Vol.2021-DPS-188 No.19
Vol.2021-EIP-93 No.19
2021/9/10

A Proposal of Enhanced JYAGUCHI Architecture for Secured Service Delivery in Cloud and Edge Computing Environment

GAUTAM Bishnu Prasad[1,a)] BATAJOO Amit[2]
SHIRATORI Norio[3]

Abstract: Cloud computing is a widely researched area; however it has some challenges such as lacking of dynamic service delivery feature that can support both edge and cloud computing environment. To address such kind of challenges, we proposed an enhanced JYAGUCHI architecture, a service delivery platform implemented in our previous work that guarantees the secured and dynamic service delivery in both cloud and edge computing environment. In this research, we enhanced JYAGUCHI platform by designing a new architectural component that augments the services as per the granularity of the applications and also ensure the access control for secured service delivery. We evaluated our system by testing the performance of each component in terms of resource consumption, latency and network bandwidth metrics within and outside the edge.

Keywords: JYAGUCHI, Cloud Computing, Edge Computing, Service Delivery, Access Control

1. Introduction

The development of cloud services and its utilization in ICT industries is increasing day by day to address the needs and varieties of challenges of the end users. For instance, micro services built in cloud systems are replacing lots of legacy systems which are difficult to update and maintain by end users. However, cloud computing technologies based on service-client architecture have security challenges during service delivery from the service provider to the service consumer. Due to the growth of services and its complex architecture of cloud, it is a challenging issue to deliver service securely to the target end user. To address this challenge, initially, the concept of JYAGUCHI[1]–[4] was brought to demonstrate how this can be exported to the client as a software service.

The software delivery concept introduced in JYAGUCHI platform literally means a tap in Japanese language. As tap can regulate the intensity or rate of flow of water, accordingly the philosophy of JYAGUCHI assume that user should be able to consume the service and can regulate the frequency of usage and duration of service usage by the client.

JYAGUCHI architecture facilitates the user or service consumer to consume the service as per use basis. Furthermore, JYAGUCHI was developed on the concept of leasing an entire service item or application from a service provider rather than owning that software completely either by installing and licensing of software. From the novice user point of view, software installation, managing of its license, updating and upgrading of that software is a critical issue. However, providing those features as implemented in JYAGUCHI reduce these kinds of management cost of the users. This sort of concept is also incorporated in SaaS based application. JYAGUCHI is not only a SaaS based platform but also an architectural model that can be enhanced to model, develop and export next generation cloud based services. It utilizes the hybrid architectural model consisting of SOA (Service Oriented Architecture) and SPA (Space Based Architecture).

In JYAGUCHI platform, services are classified into Micro, Macro and Mega services according to the varieties of features implemented in services. These features are size, prices, server available distance, users' interest on the services. The core feature which is taken as a deciding factor is the size of the service. While the size of the service exceeds the threshold, it is considered as mega service in JYAGUCHI and recommended to keep it within the edge.

The other trend of service computing which is similar to JYAGUCHI is fog computing. It is a new paradigm that enhances the Cloud computing paradigm from data center plane to the clusters of end-user-devices plane which we termed fog in this paper. Cloud computing has shifted computing resources more or less from user plane to data center plane thereby centralizing the computing infrastructure into huge data centers. In contrast, fog computing decentralized the resources from cloud centers to the end-users or to the edge of the network, thus enabling a new breed of applications and services with newer potential. More specifically, fog computing is a computing paradigm that brings data processing, service utilization, networking, storage and analytics closer to the devices and applications that are more closer to the users. [6].

