Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 5, Number 1, pages 34–39, January 2016

Prtformance Eveluation of Ad Hoc Network Protocol Implemented in Server Side Java Script

Atsushi Ito, Hiroyuki Hatano, Masahiro Fujii,
Mie Sato, Yu Watanabe
Graduate School of Engineering
Utsunomiya University
7-1-2 Yoto, Utsunomiya-shi, Tochigi, 320-8585 Japan
{at.ito, hatano, fujii, mie, yu}@is.utsunomiya-u.ac.jp

Yuko Hiramatsu, Fumihiro Sato
Department of Economics
Chuo University
742-1 Higashinakano, Hachioji, Tokyo 192-0393, Japan
{susana_y, fsato}@tamacc.chuo-u.ac.jp

Akira Sasaki
GClue Inc.
2-1-14 Higashi-Sengoku, Aizu-wakamatsu-shi, Fukushima, 965-0818, Japan
Akira@gclue.jp

***Abstract*—We are now developing a sightseeing support system in Nikko, a world heritage in Japan, using BLE beacon and smartphone. We installed 24 beacons from Nikko Station to Shinkyo-bridge. 17 beacons were set outside, so that they had to be small and inconspicuous since they did not have to disturb scenery. So that, they have been using small button shaped battery. We have to take care these beacons. They are sometimes out of battery, and the shell of a beacon is sometimes broken. We think that one of the problems of the beacon system is maintainability. We think one possibility to solve this problem is to develop ad hoc network of beacons. This network can collect information of remaining power of battery, also we can find broken beacon if it is not included in the ad hoc network. We performed preliminary study to develop ad hoc network easily. We tried to use server side Java Script to describe ad hoc network protocol. In this paper, we describe performance and productivity of OLSR using server side Java Script. We compare OLSR implemented in Java Script and implemented in C. As the result, productivity of OLSR in server side Java Script was 1.5 times higher than that was implemented in C, however, the performance was 18 times slower than that was implemented in C.**

Keywords—Ad hoc network; server side Java Script, BLE Beacon

I. INTRODUCTION

In recent years, the development process of embedded systems becoming difficult because of increasing complexity of the controlled object. Moreover, shortening development time and cost reduction request sometimes cause degrade of the quality and reliability of the system. In order to develop a large scale and complicated system in a short period of time, it is necessary to improve the portability and reusability of software. Portability and reusability are using other system program that was created for the certain system.

In a conventional embedded system, an application works on an operating system and we have developed many techniques to realize portability and reusability of software. After introducing Android [1], the situation was changed. Android was developed on Java VM, called Dalvik VM until 2014, now ART [2]. By using Java, portability and reusability of application was increased.

At the same time, technology for WEB was also developed. Progress of HTML5 [3], function and performance of Java Script on WEB browser was progressed. Especially, V8 engine [4] for Java Script increased the performance. Traditionally, Java Script has been used in WEB browser.

A technique to use Java Script on server is called Server Side Java Script. This technique runs Java Script engine on a server. By using this technique, it is possible to use Java Script to describe applications such as HTTP server on a server.

One of the popular server side Java Script environments is node.js [5]. This environment has been developing from 2009. If we describe applications on server side Java Script such as node.js, such application has high portability and reusability since they are independent from operating system such as Linux, Windows, Mac, and Android etc.

On the other hand, we are now developing a sightseeing support system using BLE beacon in Nikko. During this trial, we have to pay a lot of effort to check status of beacon, such as working well or not, battery is exhausted or not. We would like to introduce some functions to check status of beacons in the future. For that purpose, we would like to use a platform that is more convenient and independent from operating system. Details are described in Section 2.

For that purpose, we are considering to introduce Java Script based ad hoc network function on BLE module, since latest BLE module has ARM CPU inside [6] and provides reasonable execution power of application. In the future, there

may be much kind of beacons and we have to control different types of beacons as a group and maintain them collectively. For this purpose we have to understand performance of ad hoc network protocol in Java Script. To check the performance of ad hoc network on node.js, we implemented OLSR [7], a proactive ad hoc network protocol, on node.js and run a test program to measure the delay.

