NCSLab: A Web-Based Global-Scale Control Laboratory With Rich Interactive Features

Yuliang Qiao, Guo-Ping Liu, *Senior Member, IEEE*, Geng Zheng, and Wenshan Hu

***Abstract*—This paper introduces the Networked Control System Laboratory (NCSLab) at http://www.ncslab.net, which provides a complete Web-based solution for users to carry out experiments on experiment devices located globally. A scalable architecture is proposed, which is composed of Web browsers, central Web server, MATLAB servers, regional experiment servers, control units, and experiment devices. Based on the architecture, many novel features, including visual algorithm designing, simulation, compilation, visual monitor configuration, and real-time monitoring and supervisory control, are designed and implemented by a combination of state-of-the-art technologies such as Web 2.0, Java 2 Enterprise Edition, and MATLAB. Users can enjoy all these rich interactive features with a simple Web browser from anywhere at any time. The issues of device safety, network security, and instability are also tackled in NCSLab.**

***Index Terms*—Networked control, remote laboratory, rich Internet applications, Web-based laboratory.**

I. Introduction

LABORATORIES play an important role in both education and academic research. Students can deepen their comprehension of theoretical knowledge and acquire practical knowledge via experiments in labs, and researchers need to validate their ideas via tests and trials in labs. However, a hand-on laboratory is not always available for the following reasons.

1) Some universities have difficulties in having adequate resources (funds, space, or manpower, for example) to construct high-quality laboratory infrastructure.
2) Experimenters may be located far away from the laboratory, which is often the case in remote education.
3) For those who are physically disadvantaged, it is almost impossible to reach a hand-on laboratory to carry experiments.

Manuscript received November 9, 2008; revised May 14, 2009; accepted June 8, 2009. Date of publication July 28, 2009; date of current version September 10, 2010. This work was supported in part by the National Science Foundation of China under Grants 60934006 and 60621001 and in part by the National High Technology Research and Development Program of China (863 Program) under Grant 2007AA04Z202.

Y. Qiao and G. Zheng are with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: aqiaojoe@gmail.com; zgrong@sina.com).

G.-P. Liu is with the Faculty of Advanced Technology, University of Glamorgan, CF37 1DL Pontypridd, U.K., and also with the Center for Control Theory and Guidance Technology, Harbin Institute of Technology, Harbin 150001, China (e-mail: gpliu@glam.ac.uk).

W. Hu is with the Faculty of Advanced Technology, University of Glamorgan, CF37 1DL Pontypridd, U.K. (e-mail: whu@glam.ac.uk).

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/TIE.2009.2027924

Fortunately, with the population of computers and penetration of network, remote laboratory has been emerging for the last few years. By connecting experiment facilities in conventional laboratories to computer network, remote laboratories enable users to operate real devices via the network and thus conduct experiments remotely. Users just need to access a computer connected to the network, which is often available to most people. Therefore, lab resources can be shared among different universities and institutes, whose sharing promotes better utilization of them. Remote learners or researchers can access remote laboratories from anywhere at any time and obtain practical experience from experiments in remote laboratories.

There exist some solutions of remote laboratories. Some of them are restricted to a particular experiment device (a dc motor [1], an inverted pendulum [2], a robot [3], [4], and a 2-DOF helicopter [5], for example); some solutions require users to download a special software, such as Java Runtime Environment [2] or even a client terminal [1], which is often not favored by cautious Web users. As to the functionalities provided by remote laboratories, some only enable users to run predefined programs and monitor the results [6], while others allow users to adjust some given parameters and watch how the real plant/process responses [7].

An interesting platform called the Automatic Control Telelab [8] allows users to design their own controller algorithms or reference inputs. However, the laboratory does not provide mathematical models for users to make simulations before experiments, which plays a prerequisite role in designing a control system. Furthermore, the laboratory only considers the design of a feedforward controller while ignoring the fact that users may need to add some filters in the feedback channel.

The iLab [9] that started at Massachusetts Institute of Technology in 1998 is an ambitious project which aims at supporting Internet-accessible laboratories and promotes their sharing among schools and universities on a worldwide scale. It proposes the three-tiered client/broker/server architecture and has been adopted by partners from several continents.

WebLab-Deusto system [10] has also gained a lot of attention, because it is involving rapidly and steadily and thus has accumulated rich experiences in the development and usage of a remote laboratory. The system has been used to control programmable logic devices. It is characteristic of its Mobility 2.0-enabled feature.

In this paper, Networked Control System Laboratory (NCSLab), which is a remote control laboratory, is presented, which aims at integrating various experiment devices located globally and offering various experiment services for users scattering around the world. Furthermore, NCSLab also intends

to free users from the constraints of time, space, software environments of their PC (operation systems and Web browser, to be specific), and programming knowledge mastered and give them a wide range of freedom from designing control algorithms to customizing their monitoring during experiments.

This paper is organized as follows: Section II investigates the issues and challenges of NCSLab; Section III discusses the architecture of NCSLab; Section IV explains the design of integrated Web-based solutions and their technical implementation, including algorithm design, simulation, compilation, experiment monitor configuration, and real-time monitoring and supervisory control; Section V illustrates an experiment session in NCSLab; and Section VI describes its use for teaching and research. The last section summarizes this paper and suggests the possible research directions of NCSLab.

II. Issues and Challenges of NCSLab

NCSLab is designed to be a Web-based global-scale remote laboratory in the area of automatic control systems. In order to achieve this goal, the following issues and challenges need to be investigated.

1) Inadequate network bandwidth and network instability. Although computer networks have evolved very rapidly and the network bandwidth becomes increasingly larger, the networks are far from adequate and stable because the demand of the network bandwidth has been rising at the same time. Experimentation in a remote laboratory is a network application that emphasizes on the real-time interaction between remote devices and users. Thus, the possible large network transfer delay or even disconnection incidents may do harm to user's interaction.
2) Concurrent user access to the same experiment devices. An experiment session is composed of several steps in order and a device can be occupied by only one user at a specific time, but in many cases, the experiment device may be accessed by multiple users simultaneously. In traditional hand-on laboratories, this conflict can be avoided because users are aware of the presence of each other and thus able to do manual scheduling continuously. However, for a remote laboratory, users are often ignorant of each other, and the manual scheduling policy is no longer applicable. Thus, there must be a scheduling mechanization in NCSLab to avoid the conflicts caused by competition for controlling the same experiment device by concurrent users.
3) Device safety. Conducting experiments is a process of tests and trials, and it happens very often that an experimenter may send false operation commands to a control system device. In addition, physical absence from the remote laboratory increases the risk of devices because the experimenter may not take action in time in the case of device malfunction or even breakdown. Therefore, there must be a safety mechanism to protect the device from damage due to false operations, and it is also favorable to introduce a self-recovery policy to the experiment system. This mechanism can be implemented using hardware or software or both in NCSLab.
4) Network security. Any network-based application cannot ignore the issue of network security. As to remote laboratories, the issue should be given more attention because network attacks from hackers cannot only make the software to break down but also pose threat to the physical experiment devices.
5) Good user experience. Web-based applications are favorable to desktop ones because they can keep users from the trouble of installing and updating software, but at a cost of interaction experience. However, performing control experiments is a very interactive activity; thus, NCSLab must use up-to-date Web technology to enhance the interactions during users' experiments.

NCSLab should provide flexibility for different levels of users and stimulate them to think and test their ideas. For example, undergraduates may just want to tune the PID parameters of a controller in experiments, while graduate students may design a predictive controller, and researcher may carry out experiments in a networked control scheme to test their novel algorithms.

In addition, NCSLab needs to accommodate users using different mother languages and in different zones; thus, its user interfaces should support internationalization, and it can be accessed in a 7×24 basis.

Users of NCSLab should also include lab administrators and device managers. It should be made as easy as possible for them to connect experiment devices in the existing hand-on laboratories to NCSLab and perform their duties to manage the lab resources.

III. Architecture of NCSLab

The architecture of remote laboratories has been evolving from its birth. However, despite numerous configurations in realization, architectures applied to remote laboratories can be categorized as the following groups.

