stabilization of Networked Control Systems with Random Time Delays and Packet Losses

Jiawei Dong and Won-jong Kim*

Abstract: This paper proposes an output feedback method to stabilize and control networked control systems (NCSs). Random time delays and packet losses are treated separately when an NCS is modeled. The random time delays in the controller-to-actuator and sensor-to-controller links are modeled with two time-homogeneous Markov chains, while the packet losses are treated by the Dirac delta functions. An asymptotic mean-square stability criterion is established to compensate for the network-induced random time delays and packet losses in both the controller-to-actuator and sensor-to-controller links simultaneously. An algorithm to implement the asymptotic mean-square stability criterion is also proposed. Further, a DC-motor speed-control test bed with Ethernet using User Datagram Protocol (UDP) is constructed and employed for experimental verification. Two sets of experiments, with and without 10% packet losses in the links, are conducted on this NCS. Experimental results illustrate the effectiveness of the proposed output feedback method compared to conventional controllers. This method could compensate for the effects of the random time delays and packet losses and guarantee the system performance and stability. The integral time and absolute error (ITAE) of the experiments without packet losses is reduced by 13% with the proposed method, and the ITAE of experiments with 10% packet losses, by 30%. The NCS can track the reference command faithfully with the proposed method when random time delays and packet losses exist in the links, whereas the NCS fails to track the reference command with the conventional control algorithms.

Keywords: Asymptotic mean-square stability, Markov chains, networked control systems, packet losses, random time delays.

1. INTRODUCTION

NCSs are a type of distributed control systems where the reference inputs, the plant outputs, and the control inputs are exchanged over a communication network. The study of NCSs has been an active and attractive research area in the past several years due to its broad applications, such as mobile sensor networks [1], remote surgery [2], haptics collaboration over the Internet [3], [4], automated highway systems [5], and unmanned aerial vehicles [6].

The introduction of a communication network into a control system has brought many advantages, such as no additional dedicated wiring, reduced weight and space requirement, ease of system diagnosis and maintenance, and increased system agility, etc. On the other hand, the communication network inevitably presents more constraints such as random time delays and packet losses that make the analysis and design of NCSs challenging.

Manuscript received November 22, 2011; revised June 14, 2012; accepted July 5, 2012. Recommended by Editorial Board member Izumi Masubuchi under the direction of Editor Yoshito Ohta.

This work was supported in part by the TAMU Program to Enhance Scholarly and Creative Activities under Grant No. 2010-SAC-8779.

Jiawei Dong and Won-jong Kim are with the Department of Mechanical Engineering, Texas A&M University, College Station, TX, U.S.A. (e-mails: jwdong@neo.tamu.edu, wjkim@tamu.edu).
* Corresponding author.

These random time delays and packet losses can degrade the system performance or even destabilize the system. How to compensate for the random time delays and packet losses has become one of the active research areas of NCSs.

Random time delays can be divided into three major categories, time delays shorter than one sampling period, time delays longer than one sampling period but finite, and infinite delays which can also be considered as packet losses. The analysis and modeling of random time delays can be performed with a deterministic model or a stochastic model. In [7], the random time delays were assumed to be deterministic, and the controller gain was constant when a hybrid system was analyzed. In [8], a delay-distribution-dependent criterion for the mean-square stability of the NCS was derived by using the Lyapunov-Krasovskii functional approach and linear matrix inequality (LMI) technique. In [9], the random time delays were modeled with Markov chains, and the analysis mainly focused on the delays shorter than one sampling period. The control law was derived by setting up the cost function of linear-quadratic regulation (LQR) and linear-quadratic Gaussian (LQG) problems. In [10], the NCSs with unreliable data communication were studied. An observer-based controller was designed to exponentially stabilize the NCSs in the sense of mean square and also achieved the prescribed H_∞ disturbance attenuation level. An estimation method was introduced to compensate for the lost data of the NCSs in [11]. The

controller design was considered for the states available and the states unavailable, respectively.

In the discrete-time domain, [12-16] modeled the NCSs as jump linear systems and the time delays with Markov chains. Xiao *et al*. proposed two types of controller-design methods for NCSs with time delays modeled with Markov chains [13]. Zhang *et al*. proposed an output feedback method to analyze the time delays of NCSs and assumed the random time delays could only take integer values [14]. Ye *et al*. also modeled the time delays and packet losses in the NCS with Markov chains. Without the augmented state method, however, the computation effort was reduced [15]. In [16], Xiong *et al*. proposed two types of packet-loss models, the arbitrary model and the Markov-chain model. The stability conditions of NCS with packet losses were given based on a Lyapunov approach. Liu *et al*. proposed a time-delay-compensation technique using the modified model-predictive control method. The data-packet losses were compensated for with the predicative packets generated from the same model [17]. In [18], Schenato proposed an optimal estimation design for the NCS with time delays and packet losses. The stability of these estimators did not depend on packet delays but only on the overall packet-loss probability.

In the cited references [13-16], the authors assumed the Markov-chain model could intuitively include the packet losses as well. However, the packet losses actually change the structure of the model. When a packet is lost, the sensor output or the control input will be unavailable in all sense, whereas for the time-delay case, the sensor output or the control input arrives at its destination node eventually with a certain amount of delays. Hence, the Markov-chain-based packet-loss model assumes that the packet-loss information can be included by the same probability transition from the time-delay perspective will not closely catch the nature of the NCS. In [13] and [14], the stability analysis only considered the integer time-delay states. However, in the practical world, the time delays are non-integer numbers. In this paper, our methodology is based on the approach presented in [14] but treats random time delays and packet losses with separate models, which reveals the nature of the NCSs in a closer manner. The proposed models for time delays and packet losses are based on stochastic processes in the discrete-time domain so that the proposed method can be implemented on a practical NCS without much modification. The proposed method considers both integer and non-integer time delays.

The rest of the paper is organized as follows. In Section 2, the NCS system modeling methodology is formulated to include time delays modeled with Markov chains, packet losses modeled with separate Dirac delta functions, and a closed-loop NCS asymptotic mean-square stability criterion. In Section 3, the control algorithm to be implemented on a physical NCS is proposed. In Section 4, key experiments are designed and performed to illustrate the effectiveness of the proposed output feedback method and stability criterion. In Section 5, conclusions are given.

2. SYSTEM MODELING

A typical NCS has a closed-loop structure as shown in Fig. 1. As indicated in the dashed boxes, Server represents the controller on one end of the communication network whereas Client represents the plant including sensors and actuators on the other end of the communication network.

Consider the NCS setup in Fig. 1, assume the whole NCS is a linear discrete-time system. The state-space model of the plant is

$$\mathbf{x}_p(k+1) = \mathbf{A}_p\mathbf{x}_p(k) + \mathbf{B}_p\tilde{\mathbf{u}}(k), \tag{1}$$

$$\mathbf{y}(k) = \mathbf{C}_p\mathbf{x}_p(k), \tag{2}$$

where $\mathbf{x}_p(k) \in R^n$, $\mathbf{u}(k) \in R^m$, and $\mathbf{y}(k) \in R^p$ are the state, control-input, and plant-output vectors, respectively. $\tilde{\mathbf{u}}(k) \in R^m$ and $\tilde{\mathbf{y}}(k) \in R^p$ are the delayed control-input and plant-output vectors. $\mathbf{A}_p$, $\mathbf{B}_p$, and $\mathbf{C}_p$ are the known matrices with appropriated dimensions.

