ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 159

International Conference on Mathematics, Modelling, Simulation and Algorithms (MMSA 2018)

Design and Research of the Spoken English Test System Based on Node.js

Yansong Cui, Zhongyuan Yu and Jianming Huang
Beijing University of Posts and Telecommunications, Beijing, China

***Abstract*—Online spoken English test system has become one of the most important methods of examination in the Internet era. Compared with the traditional software developed by C++/C#, this paper mainly uses Nw.js development framework based on Node.js platform, which reduces the cost and difficulty of system development and enhances software compatibility and flexibility [1]. The server based on Node.js gets examination papers from the cloud and sends them to the client while the client records audio and sends the files coded with Lame back to the server, between which we use WebSocket for TCP/IP communication. The test result shows that the system can provide real-time online spoken English test for up to 5000 users.**

Keywords—recording test; Node.js; WebSocket; lame

I. INTRODUCTION

With the rapid development of Internet and computer technology, intelligent teaching system has gradually become a hot research topic. Test system that can automatically download papers, record voices, and upload answers can greatly improve the efficiency of school tests, which can not only alleviate the shortages of the supervisor and tedious work pressure but also raise the examinee's reading interests and enthusiasm. According to the designed methods proposed in this article, we used Nw.js to build a client that can run on the desktop and Lame to code user's input audio for MP3 files, then we sent the audio files to the server by WebSocket protocol, finally developed a multi-user real-time examination system of low cost and high efficient.

II. SYSTEM STRUCTURE

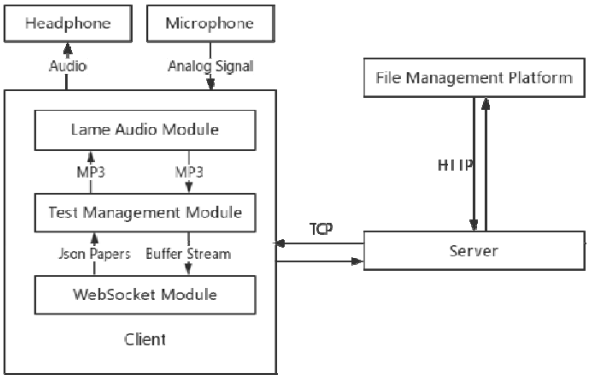

FIGURE I. SYSTEM STRUCTURE

The test system consists of audio recording equipment, headphone devices, client and server based on Node.js and remote background system, as shown in Figure 1.In this system, the server first sends the HTTP request to the background system to download and save the test data in Json format. Then it broadcasts its host information by UDP protocol to all IP addresses in the same LAN.

The client registers the listening event when it runs by online student users in the same LAN. Once the broadcast signal is listened, the TCP handshake with the host will be set up to download the test papers. After receiving the test papers in Json format, the client will parse it and get the text messages and MP3 files, then export them to the client interface and the headphone terminal. The client gets microphone input and encodes them into real-time MP3 file stream. At the end of the test, the audio data is transferred back to the server by TCP, which will be restored to MP3 files then transmitted back to the background by HTTP and finally stored to the cloud. All the features of the process have been implemented by fully testing.

III. SYSTEM KEY TECHNOLOGY

A. Flexible Environment and Adaptable Interface

In order to build a low-cost, efficient and adaptable test system, traditional C++/C# language are not adopted in this paper. Instead, this system is developed based on the Node.js operating platform. Node.js is an event-driven, non-blocking programming platform based on the V8 engine. The V8 engine uses some of the latest compilation techniques to greatly improve its speed with JavaScript. Node.js is lightweight and efficient, which can be considered as the perfect solution for real-time application system in a data-intensive distributed deployment environment [2]. As a result Node.js meets the requirements for development.

What's more, we use Nw.js framework. Nw.js is one of the most popular desktop application development SDKs, which not only integrates all running environment of the Node.js, contains a large number of the Node.js module components, also brings a lot of APIs suitable for the development of the Windows desktop application. It can be very flexible to interfaces, window and background thread by parameter configuration, the biggest advantage of which is compatible with Windows XP compared with other popular frameworks.