In the section 2, we briefly explain outline of sightseeing support system by using BLE beacon in Nikko. Then, we mention our previous works relating to ad hoc network and target of this research in section 3. In section 4, we describe result of evaluation and discussion. Section 5 mentions related works and section 6 describes conclusion.

II. OUTLINE OF SIGHTSEEING SUPPORT SYSTEM BY USING BLE BEACON IN NIKKO

In this section, we would like to explain outline of our project to develop sightseeing support system by using BLE beacon in Nikko.

Traditionally, travelling is getting away from daily life. Visiting unknown places is one of the great pleasures. We can discover many things such as hidden history of a village, original culture and wild nature. "Discover" means "dis"="unveil" "cover", so a travel gives us new information. The dream to discover unknown parts of the world brings us to unknown places. The satisfaction is not proportional to the amount of information.

We started a study to investigate what is the most attractive aspect of travel and how to increase expectation and satisfaction of travel in Nikko [8]. Nikko is one of the world heritages in Japan. In Nikko, there is the Tosyogu-shrine [9] that is a gorgeous grave of Ieyasu Tokugawa who is the first Syogun of Tokugawa Era. However, now, Nikko is not so famous for foreigners. A research by travel agency displayed that Nikko is not listed in top 30 locations where foreigners would like to visit [10]. This study was selected as one of research themes of SCOPE (Strategic Information and Communications R&D Promotion Programme) [11] funded by Ministry of Internal Affairs and Communications of Japan (MIC) [12].

A. Outline of the system

Last year, we performed a feasibility study for the development of sightseeing system by BLE beacon [13,14]. The purpose of the feasibility study was to collect real voice and impression on Nikko from visitors and to confirm the possibility to use BLE beacon for sightseeing. (Fig.1)

BLE beacons were installed from Tobu Nikko Station to Shinkyo-bridge, an entrance to Tosyogu-shirine as described in Fig. 2. White and red circle are the beacons that were installed in outdoors. Beacons in shops are 6 (green circle). To make it small, this beacon uses a button shape battery. Diameter of the beacon is 35mm. (Fig.3)

B. Problem to operate beacon

In the project, beacons were set outdoors and put on a navigation plate by double-sided tape (Fig.4). Also, they were installed in wide area. In this case, they were installed along the road and the distance was 1.5km. It was hard job to check problems of the beacons, such as failure of the BLE modules and battery exhaustion, almost every week. It took about one hour to check all beacons and change batteries etc. After the research period is finished, we are planning to move this application to a commercial service. It might be a problem to maintain beacons in outdoors.

We think that one possible solution is using ad hoc network to maintain beacons, since if they are connected by ad hoc network and gather information of soundness and battery level form one beacon such as located in the station or information in Nikko Station.

Fig. 1. Sample screens of sightseeing navigation application

Fig. 2. Map of locations of BLE beacons

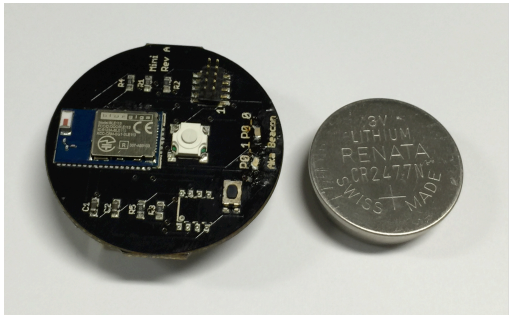

Fig. 3. Beaco module

Fig. 4. BLE beacon on a street sign

III. Our Previous Works and Target of This Research

A. Previous Works

We are developing Information Delivery System for Deaf People at a Major Disaster (IDDD) for 10 years [15] (Fig. 5). This is a group of LED displays connected by ad hoc network and some of the displays are connected to WAN. Through WAN or locally, information, such as safety information, lunch delivery time, are delivered to support people who are victims of disaster in a evacuation shelter.

