FEATURE

By Wei Zhang, Michael S. Branicky, and Stephen M. Phillips

Stability of Networked Control Systems

Feedback control systems wherein the control loops are closed through a real-time network are called networked control systems (NCSs) [1]-[4]. The defining feature of an NCS is that information (reference input, plant output, control input, etc.) is exchanged using a network among control system components (sensors, controller, actuators, etc.). Fig. 1 illustrates a typical setup and the information flows of an NCS. The primary advantages of an NCS are reduced system wiring, ease of system diagnosis and maintenance, and increased system agility.

The insertion of the communication network in the feedback control loop makes the analysis and design of an NCS complex. Conventional control theories with many ideal assumptions, such as synchronized control and nondelayed sensing and actuation, must be reevaluated before they can be applied to NCSs. Specifically, the following issues need to be addressed. The first issue is the network-induced delay (sensor-to-controller delay and controller-to-actuator delay) that occurs while exchanging data among devices connected to the shared medium. This delay, either constant (up to jitter) or time varying, can degrade the performance of control systems designed without considering the delay and can even destabilize the system. Next, the network can be viewed as a web of unreliable transmission paths. Some packets not only suffer transmission delay but, even worse, can be lost during transmission. Thus, how such packet

Zhang, Branicky (msb11@po.cwru.edu), and Phillips are with the Electrical Engineering and Computer Science Department, Case Western Reserve University, 10900 Euclid Ave., Cleveland, OH 44106-7221, U.S.A.

0272-1708/01/$10.00©2001IEEE

dropouts affect the performance of an NCS is an issue that must be considered. Another issue is that plant outputs may be transmitted using multiple network packets (so-called *multiple-packet transmission*), due to the bandwidth and packet size constraints of the network. Because of the arbitration of the network medium with other nodes on the network, chances are that all/part/none of the packets could arrive by the time of control calculation.

The implementation of distributed control can be traced back at least to the early 1970s when Honeywell's Distributed Control System (DCS) was introduced. Control modules in a DCS are loosely connected because most of the real-time control tasks (sensing, calculation, and actuation) are carried out within individual modules. Only on/off signals, monitoring information, alarm information, and the like are transmitted on the serial network. Today, with help from ASIC chip design and significant price drops in silicon, sensors and actuators can be equipped with a network interface and thus can become independent nodes on a real-time control network. Hence, in NCSs, real-time sensing and control data are transmitted on the network, and network nodes need to work closely together to perform control tasks.

Current candidate networks for NCS implementations are DeviceNet [5], Ethernet [6], and FireWire [7], to name a few. Each network has its own protocols that are designed for a specific range of applications. Also, the behavior of an NCS largely depends on the performance parameters of the underlying network, which include transmission rate, medium access protocol, packet length, and so on.

There are two main approaches for accommodating all of these issues in NCS design. One way is to design the control system without regard to the packet delay and loss but design a communication protocol that minimizes the likelihood of these events. For example, various congestion control and avoidance algorithms have been proposed [8], [9] to gain better performance when the network traffic is above the limit that the network can handle. The other approach is to treat the network protocol and traffic as given conditions and design control strategies that explicitly take the above-mentioned issues into account. To handle delay, one might formulate control strategies based on the study of delay-differential equations [10]. Here, we discuss analysis and design strategies for both network-induced delay and packet loss.

This article is organized as follows. First, we review some previous work on NCSs and offer some improvements. Then, we summarize the fundamental issues in NCSs and examine them with different underlying network-scheduling protocols. We present NCS models with network-induced delay and analyze their stability using stability regions and a hybrid systems technique. Following that, we discuss methods to compensate network-induced delay and present experimental results over a physical network. Then, we model NCSs with packet dropout and multiple-packet transmission as asynchronous dynamical systems (ADSs) [11] and analyze their stability. Finally, we present our conclusions.

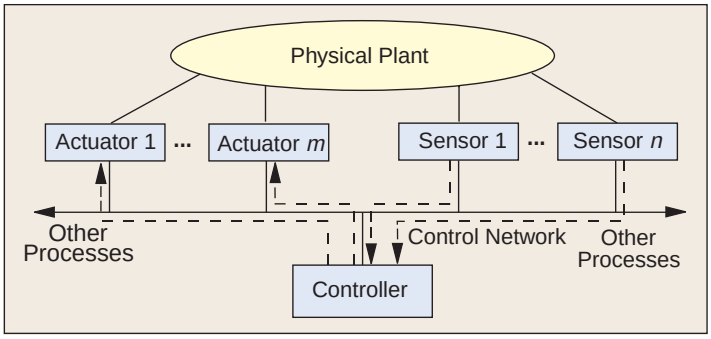

Figure 1. *A typical NCS setup and information flows.*

Review of Previous Work

Halevi and Ray [1] consider a continuous-time plant and discrete-time controller and analyze the integrated communication and control system (ICCS) using a discrete-time approach. They study a clock-driven controller with *mis-synchronization* between plant and controller. The system is represented by an augmented state vector that consists of past values of the plant input and output, in addition to the current state vectors of the plant and controller. This results in a finite-dimensional, time-varying discrete-time model. They also take *message rejection* and *vacant sampling* into account.

Nilsson [2] also analyzes NCSs in the discrete-time domain. He further models the network delays as constant, independently random, and random but governed by an underlying Markov chain. From there, he solves the LQG optimal control problem for the various delay models. He also points out the importance of time-stamping messages, which allows the history of the system to be known.

In Walsh et al. [3], the authors consider a continuous plant and a continuous controller. The control network, shared by other nodes, is only inserted between the sensor nodes and the controller. They introduce the notion of maximum allowable transfer interval (MATI), denoted by τ, which supposes that successive sensor messages are separated by at most τ seconds. Their goal is to find that value of τ for which the desired performance (e.g., stability) of an NCS is guaranteed to be preserved.

It is assumed that the nonnetworked feedback system

$$\dot{x}(t) = A_{11}x(t), \quad x(t) = \left[x_p(t), x_c(t)\right]^T$$

(where x_p and x_c represent the plant and controller state) is globally exponentially stable. Thus, there exists a P such that

$$A_{11}^T P + PA_{11} = -I. \tag{1}$$

Next, it is assumed that the network's effects can be computed by the error, $e(t)$, between the plant output and controller input. So the networked system's state vector is

$z(t) = [x^T(t), e^T(t)]^T$, and thus the networked closed-loop system is

$$\dot{z}(t) = Az(t)$$

where A can be partitioned as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{2}$$

Walsh et al. study two scheduling methods: try-once-discard (TOD) and token-ring-type static scheduling. Assuming there are p sensor nodes connected to the NCS, static scheduling simply means that each node transmits exactly once every p transmissions in a fixed order. Under the MATI constraint, the controller must receive a transmission from at least one of the sensors every τ seconds. Hence, under static scheduling, all sensor values are updated in at most $p\tau$ seconds.

TOD is a scheduling protocol in which the node with the greatest weighted error from its last reported value (to the controller) transmits its message. Again, the MATI constraint ensures at least one such transmission every τ seconds. However, TOD does not guarantee that each node will transmit once every p transmissions.

The defining feature of an NCS is that information is exchanged using a network among control system components.

For each of these protocols, one can compute an upper bound on the MATI τ that preserves stability of the closed-loop system. The result is given in the following theorem.

Theorem 1 [3, Theorem 2]: Given an NCS with p sensor nodes operating under TOD or static scheduling, define $\lambda_1 = \lambda_{\min}(P), \lambda_2 = \lambda_{\max}(P)$ (where P was defined above). If the MATI satisfies

$$\tau < \min\left\{\frac{\ln(2)}{p\|A\|}, \frac{1}{8\|A\|(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i}, \frac{1}{16\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|^2(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i}\right\},$$

then the NCS is globally exponentially stable.

The calculation of the bound for τ can be generalized and tightened by the following corollary.

Corollary 2: If the Lyapunov function $V(x) = x^T Px$ of the nonnetworked, closed-loop system satisfies

$$A_{11}^T P + PA_{11} = -Q, \tag{3}$$

(more general than (1)), where P, Q are positive-definite symmetric matrices, the bound on τ becomes

$$\tau < \min\left\{\frac{\ln(2)}{p\|A\|}, \frac{1}{8\|A\|(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i}, \frac{\lambda_{\min}(Q)}{16\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|^2(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i}\right\}.$$

Furthermore, the third term is always the smallest, so

$$\tau < \frac{\lambda_{\min}(Q)}{16\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|^2(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i} \tag{4}$$

guarantees the global exponential stability of the NCS.

Proof: See the Appendix.

Corollary 2 shows that the MATI τ depends on $\|A\|, p$, and Q; Q in turn determines P using (3). $\|A\|$ and p are fixed for a particular system setup; thus Q is the only variable in choosing τ. One might use an analytic method to find the Q that could maximize τ. By maximization we mean the largest τ possible that could still preserve stability of the NCS. However, the following example illustrates the use of random search in choosing τ.

Example 1: Consider the state-space plant model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \end{bmatrix} u,$$
$$y = [1 \quad 0]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{5}$$

A continuous-state feedback controller is $u = -Kx$, where $K = [3.75, 11.5]$ (closed-loop poles at -1/2 and -3/4).

Using Theorem 1, for $p = 1$ (only one node, which is a nonnetworked sampled-data system), we obtain $\tau = 2.7 \times 10^{-4}$ s. By randomly selecting Q and solving for P, we can calculate τ using the formula in Corollary 2. In 200 trials, the maximum τ found was 4.5×10^{-4} s. However, the maximum stable constant sampling period for this feedback control system is 1.7 s (this can be determined using the "stability region" technique we discuss below), which shows that Theorem 1 and Corollary 2 may be conservative.

