Framework for Rapid Integration of Offline Experiments into Remote Laboratory

https://doi.org/10.3991/ijoe.v13i12.7738

Ning Wang
University of Houston, Texas, USA

Gangbing Song
University of Houston, Texas, USA

Xuemin Chen(✉)
Texas Southern University, Texas, USA
chenxm@tsu.edu

Abstract—Remote laboratory (RL) has become an important component of online education and smart factory due to its capability of remote operation and cost-saving. Recent advantages of information technology have provided various ways to develop RLs. However, how to rapidly bring an offline experiment online is still a vital issue in the development of RL. In this paper, a new flexible framework based on social instant messaging (IM) application architecture is proposed for integrating an offline remote experiment into RL rapidly as a communication node with unique ID. To rapidly connect the LabVIEW controlled devices to server, a new LtoN_UID (LabVIEW to Node.js with Unique ID) module built on Socket.IO protocol is designed and implemented. To demonstrate the effectiveness of this new flexible framework, two existing engineering experiments, i.e., a Smart Vibration Platform (SVP) experiment and a PID motor speed control experiment, are integrated into remote laboratory.

Keywords—Remote Laboratory, Instant Messaging Application Architecture, Flexible Framework, LabVIEW, Node.js

1 Introduction

To significantly reduce the cost of maintaining a wide variety of equipment, more and more remote accessible networked equipment have been developed for educational and industrial applications [1]. For the Remote Laboratory (RL) development, two issues need to be addressed for integrating an offline experiment into a RL system [2], [3], i.e., 1) the employment of special software plugins are seriously affecting the cross-platform capability of the RL system; 2) the special system architecture need to be designed and implemented to connect the client applications and the experimental equipment. Current research activities to tackle these issues are summarized as follows.

A real-time remote access laboratory with distributed and modular was designed at University of Southern Queensland [4]. They proposed a general architecture for distributed Peer-to-Peer (P2P) network control systems, but only focused on Micro-Controller Unit (MCU) and communication protocol selection. However, how to develop a flexible software system to quickly develop the RL system is still not addressed clearly.

A flexible RL architecture was designed and implemented at University of Limoges and University of Mostaganem [5]. They proposed a flexible architecture based on Node.js web engine, and used a FEHI (Flexible Ethernet Hardware Interface) and PEB (Practical Evaluation Board) to directly handle the real-time communication between experiment equipment and Node.js server without using any software tools, such as NI LabVIEW, MATLAB. However, for an existing experiment, the new hardware interface and PEB need to be reconfigured and re-implemented.

A unified framework has been developed at Texas Southern University and University of Houston based on the traditional P2P RL system architecture [6], [7]. To support real-time communication, an LtoN (LabVIEW to Node.js) module was designed and implemented for traditional P2P connection between user and experimental device [6]. However, this LtoN module can only connect an experiment with server using tranditional P2P approach to achieve real-time communication. Consequently, how to rapidly integrate experiment into the RL system is still not well addressed by this framework.

As the experimental hardware and the network model limitations which are discussed above, it is hard to be extended to other institutes with their proposed solutions. Therefore, how to design a flexible approach to rapidly integrate remote experiment into RL system is still a vital research topic for RL development [5], [6], [7]. To address this essential issue, a new flexible framework based on social Instant Messaging (IM) application architecture as a TURN-KEY remote experimental integration solution is proposed and implemented in this paper. With this new framework, a new offline experiment can be added into the RL system as a node with its unique ID. With the Web 2.0 technology, this flexible framework is easy to be integrated into most of popular web-based Learning Management Systems (LMSs), such as, Moodle, iLab Shared Architecture (LSA) to connect them into social network. Moreover, a new LtoN_UID (LabVIEW to Node.js with Unique ID) module built on Socket.IO protocol is designed and implemented for rapid connecting the experimental equipment with server. To the best knowledge of the authors, this solution is the first one using the popular social IM application architecture to implement the RL system without extra plug-ins.

To demonstrate the effectiveness of this new flexible framework, a remote Smart Vibration Platform (SVP) experiment and a remote PID motor speed control experiment are revamped. Meanwhile, this new framework also can be used for the integration of industrial equipment remote control and monitoring applications. It will be an essential improvement for the RL system development in future.

The rest of the paper is organized as follows. Empirical study of related technologies are presented in Section 2. In Section 3, the architecture of the new flexible

framework is presented. In Section 4, the rapid integration of two remote experiments is discussed. Concluding remarks are drawn in Section 5.

2 Empirical Study of Related Technologies for Flexible Framework Development

The distributed RL system aims to expand a One-to-Multi (O2M) communication paradigm, where one central system serves multiple users; or to a Multi-to-Multi (M2M) communication paradigm, with many users using many experiments by different providers. In a distributed system, experimental modules need to be created and hosted by individuals. Users are all scattered in the network, and anyone can connect to any experiment if it is available. To achieve this goal, a distributed RL system with excellent features should have a flexible system architecture, a stable real-time middleware in server, and a high performance real-time data transmission protocol [4]. To achieve these excellent features of the distributed system, online social IM application architecture, Node.js and Socket.IO are the suitable candidates for the new flexible framework implementation based on our empirical research.

