Remote Internet Speed Test

[1]J.Betcy Vinitha [2]J.S.Sherine Shruthi [3]Mrs. Sherril Sophie Maria Vincent [4]Godwin Sam Josh
[1,2]Student [3]Assistant Professor [4]Technical Director
[1,2,3]Department of Electronics and Instrumentation Engineering
[1,2,3]Loyola-ICAM College of Engineering and Technology(LICET)
Chennai, India [4]GloriaTech
Chennai, India

Abstract

Among the panoply of applications enabled by the Internet of Things (IoT), the essential part of the system is the internet as they link the devices together. The existing problem is the inability to know the status of a remote sensor or tracking device that is connected to the internet from a moving vehicle or a device fixed in a remote industrial location where there is lack of access to good quality internet. The other area where a remote speed test is useful is for service providers to monitor the status of their Internet Servers to enable them to provide better quality services to their clients. This project focuses on the use of Node.js, Express and Socket.IO to solve this very specific problem. Through the Remote Internet Speed Test (RIST) application, we resolve this issue by enabling the administrators of these devices to always know the quality and status of the internet connected to it irrespective of where they are and where the devices are located. Specifically, it does not matter even if the device and the person trying to remotely test the speed of the device are behind NAT or Firewalls as long as they both have access to the internet. The ability to connect to each other across routers and firewalls is enabled by the Socket IO server. Devices register to the server using their unique device id's. The results of the application enable the administrators to improve their services as faster speed equates to better performance.

Keyword- Internet of things (IOT), Remote speed test, Socket IO, asynchronous communication, passportjs

I. INTRODUCTION

The vast network of devices connected to the Internet, from cars to washing machine and almost anything with a sensor or tracking device on it to collect and exchange data with each other is known as internet of things (IOT). This technology is being adapted in so many different fields like IT, business, cities etc. since they enable them to work more effectively through internet. Fueled by the adaptation of a variety of enabling devices such as embedded sensor and actuator nodes, the IoT has stepped out of its infancy and is on the verge of revolutionizing current fixed and mobile networking infrastructures into a fully integrated Future Internet [1].However, there are many challenges in incorporating the IoT devices into wireless systems that include sharing massive volumes of data, coexistence with existing systems and mainly to know the efficiency of each IOT enabled devices. This paper offers a solution to know the status and quality of internet in those IOT enabled devices to better understand their performance and efficiency.

The performance of an Internet connection is based on the number of bytes per second that data travels between the client and the server. If it is from the user's device to the Internet it is known as upload speed and if it is from the Internet to the user's device, it is known as download speed. We provide an opportunity to test the speed of internet remotely through RIST.

Node.js is an open-source, cross-platform JavaScript runtime environment for developing a diverse variety of server tools and applications. Although Node.js is not a JavaScript framework, many of its basic modules are written in JavaScript, and developers can write new modules in JavaScript. The runtime environment interprets JavaScript using Google's V8 JavaScript engine [12]. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications [13].

Socket.IO is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. Socket IO has two parts: Socket IO Client and Socket IO Server [2]. Use of Socket IO enables us to create an asynchronous communication between the server and the client. We use socket IO server to establish a connection to the remote client. This server is being hosted by the webserver Express which is built on Node.js.

The paper is organized as follows: the details on working of internet speed test is first given in Section II. We then present the architecture of remote internet speed test application in Section III, and particularly focus on the working of RIST application in Section IV. The various applications of RIST are listed in Section V. We also present the future thoughts and enhancements in Section VI.

II. WORKING OF INTERNET SPEED TEST

Speed test is calculated by the data transfer between the client (user's device) and an external server hosted by many people across the globe. First the client device performs ping to find the nearby servers. Ping is a computer network administration s/w utility used to test the reachability of a host on an Internet Protocol (IP) network. It measures the round-trip time for messages sent from the originating host to a destination computer that are echoed back to the source [3]. Ping with short duration denotes the nearest server. We choose nearby servers to avoid conflicts in latency. In case the server is hosted at a larger distance, then the jumps though many routers to reach the server cause variation in speed test results. When measuring internet speed, the measurement can be broken down into three categories, each with their own separate purpose and function. These categories are: download speed, upload speed and latency. Download speed refers to the speed at which a user receives information from a location on the internet, such as streaming videos on YouTube. On the other hand, upload speed measures the speed at which as user can send information to a location on the internet, such as attaching files or photos to an email. Latency is a little bit more removed from download or upload, instead it measures the length of time that occurs between the transfers of information [4].