Similarly, the controller has the model as

$$\mathbf{x}_c(k+1) = \mathbf{A}_c\mathbf{x}_c(k) + \mathbf{B}_c\mathbf{e}(k), \tag{3}$$

$$\mathbf{u}(k) = \mathbf{C}_c\mathbf{x}_c(k) + \mathbf{D}_c\mathbf{e}(k), \tag{4}$$

where $\mathbf{e}(k) = \mathbf{r}(k) - \tilde{\mathbf{y}}(k)$ is the error, and $\mathbf{r}(k) \in R^p$ is the reference command. $\mathbf{A}_c$, $\mathbf{B}_c$, $\mathbf{C}_c$, and $\mathbf{D}_c$ are to be determined to compensate for the random time delays and packet losses, which will be discussed in Section 2.3 with details that include the stability criterion and algorithm. An experimental example of how to solve the controller matrices will be given in Section 4.3.

2.1. Time delays modeled with Markov chains

In general, time delays can be categorized as deterministic delays and stochastic delays. Due to the stochastic nature of the communication network, a stochastic method is adopted in this paper to model the random time delays in the communication links since it can model the random process of the network condition more realistically compared to a deterministic method. As in [12-14], we assume the status of time delays mainly depends on the previous status such that the random time delays τ^{ca} and τ^{sc} can be modeled with finite-state time-homogeneous Markov chains. For more details of the effectiveness of time delays modeled with Markov chains, refer to [13,14].

In Fig. 1, τ^{ca} and τ^{sc} represent the random time delays in the controller-to-actuator link and the sensor-to-

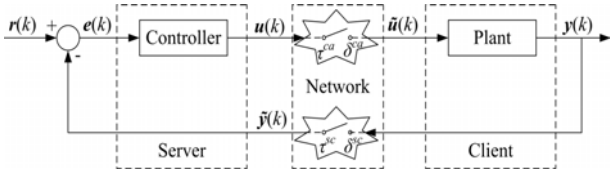


Fig. 1. NCS block-diagram. τ^{ca} and τ^{sc} represent the random time delays, and δ^{ca} and δ^{sc} represent the packet losses in the controller-to-actuator and the sensor-to-controller links, respectively.

controller link, respectively. In this paper, τ^{ca} and τ^{sc} are modeled with two time-homogeneous Markov chains with finite Markov states and take values in the set $\mathbb{P} = \{\tau_i^{ca}; i \in \mathbb{I} = 1, \cdots, p\}$ and $\mathbb{Q} = \{\tau_m^{sc}; m \in \mathbb{M} = 1, \cdots, q\}$, respectively. Their transition-probability matrices are $\mathbf{\Lambda} = \{\lambda_{ij}\}$ and $\mathbf{\Gamma} = \{\mu_{mn}\}$, respectively. These transition-probability matrices represent the probabilities that τ^{ca} and τ^{sc} jump from the state i to j and the state m to n, respectively. The definitions of λ_{ij} and μ_{mn} are

$$\lambda_{ij} = \Pr(\tau^{ca}(k+1) = \tau_j^{ca} \mid \tau^{ca}(k) = \tau_i^{ca}), \tag{5}$$

$$\mu_{mn} = \Pr(\tau^{sc}(k+1) = \tau_n^{sc} \mid \tau^{sc}(k) = \tau_m^{sc}), \tag{6}$$

where $\lambda_{ij} \geq 0$, $\mu_{mn} \geq 0$, and $\sum_{j=1}^{p} \lambda_{ij} = 1$, $\sum_{n=1}^{q} \mu_{mn} = 1$, for all $i, j \in \mathbb{I}$ and $m, n \in \mathbb{M}$.

2.2. Packet-loss modeling

The packet loss in NCSs is another challenge induced by the communication network. Packet losses could take place when the network is congested, or the queues of routers and servers are overflown. NCSs do not monitor the network conditions, so explicit packet-loss information is unavailable to either Server or Client in the sense of real time.

A simplest stochastic model treats packet losses as a Bernoulli process [5]. It can also be modeled with a Markov chain [19] or a Poisson process [20]. Normally, packet losses share no common probabilistic characteristics with the random time delays since their causes are usually different and not always coupled. In general, for the case of packet losses, the system will require extensive control input to guarantee the stability and system performance. Whenever a packet is lost, time-delay information is irrelevant and unavailable. Therefore, assuming that a packet loss can be intuitively modeled with a Markov chain together with time delays cannot represent their independence in a communication network. In this paper, a separate packet-loss model is introduced.

As illustrated in Fig. 1, the network backbone can be treated as a jump system. In this case, when a packet is lost, the current output packet or the control input packet will be unavailable to either the servers or the clients, so that the output or control input from previous sampling period will be held for the current period. The time-delay information from previous period will also be inherited.

In Fig. 1, the Dirac delta functions, δ^{ca} and δ^{sc} represent the packet losses in the controller-to-actuator and the sensor-to-controller links, respectively. The notation of the packet losses follows [5] as below.

$$\delta^{ca}(k) = \begin{cases} 1 & \text{if no packet is lost} \\ 0 & \text{if a packet is lost} \end{cases} \tag{7}$$

$$\delta^{sc}(k) = \begin{cases} 1 & \text{if no packet is lost} \\ 0 & \text{if a packet is lost} \end{cases} \tag{8}$$

Unlike [5], however, we do not assign the Bernoulli probabilities to δ^{ca} and δ^{sc}. Packet losses can be stochastic so that a pre-assigned fixed probability like $\Pr(\delta^{ca}(k) = 1)$ would not represent the nature of the packet losses realistically. That is, if either δ^{ca} or δ^{sc} takes the value 0, there is a packet lost in the corresponding links. Otherwise, only random time delays exist in the links.

2.3. Controller design

As in Fig. 1, consider the time delays τ^{ca} and τ^{sc}. We introduce the ceiling function

$$f(\tau) = \left\lceil \frac{\tau + \tau_0}{h} \right\rceil, \tag{9}$$

where τ_0 is the time threshold, and h is the sampling period. The time threshold τ_0 includes the sum of the data sampling time, data-packet generating time, packet-processing time, queuing time, etc. In each sampling period, these times may not be exactly the same, but can be quite deterministic. Therefore, an upper bound τ_0 can be set as the time threshold. Then the output data packet arriving at Server is $\tilde{\mathbf{y}}(k) = \mathbf{y}(k - f(\tau^{sc}(k)))$. This can also be applied to the control input, such that $\tilde{\mathbf{u}}(k) = \mathbf{u}(k - f(\tau^{ca}(k)))$. Note that for τ^{ca} and τ^{sc}, the threshold τ_0 may take different values.

Fig. 2 illustrates an example timing diagram of packet exchanges between Server and Client. In Fig. 2, the lines with an arrowhead represent the transmission of the sensor packets, and the lines with a circle, the control packets. The horizontal length of each line indicates the random time delays of each packet in the links. Several possible scenarios are shown in Fig. 2. The first case is that both the τ^{ca} and τ^{sc} are shorter than h. The second case is that both the τ^{ca} and τ^{sc} are longer than h. The last case in Fig. 2 is that τ^{sc} is shorter than h, and τ^{ca} is longer than h. For instance, if $\tau^{sc} + \tau_0 < h$, then $f(\tau^{sc}) = 0$. Thus, when the output packet arrives at Server, it is indicated as $\tilde{\mathbf{y}}(k)$ in the k-th sampling period. Likewise, if $h < \tau^{sc} + \tau_0 < 2h$, then $f(\tau^{sc}) = 1$. Thus the output packet arrives at Server will be $\tilde{\mathbf{y}}(k-1)$ in the k-th sampling period.