In 2013, we published a paper [16] to compare AODV [17] and OLSR [7]. In this paper, for the unexpected change of network of LED displays of IDDD, OLSR performs better than AODV. In this trial, we developed application to control LED display in Java Script and we proved that the control application could display characters very fast, such as 20 characters per second. However, we implemented both AODV and OLSR in C. So that, we would like to compare performance of ad hoc network protocol written in Java Script and in C.

B. Test program

We decided to compare OLSR running on Linux and node.js as described in Fig. 6. Left side of Fig.6 means OLSR is written in C and running on Linux, on the other hand, OLSR is written in Java Script on node.js.

OLSR is a popular ad hoc network protocol. OLSR has two important functions, one is "MPR selector detection" and another is "Topology control". OLSR use a special node "MPR (MultiPoint Relay)" for flooding. Only MPR can transfer control message. By using MPR, OLSR can reduce overhead of flooding.

For the test, we implemented a test application as described in Fig.7. This application sends "Hello" message and "TC" message. "Hello" message is sent in "2 sec – random", and "TC" message is sent in "6 sec – random".

We added a "Delay Measurement Message" to measure delay of message transfer. This is a UDP packet and it contains a message. The message contains text data. If "Delay Measurement Message" is received (T1), the process starts to prepare data transfer and send it (T2). We measure delay as T2 – T1.

We implemented it both in C and Java Script.

We set two RaspberryPi modules on a desk. The distance of them was 20cm and measured delay.

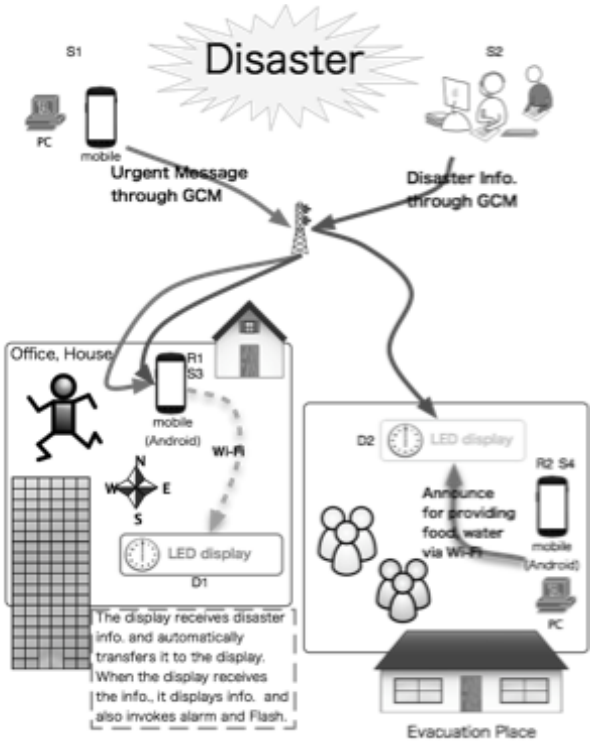

Fig. 5. Outline of IDDD

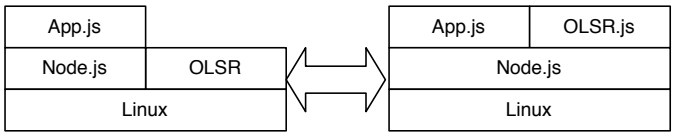

Fig. 6. Software architecture that was used for comparison

IV. Evaluation

In this section, we explain the result of evaluation.

A. Measurement of delay

We performed the test described in previous section 1,000,000 times for the test case implemented in C (Figure 8) and 100,000 times for the test case implemented in Java Script (Figure 9). Figure 10 shows a graph that removes an exceptional data in Figure 9.

Details of delay are described in TABLE II. Average delay of test program in C was 0.23millisecond and that in Java Script was 4.19 millisecond. The distribution of test program in C was 0.022 and that in Java Script was 0.826. Maximum

delay of test program in C was 206.08millisecond and that in Java Script was 12.92millisecond.

B. Measurement of productivity

To measure productivity, we counted the lines of code in C and Java Script. As described in previous section, if we assume the productivity is promotional to the numbers of line, the productivity of Java Script is about 1.6 times higher than that of in C. Also, application in Java Script works on node.js is independent from operating system. This might be another fact to contribute to productivity.