The main goal of Cloud computing is to leverage the Internet and provide on demand access to fundamental computing resources. For instance, cloud users can utilize and share processing power, storage space, bandwidth, memory, applications, and software in several ways. In response to their usage, cloud providers charges the users as per their consumption. This sort of business concept has been derived from the concept of utility business and thus cloud computing sometimes refers to utility computing too. In this way, users are not required to set up or to buy hardware by themselves as it used to be traditionally. It has brought a huge paradigm shift in the market [6]. However, it has not addressed all issues raised in the user front. Regardless of its supremacy in terms of providing resources to the end users, it has number of issues. Cloud computing has arisen with new data security challenges [7], [8]. Existing data protection mechanisms such as encryption have failed in preventing data theft attacks, especially those perpetrated

1 Kanazawa Gakuin University, Faculty of Economic Informatics, Department of Economic Informatics
2 Member, IEEE
3 Chuo University, Research and Development Initiative

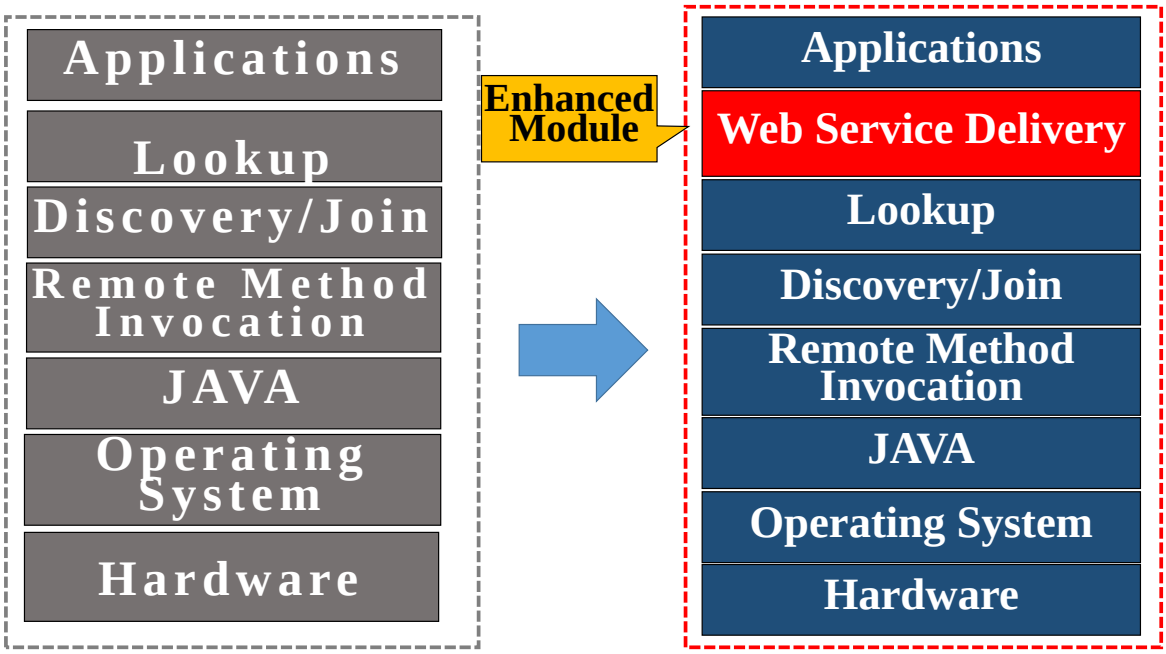

Figure 1. JYAGUCHI Server and Client Service Model

by an insider to the cloud provider.

2. Research Challenges

The distributed application architecture like JYAGUCHI must consider lots of architectural elements, components, connectors and other parameters of the system that directly or indirectly effect in the realization of the system. These elements are required to be analyzed and be designed to depict the solution before implementation leading to the best design decision in order to reduce the total cost of the system [3]. In this paper, we figured out the following research challenges which are significant to address.

A. Issue of Service Delivery to The Client Securely

Security is always a challenging issue in the cloud while providing different types of services to users. It also may reveal information which adds further complexity to security issues and risks of cloud computing systems. Most security issues in Internet are common to existing computer security problems in communication and during download and installation of software. As cloud provides a Software as a Service (SaaS), a comprehensive solution offering the entire package from infrastructure to application, service or package delivery securely is still a challenging issue. Thus, a robust access control mechanism is provided during service delivery.