Theorem 1 and Corollary 2 give sufficient conditions on the network sampling rate to guarantee that the original nonnetworked system remains stable when the control loop is closed over the network. They might be too conservative,

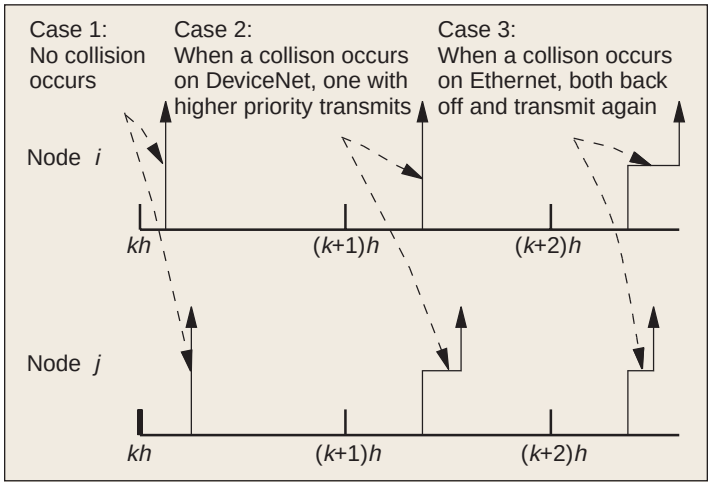

Figure 2. *Timing diagram for two nodes on a random access network.*

however, to be of practical use. We later examine stability for some specific examples to develop some insight into the problem.

Fundamental Issues in NCSs

In this section, we will analyze some basic problems in NCSs, including network-induced delay, single-packet or multiple-packet transmission of plant inputs and outputs, and dropping of network packets.

Network-Induced Delay

The network-induced delay in NCSs occurs when sensors, actuators, and controllers exchange data across the network. This delay can degrade the performance of control systems designed without considering it and can even destabilize the system.

Depending on the medium access control (MAC) protocol of the control network, network-induced delay can be constant, time varying, or even random. MAC protocols generally fall into two categories: *random access* and *scheduling* [12]. Carrier sense multiple access (CSMA) is most often used in random access networks, whereas token passing (TP) and time division multiple access (TDMA) are commonly employed in scheduling networks.

Control networks using CSMA protocols include DeviceNet [5] and Ethernet [6]. Fig. 2 illustrates various possible situations for this type of network. The figure depicts two nodes continually transmitting messages (with respect to a fixed time line). A node on a CSMA network monitors the network before each transmission. When the network is idle, it begins transmission immediately, as shown in Case 1 of Fig. 2. Otherwise it waits until the network is not busy. When two or more nodes try to transmit simultaneously, a collision occurs. The way to resolve the collision is protocol dependent. DeviceNet, which is a controller area network (CAN), uses CSMA with a bitwise arbitration (CSMA/BA) protocol. Since CAN messages are prioritized, the message with the highest priority is transmitted without interruption when a collision occurs, and transmission of the lower priority message is terminated and will be retried when the network is idle, as shown in Case 2 of Fig. 2. Ethernet employs a CSMA with collision detection (CSMA/CD) protocol. When there is a collision, all of the affected nodes will back off, wait a random time (usually decided by the *binary exponential backoff* algorithm [6]), and retransmit, as shown in Case 3 of Fig. 2. Packets on these types of networks are affected by random delays, and the worst-case transmission time of packets is unbounded. Therefore, CSMA networks are generally considered nondeterministic. However, if network messages are prioritized, higher priority messages have a better chance of timely transmission.

The TP protocol appears in token bus (IEEE Standard 802.4), token ring (IEEE Standard 802.5) [6], and the fiber distributed data interface (FDDI) MAC [13] architectures; TDMA is used in FireWire [7]. A timing diagram for this type of network is shown in Fig. 3. These protocols eliminate the contention for the shared network medium by allowing each node on the network to transmit according to a predetermined schedule. In a token bus, the token is passed around a logical ring, whereas in a token ring, it is passed around a physical ring. In scheduling networks, it is possible to arrange for periodic transmission of messages. For example, FireWire has a transmission cycle (125 μs) divided into small time slots, where each *isochronous transaction* is guaranteed a time slot to transmit in every cycle. Packet transmission delays on scheduling networks occur while waiting for the token or time slot. They can be made both bounded and constant by transmitting packets periodically.

Single-Packet versus Multiple-Packet Transmission

Single-packet transmission means that sensor or actuator data are lumped together into one network packet and transmitted at the same time, whereas in multiple-packet transmission, sensor or actuator data are transmitted in separate network packets, and they may not arrive at the controller and plant simultaneously. One reason for multiple-packet transmission is that packet-switched networks can only carry limited information in a single packet due to packet size constraints. Thus, large amounts of data must be broken into multiple packets to be transmitted. The

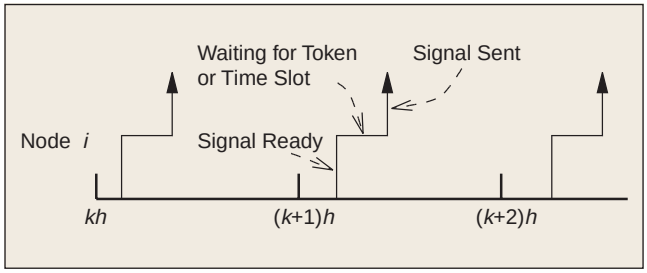

Figure 3. *Timing diagram for an arbitrary node on a scheduling network.*

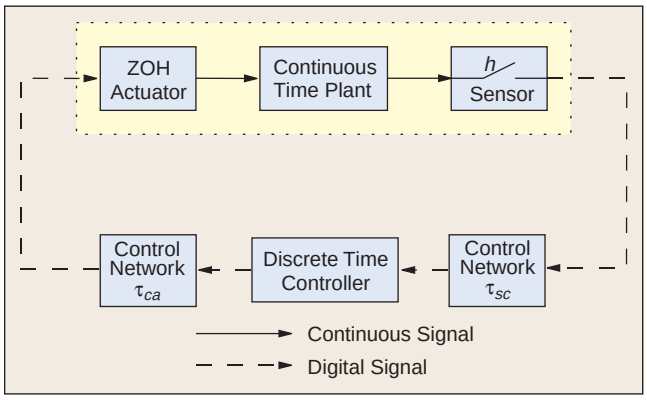

Figure 4. *NCS model with network-induced delay.*

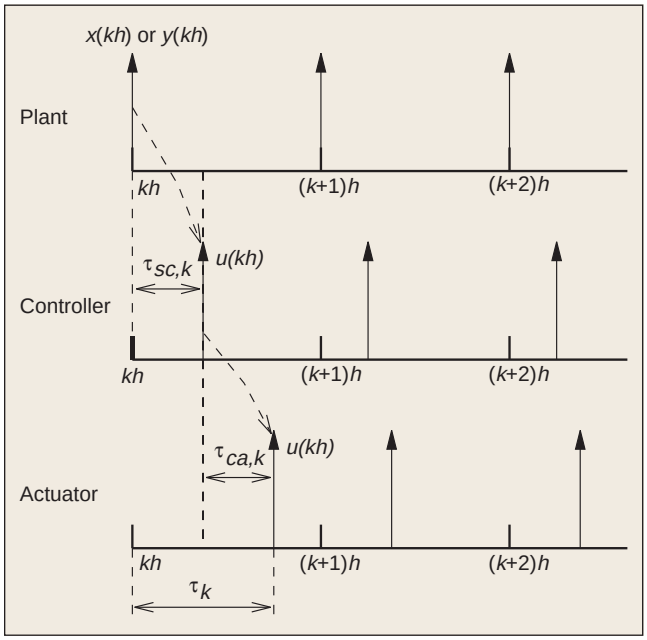

Figure 5. *Network-induced delay.*

other reason is that sensors and actuators in an NCS are often distributed over a large physical area, and it is impossible to put the data into one network packet.

Conventional sampled-data systems assume that plant outputs and control inputs are delivered at the same time, which may not be true for NCSs with multiple-packet transmissions. Due to network access delays, the controller may not be able to receive all of the plant output updates at the time of the control calculation.

Different networks are suitable for different types of transmissions. Ethernet, originally designed for transmitting information such as data files, can hold a maximum of 1500 bytes of data in a single packet [6]. Hence, it is more efficient to lump the sensor data into one packet and transmit it together-single-packet transmission. On the other hand, DeviceNet, featuring frequent transmission of small-size control data, has a maximum 8-byte data field in each packet; thus, sensor data often must be shuttled in different packets on DeviceNet.

Dropping Network Packets

Network packet drops occasionally happen on an NCS when there are node failures or message collisions. Although most network protocols are equipped with transmission-retry mechanisms, they can only retransmit for a limited time. After this time has expired, the packets are dropped. Furthermore, for real-time feedback control data such as sensor measurements and calculated control signals, it may be advantageous to discard the old, untransmitted message and transmit a new packet if it becomes available. In this way, the controller always receives fresh data for control calculation.

Normally, feedback-controlled plants can tolerate a certain amount of data loss, but it is valuable to determine whether the system is stable when only transmitting the packets at a certain rate and to compute acceptable lower bounds on the packet transmission rate.

Stability of NCSs with Network-Induced Delay

Modeling NCSs with Network-Induced Delay

The NCS model considering network-induced delay is shown in Fig. 4. The model consists of a continuous plant

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{6}$$

and a discrete controller

$$u(kh) = -Kx(kh), \quad k = 0,1,2,\ldots. \tag{7}$$

Here, $x \in \mathrm{R}^n, u \in \mathrm{R}^m, y \in \mathrm{R}^p$, and A, B, C, K are of compatible dimensions.

There are two sources of delays from the network: sensor-to-controller τ_{sc} and controller-to-actuator τ_{ca}. Any controller computational delay can be absorbed into either τ_{sc} or τ_{ca} without loss of generality [2]. For fixed control law (time-invariant controllers), the sensor-to-controller delay and controller-to-actuator delay can be lumped together as $\tau = \tau_{sc} + \tau_{ca}$ for analysis purposes.

