Switched Ethernet-Based Real-Time Networked Control System with Multiple-Client–Server Architecture

Naveen Bibinagar and Won-jong Kim, *Senior Member, IEEE*

***Abstract*—This paper experimentally verifies that a multiple-client–server architecture based on switched Ethernet can be used as a real-time communication standard for possible applications in factory automation, by observing the effects of packet delays, network congestion, and packet loss on the performance of a networked control system (NCS). The NCS experimental setup used in this research involves real-time feedback control of multiple plants connected to one or more controllers over the network. A multiclient–multiserver (MC–MS) architecture on a local area network (LAN) was developed using user datagram protocol as the communication protocol. In the single-client–single-server (SC–SS) system, as the Ethernet link utilization increased over 82%, the average packet delays and steady-state error of a dc motor speed-control system increased by 2231% and 304%, respectively. As the link utilization increased beyond the threshold, employing an additional server in the NCS reduced average packet delays and also overcame the negative effects of Ethernet's flow control mechanism. The MC–MS architecture is tested with artificially generated random packet loss. The standard deviation of steady-state error (SSE) at 80% utilization with packet loss is found to be 70.2% less than SC–SS and 200% less than multiclient–single-server architecture. The MC–MS architecture remained stable till 70% of control or measurement packet loss.**

***Index Terms*—Client–server architecture, link utilization, network control system, real-time system, switched ethernet.**

I. Introduction

The framework of an NCS with a single controller is shown in Fig. 1. In this setup, the communication link between the controller and the plant has to compete with the traffic from other controllers and applications on the network.

Li *et al*. [1] demonstrated the use of NCS and found it to be an efficient solution for large-scale mechatronic systems. Industrial communications has come a long way from a dedicated point-to-point connection to optical wireless systems [2]. In the late 1990s, Profibus [3] and FIP [(International electrotechnical commission (IEC) 61158] were made an international standard in the field of industrial automation. However, the high costs of hardware and incompatibility of multiple-vendor systems have become barriers in its acceptance. Recently, the computer network standard IEEE 802.3 ethernet has come up as an alternative for real-time communication. The advancements in Ethernet have made it possible to employ it in factory automation systems at the cell and the plant levels [4]. Ethernet has been studied extensively with a Poisson traffic model in different simulation models. However, in a real-time scenario, the traffic is mostly bursty in nature. Mazraani and Parulkar [5] found that as long as the network utilization did not reach a particular threshold the behavior of the ethernet remained the same under bursty conditions. They also observed that as the utilization increased beyond a threshold, packet delay, queue lengths, and packet loss increased drastically. To address the issues of nondeterminism, network architectures based on switching have gained significance. Switches are network devices that operate at the data-link layer interconnecting various hosts. Contrary to a shared architecture, in switched network architectures, frames are sent only to the addressed nodes reducing the number of collision domains considerably leading to a better handling of the traffic and considerable reduction in delay.

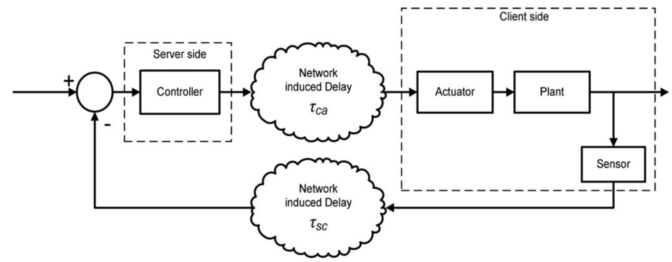


Fig. 1. Block diagram of feedback control over a network.

In this research, a similar switched Ethernet network is used. Although previous researchers [6], [7] found a switched network to be more effective than conventional Ethernet, in this paper we present an in-depth experimental analysis by comparing three types of switched network architecture under varying network load and packet loss conditions. It is found that MC–MS architecture performed better with the lowest packet delays and standard deviation of SSE.

The experimental setup including the hardware, software, and communication network is elaborated in Section II. The development of the MC–MS architecture and the implementation of the client-rejection algorithm are presented in Section III. The

Manuscript received October 13, 2010; revised January 31, 2011, May 19, 2011, and July 11, 2011; accepted July 19, 2011. Date of publication September 1, 2011; date of current version September 12, 2012. Recommended by Technical Editor G. Herrmann. This work was supported in part by the TAMU Program to Enhance Scholarly and Creative Activities under Grant 2010-SAC-8779.

N. Bibinagar is with CCI, Austin, TX 78744 USA (e-mail: Naveen.bibinagar@gmail.com).

W.-J. Kim is with the Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843-3123 USA (e-mail: wjkim@tamu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMECH.2011.2163316

Fig. 2. DC motor speed-control systems.

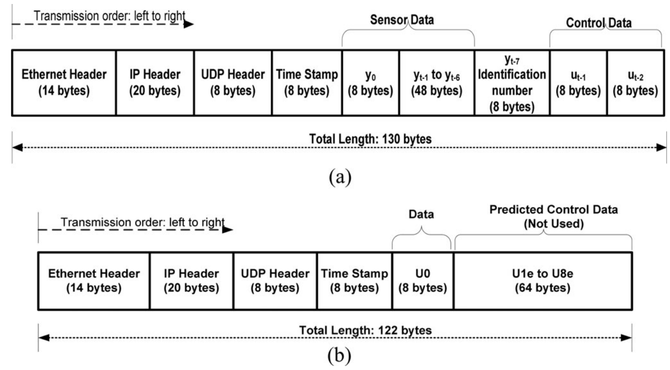


Fig. 3. Structures of (a) a sensor packet and (b) a control packet.

experimental results and the corresponding discussions are provided in Sections IV followed by conclusions in Section V.

II. Experimental Setup

DC motor speed-control systems (dc motor system) as shown in Fig. 2 are used as multiple clients in our test bed. The objective of a dc motor system is to control the speed of the motor at a predetermined speed and in this paper the speed is fixed at 20 rps. The dc motor system consists of five AMAX 26 dc motors [8] connected to five HEDS 5540 Digital Encoders. A National Instruments (NI) PCI 6221 board is connected at the controller node for data acquisition. The encoders are the sensors that send speed signal outputs in terms of pulse counts per unit time and the controller sends back the control input as a pulse width modulation (PWM) signal. The digital controller for a dc motor speed-control system is given by (1) with a sampling period of 3 ms [9]. Here, $u(k)$ is defined as the control signal, and $e(k)$ is defined as the error, being the difference between the actual speed and the reference speed:

$$u(k) = u(k-1) + 1.51e(k) - 1.5e(k-1). \tag{1}$$

From the experiments, when the three different NCS architectures are subject to varying loads, SSE is found to be an effective measure to the effects of random packet delays and losses. Therefore, the performance of the dc motor system is measured in terms of the standard deviation of the SSE. For a reference speed of 20 rps, the dc motor system is observed to reach the steady state after an exchange of about 60 packets. For a total of 5000 packets in every experiment, the standard deviation of SSE is calculated for 4900 packets.