Now consider the NCS in Fig. 1 with random time delays and packet losses. The outputs to the controller $\tilde{\mathbf{y}}(k)$ and the control inputs to the plant $\tilde{\mathbf{u}}(k)$ are

$$\begin{aligned} \tilde{\mathbf{y}}(k) = {} & \delta^{sc}(k)\mathbf{y}(k - f(\tau^{sc}(k))) \\ & + \bar{\delta}^{sc}(k)\mathbf{y}(k - 1 - f(\tau^{sc}(k-1))), \end{aligned} \tag{10}$$

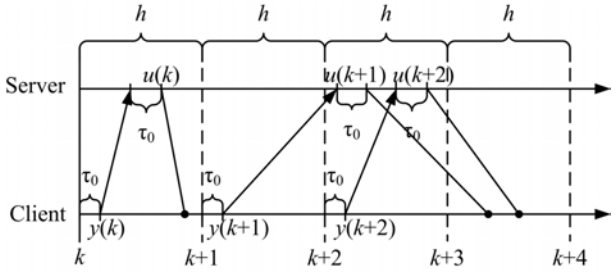


Fig. 2. Example of timing diagram of NCS communication.

$$\tilde{\mathbf{u}}(k)=\delta^{ca}(k)\mathbf{u}(k-f(\tau^{ca}(k))) +\bar{\delta}^{ca}(k)\mathbf{u}(k-1-f(\tau^{ca}(k-1))), \tag{11}$$

where $\bar{\delta}^{sc}(k)=1-\delta^{sc}(k)$ and $\bar{\delta}^{ca}(k)=1-\delta^{ca}(k)$. By (10) and (11), if packet losses take place in the links, previous data packets will be used. This provision can compensate for one packet loss. For consecutive packet losses, a more robust controller or predictor will be necessary such as autoregressive (AR) model or local cascade controller, etc. In this paper, we will focus on the case of one packet loss. The case of the consecutive packet losses is the further research effort of the authors.

Augment the plant's states as follows with all the possible Markov states of τ^{sc}

$$\bar{\mathbf{x}}_p(k)= \left[\mathbf{x}_p^T(k) \quad \mathbf{y}^T(k-1) \quad \mathbf{y}^T(k-2) \quad \cdots \quad \mathbf{y}^T(k-q-1)\right]^T.$$

Then the plant's model can be written as

$$\bar{\mathbf{x}}_p(k+1)=\bar{\mathbf{A}}_p\bar{\mathbf{x}}_p(k)+\bar{\mathbf{B}}_p\tilde{\mathbf{u}}(k), \tag{12}$$

$$\tilde{\mathbf{y}}(k)=\bar{\mathbf{C}}_p\bar{\mathbf{x}}_p(k), \tag{13}$$

where

$$\bar{\mathbf{A}}_p=\begin{bmatrix}\mathbf{A}_p & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0}\\ \mathbf{C}_p & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \ddots & \cdots & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{I} & \mathbf{0}\end{bmatrix},\quad \bar{\mathbf{B}}_p=\begin{bmatrix}\mathbf{B}_p\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{bmatrix},$$

$$\bar{\mathbf{C}}_p=\begin{bmatrix}\mathbf{0} & \cdots & \delta^{sc}(k)\mathbf{1} & \bar{\delta}^{sc}(k)\mathbf{1} & \cdots & \mathbf{0}\end{bmatrix},$$

and $\mathbf{1}$ is the unit matrix with all elements equal to 1, and the $f(\tau^{sc}(k))$-th entry equals to $\delta^{sc}(k)\mathbf{1}$.

Similarly, the augmented controller state vector is as follow with all the possible Markov states of τ^{ca}

$$\bar{\mathbf{x}}_c(k)= \left[\mathbf{x}_c^T(k) \quad \mathbf{u}^T(k-1) \quad \mathbf{u}^T(k-2) \quad \cdots \quad \mathbf{u}^T(k-p-1)\right]^T,$$

and the corresponding controller model is

$$\bar{\mathbf{x}}_c(k+1)=\bar{\mathbf{A}}_c\bar{\mathbf{x}}_c(k)-\bar{\mathbf{B}}_c\tilde{\mathbf{y}}(k)+\bar{\mathbf{B}}_c\mathbf{r}(k), \tag{14}$$

$$\tilde{\mathbf{u}}(k)=\bar{\mathbf{C}}_c\bar{\mathbf{x}}_c(k), \tag{15}$$

where

$$\bar{\mathbf{A}}_c=\begin{bmatrix}\mathbf{A}_c & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0}\\ \mathbf{C}_c & \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \ddots & \cdots & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{0} & \mathbf{0} & \cdots & \cdots & \mathbf{I} & \mathbf{0}\end{bmatrix},\quad \bar{\mathbf{B}}_c=\begin{bmatrix}\mathbf{B}_c\\ \mathbf{D}_c\\ \mathbf{0}\\ \mathbf{0}\\ \vdots\\ \mathbf{0}\end{bmatrix},$$

$$\bar{\mathbf{C}}_c=\begin{bmatrix}\mathbf{0} & \cdots & \delta^{ca}(k)\mathbf{1} & \bar{\delta}^{ca}(k)\mathbf{1} & \cdots & \mathbf{0}\end{bmatrix},$$

and the $f(\tau^{ca}(k))$-th entry equals to $\delta^{ca}(k)\mathbf{1}$.

Augmenting the new plant and controller model with $\bar{\mathbf{x}}=\begin{bmatrix}\bar{\mathbf{x}}_p^T & \bar{\mathbf{x}}_c^T\end{bmatrix}^T$, and the closed-loop dynamics will be

$$\bar{\mathbf{x}}(k+1)=(\mathbf{A}+\mathbf{BKC})\bar{\mathbf{x}}(k), \tag{16}$$

where $\mathbf{A}=\begin{bmatrix}\bar{\mathbf{A}}_p & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$, $\mathbf{B}=\begin{bmatrix}\mathbf{0} & \bar{\mathbf{B}}_p\\ \mathbf{I} & \mathbf{0}\end{bmatrix}$, $\mathbf{K}=\begin{bmatrix}\bar{\mathbf{A}}_c & -\bar{\mathbf{B}}_c\\ \bar{\mathbf{C}}_c & \mathbf{0}\end{bmatrix}$, and $\mathbf{C}=\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \bar{\mathbf{C}}_p & \mathbf{0}\end{bmatrix}$.

For the stability analysis, we adopt Definition 1 in [21] and Theorem 1 in [22].

Definition 1 [21]**:** Consider the jump linear system J_d

$$J_d:\begin{cases}\mathbf{x}(k+1)=\mathbf{A}(\eta(k))\mathbf{x}(k)+\mathbf{B}(\eta(k))u(k)\\ \mathbf{y}(k)=\mathbf{C}(\eta(k))\mathbf{x}(k)+\mathbf{D}(\eta(k))u(k),\end{cases}$$

where $\eta(k)$ is a discrete homogeneous Markov chain with states $S=(S_1,S_2,\cdots,S_N)$. The system J_d with $\mathbf{u}(k)=0$ is said to be asymptotic mean-square stable if

$$E\left\{\left\|\bar{\mathbf{x}}(k)\right\|^2\right\}\to 0 \quad \text{as} \quad k\to\infty,$$

for any initial condition $\mathbf{x}(0)=\mathbf{x}_0$ and initial distribution $\eta(0)=\eta_0$.