We consider the setup with a) clock-driven sensors that sample the plant outputs periodically at sampling instants; b) an event-driven controller, which can be implemented by an external event interrupt mechanism and which calculates the control signal as soon as the sensor data arrives; and c) event-driven actuators, which means the plant inputs are changed as soon as the data become available. The timing of signals of the setup with $\tau < h$ is shown in Fig. 5.

Delay Less than One Sampling Period

First consider the case where the delay of each sample, τ_k, is less than one sampling period, h. (Here the subscript represents the sampling instant.) This constraint means that at

most two control samples, $u((k-1)h)$ and $u(kh)$, need be applied during the kth sampling period. The system equations can be written as

$$\begin{aligned} \dot{x}(t) &= Ax(t)+Bu(t), \quad t\in[kh+\tau_k,(k+1)h+\tau_{k+1}),\\ y(t) &= Cx(t),\\ u(t^+) &= -Kx(t-\tau_k), \quad t\in\{kh+\tau_k,\ k=0,1,2,\ldots\} \end{aligned} \tag{8}$$

where $u(t^+)$ is piecewise continuous and only changes value at $kh+\tau_k$. Sampling the system with period h we obtain [14]

$$\begin{aligned} x((k+1)h) &= \Phi x(kh)+\Gamma_0(\tau_k)u(kh)+\Gamma_1(\tau_k)u((k-1)h),\\ y(kh) &= Cx(kh) \end{aligned}$$

where

$$\begin{aligned} \Phi &= e^{Ah},\\ \Gamma_0(\tau_k) &= \int_0^{h-\tau_k} e^{As}B\,ds,\\ \Gamma_1(\tau_k) &= \int_{h-\tau_k}^{h} e^{As}B\,ds. \end{aligned}$$

Defining $z(kh)=[x^T(kh),u^T((k-1)h)]^T$ as the augmented state vector, the augmented closed-loop system is

$$z((k+1)h)=\tilde{\Phi}(k)z(kh) \tag{9}$$

where

$$\tilde{\Phi}(k)=\begin{bmatrix}\Phi-\Gamma_0(\tau_k)K & \Gamma_1(\tau_k)\\ -K & 0\end{bmatrix}.$$

If the delay is constant (i.e., $\tau_k=\tau$ for $k=0,1,2,\ldots$), the system is still time invariant, which simplifies the system analysis. Thus we can envision static scheduling network protocols, such as token ring or token bus, which can provide constant delay. Even in this simplified setup, the next question is, "How much delay can the system tolerate?"

Another observation is that the sensor-controller delay can be compensated by an estimator if the messages sent out by sensors are time stamped (cf. [2]). Traditional one-step prediction estimation can compensate delays less than one sampling period, since the estimate of $x(kh)$ only depends on the value of $y((k-1)h)$. We will revisit this problem in the section on compensation for network-induced delay.

Longer Delays

When the delays can be longer than one sampling period (say, $0<\tau_k<lh$, $l>1$), one may receive zero, one, or more than one (up to l) control sample(s) in a single sampling period. In the special case where $(l-1)h<\tau_k<lh$ for all k, one control sample is received every sample period for $k>l$. In this case, the analysis follows that in [14], resulting in

$$\tilde{\Phi}(k)=\begin{bmatrix}\Phi & \Gamma_1(\tau'_k) & \Gamma_0(\tau'_k) & \cdots & 0\\ 0 & 0 & I & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ -K & 0 & 0 & \cdots & 0\end{bmatrix}, \tag{10}$$

where $\tau'_k=\tau_k-(l-1)h$ and the augmented state vector is $z(kh)=[x^T(kh),u^T((k-l)h),\ldots,u^T((k-1)h)]^T$.

In the more general case, tedious bookkeeping must be performed, as even the block structure of the matrix $\tilde{\Phi}$ is time varying, since it depends on the schedule of the receipt of the control samples.

Stability Regions

Conventionally, a faster sampling rate is desirable in sampled-data systems so the discrete-time control design and performance can approximate that of the continuous system. But in NCSs, a faster sampling rate can increase the network load, which in turn results in longer delay of the signals. Thus finding a sampling rate that can both tolerate the network-induced delay and achieve desired system performance is important in NCS design.

Plotting the stability region of an NCS with respect to the sampling rate, h, and network delay, τ, is helpful to see the relationship between these two parameters. Note that here we are considering constant delay, which can be achieved by using an appropriate network protocol.

Integrator Case

The relationship between h and τ can be derived analytically for simple scalar systems.

Example 2: Consider the *integrator* example

$$\begin{aligned} \dot{x}(t) &= u(t), \quad t\in[kh+\tau,(k+1)h+\tau), \quad \tau<h,\\ u(t^+) &= -Kx(t-\tau), \quad t\in\{kh+\tau,k=0,1,2,\ldots\}, \quad K>0. \end{aligned} \tag{11}$$

Defining $z(kh)$ as in (9)

$$\tilde{\Phi}=\begin{bmatrix}1-hK+\tau K & \tau\\ -K & 0\end{bmatrix}.$$

For this 2×2 case, we can use the stability triangle [15] to explicitly calculate the relation between τ and h. For a stable NCS, the delay τ must satisfy

$$\max\left\{\frac{1}{2}h-\frac{1}{K},0\right\}<\tau<\min\left\{\frac{1}{K},h\right\} \tag{12}$$

or

$$\max\left\{\frac{1}{2}-\frac{1}{Kh},0\right\}<\frac{\tau}{h}<\min\left\{\frac{1}{Kh},1\right\}.$$

The analytically determined stability region for $0\le\tau<h$ is shown in Fig. 6. We can see from the stability region that

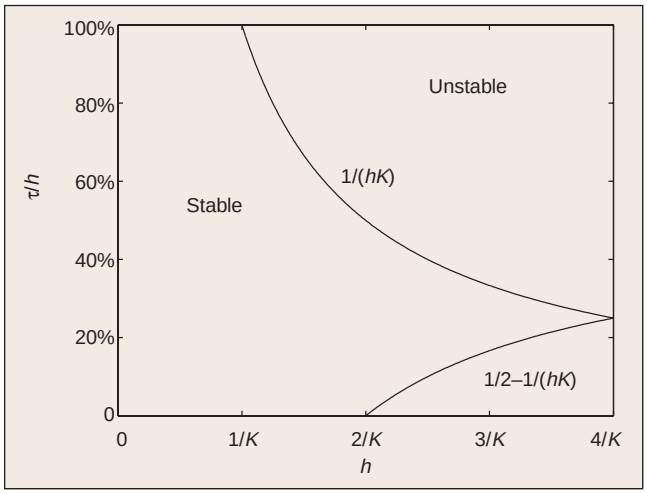

Figure 6. *Stability region of a controlled integrator.*

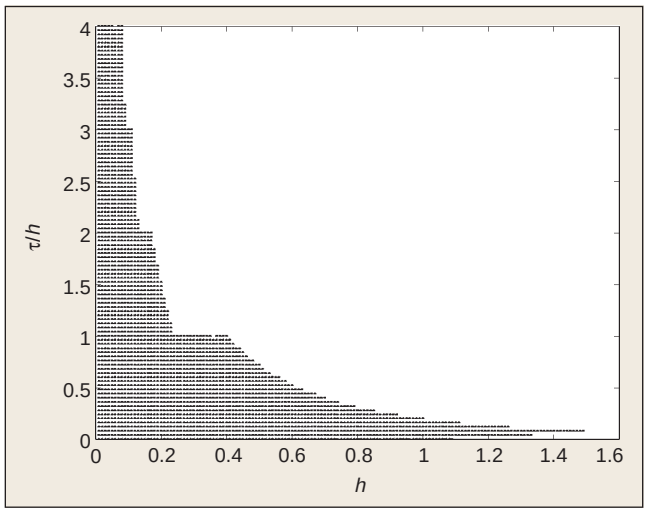

Figure 7. *Simulation of the stability region of* $\dot{x}(t) = x(t) - Kx(t-\tau)$ *with* $K = 2$ *and* $0 < \tau < 4h$.

when the sampling period h is small, the system can tolerate a delay up to one full sampling period. As h becomes larger, the upper bound on τ/h becomes smaller. Note that for $K > 2/h$, even the system with no delay is unstable.

General Scalar System

It may be analytically infeasible to derive the exact stability region for general systems; however, stability regions for such systems can still be determined by simulation. The stability region is plotted by incrementally increasing the delay, τ, and testing the closed-loop system matrix, as formulated in (9) and (10). If the closed-loop system matrix is stable, a point is marked in that location of the stability region.

Example 3: For a general scalar system

$$\begin{aligned} \dot{x}(t) &= ax(t) + u(t), \quad t \in [kh+\tau, (k+1)h+\tau), \\ u(t^+) &= -Kx(t-\tau), \quad t \in \{kh+\tau, k = 0,1,2,\ldots\}. \end{aligned}$$

Defining $z(kh)$ as in (9)

$$\tilde{\Phi} = \begin{bmatrix} e^{ah} - \dfrac{K}{a}\left(e^{a(h-\tau)} - 1\right) & \dfrac{1}{a}e^{ah}\left(1 - e^{-a\tau}\right) \\ -K & 0 \end{bmatrix}.$$

The stability region can be determined by simulation. A special scalar case with $a = 1$ and $K = 2$ is shown in Fig. 7. For this simulation, we considered delays between 0 and $4h$. We can see that when $0 \le \tau < h$, the region has a shape similar to the integrator case. The shape of the stability region is also affected by the feedback controller (in this case, the scalar feedback gain).