Kim *et al.* [10] originally developed the UDP-based software program for the SC–SS architecture on a Linux-Ubuntu 6.10 operating system. To make Ubuntu real-time capable, open-source real-time application interface (RTAI) 3.4 [11] was installed on it. RTAI modifies the Ubuntu kernel to make it fully preemptive. Linux control and measurement device interface (COMEDI) [12] was installed to act as a data acquisition interface between the NI 6221 board and Ubuntu. The combination of Ubuntu, RTAI, COMEDI, and NI 6221 makes it a complete real-time data-acquisition system.

There are many factors that affect the performance of ethernet [13]. The communication network used in this research is based on switched ethernet consisting of a Cisco 2940G series switch operating in a store-and-forward switching mode. In this mode, when a client sends a packet to a server, the switch in-between receives the packet and does an error checking on it. Based on the destination medium access control (MAC) address, it forwards the packet to that particular egress queue. If there are packets already in the queue, it is stored in the queue till its turn, else it is routed immediately. Hence, as more clients try to connect to a single server, the link utilization starts to increase and the packets from different clients have to wait in the queue before being sent to that particular server. The average propagation delay measured using the PING command is observed to be 0.388 ms [10]. The minimum packet length is 64 bytes, including a 4-byte frame check sequence and a 14-byte header. The maximum allowable packet length is restricted at 1518 bytes. Theoretically, ethernet standard specifies 100 hosts per segment and 1024 total number of hosts in a multisegment [13]. In this experimental setup and specifically in this multisegment subnet, the total number of hosts including the NCS is currently around 110. The hosts in the NCS are assumed to have fixed buffer sizes.

The sensor packet structure is shown in Fig. 3(a); the total length of the packet is 130 bytes with the protocol headers taking 42 bytes. The time stamp is used to calculate the average delay ($T_{\rm av}$) of the packet. The fields y_0 to y_{t-6} are used to carry sensor data. The field y_{t-7} is used as an identification number to run different control loops for different clients and also used to accept or reject clients in the MC–MS scenario, explained in Section III. The control packet structure is shown in Fig. 3(b); the total length of the packet is 122 bytes including the headers. Server receives the time stamp from the sensor packet and copies it in the corresponding time stamp field in the control packet. The field u_0 is used to send the actual control signal. Kim *et al.* [10] in his ball maglev research used the fields u_{1e} to u_{4e} to send predicted control values in the case of artificial packet losses. As the motivation of this research is to investigate and compare the performances of various NCS architectures under varying network load and link utilization, no extra data handling in terms of predictors or estimators is used and u_0 is the only control signal field used in this research.

The performance of a network is measured in various factors such as average delay, throughput, channel capacity, stability, and fairness. It is important to differentiate these performance measures with respect to different types of networks. For example, in a real-time system, average delay and network load may be the most important performance measures [13] whereas in a non-real-time system, throughput and channel capacity may be more important. The performance measures studied in this research are average delay ($T_{\rm av}$), and ethernet link utilization. It is worthwhile to clarify that in a switched Ethernet network, the load on the network translates to the communication load on the link between a client, switch and a server. This network load is amply accounted through the ethernet link-utilization parameter. Throughput is defined as the amount of bandwidth available for a network application during a given time and this is complementary to the measured link utilization.

Total delay ($T_{\rm d}$) is measured as the time elapsed from when the sender tries to acquire a channel for transmitting data to the time that the receiver receives them. $T_{\rm pre}$ and $T_{\rm post}$ are minor components of $T_{\rm d}$ that depends on the processing power of a compute, termed as $T_{\rm proc}$. Both the components can be safely assumed to be fixed and equal because the sender and receiver are identical computers. $T_{\rm tx}$ is a function of the frame length, bit rate of the channel and propagation delay ($T_{\rm prop}$):

$$T_{\rm d} = T_{\rm destination} - T_{\rm source} = T_{\rm pre} + T_{\rm wait} + T_{\rm tx} + T_{\rm post}$$

$$T_{\rm d} = 2 \times T_{\rm proc} + T_{\rm wait} + T_{\rm tx}; \quad (T_{\rm pre} = T_{\rm post}). \tag{2}$$

In the experimental setup, the hosts are going to be at a fixed position and so, the $T_{\rm prop}$ can be assumed as constant and the frame transmission time ($T_{\rm frame}$) is not going to vary much due to high bit rates used. Because this research deals with full-duplex switched ethernet [14], the waiting time ($T_{\rm wait}$) at the hosts is eliminated but still present at the switch in the form of queues. $T_{\rm wait}$ depends on the network load, arrival rate of packets and the blocking time of the previous packets already present in the queue. Average packet delay ($T_{\rm av}$) is measured from the time stamp field in the packet. During the preparation of the sensor packet at the client, the time stamp field is initialized with the current time using the RTAI function rt_get_time(). At the server the controller program copies the time stamp from the sensor packet to the control packet and sends it back to the client.

At the client, the average delay ($T_{\rm av}$) is calculated by comparing the time stamp field with the current time. From Fig. 4, it should be noticed that the delay experienced from client to server need not be equal to that from server to client because of the probabilistic nature of waiting times at the switch. So, the calculated round-trip delay gives a more accurate figure of the transmission delay.