B. Audio Encoding

1) HTML5 Audio APIs: The test system needs user's microphone input first. The traditional desktop application needs to call the interface in system level while trying to obtain the computer audio input, which is complicated and difficult to operate. With the rapid development of Web technology in recent years, HTML5 brings more perfect and powerful APIs for video and audio input of computers. We can easily access the user's microphone input by using them because Nw.js has a Chrome kernel embedded in its Node.js development environment. These APIs makes full use of the advantages of simple and flexibility of HTML5, which greatly reduces development cost [3].

2) Background Coding in WebWorders: While executing a script in an HTML page, the state of the page is unresponsive until the script is completely executed. In the program, as the recording time becomes longer, the output of the file not only has a larger time delay, but also may block threads, which affects user's interacting with the interface.

WebWorkers is a way of browser to make JavaScript running in the background. You can make some part of the JavaScript run independently of other scripts by using threads, background processes, or other processor cores, which will not affect the performance of the page. Users can continue to do interactive operations while WebWorkers is running in the background. In this paper, we imported the Lame library into WebWorkers, and implemented the real-time operation of edge recording and coding [4].

3) MP3 Encoded by Lame: The recording of the users in the test needs to be saved in a uniform and stable format, while audio files such as WAV are too large to be suitable for actual transmission. However, MP3 is still a popular kind of audio format on the Internet, the advantage of which is that its file has smaller size and less storage capacity than other formats. Therefore in this paper we chose to keep files of questions and answers in MP3 format.

In order to convert the audio stream of the system to the MP3 format, the audio needs to be digitally encoded. Using a custom encoding library is not only inefficient but also error-prone, besides, noise processing and sound fidelity is difficult to solve, which makes development very difficult.

Lame is the most popular and efficient MP3 encoding engine. MP3 encoded by Lame has tone pure, broad space, clear bass and good detail performance. Its original psychological acoustics model technology guarantees the authenticity of the audio reduction. Cooperating of VBR and ABR parameters, it can almost be comparable to CD audio sound quality while the file size is very small [5]. Therefore this paper implemented the corresponding sampling, coding and conversion of binary audio mainly through importing the library named Lame.js, which converted the original acoustic signal from user's microphone to MP3 format, completing the feature of test system in the end.

C. Network Real-time Communication by TCP

During the test the server needs the test data of multiple clients, as well as some interactive features such as sending test signals and providing login information. HTTP request usually initiate a single request from one end to the other, especially for large data volumes while the TCP mode with full duplex, real-time interaction and support of reconnection is more suitable for this system.

Here we chose the TCP communication module of Node.js named socket.io. Socket.io encapsulates WebSocket and polling mechanisms and other real-time communication methods into a common interface and implements the corresponding code for these real-time mechanisms at the server [6]. UDP broadcast technology is also used in this system, which named udp4, a module of Node.js. It also establish listening event, which receives the host and test information of the server.

IV. IMPLEMENTATION OF THE CLIENT

A. Implementation of the Interface

First of all, we need to configure the parameters of the system by writing the package.json file in Nw.js. The start-up interface resolution is set to 1366×768 while the minimum resolution is set to 1024×576, in which range the user can freely drag the window. The flex attribute is used in different regions in the page, which makes the left and right blocks keep the ratio of 1 to 5 while the up and bottom blocks keep the ratio of 1 to 2. It makes the interface with perfect adaptability. Canvas is used to make the countdown icon while the rest time is determined by setting timers, which changes the shape of the Canvas element, so we can provide users with rich and colorful interface effect [7]. Part of the interface is shown in Figure 2.

B. Implementation of Recording and Encoding

The program of the test interface first detects that if getUserMedia() method is effective, and then according to the way mentioned above, will firstly return a MediaStream object in the callback function of getUserMedia() method and secondly pass it to the AudioContext object, finally calls the corresponding method to create a MediaStreamSource object.

