IECON'01: The 27th Annual Conference of the IEEE Industrial Electronics Society

Network-Based Control Systems: A Tutorial

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

***Abstract*—For many years now, data networking technologies have been widely applied in the control of industrial and military applications. These applications include manufacturing plants, automobiles, and aircrafts. Connecting the control system components in these applications, such as sensors, controllers, and actuators, via a network can effectively reduce the complexity of the systems with nominal economical investments. Furthermore, the applications connected through a network can be remotely controlled from a long-distance source. Traditionally, the networks used in the aforementioned applications are specific industrial networks, such as CAN (Controller Area Networks), and PROFIBUS. However, general data networks such as Ethernet and Internet are rapidly advancing to be the networks of choices for many applications due to their flexibility and lower costs.**

There are two general structures to design a control system through a network. The first structure is to have several subsystems, in which each of the subsystem contains a set of sensors, a set of actuators, and a controller by itself. These system components are attached to the same control plant. In this case, a subsystem controller receives a set point from the central controller. Another structure is to connect a set of sensors and a set of actuators to a network directly. Sensors and actuators in this case are attached to a plant, while a controller is separated from the plant via a network connection to perform a closed-loop control over the network.

Both structures have different advantages. The first structure is more modular. A control loop is simpler to be reconfigured. The second structure has better interaction because data are transmitted to components directly. A controller in the second structure can observe and process every measurement, whereas a (central) controller in the first structure may have to wait until the set point is satisfied to transfer the complete measurements, status signals, or alarm signals. A control system in the second structure is so-called *networked control system* or *network-based control system* depending on different authors' preference.

A challenging problem in control of networked-based system is *network delay effects*. The time to read a sensor measurement and to send a control signal to an actuator through the network depends on network characteristics such as their topologies, routing schemes, etc. Therefore, the overall performance of a network-based control system can be significantly affected by network delays. The severity of the delay problem is aggravated when data loss occurs during a transmission. Moreover, the delays do not only degrade the performance of a network-based control system, but also can destabilize the system.

This tutorial presents fundamental details of network-based control and recent network-based control techniques for handling the network delays. The techniques are based on various concepts such as state augmentation, queuing and probability theory, nonlinear control and perturbation theory, and scheduling. A general structure of a network-based control system, delay types, and delay behaviors are also described in this tutorial. In addition, advantages and disadvantages of these techniques are discussed.

Index Terms—**Delay, time constraint, network-based control, networked control.**

The authors are with the Advance Diagnosis and Control Laboratory (ADAC), Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27606 USA (email: chow@eos.ncsu.edu or ytipsuw@unity.ncsu.edu).

I. Introduction

Data networks play important roles not only in the areas of communication and computing, but also in many control applications. In these applications, networks are used as mediums to transfer control data, such as control signals, alarm signals and sensor measurements, for different purposes. In general, two major types of control systems that utilize networks are *1)* complex control systems and *2)* remote control systems.

1) Complex control systems

A complex control system is a large-scale system containing several subsystems [1]. A subsystem by itself can be thought of as a control system with three system components: *sensor, actuator,* and *controller.* In many cases, sensors and actuators are attached to a plant at different locations from the controller. An example of a complex control system is an altitude control system of an airplane [37], as shown in Fig. 1.

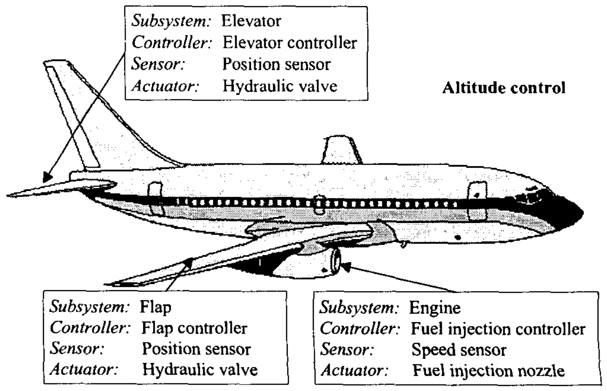

Subsystem: Elevator
Controller: Elevator controller
Sensor: Position sensor
Actuator: Hydraulic valve
Altitude control
Subsystem: Flap
Controller: Flap controller
Sensor: Position sensor
Actuator: Hydraulic valve
Subsystem: Engine
Controller: Fuel injection controller
Sensor: Speed sensor
Actuator: Fuel injection nozzle

Fig 1. Altitude control system of an airplane.

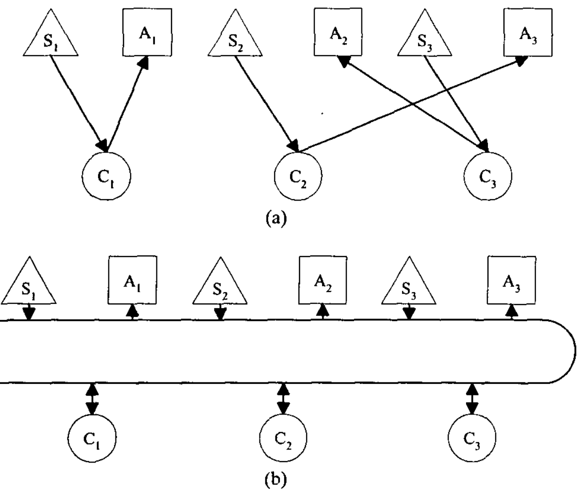


Fig. 2. Comparison between (a) Direct connections and (b) Shared-network connections: S_i: the ith set of sensors, A_i: the ith set of actuators, and C_i: the ith controller.

Since an actual complex system is typically large, connecting a large number of sensors, actuators, and controllers directly by electrical wires results in complicated circuits. These connections cause the whole system to be difficult to install and maintain. For example, consider Fig. 2(a), in which the numbers indicate the index of the subsystem and the arrows show the direction of data flow between two components. If there was only one single subsystem, such as the system shown at the left of Fig. 2(a), then the connections would be very simple. However, the system complexity increases when the complex system has more than one subsystem, as shown at the right of Fig. 2(a). The connections in a complex system are even more complicated if the system components are not physically closely located.

Using a shared-network connection to transfer sensor measurements to controllers and control signals from controllers to actuators can greatly reduce the complexity of the connections [1]. As shown in Fig. 2(b), this method is much more systematic and structured than direct connections. It is also more flexible to install and easier to maintain. Furthermore, networks enable communication among control loops. This feature is extensively useful when a control loop exchanges information with other control loops to perform more sophisticated controls.

Similar complex system applications such as automobiles, aircraft, chemical processes, and manufacturing plants have successfully used networks for system control. Some examples of these applications in automobiles and industrial plants are published in [2-4]. In [2, 3], a network was applied to the active suspension control system of an automobile. The oxygen extraction plant in [4] utilized a network as well.

2) Remote control systems

Remote control systems, including remote data acquisition systems and remote monitoring systems [5], have been used for two general reasons: convenience and safety. For human operators, this type of system saves place-to-place traveling time and protects them from dangers in hazardous environments.