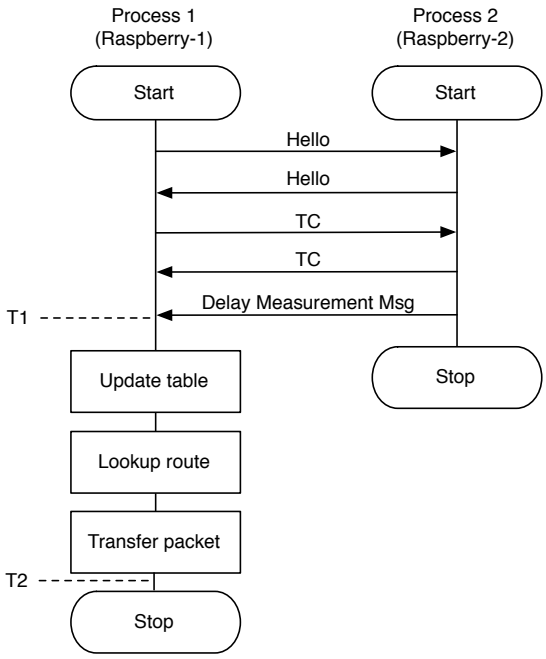

Fig. 7. Test sequence

TABLE I. TEST ENVIRONMENT

Device, SW	Model	Version	Clock [MHz]	Memory [MB]
CPU	Raspberry Pi B+		700	512
WiFi	GW-USNano2			
OS	Linux	3.6		
Node.js		0.10		
C Compiler	GCC	4.6		

TABLE II. AVERAGE OF DELAY, DISTRIBUTION AND MAXIMUM OF DELAY

	Average delay	Distribution	Maximum delay
Java Script	4.19 [ms]	0.826	206.08 [ms]
C	0.23 [ms]	0.022	12.92 [ms]
Ratio (JS/C)	18.22	37.55	15.95

C. Discussion

In this subsection, we discuss on the result described in above subsections.

1) Delay

In this experiment, delay of test application in Java Script is 18 times longer than that in C.

Fig. 11. shows distribution of delay except one exceptional result. This graph shows that except one case, maximum delay of application in Java Script is about 15millisecond. It might be acceptable for normal use such as to transfer user information to server from a beacon or send maintenance information.

However, for the real time service, it sometimes becomes a problem. For example, if Java Script is used for IDDD. The display can display 20 characters in a second at a maximum speed. This is a maximum speed for deaf people to read on a LED display. (Normal people cannot catch up such fast scroll.) It means that 50 millisecond for one character. If 4 displays are connected, it may cause 15 x 4 = 60 millisecond delay. This means that these displays cannot synchronize to display messages.

2) Productivity

To measure productivity, we counted the lines of code in C and Java Script. As described in previous section, if we assume the productivity is promotional to the numbers of line, the productivity of Java Script is about 1.6 times higher than that of in C. Also, application in Java Script works on node.js is independent from operating system. This might be another fact to contribute to productivity.

3) Further Study

a) Delay

As described in Fig.9, a large delay (206.08 [ms]) was measured. In another trial, we measured similar value. We have not yet found the reason.

b) Applying to BLE beacon

If we would like to use node.js, we have to check the specification of the latest BLE module.

If we would like to run node.js on Coretex-M3, it may require the following specification. Espruino [18] is one of the smallest MPU that Java Script can be executed.

Espruino's specification is as follows.

- 32-bit 72MHz ARM Cortex-M3
- 256KB of Flash memory, 48KB of RAM

For example, nRF52 [6] has the following specification.

- 64-MHz ARM Cortex-M4
- 512KB flash and 64KB RAM

The performance between nRF52 and Espruino is almost same. So that it may be possible to run node.js on nRF52.