2.1 Document title and meta-data

Social IM application can refer to any kind of communication over the Internet that offers a real-time communication from any senders to any receivers. It addresses One to One (O2O) communications as well multicast communications (M2M) from many senders to many receivers, or may be a feature of a web conferencing service [8]. For a web-based online classroom, which is a kind of social IM application, could provide effective interactive tools and contextual learning scene, and it can deliver two part services, instructional communicating service and collaborative learning environment service [9]. The most notable strength of online IM application is to flexibly and rapidly connect the different users together [9]. As shown in Fig. 1, a social IM application has two parts, chat server module and client web module. In the server module, normally, there is a user management module to create the unique user ID for every user. Meanwhile, the server module creates a thread based on the unique user ID to support the users' real-time communication. The user management module supports three chat patterns: O2O, O2M and M2M. The different chat patterns can flexibly combination based on the users' requirement. The new flexible framework inherits this advantage from the online IM application to provide three different communication patterns for RL integration. Thus, the new framework can offer a more flexible way to build up different pattern remote experiments for engineering education, research activities and industrial applications.

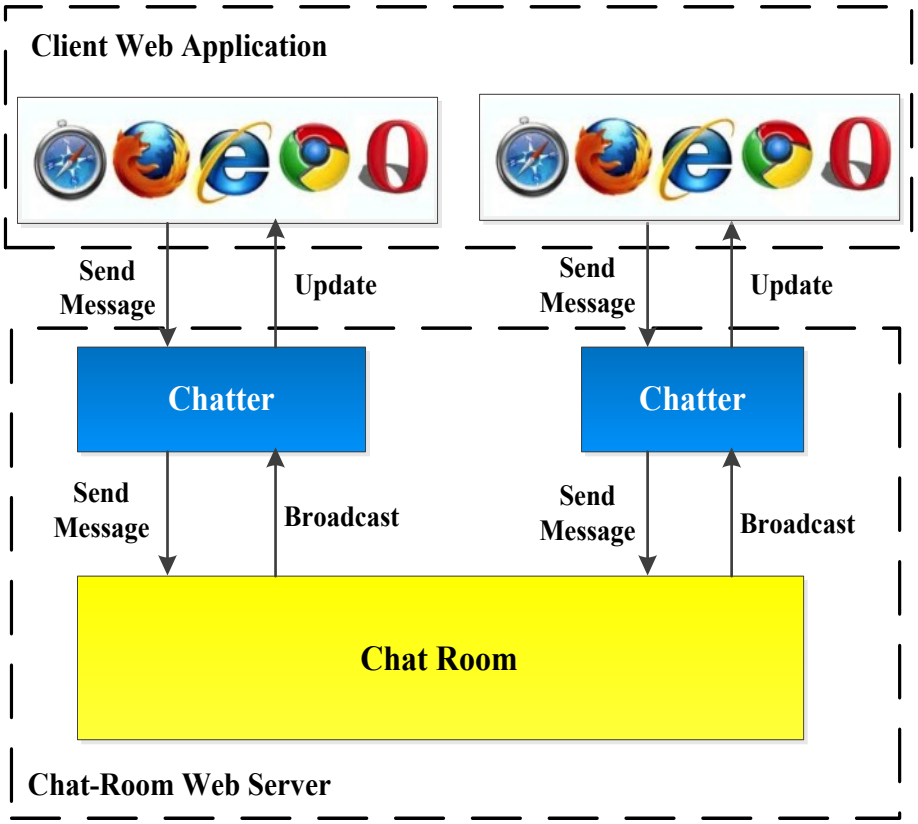

Fig. 1. The web-based IM system architecture.

2.2 Node.js Web Engine

To implement an efficient middleware for supporting high performance real-time data transmission between experimental devices and users, a stable web engine must be chosen. With the Internet of Things (IoT) technology explosive development, the Node.js is known for its speed, scalability and efficiency making it suitable candidate for data-intensive and real-time devices and applications development [10]. With these advantages, it has become a prime candidate for RL system implementation. Node.js is a web engine worked in server side, and is designed to notably setup web server for writing scalable Internet real-time communication applications [11]. Futhurmore, an event driven operation mode and asynchronous I/O port are used in Node.js to minimize overhead and maximize scalability. It is unlike the tranditional JavaScript programs, Node.js is exe-cuted is executed as a server side JavaScript application. In addition, Node.js can implement multiple common JavaScript specifications in server side. Meanwhile, it can provide a Read Eval Print Loop (REPL) environment for interactive Javascript app testing. Comparing with Apache web en-

gine, the Node.js is an especially fast and efficient web engine which is more suitable to handle the high perfor-mance real-time communication.