Analyzing Stability Using a Hybrid Systems Technique

The stability of an NCS with network-induced delay can also be analyzed using a hybrid systems stability analysis technique. Hybrid systems contain continuous dynamics and discrete events [16]. The NCS model we are studying resembles a class of hybrid systems with fixed instants of impulse effect. The stability of such continuous-discrete systems was reviewed and extended in [17], where linearized hybrid systems of the following form are considered:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + f(x(t), u(t), t), \quad t \in I \setminus \Theta, \\ u(t^+) &= Cx(t) + Du(t) + \phi(x(t), u(t), t), \quad t \in \Theta, \end{aligned} \tag{13}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $\Theta = \{t_k \mid t_k = kh,\ h > 0,\ k = 0,1,2,\ldots\}$. Let $z(t) = [x^T(t), u^T(t)]^T$; then $f(z,t): \Omega_0 \times I \to R^n$ is continuous in z in the neighborhood $\Omega_0 \subset \mathbb{R}^{n+m}$ for any t in the interval $I \subset \mathbb{R}_+$. Furthermore, $f(0,t) = 0, t \in I$, $\phi(0,t) = 0, t \in \Theta$ and for $z', z'' \in \Omega_0$, the conditions

$$\|f(z',t) - f(z'',t)\| \le L_1 \|z' - z''\|^{1+\alpha}; \quad L_1, \alpha > 0; \quad t \in I$$

and

$$\|\phi(z',t) - \phi(z'',t)\| \le L_2 \|z' - z''\|^{1+\alpha}; \quad L_2, \alpha > 0; \quad t \in \Theta$$

hold. The stability of this type of system reduces to evaluating the Schur-ness (i.e., whether all the eigenvalues of a matrix have magnitude less than one) of

$$H = \begin{bmatrix} e^{Ah} & \tilde{B} \\ Ce^{Ah} & C\tilde{B} + D \end{bmatrix},$$

where $\tilde{B} = \int_0^h e^{A(h-s)} B\, ds \equiv E(h)B$.

Theorem 3 [17, Corollary 14]: If H is Schur, then the zeroth solution of (13) is asymptotically stable.

We now apply this to an NCS. Using the NCS model in (8) and referring to the timing of the signals shown in Fig. 5, the following is suitable for our analysis:

$$\begin{aligned} \dot{x}(t) &= Ax(t) - BK\hat{x}(t), \quad t \in [kh+\tau, (k+1)h+\tau), \\ \hat{x}(t^+) &= x(t-\tau), \quad t \in \{kh+\tau, k = 0,1,2,\ldots\}. \end{aligned} \tag{14}$$

Writing $x(t-\tau)$ in terms of $x(t)$ and $\hat{x}(t)$, we obtain

$$x(t-\tau) = e^{-A\tau}x(t) + e^{-A\tau}E(\tau)BK\hat{x}(t)$$

and

$$C = e^{-A\tau},$$
$$D = e^{-A\tau}E(\tau)BK.$$

Comparing this to (13), we obtain the following corollary.

Corollary 4: The stability of an NCS with constant delay reduces to examining the Schur-ness of

$$H = \begin{bmatrix} e^{Ah} & -E(h)BK \\ e^{A(h-\tau)} & -e^{-A\tau}(E(h)-E(\tau))BK \end{bmatrix}.$$

We can use this to recalculate the integrator example of (11). By setting $A=0$ and $B=1$ in (14), we find

$$H = \begin{bmatrix} 1 & -hK \\ 1 & -(h-\tau)K \end{bmatrix}.$$

For H to be Schur, τ must satisfy (12), which verifies Example 2.

Compensation for Network-Induced Delay

Sensor-to-controller delay, τ_{sc}, and controller-to-actuator delay, τ_{ca}, have different natures. Sensor-to-controller delay can be known when the controller uses the sensor's data to generate the control signal, provided the sensor and controller clocks are synchronized and the message is time stamped. Thus an estimator can be used to reconstruct an approximation to the undelayed plant state and make it available for the control calculation. Controller-to-actuator delay is different, however, in that the controller does not know how long it will take the control signal to reach the actuator; therefore, no exact correction can be made at the time of control calculation.

We present a method of estimating the undelayed plant state using time domain solutions of the plant state equations.

An NCS estimator should have two primary functions. One is to work as a conventional state estimator to estimate the full state of the plant using partial state measurements (i.e., the plant outputs); the other is to compensate for the sensor delay to make a more accurate estimate. Thus, two situations should be analyzed: systems with full-state feedback and those with partial-state feedback. The estimation of the plant state $x(kh+\tau_{sc,k})$ is based on the sensor measurement at time kh and the sensor-to-controller delay $\tau_{sc,k}$. Here we assume single-packet transmission and delay less than one sampling period (i.e., $\tau_{sc} < h$).

Full-State Feedback

With full-state feedback, the only task of the estimator is to compensate for the delay, τ_{sc}, to achieve a more accurate plant state at the time the control signal is calculated. Assuming the plant and controller models are given by (6) and (7), this can be done using

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-s)}Bu(s)ds. \tag{15}$$

The estimation scheme is illustrated in Fig. 8. There, $\tau_{sc,k}$ denotes the sensor-to-controller delay for plant state $x(kh)$, and $\bar{x}(kh+\tau_{sc,k})$ denotes the plant state estimate at the time $x(kh)$ is received. Assuming there is no measurement noise, $\bar{x}(kh+\tau_{sc,k})$ can be calculated by

$$\begin{aligned}\bar{x}(kh+\tau_{sc,k}) &= x(kh+\tau_{sc,k}) \\ &= e^{A\tau_{sc,k}}x(kh) + \int_{kh}^{kh+\tau_{sc,k}} e^{A(kh+\tau_{sc,k}-s)}Bu(s)ds\end{aligned} \tag{16}$$

and the control law is computed by

$$u(kh+\tau_{sc,k}) = -K\bar{x}(kh+\tau_{sc,k}). \tag{17}$$

Using this control law, the closed-loop system becomes

$$x\big((k+1)h+\tau_{k+1}\big) = \tilde{\Phi}(\delta_k)x\big(kh+\tau_k\big) \tag{18}$$

where

$$\delta_k = h + \tau_{sc,k+1} - \tau_{sc,k},$$
$$\tilde{\Phi}(\delta_k) = \Phi(\delta_k) - \Gamma(\delta_k)K,$$
$$\Phi(\delta_k) = e^{A\delta_k},$$
$$\Gamma(\delta_k) = \int_0^{\delta_k} e^{As}B\,ds.$$

Output Feedback

When full-state information is not available for calculation of the control signal, a state estimator is built to estimate the plant state. A conventional current-state estimator esti-

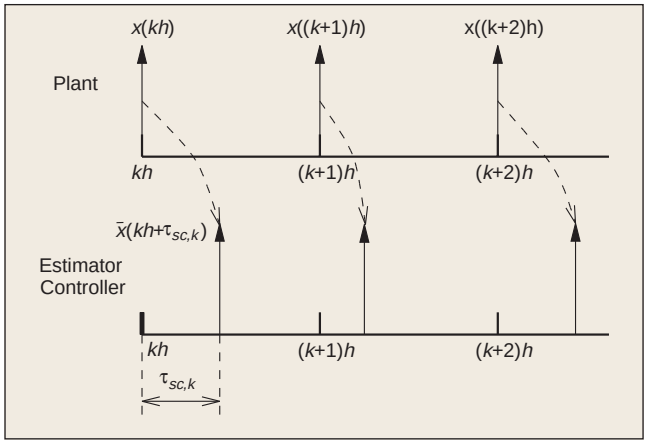

Figure 8. *Plant and estimator timing diagram with full-state feedback.*

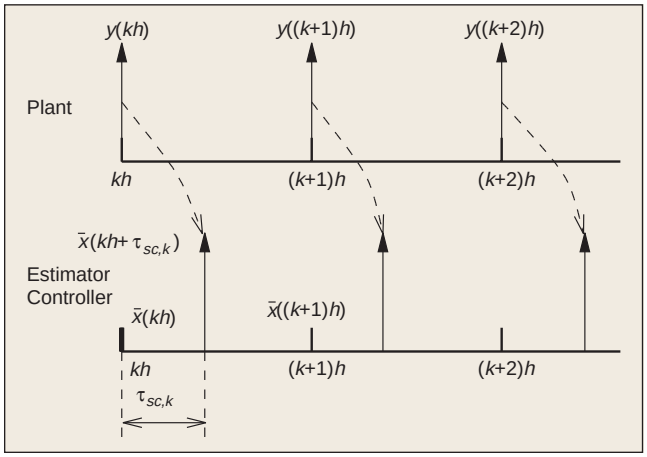

Figure 9. *Plant and estimator timing diagram with output feedback.*

mates the plant state $x(kh)$ using the plant output $y(kh) = Cx(kh)$, as described in [15]:

$$\hat{x}((k+1)h) = \Phi\bar{x}(kh) + \Gamma u(kh) \tag{19}$$

$$\bar{x}((k+1)h) = \hat{x}((k+1)h) + L_c(y((k+1)h) - C\hat{x}((k+1)h)) \tag{20}$$

where L_c denotes the current estimator gain. The calculation is done in two steps; first, the estimator state $\bar{x}(kh)$ is projected forward by one sampling period to obtain $\hat{x}((k+1)h)$, and then $\hat{x}((k+1)h)$ is corrected based on the plant output received.

