Design of a Scalable Real-Time Robot Controller and Application to a Dexterous Manipulator

Paul Thienphrapa and Peter Kazanzides

***Abstract*— Research in surgical robots often calls for multi-axis controllers and other I/O hardware for interfacing various devices with computers. As the need for dexterity is increased, the hardware and software interfaces required to support additional joints can become cumbersome and impractical. To facilitate prototyping of robots and experimentation with large numbers of axes, it would be beneficial to have controllers that scale well in this regard.**

This paper discusses the design of a real-time (one kilohertz) robot controller based on a centralized processing, distributed I/O architecture. We combine powerful yet accessible real-time technologies such as IEEE 1394 (FireWire) and low-latency field programmable gate arrays (FPGAs). The device is developed and used with a real-time operating system, and scalability is demonstrated on a novel snake-like surgical manipulator. Results on a 21-axis prototype suggest that the proposed solution can help increase the viability of complex robots, particularly in education and research. In that spirit, the robot control software libraries have been released as open source, and efforts are underway to release the electronic designs.

I. Introduction

A. Background

Minimally invasive surgery (MIS) is often beneficial for patients due to reduction of trauma, leading to fewer complications and shorter hospital stays. However, MIS poses a number of challenges for surgeons, including constrained workspaces, limited field of view, and lack of dexterity at the distal end. These challenges remain despite the availability of manual MIS-specific instruments, in part because these instruments are rigid, difficult to manipulate through narrow insertion tubes, and lack adequate suturing and tissue reconstruction capabilities. In such situations, the efficacy of a surgical robot is strongly tied to its dexterity.

Research on the Snake Robot [1] seeks to improve MIS of the throat and upper airways by providing surgeons with highly dexterous robotically-controlled tools. This dexterity is achieved by incorporating more degrees of freedom (dof). More sophisticated surgical tasks can be accomplished by increasing dof, but the corresponding hardware increase imposes a practical limit on the exploration of these ideas. Similarly, research on different types of multi-axis surgical robots is often mired in the hardware construction effort. In response to these difficulties, this paper presents the development of a system that is well suited for real-time control of robots with many axes of control.

Manuscript received June 29, 2011. This work was funded by the National Science Foundation (NSF) under Engineering Research Center grant #EEC9731748, NSF grant #MRI-0722943, and internal funds. Paul Thienphrapa is supported by a Philips Research North America fellowship.

Authors are with the Department of Computer Science and ERC CISST, Johns Hopkins University, Baltimore, MD, USA.

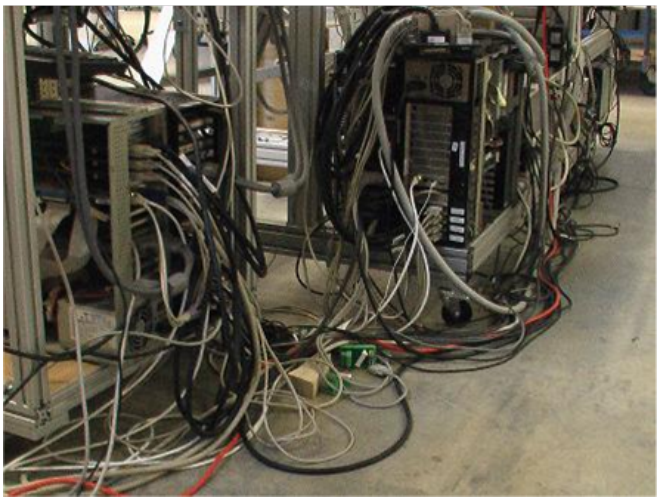

Fig. 1. Under centralized I/O, conversions are performed on boards attached to the local computer bus, resulting in a complex cabling scenario.

B. Motivations and Objectives

The efforts presented here and in [2], [3], [4] originate in part from the need to improve upon the old multi-axis controller [5] of the Snake Robot and replicate the controller for other research projects. The unique requirements of the robot necessitated a custom solution [5]. The original controller utilizes a centralized I/O arrangement, whereby command and feedback signals are transmitted in raw analog form over long cables running between the robot and the computer. The I/O devices reside on custom circuit boards, which in turn are directly attached to the computer via the ISA bus. Though the design is conceptually straightforward, the wiring associated with it introduces complications such as noise, cable drag, reduced reliability, and greater construction effort. The debug space is vast as there are many candidates for connectivity problems. These issues are captured in Fig. 1.