Once the original and operable analog signal is obtained, on the one hand, it will be passed to the GainNode, which is used to play the recorded audio to the user in real time, on the other hand, due to the user need to test the microphone volume before the test, we need to create a microphone real-time synchronized audio source. Here create a GainNode, then the user's real-time input can be heard through headphones, and you can also call its other methods to control the parameters of the GainNode.

This signal will be passed to the WebWorkers module running in the background, where the communication between the two relies mainly on the postMessage() method. In the program, firstly the processing of sampling, quantization and encoding is performed for the analog signal, and then it will be passed to the Lame encoding module, so that the audio can be encoded simultaneously in almost synchronous mode. Finally,

the high quality MP3 audio file is returned to the client. The specific process is shown in Figure 3.

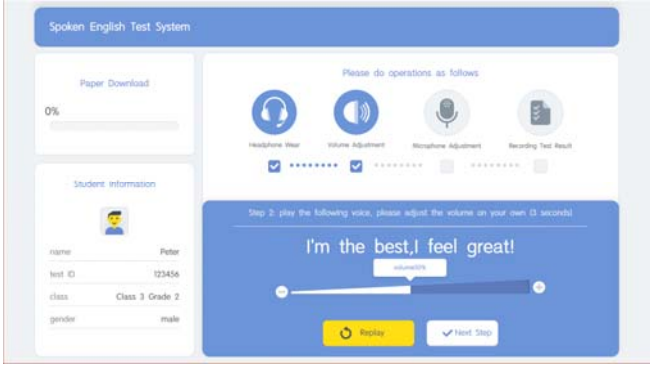

FIGURE II. HEADPHONE TESTING INTERFACE

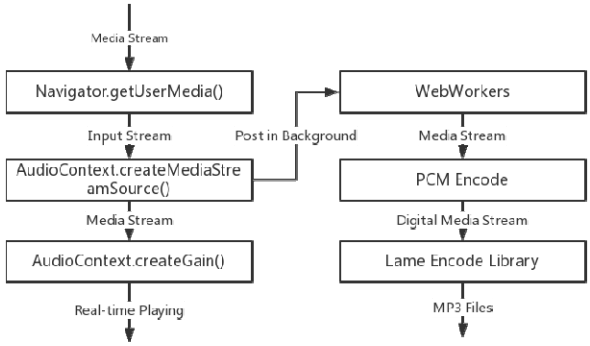

FIGURE III. AUDIO STREAM TRANSMISSION

C. *Implementation of TCP Communication and File Transfer*

By using the socket.io and udp4 modules, the server sends UDP broadcast to the LAN when its starts, which contains its own IP, port number and test information. The client first listens to the broadcast, then gets the corresponding parameters by calling the connect() method to establish a TCP connection, at the same time registers a series of socket event listeners for subsequent interactions [8].

The TCP connection set up by socket.io has great performance for binary file transfer, where we used the binary file stream for transmission between the two terminals. The final output of the audio processing module mentioned above is file stream in the form of MP3Blob. However, this type cannot be transmitted directly using the socket. So we used the FileReader class of Javascript here. When users are ready to upload the answers at the end of the test, the program firstly reads the audio files in turn, then readAsDataURL() method of the event object will be used to convert them to Base64 format. But the base64 format still contains redundant information, so we need to use the Buffer class of Node.js. As a result, the data can be eventually converted to binary Buffer type [9].

D. *Exception Handling*

Due to the instability of network and other sudden interference factors in practical application scenario, it could lead to network paralysis or software accidentally closing. While this situation can by recovered by configuring the parameters of reconnection in socket.io. Once the network is reconnected, TCP connection will be re-established, the information of communication before can be restored immediately.