In order to implement a remote control system, communication between local and remote sites must be established. Currently, many remote control systems have their own specific communication connections. These connections are usually conservative and are limited to only one single platform. A simple example is an RC (Radio-Controlled) airplane. With a specific frequency to control one airplane, other airplanes cannot use the same frequency. Also, installation, maintenance, and expansion of this system are costly and time-consuming.

A more efficient way is to connect remote systems and controllers through a shared-medium network. Using this scheme, several remote control systems can operate with more utilization of the medium. Without extra modifications on a network, reconfiguration and expansion of remote control systems can be performed more easily. This scheme is more cost effective and extremely useful, especially when used on a readily available large network such as the Internet. Various examples of remote control systems on networks also include teleoperation [6, 9] and teaching labs [7, 8].

There are two general ways to design a control system via a network [7, 8]. The first way is to have several subsystems, and then form a *hierarchical structure*, in which each of the subsystems contains a set of sensors, a set of actuators, and a controller by itself, as depicted in Fig. 3 and with the same notation as in Fig. 2. These system components are attached to the same control plant. In this case, a subsystem controller receives a set point from the central controller C_M as shown in Fig. 4(a). The subsystem then tries to satisfy this set point by itself. The sensor data or status signal is then transmitted back via a network to the central controller.

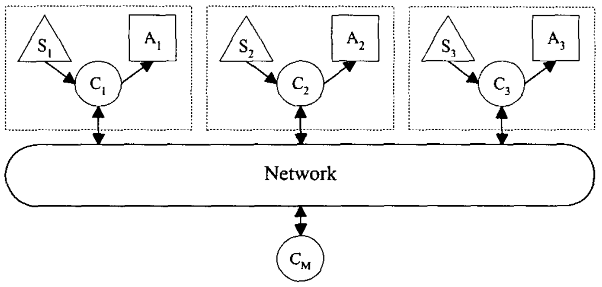


Fig. 3. Hierarchical structure.

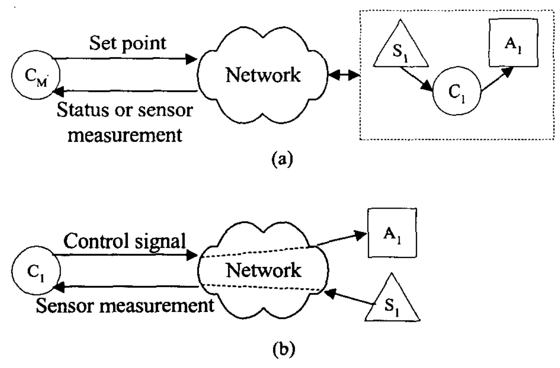


Fig. 4. Data transfers of (a) Hierarchical structure. (b) Direct structure.

The structure shown in Fig. 4(b) is denoted here as the *direct structure* because sensors and actuators of a control loop are connected to a network directly. In this case, a set of sensors and a set of actuators are attached to a plant, while a controller is separated from the plant via a network connection. The controller has to perform the following steps:

a. *Read sensor measurements via the network.*
b. *Compute control signals.*
c. *Send the control signals to the set of actuators through the network.*

Both structures have their own advantages. The hierarchical structure is more modular. A control loop is simpler to reconfigure. The direct structure has better interaction between system components because data are transmitted to components directly. A controller in the direct structure can observe and process every measurement, whereas a (central) controller in the hierarchical structure may have to wait until the set point is satisfied before transferring the complete measurements, status signals, or alarm signals. Examples of both structures are shown in a remote teaching lab [10]. A control system in the direct structure is known as a *network-based control system* or *networked control system*. These two terms are somewhat interchangeable depending on different authors' preferences [31-32].

A general problem of network-based control is the problem of *network delays*. Since the time to process step *a.* and step *c.* above depends on the characteristics of networks, such as their topologies and routing schemes, the overall performance of a network-based control system can be significantly affected by network delays. The severity of the delay problem is aggravated when data loss occurs during a network transmission. Moreover, the delays not only degrade the system performance, but can also destabilize the system.

This tutorial focuses on recent techniques for network-based control design on different kinds of networks. In order to describe these techniques, an overview of the structure of network-based control systems and the characteristics of their delays are first described in section II. Then, section III discusses various control techniques of the network-based systems in more details. Pros and cons of each technique are also briefly provided. The conclusion of this tutorial is given in section IV.

II. Overview of Network-Based Control Systems

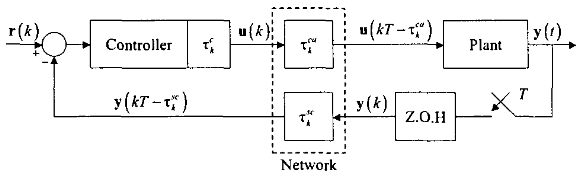
A. Structure of Network-Based Control Systems

As shown in Fig. 2(b), a network-based control system has three major system components: *controller*, *sensor*, and *actuator*. A shared-network acts as a medium to connect all these components together. This network can also be shared with other control loops and network resources. Each sensor and actuator is connected to a process or a plant. In general, one plant can connect to more than one sensor and one actuator. The message arrivals at a controller or an actuator are discrete events. As such, they cannot be easily handled by a continuous-time model. Therefore, a discrete-time model is more preferred.

A sensor, controller, and actuator can either be a time-driven device or an event-driven device. In a time-driven device, input reception or output transmission is controlled by a sampling time, which could be represented by a clock signal. On the other hand, an event-driven device starts to process immediately at the arrival (time) of a device input. Whether a system component should be time-driven or event-driven depends on the control techniques used. For example, in [17], time-driven sensors, event-driven controllers, and event-driven actuators are all used, whereas in [2] all devices are time-driven.

B. Delays in Network-Based Control Systems

Due to data transmissions and controller processing, there are network delays in a discrete-time network-based control loop, in addition to delays from a sampling period T. Major types of these delays, along with a block diagram of a discrete-time model, are depicted in Fig. 5. The timing diagram of delay generations is shown in Fig. 6. The three types of delays in Figs. 5 and 6, τ_k^{sc}, τ_k^{ca}, and τ_k^{c}, are described as follows.



$y(t)$	: Continuous system output
$r(k)\triangleq r(kT)$	: Discrete reference signal
$u(k)\triangleq u(kT)$	: Discrete control signal
$y(k)\triangleq y(kT)$	: Discrete system output

Fig. 5. Block diagram of a discrete-time model of a network-based control system.