Prompted by these motivations, we have developed a real-time control system that takes advantage of a high-speed serial bus, IEEE 1394, and capitalizes on the processing power of contemporary computers. The benefits of using this architecture are multifold. The centralized processing of computational tasks eases software development efforts, and a standard API is developed to exploit this benefit. A high speed serial link is capable of multiplexing a large number of I/O data streams between the computer and low-latency field-programmable gate arrays (FPGAs); this distribution of I/O replaces long, dense bundles of analog cables with a single communication channel, thereby mitigating common

sources of failure associated with signal integrity and cable complexity. We selected IEEE 1394 among other viable candidates because it suited our purposes well [3].

This work is ultimately intended to provide an abstract interface and scalable mechanisms for highly configurable, fine-grain, real-time controllers, in order to simplify the development of dexterous surgical robots from both hardware and software perspectives. In an effort to disseminate these benefits, we are making the electronic designs publicly available once mature, while the robot control libraries [6] have already been released as open source software. Particularly for education and research environments, this would enable the exploration of complex surgical systems and technologies by allowing for more dexterity, reliability, and scalability.

II. Related Work

Though based on IEEE 1394, the works of [7] and [8] focus on real-time control bandwidth and not on the physical benefits of distributed I/O and centralized processing. The differences manifest in their use of IEEE 1394 as a link to an onboard computer, contrasting with our use of compact custom electronics. Our work is most similar to that of [9], where custom FPGA-based I/O boards communicate with the computer over IEEE 1394. The bandwidth was sufficient for at least six (possibly 12) dof to be updated at 1 kHz, with unit delay latency. On the other hand, the current study emphasizes scalability, performance, and the physical benefits of the architecture. Ref [10] notes that using IEEE 1394 for high bandwidth PET scan data acquisition is viable due to the availability of powerful commodity computers. We agree in principle, though our respective applications are fundamentally different.

A. Ethernet-Based Alternatives to IEEE 1394

Fair bus access is incorporated into IEEE 1394 hardware; bus arbitration in Ethernet is nondeterministic, but kilohertz-range motor control is achievable on isolated networks with software modifications [11], [12]. Several Ethernet variations have been developed that make the medium very promising. Powerlink [13] employs a bus manager that schedules 200-μs cycles of isochronous and asynchronous phases. SERCOS approached a communication bottleneck in [14] with increasing axes and cycle rates, but its recent combination with Ethernet (SERCOS-III) has endowed it with the ability to update 70 axes every 250 μs.

A relative newcomer, EtherCAT [15] is an attractive protocol in which the nodes forward and append packets on-the-fly using dedicated hardware and software, resulting in the ability to communicate with 100 axes in 100 μs; [16] is an example showing its potential.

B. Other Alternatives to IEEE 1394

Many of the themes highlighted in this paper, including distributed I/O, centralized computing, scalability, and form factor, echo those of [17], which documents the MIRO surgical robot developed by the German Aerospace Center (DLR). Scalability in the MIRO robot is aided by the use of SpaceWire, a 1 GB/s full duplex serial link with latency less than 20 μs. Whereas SpaceWire has been developed by major international space agencies for space-borne systems, we prefer IEEE 1394 as it is a more accessible protocol for research, and its performance is more than adequate for satisfying our requirements. We are particularly more interested in the software-induced latency and overcoming this latency to enhance scalability.

PCI Express is a fairly new serial interface designed to replace computer expansion buses; a cable-based standard was not fully established at the time of the designs presented in this paper. PCI Express supports real-time applications such as the industrial control example in [18]. High data rates are readily available with USB, but its reliance on the host processor for bus level tasks compromises its scalability in real-time control. Conversely, IEEE 1394 self-manages the bus at the physical layer. The Controller Area Network (CAN) [19] bus is well-suited for real-time control and has been widely used, but its bandwidth is limited to 1 Mbps. Though not a serial bus, CompactPCI is an industrial backplane interface capable of 132 MB/s throughput, used notably by NASA in transitioning from VMEbus [20].

III. Centralized Processing, Distributed I/O

Robot systems are concurrent by nature: multiple joints must be controlled simultaneously, and there is often a hierarchy of control strategies. A typical robot controller contains "loops" for servo control, supervisory (e.g., trajectory) control, and the application.