If disconnection happens in some steps especially during the exam, a file is needed to storage current client information, which should include the test machine's overall progress, the page address, the completed recording data and so on. According to this information structure, we adopted the Json format to save these information. Every time the software goes to a new step, information will be written into the debug with fs.writeFile() method of Node.js. And the Json file is read first by the client before it starts every time, once the abnormal state saved previously is detected, it will read all the test information and the client status will be recovered fully before the break. After fully testing, the client after the start can be back to the previous interface and maintain correct parameters even in any one place unexpectedly.

V. SYSTEM TEST

The system was tested after the completion of the development, which mainly investigating the system's functional integrity, cross-operating system compatibility, resource appropriation, and the stability of the system on computers of different configurations.

A. *Performance Test*

Based on the functional requirements of the test system, the performance test selected the typical application scenario to simulate the real user initiating the system access request, including the main steps in the examination process. After installing the software on the computer with the common operating system, the simulation is conducted according to the test process. The specific contents are shown in Table 1.

The CPU, I/O and memory resources utilization of client computer in the testing process is smooth, without downtime, which is basically stable. After sufficient testing, the system can be fully compatible with the above operating system, and can meet all the functions designed. The specific test data is shown in Figure 4 and Figure 5.

As can be seen from the figures, when the system is running on computers of different configuration, their average CPU usage is less than 25%, while their average memory usage rate is lower than 40%, and the system average delay is under 40ms. It shows that the system has a good performance and can satisfy actual application scenario.

TABLE I. CONTENT OF PERFORMANCE TEST

Name	Content
Link	link, sign in, paper download, audio saving, files upload
Indicator	average response time, CPU occupancy, memory occupancy
System	WindowsXP,Windows7,Windows10

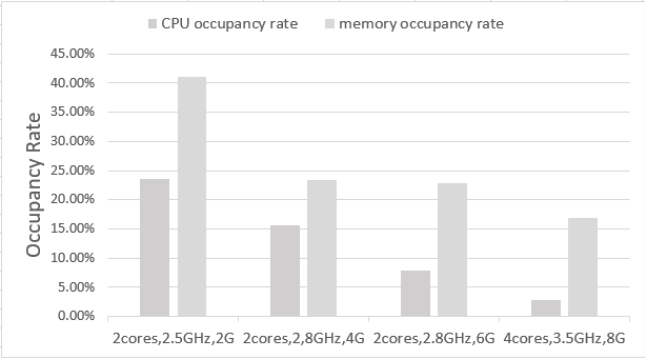

FIGURE IV. THE OCCUPANCY RATE OF DIFFERENT CPU AND MEMORY.

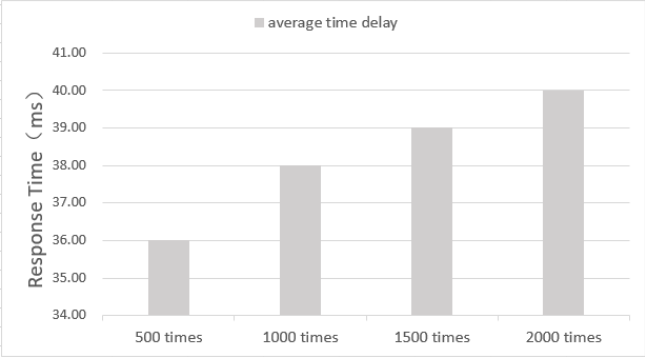

FIGURE V. AVERAGE DELAY OF DIFFERENT TEST TIMES.

B. *Stress Test*

The system also needs loading capacity test, where we used TCP/UDP Test Tool designed for WebSocket. The target server IP, port number, name of the socket events, information and parameters such as maximum number of connections can be configured in the software. It also supports for multiple codes, and can display real-time connection status and the amount of data. The specific test interface is shown in Figure 6.