B. Limitation in Service Lookup in Legacy System

The lookup service in previous JYAGUCHI serves as a central repository of services. Entries in the lookup service are Java objects, which can be downloaded as local proxies to the service that registered with the lookup service [21]. In order to utilize these Java objects, most of Java services in distributed computing provided by legacy systems needs Java supporting clients. This was also the primary limitation of this system. This limitation is overcome by enabling the services which are accessible through internet. In the enhanced architecture, distributed Java objects are accessible behind the firewalls also.

C. Dynamic Service Delivery

The network protocol that is used to communicate between a discovering entity and an instance of the discovery request service is assumed to be unreliable and connectionless, and to provide unordered delivery of packets. In an environment that makes use of IP multicast or a similar protocol, the joining entity should restrict the scope of the multicasts it makes by setting the time-to-live (TTL) field of outgoing packets appropriately [21].
This maps naturally onto both IP multicast and local-area IP broadcast but should work equally well with connection-oriented reliable multicast protocols. Dynamic service delivery is possible by providing the dynamic proxies and avail the rest of functions through single method invocation.

3. Proposed Solution

To address the above challenges, we proposed the enhanced JYAGUCHI architecture which incorporates the essence of architectural styles adopted in the distributed application and understand its inherent problem. The architectural differences between the previous JYAGUCHI and enhanced JYAGUCHI is portrayed in Figure 1. The original architecture of JYAGUCHI was started a decade ago to analyze the widely used architectural style in the file of distributed application and further deepen our understandings of internal architecture of those applications. On the basis of this architecture, we have developed some services which were developed emphasizing on the scalability of component and the reduction of dependencies among them. Furthermore, it maintains the principle of encapsulating legacy system by providing the simple methodology of interfacing the underlying software components and the way of enhancing them

to be well defined services [3].