In the early days of robotics, controllers consisted of a central computer with a rack of joint-level control boards on a parallel bus, such as ISA, Q-Bus, Multibus, or VME; this was necessary for performance reasons, as computers and networks were slow. Although the joint-level boards usually contained embedded processors, this architecture can be characterized as centralized processing and I/O. This approach does not extend well as the number of control axes increases because the requisite cabling can become unwieldy.

With the emergence of high-speed serial networks, such as CAN, Ethernet, USB, and IEEE 1394, it became possible to physically distribute the joint controller boards and associated power amplifiers. By placing these components inside the robot arm, or at its base, significant reductions in cabling could be achieved. Thick cables containing multiple wires for motor power and sensor feedback could be replaced by thin network and power cables. These types of systems can be characterized as distributed processing and I/O.

Given the recent advances in processor performance, especially the move to multi-core architectures, coupled with the extraordinarily high data rates of modern serial networks, we advocate a different approach for robot control: centralized processing and distributed I/O (Fig. 2). This can be achieved by replacing the microprocessors with FPGAs that provide direct, low-latency interfaces between the high-speed serial network and the I/O hardware. As mentioned, this preserves the advantages of reduced cabling while allowing all software to be implemented on a single high performance computer

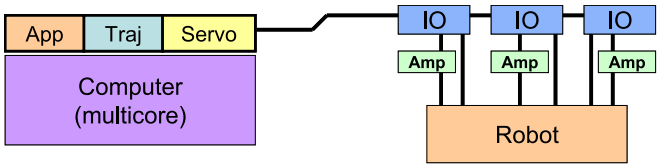

Fig. 2. Centralized processing, distributed I/O architecture.

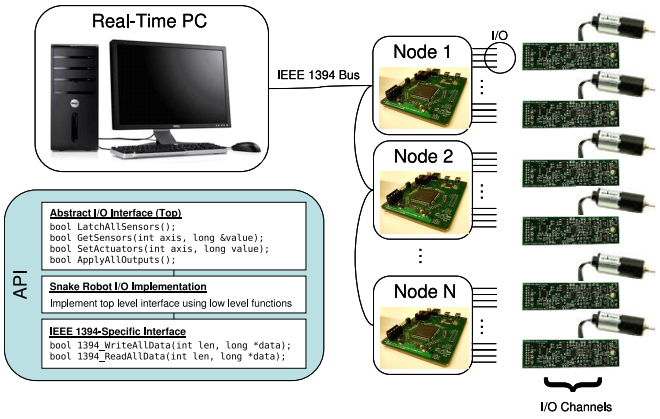

Fig. 3. Overview of the control system; I/O conversions take place near the actuator sites, linked to the computer via an IEEE 1394 bus.

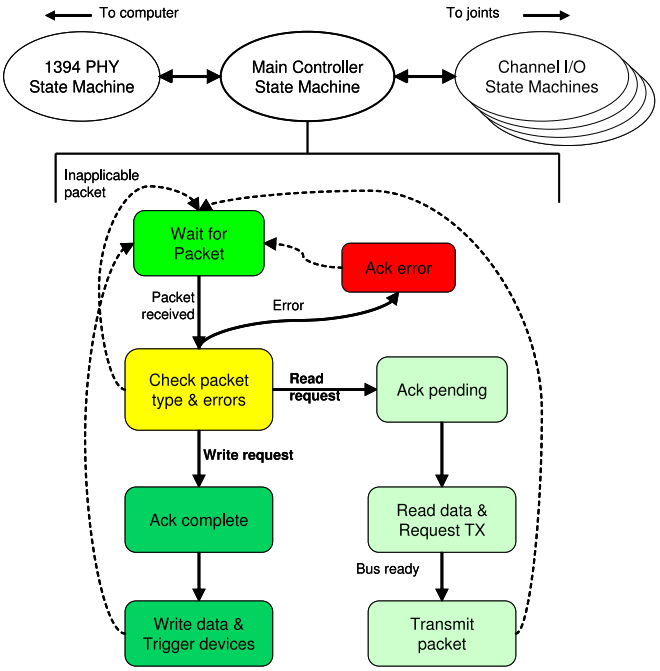

Fig. 5. FPGA structure and operation.

with a familiar software development environment. This frees developers from having to learn the idiosyncrasies of various embedded microprocessors. Another key factor is the availability of low-cost real-time operating systems, such as those based on Linux.