1) Remote PC—proxy PC—experiment device. The most original form of remote labs has a simple proxy-PC architecture. A local computer is connected to an experiment device via cables (RS232, for example) directly and acts as a proxy to carry out experiments. Remote users need to install special software before making experiments. Via this software, they send operation commands to the proxy computer, and then, the proxy computer follows the commands and performs the corresponding operations on the experiment device. This architecture is characteristic of single user access and single device connected and is a kind of remote control in its essence. A typical example based on this architecture can be seen in literature [11].
2) Client—server—control units—experiment devices. Under this architecture, a server does not control an experiment device directly; instead, it is connected to a remote control unit via network first, and the control unit controls the device then. Thus, it is possible to connect multiple devices to the server because multiple control units can be connected to the network. Users still need to install special client software to connect to the server and then

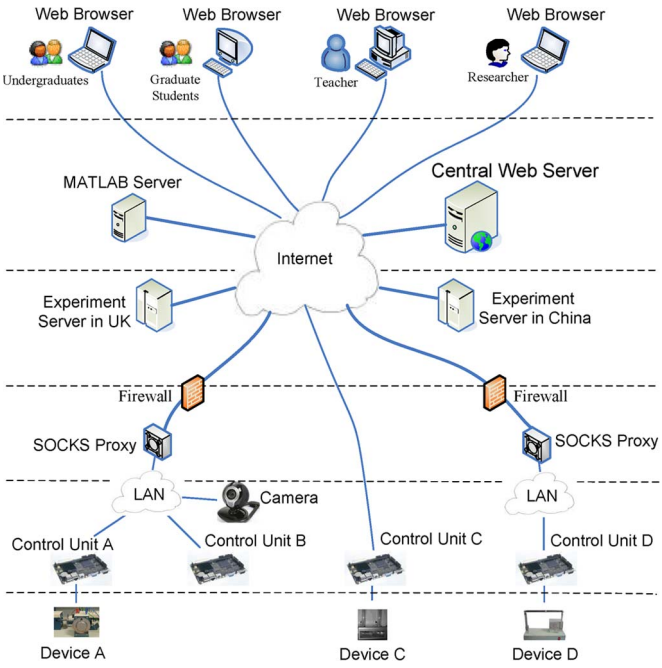


Fig. 1. Six-tier architecture of NCSLab.

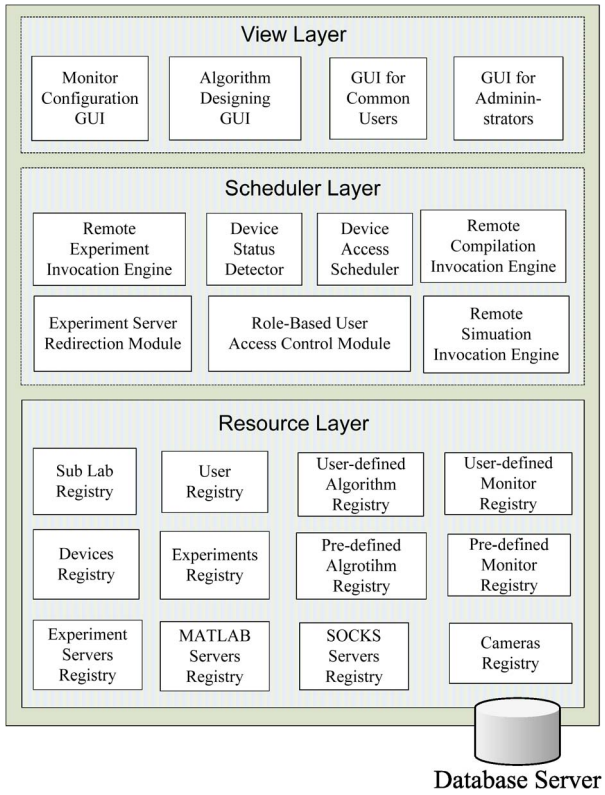


Fig. 2. Software structure of the central Web server.

control the remote device remotely. The architecture has been adopted widely for the last ten years. The NetLab project of Zhejiang University in China provides a good example [12].

3) Browser—Web server—experiment server—control units—test rigs. With the population of Web technology, the client software platforms of remote laboratories are increasingly moved to Web browsers, where browser plug-ins such as ActiveX and Java Applets are contained. At the same time, Web servers are added to the server side to act as a communication interface between the local experiment server and remote Web browsers. This idea is realized with various degrees of success, and many existing remote laboratories are built on this structure [13].

In view of the issues and challenges mentioned earlier, this paper proposes and designs a six-tier architecture for NCSLab. From a top–down view, the architecture of NCSLab consists of Web browsers, central Web server, MATLAB servers, regional experiment servers, SOCKS proxy servers, device control units, and experiment devices, as shown in Fig. 1.

A. Web Browsers

Considering that the client of a remote laboratory should avoid unnecessary restrictions on users and a proper selected client technology can increase availability, NCSLab adopts the Web browser as the only client software environment and employs Asynchronous JavaScript And eXtensible markup language (AJAX) and Flash technology to enhance the users' interaction with Web browsers and Web servers. By communicating with Web servers using hypertext transfer protocol, Web browsers can submit information to Web servers, as well as fetch Web pages or real-time data from them. The users of NCSLab may include undergraduates in a classroom, graduate students in their dormitories, teachers at home, and researchers on a journey, and the browsers used by them may be Internet Explorer, Mozilla Firefox, Safari, Netscape or Opera, etc.

B. Central Web Server

The central Web server resides on the top of the architecture except Web browsers, and its software structure has three layers: view, scheduler, and resource layers, as shown in Fig. 2. The view layer is responsible for interacting with Web browsers; it accepts requests from the users' browsers and generates dynamic Web pages to be displayed in the users' browsers. The resource layer manages the global resources of NCSLab, such as experiment server registry, MATLAB server registry, experiment device registry, user registry, etc., and the data of these resources are stored persistently in a relational-database server. The scheduler layer schedules the users' access to these resources and services provided by other servers. The scheduler runs some housekeeper programs, such as device status detector, so as to watch the status of these resources; it also receives the users' requests for accessing these resources and schedules their access based on their roles. For services provided by other servers, it launches the appropriate remote invocation engine to fetch these services for users; in the process of experiments, it redirects the users' Web browsers to the appropriate regional experiment servers to improve real-time performance of remote experiments.

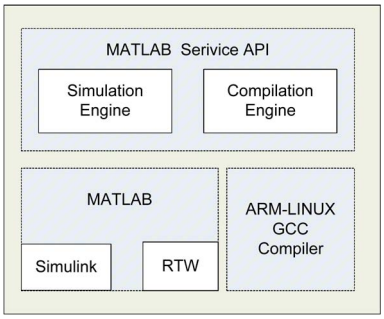


Fig. 3. Structure of a MATLAB server.

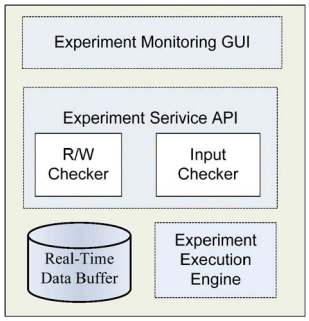


Fig. 4. Structure of an experiment server.

C. MATLAB Servers

A MATLAB server (Fig. 3) is an application server that provides service of remote simulation and remote compilation for the central Web server, and thus, users can perform Web-based simulation and compilation via the central Web server without installing MATLAB on their local computers. NCSLab adopts MATLAB as the modeling and simulation kernel because MATLAB is very popular in academia and industry; thus, users can learn designing an experiment in NCSLab rapidly with MATLAB's experience. MATLAB provides simulation functionality with its Simulink package and limited compilation ability with real-time workshop (RTW henceforth) toolbox in a local PC environment but does not support using these functionalities remotely. Therefore, GCC compiler and some other Java programs are utilized in MATLAB servers to enable remote simulation and compilation by cooperation with MATLAB. MATLAB servers are also designed to be scalable to allow concurrent tasks of simulation and compilation.