Theorem 1 [22]**:** Let there exists the nonnegative functional $V_i=V(i,x_{-h},\cdots,x_i)$, $i\in Z$ for which the conditions

$$E\{\Delta V_i\}\le -cE\{x_i^2\}, \quad i\in Z,$$

where $\Delta V_i=V_{i+1}-V_i$ and $c>0$ hold. Then the system J_d is asymptotic mean-square stable.

With Theorem 1, the sufficient and necessary conditions of the asymptotic mean-square stability of the closed-loop system (16) can be derived as follows.

Theorem 2: The closed-loop NCS in (16) is asymptotic mean-square stable if and only if there exists $\mathbf{P}(i,m)=\mathbf{P}^T(i,m)>0$ such that the following matrix inequality

$$\mathbf{H}(i,m)=(\mathbf{A}+\mathbf{BKC})^T\left(\sum_{j=1}^{p}\sum_{n=1}^{q}\lambda_{ij}\mu_{mn}\mathbf{P}(j,n)\right)(\mathbf{A}+\mathbf{BKC}) -\mathbf{P}(i,m)<0 \tag{17}$$

holds for all $i\in\mathbb{I}$ and $m\in\mathbb{M}$.

Proof: Sufficiency: For the closed-loop NCS in (16), construct the Lyapunov function as

$$V(\bar{\mathbf{x}}(k),k)=\bar{\mathbf{x}}^T(k)\mathbf{P}(\tau^{ca}(k),\tau^{sc}(k))\bar{\mathbf{x}}(k). \tag{18}$$

Then

$$\begin{aligned}\Delta V(\bar{\mathbf{x}}(k),k)&=V(\bar{\mathbf{x}}(k+1),k+1)-V(\bar{\mathbf{x}}(k),k)\\ &=\bar{\mathbf{x}}^T(k+1)\mathbf{P}(\tau^{ca}(k+1),\tau^{sc}(k+1))\bar{\mathbf{x}}(k+1)\\ &\quad-\bar{\mathbf{x}}^T(k)\mathbf{P}(\tau^{ca}(k),\tau^{sc}(k))\bar{\mathbf{x}}(k).\end{aligned} \tag{19}$$

Assume that τ^{ca} is at the Markov state i in the k-th

sampling period and will be at the Markov state j in the next sampling period. Similarly, τ^{sc} is at the Markov state m in the k-th sampling period and will be at the Markov state n in the next sampling period. For the simplicity, we denote $\mathbf{P}(\tau^{ca}(k), \tau^{sc}(k))$ as $\mathbf{P}(i, m)$ hereafter. Note that (i, m) is not the corresponding entry of matrix $\boldsymbol{P}$, but the corresponding Markov states of the time delays.

Then (19) can be reformulated as

$$\begin{aligned}&\Delta V(\overline{\mathbf{x}}(k), k)\\&= \overline{\mathbf{x}}^T(k+1)\mathbf{P}(j, n)\overline{\mathbf{x}}(k+1) - \overline{\mathbf{x}}^T(k)\mathbf{P}(i, m)\overline{\mathbf{x}}(k)\\&= \overline{\mathbf{x}}^T(k)[(\mathbf{A}+\mathbf{BKC})^T\mathbf{P}(j, n)(\mathbf{A}+\mathbf{BKC}) - \mathbf{P}(i, m)]\overline{\mathbf{x}}(k).\end{aligned} \tag{20}$$

The next time-delay state will depend on the current one, and the conditional expectation of (20) is as follows.

$$\begin{aligned}&E\{\Delta V(\overline{\mathbf{x}}(k), k)\}\\&= E\{\overline{\mathbf{x}}^T(k)[(\mathbf{A}+\mathbf{BKC})^T(\mathbf{P}(j, n) \mid \mathbf{P}(i, m))(\mathbf{A}+\mathbf{BKC})\\&\quad - \mathbf{P}(i, m)]\overline{\mathbf{x}}(k)\}\\&= E\{\overline{\mathbf{x}}^T(k)(\mathbf{A}+\mathbf{BKC})^T\left(\sum_{j=1}^{p}\sum_{n=1}^{q}\lambda_{ij}\mu_{mn}\mathbf{P}(j, n)\right)(\mathbf{A}+\mathbf{BKC})\\&\quad - \mathbf{P}(i, m)\overline{\mathbf{x}}(k)\}\\&= E\{\overline{\mathbf{x}}^T(k)\mathbf{H}(i, m)\overline{\mathbf{x}}(k)\}.\end{aligned} \tag{21}$$

If $\mathbf{H}(i, m) < 0$, then

$$\begin{aligned}E\{\Delta V(\overline{\mathbf{x}}(k), k)\} &= E\{\overline{\mathbf{x}}^T(k)\mathbf{H}(i, m)\overline{\mathbf{x}}(k)\}\\&\le E\{-\sigma_{\min}(i, m)\overline{\mathbf{x}}^T(k)\overline{\mathbf{x}}(k)\}\\&\le -\sigma E\{\|\overline{\mathbf{x}}(k)\|^2\},\end{aligned} \tag{22}$$

where $\sigma_{\min}(i, m) = \sigma_{\min}(-\mathbf{H}(i, m))$ is the minimum eigenvalue of $-\mathbf{H}(i, m)$ and $\sigma = \inf\{\sigma_{\min}(i, m),\ i \in \mathbb{I},\ m \in \mathbb{M}\} > 0$ is the infimum of these minimum eigenvalues.

According to Theorem 1, if $\mathbf{H}(i, m) < 0$, then the closed-loop system (16) is asymptotic mean-square stable.

Necessity: Under the assumption that (16) is mean-square stable and Theorem 1, with some $\alpha > 0$

$$\begin{aligned}E\{\Delta V(\overline{\mathbf{x}}(k), k)\} &= E\{\overline{\mathbf{x}}^T(k)\mathbf{H}(i, m)\overline{\mathbf{x}}(k)\}\\&\le -\alpha E\{\|\overline{\mathbf{x}}(k)\|^2\}\\&= E\{\overline{\mathbf{x}}^T(k)(-\alpha\mathbf{I})\overline{\mathbf{x}}(k)\}\end{aligned} \tag{23}$$

such that $E\{\overline{\mathbf{x}}^T(k)\mathbf{H}(i, m)\overline{\mathbf{x}}(k)\} - E\{\overline{\mathbf{x}}^T(k)(-\alpha\mathbf{I})\overline{\mathbf{x}}(k)\} = E\{\overline{\mathbf{x}}^T(k)[\mathbf{H}(i, m)+\alpha\mathbf{I}]\overline{\mathbf{x}}(k)\} \le 0$, and $\mathbf{H}(i, m)+\alpha\mathbf{I} \le 0$. Then $\beta + \alpha \le 0$, where $\beta = \sup\{\beta_{\max}(i, m) = \beta_{\max}(\mathbf{H}(i, m)), i \in \mathbb{I}, m \in \mathbb{M}\}$ is the supremum of the maximum eigenvalues of $\mathbf{H}(i, m)$. Since $\alpha > 0$, such that $\beta < 0$, and $\mathbf{H}(i, m) < 0$. Hence, the closed-loop system (16) is asymptotic mean-square stable, and $\mathbf{H}(i, m) < 0$.

This completes the proof.

The aforementioned asymptotic mean-square stability condition, (17) is nonlinear and difficult to be implemented in real time. A linear criterion will be introduced based on the LMIs.