IV. System Description

Fig. 3 provides an overview of the control system. Each node on the IEEE 1394 bus contains multiple axes of control. Nodes can be daisy-chained or directly connected to the computer. The bus is attached to a real-time computer that reads feedback signals from the robot and generates corresponding actuation commands for each axis. The controller is shown in Fig. 4 installed on the Snake Robot.

A. Electronics Design

1) Amplifier Section: The amplifier section, which we refer to as an axis or I/O channel depending on context, contains the power amplification and I/O required to control one dc motor. I/O signals include a motor enable line as well as an op amp fault line to indicate overheating. Other components include an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), and a digital potentiometer. The dual-channel ADC digitizes analog feedback, namely the potentiometer voltage and motor current. The DAC provides motor speed and current limit commands. The board passes incremental encoder pulses to the FPGA in the digital section; these are decoded by the FPGA to compute relative displacement and velocity.

The motor current feedback gain can be programmed through the digital potentiometer and the operation mode (speed or torque control) can be set via a digital output. This flexibility allows for adaptation of the board to robots with different low-power motors.

2) Digital Section: The digital section, which we often refer to as a node following IEEE 1394 parlance, contains circuitry for accessing the I/O channels and handling bus transactions. Its core components are an FPGA and a two-port IEEE 1394a physical layer chip.

IEEE 1394 allows up to 63 nodes per bus. The maximum number of channels one node can accommodate is governed largely by its resident FPGA. The low-cost Altera Cyclone II EP2C8Q208C7N is comfortably able to control a Snake Robot, though we have subsequently moved toward more powerful, moderately priced devices.

3) Firmware Section: Most of the functionality of each node is implemented as firmware on the FPGA. The state machine in Fig. 5 describes the top level operation. Action is initiated when the computer requests a data transaction, for instance a sensor read or an actuator write. The FPGA on the addressed node responds immediately with

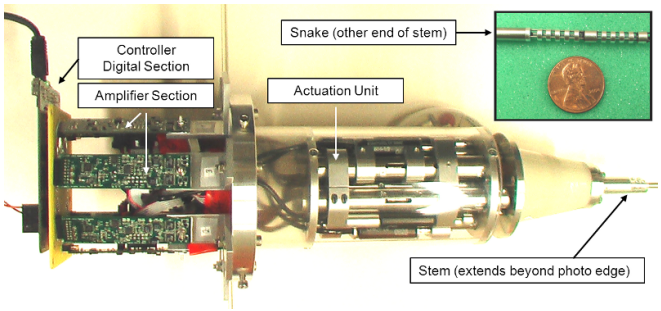

Fig. 4. Completed controller mounted on the robot actuation unit.

an acknowledgment, per the IEEE 1394 protocol. For read requests, the FPGA then fetches data from a continuously refreshed buffer and sends them to the computer with a timestamp. For write requests, it loads the appropriate buffers and triggers the I/O devices. These *concatenated read* and *unified write* transactions execute atomically and work well for real-time control. This is in contrast to the delayed and split transactions also supported by the protocol.

I/O operations for all channels, such as encoder counting and Serial Peripheral Interface communication (for the ADCs, DACs, and digital pots), run in parallel and are accessed by the higher level part of the firmware that also manages bus transactions.

B. Application Programming Interface

We developed an application programming interface (API) for the control system described above. The interface has the three-layer hierarchy shown in Fig. 6. The top layer consists of abstract I/O operations that can be used for different robots. It includes commands to latch all sensors and to apply all outputs (e.g., for double-buffered DACs). Read and write operations access one axis at a time, so only primitive C data types (e.g., int, long) are needed.

The next layer down implements the abstract interface for a particular robot. We previously measured the bus speed [4] and found it to be plentiful for many axes of control, but software-induced latency is a key concern, as in [21]. To overcome this latency (about 35 μs per transaction [2], [3]), the middle API layer for the Snake Robot is customized to bundle data for all axes of a node into a single bus transaction. This layer provides access to individual axes via local buffers that are filled by `LatchAllSensors` and emptied by `ApplyAllOutputs`. So that a fixed block size can be used to simplify the FPGA implementation, this layer maintains a valid bit for each axis that tells the FPGA whether or not to write the corresponding axis. The bottom layer contains function calls to the IEEE 1394 API library (libraw1394); RT-FireWire [8] is an alternative of interest.