Because Node.js stays away from certain undesirable interfaces, such as synchronous I/O, it only exposes non-blocking asynchronous interfaces to the programmer. As each web application running on the Node.js is a single thread, the users don't need to consider an event completing and taking over while they are in the middle of another task. Node.js uses the module architecture to simplify the creation of complex web applications [12]. Each module contains a set of functions related to the 'subject' of the module. For example, the Node-HTTP-proxy module contains functions specific to HTTP (Hypertext Transfer Protocol) Proxy. Furthermore, it provides some core modules to support user to access files on the file system, to create HTTP Proxy and Socket.IO, and to perform other useful functions. Node.js is also a promising technology and an excellent choice for high load web applications.

2.3 Data Transmission Protocol Selection

Currently, most of the RL systems have used the Web Service technology to handle the real-time data transmission. However, the drawback of the Web Service for real-time data communication is the low transmission efficiency [2], and normally some extra plugins, such as Java Applet, Adobe Flash component, are required to fill this gap. With the advent of HTML5, WebSocket protocol provides a new approach to address the low transmission efficiency and extra plugin issues of Web Service technology [13]. The WebSocket protocol makes more interaction between a web browser and a server-side middleware possible, and facilitates the real-time duplex data transmission between client-side web apps and the server-side middleware. In addition, the real-time data communications can traverse the TCP (Transmission Control Protocol) port 80 with the WebSocket protocol, which is of benefit for those environments blocking non-web Internet connections by using a firewall [14]. Generally, the WebSocket mainly used to solve several key issues which cause the low transmission efficiency with REST (REpresentational state transfer) and HTTP. These issues include 1) Bi-directional: Normally, HTTP is an un-directional protocol, so the client always initiates a request, and returns a response after server processes. At last, the client consumes the response from server. However, as a bi-directional protocol, WebSocket has not the pre-defined message patterns such as request/response [15]. With the WebSocket, Either client or server can send a message to the other part. 2) Full-duplex: HTTP allows a request message from client to server and then server sends a response message to the client. However, WebSocket allows client and server to talk independent of each other [15]. 3) Single TCP Connection: Typically, a new TCP connection is initiated for a HTTP request, and is terminated after the response is received. The other new TCP connection should be established for another HTTP request/response. However, for WebSocket, the HTTP connection is upgraded with the standard HTTP Upgrade mechanism [15]. The client communicates with the server via the same TCP connection for the lifecycle of WebSocket connection. 4) Lean Protocol: As the HTTP is a chatty protocol, the Advanced REST Client extension can send a set of HTTP headers in request message. However, the purposes of WebSock-

et are to break the limitations of the request/response protocol such as HTTP [15]. In summary, WebSocket provides an alternative approach to the REST/SOAP (Simple Object Access Protocol)/AJAX (Asynchronous JavaScript and XML) technologies for developing the real-time communication web application, such as, some web-based control applications [14]. Moreover, WebSocket is the next generation method of asynchronous communication between client and server, and is already standardized by the World Wide Web Consortium (W3C). Currently, it is already implemented in the most of the popular web browsers, such as Microsoft Edge, Chrome, Safari, Firefox, Opera, and among others [15].

Socket.IO is designed based on WebSocket, and enhances the WebSocket by providing built-in multiplexing, horizontal scalability, automatic JSON (JavaScript Object Notation) encoding/decoding, and more [16]. In general, Socket.IO uses feature detection to decide which approach, such as WebSocket, AJAX long polling or Flash, will be used to establish the connection for real-time web applications. Through blurring the differences between the different transmission mechanisms, it is possible for Socket.IO to support the real-time web applications in any popular browsers. Socket.IO includes two parts: a client-side library for the browsers, and a server-side library supported by Node.js. Although Socket.IO works as simply a wrapper for the WebSocket, it provides more features, which includes broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O, to support real-time web applications development.

To compare the different real-time data transmission protocol performance, we tested 10, 100, 250 and 500 data exchanges per millisecond between the client web module and the server-side middleware. Each data exchange between the Node.js server and the Chrome browser is a 4K bytes random data string. The Node.js server is running in release mode. Meanwhile, the console messages are minimized output for both server and client. The server is HP Proliant DL380e Gen8. Hardware of the server includes Intel Xeon E5 2.5 GHz processer, 16 GB of RAM. The network is the University of Houston's main Campus Wi-Fi network, and the download speed of this network is around 45 Mbps and the upload speed around 75 Mbps. Based on the test results listed in Table I, the Socket.IO and WebSocket have better performane than AJAX/REST in general. Comparing the SocketIO with WebSocket, SocketIO has better performances than WebSocket has. Consequently, Socket.IO is selected for implementing real-time data transmission between the users and the experimental devices.