Theorem 3: There exists a controller that has the form as in (3) and (4) such that the closed-loop system (16) is asymptotic mean-square stable if and only if there exists $\mathbf{P}(i, m) = \mathbf{P}^T(i, m) > 0$ satisfying

$$\begin{bmatrix}\mathbf{P}(i, m) & \mathbf{N}(i, m)\\ \mathbf{N}^T(i, m) & \mathbf{G}(j, n)\end{bmatrix} > 0 \tag{24}$$

with $\mathbf{N}(i, m) = \sum_{j=1}^{p}\sum_{n=1}^{q}\lambda_{ij}^{\frac{1}{2}}\mu_{mn}^{\frac{1}{2}}(\mathbf{A}+\mathbf{BKC})^T$.

Proof: The proof is obtained by Schur complement with $\mathbf{G}(j, n)\mathbf{P}(j, n) = \mathbf{I}$ and Theorem 2.

This completes the proof.

The conditions in Theorem 3 are in fact a set of LMIs with non-convex constraints. The on-line calculation of the coefficient matrices may require long computational time and induce more time delays to the data processing and control-law generation. Hence, the off-line calculation is adopted in this paper, and experiments are conducted for its effectiveness to the NCSs in the real-time sense. The control law with various levels of time delays and packet losses will be computed off-line and be tabulated for looking up during the implementation.

3. ALGORITHM IMPLEMENTATION

Practical NCSs normally have no clock synchronization mechanism over the entire communication network. Therefore, no explicit time-delay information is available to Server and Client in real time. Similarly, no explicit packet-loss information can be detected in real time either. All the information can be obtained by the next sampling period based on the assumption in the paper without consecutive packet losses. Due to the stochastic nature of the communication network, the packets containing the control inputs of each loop that arrives at Client may not be in the same sequence as they were initially sent by Server. All these possibilities make it challenging to implement the controller in the practical NCSs.

By setting up the time-stamp segment in the packets traveling through the communication network, the total time delays and packet losses can be detected by Client at the end of each sampling period. The total time delays can be inferred by calculating the difference between the time instance Client sends sensor packets to Server and the time instance Client receives control packets from Server. The structure of the total time delays has the following form.

$$\Delta timestamp(k) = \tau^{ca}(k) + \tau^{sc}(k) + \tau(k), \tag{25}$$

where $\tau(k)$ includes the packet-processing time, queuing time, other calculating time, etc., on both Server and Client. Note that, in general, τ^{ca} and τ^{sc} are not necessarily the same. Without a clock-synchronization mechanism, the exact τ^{ca} and τ^{sc} are unavailable, and we use

$$\tau^{ca}(k) \approx \tau^{sc}(k) \approx \frac{1}{2}\Delta timestamp(k) \tag{26}$$

in the controller design and implementation to be pre-

sented in Section 4.3. Compared to the time delays over the communication network, the packet-processing time, queuing time, or calculating time can be much smaller or neglected under certain circumstances. If Client receives no updated control signal within a certain time period, it may assume the packet has been lost.

Note that the time delays calculated by tracking the time stamps can only be accessed by the end of each sampling period, so the current time-delay information will only be able to be applied to the NCS by the next sampling period. Packet losses can be handled similarly. Server will use the time-delay information carried from the previous data packet to compensate for the effect of time delays and packet losses a sampling period later.

The implemented algorithm is illustrated in Fig. 3. This flow chart shows the single-server single-client case and can also be applied to single-server multiple-client and multiple-server multiple-client cases. In each sampling period, Server waits for the output packet arriving from Client. The details of the data-packet structures will be given in Section 4. All the control laws for various time-delay states and packet losses are calculated off-line and tabulated on Server. When the packet arrives, Server first checks whether a packet was lost in the previous sampling period by checking the corresponding data segment in the packet. If a packet was lost, Server uses the previous packet, chooses the control law with packet losses, and calculates the corresponding control input for the plant. If no packet was lost, Server checks the random time-delay states. The time-delay information is contained in the corresponding data segment. Then Server chooses the control law to calculate the control input based on the time-delay states. Then it sends the control-input packet back to Client to actuate the plant. If the newly updated control-input packet is lost in the link, Client will use previous control-input data to actuate the plant.

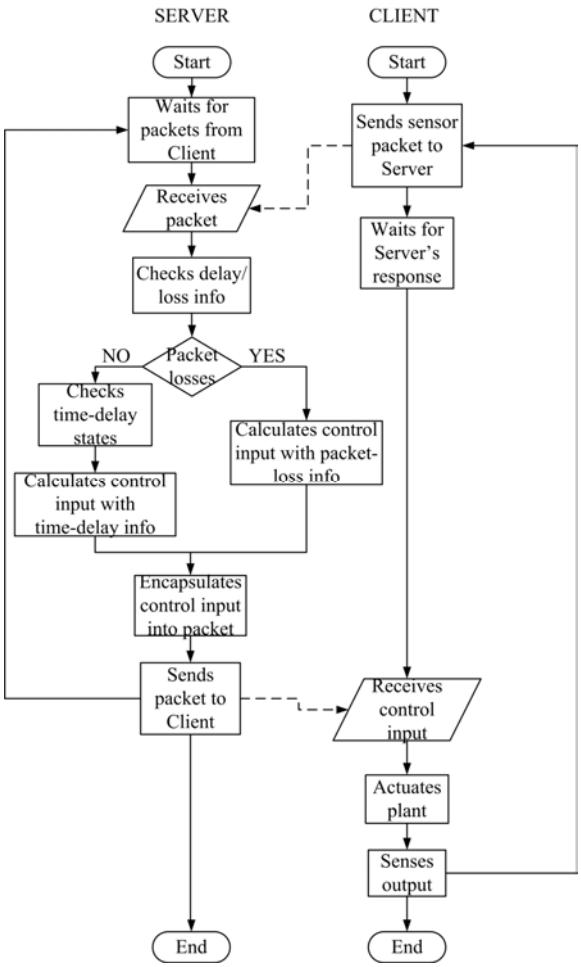


Fig. 3. Flow chart of algorithm implementation. The solid lines represent the independent control flow on Server and Client. The dashed lines represent the chronological data exchange between Server and Client.

4. CONTROLLER IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, key experimental results are provided to verify the effectiveness of our Markov-chain-based output feedback method. A DC-motor speed-control system was set up as the plant [23]. The speed of a DC motor is controlled over the NCS based on the Ethernet local area network (LAN) within our lab.

4.1. Experiments setup

Linux Redhat 7.3 with Real-Time Application Interface 3.4 (RTAI 3.4) [24] is the operating system running on Server, and Linux Ubuntu 6.10 with RTAI 3.4, on Client. The control and measurement device interface (Comedi) [25] is used as the drivers and libraries of data acquisition on Client. A PCI-6221 data-acquisition card by National Instruments enables the DC motor test bed to send out sensor-output data packets and receive control-input data packets through the LAN. The speed control is achieved by controlling the output voltage of a pulse-width modulation (PWM) amplifier. Fig. 4 shows the block diagram of the entire experimental setup.

The communication network in the experiment is the 100-Mbps Ethernet with unblocked UDP sockets. UDP is fast, unreliable, and connectionless compared to Transmission Control Protocol (TCP), but has a compact header. Depending on the specifications and system requirements of various NCSs, UDP can be a possible choice as a suitable protocol. For some NCSs, UDP is a preferred protocol for better performance [26]. Due to the real-time characteristics of our NCS, UDP is chosen to be the protocol for the experiment. More details about the comparison of TCP and UDP and the reasons to choose UDP as the communication protocol can be found in [26-28]. The data-packet structures for both Server and Client are as follow.