The network utilization is termed as either Ethernet link utilization or link utilization and is measured in terms of the capacity of the channel, as shown in the following:

$$\text{Utilization (Mbps)} = \text{packet rate}\left(\frac{\text{pkts}}{\text{s}}\right) \times \text{packet length} \times \left(\frac{\text{bytes}}{\text{pkt}}\right) \times 8 \left(\frac{\text{bits}}{\text{byte}}\right) \tag{3}$$

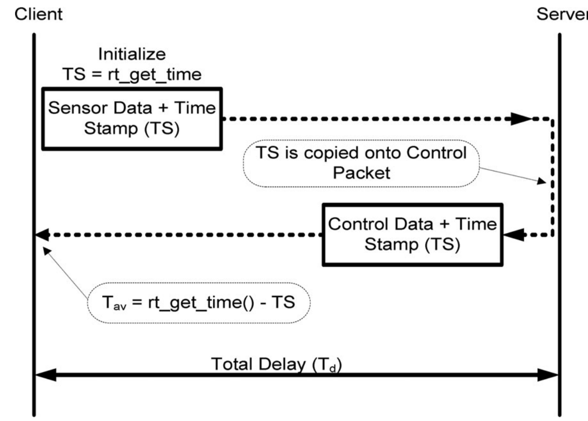


Fig. 4. Average packet delay calculation.

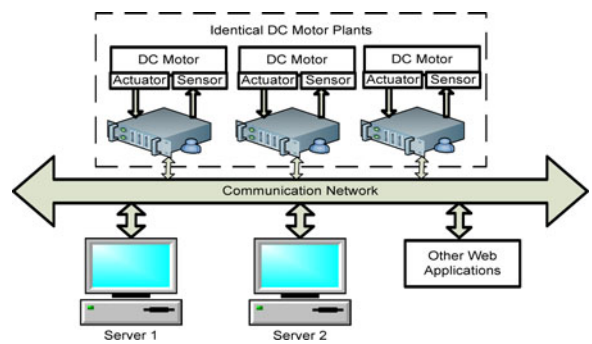


Fig. 5. Block diagram of multiclient–multiserver architecture.

III. Multiclient–Multiserver Architecture

Kim *et al.* [10] designed a single-client–single-server closed-loop real-time architecture that had the ball maglev system attached to the client computer and the controller program running on the server computer. Although ethernet allows up to 1024 hosts per multisegment, there is always a physical limitation on the number of clients that a single server can handle. As the number of clients being served by a single server is increased, the communication is not only affected by network-induced delays, but also by the delays caused due to the processing time at the server. If the plants connected to the clients have a high sampling frequency, then it burdens the server with heavy timing requirements. To overcome these challenges a multiserver architecture as shown in Fig. 5 was developed.

A multiserver architecture improves the scalability of an NCS by giving the flexibility for a client to connect to a server based on the network load. In this research, the architecture is developed with the approach that the server can accept or reject clients based on the load, and the rejected client can automatically send a new request to a different server. The server decides solely based on the client's identification number. The previously developed architecture satisfies our needs because we are mainly concerned about the performance evaluation of an NCS under varying loads in an MC–MS scenario.

Our MC–MS client-rejection algorithm is as follows.

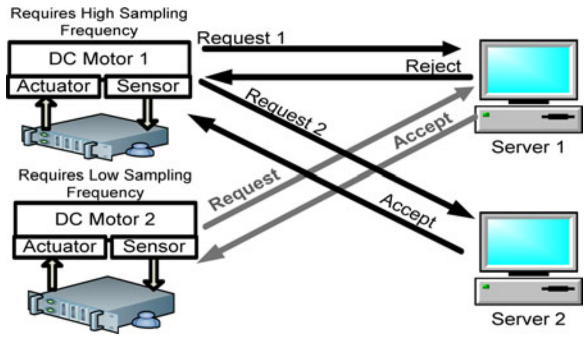


Fig. 6. Illustration of the client rejection algorithm.

1) The server program starts and waits for requests from clients.
2) A client initiates a connection by sending a sensor packet with sensor signal values and an identification number.
3) The server checks for the identification number and decides whether it has to serve the client or not.
4) If it is found eligible for service, the server program selects the control loop relevant to the client and executes the sensor-control communication.
5) If it is found not eligible, the server sends a control packet with the control signal equal to "0." In a normal scenario the control signal is always found to be offset from zero. So, when the client receives a perfect "0" control signal, it considers it as the "reject" packet. The client then selects a different server and proceeds on with Step 2.

A general overview of the client-rejection process is illustrated in Fig. 6. DC motor 1 is attached to Client 1 and dc motor 2 is attached to Client 2. In this setup, Server 1 is programmed to accept only slower clients like Client 2 that is operating at low sampling frequency. So, when Client 2 sends a request, it gets accepted by Server 1 whereas the request from Client 1 to Server 1 gets rejected. Based on the client-selection–rejection algorithm, Client 1 sends a new request to Server 2. Because Server 2 is programmed to accept faster clients like Client 1, the request gets accepted as illustrated. Although the connection requests between the clients and servers are known previously, the motivation behind the client-rejection algorithm is to accommodate the dynamics of increasing network utilization and load. It is to be noted that the client rejection happens only during the starting of the connection process. From the experiments, it is found that the delay introduced between the client rejection and new server selection is in the order of 0.5 ms. This delay is both negligible and is worth the investment to have a congestion-free and nondisruptive real-time performance.

When there are multiple clients connected to a single low-end server, the formerly minor component like T_{proc} might become a major component of T_{d}. The multiserver architecture proved to be an effective solution in such scenarios. By providing an additional server the waiting time in the queues is reduced. To investigate for network load, a maximum sampling frequency is chosen to generate the maximum number of control-sensor packets. Due to the physical limitation of the RTAI being used, a maximum sampling frequency of 333 Hz equivalent to a 3-ms sampling period is chosen.

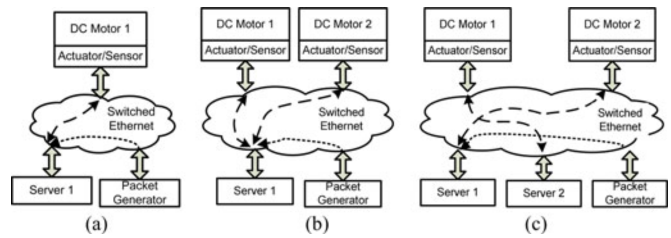


Fig. 7. NCS Architectures (a) SC–SS, (b) MC–SS, and (c) MC–MS.