To aid software development using this API, it is being integrated into the *cisst* libraries [22], [6]. These libraries contain resources that ease the development of robotic systems, which typically consist of concurrent and interacting devices, processes, and data streams (as mentioned in Section III). Software for the Snake Robot is based on these libraries.

C. Use Case: The Snake Robot

A unique design targeted for MIS of the upper airways, the Snake Robot features a teleoperated dexterous end effector that appears at the distal end of a narrow (4 mm), meter-long stem through which the actuation wires run; the *actuation unit* (AU) is located at the proximal end of the stem.

The end effector, or *snake-like unit* (SLU, Fig. 7), is constructed using four superelastic NiTi backbones. The central *primary* backbone is surrounded by three parallel *secondary* backbones at equally-spaced radial distances and angles. All four backbones are fixed to the distal end disc, while only the primary backbone is fixed to the proximal

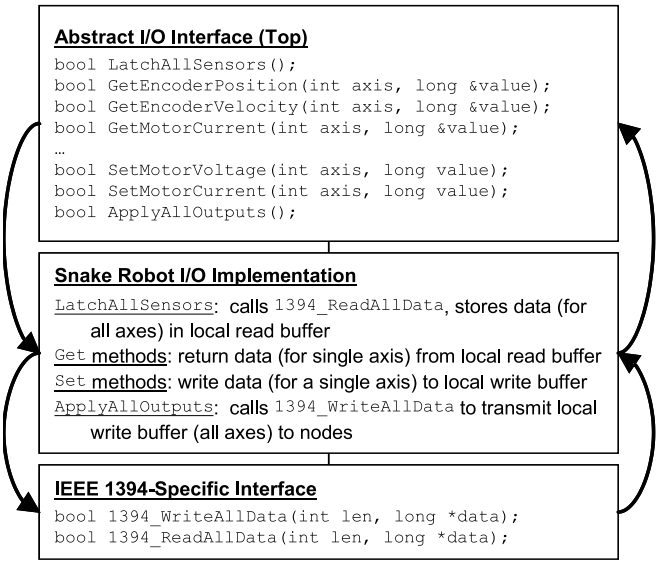

Fig. 6. Robot control API methods and hierarchy.

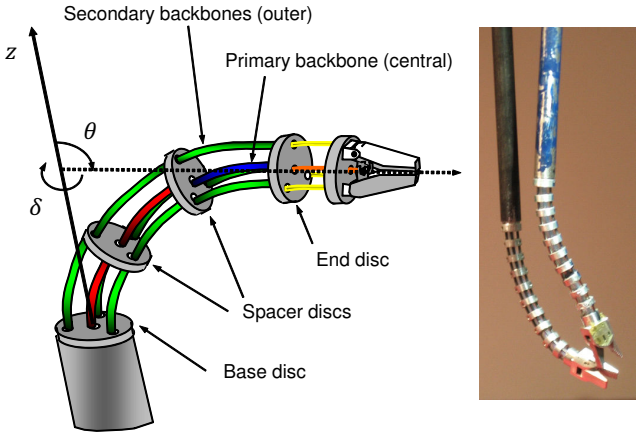

Fig. 7. Anatomy of a snake-like unit (*left*) and photo (*right*) [5].

base disc. The secondary backbones are free to glide through holes in the intermediate spacer discs.

When straight, the primary backbone is aligned with the robot z-axis. The angle of bend *from* the z-axis is described by θ, and the direction of bend (i.e. the angle *about* the z-axis) is described by δ. These two degrees of freedom are controlled by pushing and pulling the secondary backbones.

In the Snake Robot, two 4.2 mm diameter SLUs are connected in series for a total length of about 40 mm, achieving four degrees of freedom via six actuators (three per SLU). Including gripper actuation, the AU contains a total of seven control axes. In a separate setup, the AU itself is teleoperable in four degrees of freedom (three translations and a rotation about the stem) for coarse positioning. A second Snake and a pair of seven-dof da Vinci master controllers are integrated to create a 36-axis bimanual manipulator platform. The robot, as well as analysis of the push-pull actuation modes, is described in greater depth in [1], [5], [23], [24].