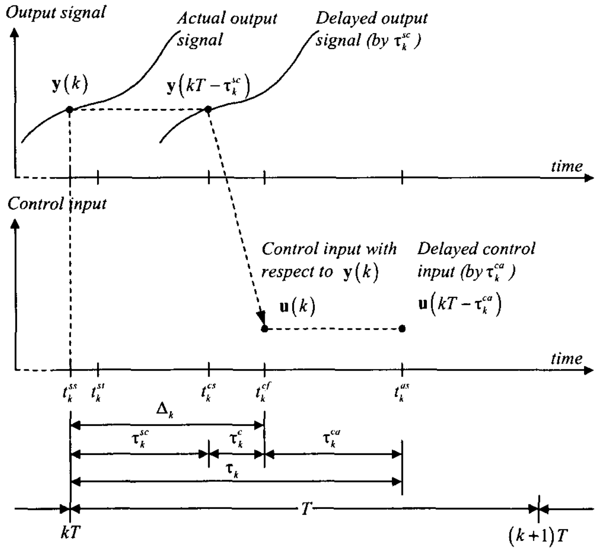


Fig. 6. Timing diagram of delay generations.

1) τ_k^{sc} : sensor-to-controller delay

This delay is generated when a sensor transmits a measurement to a controller. The sensor-to-controller delay at time index k is computed by:

$$\tau_k^{sc} = t_k^{cs} - t_k^{ss}, \tag{1}$$

where t_k^{cs} and t_k^{ss} are the time instants at which the controller starts to compute the control signal and the sensor starts to measure the system output, respectively.

2) τ_k^{c} : computational delay

Computational delay is the time needed for a controller to compute a control signal based on the received measurement. This delay is described by:

$$\tau_k^{c} = t_k^{cf} - t_k^{cs}, \tag{2}$$

where t_k^{cf} is the time instant when the controller finishes computing a control signal.

3) τ_k^{ca} : controller-to-actuator delay

This delay occurs when a controller sends a control signal to an actuator. The controller-to-actuator delay is defined as:

$$\tau_k^{ca} = t_k^{as} - t_k^{cf}, \tag{3}$$

where t_k^{as} is the time instant when the actuator receives the control signal and starts to operate. An actuator normally starts to perform its task immediately after it receives the control signal.

The sum of the delays in (1), (2) and (3) is referred to as the control delay τ_k [17]. The difference between the sampling time of the sensor and the sampling time of the controller is the *time skew* Δ_k [11].

C. Behaviors of Delays in Network-Based Control Systems

The behaviors of the delays in a network-based control system are described in this section.

1) Behaviors of computational delay τ_k^{c}

The computation delay is time-varying, but the variation is not substantial compared to the sensor-to-controller and controller-to-actuator delays if a controller is designed properly. The variation of τ_k^{c} can be caused by hardware or software [15]. Some hardware architectures inside a controller's processor, such as branch prediction and caches, can vary the execution time [14]. Likewise, software codes containing several conditional statements may result in the variation as well.

Time-variation of τ_k^{c} is unavoidable for commercial products, but it can be reduced by using appropriate hardware design schemes, real-time OS, and efficient software codes. Nevertheless, τ_k^{c} is typically very short compared to τ_k^{sc} and τ_k^{ca}. Therefore, this delay is insignificant in many control techniques and could be included in τ_k^{ca} [17].

2) Behaviors of sensor-to-controller delay τ_k^{sc} and controller-to-actuator delay τ_k^{ca}

In reality, τ_k^{sc} and τ_k^{ca} vary because of network protocol behaviors, as described below.

a. Cyclic service network

In a network in which data at each system component is transmitted in a cyclic order such as token passing (e.g., IEEE 802.4, SAE token bus, PROFIBUS, etc.), token ring (e.g., IEEE 802.5, SAE token ring), and polling (e.g., MIL-STD-1553B, FIP, etc.), the variation of τ_k^{sc} and τ_k^{ca} can be periodic and deterministic under the *error-free communication assumption* [16]. Nevertheless, the following detrimental factors can destroy the periodic property of networks.

i. Communication errors

Communication errors such as data loss in a transmission can influence τ_k^{sc} and τ_k^{ca}. When these errors happen, a data message may not arrive at a destination component before the next sampling time period occurs. This means that there is no measurement for computing the next control input. This phenomenon is called *vacant sampling* [11]. On the other hand, two data messages may arrive at the destination in the same sampling time period. In this case, only the most recent data message is acceptable for the controller and the previous data is then discarded. This problem is referred to as *message rejection* [11].

In fact, vacant sampling and message rejection are identical to *lost frame* and *duplicate detection* [36] in the networking area. Both vacant sampling and message rejection are illustrated in Fig. 7.

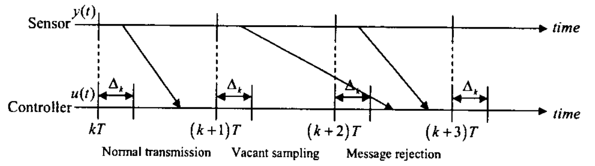


Fig. 7. Vacant sampling and message rejection in sensor-to-controller transmissions.

ii. Clock drift

System clocks in different components may have discrepancies in clock signal because there are differences in the material that gives clock signals. For this reason, τ_k^{sc}, τ_k^{ca}, and Δ_k become functions of time and time discrepancies in system components [17]. Clock drift also changes the probabilities of the occurrences of vacant sampling and message rejection [12]. System components can perform clock synchronization to reduce the clock drift problem [17].

b. Random access network

In random access networks such as CAN, Ethernet, and Internet, τ_k^{sc} and τ_k^{ca} are stochastic processes [18]. Thus, stochastic approaches may be required to model the behaviors of τ_k^{sc} and τ_k^{ca}. Some of modeling approaches are [17]:

i. Modeling τ_k^{sc} and τ_k^{ca} as independent transfer-to-transfer distributions

This scheme assumes that the current delay τ_{k-1} is dependent on the previous delay τ_k. This model is simple, but it is not realistic in practice because the previous transmission usually correlates to the current transmission. The probability distributions of τ_k^{sc} and τ_k^{ca} are dependent on the characteristics of network protocols and configurations. Modeling τ_k^{sc} and τ_k^{ca} of CAN in [17] is an example of this approach. The probability density functions of τ_k^{sc} and τ_k^{ca} in this case were approximated as:

$$f_\tau\left(\tau_k^{ca}\right)=\delta\left(\tau_k^{ca}-a\right)\cdot\left(1-p_{ca}\right)+\delta\left(\tau_k^{ca}-b\right)\cdot p_{ca}, \quad (4)$$

$$f_\tau\left(\tau_k^{sc}\right)=\begin{cases}\delta\left(\tau_k^{sc}-a\right)\cdot\left(1-p_{sc}\right), & \tau_k^{sc}=a\\ p_{sc}/\left(b-a^+\right), & \tau_k^{sc}\in(a,b],a<b\\ 0, & \tau_k^{sc}\notin[a,b]\end{cases} \quad (5)$$

where $\delta(\bullet)$ is the Dirac delta function, and $p_{sc}, p_{ca}\in[0,1]$, are parameters of the network.

ii. Modeling τ_k^{sc} and τ_k^{ca} *using a Markov chain*

This method utilizes a Markov chain to model the dependence between the previous delay τ_{k-1} and the current delay τ_k. States of the Markov chain are defined as different network load conditions. The probability distribution of τ_k^{sc} and τ_k^{ca} are then determined from the states of the Markov chain. A simple example of this model in [17] uses three network load states: *low* (*L*), *medium* (*M*), and *high* (*H*). The transition probability in this model is described as follows:

$$q_{ij}=P\left\{r_k=j\,|\,r_{k-1}=i\right\},\ i,j\in\{L,M,H\}, \quad (6)$$

where r is the state of the Markov chain. The network load states of this Markov chain are depicted in Fig. 8. The probability distributions of τ_k^{sc} and τ_k^{ca} are then obtained from the corresponding states. This scheme is more realistic than the previous modeling approach. Nevertheless, finding the Markov relation of a delay such as q_{ij} is a challenging task.