With this sampling, a single client-server pair generates sensor-control packets at a constant rate of 333.3 packets/s that sums up to 666.6 packets/s in both directions. So, as the number of client–server pair increases, the rate at which the exchange of packets occurs also increases. For one client–server pair the link utilization is found to be 0.61 Mbps, and with two clients and two servers in the MC–MS architecture, the maximum network utilization is found to be 1.22 Mbps that is equivalent to less than 1.5% of the total capacity. Therefore, to maximize the effects of network load or link utilization, an artificial packet generator tool [15] is used. The packet generator is used to generate ethernet packets at different lengths and varying packet rates to load the network according to the experiments. The NCS experimental setup is designed into three architectures. DC motor speed control systems are used as clients with the data acquisition program running on the clients and the controller program running on the server computer. Based on the number of clients and servers, the architectures are classified as SC–SS, MC–SS, and MC–MS architectures. The packet generator computer remains a standard attachment in all the architectures; it is the number of clients and servers that differ. For example in the MC–SS architecture shown in Fig. 7, there is a bidirectional communication between the two dc motor clients and one server, and a one-way communication from packet generator to the Server 1. The other two architectures SC–SS and MC–MS can be understood from the illustrations in Fig. 7.

IV. Experimental Results

Experiments were designed to provide maximum coverage with respect to network load, utilization, and packet loss. To create scenarios of real-time and non-real-time network utilization [13], the experimental procedure is divided into two types of experiments. A non-real-time scenario is conducted by transmitting larger packet sizes at lower transmission rates, and a real-time scenario, by transmitting smaller sized packets at higher packet rates.

1) Increasing the Packet Length While Keeping the Packet Rate Constant: In each round of experiments, the packet rate was fixed, and the packet length was increased for every consecutive iteration. The packet length was increased from 100 bytes to 1500 bytes in steps of 300 bytes generating a total of eight data points for every experiment. A total of six experiments were conducted by increasing the packet rate from 9000 to 14,000 packets/s in steps of 1000 packets/s. Hence, a total of 48 data

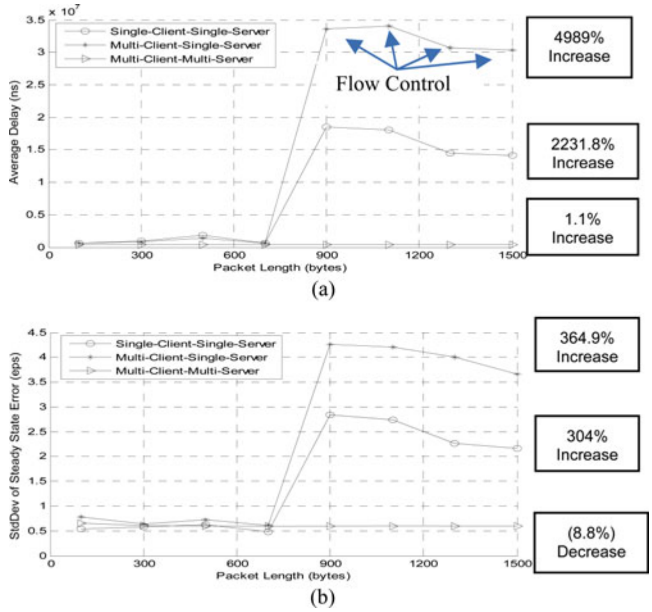


Fig. 8. Performance comparisons of experiments SC–SS-1, MC–SS-1, and MC–MS-1.

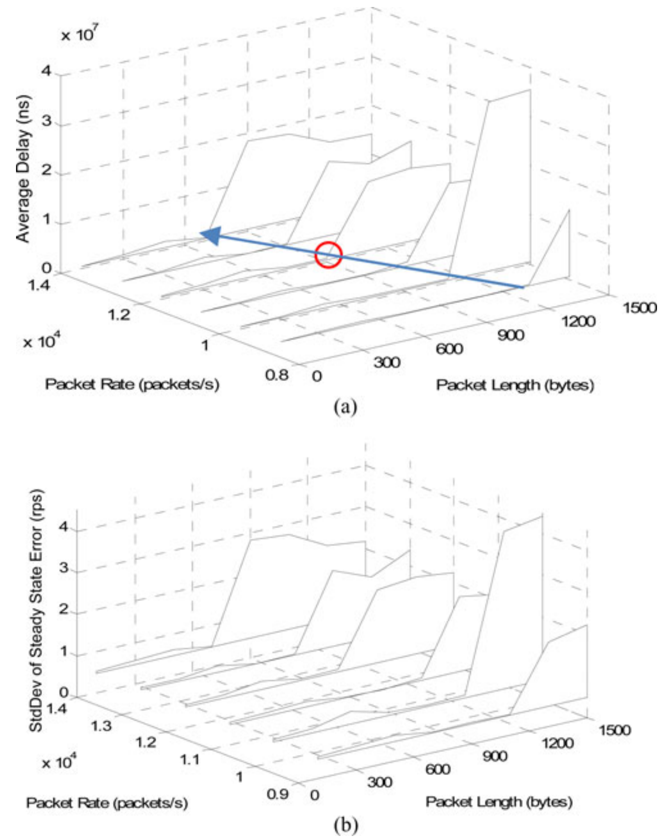


Fig. 9. SC–SS-1: (a) Average delay. (b) Standard deviation of SSE.

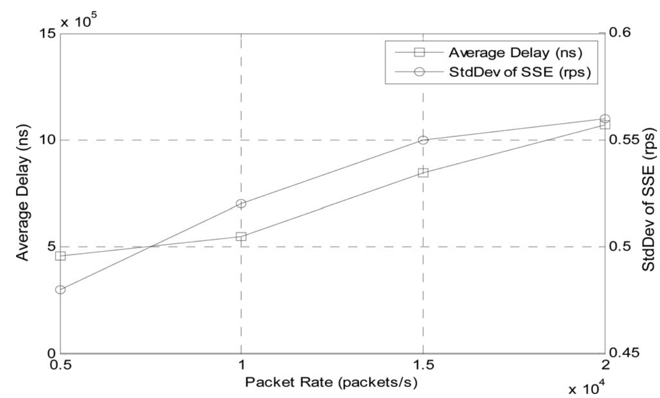


Fig. 10. SC–SS-2: Packet rate versus average delay (ns) and standard deviation of the SSE (rps), at constant packet length of 64 bytes.

points were generated in this test. This testing method verifies a scenario of non-real-time network utilization.

2) Increasing The Packet Rate While Keeping the Packet Length Constant: The packet generator was used to send packets at an increasing packet rate but a constant length of 64 bytes per packet is maintained. On every consecutive iteration, the packet rate was increased from 5000 to 20,000 packets/s in steps of 5000. Due to the physical limitation of the sending and receiving computers the maximum packet rate was fixed at 20,000 packets/s. This testing method verifies a scenario of real-time network utilization.