D. Regional Experiment Servers

Regional experiment servers are located in Internet environment, just like the central Web server. A regional experiment server (shown in Fig. 4) provides experiment services, such as starting and stopping experiments for the central Web server. Furthermore, the regional server that is being connected to the experiment device provides real-time monitoring service to Web users during the process of experiment. The real-time data are stored in a memory buffer instead of a database to avoid storing and fetching data from database that would introduce additional program delays during monitoring. The experiment execution engine can communicate with device control units, so as to invoke operations such as launching an experiment with a new algorithm, fetching in-process data, or sending tuning commands. The number of regional servers can be scaled according to the number and network distribution of experiment devices all over the world; now, there are two regional servers deployed in the current NCSLab: One in U.K. and the other in China.

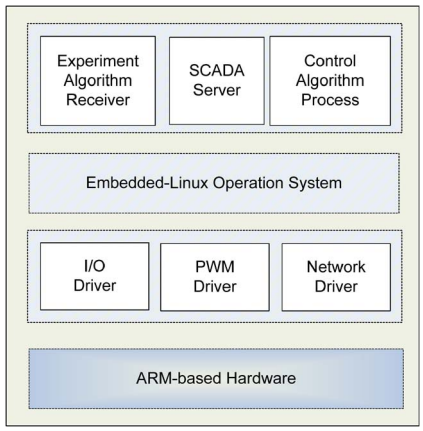


Fig. 5. Structure of a control unit.

E. SOCKS Proxy Servers

The tier of SOCKS proxy servers is introduced to solve the cross-firewall problem encountered in a campus network. Because an experiment device is often located in a campus local area network (LAN) and the firewall in the LAN often denies network connections from outside the LAN, the experiment server thus has to find some other routes to communicate with the experiment devices. Fortunately, SOCKS protocol provides the proxy mechanism that allows hosts outside a LAN to bypass the firewall and access the hosts inside the LAN while not degrading the security degree for LAN. Providing that a SOCKS proxy server is connected to both Internet and the LAN, the experiment server and the experiment device can communicate with each other via this proxy. Because a SOCKS proxy server just forwards network connections and does not need to understand the message included in them, it is highly efficient in network transition. A SOCKS proxy server is not a prerequisite if an experiment device is connected to the Internet directly, as Device C in Fig. 1 illustrates.

F. Device Control Units

Device control units (shown in Fig. 5) are physically connected to experiment devices via cables. The hardware environment of control units is mainly based on a high-performance 32-b ARM reduced instruction set computer microprocessor. It is composed of a main board, an AD/DA board, an LCD board, and an I/O board. The main board has a 32-b ARM CPU (200 MHz), 64-MB memory, a network port, and two Universal Serial Bus ports. The AD/DA provides 12 analog–digital channels and four digital–analog channels [15]. The solution is very cost effective for computer-controlled systems.

Based on the hardware, a concise Linux operation system runs in a control unit. In addition, device driver I/O modules,

pulsewidth modulation (PWM) driver modules, and network driver modules can be loaded whenever needed. The device I/O and PWM modules enable a control unit to collect sensor signals from its controlled device and drive the device, and the network driver module can be used to communicate with other hosts or control units in the network. These driver modules can be accessed by calling some C subroutines programmed by developers of device level.

During experiment, there are two additional programs running in the control unit besides the control algorithm program: One is experiment algorithm receiver, which is used for changing experiment algorithm; and the other is a simple Supervisory Control And Data Acquisition (SCADA) server, which enables remote monitoring and supervisory control.

Control units can be served as controller or sensor or both in a control experiment. For each experiment device, there should be a control unit assigned to it.

G. Experiment Devices

An experiment is physically connected to the AD/DA boards of the control unit assigned to it. For an experiment device to be incorporated into NCSLab, it has to be driven by electrical signals such as voltage or current, which is often possible for modern experiment devices. Furthermore, which input signals can be tunable and which output signals can be measured must be clearly defined. By doing so, an appropriate model for the experiment device can be built, which is essential for the design of the controller. There are many kinds of devices connected to NCSLab, such as servomotor, water tank, ball beam, magnetic levitation, and so on.

It can be seen earlier that the architecture of NCSLab is very different from the ones of other remote laboratories, and it has the following advantages.

1) Service oriented. The tiers of NCSLab are clearly defined, and the nodes in a higher tier are basically serviced by the nodes in direct lower tier via an application programming interface. These features make the tiers of NCSLab architecture loosely coupled with each other and interact in a service-oriented style, which establish a ground for its good scalability. To be specific, the central Web server communicates with experiment servers and MATLAB servers via remote procedure calls, and an experiment server communicates with device control units via a networked supervisory protocol that is built upon the Transmission Control Protocol/Internet Protocol (TCP/IP). Although Simple Object Access Protocol is claimed to be more suitable for service-oriented applications [14], it is not used in the NCSLab architecture because it lacks efficiency in transferring real-time data.
2) Good scalability. The architecture proposed has good scalability. Multiple experiment devices can be added to NCSLab without worrying about their geographical location. For the time being, NCSLab has incorporated 12 experiment devices located separately in the U.K. and China. The number of experiment servers and MATLAB servers can be increased to accommodate the growing load of more experiment devices and users' access. On the other hand, this architecture can be minimized so as to be applied in an isolate campus LAN.
3) Network security. Because control units and experiment devices are hiding behind the firewall of the LAN, they can be protected from the malicious attacks from Internet by virtue of interception of LAN firewall. Furthermore, additional security strategies such as IP filters can be implemented on the SOCKS proxy server to construct a more secure experiment environment. For the central Web server and regional experiment servers that are open to the Internet environment, the Internet security strategies can be employed very conveniently because these servers are deployed on workstation computers.
4) Ease of deployment and management. Because all the global information of NCSLab is stored in the database of the sole central Web server, it is possible to manage the configurations and resources of the NCSLab system by accessing the administrative Web pages of NCSLab system. In addition, the introduction of SOCKS proxy server also frees the one in charge in the deployment in the campus network from the troublesome work of asking for allocating an Internet-recognizable IP address and opening some special network ports to make their experiment devices accessible from the Internet.
5) Reduced data transfer delay. The architecture reported in [16] did not introduce regional servers' tiers, and it only deployed a central server located in U.K. When the Web users in China performed experiments on experiment devices located in China, the experiment data would be transferred to the central server in U.K. first and then resent to China, which caused many network hops and introduced large data transfer delay. The present architecture now allows Web browsers to automatically redirect to the regional servers located nearest to the experiment device during real-time experiment, and therefore, the user in China can interact with remote devices in China via the experiment server in China. Furthermore, the design of storing real-time data in a memory pool instead of a database saves the time consumed by database I/O communication. The subserver is also replaced by the highly efficient SOCKS proxy server, whose deployment may even be taken out if devices are connected to the Internet directly. As a result, the experiment server can communicate directly with control units and send real-time data directly to the users' browser, and the data transfer delay is reduced dramatically.

IV. Design and Implementation of NCSLab

Based on the aforementioned architecture, NCSLab offers powerful functionalities and rich features, which render users a whole Web-based control experiment solution from algorithm designing, simulation, compilation, monitoring configuration to real-time monitoring, and supervisory control. Users can complete the whole process rapidly with just a simple Web browser, and they can enjoy desktoplike interactive experience. This section will present the design and implementation of these functionalities.

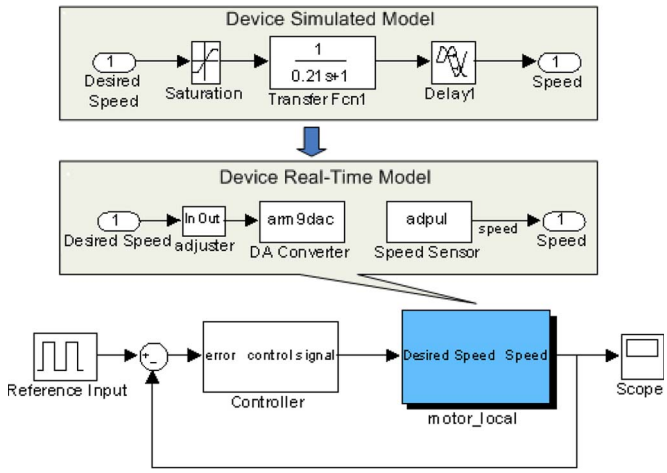


Fig. 6. Replacement from the simulated model to the real-time model.