Table 1. Comparison between different data transmission protocol

Data Exchanges (Number of data exchanges)	AJAX/REST (kb/ms)	Socket.IO (kb/ms)	WebSocket (kb/ms)
10	40	32	30
100	320	340	330
250	800	830	820
500	1500	1600	1600

3 Methodology

To answer our research question, "*How to design a framework to rapidly and flexibly integrate the remote laboratory system?*", a novel flexible framework is proposed. It is designed based on the online IM application architecture, and a new version LtoN module is developed to rapidly and flexibly integrate the remote laboratory. Fig. 5 shows the architecture of the novel flexible framework. The novel flexible framework includes three parts: client web module, server module, and LtoN module for experiment control.

3.1 Client Web Module

In the client web module design and implementation, the MVC (Model-View-Controller) software architectural pattern is used for implementing the User Interfaces (UI) as shown in Fig. 2. As client web technology has matured, frameworks, such as AJAX, JavaScript MVC, AngularJS, have been created that allow the MVC components to execute partly on the web browsers. To run the client web module on any popular browsers, the HTML5 technology is used for the web application implementation. Some popular development languages, HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JQuery/JQuery-Mobile JavaScript libraries, for web application development are involved as well. In addition, to resolve the need of installing extra plugins for support efficient real-time data transmission, the Socket.IO module is used to client web module implementation. Meanwhile, the server-based Mashup technology is used for User Interface (UI) integration as well.

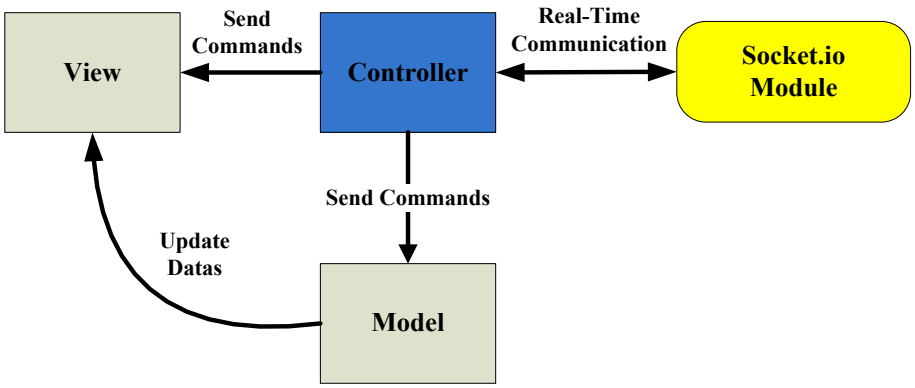

Fig. 2. The client web module designed model with MVC.

3.2 Server Module

To rapidly and flexibly connect the users with experiment equipment together, a communication management module is designed based on the online IM application architecture. Every user, who is real person or experimental equipment, is arranged a

unique ID, and system creates a communication thread and pair them together based on this unique ID. With this module, the remote laboratory system based on the novel flexible framework can flexibly create one user to one equipment experiment, multi users to one equipment cooperative experiment, one user to multi equipment experiment and multi users to multi equipment cooperative experiment. It will significantly benefit the remote laboratory implementation. Fig.3 shows the architecture of server module.

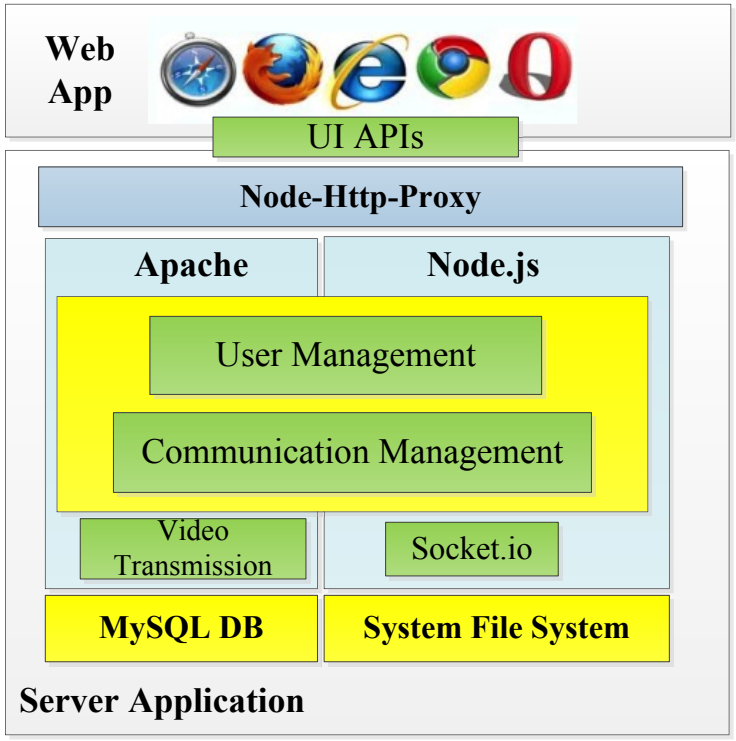

Fig. 3. The architecture of server module.