The two experiments were applied to each of the following three architectures, and Figs. 8–14 are labeled accordingly. For example, SC–SS-1 summarizes the result of Experiment 1 applied to the SC–SS architecture. A third type of experiments is carried out by introducing artificial packet losses that is explained in detail at the end of Section IV.

From the experiments it is found that the best results were observed to occur at higher link utilization and higher packet rates. So, a packet rate of 14,000 packets/s is selected to present the snapshot of performance of three architectures. Fig. 8 shows the average delay and SSE observations when the packet length increased from 100 bytes to 1500 bytes resulting in a gradual increase of link utilization as shown in Table I. In both SC–SS and MC–SS architectures, the average delay and SSE show a gradual increasing pattern up to 500-bytes packet length. As the packet length reaches 700-bytes representing a utilization of 80%, the $T_{\rm av}$ of Client 1 in the MC–SS architecture increases rapidly. At the maximum packet rate and packet length, the average packet delay in the MC–SS architecture shows an increase of 4989% from its initial delay. The SC–SS architecture also exhibited a similar drastic increase. It had only one client, however, its delay increased by 2231.8%. Although we kept increasing the packet length, the link utilization saturated at 97.4% because of the switch's flow control mechanism [14] as highlighted in Fig. 8(a) with arrows. This mechanism reduced packet loss, by forcing the hosts to decrease the packet rate as shown in the third column of Table I, to restrict the link utilization under the physical limit of 100 Mbps. On the other hand, it restricts the real-time communication between the plant and the controller, drastically increasing the packet delays and SSE of the system.

As shown in Table I, packet delays in the MC–MS architecture remained almost constant, exhibiting a slight increase of 1.1%. In this architecture, Server 1's link was congested because on one side it was serving Client 2 and on the other side it was also getting swamped by the packet generator. Client 1 by

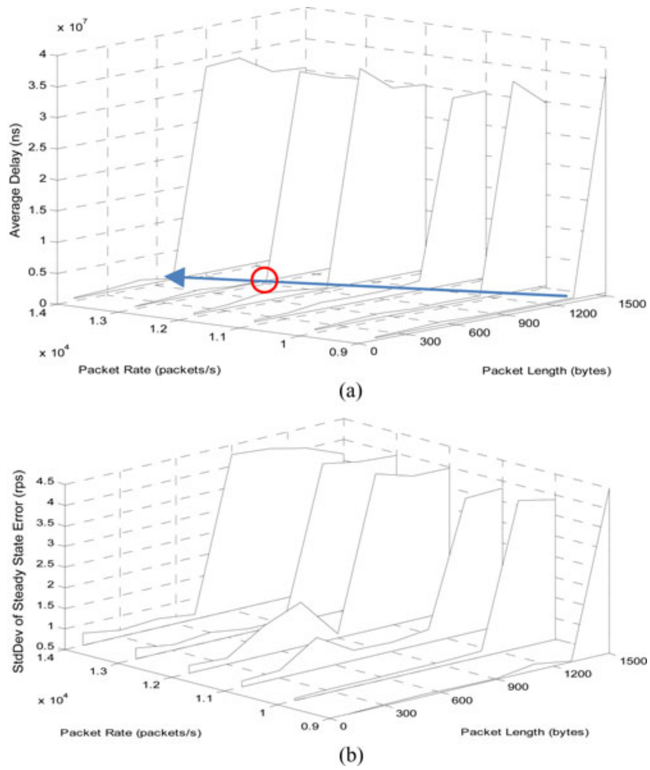


Fig. 11. MC–SS-1: (a) Average delay. (b) Standard deviation of SSE.

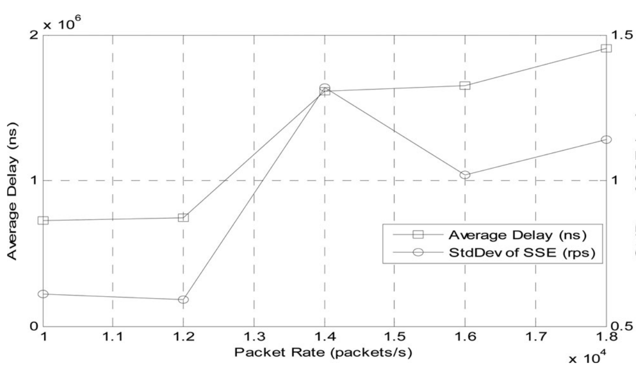


Fig. 12. MC–SS-2: Packet rate versus average delay (ns) and standard deviation of SSE (rps), at constant packet length of 64 bytes.

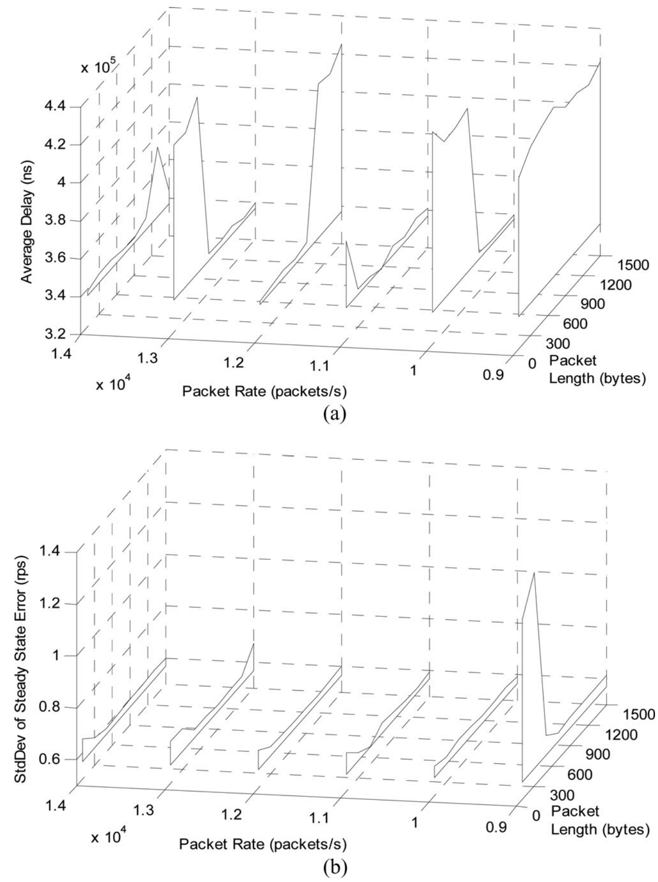


Fig. 13. MC–MS-1: (a) Average delay. (b) Standard deviation of SSE.