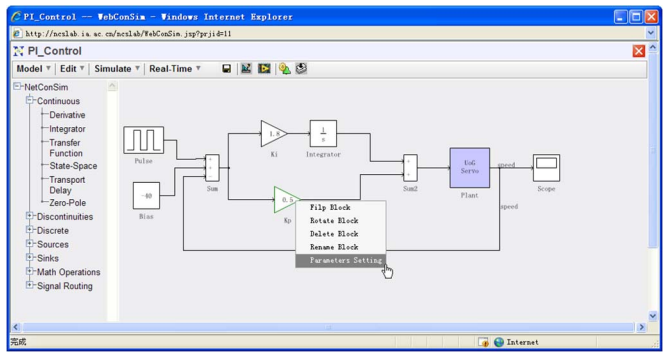


Fig. 7. Web-based GUI where users can design control algorithm and do simulation and compilation.

A. Web-Based Designing, Simulation, and Compilation

1) Modeling of Experiment Devices: NCSLab views every experiment device as a model that has known specifications of inputs and outputs. The behavior characteristic of experiment device can be simulated using a combination of Simulink blocks, and a model created in this way is viewed as a simulated model. In the process of simulation, it is the equivalent of the controlled experiment device. However, during practical experiment, the simulated model is meaningless and must be replaced by a real-time model that can influence the physical experiment devices. As mentioned in the last section, the physical inputs and outputs of experiment devices (I/O, PWM, etc.) can be accessed indirectly by calling preprogrammed C subroutines. Moreover, Simulink S-function blocks are utilized to encapsulate these subroutines; therefore, real-time models can be created by dragging these S-function blocks, together with other Simulink blocks.

For this reason, there are two Simulink model libraries designed in MATLAB server: One is the aggregation of simulated models for all experiment devices, and the other is the aggregation of real-time models for all experiment devices. For an experiment device, the model in simulated library has the same inputs and outputs as the one in the real-time library, although their inner implementations are very different. Thus, the simulated and real-time models expose the same interfaces to its outside. This ensures the seamless transition from the simulated model to the real-time model of experiment devices without modifying the other parts of the control system, such as reference input and controller during experiment. Fig. 6 shows the replacement from the simulated model to the real-time model for a dc motor in the Chinese Academy of Sciences.

2) Web-Based Designing of Control Scheme: NCSLab provides some predefined control algorithms for every experiment device, and users can carry out experiment using these algorithms. They can also design their own reference inputs and controllers in a Web-based graphic user interface (GUI). Therefore, NCSLab can accommodate requirements from different levels of users and allow them to test their own ideas.

The GUI for designing control scheme is a characteristic of "What You See Is What You Get," as shown in Fig. 7. To design a control scheme, users just need to draw a control diagram by clicking and dragging, and they can get nearly the same experience as using Simulink. They can drag a block from the left block-library tree; they can draw lines to connect the output of a block to the inputs of other blocks. If they double click a block or a line, a dialog window will pop up where they can configure its parameters. This visual designing method is very intuitive, and users do not need to have knowledge of any programming language. The GUI just uses the Web browsers' built-in functionality (such as Vector Markup Language for Microsoft Internet Explorer and Scalable Vector Graphics for Mozilla Firefox) to support drawing graphics; thus, users can enjoy the graphic designing experience without installing any browser plug-in.

The users' actions in GUI are responded by their local Web browsers, instead of the remote Web server. This ensures a timely interaction and improves the users' experience. The users' control diagram data are stored temporally in the memory of local browser and are modified according to the change of control diagram. The JavaScript Object Notation (JSON) format is used to organize the data for the consideration that it is light weighted and easy to parse and generate for a Web browser. The "MyModel.json" file in Fig. 8 shows the content of JSON data for the control diagram designed on the top of the figure. It is obvious that the user's control algorithm data should include global configuration data, list of all blocks, and list of all signal lines. The control diagram data can also be saved persistently on the server side if the user clicks the "Save" button, and it can also be loaded from the server side if the user clicks the "Load" menu item.

3) Web-Based Simulation: Because simulation can help users to learn how experiment devices work and validate the control scheme without worrying about doing damages to a real system, NCSLab advocates Web-based simulation before conducting a physical experiment remotely.

In fact, the real simulation environment resides in a MATLAB server, which may not be at the same host with the central Web server; thus, there must be a communication mechanism to support remote simulation from the central Web server. The developers of MATLAB servers write some extra JAVA programs to wrap the simulation functionality of MATLAB as a service and expose the interface of simulation service to the outside world via a JAVA remote method invocation (RMI)

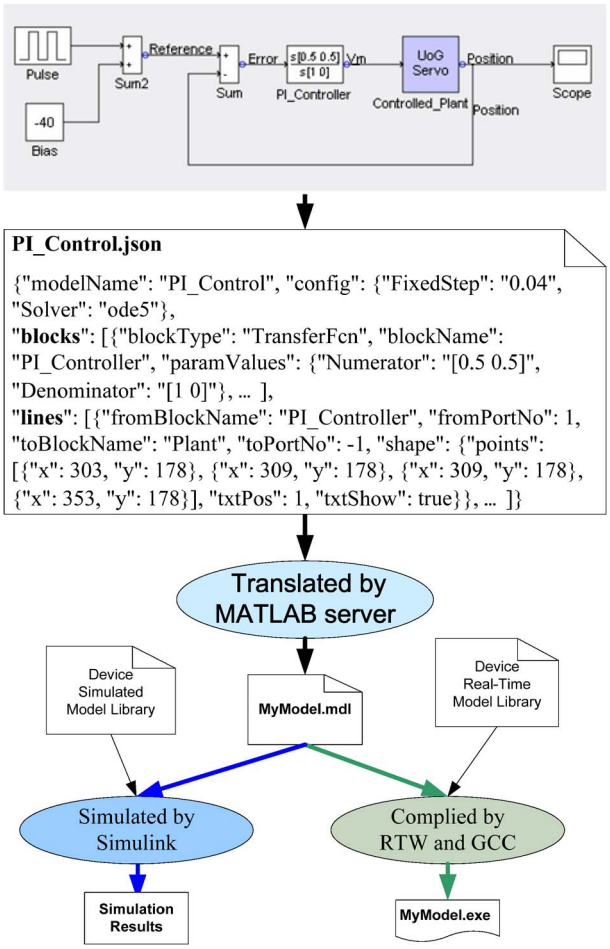


Fig. 8. Process of simulation and compilation for users' control scheme.

technology. The central Web server can find the URL of the exposed service by looking up the registry of MATLAB servers and remotely call this service via the help of Remote Simulation Invocation Engine, which implements an RMI client. In addition, because JSON data cannot be recognized by MATLAB to do simulation, a special program is written for the MATLAB server to parse the data of the JSON model file and translate them into a Simulink model file.

When a user finishes designing a control diagram and clicks the "Simulate" button, the JSON representation of the diagram is submitted to the central Web server. The central Web server launches the remote simulation invocation engine and includes the JSON data as the arguments of the remote invocation. MATLAB server then parses the JSON data and translates them into a Simulink model file. Of course, the model of the controlled device in the file is linked to be the model in device simulated library, as described earlier. The MATLAB server is then going to do simulation based on the model file using Simulink and generates the simulation result files, including the textual data and graphic curves. The results are sent to the central Web server as the return value of the remote invocation. The graphic curves and data are then shown in the user's Web browser and can be downloaded from the Web page for further analysis. Fig. 8 shows the transition process from a control diagram to its simulation results, and its flow starts on the top and ends with the left bottom of the figure.