To resolve the Web Service technology performance issue, a combined solution of both Apache web engine and Node.js web engine is implemented for the real-time communication between experiment equipment and end users. Node.js enables web developers to create a real-time communication web application in JavaScript which are both server-side and client-side. In the Node.js server-side software system, Socket.IO, a JavaScript library, is used to support real-time communication between server and client. Moreover, a real-time video transmission solution based on the HTTP Living Streaming (HLS) protocol also is implemented in server side. Thus, the server module is directly built on the top of an Apache web server engine, a Node.js web server engine, and a MySQL database. In addition, the Operation System (OS) of the server uses Centos 7.0 OS to better support the server module.

3.3 The New LtoN_UID Module

To rapidly connect the experimental equipment into the remote laboratory platform, an essential issue, "How to design a real-time communication module which can be easily used to connect experimental equipment and end users?", must be addressed. To address this issue, an easy to use real-time communication module based on Socket.IO, namely LtoN_UID (LabVIEW to Node.js with Unique ID) is proposed and implemented using LabVIEW [17], [18]. To fulfill the requirement of the new online IM application architecture based framework, the unique ID management function has been integrated into the LtoN_UID module. The new LtoN_UID module can support P2P and M2M real-time communication via the unique ID arranged by server. Moreover, some bugs also have been fixed in this new LtoN_UID module to improve its performance.

With the development of computer technology, LabVIEW also integrated a new feature to interact with the experiment Virtual Instruments by using RESTful web services technology. REST (Representational State Transfer) provides a lightweight protocol accessible to a wide variety of clients. The architecture does not require complex message passing and provides a simple interface for user to begin using Web services in LabVIEW. However, it requires the client interface to be developed using different technologies, and LabVIEW plug-ins must be installed in the browsers [43]. To resolve this plug-in issue and support efficient data transmission, the new real-time communication module, LabVIEW to Node.js with Unique ID (LtoN_UID) is designed and implemented.

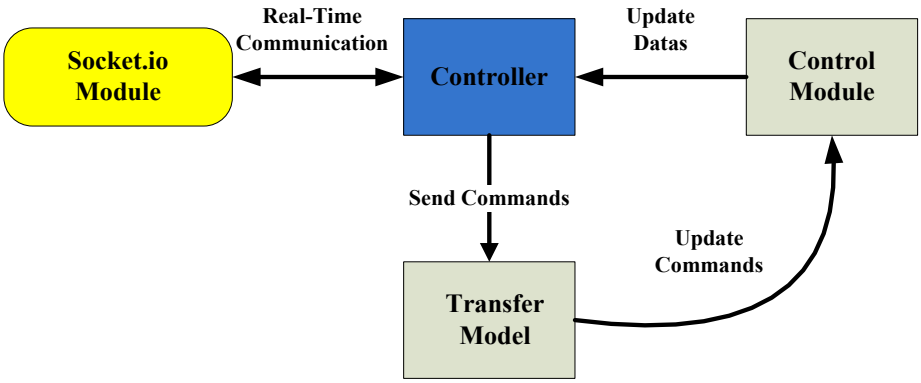

Fig. 4. the LtoN_UID module designed model with MVC.

To design the LtoN_UID module, the MVC design approach is used as well. As shown in Fig. 4, the LtoN_UID includes a real-time data transmission controller and an experiment data transfer model. With the new LtoN_UID, the experiment control module, which implements the experiment control logic, is connected with Socket.IO module together. In order to achieve the secure data transmission, a new message packet approach is designed and implemented. The example LabVIEW code of LtoN_UID module is shown in Fig. 5. With this new version LtoN_UID module, the

client web module can real-time communicate with experimental equipment without LabVIEW plug-ins. Meanwhile, any experimental equipment controlled by LabVIEW can be rapidly integrated into the remote laboratory system anywhere.