V. Experiments and Results

Robot control software runs on an RTAI-patched Linux PC with a 2 GHz Pentium 4 processor and 512 MB of RAM. At the trajectory level, the software accepts six-dof Cartesian configurations and computes the inverse kinematics in 125 Hz cycles. The resulting joint values are transferred to the servo level, which updates the actuator positions at 1 kHz (nominal). The algorithm follows the constrained optimization approach described in [24], [25].

The optimization framework implemented as part of the robot control software allows us to run simplified experiments without having to reduce the complexity of the original algorithm. Given that the end effector can move with only four degrees of freedom, orientations within the dexterous workspace can be ensured at *some* position by adjusting the weights of the least squares minimization problem. The tip orientation is given by the forward kinematics in (1) [1].

$$R_{tip} = \begin{bmatrix} c^2\delta(s\theta-1)+1 & -s\delta c\delta(s\theta-1) & c\theta c\delta \\ -s\delta c\delta(s\theta-1) & -c^2\delta(s\theta-1)+s\theta & -c\theta s\delta \\ -c\theta c\delta & c\theta s\delta & s\theta \end{bmatrix} \quad (1)$$

Here, θ and δ are the angles illustrated in Fig. 7 and s and c are shorthand for *sin* and *cos*. By equating (1) to the yaw-pitch-roll representation of a rotation matrix, we find an expression for the orientation in terms of yaw (γ), pitch (β), and roll (α) angles. Then solving for α, we obtain

$$\alpha = \tan^{-1}\left(\frac{\sin\beta\sin\gamma}{\cos\beta+\cos\gamma}\right). \quad (2)$$

In one test we measure the robot path corresponding to an input path, a composition of yaw and pitch varying sinusoidally ($\pm 45^\circ$, 0.1 Hz, in phase). Because the roll angle is a function of yaw and pitch and varies slowly, only α needs to be shown. As a reference, γ and β correspond to rotations about the robot y- and x-axes respectively, and the z-axis is aligned with the stem as mentioned above. Also listed are servo loop duty cycles (busy periods), measured by pulsing the parallel port and capturing the signals on an oscilloscope.

A. Number of Nodes

To determine the scalability of the system, the test input path is applied with one, two, and three nodes connected. Though the extra nodes do not drive mechanical components, the behavior of an n-node, $7n$-axis system is faithfully represented from an I/O latency standpoint. The resulting paths are plotted in Figs. 8a and b. There appears to be no effect from increasing nodes, likely because the extra overhead does not lead to violated timing requirements.

The times listed in Table I show that the duty cycle of the servo loop increases with the number of nodes. The computer can write all nodes with a single broadcast packet, so the time increase is due primarily to extra read transactions (computation and transfer times for additional axes are negligible, on the order of a few microseconds total [4]). A rough extrapolation based on an additional 40 μs of latency per node suggests that this specific system can accommodate up to six nodes while running at a 50% duty cycle. This figure may be increased by improving upon the naive implementation of read transactions.

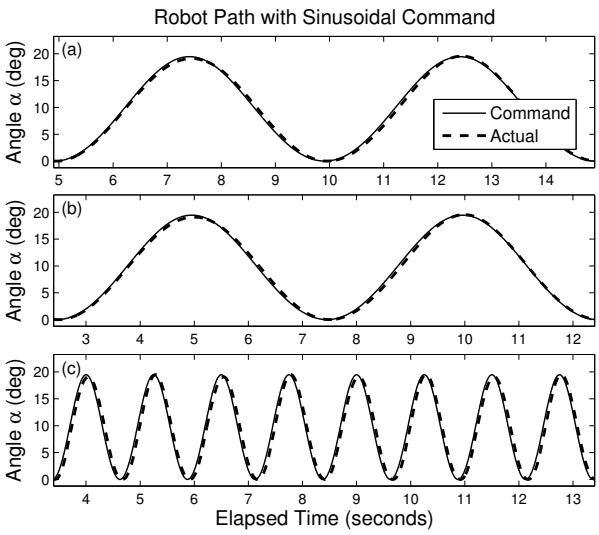

Fig. 8. Path of the robot with test input and 1 kHz servo rate. (a) With one node (n=1). (b) With n=3. (c) With n=1 and a 4x faster input. There are no noticeable differences between 1-3 nodes, while a slight lag in (c) is visible. Time axes have been shifted and truncated (from 80 s) for comparison.