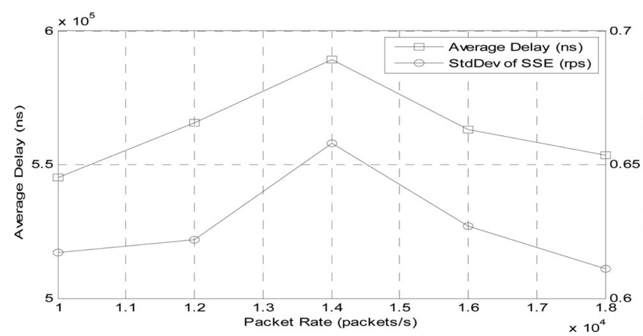


Fig. 14. MC–MS-2: Packet rate versus average delay (ns) and standard deviation of steady-state error (rps), at constant packet length of 64 bytes.

connecting to Server 2 avoided congestion, exhibiting very low packet delays and SSE. As listed in Table I, for a 3-ms sampling system, the average packet delay values at 14,000 packets/s and at 1500-bytes packet length for the MC–MS architecture (0.34 ms) is found to be 4073% (14.19 ms) less than the SC–SS architecture and 8841% (30.4 ms) less than the MC–SS architecture. As expected, the standard deviation of the SSE showed highest increase in the MC–SS architecture with a rise of 364% from its initial value. Due to a single client, the SC–SS architecture performed a little better than the MC–MS architecture with an increase of 304% from its initial value. Table II shows the observed average packet delay and SSE at an increasing packet rate from 5000 to 15,000 packets/s with the constant packet length of 64 bytes. It is clear from the observations that as the packet rate increases, the average delay also increases in both SC–SS and MC–MS architectures. At the final packet rate of 15,000 packets/s, the average packet delay of the client in the MC–SS architecture showed an increase of 281% from its initial delay value when compared to 86% increase in the SC–SS architecture.

The plant in the MC–SS architecture experienced the highest increase in the standard deviation of the SSE at 96.6%. The SSE in SC–SS architecture showed an increase of 14.5% compared to 4.9% increase in the MC–MS architecture. From the Table II, it is evident that MC–MS architecture proved to be advantageous with the lowest packet delays and SSE.

In the following, more detailed experimental observations specific to each architecture is presented.

TABLE I
AVERAGE DELAY AND SSE OBSERVATIONS FOR SC–SS-1, MC–SS-1, AND MC–MS-1

Link Utiliz- -ation (%)	Packet Rate (packets/s)	Packet Len- -gth (bytes)	Average Delay (ms)			StdDev (rps)		
			SC-SS	MC-SS	MC-MS	SC-SS	MC-SS	MC-MS
13.6	14000	100	0.60	0.59	0.33	0.53	0.78	0.65
35.6	14000	300	0.97	0.82	0.34	0.57	0.64	0.60
57.6	14000	500	1.86	1.34	0.34	0.62	0.72	0.59
79.7	14000	700	5.82	5.97	0.34	0.48	0.60	0.59
97.4	13500	900	18.48	33.63	0.33	2.84	4.26	0.60
97.6	11105	1100	18.03	34.01	0.34	2.74	4.20	0.60
97.6	9421	1300	14.47	30.70	0.37	2.27	4.00	0.60
97.6	8202	1500	14.19	30.40	0.34	2.17	3.66	0.59

TABLE II
AVERAGE DELAY AND SSE OBSERVATIONS FOR SC–SS-2, MC–SS-2, AND MC–MS-2

Link Utiliz- -ation (%)	Packet Rate (packets/s)	Average Delay (ms)			StdDev (rps)		
		SC-SS	MC-SS	MC-MS	SC-SS	MC-SS	MC-MS
13.6	5000	0.45	0.55	0.52	0.48	0.59	0.61
35.6	10000	0.55	0.72	0.54	0.52	0.61	0.61
57.6	15000	0.84	2.10	0.58	0.55	1.16	0.64
% Increase		86.6%	281%	11.5%	14.5%	96.6%	4.9%

A. SC–SS Architecture

Fig. 9 shows that the change in the standard deviation of the SSE is proportional to the change in average delay. As the packet length increases, all curves except that of 10,000 bytes packet length showed a consistent increase in average delay. The 10,000 packets/s curve shows unusual spikes and can be ignored. It can be observed that the spikes in the average delay and the standard deviation of SSE occurred at shorter packet rates because the increasing packet rate reached the threshold of Ethernet link utilization at shorter packet lengths. For example, the highlighted part in Fig. 9(a) shows that at 12,000 packets/s the spike starts to occur at 900-bytes packet length because at that point the corresponding Ethernet link utilization reaches 86.6%. The final values of average packet delay is observed to be similar for all the packet rates because once the link utilization exceeds 80%, the link becomes so congested that any further increase in the packet rate or the packet length has no effect on packet delay.

Fig. 10 presents a correlation between the packet rate and the average delay. As the packet rate increases, there is an almost linearly proportional increase in the average packet delay. Contrary to the observations in the test SC–SS-1, the average packet delay and SSE shows an always increasing trend because the link utilization is still 15% even at 20,000 packets/s. Theoretically, if the packet rate is increased further beyond 20,000 packets/s, the average packet delay and standard deviation in SSE will still show an increase till the link utilization hits the threshold of 80% after which the flow control gets activated. Fig. 10 shows that for a 274% increase in average delay from the initial packet rate to the final packet rate, the standard deviation of the SSE increased by 16.5%.

B. MC–SS Architecture

As shown in Fig. 11, the spikes in average delay for the MC–SS architecture occur at the same juncture as in the previous SC–SS architecture because the link utilization is found to hit the threshold at similar values. For example, the highlighted part in Fig. 11(a) shows that at 12,000 packets/s the spike starts to occur at 900-bytes packet length. The main difference between the single and multiclient scenarios is the magnitude of the delay. These values are almost twice that of those observed in the SC–SS architecture. This drastic increase can be attributed to the rapid increase of queues at the switch because of two clients and a packet generator swamping a single server.

Fig. 12 shows that as the packet rate increases, the average delay also increases. Theoretically, if the packet rate increases further beyond 18,000 packets/s, the average packet delay and standard deviation of SSE will always exhibit an increase till the link utilization hits the threshold of 80% after which the flow control gets activated. Fig. 12 shows that from 10,000 packets/s to 18,000 packets/s, an increase of 164% in average delay led to an increase of 85.3% in the standard deviation of SSE.