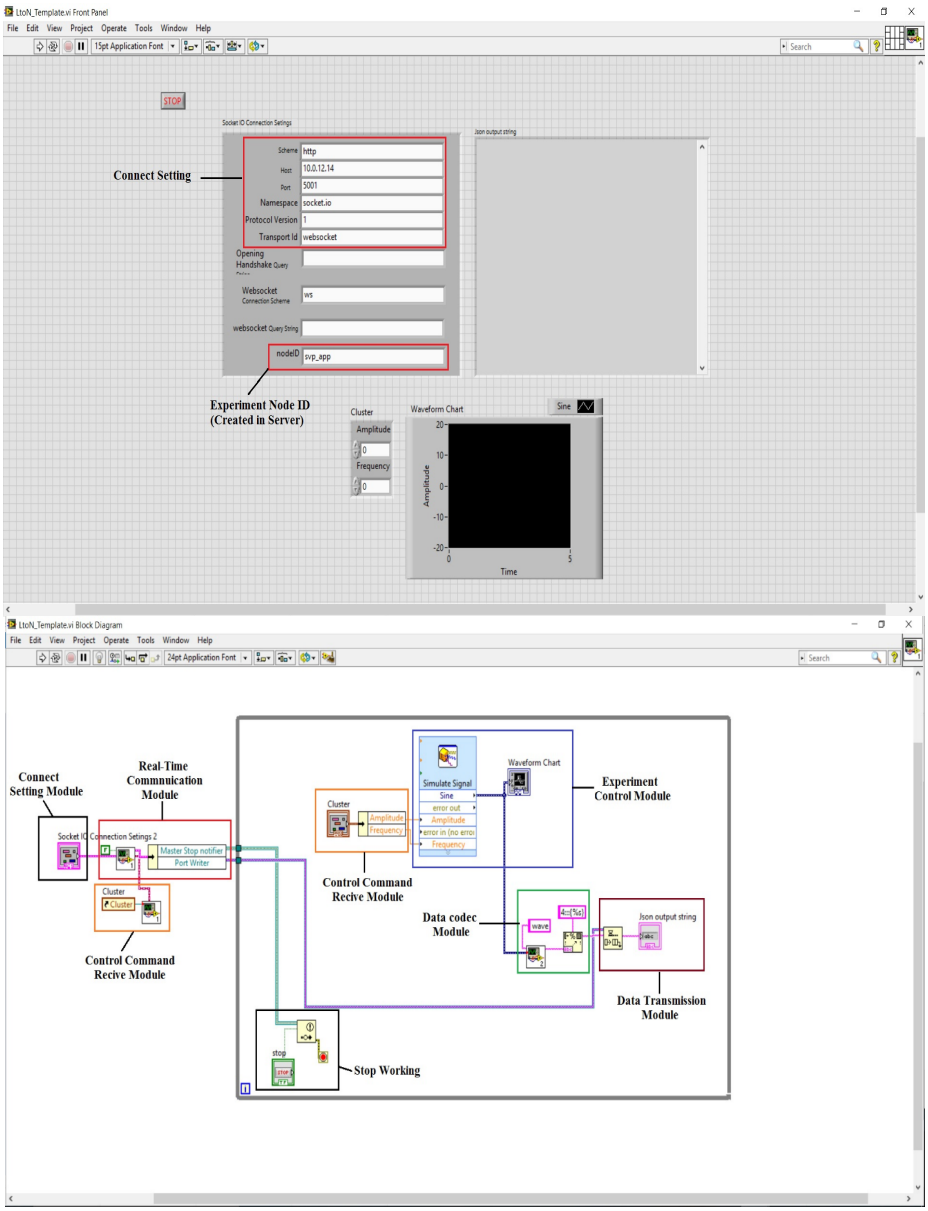

Fig. 5. the code examples of new LtoN_UID module with experiment ID in LabVIEW.

4 Implementation Discussion

To illustrate the effectiveness of the new flexible framework, a SVP experiment and a PID motor speed control experiment have been rapidly connected with users. Fig. 6 shows the whole implementation process of two remote experiments based on the new flexible framework. The detailed process of the experiment integration is given below

4.1 Experiment Hardware Setup.

The SVP hardware platform include a flexible steel frame fixed on top of a plexiglass box, a motor, two Shape Memory Alloy (SMA) wires and a purchased magnetic iron clamped on a container of Magneto-Rheological (MR) fluid. To measure the acceleration, vibration of the platform and temperature of SMA, a data acquisition device, NI DAQ 6008 USB, to get the experimental data from the sensors of SVP.

The PID motor speed control hardware contains a) DC motor with tachometer, b) power amplifier c) power supply for the amplifier, d) NI USB-6361 X Series DAQ and e) PC workstation.

4.2 Experiment Software Integration.

The software implementation of two remote experiments includes three tasks: the client module implementation, server module integration and experiment control application implementation.

Client Web module Implementation. To implement the client app, HTML5 Technology is used to implement the user interface. With the server-based Mashup technology, the data was analyzed and reformatted on the server module, and then the data was transmitted to the user's browsers. To achieve the real-time communication with server module, a communication function with UID is implemented using Socket.IO module in client app. Through this communication function, client app can be rapidly and flexibly connected with two experiments through server module.

As shown in Fig. 6, to implement the web application, HTML5 Technology is used to implement the user interface. The new LtoN_UID protocol module is used to implement the real-time data communication. With the server-based Mashup technology, the data was analyzed and reformatted on the remote server, and then the data was transmitted to the user's browsers. The UI includes three parts: 1) experiment real-time video; 2) real-time experiment data display; 3) experiment control components.