To illustrate the dominance of the robot's mechanical bandwidth over the controller latency, the result of a faster sinusoidal command (0.4 Hz) is shown in Fig. 8c. There is a delay of about 60 ms, whereas no delay is discernible in the plots above it.

TABLE I
Servo Loop Duty Cycles

# of Nodes (At 1 kHz)	
Nodes	Duty Cycle
1	298 μs – 30%
2	334 μs – 33%
3	356 μs – 36%

Servo Rates (One Node)	
Rate	Duty Cycle
800 Hz	288 μs – 23%
1000 Hz	298 μs – 30%
1250 Hz	290 μs – 36%
2000 Hz	294 μs – 59%

B. Servo Rates

We test the performance of the system at various servo rates, using one node, to determine its configurability in this scenario. For servo rates of 800, 1000, 1250, and 2000 Hz, the maximum error of the robot position from the command is 6.9×10^{-3} degrees in each case under the test input. The paths are similar to those shown in Figs. 8a and b and are thus omitted. These results suggest that the controller is able to run at different rates without affecting functionality. The servo loop duty cycles are listed in Table I. Bus transaction latencies prescribe an upper limit on the servo rate. In keeping the duty cycle at a safe margin (roughly 50%), the robot appears to work as intended.

C. Teleoperation

A teleoperation setup is used to evaluate the performance of the control system under realistic conditions. Position

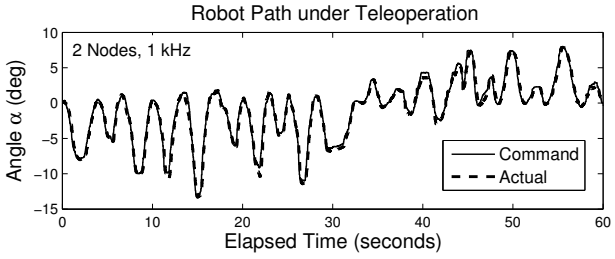

Fig. 9. Path of the robot teleoperated by a PHANTOM Omni.

commands from a PHANTOM Omni haptic device are sent over a local area network to the robot computer. The robot of Fig. 4 is used. To expand this test, two nodes (14 control axes) are used; the servo rate is set at 1 kHz.

Fig. 9 displays the path taken by the robot in response to arbitrary motion commands from the Omni over a period of 60 seconds. The system appears to work as intended, except where abrupt changes in the path are commanded; we suspect these are due to the limitations of the robot itself.

VI. Conclusions and Future Work

We presented a scalable robot controller design that uses a high-speed serial network, IEEE 1394, as the communication link between the computer and actuated joints. We described this centralized computation, distributed I/O approach, discussed the technologies that enable it, and outlined its advantages over traditional control schemes.

Real-time performance of the controller was demonstrated by application to the Snake Robot, a dexterous multi-axis robot designed for MIS of the throat. This approach is feasible for real-time control with rates up to several kilohertz; higher rates such as 10 kHz appear to be challenging with the current setup based on the measured latencies.

Future work includes implementing efficient read transactions for multiple nodes, and making the API compatible with the Surgical Assistant Workstation (SAW), a medical robotics framework currently in development [26].

Acknowledgment

Authors thank Hamid Wasti of Regan Designs, Inc. (Coeur d'Alene, Idaho) for his help with the design and layout of the controller boards, Ankur Kapoor for sharing his knowledge and experience, Nabil Simaan for his help with the robot prototype, and Russell Taylor for his advisement.

References

[1] N. Simaan, R. Taylor, and P. Flint, "A dexterous system for laryngeal surgery," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Apr 2004, pp. 351–357.

[2] P. Thienphrapa and P. Kazanzides, "A distributed I/O low-level controller for highly-dexterous snake robots," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Nov 2008, pp. 9–12.

[3] P. Kazanzides and P. Thienphrapa, "Centralized processing and distributed I/O for robot control," in *IEEE Int. Conf. on Technologies for Practical Robot Applications (TePRA)*, Nov 2008, pp. 84–88.

[4] P. Thienphrapa and P. Kazanzides, "A scalable system for real-time control of dexterous surgical robots," in *IEEE Int. Conf. on Technologies for Practical Robot Applications (TePRA)*, Nov 2009, pp. 16–22.

[5] A. Kapoor, N. Simaan, and P. Kazanzides, "A system for speed and torque control of DC motors with application to small snake robots," in *IEEE Int. Conf. on Mechatronics and Robotics*, Sep 2004.