C. MC–MS Architecture

Fig. 13(a) and (b) show that in the MC–MS architecture, there is no effect of increasing packet lengths or packet rates on the average delay and SSE. This is because the client avoided the effects of congestion by connecting to an additional server. The 9000 packet/s curve can be ignored because it shows unusually higher delays and higher SSE. Fig. 14 shows that the highest average delay is observed to be 0.6 ms, which is equivalent to the default delay observed in a normal scenario without any high link utilization. These lower delays are due to the additional server overcoming the effects of a congested link by providing an alternate congestion-free client–server communication link. Although the standard deviation of SSE is observed to vary with the packet delays, the delay variations were found to be much lower to have any appreciable effect on the SSE.

3) Experimental Analysis Due to Random Packet Drops: From the aforementioned experimental results, the MC–MS architecture proved to be advantageous under high network utilization scenarios. In addition to the random delays, communication channels are generally noisy, and data transfer is subjected to packet loss and distortion. Tatikonda and Mitter [16] and Elia [17] discussed the effects of packet drop and distortion on the performance of an NCS. The campus network used in this experimental research is found to be robust, and no packet losses or distortion were observed during the experiments. To further test the MC–MS architecture, artificial packet losses were introduced. As the objective is to compare the performance of the MC–MS with the other two architectures during packet loss, no predictors or estimators were used. The packet losses can be divided into two types—control packet loss and measurement packet loss. Farhadi and Ahmed studied the problem of tracking nonlinear dynamic systems subject to noise in both process and measurement [18]. Control packet loss is implemented by dropping a control packet with a uniformly distributed random probability and applying hard control output loss. For the

TABLE III
PERFORMANCE OF THE SC–SS AND THE MC–SS ARCHITECTURES WITH PACKET DROPS

Link Utiliz-ation (%)	StdDev of SSE (rps)							
	SC-SS				MC-SS			
	Packet-Drop Probability							
	10%	20%	30%	40%	10%	20%	30%	40%
60	0.71	0.94	1.22	1.25	0.78	0.98	1.12	1.40
70	0.74	0.98	1.22	1.26	0.78	0.99	1.14	1.42
80	1.15	1.23	1.33	1.72	2.19	2.48	2.54	3.03
90	1.15	1.25	1.35	1.75	2.20	2.50	2.55	3.05
100	1.17	1.26	1.37	1.76	2.20	2.51	2.54	3.04

TABLE IV
PERFORMANCE OBSERVATIONS OF THE MC–MS ARCHITECTURE WITH PACKET DROPS

Link Utilization (%)	MC-MS - StdDev of SSE (rps)			
	Packet-Drop Probability			
	10%	20%	30%	40%
60	0.66	0.68	0.88	0.99
70	0.65	0.73	0.89	1.02
80	0.66	0.74	0.87	1.01
90	0.67	0.69	0.89	1.03
100	0.68	0.71	0.90	1.02

second type of experiments, the measurement packet loss is implemented by forcing the client to drop a sensor packet with a uniformly distributed random probability. The server responds to this sensor packet loss by sending the "previous" control output. We conducted the packet loss experiments in two sets. In the first set, control packet loss experiments were conducted on all the three NCS architectures by varying the probability of packet drops from 10% to 40% for each link utilization percentage ranging from 60% to 90%. In the second set, only the MC–MS architecture was further investigated by extending the packet drop probability to 90%. The experimental observations shown here represent an average of five samples.

As the percentage of packet drops increased, the standard deviation of SSE also increased. Both the SC–SS and MC–SS architectures in Table III showed a steep increase in the SSE as the utilization reached 80%. At 80% utilization and 40% packet drop, when compared to the MC–MS architecture, the standard deviation of SSE in the SC–SS architecture is 70.2% higher, and that of the MC–SS architecture is 200% higher. The MC–MS architecture proved to be advantageous again by showing the lowest increase in SSE compared to the other two.

To further investigate the MC–MS architecture, the control packet loss increased till 90%. As the MC–MS architecture was found to be unaffected by the utilization, experiments were conducted with the utilization fixed at 60%. Tables V–VII give a deeper insight of the transient characteristics of MC–MS architecture during packet loss. When the MC–MS architecture was subjected to control packet loss and measurement packet loss separately, it was able to maintain stability till 70% of packet loss. However, when it was subjected to both types of packet loss, it became unstable beyond 60% packet loss. When compared to the control packet loss scenario, the dc motor showed

TABLE V
PERFORMANCE OBSERVATIONS OF THE MC–MS ARCHITECTURE WITH CONTROL PACKET DROPS

MC-MS - Control Packet Loss							
Packet-Drop Probability							
Transient Response	10%	20%	30%	40%	50%	60%	70%
(%) Overshoot	4.20	5.69	6.66	6.90	7.20	15.20	35.09
Settling Time (ms)	60	63	81	69	90	144	201
Rise Time (ms)	21	21	23	24	27	30	24
StdDev of SSE (rps)	0.68	0.90	1.13	1.21	1.48	2.20	2.01

TABLE VI
PERFORMANCE OBSERVATIONS OF THE MC–MS ARCHITECTURE WITH MEASUREMENT PACKET DROPS

MC-MS - Measurement Packet Loss							
Transient Response	Packet-Drop Probability						
	10%	20%	30%	40%	50%	60%	70%
(%) Overshoot	1.38	3.30	3.50	2.90	4.20	4.27	28.09
Settling Time (ms)	48	57	51	57	54	63	135
Rise Time (ms)	24	24	25	26	27	30	36
StdDev of SSE (rps)	0.74	0.88	1.03	1.10	1.48	1.71	2.40

TABLE VII
PERFORMANCE OBSERVATIONS OF THE MC–MS ARCHITECTURE WITH BOTH CONTROL AND MEASUREMENT PACKET DROPS

MC-MS - Control and Measurement Packet Loss						
Transient Response	Packet-Drop Probability					
	10%	20%	30%	40%	50%	60%
(%) Overshoot	3.23	3.69	4.29	9.20	9.10	21.57
Settling Time (ms)	57	63	66	81	99	141
Rise Time (ms)	27	27	27	36	33	36
StdDev of SSE (rps)	1.05	1.11	1.32	1.69	2.07	2.63

a better performance in the measurement packet loss scenario because it always received a control packet.