4) Web-Based Compilation: A Simulink model file can only be recognized by MATLAB, and it cannot be run directly in the control unit. Thus, it is required to compile the Simulink model file into real-time executable codes before a real-time experiment. Fortunately, RTW provides automatic code generation from Simulink model to C codes. However, to compile C codes generated by RTW into real-time executable codes, the problem of cross-compilation arises, which means that codes compiled on one operation system platform cannot run on another platform. In the case of NCSLab, a MATLAB server is deployed on Windows-NT, while control units are deployed on ARM-Linux. The solution is based on CygWin, which builds a Linux-like environment in Windows and provides rich cross-compilation tools. In the MATLAB server, it is specified that the complier and linker tools be ARM Linux GCC that runs on CygWin. When RTW finishes generating C codes, the C codes can be immediately compiled by CygWin GCC into executable codes. As a result, the executable codes can be transplanted directly to run on the ARM-Linux operating system (OS) in a control unit.

The complicated compilation process is also wrapped as a service, and NCSLab can remotely call the service via Remote Compilation Invocation Engine. The service mode is very similar to the remote simulation and is consistent with the service-oriented architecture of NCSLab.

The flow from the top to the right bottom in Fig. 8 shows the compilation process on the server side. It should be pointed out that, in the compilation process, the MATLAB server is configured to select the model of the controlled device from the device real-time model library, and thus, the compiled executable codes have got the ability to interact with a physical device. At last, the generated executable codes are returned to the Web server as the result of remote complication invocation and stored in the algorithm registry of the central Web server for later utilization.

Although the backstage mechanization of Web-based compilation is very complicated, the processes are executed consecutively and intelligently on the server side. On the side of the browser, the user just needs to click a "Compile" button and wait for the results. Thus, users can concentrate themselves on the design of control algorithms without caring about how to enable the algorithms to run on a practical experiment.

B. Web-Based Experiment

The most important functionality of a remote control laboratory is that practical experiments can be carried out in it. Experiment servers, together with control units, provide the experiment service for NCSLab.

1) Scheduling of User Access to the Device: To avoid the possible collisions caused by concurrent access to the same experiment device (which has been described in the second section), a scheduling mechanism is designed and implemented. This mechanism borrows ideas from the Linux resource read/write privileges system and defines three privileges to access an experiment device: none privilege, watch privilege, and full control privilege. A user who wins the full control privilege can change control algorithms for the device, tune parameters, and watch signals/videos during experiments, while

users with watch privilege can only watch the signals/videos but cannot make changes to the device. Users who log in the NCSLab system get watch privilege automatically, while users who do not own none privilege.

Only a single user can win the full control right at a time, and the competition for full control is solved by the first come, first served (FCFS) priority queuing algorithm. If a log-in user wants to get full control right, he/she must wait in a queue. The users' position in the queue is decided by both his/her arrival time and the priority inherited from his/her role in the NCSLab system. Balancing fairness and preemption, the queuing weight which is used to decide the position in the queue can be calculated using $1/(T_{\text{arrival}} - \alpha P)$, where T_{arrival} is the time when the user joins the queue, P is the user's queue priority, and α represents the time units that the user's arrival time can be shifted earlier if the user's priority increases by one level. Thus, $T_{\text{arrival}} - \alpha P$ can be viewed as the acting arrival time, and its inverse is used here because an earlier arrival time should lead to a larger queuing weight. Thus, a user who arrives earlier or has a higher queue priority will queue at a position closer to the queue head and will be served at an earlier time. The user in the head of the queue is granted with the full control right and should not be preempted; as soon as the user leaves the position of the full control, the second user in front of the queue can take over the right of full control.

A user cannot get full control right of an experiment device forever, and there is a time limit for each experiment device. The time limit can be adjusted according to the difficulty of experiment and the actual situations. The user can cancel the full control right initiatively if the granted time has not been used up; moreover, the system can withdraw the full control right forcibly if the granted time expires.

In additional to the aforementioned automatic queuing mechanism, the device manager is also endowed with the privilege to interrupt the scheduling status and make some flexible decisions, such as stripping the control right of the current user and switching the full control right to any other user in the queue. This human interruption can increase the utilization of the experiment device, particularly when applied in the situation of online teaching.

2) Web-Based Algorithm Switching: The generated executable algorithms are stored in the database of the central Web server, and their entries are listed on the Web page of NCSLab. If a user gets full control right of the experiment device, he can click an algorithm entry in order to start an experiment with the selected algorithm. The central Web server then finds an appropriate experiment server for the device and sends the algorithm codes to the experiment server. From then on, the experiment-related works including downloading algorithm, tuning parameters, and fetching real-time data are left to the experiment server. The work is accomplished by the part of experiment execution engine in the experiment server. The experiment server downloads the algorithm to the control unit as soon as the algorithm is received from the central Web server.

In the control unit, there is a special service program called experiment algorithm receiver running on the Linux OS. This program listens at a specified network port and receives algorithm via the port. When a new algorithm is received without any mistakes, the service stops the current running algorithm and starts the new one. Then, the process of algorithm switching is finished. The algorithm switching is very rapid and convenient compared with the traditional industrial method by which a new control algorithm is burned into the control unit via computer serial ports and takes into effect after restarting the whole system.

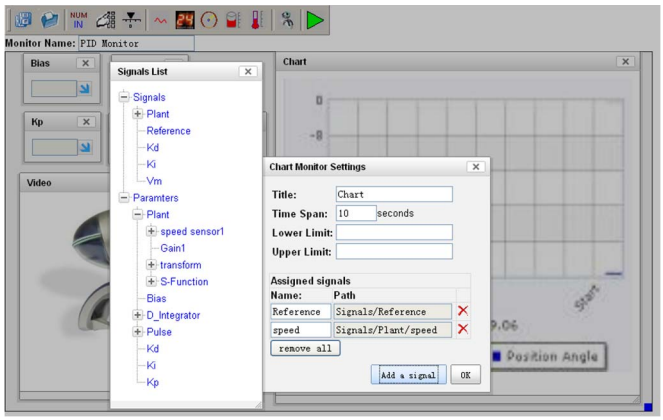


Fig. 9. Web interface of monitoring configuration.

In fact, the real-time algorithm runs as a separate computer process in the control unit. Aside from controlling the physical plant, the control unit automatically launches a SCADA server at the same time. This server can provide built-in monitoring service which enables remote monitoring and supervisory control.

3) Visual Configuration of Monitoring Interface: NCSLab also gives users the freedom to customize their own monitor interface, in which they can decide which signals to be observed and which parameters to be tuned during experiments. This is a creative solution which the other remote laboratories do not provide.

As mentioned earlier, the running real-time algorithm can provide monitoring service, to be exact, SCADA service. Remote network nodes can get this service as long as they follow the Netcon Supervisory Protocol [17]. This protocol is built upon the prevalent TCP/IP protocol and implemented in the NCSLab using C Socket programming.

The experiment execution engine of experiment server and control units exchange information via this protocol. As soon as an algorithm is downloaded to the control unit, the experiment execution engine connects to the SCADA service port of control unit, sends an EXT_GET_PARAMSIGNAL message, and obtains a list of all the tunable parameters and observable signals. The list of parameters and signals is then stored in the server for users' later selection.

To support more friendly use, NCSLab developers have designed the configuration page to be visualized, as shown in Fig. 9. By virtue of Yahoo! user interface JavaScript library (which is also an open-source JavaScript toolkit licensed under BSD) and fusion charts (which is a flash charting component that can be used to render data-driven and animated charts for Web applications), a set of virtual instrumentation is implemented in the configuration page. They include the components of numeric input, slider input, real-time chart, angular gauge,

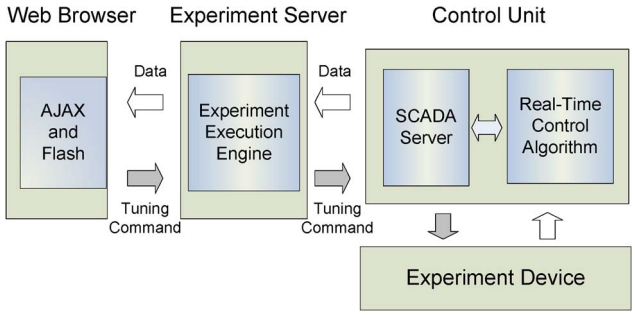


Fig. 10. Web-based monitoring and tuning.

cylinder gauge, thermometer, and video monitor. Users can set up a fresh and colorful configuration just by clicking and dragging these virtual instruments. They can double click the virtual instruments to select signals or parameters they are interested in for the instrument from the tree list which includes all the parameters and signals, and their configuration can be saved on the central Web server for a second use.