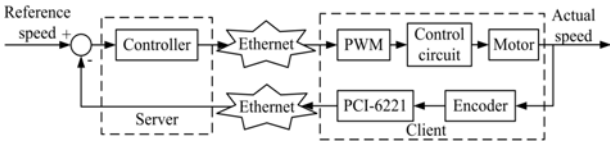


Fig. 4. Block diagram of the DC motor system.

Server	Ethernet Header (14 bytes)	IP Header (20 Bytes)	UDP Header (8 Bytes)	Control Data (16 Bytes)	Timestamp (8 Bytes)	Identifier (8 Bytes)

Client	Ethernet Header (14 bytes)	IP Header (20 Bytes)	UDP Header (8 Bytes)	Sensor Data (56 Bytes)	Timestamp (8 Bytes)	Identifier (8 Bytes)	Delay/loss info (8 Bytes)

Fig. 5. Data-packet structures.

As in Fig. 5, the Ethernet header, IP header, and UDP header are the standard protocol headers. Control data and sensor data are the data segments generated by Server and Client, respectively. Timestamp is set up by Client to track the total time delays. Identifier is to identify Server and Client. Delay/loss info is used to track the random time delays and packet losses. If a packet is lost, this segment contains negative value to notify Server. If no packet is lost, this segment contains the total time delays for Server to compare with the random time-delay states as defined above.

4.2. System modeling

Based on the DC motor datasheet [29], its state-space model can be represented as

$$x_p(k+1) = -0.26x_p(k) + 2.04u(k), \tag{27}$$

$$y(k) = x_p(k), \tag{28}$$

where $u(k) \in R$ is the input voltage, and $y(k) \in R$ is the angular velocity, respectively.

The network-induced time delays were measured to determine the key statistical characteristics of the NCS test bed. The time delays in the NCS with this DC motor system are shown in Fig. 6.

The random time-delay experiment was performed for 10,000 iterations, and the delays were measured in milliseconds. As aforementioned in Section 3, the time delays attained by the experiment are the total delays in the NCS. From Fig. 6, the average of the time delays is between 0.45 and 0.5 ms, and there are some jitters existing in time delays with the average of 0.8 ms. We took these two cases as two time-delay states for the Markov-chain-based model. According to the algorithm in Section 3, τ^{ca} and τ^{sc} in this experiment will be one half of the total time delays as indicated in (26) such that the time-delay Markov states of τ^{ca} and τ^{sc} will be

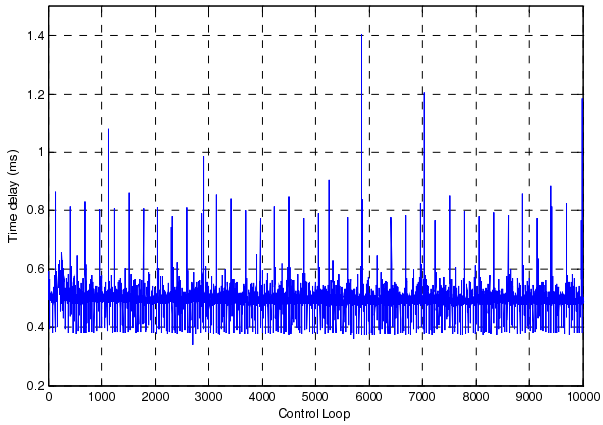


Fig. 6. Total time delays in the NCS.

$$\mathbb{P} = \mathbb{Q} = \{0.23, 0.4\}. \tag{29}$$

Equation (29) gives the Markov time-delay states of the experiments. The first Markov state of 0.23 ms represents the average of the time delays, and the second Markov state of 0.4 ms represents the jitters in either the controller-to-actuator link or the sensor-to-controller link. τ^{ca} and τ^{sc} will take one of the values in the set. As mentioned in Section 3, τ^{ca} and τ^{sc} are not necessarily the same, but we assume they are since the explicit time-delay information is unavailable in the experiment. Note that the random time delays may not be exactly the same for each sampling period, and each state in the set actually represents certain time intervals. A time delay shorter than 0.35 ms represents the first Markov state, and any time delay longer than 0.35 ms, the second Markov state.

By fixing the current Markov state, the transition-probability matrix can be constructed by counting the number of the next Markov state that falls into either the first Markov state or the second Markov state in (29). The transition-probability matrices of the two Markov states are determined experimentally as

$$\mathbf{\Lambda} = \mathbf{\Gamma} = \begin{bmatrix} 0.93 & 0.07 \\ 0.75 & 0.25 \end{bmatrix}. \tag{30}$$

Equation (30) gives the probability that the time delays jump from the current Markov state to the next Markov state. (i.e., if the current time delay is 0.23 ms, then the next time delay will be 0.23 ms at 93% probability, and be 0.4 ms, 7%).

4.3. Controller design and implementation

The LMI stability criterion developed in Section 2 has been applied in the Matlab with the LMI Toolbox, and the V-K iteration algorithm in [13] with the following initial **P** matrix. The matrix **P**(i, m) depends on the Markov states of τ^{ca} and τ^{sc}. For instance, if τ^{ca} is at the first Markov state of 0.23 ms, and τ^{sc} is at the second Markov state of 0.4 ms, **P**(i, m) will be denoted as **P**(1, 2). Set a state vector $\mathbf{w}_{12} = [0.23 \quad 0.4]^T$ for **P**(1, 2), and define $\mathbf{P}(1,2) \triangleq \gamma \cdot diag(\mathbf{w}_{12}\mathbf{w}_{12}^T) \otimes \mathbf{I}$, where γ is a weight coefficient for the optimization and $\otimes$ is the Kronecker product. The dimension of **I** depends on the problems, where **I** is 4×4 identity matrix in our experiments. All the other **P**(i, m) can be constructed in the same way. These initial **P**(i, m) will be applied to start the LMI solver and V-K iteration algorithm, which will converge to the final states at the end of all the iterations or when the errors satisfy the pre-set error bounds. The choice of initial **P**(i, m) may vary. The convergence of the V-K iteration algorithm can be referred to [13]. Then with solving (24) using Matlab LMI Toolbox and the V-K iteration algorithm with the corresponding constraints, the controller can be designed.

The controllers are designed as presented in Table 1. The 4-tuple $\{\tau^{ca}, \tau^{sc}, \delta^{ca}, \delta^{sc}\}$ in Table 1 represents different Markov states of the random time delays and packet losses as defined in Section 2. A_c, B_c, C_c, and D_c are the controller matrices defined in (3) and (4) and derived

Table 1. Controller parameters.

$\{\tau^{ca}, \tau^{sc}, \delta^{ca}, \delta^{sc}\}$	A_c	B_c	C_c	D_c
{0.23, 0.23, 1, 1}	1.0102	0.9687	0.0396	1.7621
{0.23, 0.4, 1, 1}	1.0155	0.9879	0.0408	1.7889
{0.4, 0.23, 1, 1}	1.0155	0.9879	0.0408	1.7889
{0.4, 0.4, 1, 1}	1.0412	1.0030	0.0421	1.8162
{–, –, 0, 0}	1.1974	1.1534	0.0557	2.0886

from Theorem 2. The order of the controller can be set as needed. A higher-order controller may promise more robust system performance but require more computational efforts and bring more complexity to the system. In our experiments, the plant represented with (27) and (28) is a first-order system. We design the controller to be first-order, so the whole closed-loop system is second-order.