III. SYSTEM ARCHITECTURE OF RIST

The RIST system architecture consists of the register user, Socket IO viewer client, Express webserver, Socket IO device client and the external server. It is mandatory for users to register into the RIST system and add their devices. A user can add the devices to the RIST system by using the unique machine ID which can be generated by a module available in Node.js. Admin has the ability to manage the users. When the user logs in with valid credentials he can view the list of his registered devices. On choosing a device, Socket IO server establishes an asynchronous communication with the Socket IO client device. Then the remote client device performs speed test with an external best server found using nearest ping. Then the result of the speed test is passed to the socket IO server. Finally, the speed test result is displayed on the client screen. Our application has three sections: Socket IO server, Socket IO device client and viewer client in form of web browser. The working of these three modules are elaborated in the following section.

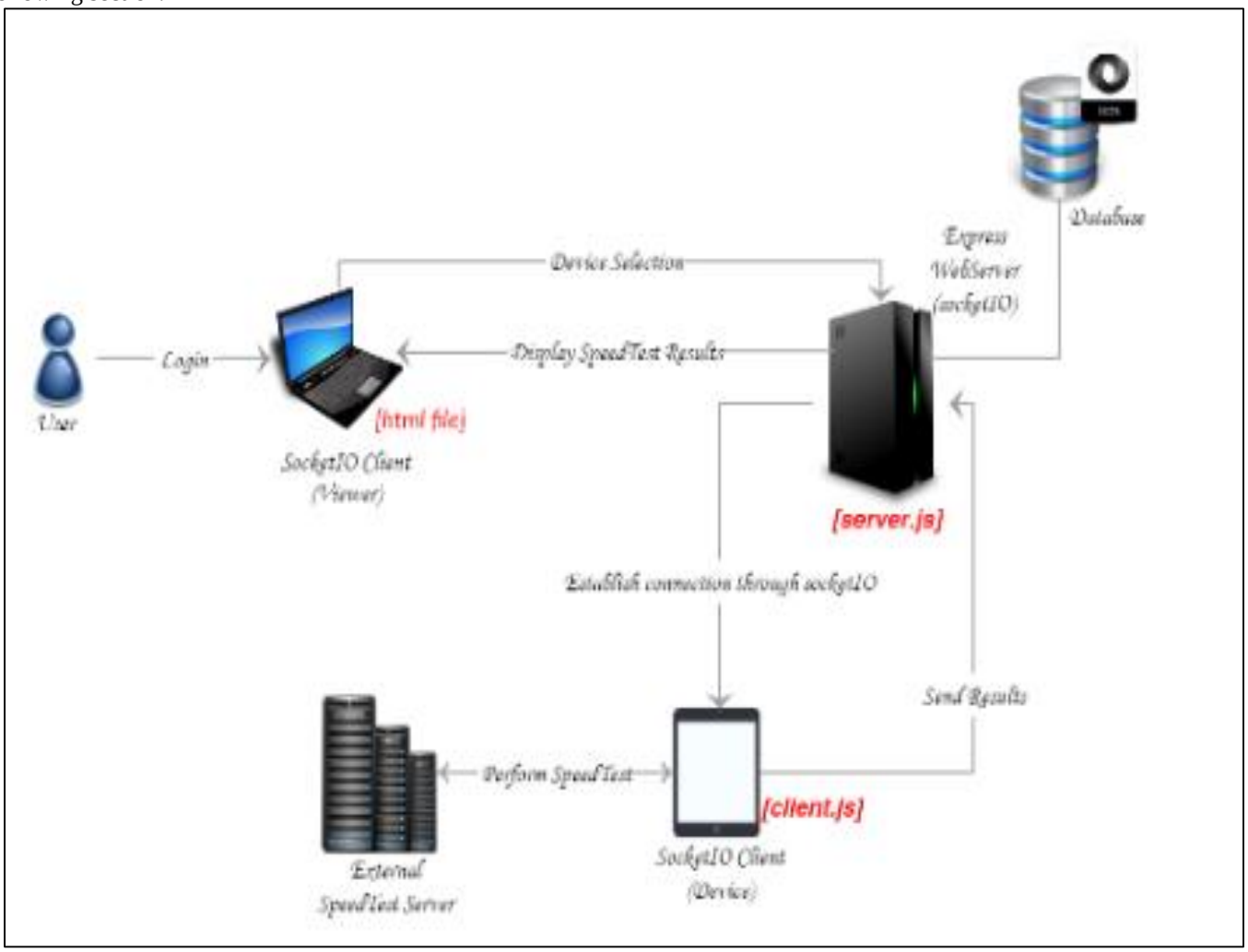

Fig. 1: RIST System Model

IV. WORKING OF RIST

The first module in RIST system is the server.js file, then the client.js and finally the html web browser for the users to interact with the RIST system.