4) Web-Based Monitoring and Supervisory Control: When the user finishes the configuration of monitoring interface, he/she can move on to do real-time monitoring and supervisory control.

During experiment, the users' interactions in monitoring interface with NCSLab are handled by the regional experiment server which establishes a network connection with the control unit. The control unit is not allowed to interact with monitoring users directly because concurrent monitoring by a potential large number of users may consume too much or even exhaust the limited CPU resources of the control unit and thus degrade the real-time performance of the interaction between the control unit and the experiment device.

The user who gets the full control right can tune some parameters of the real-time algorithm, and his/her tuning commands are received by the regional experiment server. The experiment engine then encapsulates the parameter name together with its new value into an EXT_SETPARAM message and sends the message to the control unit, which adjusts immediately the specified parameter of the algorithm according to the body content of the message; the control unit also sends EXT_UPLOAD_DATA messages continuously, which contain the latest real-time data of all selected signals. The newest data are immediately received by the regional server and stored in a buffer of memory. Different from previous design [16], the tuning command and real-time data do not need to pass additional network nodes such as subserver and database server; thus, they can be transferred more quickly between control unit and experiment servers, which ensures real-time performance of monitoring and supervisory control while saving investments in subservers and database server. Fig. 10 shows how the information flows in the process of remote monitoring and tuning.

However, the continuous updating of real-time data is difficult to implement on Web-based monitoring interface. Although it is very easy for a desktop application to update its client, the technology cannot be employed on a Web-based application. With the traditional Web technology, the server pushes data to the Web browser just when a Web page is loaded. If a user wants to watch up-to-date data, he/she has to click the "refresh" button in the browser. Owing to the emergence of Web 2.0 [18] technology such as AJAX and Flashes, Web browsers can acquire the up-to-date data automatically by pulling them from the server side initiatively. In addition, JavaScript document object model operations are employed to enable the Web page to refresh a special area without changing the other areas; therefore, the monitoring interface looks very smooth and friendly without frequent screen blinking, which is often encountered in the interactive interfaces of other Web-based remote labs.

5) Networked Control Experiment: Another distinguished feature of NCSLab is that networked control experiments are supported on NCSLab. Networked control system refers to the control system which is closed via a network, and it has attracted a lot of attention from researchers for the recent years [19]–[26]. Although many laboratories can control the experiment devices via a network, the experiments cannot be called networked control ones because all the nodes that form a control loop, namely, controller, actuator, sensor, and controlled device, are still located in a local environment, and the control and sensor signals are delivered by a local cable, instead of a network.

The control units designed for NCSLab have built-in network ability, for both wired and wireless networks. In addition, network Send/Receive S-function blocks have been provided to facilitate the design of networked control algorithm; these S-functions are programmed with C Socket and implement User Datagram Protocol and TCP transport protocols. These conditions enable the design and implementation of experiments in a networked control scheme.

In a typical networked control experiment on NCSLab, two control units are required. One control unit serves as sensor and actuator node, and the other serves as controller node. Users may carry out networked control experiment with two choices. One is physical network and physical controlled plant, and the other is physical network and simulated controlled plant. Both configurations are useful for them to validate their ideas of networked control. The literature [27] demonstrates a networked control experiment which implements a networked predictive control (NPC) strategy on a servomotor system.

C. Device Safety, Network Security, and Stability

Because the experiment devices connected to NCSLab are running 24 h per day in an unattended condition, the issues of device safety, network security, and stability must be carefully handled.

1) Device Safety: NCSLab has adopted many measures to ensure the safety of devices at different tiers, and the safety measures adopted may vary from one device to another according to their characteristics. The ball-beam system in CASIA can be used as an example because of its inner instability, and its safety measures include the following.

1) A transparent mantle is attached to the beam to prevent the ball from falling down the beam rail because of possible heavy vibration.
2) A watchdog program is running in its control unit all times to ensure the correctness of DA outputs.

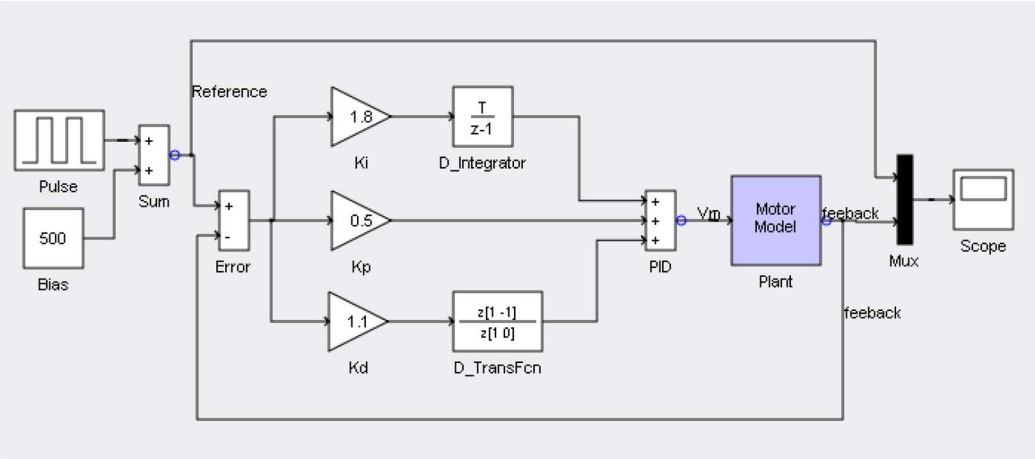


Fig. 11. PID control diagram of motor speed control system.

3) When a new algorithm is downloaded to the control unit, there is a validation process to make sure the algorithm works. If it fails to pass the validation, a reset protection algorithm is downloaded to the control unit. NCSLab also downloads the reset algorithm to the control unit when no one controls the experiment device.
4) The most dangerous situation occurs when users give an inappropriate reference signal or design unstable controller for the experiment devices in customized algorithms. Because the users' customized algorithms are compiled into executable codes by MATALB server and the controlled plant is represented by a real-time model before compilation, some protection blocks can be added into the real-time model. The most often used protection blocks are saturation block; to implement more complex protection blocks, S-function blocks can be used. These protection blocks can filter some dangerous signals and reduce the risk of damaging the device.

2) Network Security and Stability: The network security issue has been taken into consideration at the beginning when the architecture is designed. Because control units only need to interact with regional experiment, it can be configured to deny any connection except those from its regional server, and this method can reduce the risk of network intrusion from virus and hackers to a large extent. In networked control experiment, the data exchanged between local and remote control units can be encrypted to prevent interception of malignant users. Regional servers are deployed on a workstation computer; thus, the existing network security protection methods can be employed easily. A role-based access control model is adopted and implemented in the central Web server to satisfy the advanced security requirements for system administrators, device managers, and common users. Java 2 Enterprise Edition is utilized as the main application infrastructure of NCSLab for not only its cross-platform ability but also its advanced security mechanism.

To reduce the impact of network instability, a fault-tolerance device connection mechanism is designed. NCSLab does not bind an experiment device to a particular experiment server; instead, more than one experiment server candidates can be allocated to a device. A device status detector module is running as a demon program in the NCSLab Web server, and it polls the connect ability from these candidates to each experiment device at regular intervals. As long as there is one server that can connect to the experiment device, the device is available for experiment. Of course, there is a connection priority for each connection candidate based on its network hop distance, and the connection with fewer hops will be configured with a higher priority. NCSLab central Web server chooses from the experiment servers that can connect the device and selects the one with the highest priority to perform tasks of carrying out real-time experiment. This mechanism gains stability at a cost of redundant experiment servers and is very efficient for remote laboratories located at an unattended Internet-based environment such as NCSLab.