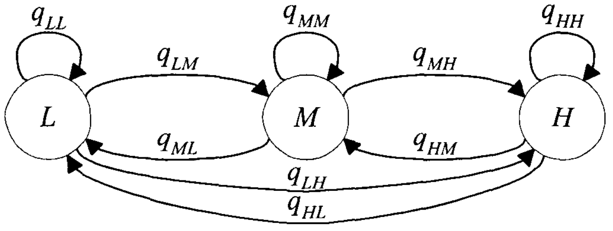


Fig. 8. An example of a Markov chain load modeling.

D. *Effects of Delays on Network-Based Control Systems*

General effects of delays on a network-based control system [19] are described as follows.

1) Poor system response

Generally, delays can affect the magnitude of the transient behaviors in a control system. An investigation of this delay effect on network-based control systems was done in [20, 21]. The results supported that longer network delays could increase the overshoot and settling time of the system. This study also showed that a large overhead in a packet could aggravate the control system performance.

2) Decreased stability margin

Delays can reduce the stability margin of a system and cause the system to be unstable. A technical example of the delay effect on a decreased stability margin can be seen in [22]. In this work, the authors examined the stability region of a first-order system. The results in this case indicated that a time-varying delay could decrease the stability region of the first-order system.

Because network-based control systems usually involve delays, stability analysis is one of the most important concerns in the analysis and control of network-based systems. Most network-based control techniques have been developed along with an analysis of stability. Nevertheless, analysis methods are usually specific to the control techniques used and may not be applicable to other control techniques due to different network setups and mathematical formulations.

III. Recent Control Techniques

A. *General Aspects*

Due to network delay concerns, the techniques used to control network-based systems have to maintain the performance and the stability of the systems in addition to controlling the plant. Various methods have been formulated differently based on multiple types of network behaviors and configurations in conjunction with several ways to treat the delay problems in network-based control systems. Several assumptions have been made in these approaches in order to develop generic network-based control methods. Some of these assumptions are:

a. *Network transmissions are error-free.*
b. *Every data message always has the same constant length.*
c. *The time skew* Δ_k *is constant.*
d. *The computational delay* τ_k^c *is constant and is much less than the sampling period T.*
e. *Network traffic cannot be overloaded.*
f. *Every dimension of output measurements or control inputs is packed into one single data message.*

Relaxation of some assumptions has been reported in some control techniques [12, 27]. On the other hand, some techniques may require additional assumptions [17, 16, 26, 28]. Some of these assumptions, such as *a.*, are obviously not practical. Hence, some control methods utilizing unrealistic assumptions may not be applicable for real life applications.

A widely used fundamental idea to formulate many network-based control methods is to use *state augmentation*. This approach has been used in discrete-time systems with constant delays for several years [23]. Several researchers adopted the state augmentation approach as the first step to developing their own techniques [11, 16, 24-26]. The main concept of state augmentation is to derive additional state-space equations to integrate with the original state-space equations of a system, as described in the following mathematical formulation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},t), \mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathbb{R}^m, \quad (7)$$

$$\dot{\mathbf{v}} = \mathbf{g}(\mathbf{x},\mathbf{u},\mathbf{v},t), \mathbf{v} \in \mathbb{R}^q, \quad (8)$$

where $\mathbf{x}$ is the state vector of the original system, $\mathbf{u}$ is the system input vector, t is time, $\mathbf{v}$ is the augmented state vector, and $\mathbf{f}(\bullet)$ and $\mathbf{g}(\bullet)$ are the functions representing the dynamics of the system and augmented states, respectively.

The combination of (7) and (8) forms the new state-space equation as:

$$\dot{\mathbf{s}} = \mathbf{f}_g(\mathbf{s},\mathbf{u},t), \mathbf{s} \in \mathbb{R}^{n+q}, \quad (9)$$

where $\mathbf{s} = \left[\mathbf{x}^T, \mathbf{v}^T\right]^T$, and $\mathbf{f}_g(\bullet)$ is the augmented dynamics equations of original and additional states.

The augmented state vector $\mathbf{v}$ can be obtained in different ways depending on different aspects of network-based control systems. In addition to the state augmentation approach, several other techniques have been developed to handle network delays [1, 17, 16, 26].

B. Augmented Deterministic Discrete-time Model Method

Halevi and Ray proposed a method to control network-based systems on a cyclic service network in [11]. In this method, a controller and a sensor are time-driven, whereas an actuator is event-driven. This approach can be applied to a linear plant described by the standard state-space equations as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \quad (10)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}, \quad (11)$$

where $\{\mathbf{A},\mathbf{B},\mathbf{C}\}$ are the realization of the system in (10) and (11).

This linear plant can be converted to the discrete-time equations [23] under the assumption that $\mathbf{u}$ is piecewise constant during each sampling time T as follows:

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{u}(k), \quad (12)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k), \quad (13)$$

where $\mathbf{\Phi} = \exp(\mathbf{A}T)$, and $\mathbf{\Gamma} = \int_0^T \exp(\mathbf{A}\zeta)d\zeta\mathbf{B}$.

A linear controller is used in this approach. In the case of a zero value set point, the controller dynamics can be described by:

$$\xi(k+1) = \mathbf{F}\xi(k) - \mathbf{G}\mathbf{z}(k), \quad (14)$$

$$\mathbf{u}(k) = \mathbf{H}\xi(k) - \mathbf{J}\mathbf{z}(k), \quad (15)$$

where ξ is the controller state vector, $\mathbf{z}(k)$ is the last available measurement at the instant when $\mathbf{u}(k)$ is processed by the controller (i.e., $\mathbf{z}(k) = \mathbf{y}(k-i), i = \{1,\dots,j\}$), and $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{J}$ are constant matrices describing the dynamics of the controller. The control input $\mathbf{u}$ for the plant in (12) is the output of this controller.

The main method used to handle network delay problems in this approach is to combine and rearrange (12)-(15) into an augmented state-space equation in the following form for at most $(l+1)$ different values of $\mathbf{u}$ in each sampling period T:

$$\mathbf{X}(k+1) = \mathbf{\Omega}(k+1)\mathbf{X}(k), \quad (16)$$

where $\mathbf{X}(k) = \left[\mathbf{x}^T(k),\ \mathbf{y}^T(k-1),\ \dots,\ \mathbf{y}^T(k-j),\ \xi^T(k),\ \mathbf{u}^T(k-1),\ \mathbf{u}^T(k-l)\right]^T$ is the augmented state vector, and $\mathbf{\Omega}(k+1)$ is the augmented state transition matrix computed from $\mathbf{\Phi}$, $\mathbf{\Gamma}$, $\mathbf{C}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{J}$.