Server Module Implementation. The server module is directly built on the top of a wiki based remote laboratory platform [19]. It includes two server engines working together, Apache server engine and Node.js server engine. Meanwhile, the Node.js server engine to support the new LtoN_UID protocol module to handle the experiment data in real-time transmission. Based on the online IM application architecture, a communication management module, which is used to handle the real-time data transmission through the unique ID, is implemented. Moreover, the user management

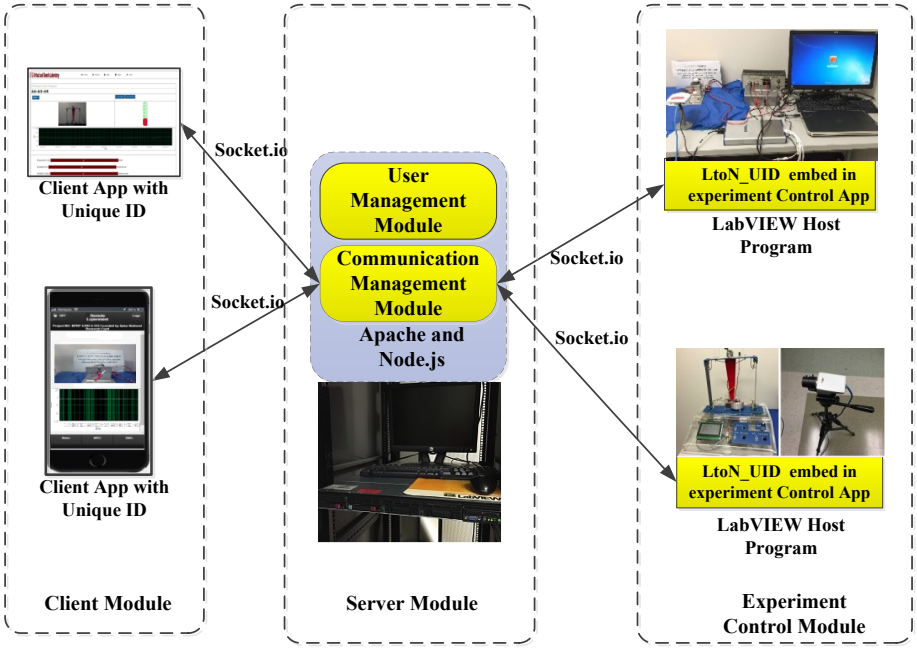

Fig. 6. The whole implementation process of a remote PID experiment and a remote SVP experiment.

module, which is used to arrange and manage the user unique ID, is implemented and integrated into the server module as well.

Integrate the New LtoN_UID Module to the Experiment Control Application. With the new version LtoN module, students can conduct the experiment, save the experiment data to file system. More details of this new real-time data transmission protocol are illustrated in the following:

- The new module includes two parts, experiment equipment control part running in workstation and implementing by LabVIEW; and server part running in web server which was developed using JavaScript language and enhanced by the Socket.IO.
- In this new LtoN_UID module, some special communication instruction set based on the experiment unique ID to connect the experiment with server-side user management module to handle the real-time data transmission.
- In this new module, we revise some brief instructions to control experiment progress for improving the real-time communication performance.

Based on the above novel flexible framework, users can build up a distributed remote laboratory and rapidly and flexibly integrate any experiment developed by LabVIEW via its unique ID into the remote laboratory.

5 Future Works

Although the novel flexible framework delivers a new rapid integration approach to support remote laboratory development, there are still further development required to improve the new framework stability and usability. More specifically, issues that need improvement are as follows:

1. Integrating the novel flexible framework into more Learning Management Systems. Currently, we implemented our Wiki-based remote laboratory management platform based on this novel framework, and it includes a scheduler, a communication management module, a user management module, and a learning materials management module. In future, we plan to integrate the novel flexible framework into some popular LMS (e.g., Moodle, MIT iLab Shared Architecture) to avoid double registration of students.
2. Integrating some industrial experiments based on our novel flexible framework. Such as, industrial equipment remote monitoring application, industrial equipment remote training application, etc.

6 Conclusion

In this paper, a new flexible framework has been designed and implemented successfully with a combination of advantages of social online IM application architecture and the new LtoN_UID module for experiment equipment real-time control. Comparing to other remote laboratory development framework including the unified framework we designed before, this new framework solved the issues of flexibility and scalability in remote laboratory development. Remote laboratory developers can bring the LabVIEW controlled offline experiment or equipment online with less effort with this framework. This flexible framework will significantly benefit remote laboratory development for online engineering education and smart factory.