V. Experiment Example

This section will demonstrate an example to illustrate how to carry out experiments on NCSLab website. In the example, a user plans to control the speed of dc motor located in CASIA, and he/she wants to complete the whole process from simulation to real-time supervisory control during the experiment.

Following the instruction document of NCSLab, the user logs in the NCSLab and navigates to the experiment Web page which belongs to the dc motor device. The user notices that a model for the controlled device has been created by the device manager. The model uses a setting speed as input and the response speed as output; its transfer function representation is identified approximately as

$$G(s) = 1/(0.21s + 1)\exp(-0.35s).$$

Based on the model, the user designs a control diagram online. The diagram gives a square-wave signal as the reference input and uses a discrete PID controller, as shown in Fig. 11.

Then, the user clicks the "Simulate" button, and a panel pops up to show the simulation progress. After a few seconds, the simulation is completed at the remote server, and the graphic result is displayed on the pop-up panel in the local Web browser, as shown in Fig. 12. The user can return to the design panel and modify the PID parameters and discrete step size of the algorithm until he/she obtains a satisfactory simulation result.

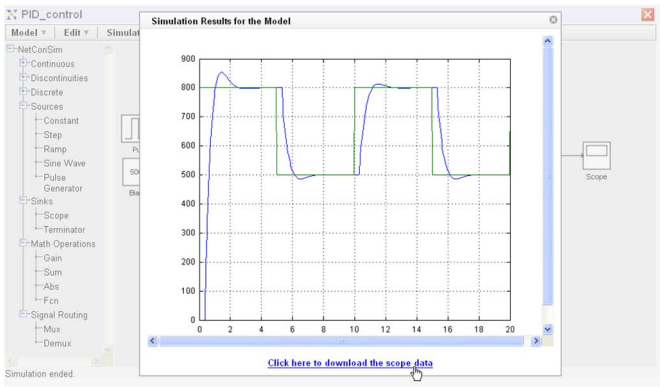


Fig. 12. Web interface which shows a simulation result.

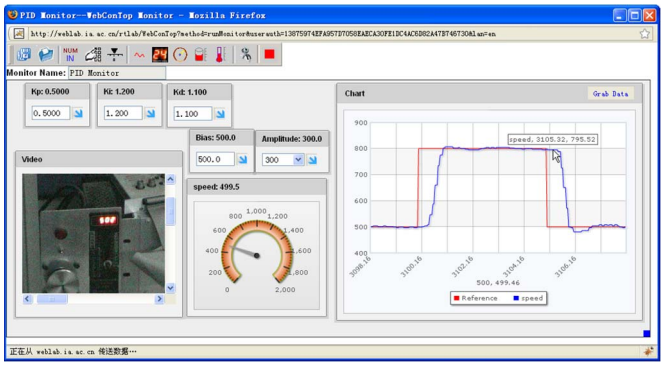


Fig. 13. Web interface for real-time monitoring.

After simulation, the user can click the "Compile" button, and the previous panel will pop up again to show the progress of remote compilation. The real-time codes of his algorithm are generated in several seconds, and the hyperlink to the algorithm is displayed immediately. The user then navigates to the page of PID algorithm after clicking the hyperlink.

The user then requests the full control of the dc motor device. It is assumed that there are no other users who compete for controlling the same device; thus, he/she obtains the control right immediately. After he/she clicks the "Start Experiment" button, the real-time algorithm codes are downloaded to the control unit automatically, which starts the practical control experiment.

Then, the user begins to configure a monitoring interface. He drags a video widget to watch the on-the-spot scene, a chart widget to display the curve of the speed response, as well as the desired speed, and some numeric input widgets for PID parameters.

At last, the user clicks the "Run Monitor" button, and then, the real-time monitoring GUI is displayed, as shown in Fig. 13. The widgets configured just now begin to work and automatically fetch the latest data/images and refresh themselves. Due to the uncertainty of the device model, the practical response is not totally the same with the simulation result; therefore, the PID parameters can be tuned until the motor speed control system reaches the control objective both in static error and dynamic responses. At last, the user can click the "Grab Data" button to save the experiment results on his local computer for further study.

VI. Use for Teaching and Research

NCSLab was initiated by the Advanced Control and Network Technology Research Unit in the University of Glamorgan, Pontypridd, U.K., in 2006, and it has won widespread cooperation from the Chinese Academy of Sciences, Tsinghua University, Central South University of China, and Nanjing Normal University of China since then. There are still many universities who want to update their hand-on laboratories and share their experimentation apparatus with NCSLab. Up to now, NCSLab has been running in 24 h a day and seven days a week for nearly three years, and it incorporated 12 experiment devices in both the U.K. and China.

Undergraduate and postgraduate students from the aforementioned institutes and universities have carried out experiments thousands of times, and experiments on NCSLab have also been included in the module of "Electrical Principles, Methods and Simulation," the final year projects of undergraduate students in control systems in the University of Glamorgan, and the course of "Process and Motion Control" for undergraduate students in the Central South University in China.

NCSLab is also open to universities and institutes around the world. Many international researchers and learners have contributed various excellent control algorithms to experiment devices in NCSLab. In addition, NCSLab is used by researchers to demonstrate their novel control algorithms, such as NPC at any place.

Various experiments have been performed for 1156 times since its last updated deployment in December 2008, which added the functionality of recording user access statistics. The users' evaluations are very positive according to the survey results of an online questionnaire on the NCSLab website. The majority of users agree that NCSLab is interesting and helpful to their experiment; however, they feel that the documents provided are not adequate and suggest that a typical example control diagram should be given for each experiment device for them to follow. Thus, it is a remaining task to make NCSLab easier to use.

VII. Conclusion

In this paper, NCSLab, which is a Web-based remote laboratory that provides rich user interactive features, has been presented. To integrate the globally located devices into NCSLab, a six-tier architecture has been proposed. Based on the scalable architecture, rich functionalities and features are designed to provide users with a complete Web-based solution from visual algorithm designing, simulation, compilation, monitoring configuration to real-time monitoring, and supervisory control. Users can thus carry out experiments on NCSLab without installing any software and mastering any complicated programming language. Thus, NCSLab can be used in the area of automatic control systems to test novel ideas in academic research or enhance practical knowledge in e-learning, and it can also be extended to other engineering fields.

Although NCSLab is sophisticated and easy to use, its effects on control education and research are to be further evaluated. In addition, it is also a challenge for NCSLab to keep track of the ever-evolving network and Web technology and benefit

from them in order to become more powerful, friendly, and secure. Moreover, future work on NCSLab can be devoted to the following directions:

1) to introduce experiment collaboration and competition of control performance between users into NCSLab;
2) to publish experiments on NCSLab as a Web service so that other kinds of remote laboratories can find and use them conveniently;
3) to win cooperation from more institutions and universities so that they can share their lab resources.

REFERENCES

[1] T. Kikuchi, T. Kenjo, and S. Fukuda, "Remote laboratory for a brushless DC motor," *IEEE Trans. Educ.*, vol. 44, no. 2, pp. 207–212, May 2001.

[2] J. Sanchez, S. Dormido, R. Pastor, and F. Morilla, "A Java/MATLAB based environment for remote control system laboratories: Illustrated with an inverted pendulum," *IEEE Trans. Educ.*, vol. 47, no. 3, pp. 321–329, Aug. 2004.

[3] M. Wu, J.-H. She, G.-X. Zeng, and Y. Ohyama, "Internet-based teaching and experiment system for control engineering course," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2386–2396, Jun. 2008.

[4] C. S. Tzafestas, N. Palaiologou, and M. Alifragis, "Virtual and remote robotic laboratory: Comparative experimental evaluation," *IEEE Trans. Educ.*, vol. 49, no. 3, pp. 360–369, Aug. 2006.

[5] C. C. Ko, B. M. Chen, J. P. Chen, J. Zhang, and K. C. Tan, "A Web-based laboratory on control of a two-degrees-of-freedom helicopter," *Int. J. Eng. Educ.*, vol. 21, no. 6, pp. 1017–1030, 2005.