As mentioned in Section 3, when the packet is lost, no time-delay information will be available. The 4-tuple {–, –, 0, 0} represents the case that both the controller-to-actuator and the sensor-to-controller packets are lost. The sensor-to-controller packet loss is represented by {–, –, 1, 0}. However, Server will not be able to calculate the updated control input since it has not received any newly updated output information. The other case of the controller-to-actuator packet loss is represented by {–, –, 0, 1}. When this happens, the updated control input is calculated by Server but cannot arrive at Client. Therefore, all these cases can be grouped into the case {–, –, 0, 0} in the experiments since Client will not receive any updated control input for all these three cases.

4.4. Experimental result

The system performance with the proposed method is used to compare the performance with that of the Proportional-Integral (PI) controller in [23]. The difference equation of the PI controller is

$$u(k) = u(k-1) - 1.4925e(k) + 1.5075e(k-1). \quad (31)$$

All experiments were executed with a 3-ms sampling period for 500 iterations. The reference speed at the DC motor in both of the experiments was set to be 10 revolutions per second (rps).

Two separate experiments, without packet losses and with 10% packet losses, were conducted to evaluate the effectiveness of the proposed method. All the experiments were executed under the same network condition as the time-delay experiment in Fig. 6. The Ethernet in our lab was robust that no packet losses occurred even with UDP. Therefore artificial packet losses were introduced to the NCS with an approximate 10% loss rate. A random function that takes the value from 0 to 1 was introduced, and a threshold of 0.1 (10% loss rate) was set for the comparison. If the random number was less than the threshold, the packet would be dropped from the NCS. Note that the random modulo operation does not generate a truly uniformly distributed random number in [0, 1], but it is generally a good approximation. Since we run the experiments with a large number of iterations, we assume that the packet-loss rate is about 10%.

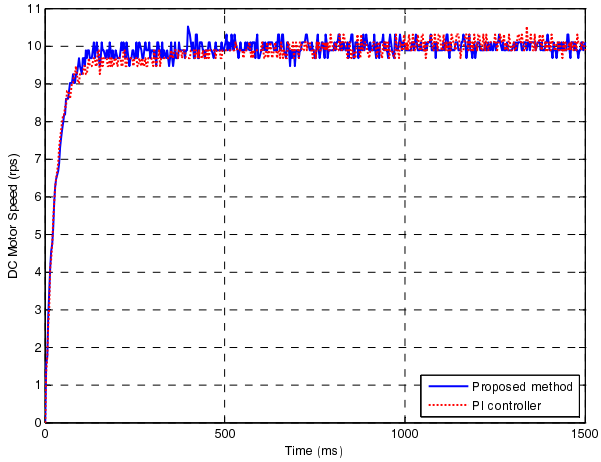


Fig. 7. Step responses of DC motor without packet losses.

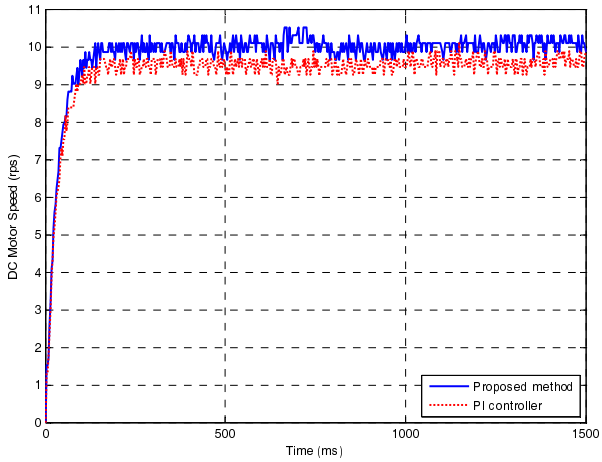


Fig. 8. Step responses of DC motor with 10% packet losses.

The results are shown in Figs. 7 and 8. Fig. 7 shows the results of the PI controller and the proposed controller without artificial packet losses. Without packet losses, the steady-state errors of the conventional PI control and the method proposed in this paper are almost the same. Fig. 8 shows the experimental results with 10% random artificial packet losses. As shown in Fig. 8, even when packets were lost in the communication network, our approach could track the reference command faithfully whereas the PI controller could not compensate for the random time delays and packet losses. The proposed method not only uses the previous control data but also compensates for the effect of the packet losses. Hence the system performance can be enhanced.

Figs. 9 and 10 show the ITAE of both of the experiments with the proposed method and the PI controller. Each figure shows the ITAE of the experimental data without packet losses and with 10% packet losses. From these figures, we can see when packets are lost in the communication network, system errors dramatically increase. From Figs. 9 and 10, the proposed method reduced the ITAE by about 13% without packet losses. For the packet losses case, the proposed method reduced the ITAE by as much as 30% compared to the PI controller.

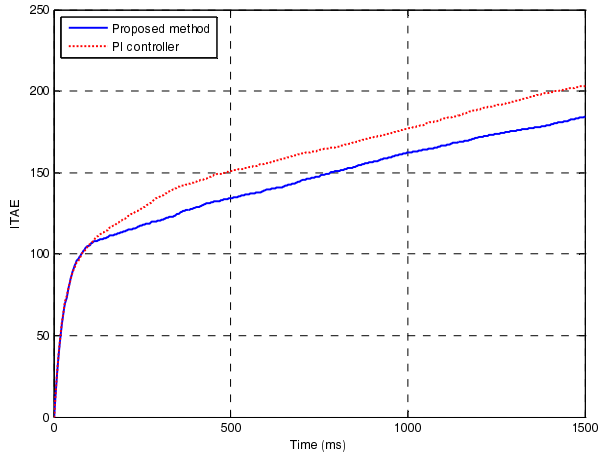


Fig. 9. ITAE comparisons of proposed method and the PI controller without packet losses.

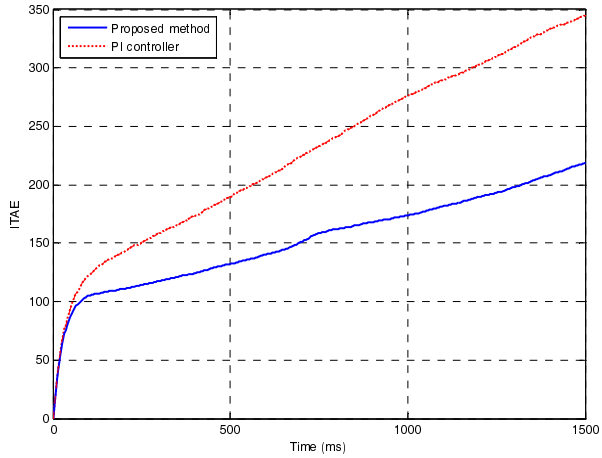


Fig. 10. ITAE comparisons of proposed method and the PI controller with packet losses.

In all these results, the Markov-chain-based method proposed in this paper exhibited excellent system performance.

5. CONCLUSION

This paper proposed an output feedback method for the stabilization and control of NCSs with random time delays and packet losses. This proposed method requires less computational effort and memory usage. By modeling the random time delays with time-homogeneous Markov chains and packet losses with Dirac delta functions, the closed-loop system was stabilized, and the performance was much enhanced compared to a conventional control method. The asymptotic mean-square stability criterion for the NCSs was obtained in terms of a Lyapunov function and a set of LMIs with matrix constraints. An algorithm implementation of the stability criterion was also presented in the paper. We constructed and employed a DC-motor speed-control test bed over a 100-Mbps Ethernet using unblocked UDP sockets for experimental verification. The experimental results demonstrated the feasibility and effectiveness of the proposed method. The proposed method enhanced the system performance with and without packet losses compared to a conventional control algorithm. The ITAE without packet losses was reduced by 13% with the proposed method, and the ITAE with 10% packet losses, by 30%. The NCS could track the reference command faithfully with the proposed method when random time delays and packet losses existed in the links whereas the NCS failed to track the reference command with a conventional control algorithm.