The estimator with sensor measurement delay is based on the current-state estimator. Fig. 9 shows how the estimation is carried out. The estimation scheme at time kh is as follows:

1) Correction based on $y(kh)$:

$$\bar{x}(kh) = \hat{x}(kh) + L_c\big(y(kh) - C\hat{x}(kh)\big) \tag{21}$$

2) Forward to $kh + \tau_{sc,k}$:

$$\bar{x}(kh+\tau_{sc,k}) = e^{A\tau_{sc,k}}\bar{x}(kh) + \int_{kh}^{kh+\tau_{sc,k}} e^{A(kh+\tau_{sc,k}-s)}Bu(s)ds \tag{22}$$

3) Calculate control law:

$$u(kh+\tau_{sc,k}) = -K\bar{x}(kh+\tau_{sc,k}) \tag{23}$$

4) Forward to $(k+1)h$:

$$\hat{x}((k+1)h) = e^{A(h-\tau_{sc,k})}\bar{x}(kh+\tau_{sc,k}) + \int_{kh+\tau_{sc,k}}^{(k+1)h} e^{A((k+1)h-s)}Bu(s)ds. \tag{24}$$

Remark 5: The separation principle holds for the estimation scheme described by (21)-(24). Let $z(kh+\tau_k) = [x^T(kh+\tau_k), \tilde{x}^T(kh+\tau_k)]^T$, where $\tilde{x}(kh+\tau_k)$ is the estimation error and is defined as

$$\tilde{x}\big(kh+\tau_k\big) = \bar{x}\big(kh+\tau_k\big) - x\big(kh+\tau_k\big). \tag{25}$$

Using the notation defined in (18), the closed-loop system with the estimator is

$$z\big((k+1)h+\tau_{k+1}\big) = \tilde{\Phi}(\delta_k)z(kh+\tau_k) \tag{26}$$

where

$$\tilde{\Phi}(\delta_k) = \begin{bmatrix} \Phi(\delta_k) - \Gamma(\delta_k)K & -\Gamma(\delta_k)K \\ 0 & \Phi(\delta_k) - L_cH\Phi(\delta_k) \end{bmatrix}.$$

Proof: See the Appendix.

We can see that the separation principle holds for the estimation scheme. Thus the plant and the estimator can be designed separately, and we can guarantee the stability of both.

Control Experiments over a Physical Network

Setup

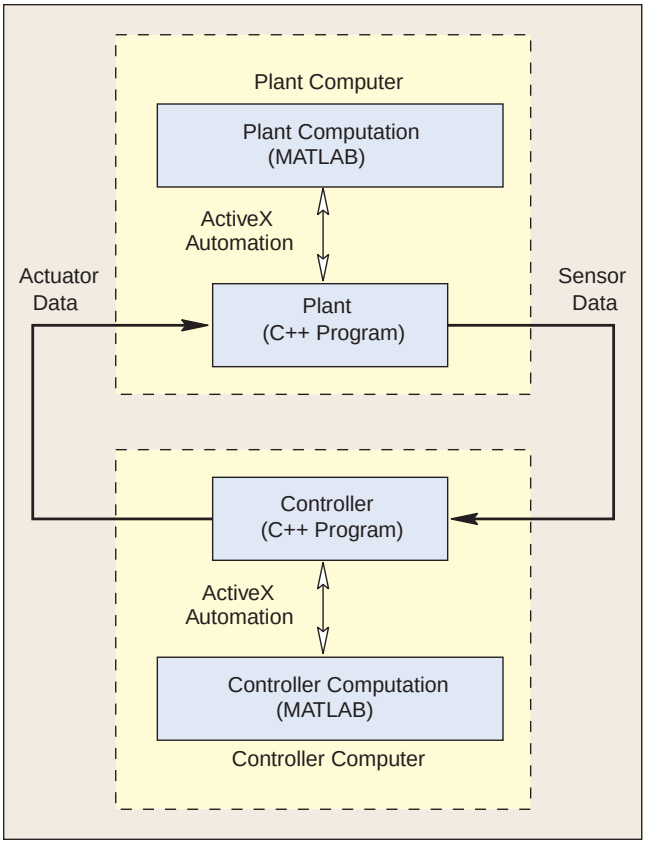

Figure 10. *Experimental setup.*

To show the influence of networks on control system performance and to test the compensation and control strategies, control experiments over a physical network were performed. This experiment allows real network traffic to be involved in the feedback control of the plant. The experimental setup is shown in Fig. 10.

For our experiments, we use the Case Western Reserve University campus-wide network (CWRUnet), which is a wide-area network containing both Ethernet and ATM at the physical layer. Communication between nodes is done using TCP/IP sockets (at the transport and network layer, respectively). TCP/IP sockets provide reliable transmission of data packets, regardless of possible collisions that might happen on the physical transmission medium. Therefore, TCP/IP will result in packet delay but not packet loss.

In our control setup, two computers, working as plant and controller, respectively, are connected over CWRUnet. Each computer runs a Visual C++ program as the user interface for setting up the sockets between them and accepting various configuration parameters, such as sampling period, control with estimation, and clock synchronization. The computation for simulating the plant and controller is carried out using MATLAB; that is, MATLAB works as a computation engine for each program on its own computer. MATLAB is invoked as an ActiveX automation server. On the plant computer, the C++ program obtains the control signal from the network, passes it to MATLAB for plant state and output calculation, and then sends the plant output to the controller computer. On the controller computer, the C++ program obtains the plant output from the network, uses MATLAB to calculate the control signal, and sends the signal to the plant computer. In this way, sensor data and control data are passed on the network, along with other campus network traffic, and may experience collision or delay.

For a detailed description of an alternate experimental testbed for controlling systems over the Internet, see [18].

Clock Synchronization

In the experiment, every message sent out by the plant and controller is time stamped. To calculate the delay accurately, plant and controller clocks must be synchronized.

Clock synchronization can be achieved in several ways, such as software synchronization, hardware synchronization, or a combination of the two. Clock synchronization in our experiment has been done as in [19]. Suppose the controller wants to synchronize its clock with the plant. It sends a message to the plant, and the plant will send its clock reading back to the controller. The controller records the clock offset and the round-trip time. The measurement is carried out many times, and the controller uses the clock offset with the minimum round-trip time.

An Example

The following example illustrates the effectiveness of the compensation scheme described above.

Example 4: Consider the state-space plant model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 5 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

A state feedback controller is $u = -Kx$, where $K = [25, 10]$ (closed-loop poles at $-5 \pm 10j$).

The plant and controller are implemented in two MATLAB M-files that run on two computers, as described above. The plant state $x(t)$ and the control signal $u(t)$ are sent using TCP/IP sockets. A scaled step response using full-state feedback is shown in Fig. 11, with comparison to the nonnetworked sampled data system. Network-induced delay causes the closed-loop plant to be underdamped, resulting in larger overshoot in the step response. After using

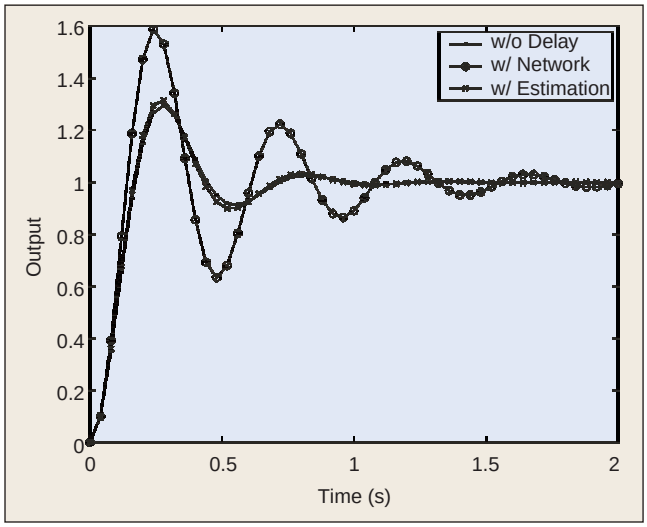

Figure 11. *Comparison of scaled step responses with full-state feedback.*

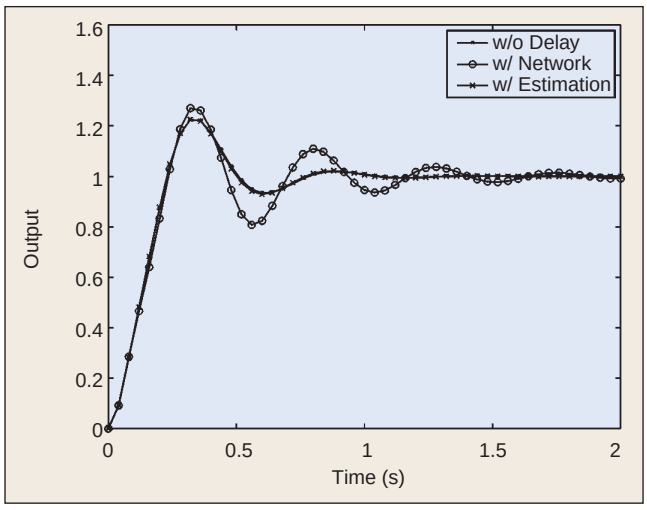

Figure 12. *Comparison of scaled step responses with unmodeled nonlinearity.*

delay compensation, the response is very similar to the nonnetworked system. Indeed, the difference is caused by the controller-to-actuator delay, which is not compensated by this scheme.

This linear estimator also works well when there is an unmodeled nonlinearity in the plant model. For example, consider the plant

$$\dot{x}_1 = 5\sin(x_2),$$
$$\dot{x}_2 = u.$$

The experimental result, shown in Fig. 12, illustrates that the compensation scheme still works well.

Stability of NCS with Data Packet Dropout

When using an NCS, one must consider not only network-induced delay, but also data packet dropout. Networks can be viewed as unreliable data transmission paths, where packet collision and network node failure occasionally occur. When there is a packet collision, instead of repeated retransmission attempts, it might be advantageous to drop the old packet and transmit a new one. Thus it is valuable to analyze the rate (percentage successful) at which the data should be transmitted to achieve the desired performance (stability).

An NCS with data packet dropout can be modeled as an asynchronous dynamical system (ADS) with rate constraints on events. The stability of this type of system is studied in [11]. We will extend a result therein to NCSs.

ADSs with Rate Constraints

ADSs, like hybrid systems, are systems that incorporate continuous and discrete dynamics. The continuous dynamics are governed by differential or difference equations, whereas the discrete dynamics are governed by finite automata that are driven asynchronously by external discrete events with fixed rates [11].