7 References

[1] L. Gomes and S. Bosgoyan, "Current trends in remote laboratories", *IEEE Transactions on Industrial Electronics*., vol. 56, no. 12, pp. 4744-4756, Dec. 2009. https://doi.org/10.1109/TIE.2009.2033293

[2] M. Tawfik, D. Lowe, C. Salzmann, D. Gillet, E. Sancristobal and M. Castro, "Defining the Critical Factors in the Architectural Design of Remote Laboratories." *IEEE-RITA,* vol. 10, no. 4, pp. 269-279. Nov. 2015. https://doi.org/10.1109/RITA.2015.2486388

[3] M.A. Prada, J.J. Fuertes, S. Alonso, S. García, and M. Domínguez, "Challenges and solutions in remote laboratories. Application to a remote laboratory of an electro-pneumatic classification cell." *Computers & Education*, vol.85, pp.180-190, 2015. https://doi.org/10.1016/j.compedu.2015.03.004

[4] A. Maiti, A.A. Kist and A.D. Maxwell, "Real-time remote access laboratory with distributed and modular design." *IEEE Transactions on Industrial Electronics*, vol.62, no.6, pp. 3607-3618, 2015.

[5] S. Farah, A. Benachenhou, G. Neveux, D. Barataud, G. Andrieu and T. Fredon, "Multi-User And Real-Time Flexible Remote Laboratory Architecture for Collaborative and Co-operative Pedagogical Scenarios." *iJOE*, vol.12, no.4, pp. 33-36, 2016.
[6] N. Wang, X. Chen, G. Song, and H. Parsaei, "Using Node-HTTP-Proxy for Remote Experiment Data Transmission Traversing Firewall." *International Journal of Online Engineering (iJOE)*, Vol 11, No.2, pp.60-67, 2015. https://doi.org/10.3991/ijoe.v11i2.4443
[7] N. Wang, X. Chen, G. Song, and H. Parsaei, "Remote experiment development using an improved unified framework." *in World Confer- ence on E-Learning in Corporate, Government, Healthcare, and Higher Education*, vol. 2014, no. 1, pp. 2003–2010, 2014.
[8] C.T. Moncreiff, "Computer network chat room based on channel broadcast in real time." *U.S. Patent* 6,061,716, 2000.
[9] M. Dereshiwsky, "Equity in the Online Classroom: Adolescent to Adult." *In Proc. Social Justice Instruction Conf*. 2016, pp. 33-42. https://doi.org/10.1007/978-3-319-12349-3_4
[10] S.L. Bangare, S. Gupta, M. Dalal, and A. Inamdar, "Using Node. Js to Build High Speed and Scalable Backend Database Server." *In Proc. NCPCI. Conf. 2016*, vol.2016, pp.19.
[11] T. Hughes-Croucher, M. Wilson "Up and Running with Node.js ", *Node: Up and Running (1st Ed.), Sebastropol: O'Reilly Media*, pp. 4-11, ISBN 978-1-4493-9858-3, April, 2012
[12] P. Teixeira, "Professional Node.js: Building Javascript based scalable software." *John Wiley & Sons*, 2012.
[13] I. Fette, "The websocket protocol." *Internet Engineering Task Force (IETF),* 2011. pp. 5-6, ISSN: 2070-1721.
[14] V. Pimentel, B.G. Nickerson, "Communicating and displaying real-time data with WebSocket." *IEEE Internet Computing*., vol.16, no.4, pp.45-53, 2012. https://doi.org/10.1109/MIC.2012.64
[15] V. Wang, F. Salim, P. Moskovits, "The definitive guide to HTML5 WebSocket" *Berkeley, Calif*, 2013, Vol. 1, USA: Apress, https://doi.org/10.1007/978-1-4302-4741-8
[16] R. Rai "The Socket.IO protocol", *Socket.IO Real-time Web Application Development, Sebastropol: O'Reilly Media*, pp. 87-91, ISBN 178-2-1607-87, February, 2013.
[17] N. Wang, X. Chen, G. Song, and H. Parsaei, "A novel real-time video transmission approach for remote laboratory development." International Journal of Online Engineering (iJOE), Vol. 11, No. 1, pp. 1–4, 2015.
[18] Wang, N., Chen, X., Song, G. and Parsaei, H., 2015. "An Experiment Scheduler and Federated Authentication Solution for Remote Laboratory Access." International Journal of Online Engineering, 11(3).
[19] N. Wang, X. Chen, Q. Lan, G. Song, H. Parsaei, and S.C. Ho, "A Novel Wiki-Based Remote Laboratory Platform for Engineering Education." IEEE Transactions on Learning Technologies, Vol. PP, No. 99. 2016 https://doi.org/10.1109/TLT.2016.2593461

8 Authors

Ning Wang is with University of Houston, Texas, USA.
Gangbing Song is with University of Houston, Texas, USA.
Xuemin Chen is with Texas Southern University, Texas, USA (chenxm@tsu.edu).

Article submitted 22 September 2017. Published as resubmitted by the authors 27 October 2017.