For periodic delays, there exists a positive integer M such that $\tau_{k+M}^{sc} = \tau_k^{sc}$. Using this property, the authors of [11] proved that the system in (16) is asymptotically stable if all eigenvalues of $\Xi_k^M = \prod_{j=1}^{M} \mathbf{\Omega}(k+M-j)$ are contained within the unit circle. $\Xi_k^M = \Xi_1^M$ was also proved as a lemma in this work. Ray and Halevi also suggested a way to improve this network-based system by appropriately selecting Δ_k [12].

The structure of the augmented discrete-time model is straightforward and easy for a network-based control system with periodic network delays. Moreover, this method can be modified to support non-identical sampling periods of a sensor and a controller as mentioned in [13]. However, as a result of extensive state augmentation, the *complexity* of the system increases significantly and proportionally to the dimension of the states and inputs. More computational time is required as well.

C. Queuing Method

1) Deterministic predictor-based delay compensation method

Luck and Ray derived a delay compensation method for network-based control systems with random delays in [24, 29]. The main idea of delay compensation is to utilize an observer to estimate the plant states and a predictor to compute predictive control inputs based on past output measurements, as depicted in Fig. 9.

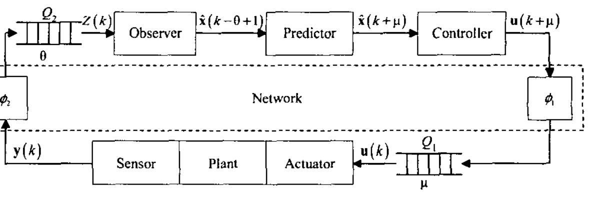


Fig. 9. Block diagram of the deterministic predictor-based delay compensation method.

In order to keep track of past measurements, these data have to be stored in a FIFO (First-In-First-Out) queue, denoted as Q_2, for a controller. Another queue at an actuator, denoted as Q_1, is used to store predictive control inputs from the controller. Thus, both sensor-to-controller delay and controller-to-actuator delay are transformed to constant delays. Also, the network-based system becomes time-invariant, which is much easier to control than random delay systems.

The delay compensation method can be applied to a network-based control system, which has the following dynamics:

$$\mathbf{x}(k+1) = \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{u}(k-\phi_1), \quad (17)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k), \quad (18)$$

$$\mathbf{z}(k) = \mathbf{y}(k-\phi_2), \quad (19)$$
$$Z(k) = \{\mathbf{z}(k), \mathbf{z}(k-1), \ldots\}, \quad (20)$$
$$\mathbf{u}(k) = \Upsilon(Z(k)), \quad (21)$$

where $\mathbf{z}(k)$ is the delayed output vector, $Z(k)$ is the set of past delayed outputs, and $\Upsilon(\bullet)$ is the control law. The non-negative integers ϕ_1 and ϕ_2 are the number of delayed samples at the system input and output vectors, respectively. The sizes of queues at the actuator and the controller are determined from the upper bounds of ϕ_1 and ϕ_2, denoted as μ and θ, respectively. Therefore, the first measurement in Q_2 processed by the observer is $\mathbf{y}(k-\theta)$, and the control input at the beginning of Q_1 has to be $\mathbf{u}(k+\mu)$.

The steps for applying the delay compensation algorithm are listed below.

a. *Using the set of past measurements* $Z(k)$ *in* Q_2, *the observer estimates the plant state* $\hat{\mathbf{x}}(k-\theta+1)$.
b. *The predictor uses the estimated state to predict* $\hat{\mathbf{x}}(k+\mu)$.
c. *The controller computes the predictive control input* $\mathbf{u}(k+\mu)$ *from* $\hat{\mathbf{x}}(k+\mu)$, *and then stores this control input into* Q_1.

Since the performance of the observer and predictor are highly dependent on the model certainty, the dynamic model of the plant has to be very precise. A way to analyze the model uncertainty is suggested in [24], but this approach results in unnecessarily long delays because of queues. The authors also mentioned that the stability of the system in this method can be analyzed, but they did not explicitly state how to perform the analysis.

2) Probabilistic predictor-based delay compensation method

Chan and Özgüner developed another predictor-based method for network-based control systems with random delays by using a probabilistic approach along with knowledge of the data lengths in a queue to improve the prediction [1]. A time-driven sensor, a time-driven controller, and an event-driven actuator are used in this approach. This technique can be applied to network-based systems containing the discrete-time plant in (12)-(13). In order to utilize the queue lengths, the plant has to be connected as illustrated in Fig. 10.

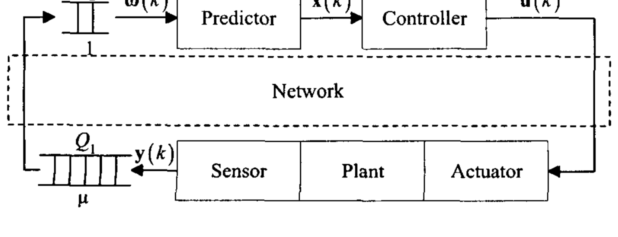


Fig. 10. Block diagram of the probabilistic predictor-based delay compensation method.

The FIFO queue Q_1 at the sensor has a capacity of μ, while the shift register Q_2 at the predictor has only one unit length. The output measurement $\mathbf{y}(k)$ is stored in Q_1 to be sent to Q_2 when the network traffic is available based on the priorities of system components. The number of data messages stored in the Q_1 queue is defined as *i*. The output from Q_2 is denoted as $\boldsymbol{\omega}(k)$.

At the k^{th} sampling period, if the sensor cannot send $\mathbf{y}(k)$ before Q_2 is read, $\boldsymbol{\omega}(k)$ will be set to equal the previous measurement $\boldsymbol{\omega}(k-1)$. In other cases, $\boldsymbol{\omega}(k)$ can be identical to any values in the set $\{\mathbf{y}(k), \mathbf{y}(k-1), \ldots, \mathbf{y}(k-\mu)\}$. However, the possible choices of $\boldsymbol{\omega}(k)$ can be reduced to only two values if *i* is known. Using a lemma stated by the authors [1], $\boldsymbol{\omega}(k)$ is either $\mathbf{y}(k-i)$ or $\mathbf{y}(k-i+1)$. To satisfy this case, the value of *i* has to be attached to every data message of $\mathbf{y}(k)$ for predictor processing when the message is sent.