In the MC–SS architecture, the effects of packet delays, packet loss, and load on the network due to higher number of clients can be overcome by the use of predictors and estimators. Walsh *et al*. proposed a new protocol "Try-Once-Discard" that deals with dynamic scheduling of network resources based on the need [19].

V. CONCLUSIONS

Various experiments with real-time and non-real-time network load and packet losses were conducted on the MC–MS architecture that we developed in this research. It is found that, having an additional server in a switched Ethernet network showed 300% to 364% smaller standard deviation of SSE compared to that of the SC–SS and MC–SS architectures, respectively. It also overcame the negative effect of ethernet's flow control mechanism on real-time communication. Overall, the

MC–MS architecture performed better than other architectures because it could overcome the effects of network load and was unaffected by the packet loss. The developed experimental setup will be used to address fundamental research questions includes dynamic and optimal resource allocation and Markov-chain-based output feedback method for stabilization of networked control systems with random time delays and packet losses.

As long as the load on the network is kept below the maximum threshold of link utilization, the NCS showed excellent performance. However, in a real-time scenario a 100-Mbps high speed-ethernet network is seldom loaded to its maximum capacity. Based on the earlier observations, it is evident that switched ethernet holds a great potential for possible applications in factory automation.

References

[1] Y. Li, L. Yang, and G. Yang, "Network-based coordinated motion control of large-scale transportation vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 2, pp. 208–215, Apr. 2007.

[2] A. Paraskevopoulos, J. Vucic, S. Voss, R. Swoboda, and K. Langer, "Optical wireless communication systems in the Mb/s to Gb/s range, suitable for industrial applications," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 4, pp. 541–547, Aug. 2010.

[3] J. P. Thomesse, "Fieldbus technology in industrial automation," *Proc. IEEE*, vol. 93, no. 6, pp. 1073–1101, Jun. 2005.

[4] F. L. Lian, J. R. Moyne, and D. M. Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," *IEEE Control Syst. Mag.*, vol. 21, no. 1, pp. 66–83, Feb. 2001.

[5] T. Y. Mazraani and G. M. Parulkar, "Performance analysis of the ethernet under conditions of bursty traffic," in *Proc. IEEE Global Telecommun. Conf.*, Orlando, FL, 1992, vol. 1, pp. 592–596.

[6] K. Walker, Y. Pan, and J. Gu, "Bilateral teleoperation over networks based on stochastic switching approach," *IEEE/ASME Trans. Mechatronics*, vol. 14, no. 5, pp. 539–554, Oct. 2009.

[7] K. Lee and S. Lee, "Performance evaluation of switched ethernet for real-time industrial communications," *Comput. Standards Interfaces*, vol. 24, no. 5, pp. 411–423, Nov. 2002.

[8] Maxon motor, [Online] Available: http://www.maxonmotorusa.com/product_overview_ details_maxon_a-max.html [Aug. 2010]

[9] *DC Motor Closed-Loop Speed Control, MEEN 667 Laboratory Manual,* Dept. Mech. Eng., Texas A&M Univ., TX, 2008.

[10] W.-J. Kim, Kun Ji, and A. Ambike, "Real-time operating environment for networked control systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 3, pp. 287–296, Jul. 2006.

[11] L. Dozio and P. Mantegazza, "Real time distributed control systems using RTAI," in *Proc. 6th IEEE Int. Symp. Object-Oriented Real-Time Distributed Comput.*, Hokkaido, Japan, May 2003, pp. 11–18.

[12] D. Schleef, F. Hess, H. Bruyninckx Comedi. (2010 Aug.). Linux control and measurement device interface, [Online] Available: http://www.comedi.org

[13] D. R. Boggs, J. C. Mogul, and C. A. Kent, "Measured capacity of an ethernet: Myths and reality," *ACM Special Interest Group on Data Commun. Comput. Commun. Rev.*, vol. 18, no. 4, pp. 222–234, Aug. 1988.

[14] O. Feuser and A. Wenzel, "On the effects of the IEEE 802.3× flow control in full-duplex ethernet LANs," in *Proc. 24th Annu. IEEE Conf. Local Comput. Networks*, Tampa, FL, Oct. 1999, pp. 160–168.

[15] M. Jemec. (2010 Aug.). *packETH - ethernet packet generator,* [Online] Available: http://packeth.sourceforge.net

[16] S. Tatikonda and S. Mitter, "Control over noisy channels," *IEEE Trans. Autom. Control*, vol. 49, no. 7, pp. 1196–1201, Jul. 2004.

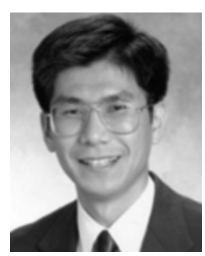
[17] N. Elia, "Remote stabilization over fading channels," *Systems Control Lett.*, vol. 54, no. 3, pp. 237–249, Mar. 2005.

[18] A. Farhadi and N. U. Ahmed, "Tracking nonlinear noisy dynamic systems over noisy communication channels," *IEEE Trans. Commun.*, vol. 59, no. 4, pp. 955–961, Apr. 2011.

[19] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 3, pp. 438–446, May 2002.

Naveen Bibinagar received the dual B.Tech. degree in mechanical engineering and information technology from Vellore Institute of Technology, Vellore, India, in 2005. He received the M.S. degree in mechanical engineering from Texas A&M University, College Station, TX, in 2010.

He was a Programmer Analyst at Cognizant Technology Solutions, Hyderabad, India, for two years. He is currently with CCI, Austin, TX, as a Network Software Developer. His research interests include networked control systems, real-time computing, and networking.



Won-jong Kim (S'89–M'97–SM'03) received the B.S. (*summa cum laude*) and M.S. degrees in control and instrumentation engineering from Seoul National University, Seoul, Korea, in 1989 and 1991, respectively, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1997.

Since 2000, he has been with the Department of Mechanical Engineering, Texas A&M University, College Station, where he is currently an Associate Professor and was the inaugural Holder of the Dietz Career Development Professorship II in 2007–2010. He is a Technical Editor of the IEEE/ASME Transactions on Mechatronics, *ASME Journal of Dynamic Systems, Measurement and Control, International Journal of Control, Automation, and Systems*, and *Asian Journal of Control*. He is the holder of three U.S. patents on precision positioning systems. His current research interests include the analysis, design, and real-time control of mechatronic systems, networked control systems, and nanoscale engineering and technology.

Dr. Kim is Fellow of the American Society of Mechanical Engineers and a member of Pi Tau Sigma.