We consider a simplified ADS with rate constraints that can be described by a set of difference equations

$$x(k+1) = f_s(x(k)), \quad s = 1,2,\ldots,N,$$

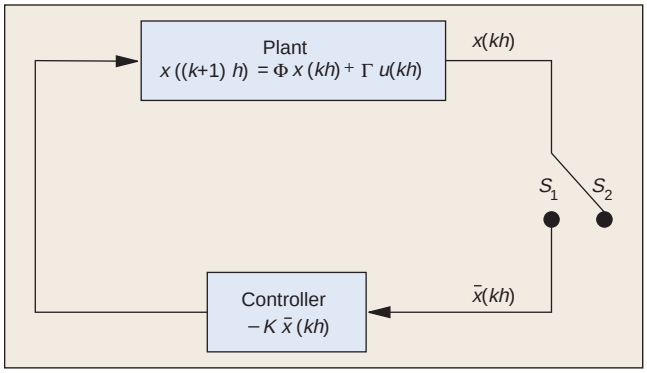

Figure 13. *NCS with data packet dropout.*

with continuous-valued state $x(k) \in \mathbb{R}^n$. Here, $1,2,\ldots,N$ represents the set of discrete states, which has a corresponding set of rates $r_1, r_2, \ldots, r_N$. These rates represent the fraction of time that each discrete state occurs; thus $\Sigma_{i=1}^{N} r_i = 1$.

The stability of such an ADS is given by the following theorem.

Theorem 6 [11]: Given an ADS as defined above. If there exist a Lyapunov function $V(x(k)):\mathbb{R}^n \to \mathbb{R}_+$ and scalars $\alpha_1, \alpha_2, \ldots, \alpha_N$ corresponding to each rate such that

$$\alpha_1^{r_1}\alpha_2^{r_2}\cdots\alpha_N^{r_N} > \alpha > 1 \tag{27}$$

and

$$V(x(k+1)) - V(x(k)) \le \left(\alpha_s^{-2} - 1\right)V(x(k)), \quad s = 1,2,\ldots,N, \tag{28}$$

then the ADS remains exponentially stable, with decay rate greater than α.

Theorem 6 requires the ADS to be stable on the average. It does not require every difference equation of the ADS to be stable, but rather it guarantees the ADS to be stable on the whole. If the discrete state dynamics is given by $x((k+1)h) = \Phi_s x(kh)$ for $s = 1,2,\ldots,N$, the search for the Lyapunov function of type $V(x(kh)) = x^T(kh)Px(kh)$ and the scalars $\alpha_1, \alpha_2, \ldots, \alpha_N$ can be cast into a bilinear matrix inequality (BMI) problem [11]. Equations (27) and (28) can be rewritten as

$$r_1 \log\alpha_1 + r_2 \log\alpha_2 + \cdots + r_N \log\alpha_N > 0$$

and

$$\Phi_s^T P \Phi_s \le \alpha_s^{-2} P, \quad s = 1,2,\ldots,N.$$

This is a BMI problem in P and the $\log\alpha_i$s.

Modeling an NCS with Data Packet Dropout

Fig. 13 illustrates an NCS setup with the possibility of dropping data packets. Here we assume that the nonnetworked system is stable and the network is only inserted from the plant to the controller. The network can be modeled as a switch that closes at a certain rate r. When the switch is closed (position S_1), the network packet containing $x(kh)$ is transmitted, whereas when it is open (position S_2), the output of the switch is held at the previous value and the packet is lost. Thus the dynamics of the switch (state $\bar{x}$) can be modeled as

$$S_1: \bar{x}(kh) = x(kh),$$
$$S_2: \bar{x}(kh) = \bar{x}((k-1)h).$$

Let $z(kh) = [x^T(kh), \bar{x}^T(kh)]^T$ be the augmented state vector; the closed-loop system with the network packet dropout effect is represented by

$$z((k+1)h) = \tilde{\Phi}_s z(kh)$$

for $s = 1,2$. When the switch is in position S_1,

$$\tilde{\Phi}_1 = \begin{bmatrix} \Phi & -\Gamma K \\ \Phi & -\Gamma K \end{bmatrix};$$

when the switch is position S_2,

$$\tilde{\Phi}_2 = \begin{bmatrix} \Phi & -\Gamma K \\ 0 & I \end{bmatrix}.$$

Normally, a feedback control system can tolerate a certain amount of feedback data loss. The following corollary can be used to test the system stability for a certain rate of data loss.

Corollary 7: For the above system setup, assume the plant state $x(kh)$ is transmitted at the rate of r. If there exist a Lyapunov function $V(x(kh)) = x^T(kh)Px(kh)$ and scalars α_1 and α_2 such that

$$\begin{aligned} &\alpha_1^r \alpha_2^{1-r} > 1, \\ &\tilde{\Phi}_1^T P \tilde{\Phi}_1 \le \alpha_1^{-2} P, \\ &\tilde{\Phi}_2^T P \tilde{\Phi}_2 \le \alpha_2^{-2} P \end{aligned}$$

the system is still exponentially stable.

With the plant state being transmitted at rate r, the effective sampling period becomes $h_{\text{eff}} = h / r$. This suggests that the plant can be stabilized by a slower sampling rate. In other words, the result shows that when we have fast sampling, we can drop the samples at a certain rate to save network bandwidth and still provide a stable feedback control system.

Example 5: Consider the state-space plant model in Example 1. When the plant is sampled with a sampling period $h = 0.3$ s, we obtain

$$\Phi = \begin{bmatrix} 1.0 & 0.2955 \\ 0 & 0.9704 \end{bmatrix}, \quad \Gamma K = \begin{bmatrix} 0.0167 & 0.0512 \\ 0.1108 & 0.3399 \end{bmatrix},$$

and the closed-loop system $(\Phi - \Gamma K)$ is still stable with the continuous controller. With the setup shown in Fig. 13, and assuming the transmission rate $r = 0.7$, we solve the LMI problem [20], [21] in Corollary 7 to find

$$\alpha_1 = 1.1288, \quad \alpha_2 = 0.7552$$

and

$$P = \begin{bmatrix} 0.9210 & 0.9196 & -0.6578 & -0.5144 \\ 0.9196 & 2.4788 & -0.5232 & -1.6644 \\ -0.6578 & -0.5232 & 0.7003 & 0.6461 \\ -0.5144 & -1.6644 & 0.6461 & 2.0562 \end{bmatrix},$$

which proves the stability of the system. This means that when the plant state is sampled every 0.3 s, if 70% of the packets are delivered to the controller, we can still guarantee the stability of the feedback control system. The result shows an effective sampling period of $h_{\text{eff}} = h / r = 0.43$ s; in fact, the maximum stable constant sampling period for this system is 1.7 s. The comparison of step responses with packet dropouts is shown is Fig. 14. We can see that the step response with 70% packets transmitted is similar to the original system. A large difference can be seen when only 20% of packets are transmitted (but the system is still stable, as we prove below).

The setup in Fig. 13 has also been considered by Walsh et al. [3], who used a different approach to determine the MATI τ when the feedback loop is closed over the network. Using this conservative approach, the MATI τ of this example is 4.5×10^{-4} s, which will consume a lot of network bandwidth if it is implemented in a real application. The method presented here takes a probabilistic approach while guaranteeing the exponential stability of the NCS, and it only requires plant state to be transmitted at a certain rate. This reduces network traffic without sacrificing stability.

The lower the transmission rate, the less network bandwidth used. The next question would be, "What is the lower bound on transmission rate r that still guarantees the stability of the system?" Theorem 8 involves the bound on the transmission rate r for a stable NCS.

Theorem 8: Consider the setup of Fig. 13, assuming that the closed-loop system with no dropout is stable (i.e., $\Phi - \Gamma K$ is Schur).

- If the open-loop system (Φ) is marginally stable, then the system is exponentially stable for all $0 < r \le 1$.
- If the open-loop system is unstable, then the system is exponentially stable for all

$$\frac{1}{1 - \gamma_1 / \gamma_2} < r \le 1$$

where

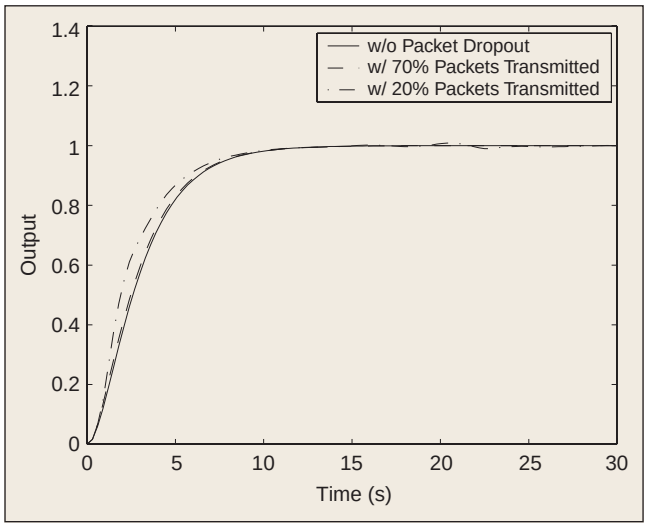

Figure 14. *Comparison of scaled step responses with packet dropouts.*

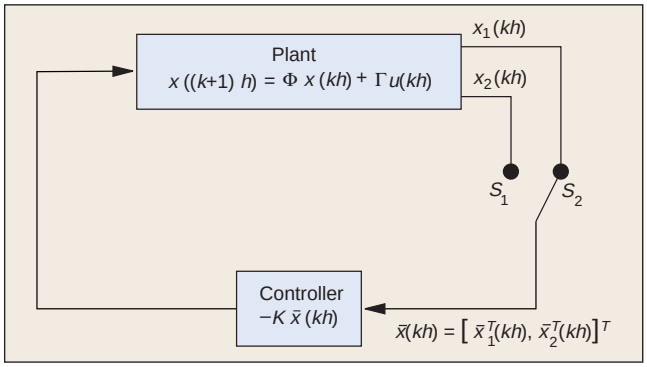

Figure 15. *NCS with multiple-packet transmission.*

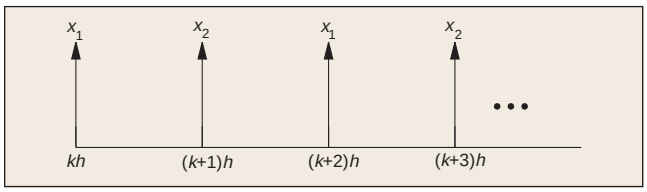

Figure 16. *Multiple-packet transmission with static scheduling.*

$$\gamma_1 = \log\left[\lambda^2_{\max}(\Phi - \Gamma K)\right], \quad \gamma_2 = \log\left[\lambda^2_{\max}(\Phi)\right].$$

Proof: See the Appendix.