Applying the information of Q_1, the predictor then computes the estimated current state $\hat{\mathbf{x}}(k)$ by:

$$\hat{\mathbf{x}}(k) = \mathbf{P}_0\left(\boldsymbol{\Phi}^{i-1}\boldsymbol{\omega}(k) + \mathbf{W}_i\right) + \mathbf{P}_1\left(\boldsymbol{\Phi}^{i}\boldsymbol{\omega}(k) + \mathbf{W}_{i+1}\right), \quad (22)$$

where

$$\mathbf{W}_i = \begin{cases} 0, & i = 1 \\ \left[\boldsymbol{\Gamma} \quad \boldsymbol{\Phi\Gamma} \quad \ldots \quad \boldsymbol{\Phi}^{i-2}\boldsymbol{\Gamma}\right] \cdot \\ \left[\mathbf{u}^T(k-1) \quad \mathbf{u}^T(k-2) \quad \ldots \quad \mathbf{u}^T(k-i+1)\right]^T, & i \neq 1 \end{cases}$$

The matrices $\mathbf{P}_0$ and $\mathbf{P}_1$ are weighting matrices. These two matrices are computed from the probabilities of the occurrences of $\mathbf{y}(k-i)$ and $\mathbf{y}(k-i+1)$. This formula is based on the assumption that the full state information is available (i.e., $\mathbf{y}(k) = \mathbf{x}(k)$). If the full state information cannot be obtained, an observer can be applied in this case without major modification. With the predictive states, various control laws can be applied in conjunction with this scheme. However, since this approach is not a control algorithm by itself, the stability test has to be evaluated on a controller algorithm. Its vigorous proof was not shown in this work.

Even prediction can be improved by this method. Unnecessary delays from queues still remain as in the case of the previous approach. Also, this method can only be applied to sensor-to-controller transmissions since the effects of controller-to-actuator delays τ_k^{sc} are neglected.

D. Optimal Stochastic Control Method

Nilsson and Wittenmark published an optimal stochastic control approach for control of a system plant over a random delay network in [17]. The effects of network delays in network-based systems were treated as an LQG (Linear-Quadratic-Gaussian) problem. This approach uses an event-driven controller, an event-driven actuator, and a time-driven sensor. In addition to the general assumptions in section III.*A*, the following assumptions are required:

1) The sum of sensor-to-controller delay and controller-to-actuator delay is less than the sampling period (i.e., $\tau_k^{sc}+\tau_k^{ca}<T$*).*

2) The controller delay τ_k^{c} *is included in* τ_k^{ca}.

3) The information of all past delays (i.e., $\{\tau_0^{sc},\tau_1^{sc},\ldots,\tau_k^{sc},\tau_0^{ca},\tau_1^{ca},\ldots,\tau_{k-1}^{ca}\}$*) is known.*

The dynamics of a linear plant derived from the continuous plant in (10)-(11) for this technique is described by:

$$\mathbf{x}(k+1)=\mathbf{\Phi}\mathbf{x}(k)+\mathbf{\Gamma}_0(\boldsymbol{\tau}_k)\mathbf{u}(k)+\mathbf{\Gamma}_1(\boldsymbol{\tau}_k)\mathbf{u}(k-1)+\mathbf{v}(k),\quad(23)$$

$$\mathbf{y}(k)=\mathbf{C}\mathbf{x}(k)+\mathbf{w}(k)\quad(24)$$

where
$$\boldsymbol{\tau}_k=\left[\tau_k^{sc},\tau_k^{ca}\right]^T,$$
$$\mathbf{\Phi}=\exp(\mathbf{A}T),$$
$$\mathbf{\Gamma}_0(\boldsymbol{\tau}_k)=\int_0^{T-\tau_k^{sc}-\tau_k^{ca}}\exp(\mathbf{A}\zeta)d\zeta\mathbf{B}\text{, and}$$
$$\mathbf{\Gamma}_1(\boldsymbol{\tau}_k)=\int_{T-\tau_k^{sc}-\tau_k^{ca}}^{T}\exp(\mathbf{A}\zeta)d\zeta\mathbf{B}.$$

The stochastic processes $\mathbf{v}(k)$ and $\mathbf{w}(k)$ are uncorrelated Gaussian white noises with zero means. These equations are modified from the constant delay system in [23].

The goal of the optimal stochastic control method is to minimize the following cost function in the case that full state information is available:

$$J(k)=E\left[\mathbf{x}^T(N)\mathbf{Q}_N\mathbf{x}(N)\right]+E\sum_{k=0}^{N-1}\begin{bmatrix}\mathbf{x}(k)\\\mathbf{u}(k)\end{bmatrix}^T\mathbf{Q}\begin{bmatrix}\mathbf{x}(k)\\\mathbf{u}(k)\end{bmatrix},\quad(25)$$

where $\mathbf{Q}_N$ and $\mathbf{Q}$ are weighting matrices.

Based on this cost function, the authors derived a control law for optimal state feedback by using dynamic programming, which is conceptually described as:

$$\mathbf{u}(k)=-\mathbf{L}(k,\boldsymbol{\tau}_k)\begin{bmatrix}\mathbf{x}(k)\\\mathbf{u}(k-1)\end{bmatrix},\quad(26)$$

where $\mathbf{L}(k,\boldsymbol{\tau}_k)$ is a function of k and $\boldsymbol{\tau}_k$. The delays are assumed to be independent in this case. If the full state information is not available, optimal estimators such as Kalman filter can be applied for (26). Nevertheless, this case requires the past information of plant output and input (i.e., $\{\mathbf{y}(0),\ldots,\mathbf{y}(k),\mathbf{u}(0),\ldots,\mathbf{u}(k-1)\}$ in conjunction with the past information of the delays. In addition, the authors derived a modified version of this control law to use with the delays modeled by using a Markov Chain in [27]. The stability of the network-based system for both independent delays and delays modeled by Markov chains are also discussed by using stochastic stability analysis.

Based on the cost function in (25), this approach has been shown to have better performance than the deterministic predictor-based method since the delay problems are solved by not using queues. However, the drawback is a large requirement of controller memory to store a large amount of past information from the initial point. Also, since this approach is mainly based on assumption *1)*, this approach may not be effective for a system with fast response time.

E. Perturbation Method

This method was introduced by Walsh, Beldiman, and Bushnell [25, 28]. A time-driven sensor, an event-driven controller, and an event-driven actuator are utilized as the basic system components in this method. However, only data messages from a sensor can be transmitted through a network. A control loop in this case consists of a nonlinear controller and a nonlinear plant. The analysis and derivations used can be similarly applied to linear systems, as described in [28]. Fig. 11 shows the block diagram of this system.

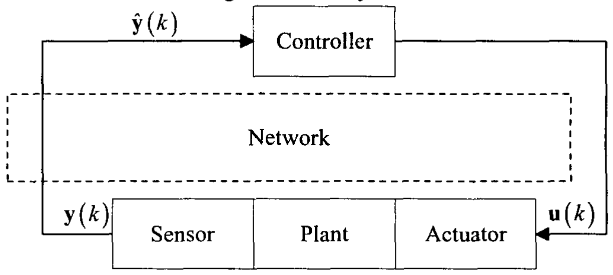


Fig. 11. Block diagram of a system using the perturbation method.

The perturbation method can be applied to both cyclic service and random access types of networks. However, these network are restricted to be *priority-based networks.* This means that the networks can assign different priorities to data transmissions among system components. The priorities are managed by priority scheduling algorithms, which can be either fixed or changeable. Various scheduling algorithms for perturbation method are discussed in [31].