REFERENCES

[1] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: adaptive gradient climbing in a distributed environment," *IEEE Trans. on Automatic Control*, vol. 49, no. 8, pp. 1292-1302, August 2004.

[2] C. Meng, T. Wang, W. Chou, S. Luan, Y. Zhang, and Z. Tian, "Remote surgery case: robot-assisted teleneurosurgery," *Proc. of IEEE International Conference of Robot and Automation*, vol. 1, pp. 819-823, April 2004.

[3] J. P. Hespanha, M. L. McLaughlin, and G. Sukhatme, "Haptic collaboration over the internet," *Proc. of the 5th Phantom Users Group Workshop*, pp. 9-13, October 2000.

[4] K. Hikichi, H. Morino, I. Arimoto, K. Sezaki, and Y. Yasuda, "The evaluation of delay jitter for haptics collaboration over the internet," *Proc. of IEEE Global Telecomm Conference*, vol. 2, pp. 1492-1496, November 2002.

[5] P. Seiler and R. Sengupta, "Analysis of communication losses in vehicle control problems," *Proc. of American Control Conference*, vol. 2, pp. 1491-1496, June 2001.

[6] P. Seiler and R. Sengupta, "An H_∞ approach to networked control," *IEEE Trans. on Automatic Control*, vol. 50, no. 3, pp. 356-364, March 2005.

[7] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control system," *IEEE Control Systems Magazine*, vol. 21, no. 2, pp. 84-99, February 2001.

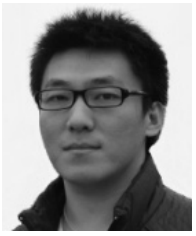
[8] S.-L. Hu, J.-L. Liu, and Z.-P. Du, "Stabilization of discrete-time networked control systems with partly known transmission delay: a new augmentation approach," *International Journal of Control, Automation and Systems*, vol. 9, no. 6, pp. 1080-1085, December 2011.

[9] J. Nilsson, *Real-time Control Systems with Delays*, Ph.D. dissertation, Lund Institute of Technology, Lund, Sweden, 1998.

[10] F. Yang, W. Wang, Y. Niu, and Y. Li, "Observer-based H_∞ control for networked systems with consecutive packet delays and losses," *International Journal of Control, Automation and Systems*, vol. 8, no. 4, pp. 769-775, August 2010.

[11] T. Jia, Y. Niu, and X. Wang, "H_∞ control for networked systems with data packet dropout," *International Journal of Control, Automation and Systems*, vol. 8, no. 2, pp. 198-203, April 2010.

[12] R. Krtolica, U. Ozguner, H. Chan, H. Goktas, J. Winkelman, and M. Liubakka, "Stability of linear feedback systems with random communication delays," *International Journal of Control*, vol. 59, no. 4, pp. 925-953, April 1994.
[13] L. Xiao, A. Hassibi, and J. P. How, "Control with random communication delays via a discrete-time jump system approach," *Proc. of American Control Conference*, vol. 3, pp. 2199-2204, June 2000.
[14] L. Zhang, Y. Shi, T. Chen, and B. Huang, "A new method for stabilization of networked control systems with random delays," *IEEE Trans. on Automatic Control*, vol. 50, no. 8, pp. 1177-1181, August 2005.
[15] X. Ye, S. Liu, and P. X. Liu, "Brief paper: modeling and stabilisation of networked control system with packet loss and time-varying delays," *IET Control Theory and Application*, vol. 6, no. 6, pp. 1094-1100, June 2010.
[16] J. Xiong and J. Lam, "Stabilization of linear systems over networks with bounded packet loss," *Automatica*, vol. 43, no. 1, pp. 80-87, January 2007.
[17] G. P. Liu, J. X. Mu, D. Rees, and S. C. Chai, "Design and stability analysis of networked control systems with random communication time delay using the modified MPC," *International Journal of Control*, Vol. 79, no. 4, pp. 288-297, April 2006.
[18] L. Schenato, "Optimal estimation in networked control systems subject to random delay and packet drop," *IEEE Trans. on Automatic Control*, vol. 53, no. 5, pp. 1311-1317, June 2008.
[19] S. C. Smith and P. Seiler, "Estimation with lossy measurements: jump estimators for jump system," *IEEE Trans. on Automatic Control*, vol. 48, no. 22, pp. 2163-2171, December 2003.
[20] J. Xu and J. P. Hespanha, "Estimation under uncontrolled and controlled communications in networked control systems," *Proc. of the 43rd Conf. Decision and Contr.*, pp. 3527-3532, December 2005.
[21] L. Zhang, B. Huang, and J. Lam, "H_∞ model reduction of Markovian jump linear systems," *Systems & Control Letters*, vol. 50, pp. 103-118, October 2003.
[22] L. E. Shaikhet, "Necessary and sufficient conditions of asymptotic Mean-square stability for stochastic linear difference equations," *Appl. Math. Letters*, vol. 10, no. 3, pp. 111-115, May 1997.
[23] M. H. Lee, *Real-time Networked Control with Multiple Clients*, M.S. thesis, Texas A&M University, College Station, TX, August 2009.
[24] P. Mantegazza, *DIAPM RTAI - Real-time Application*, [Online] Available: http://www.rtai.org.
[25] D. Schleef, *Linux Control and Measurement Device Interface*, [Online] Available: http://www.comedi.org.
[26] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, "Closed-loop control over wireless network," *IEEE Control System Magazine*, vol. 24, no. 3, pp. 57-71, June 2004.
[27] K. Ji and W. J. Kim, "Real-time control of networked control systems via Ethernet," *International Journal of Control, Automation and Systems*, vol. 3, no. 4, pp. 591-600, December 2005.
[28] I. Lopez, J. L. Piovesan, C. T. Abdallah, D. Lee, O. Martinez, and M. Spong, "Practical issues in networked control systems," *Proc. of American Control Conference*, pp. 4201-4206, June 2006.
[29] A-max 26, *Maxon DC Motor Datasheet*, [Online] Available: http://www.maxonmotorusa.com.



Jiawei Dong received his B.S. degree in mechanical manufacture and automation from the Beijing Technology and Business University (BTBU), Beijing, China in 2005, and his M.S. degree in advanced robotics from the Beijing University of Aeronautics and Astronautics (BUAA), Beijing, China in 2008, respectively. He is currently working toward a Ph.D. degree with the Department of Mechanical Engineering, Texas A&M University (TAMU), College Station. His research interests include networked control systems, real-time systems, optimization, and advanced robotics.

Won-jong Kim received his B.S. (*summa cum laude*) and M.S. degrees in control and instrumentation engineering from Seoul National University, Seoul, Korea, in 1989 and 1991, respectively, and his Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1997. Since 2000, he has been with the Department of Mechanical Engineering, Texas A&M University (TAMU), College Station, where currently he is Associate Professor and was the inaugural Holder of the Gulf Oil/Thomas A. Dietz Career Development Professorship II in 2007-2010. His current research interests include the analysis, design, and real-time control of mechatronic systems, networked control systems, and nanoscale engineering and technology. He is the holder of three U.S. patents on precision positioning systems. He is Fellow of ASME, Senior Member of IEEE, and Member of Pi Tau Sigma. Prof. Kim is Technical Editor of *IEEE/ASME Transactions on Mechatronics*, *ASME Journal of Dynamic Systems, Measurement and Control, International Journal of Control, Automation, and Systems*, and *Asian Journal of Control*.