A. SOCKET IO SERVER

We have created a socket IO server module and named the file as server.js. This file is being hosted by a hardware server. It has the information about the various users and their registered devices in form of JSON files. Each user has a separate JSON file with his login credentials like email, name, login ID, password and list of registered device. Once the user selects a device to perform the internet speed test, this choice is sent to the socket IO server. Then the socket IO server establishes an asynchronous connection to the client device that was chosen. This connection is established irrespective of client initiating the connection first. Generally, and in old trends a server can contact a client only after the client sends a request for the server through the webserver. But here the connection is being established whether or not client sends out request for the server. This enables the remote client device connect to the server. Once the client connection is established the socket IO server send out a command to start the speed test in the client device through the emit function. After the speed test performance in the client device the result is passed to the socket IO server which re-routes the result to the user's browser who chose the device.

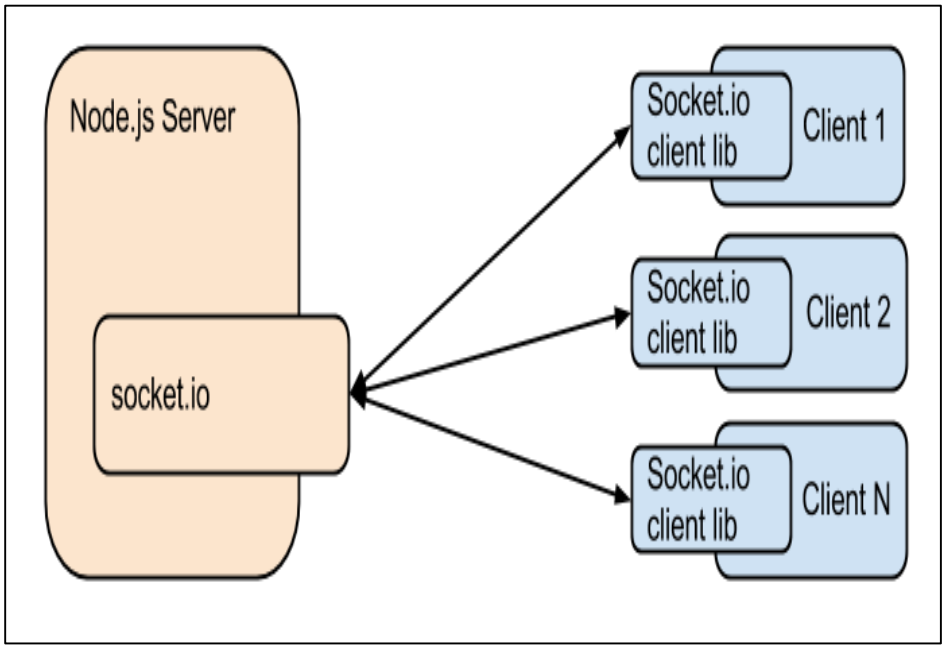

Fig. 2: Socket IO Connection [5]

B. SOCKET IO DEVICE CLIENT

The client.js file is created to run inside every device such that they can connect to the socket IO server. This file must be running inside the remote devices and the device should have an active internet connection for the server to test its internet speed. This file has two main libraries by Node.js: machine-uuid and speedtest-net. The machine-uuid library helps in generating the machine's unique ID and adds the devices into the socket IO user [6]. Administrator can add devices into the RIST system manually by entering the machine ID or by running client.js file in the device. But when the user adds the devices it is registered under his unique ID in the RIST system such that he can check its internet speed from any location irrespective of the device's location too. The speedtest-net library helps the device perform speed test with an available and active external server hosted by anyone around the world [7]. It choses servers from the servers hosted in speedtest.net website. Many individuals and organizations have registered their servers in this website [8]. At the end of the speed test, the result is sent to the socket IO server and the server displays the results to the user in the web browser.