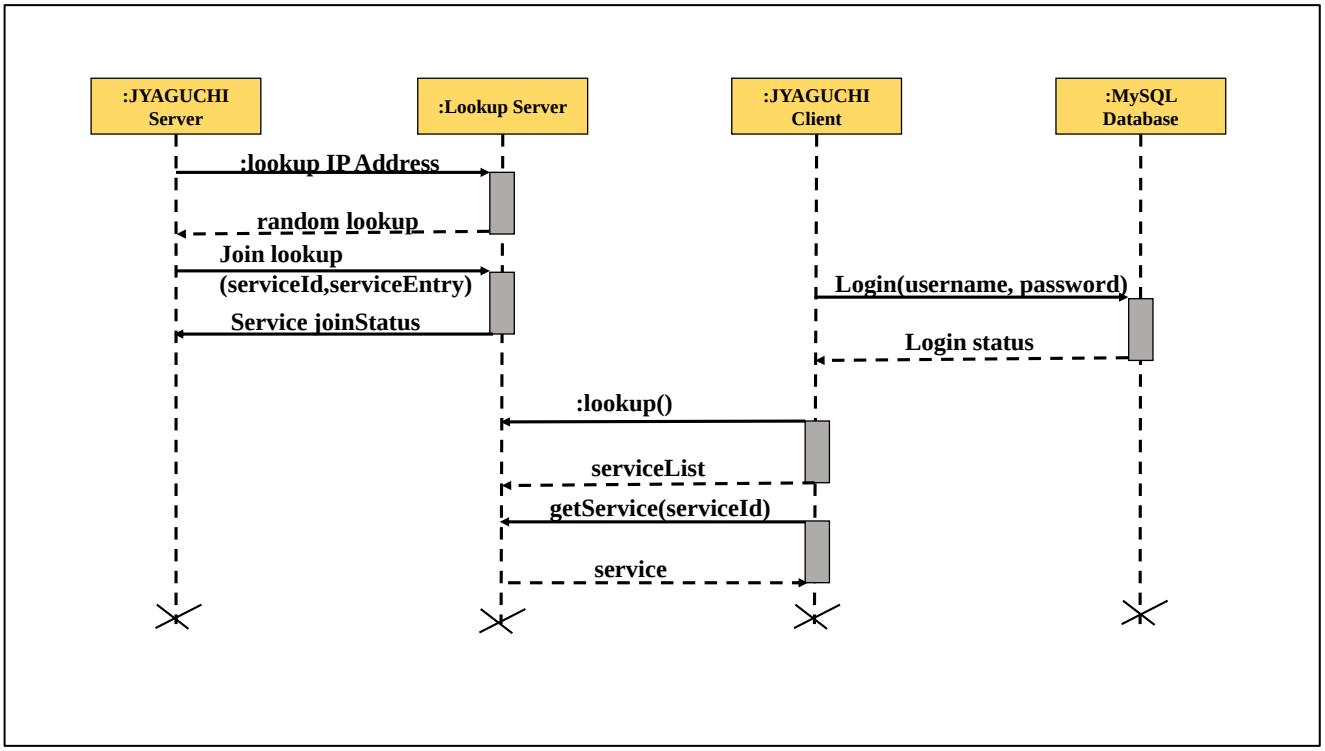

Figure 2. JYAGUCHI-sequence diagram enhanced

To address enhanced proposed solution, there is no longer any need for distributed Java application to be installed directly on the user's computer anymore, as with proposed in the Section 4 i.e., Enhanced JYAGUCHI architecture it is now accessible via Internet by entering the correct URL address.

4. Enhanced JYAGUCHI Architecture

Distributed application varies in granularity and infrastructure. Traditionally, distributed applications were two-tiered, three-tiered or multi-tiered in their architecture which collectively makes a single system. This notion of single system has evolved to the creation of from tire based system to a more huge virtualized system such as grid system [13], [14]. We experienced that the emergence of cloud computing as a new platform for enterprise. From its very inception, JYAGUCHI service development model utilizes the legacy computing infrastructure thereby creating a cluster of possible hardware that can participate in the federations of JYAGUCHI services. The hardware and services that participate in JYAGUCHI services are capable of addressing the problem in a co-operative manner. The notion of co-operation has been executed by utilizing the concept of SOA in which the underlying computing infrastructure or underlying middleware are encapsulated and the detailed of which are not required while providing the services to the end user. In fact JYAGUCHI services can be built in a number of multiple technologies and protocols [17], [18]. Though there are the different architectures underlying, JYAGUCHI service models can produce similar characteristics of web service and can be used together with other web service like technologies too. This ability is referred to as co-operative computing infrastructure. Figure 2 shows the relation between each device and the underlying software components that co-operate while developing, deploying and using of JYAGUCHI service. In particular usage scenario, JYAGUCHI client send request to the lookup server, this server provides the proxy required to the client and with the help of this proxy, client will be able to interact and can download the remaining codes from web server. In this way, a solution is achieved. In order to scale out the co-operative infrastructure, the underlying hardware federation can be increased by virtualization. We are also exploring the ways to build the services by using encapsulation service modularization approach. We have developed complete package of middleware by integrating different kinds of underlying technologies.

We agree that JYAGUCHI platform supports the development of distributed object. The main requirement of distributed object is its ability to create, invoke and deliver the objects in a remote host while providing the environment as if they were invoked in local infrastructure. These sorts of remote object invocation have been implemented in COM, COBRA and RMI and many other technologies until few years ago. JYAGUCHI employs a similar kind of concept that invokes the total bundle of service executed in remote server. We named this invocation model as RSI. The underlying protocol to call the remote service is JERI and JRMP [10], [11]. Recently, a different type of invocation model is often utilized in web service technology such as WSIF [12]. Web service technologies has passed different stages of evolution phases in terms of utilizing underlying message passing protocols such as SOAP and REST.