[6] S. Qin, L. Bo, and X. Liu, "Development of the networked virtual instrument lab for vibration measuring based on Microsoft.NET," in *Proc. 21st IEEE Instrum. Meas. Technol. Conf.*, 2004, vol. 2, pp. 1286–1289.

[7] D. Hercog, B. Gergic, S. Uran, and K. Jezernik, "A DSP-based remote control laboratory," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3057–3068, Dec. 2007.

[8] M. Casini, D. Prattichizzo, and A. Vicino, "The automatic control Telelab: A Web-based technology for distance learning," *IEEE Control Syst. Mag.*, vol. 24, no. 3, pp. 36–44, Jun. 2004.

[9] V. J. Harward, J. A. Del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, I. Jabbour, P. D. Long, T. Mao, L. Naamani, J. Northridge, M. Schulz, D. Talavera, C. Varadharajan, S. Wang, K. Yehia, R. Zbib, and D. Zych, "The iLab shared architecture: A Web services infrastructure to build communities of Internet accessible laboratories," *Proc. IEEE*, vol. 96, no. 6, pp. 931–950, Jun. 2008.

[10] L. Gomes and J. García-Zubía, *Advances in Remote Laboratory and e-Learning Experiences*. Bilbao, Spain: Univ. Deusto, 2007.

[11] D. A. Miele, B. Potsaid, and J. T. Wen, "An Internet-based remote laboratory for control education," in *Proc. Amer. Control Conf.*, 2001, vol. 2, pp. 1151–1152.

[12] S. Ying and S. Zhu, "Remote laboratory based on client–server–controller architecture," in *Proc. Control, Autom., Robot. Vis. Conf.*, 2004, vol. 3, pp. 2194–2198.

[13] A. Agrawal and S. Srivastava, "WebLab: A generic architecture for remote laboratories," in *Proc. Int. Conf. Adv. Comput. Commun.*, 2007, pp. 301–306.

[14] Y. Yan, Y. Liang, X. Du, H. S. Hassane, and A. Ghorbani, "Putting labs online with Web services," *IEEE IT Prof.*, vol. 8, no. 2, pp. 27–34, Mar./Apr. 2006.

[15] H. Bian, G. P. Liu, and Z. Dong, "Structure design and application of embedded Ethernet based control systems," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2007, pp. 47–51.

[16] W. S. Hu, G. P. Liu, D. Rees, and Y. L. Qiao, "Design and implementation of Web-based control laboratory for experiment devices in geographically diverse locations," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2343–2354, Jun. 2008.

[17] Y. Zhu, G. Zheng, Z. Dong, and G. P. Liu, "Design of supervisory software for Ethernet-based control systems," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2007, pp. 36–41.

[18] T. O'Reilly, *What is Web 2.0*, Sep. 2005. [Online]. Available: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-Web-20.html

[19] J. W. Overstreet and A. Tzes, "An Internet-based real-time control engineering laboratory," *IEEE Control Syst. Mag.*, vol. 19, no. 5, pp. 19–34, Oct. 1999.

[20] G. P. Liu, J. X. Mu, D. Rees, and S. C. Chai, "Design and stability analysis of networked control systems with random communication time delay using the modified MPC," *Int. J. Control*, vol. 79, no. 4, pp. 288–297, Apr. 2006.

[21] X.-M. Tang and J.-S. Yu, "Feedback scheduling of model-based networked control systems with flexible workload," *Int. J. Autom. Comput.*, vol. 5, no. 4, pp. 389–394, Oct. 2008.

[22] C. Lazar and S. Carari, "A remote-control engineering laboratory," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2368–2375, Jun. 2008.

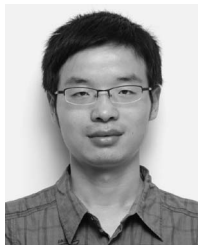
[23] G. P. Liu, Y. Xia, J. Chen, D. Rees, and W. S. Hu, "Networked predictive control systems with random network delays in both forward and feedback channels," *IEEE Trans. Ind. Electron.*, vol. 54, no. 3, pp. 1282–1297, Jun. 2007.

[24] J. Baillieul and P. J. Antsaklis, "Control and communication challenges in networked real-time systems," *Proc. IEEE*, vol. 95, no. 1, pp. 9–28, Jan. 2007.

[25] W. S. Hu, G. P. Liu, and D. Rees, "Event-driven networked predictive control," *IEEE Trans. Ind. Electron.*, vol. 54, no. 3, pp. 1603–1613, Jun. 2007.

[26] L. Yang, X.-P. Guan, C.-N. Long, and X.-Y. Luo, "Feedback stabilization over wireless network using adaptive coded modulation," *Int. J. Autom. Comput.*, vol. 5, no. 4, pp. 381–388, Oct. 2008.

[27] W. S. Hu, G. P. Liu, and D. Rees, "Networked predictive control over the Internet using round-trip delay measurement," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 10, pp. 2231–2241, Oct. 2008.



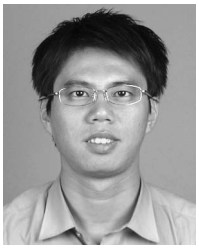
Yuliang Qiao received the B.Sc. degree from Tianjin University, Tianjin, China, in 2005. He is currently working toward the Ph.D. degree in the Complex Systems and Intelligence Science Laboratory, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His current research interests include networked control systems and Web-based remote laboratories.

Guo-Ping Liu (M'97–SM'99) received the B.Eng. and M.Eng. degrees in electrical and electronic engineering from the Central South University of Technology (currently the Central South University), Changsha, China, in 1982 and 1985, respectively, and the Ph.D. degree in control engineering from the University of Manchester Institute of Science and Technology (currently the University of Manchester), Manchester, U.K., in 1992.

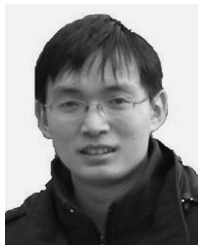
He did a postdoctoral research at the University of York, York, U.K., in 1992–1993. He was a Research Fellow with The University of Sheffield, Sheffield, U.K., in 1994. During 1996–2000, he was a Senior Engineer with GEC-Alsthom and ALSTOM and then a Principal Engineer and a Project Leader with ABB ALSTOM Power. He was a Senior Lecturer with The University of Nottingham, Nottingham, U.K., in 2000–2003. He has been a Professor with the University of Glamorgan, Pontypridd, U.K., since 2004, where he is currently the Chair of control engineering; a Hundred-Talent Program Visiting Professor with the Chinese Academy of Sciences, Beijing, China, since 2001; and a Changjiang Scholar Visiting Professor with the Center for Control Theory and Guidance Technology, Harbin Institute of Technology, Harbin, China, since 2008. He is the Editor-in-Chief of the *International Journal of Automation and Computing*. He has more than 400 publications on control systems and is the author/coauthor of six books. His main research areas include networked control systems, modeling and control of fuel cell vehicles, advanced control of industrial systems, nonlinear system identification and control, and multiobjective optimization and control.

Dr. Liu was the General Chair of the 2007 IEEE International Conference on Networking, Sensing and Control. He was awarded the Alexander von Humboldt Research Fellowship in 1992. He received the best paper prize for applications at the UKACC International Conference on Control in 1998, the GuanZhaoZhi best paper prize at the 26th Chinese Control Conference in 2007, and the best paper prize in automation at the 13th International Conference on Automation and Computing in 2007.



Geng Zheng received the B.S. degree from Zhejiang University, Hangzhou, China, in 1997, the M.Sc. degree from the Beijing Institute of Machinery, Beijing, China, in 2002, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2006.

He is currently an Assistant Researcher with the Institute of Automation, Chinese Academy of Sciences. His current research interests include networked control systems and embedded systems.



Wenshan Hu received the B.Sc. and M.Sc. degrees from Wuhan University, Wuhan, China, in 2002 and 2004, respectively, and the Ph.D. degree from the University of Glamorgan, Pontypridd, U.K., in 2008.

Since 2008, he has been a Visiting Research Fellow with the Faculty of Advanced Technology, University of Glamorgan. His current research interests are power electronics and superhigh speed electric motor control.