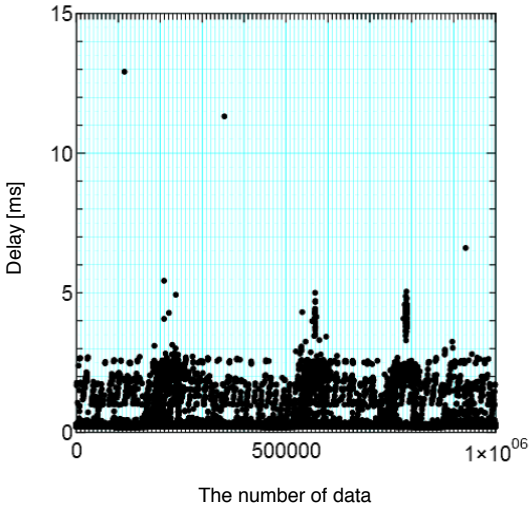

Fig. 8. Delay (in C)

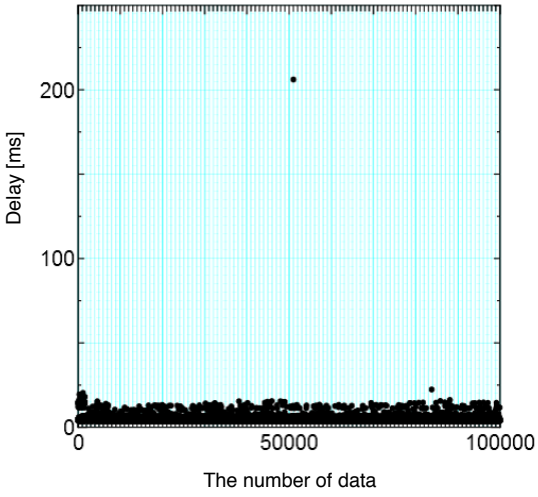

Fig. 9. Delay (in Java Script)

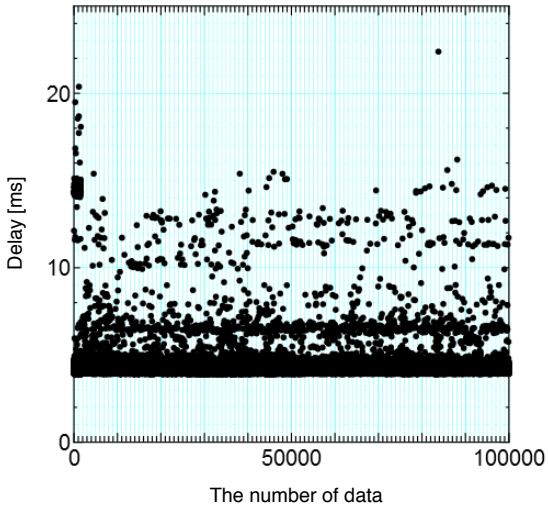

Fig. 10. Delay (in Java Script without an exceptional data)

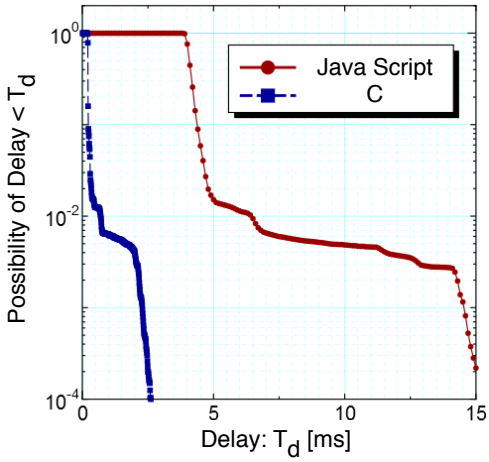

Fig. 11. Distribution of delay

V. Related Work

Reference [19] describes an example to implement high performance network protocol in node.js, unfortunately the example is not an ad hoc networking protocol,. The authors wrote "Node.js's architecture makes it easy to use a highly expressive, functional language for server programming, without sacrificing performance and stepping out of the programming mainstream. ".

We can find several papers to evaluate performance of node.js as a web server [20, 21]. They show the performance of Java Script application running on node.js is good.