C. VIEWER CLIENT

Viewer client or web browser or http browser is created using html and plain JavaScript. This creates the interface between the user and the RIST system. Designing of the web browser is done using bootstrap. Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only [9].We make use of passportjs, an authentication middleware of Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application. A comprehensive set of strategies support authentication is offered using a username and password or Facebook or Twitter, and much more [10]. We use google login strategy to make it simpler and easier for the user. The npm passport-google-Oauth is used to

create the google authentication strategy. In order to display the speed using a speedometer we use the plug-in just gage. Just Gage is a handy JavaScript plugin for generating and animating nice & clean gauges. It is based on Raphael library for vector drawing, so it's completely resolution independent and self-adjusting [11]. The user has to register by giving valid credentials and add the devices to the RIST system through this UI. Then the user is directed to a page where his registered devices are displayed. In this page he can add more device, delete an existing device from his list of devices or choose a device to perform speed test. If he does device selection, then the user is guided to the next page where he can view the speed test results. Then he can go back to the previous page to test another device or logout of the RIST system.

V. RIST APPLICATIONS

RIST can be used in devices where high data transfer rate is required to provide a faster service. This includes devices used in live online streaming. In order to stream a live occasion or event like wedding we need high internet speed.

We can use RIST in IOT devices which must have high internet speed in order to transmit data or a message very swiftly. For instance, early warning system must be connected to high speed internet to alert the related individuals [14]. Another example is IOT enabled sensors which must be connected to high speed internet to continuously work efficiently. And all these devices or sensors needn't necessarily be in one place. They will be located in different places. Thus RIST enables the device administrators to check the speed of all these devices from one convenient place.

VI. CONCLUSION AND FUTURE ENHANCEMENT

In this paper we have provided a solution to test internet speed of any device remotely irrespective of the location of both the user and the device through Remote Internet Speed Test (RIST) system. This can be used by internet service providers who wants to view their service on client side or by organizations who provide services that depends on the internet access with high speed.

In future RIST system can be enhanced to store the history of the results such that users can know about the improvement of their internet speed. The RIST system can be made to perform analysis on the history of speed test results and provide the user with graphical representation of the analysis to help them improve their services.

REFERENCES

[1] Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami, "An Information Framework for Creating a Smart City through Internet of Things," IEEE Internet of things journal, vol. 1, no. 2, April. 2014
[2] "Socket.IO". En.wikipedia.org. N.p., 2017. Web. 2 Mar. 2017.
[3] "Ping". En.wikipedia.org. N.p., 2017. Web. 2 Mar. 2017.
[4] "How is internet speed measured ". Testinternetspeed.org. N.p., 2017. Web. 2 Mar. 2017.
[5] Fabiano, Simone. "Benchmarking Socket.IO Vs. Lightstreamer with Node.Js". Blog.lightstreamer.com. N.p., 2017. Web. 2 Mar. 2017.
[6] "Machine-Uuid". npm. N.p., 2017. Web. 2 Mar. 2017.
[7] "Github - Ddsol/Speedtest.Net: Node.Js Speedtest.Net Client Module". Github.com. N.p., 2017. Web. 2 Mar. 2017.
[8] "Speed test servers ". Speedtest.net. N.p., 2017. Web. 2 Mar. 2017.
[9] "Bootstrap (Front-End Framework)". En.wikipedia.org. N.p., 2017. Web. 2 Mar. 2017.
[10] "Passport". Passportjs.org. N.p., 2017. Web. 2 Mar. 2017.
[11] "Justgage - Nice & Clean Dashboard Gauges". JustGage. N.p., 2017. Web. 2 Mar. 2017.
[12] "Node.Js". En.wikipedia.org. N.p., 2017. Web. 2 Mar. 2017.
[13] "Express - Node.Js Web Application Framework". Expressjs.com. N.p., 2017. Web. 2 Mar. 2017.
[14] Stefan poslad et al.," A Semantic IoT Early Warning System for Natural Environment Crisis Management," IEEE Transactions on Emerging Topics in Computing, vol. 3, no. 2, May.2015
[15] Shaofang Gong and Magnus Karlsson," Pushing the Wireless Data Rate to the Internet Speed," IEEE Access Special section on optimization for emerging wireless networks: iot, 5g and smart grid communications, vol.4, November.2016