The key concept of the perturbation method is to formulate delay effects in network-based systems as a perturbation of a continuous-time system under the assumption that there is no observation noise. This case requires a very small sampling period so that a network-based system can be approximated by a continuous-time model.

The dynamics of the network-based system are represented by:

$$\dot{\mathbf{x}}(t)=\mathbf{f}(t,\mathbf{x}(t),\mathbf{e}(t)),\quad(27)$$

where $\mathbf{x}(t)=\left[\mathbf{x}_p^T(t),\mathbf{x}_c^T(t)\right]^T$ is the augmented state vector containing the plant state vector $\mathbf{x}_p(t)$ and the controller state vector $\mathbf{x}_c(t)$. The function $\mathbf{f}(\bullet)$ is a nonlinear function.

The most updated output received by the controller is denoted as:

$$\hat{\mathbf{y}}(t)\triangleq\mathbf{y}(t-\tau^{sc}),\quad(28)$$

where $\mathbf{y}(t)$ is the output of the plant defined in (11), and τ^{sc} indicates the sensor-to-controller delay in the continuous-time case. The difference between $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t)$ is formulated as an error of the network-based system described as:

$$\mathbf{e}(t)=\mathbf{y}(t)-\hat{\mathbf{y}}(t).\quad(29)$$

The dynamics of $\mathbf{e}(t)$ can also be described by a nonlinear equation, which is a function of t, $\mathbf{x}(t)$, and $\mathbf{e}(t)$:

$$\dot{\mathbf{e}}(t)=\mathbf{g}(t,\mathbf{x}(t),\mathbf{e}(t)). \tag{30}$$

The main idea of all the derivations in this work is to model $\mathbf{e}(t)$ as a special type of vanishing perturbation [30]. This concept was used to derive a delay bound ρ such that the network-based system remains stable if the sensor-to-controller delay is less than this bound (i.e., $\tau^{sc}<\rho$). The exponential stability of network-based systems associated with a proposed scheduling algorithm was proved by considering ρ.

A major advantage of the perturbation method is that it can be applied to a nonlinear system. However, this approach is not applicable to a system containing controller-to-actuator delays.

F. Sampling Time Scheduling Method

The sampling time scheduling method can be applied to cyclic service networks in which all system component connections on the network are known in advance. The fundamental concept of this technique, introduced by Hong [16], is to appropriately select a long enough sampling period for a discrete-time network-based system such that network delays do not affect the control performance and the system can remain stable. The control delay τ_k in a control loop must be assumed to be less than the sampling period of the loop. In this case, the loop consists of a time-driven sensor, a time-driven controller and an event-driven actuator.

For the discrete-time system in (12)-(13), the effects of varying network delays are exerted by the interval $[kT,(k+1)T)$ rather than by the delay occurring in the same interval. Thus, if the worst case of delays in network transmissions can be found, and this delay does not exceed the sampling period *T*, the network-based control system will operate in the same way as a generic discrete-time system. Nevertheless, the sampling time *T* cannot be greater than a maximal allowable bound φ. The bound φ is defined as the longest possible sampling period of the system, in which the system is still stable. Different control loops on the same network can have different bounds. This concept is similar to the delay bound ρ in the previous method, but it is used for different purposes. This technique can only be used in case that the system is one-dimensional. General frequency-domain analysis can be used to find φ and to check system stability.

The algorithm of sampling time scheduling method is based on the window concept illustrated in Fig. 12, where *L* and σ are the transmission periods of a pure data message and its overhead, respectively; T_1 is the sampling period of the system, which has the smallest allowable delay bound compared to other control loops on the same network; and *r* is the number of data messages that can be served by the network on the medium during the worst case of network transmissions. The maximal delay bound of this system is denoted as φ_1. The number of control loops on the network is defined as *M*.

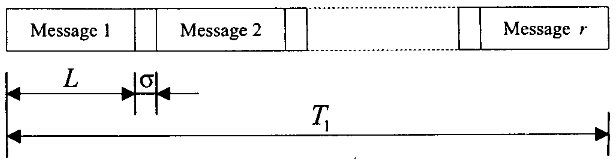


Fig 12. Windows of data transmissions in a sampling period T_1.

The author proved that the sampling time T_1 could be computed by:

$$T_1=\frac{\varphi_1+L}{3}. \tag{31}$$

In order to assign sampling periods to other control loops (i.e., the 2nd loop to the M^{th} loop) on the same network by the scheduling algorithm, these control loops have to be indexed in an ascending order from the sizes of their maximal allowable delay bounds. The maximum allowable delay bounds of the 2nd loop to the M^{th} loop are denoted as φ_2 to φ_M, respectively. Using this index, the sampling time of the 2nd to M^{th} control loops were derived as:

$$T_i=k_iT_1,\ \forall i\in\{2,3,\ldots,M\}, \tag{32}$$

$$k_i=\Lambda\left(\frac{\varphi_i-(T_1-L)}{2T_1}\right), \tag{33}$$

where $a=\Lambda(b)$ indicates that $a=2^{v_i}, v_i\in\{0,1,2,\ldots\}$, which is the "closest" to but does not exceed *b*.

In a special case, in which the number of all system components and other resources connected on the same network is less than *r*, the sampling time periods of all control loops are determined by:

$$T_i=\frac{\varphi_i-(T_1-L)}{2},\ \forall i\in\{1,2,\ldots,M\}. \tag{34}$$

It was also proved that the network utilization is optimum if:

$$2\sum_{i=1}^{M}\frac{T_1}{T_i}=r. \tag{35}$$

Kim, Kwon, and Park expanded the concept of maximal allowable bound in Hong's approach to develop another algorithm for the multi-dimensional cases in [26, 32]. In their approach, the delay bound of each system is obtained from different stability criteria. The dynamics of such a multi-dimensional system on the network is briefly expressed as follows:

$$\begin{aligned}\dot{\mathbf{x}}(t)=\mathbf{A}\mathbf{x}(t)+\mathbf{A}_1\mathbf{x}(t-\tau^{sc})+\mathbf{A}_2\mathbf{x}(t-\tau^{sc}-\tau^{c})+\\ \mathbf{A}_3\mathbf{x}(t-\tau^{ca}-\tau^{c}),\end{aligned} \tag{36}$$

where $\mathbf{x}(t)=[\mathbf{x}_p^T(t),\mathbf{x}_c^T(t)]^T$, $\mathbf{x}_p(t)$ is the plant state vector, $\mathbf{x}_c(t)$ is the controller state vector, τ^{sc}, τ^{c}, and τ^{ca} are sensor-to-controller, computational, and controller-to-actuator delays, respectively, denoted for this continuous-time model. The matrices $\mathbf{A}$, $\mathbf{A}_1$, $\mathbf{A}_2$, and $\mathbf{A}_3$ are calculated from the realizations of the plant and controller. Two existing asymptotically stable criteria based on the Lyapunov function can be used to find the maximal allowable delay of this system [33-35].