These technologies, must of the time, utilize XML data format to

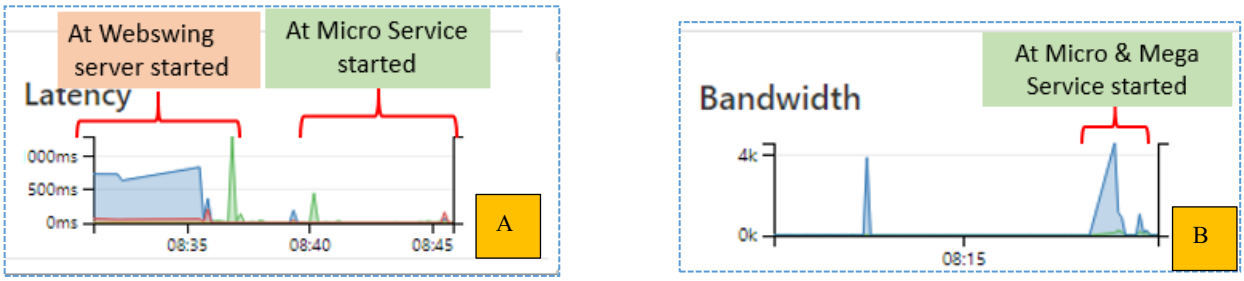

Figure 3. Latency and Bandwidth Report of JYAGUCHI services at Cloud environment

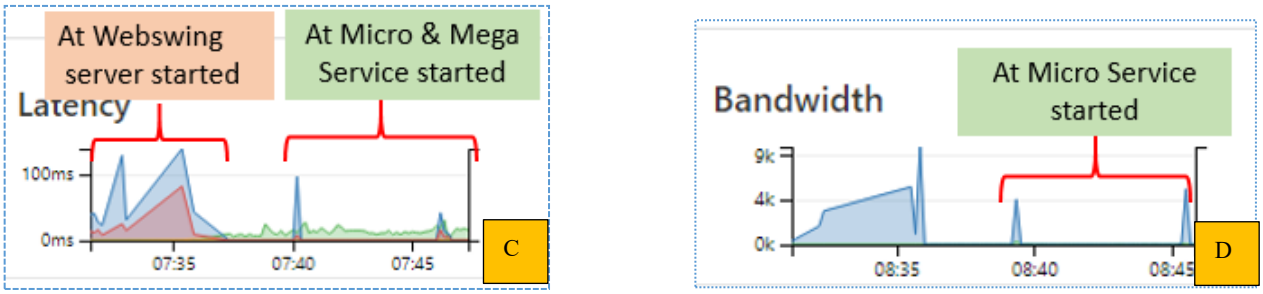

Figure 4. Latency and Bandwidth Report of JYAGUCHI services in Edge environment

send and receive the message. Messages can also be passed in JSON format too. However service call in web service technology and JYAGUCHI is different. Most of the web service related technologies utilizes message passing technique whereas JYAGUCHI utilizes RSI at which parameters are passed as the reference of Java object. We did not utilize message passing rather we utilized service calling approach to reduce the overhead occurs in message passing. In message passing, it must copy the existing arguments and append it to the new portion of the message resulting to a large size of message.

JYAGUCHI emphasizes loose coupling of the components thereby reducing the dependencies of the components participated in the foundation for the architecture. The overall architectural style presented by JYAGUCHI never tries to replace the prevalent architecture but try to leverage and show the guideline for the next generation applications by utilizing hybrid architecture style [2]. The post notification and dynamic deliver of the service omit the requirement of complex procedure of software installation for the client.

Particularly, these services are coded as java objects and are wrapped with JYAGUCHI service and the whole service is remarshaled in the user device. In the following sections, we describe the total scenarios of JYAGUCHI infrastructure, service wrapping scenarios and the concept of service granularity. In our new approach in the Figure 1, we have implemented the new platform for secure deliver web server which simply run JYAGUCHI application and is able to be delivered and accessible via Internet, by just entering the correct URL address. We also ensure stable performance and regular delivery of JYAGUCHI application via internet. As the web interface became the primary platform for software distribution in cloud. The delivery of JYAGUCHI services has become platform and device independent.