VI. Conclusion

In this paper, firstly, we mention the problem that we are facing to develop sightseeing support system by using BLE beacon in Nikko. The problem is maintaining BLE beacons that are installed in outdoors.

To solve this problem, we would like to introduce ad hoc network in a beacon to maintain a beacon group. For that purpose, we are considering to introduce Java Script based ad hoc network function on BLE module.

To understand performance of ad hoc network described in Java Script, we performed comparison OLSR in Java Script running on node.js and in C.

As the result, productivity of OLSR using server side java script was 1.5 times higher than that was implemented in C, however, the performance was 18 times slower than that was implemented in C.

In addition, we compare performance of latest BLE beacon and MCU that can be used to run node.js. The specifications of them are almost same. So that, we are positive to use Java Scrip on BLE beacon and introduce portable and reusable application in a beacon to provide powerful application.

ACKNOWLEDGMENT

Authors also would like to express special thanks to Mr.Kobayashi, graduated Utsunomiya University March 2015, who supported this research.

Authors also would like to express special thanks to Mr.Funakoshi of Nikko Tourism Association, Mr.Ishihara of Educational Tour Institute, Mr.Miyamoto of Kinki Nippon Tourist Co., Ltd., Mr.Muroi and Mr.Ueda of H.I.S Co.,Ltd., Mr.Takamura and Mr.Yoshida of Hatsuishi-kai that is an association of shopping street of Nikko, Mr.Nakagawa of Kounritsuin Temple, and Dr.Nagai who is a Professor Emeritus of Utsunomiya University.

This research was performed as a project of SCOPE (Strategic Information and Communications R&D Promotion Programme) funded by Ministry of Internal Affairs and Communications in Japan.

REFERENCES

[1] https://source.android.com/index.html

[2] https://source.android.com/devices/tech/dalvik/

[3] http://www.w3.org/TR/html5/

[4] https://code.google.com/p/v8/

[5] https://nodejs.org/en/

[6] https://www.nordicsemi.com/eng/Products/Bluetooth-Smart-Bluetooth-low-energy/nRF52832

[7] RFC3626: Optimized Link State Routing Protocol (OLSR), https://www.ietf.org/rfc/rfc3626.txt

[8] http://www.city.nikko.lg.jp.e.tj.hp.transer.com

[9] http://www.toshogu.jp/english/index.html

[10] Trip advisor, “The most populer spot for visitors to Japan 2015”, http://tg.tripadvisor.jp/news/ranking/inboundattraction_2015/

[11] http://www.soumu.go.jp/main_sosiki/joho_tsusin/scope/ (in Japanese)

[12] http://www.soumu.go.jp/english/index.html

[13] http://blog.bluetooth.com/bluetooth-sig-introduces-new-bluetooth-4-1-specification/

[14] http://www.braveridge.com/BLE%20guide.html

[15] Atsushi Ito, Hitomi Murakami, Yu Watanabe, Masahiro Fujii, Takao Yabe and Yuko Hiramatsu: “Information Delivery System for Deaf People at a Larger Disaster”, Biomedical Research Vol.1, no.2, 17 Pages (2013.6)

[16] Atsushi Ito, Takao Yabe, et.al., "A study of optimization of IDDD (Information Delivery System for Deaf in a Major Disaster)", Proc. First International Symposium on Computing and Networking (CANDAR 2013), 6th International Workshop on Autonomous Self-Organizing Networks (ASON 2013), pp.422-428, 2013

[17] RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing, https://www.ietf.org/rfc/rfc3561.txt

[18] http://www.espruino.com

[19] Stefan Tilkov and Steve Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs",IEEE Internet Computing, Issue No.06, pp: 80-83 (2010 November)

[20] Kovatsch, M., Lanter, M. and Duquennoy, S.,"Actinium: A RESTful runtime container for scriptable Internet of Things applications", Internet of Things (IOT), 2012 3rd International Conference, pp.135 - 142, (Oct. 2012)

[21] Daniele Bonetta, Achille Peternier, Cesare Pautasso and Walter Binder, "S: a Scripting Language for High-Performance RESTful Web Services", PPoPP’12, pp.97-106, (February, 2012)