With the scheduling method, not only can the delay problem be partially solved, but also network utilization can be assigned appropriately for control loops and other resources connected on the same network. However, since the sampling times of control loops are multiples of T_1, the sampling time delays may be longer than necessary, similar to the queuing methods mentioned earlier in section III.*C*.

IV. Conclusion

From the presented network-based control techniques, it is noticeable that these techniques have different advantages and disadvantages. In order to apply one of these techniques, designers have to consider both the network and the control algorithm. Nevertheless, none of these techniques is perfect because many unrealistic assumptions are extensively used. Network-based control needs much research and improvement in order to be used for practical applications. The improvement of network-based control would be substantially useful for various control applications operated on fast-growing networking technologies in the future.

References

[1] H. Chan and U. Özgüner, "Closed-loop control of systems over a communication network with queues," *International Journal of Control*, vol. 62, pp. 493-510, 1995.

[2] Ü. Özgüner, H. Göktas, H. Chan, J. Winkelman, M. Liubakka, and R. Krotolica, "Automotive suspension control through a computer communication network," presented at IEEE Conference on Control Applications, 1992.

[3] N. Boustany, M. Folkerts, K. Rao, A. Ray, L. Troxel, and Z. Zhang, "A simulation based methodology for analyzing network-based intelligent vehicle control systems," presented at Intelligent Vehicles Symposium, 1992.

[4] B. P. Zeigler and J. Kim, "Extending the DEVS-Scheme knowledge-based simulation environment for real-time event-based control," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 351 - 356, 1993.

[5] K. B. Lee and R. D. Schneeman, "Internet-based distributed measurement and control applications," *IEEE Instrumentation & Measurement Magazine*, vol. 2, pp. 23 - 27, 1999.

[6] T.-J. Tarn and N. Xi, "Action synchronization and control of internet based telerobotic systems," presented at IEEE International Conference on Robotics and Automation, 1999.

[7] J. W. Overstreet and A. Tzes, "An Internet-based real-time control engineering laboratory," *IEEE Control Systems Magazine*, vol. 19, pp. 19-34, 1999.

[8] G. V. Kondraske, R. A. Volz, D. H. Johnson, D. Tesar, J. C. Trinkle, and C. R. Price, "Network-based infrastructure for distributed remote operations and robotics research," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 702-704, 1993.

[9] J. E. F. Baruch and M. J. Cox, "Remote control and robots: an Internet solution," *Computing & Control Engineering Journal*, vol. 7, pp. 39-45, 1996.

[10] J. W. Overstreet and A. Tzes, "Internet-based client/server virtual instrument designs for real-time remote-access control engineering laboratory," presented at American Control Conference, 1999.

[11] Y. Halevi and A. Ray, "Integrated communication and control systems: Part I - Analysis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 367-373, 1988.

[12] A. Ray and Y. Halevi, "Integrated communication and control systems: Part II - Design considerations," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 374-381, 1988.

[13] L.-W. Liou and A. Ray, "Integrated communication and control systems: Part III - Nonidentical sensor and controller sampling," *Journal of Dynamic Systems, Measurement, and Control*, vol. 112, pp. 357-364, 1990.

[14] K. G. Shin and P. Ramanathan, "Real-time computing: a new discipline of computer science and engineering," *Proceedings of the IEEE*, vol. 82, pp. 6-24, 1994.

[15] M. Törngren, "Modeling and design of distributed real-time control applications," 1995 Ph.D. thesis, Royal Institute of Technology, KTH, Sweden.

[16] S. H. Hong, "Scheduling algorithm of data sampling times in the integrated communication and control systems," *IEEE Transactions on Control Systems Technology*, vol. 3, pp. 225 - 230, 1995.

[17] J. Nilsson, B. Bernhardsson, and B. Wittenmark, "Stochastic analysis and control of real-time systems with random time delays," *Automatica*, vol. 34, pp. 57-64, 1998.

[18] D. Bertsekas and R. Gallager, *Data networks*, 2 ed. Upper Saddle River, NJ: Prentice Hall, 1992.

[19] K. Zahr and C. Slivinsky, "Delay in multivariable computer controlled linear systems," *IEEE Transactions on Automatic Control*, vol. AC-19, pp. 442-443, 1974.

[20] A. Ray, "Distributed data communication networks for real-time process control," *Chemical Engineering Communications*, vol. 65, pp. 139-154, 1988.

[21] A. Ray, S. H. Hong, and S. Lee, "Discrete-event/continuous-time simulation of distributed data communication and control systems," *Transactions of The Society for Computer Simulation*, vol. 5, pp. 71-85, 1988.

[22] K. Hirai and Y. Satoh, "Stability of a system with variable time delay," *IEEE Transactions on Automatic Control*, vol. AC-25, pp. 552-554, 1980.

[23] K. J. Åström and B. Wittenmark, *Computer-controlled systems: Theory and Design*, 2 ed. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1990.

[24] R. Luck and A. Ray, "An observer-based compensator for distributed delays," *Automatica*, vol. 26, pp. 903-908, 1990.

[25] G. C. Walsh, O. Beldiman, and L. Bushnell, "Asymptotic behavior of networked control systems," presented at IEEE International Conference on Control Applications, 1999.

[26] Y. H. Kim, H. S. Park, and W. H. Kwon, "Stability and a scheduling method for network-based control systems," presented at IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation, 1996.

[27] J. Nilsson, "Real-time control systems with delays," 1998 Ph.D. thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

[28] G. C. Walsh, H. Ye, and L. Bushnell, "Stability analysis of networked control systems," presented at American Control Conference, 1999.

[29] R. Luck and A. Ray, "Experimental Verification of a delay compensation algorithm for integrated communication and control systems," *International Journal of Control*, vol. 59, pp. 1357-1372, 1994.

[30] H. K. Khalil, *Nonlinear systems*, 2 ed. Upper Saddle River, NJ: Prentice Hall, 1995.

[31] G. C. Walsh, O. Beldiman, and L. Bushnell, "Error encoding algorithms for networked control systems," presented at IEEE Conference on Decision and Control, 1999.

[32] Y. H. Kim, H. S. Park, and W. H. Kwon, "A scheduling method for network-based control systems," presented at American Control Conference, 1998.

[33] T.-J. Su and C.-G. Huang, "Robust stability of delay dependence for linear uncertain systems," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1656-1659, 1992.

[34] P.-L. Liu and T.-J. Su, "On a sufficient condition for robust stability of delay systems," presented at American Control Conference, 1995.

[35] T. Mori, N. Fukuma, and M. Kuwahara, "Simple stability criteria for single and composite linear systems with time delays," *International Journal of Control*, vol. 34, pp. 1175-1184, 1981.

[36] W. Stallings, *Data & computer communication*, 6 ed. Upper Saddle River, NJ: Prentice Hall, 2000.

[37] B. Etkin and L. D. Reid, *Dynamics of flight: stability and control*. New York: Wiley, 1996.