[6] The *cisst* Libraries Homepage, https://trac.lcsr.jhu.edu/cisst.

[7] M. Sarker, C. Kim, S. Baek, and B.-J. You, "An IEEE-1394 based real-time robot control system for efficient controlling of humanoids," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2006, pp. 1416–1421.

[8] Y. Zhang, B. Orlic, P. Visser, and J. Broenink, "Hard real-time networking on FireWire," in *RT Linux Workshop*, Nov 2005.

[9] G. Pratt, P. Willisson, C. Bolton, and A. Hofman, "Late motor processing in low-impedance robots: impedance control of series-elastic actuators," in *American Control Conference*, Jun 2004, pp. 3245–3251.

[10] T. Lewellen, C. Laymon, R. Miyaoka, K. S. Lee, and P. Kinahan, "Design of a FireWire based data acquisition system for use in animal PET scanners," in *IEEE Nuclear Science Symposium Conference Record*, Nov 2001, pp. 1974–1978.

[11] S. Schneider, "Making Ethernet work in real time," *Sensors Magazine*, vol. 17, no. 11, Nov 2000.

[12] J. Kerkes, "Real-time Ethernet," *Embedded Systems Design*, vol. 14, no. 1, Jan 2001.

[13] Ethernet Powerlink, http://www.ethernet-powerlink.org/.

[14] S.-Y. Lin, C.-Y. Ho, and Y.-Y. Tzou, "Distributed motion control using real-time network communication techniques," in *Int. Power Electronics and Motion Control Conference (IPEMC)*, Aug 2000, pp. 843–847.

[15] EtherCAT Technology Group, http://www.ethercat.org/.

[16] S. G. Robertz, K. Nilsson, R. Henriksson, and A. Blomdell, "Industrial robot motion control with real-time Java and EtherCAT," in *IEEE Conf. on Emerging Technologies & Factory Automation (ETFA)*, Sep 2007, pp. 1453–1456.

[17] U. Hagn, M. Nickl, S. Jörg, G. Passig, T. Bahls, A. Nothhelfer, F. Hacker, L. Le-Tien, A. Albu-Schäffer, R. Konietschke, M. Grebenstein, R. Warpup, R. Haslinger, M. Frommberger, and G. Hirzinger, "The DLR MIRO: a versatile lightweight robot for surgical applications," *Industrial Robot: An International Journal*, vol. 35, no. 4, pp. 324–336, 2008.

[18] C. Szydlowski, "Implementing PCI Express for industrial control," *RTC Magazine*, vol. 13, Sep 2004.

[19] Controller Area Network, http://www.can-cia.org/.

[20] B. Walls, M. McClelland, S. Persyn, and D. Werner, "Leveraging flight heritage to new CompactPCI space systems: a fusion of architectures," in *Digital Avionics Systems Conference (DASC)*, vol. 2, Oct 2001, pp. 8C4/1–8C4/7.

[21] M. Sarker, C. Kim, J.-S. Cho, and B.-J. You, "Development of a network-based real-time robot control system over IEEE 1394: using open source software platform," in *IEEE Int. Conf. on Mechatronics*, Jul 2006, pp. 563–568.

[22] A. Kapoor, A. Deguet, and P. Kazanzides, "Software components and frameworks for medical robot control," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2006, pp. 3813–3818.

[23] N. Simaan, R. Taylor, and P. Flint, "High dexterity snake-like robotic slaves for minimally invasive telesurgery of the upper airway," in *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 3217, Sep 2004, pp. 17–24.

[24] A. Kapoor, "Motion constrained control of robots for dexterous surgical tasks," Ph.D. dissertation, Johns Hopkins Univ., Sep 2007.

[25] J. Funda, R. Taylor, B. Eldridge, S. Gomory, and K. Gruben, "Constrained Cartesian motion control for teleoperated surgical robots," *IEEE Trans. Robot. Autom.*, vol. 12, no. 3, pp. 453–465, Jun 1996.

[26] B. Vagvolgyi, S. DiMaio, A. Deguet, P. Kazanzides, R. Kumar, C. Hasser, and R. Taylor, "The Surgical Assistant Workstation: a software framework for telesurgical robotics research," in *MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions*, Midas Journal, Sep 2008. [Online]. Available: http://hdl.handle.net/10380/1466