Example 6: Example 5 has Φ marginally stable, so the networked controller is stable for all $r > 0$.

Modeling an NCS with Multiple-Packet Transmission

An NCS with multiple-packet transmission can also be modeled as an ADS. In multiple-packet transmission mode, plant state or output are split into separate packets. Fig. 15 illustrates a case where the plant state is transmitted in two packets. The dynamics of the network is given by

$$\begin{aligned} S_1: &\quad \bar{x}_1(kh) = x_1(kh), &\quad \bar{x}_2(kh) = \bar{x}_2((k-1)h), \\ S_2: &\quad \bar{x}_1(kh) = \bar{x}_1((k-1)h), &\quad \bar{x}_2(kh) = x_2(kh). \end{aligned}$$

Assume

$$x(kh) = [x_1^T(kh), x_2^T(kh)]^T, \quad \Phi = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix},$$

$$\Gamma = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \end{bmatrix}, \quad K = [K_1, K_2].$$

Let the augmented state be $z(kh) = \left[x_1^T(kh), x_2^T(kh), \bar{x}_1^T(kh), \bar{x}_2^T(kh)\right]^T$. We can now write the closed-loop system with two-packet transmission as

$$z((k+1)h) = \tilde{\Phi}_s z(kh)$$

for $s = 1,2$. When the switch is at S_1,

$$\tilde{\Phi}_1 = \begin{bmatrix} \Phi_{11} & \Phi_{12} & -\Gamma_1 K_1 & -\Gamma_1 K_2 \\ \Phi_{21} & \Phi_{22} & -\Gamma_2 K_1 & -\Gamma_2 K_2 \\ \Phi_{11} & \Phi_{12} & -\Gamma_1 K_1 & -\Gamma_1 K_2 \\ 0 & 0 & 0 & I \end{bmatrix},$$

whereas when the switch is at S_2,

$$\tilde{\Phi}_2 = \begin{bmatrix} \Phi_{11} & \Phi_{12} & -\Gamma_1 K_1 & -\Gamma_1 K_2 \\ \Phi_{21} & \Phi_{22} & -\Gamma_2 K_1 & -\Gamma_2 K_2 \\ 0 & 0 & I & 0 \\ \Phi_{21} & \Phi_{22} & -\Gamma_2 K_1 & -\Gamma_2 K_2 \end{bmatrix}.$$

The modeling can be easily extended to systems with more than two packets.

Stability in Scheduling Networks

We know that in scheduling networks, packets are sent out sequentially in a predetermined order, as shown in Fig. 16. The packet sequence received by the controller is $x_1 \to x_2 \to x_1 \to x_2 \to \cdots$.

Remark 9: If $\tilde{\Phi}_1 \cdot \tilde{\Phi}_2$ is Schur, the NCS with static scheduling of transmitting plant states is exponentially stable.

This shows that it is not necessary to ensure that both $\tilde{\Phi}_1$ and $\tilde{\Phi}_2$ are Schur. The NCS is stable if $\tilde{\Phi}_1 \cdot \tilde{\Phi}_2$ is Schur, as the following example illustrates.

Example 7: Consider the system in Example 5 with the plant state x_1, x_2 transmitted separately in two packets. With $h = 0.3$ s, we have

$$\tilde{\Phi}_1 = \begin{bmatrix} 1.0 & 0.2995 & -0.0167 & -0.0512 \\ 0 & 0.9704 & -0.1108 & -0.3399 \\ 1.0 & 0.2995 & -0.0167 & -0.0512 \\ 0 & 0 & 1.0 & 0 \end{bmatrix}$$

$$\tilde{\Phi}_2 = \begin{bmatrix} 1.0 & 0.2995 & -0.0167 & -0.0512 \\ 0 & 0.9704 & -0.1108 & -0.3399 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0.9704 & -0.1108 & -0.3399 \end{bmatrix}.$$

Neither $\tilde{\Phi}_1$ nor $\tilde{\Phi}_2$ is Schur; however, $\tilde{\Phi}_1 \cdot \tilde{\Phi}_2$ is Schur, which proves the stability of the system if a scheduling network is applied.

Using static scheduling on this two-packet setup, each packet is transmitted 50% of the time; thus the effective sampling period is $h_{\text{eff}} = h / 0.5 = 0.6$ s. The step response of this two-packet transmission setup is similar to the original system.

Conclusions

This article analyzed several fundamental issues in network control systems. One issue is the network-induced delay when transmitting sensor data and control data. Depending on the control network protocol employed, the delay can be either constant or time varying. The relationship between

the sampling rate h and the network-induced delay τ was captured using a stability region plot. Stability of an NCS was also characterized using a hybrid systems stability analysis technique. Methods to compensate network-induced delay using the time-domain solution of the plant model were discussed, and experimental results over a physical network (CWRUnet) were presented. We then modeled an NCS with packet dropout and multiple-packet transmission (which may occur due to the limitation of the control network) as an asynchronous dynamical system. We determined whether the NCS is stable at a certain rate of data loss, and we searched for the highest rate of data loss for the NCS to be stable.

References

[1] Y. Halevi and A. Ray, "Integrated communication and control systems: Part I-Analysis," *J. Dynamic Syst., Measure. Contr.*, vol. 110, pp. 367-373, Dec. 1988.

[2] J. Nilsson, "Real-time control systems with delays," Ph.D. dissertation, Dept. Automatic Control, Lund Institute of Technology, Lund, Sweden, January 1998.

[3] G.C. Walsh, H. Ye, and L. Bushnell, "Stability analysis of networked control systems," in *Proc. Amer. Control Conf.*, San Diego, CA, June 1999, pp. 2876-2880.

[4] M.S. Branicky, S.M. Phillips, and W. Zhang, "Stability of networked control systems: Explicit analysis of delay," in *Proc. Amer. Control Conf.*, Chicago, IL, June 2000, pp. 2352-2357.

[5] W. Lawrenz, *CAN System Engineering: From Theory to Practical Applications*. New York: Springer-Verlag, 1997.

[6] A.S. Tanenbaum, *Computer Networks*, 3rd ed. Upper Englewood Cliffs, NJ: Prentice-Hall, 1996.

[7] D. Anderson, *FireWire System Architecture*. Reading, MA: Addison-Wesley, 1998.

[8] E. Altman, T. Başar, and R. Srikant, "Congestion control as a stochastic control problem with action delays," *Automatica*, vol. 35, pp. 1937-1950, Dec. 1999.

[9] S. Mascolo, "Classical control theory for congestion avoidance in high-speed Internet," in *Proc. IEEE Conf. Decision and Control*, Phoenix, Dec. 1999, pp. 2709-2714.

[10] L. Dugard and E.I. Verriest, Eds., *Stability and Control of Time-Delay Systems* (Lecture Notes in Control and Information Sciences), vol. 228. Heidelberg, Germany: Springer-Verlag, 1997.

[11] A. Hassibi, S.P. Boyd, and J.P. How, "Control of asynchronous dynamical systems with rate constraints on events," in *Proc. IEEE Conf. Decision and Control*, Phoenix, AZ, Dec. 1999, pp. 1345-1351.

[12] J.D. Spragins, J.L. Hammond, and K. Pawlikowski, *Telecommunications: Protocols and Designs*. Reading, MA: Addison-Wesley, 1991.

[13] M. Tangemann and K. Sauer, "Performance analysis of the timed token protocol of FDDI and FDDI-II," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 271-278, Feb. 1991.

[14] K.J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[15] G.F. Franklin, J.D. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 3rd ed. Reading, MA: Addison Wesley Longman, 1997.

[16] M.S. Branicky, "Hybrid systems: Modeling, analysis, and control," Sc.D. dissertation, Dept. Electrical Engineering and Computer Science, Massachusetts Institute of Technol., Cambridge, MA, June 1995.

[17] M.S. Branicky, "Stability of hybrid systems: State of the art," in *Proc. IEEE Conf. Decision and Control*, San Diego, CA, Dec. 1997, pp. 120-125.

[18] J.W. Overstreet and A. Tzes, "An Internet-based real-time control engineering laboratory," *IEEE Contr. Syst. Mag.*, vol. 19, pp. 19-34, Oct. 1999.

[19] D. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, pp. 1482-1493, Oct. 1991.

[20] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia, PA: SIAM, 1994.

[21] P. Gahinet, A. Nemirovshi, A. Laub, and M. Chilali, *MATLAB LMI Control Toolbox*. Natick, MA: MathWorks, May 1995

[22] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1996.

Appendix

Proof (Corollary 2): Throughout, we use vector and matrix norm definitions and inequalities [22].