We set the number of users to 500, 2000 and 5000 through the stress test tool to simulate the application scenarios and sent the data of different size. Respectively according to

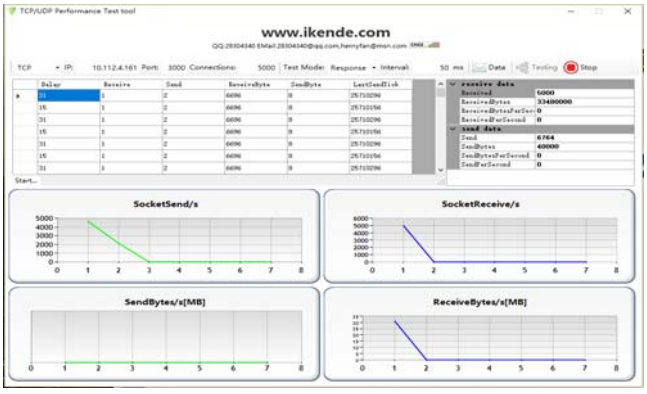

FIGURE VI. STRESS TEST OF DIFFERENT USERS

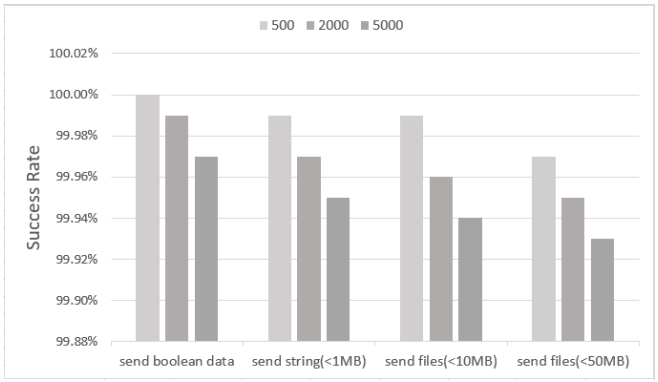

FIGURE VII. THE SUCCESS RATE OF DIFFERENT DATA AND NUMBER OF CONNECTIONS

the results in Figure 7 we can see under the high density of users the system also has high stability. It supports up to 5000 users to send data at the same time, which can meet the needs of school application scenario.

VI. CONCLUSION

This paper mainly introduces the test system based on the Node.js, which after full testing can satisfy the needs of schools, letting the client automatically download test papers, record answers and upload test files. It not only makes daily English teaching easier, also improves the students' interest in learning, making a step forward in the direction of intelligent education. The system is small in size, fast in operation, good in cross-platform and stable in performance, and has wide application and commercial value.

REFERENCE

[1] [1]Tan Xue. Interactive design and implementation of automatic railway ticketing system. [D]. China Academy of Railway Sciences,2016.

[2] [2]Node.js: A new Web application build technology. [J]. Wang Jinlong, Song Bin, Ding Rui. Modern Electronics Technique. 2015(06)

[3] [3]Chen Letian. Design and implementation of video module of online learning platform based on HTML5 video control technology. [D]. Tianjin Normal University,2017.

[4] [4]Yu Qiyang. Embedded JavaScript engine parallelization research and design. [D]. University of Electronic Science and Technology of China,2013.

[5] [5]Hu Xuemei. To improve the real-time CD recording speed of lossless audio coding. [D]. University of Electronic Science and Technology of China,2015.

[6] [6]Chen Xiqiu. The implementation of two-way real-time communication between the server and the browser based on the Socket.IO framework. [J]. Journal of Changjiang Engineering Vocational College,2016,33(01):31-32.

[7] [7]A Review of “The Modern Web: Multi-Use Web Development with HTML5, CSS3, and Javascript”[J] . LisaA. Ennis. Journal of Web Librarianship . 2014 (1)

[8] [8]J. Domańska,A. Domański,T. Czachórski,J. Klamka. Fluid flow approximation of time-limited TCP/UDP/XCP streams[J]. Bulletin of the Polish Academy of Sciences Technical Sciences,2014,62(2).

[9] [9]Munawar Hafiz,Samir Hasan,Zachary King,Allen Wirfs-Brock. Growing a language: An empirical study on how (and why) developers use some recently-introduced and/or recently-evolving JavaScript features[J]. The Journal of Systems & Software,2016,121.