The Lyapunov function $V(x)$ must satisfy the following inequalities:

$$\lambda_1\|x\|^2 \le V(x) \le \lambda_2\|x\|^2$$
$$\dot{V}(x) = -X^TQX \le -\lambda_{\min}(Q)\|x\|^2.$$

Thus, following the proof of [3, Theorem 2], (8) of [3] should be rewritten as:

$$\dot{V}(x(\hat{t})) \le \|x(\hat{t})\|\left(-\lambda_{\min}(Q)\|x(\hat{t})\| + 2\lambda_2\|A\|\gamma_1\|z(t_0)\|\right).$$

Because of the choice of τ,

$$\gamma_1 < \frac{\lambda_{\min}(Q)}{8\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|} \le \frac{\lambda_{\min}(Q)}{8\lambda_2\|A\|}.$$

The same result as in [3] can then be obtained: for all $t > t_0 + p\tau, V(x(t)) < \gamma_2\|z(t_0)\|^2$. Setting

$$\gamma_1 < \min\left\{\frac{1}{4}, \frac{\lambda_{\min}(Q)}{8\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|}\right\}$$

we still have $V(x(t)) < \gamma_2\|z(t_0)\|^2, \|e(t)\| < \gamma_1\|z(t_0)\|$ for all $t > t_0 + p\tau$.

Viewing the NCS as perturbed by the bounded error signal $e(t)$, consider the system

$$\dot{x}_z(t) = A_{11}x_z(t) + A_{12}e(t)$$

starting at $t = t_0 + p\tau$ with zero initial condition $x(t_0 + p\tau) = 0$. We can conclude that for all $t > t_0 + p\tau$,

$$V_z(x_z(t)) < \frac{4\lambda_2^3\|A\|^2\gamma_1^2\|z_0(t)\|^2}{\lambda_{\min}^2(Q)}$$
$$\|x_z(t)\| < \frac{2\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|\gamma_1\|z(t_0)\|}{\lambda_{\min}(Q)}$$

and, by the choice of γ_1 above, $\|x_z(t)\| < (1/4)\|z(t_0)\|$. The rest of the proof for this part continues as in [3].

We now proceed to prove that the third bound is always the smallest among the three terms above. We first prove

$$\frac{1}{8\|A\|(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i} < \frac{\ln(2)}{p\|A\|}.$$

We know that $\lambda_2/\lambda_1 \ge 1, \sum_{i=1}^{p} i = p(p+1)/2$ and $p \ge 1$, hence

$$\frac{1}{8\|A\|(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i} \le \frac{1}{8\|A\|p(p+1)} < \frac{1}{8p\|A\|}.$$

Since ln(2) is greater than 0.125,

$$\frac{\ln(2)}{p\|A\|} > \frac{1}{8p\|A\|}$$

and the result follows. We then can prove

$$\frac{\lambda_{\min}(Q)}{16\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|^2(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i} \le \frac{1}{8\|A\|(\sqrt{\lambda_2/\lambda_1}+1)\sum_{i=1}^{p} i}.$$

Canceling common terms, we must show

$$\frac{\lambda_{\min}(Q)}{2\lambda_2\sqrt{\lambda_2/\lambda_1}\|A\|} \le 1.$$

Since $\sqrt{\lambda_2/\lambda_1} \ge 1$ by definition, it is enough to show

$$\frac{\lambda_{\min}(Q)}{2\lambda_2\|A\|} \le 1.$$

Taking norms on both sides of (3), we obtain

$$\begin{aligned}\|-Q\| &= \left\|A_{11}^T P + PA_{11}\right\| \\ &\le \left\|A_{11}^T\right\| \ \|P\| + \|P\| \ \|A_{11}\| \\ &= 2\|P\| \ \|A_{11}\|\end{aligned}$$

where the inequality follows from the triangle inequality plus the submultiplicative property of the matrix 2-norm. Now, since P is positive-definite symmetric, $\|P\| = \lambda_{\max}(P) = \lambda_2$. Also, $\|A\| \ge \|A_{11}\|$ follows easily from the definition of the induced norm, since the latter is a submatrix of the former (cf. (2)). Therefore,

$$\lambda_{\min}(Q) \le \|-Q\| \le 2\lambda_2\|A\|$$

and

$$\frac{\lambda_{\min}(Q)}{2\lambda_2\|A\|} \le 1.$$

Q.E.D.

Proof (Corollary 5): The plant model is given by

$$x\left((k+1)h + \tau_{sc,k+1}\right) = \Phi(\delta_k)x(kh + \tau_{sc,k}) + \Gamma(\delta_k)u(kh + \tau_{sc,k}),$$

where δ_k, $\Phi(\delta_k)$, and $\Gamma(\delta_k)$ are given (18).

The estimation scheme is given by (21)-(24); from (21) and (24) we have

$$\begin{aligned}\bar{x}((k+1)h) = {} & e^{A(h-\tau_{sc,k})}\bar{x}(kh + \tau_{sc,k}) \\ & + \int_0^{h-\tau_{sc,k}} e^{As}B\,ds\,u(kh + \tau_{sc,k}) \\ & + L_c H e^{A(h-\tau_{sc,k})}\left(x(kh + \tau_{sc,k}) - \bar{x}(kh + \tau_{sc,k})\right)\end{aligned}$$

and from (22) we have

$$\begin{aligned}\bar{x}\left((k+1)h + \tau_{sc,k+1}\right) = {} & \Phi(\delta_k)\bar{x}(kh + \tau_{sc,k}) \\ & + \Gamma(\delta_k)u(kh + \tau_{sc,k}) \\ & + L_c H\Phi(\delta_k)\left(x(kh + \tau_{sc,k}) \right. \\ & \left. - \bar{x}(kh + \tau_{sc,k})\right).\end{aligned}$$

The estimation error is defined in (25), and the error equation is

$$\tilde{x}\left((k+1)h + \tau_{sc,k+1}\right) = \left(\Phi(\delta_k) - L_c H\Phi(\delta_k)\right)\tilde{x}\left(kh + \tau_{sc,k}\right). \tag{29}$$

Now apply the control law to form the closed-loop system

$$u(kh + \tau_{sc,k}) = -K\bar{x}(kh + \tau_{sc,k}).$$

The closed-loop system is

$$\begin{aligned}x\left((k+1)h + \tau_{sc,k+1}\right) &= \Phi(\delta_k)x\left(kh + \tau_{sc,k}\right) \\ &\quad - \Gamma(\delta_k)K\bar{x}\left(kh + \tau_{sc,k}\right) \\ &= \left(\Phi(\delta_k) - \Gamma(\delta_k)K\right)x\left(kh + \tau_{sc,k}\right) \\ &\quad - \Gamma(\delta_k)K\tilde{x}\left(kh + \tau_{sc,k}\right).\end{aligned} \tag{30}$$

Combining (29) and (30), we have (26).

Q.E.D.

Proof (Theorem 8): Consider the setup of Corollary 7, and define $\beta_i = \alpha_i^{-2}$ for $i = 1,2$. Substituting and taking logs, if the transmission rate r satisfies

$$\frac{\log\beta_2}{\log\beta_2 - \log\beta_1} < r \le 1 \tag{31}$$

where $\beta_1 < 1$ and $\beta_2 > \beta_1$ are positive constants and P is a positive-definite symmetric matrix such that

$$\begin{aligned}\tilde{\Phi}_1^T P\tilde{\Phi}_1 &\le \beta_1 P, \\ \tilde{\Phi}_2^T P\tilde{\Phi}_2 &\le \beta_2 P\end{aligned}$$

the system is exponentially stable.

Transmission rate r depends on β_1 and β_2, and the choice of β_1, β_2 must satisfy $\beta_1 \ge \lambda_{\max}^2(\tilde{\Phi}_1)$ and $\beta_2 \ge \lambda_{\max}^2(\tilde{\Phi}_2)$.

Looking at (31), it makes sense to minimize both $\log\beta_2$ and $\log\beta_1 < 0$ to obtain the weakest lower bound on r possible from that equation.

Now note that

$$\lambda_{\max}^2\left(\tilde{\Phi}_1\right) = \lambda_{\max}^2(\Phi - \Gamma K),$$

since the two matrices share the same spectrum (although each eigenvalue of the former has twice the multiplicity of the latter). Also note that the block-diagonal structure of $\tilde{\Phi}_2$ implies

$$\lambda_{\max}^2\left(\tilde{\Phi}_2\right) = \max\left\{1, \lambda_{\max}^2(\Phi)\right\},$$

with one achieving the maximum if and only if Φ is stable. The theorem is now seen to easily follow.

Q.E.D.

Wei Zhang received his B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, China, in 1993 and 1996, respectively. He then worked for the Industrial Automation and Control Division of Honeywell (Tianjin) Ltd. as a Systems Engineer from 1996 to 1997. He is now pursuing his Ph.D. degree in electrical engineering and computer science at Case Western Reserve University. His research interests include the modeling, analysis, and design of networked control systems.

Michael S. Branicky received the B.S. (1987) and M.S. (1990) degrees in electrical engineering and applied physics from Case Western Reserve University (CWRU). In 1995, he received his Sc.D. in electrical engineering and computer science from the Massachusetts Institute of Technology. In 1997, he rejoined CWRU as an Assistant Professor of Electrical Engineering and Computer Science. He has held research positions at MIT's AI Lab, Wright-Patterson AFB, NASA Ames, Siemens Corporate Research (Munich), and Lund Institute of Technology's Dept. of Automatic Control. His research interests include hybrid systems, intelligent control, and learning, with applications to robotics, flexible manufacturing, and control over networks.

Stephen M. Phillips received the B.S. degree with distinction in electrical engineering from Cornell University in 1984 and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1985 and 1988, respectively. He joined the faculty of Case Western Reserve University in 1988, where he is currently Associate Professor in the Department of Electrical Engineering and Computer Science. He serves as Director of the Center for Automation and Intelligent Systems and is a registered professional engineer. His research interests include sampled-data control, system identification, and adaptive control, with applications to manufacturing, aerospace, and microelectromechanical systems.