A. Service-client Federation

JYAGUCHI platform provides services in the manner of service client federations [1]. In order to consume the service, service provider must publish his service in a network. In order to consume the service, user needs to pass the authentication process. After successful authentication, he or she can utilize the service by simply clicking the icons displayed in the browser. In order to complete this process, service and client have an interaction for number of times.

5. Implementation

The concept of modularization is to separate the concern and context from once piece of program to other so as to minimize the effect that changes in one module may have on other modules. Separating the unrelated concerns from the modules has a great advantage of reducing interdependencies of modules so as to minimize the coupling between the server and client program. While there is a maximum coupling between server and client program, a small change in server program needs to be informed to the user. While this sort of software cohesion and coupling issues can be addressed in service modularization [14], [15].

5.1 Mini Service

Mini services in JYAGUCHI are the service which can be downloaded over a network and these services can be exported as a complete software package. To utilize this service, JYAGUCHI client even does not require knowing about the interface. However JYAGUCHI client must possess universal browser, where we have implemented universal interface that can be used to call any JYAGUCHI mini service. These services are implemented for the users who have low internet bandwidth.

5.2 Macro Service

Macro services are the services which can either be downloaded or

can be accessed via web browser. As the size of service becomes larger than micro service, we categorized these kinds of services to macro service and put option to the client to choose whether he or she wants to download entire service or can accessed by using web browser as like other web services.

5.3 Mega Service

Mega services in JYAGUCHI are those services which granularity is extremely larger than macro services and are not feasible to serialize as a complete software that can be done for micro and mega services. Option is also omitted and client can only use this kind of service as like web services by using web browser. In order to ensure the access of service at different network mode as in Figure 1 we have proposed the solution as follow:

A. JYAGUCHI Service at public network (Within Cloud)

Using the web browser, users can access the JYAGUCHI client system. The network area with the public or outside the company will allow the users to access only Micro service. The Mega & Macro services are disabled and users are not allow to access them.

B. JYAGUCHI Service at private network (Within Edge)

Using the web browser, users can access the JYAGUCHI client system. In the following subsections, we describe our guideline indicator about the notion of services and we categorized the services into 3 different sub-groups. Post notification implies that services deployed in the registry are notified to the end-user without human interference. Thus, end-users do not need to worry about new installation and update of the software. Service providers can update the services in the server without interfering the usage of the service by the client so that client can utilize the updated services immediately [2].

6. Security in JYAGUCHI

The design of the security model for JYAGUCHI is built on the twin notions of a principal and an access control list. JYAGUCHI services are accessed which are generally traces back to a particular user of the system. Services themselves may request access to other services based on the identity of the object that implements the service. Whether access to a service is allowed depends on the contents of an access control list that is associated with the object [13].

The solution provided in cloud infrastructures raise a security concern that data stored in the cloud might have same privacy. And the due to process as does data stored in your own infrastructure. There is no guarantee of data and privacy protection in the third party data center [2]. In JYAGUCHI, data does not retain in the server side rather they retain in the client side so that end-user can have control over their data. This solution in fact increase the trust over the system as no data are transferred to the server side except the data related to user authentication. In the client side, JYAGUCHI once the user success to authenticate login, client will be able to send a request to search the service registry in the network. To fine the registry, we have tested both unicast and multicast discovery [2].

Meanwhile, in the Enhanced JYAGUCHI Architecture the original concept of user authentication for the Macro and Mega services are same in Edge environment. In addition, to ensure the Micro services are secure in Cloud environment there is a user authentication before they access to services.

In order to maintain high security for the services when user access from in Cloud environment, the information for the authentication are privately provided by System Administrator after their request. Moreover, even when user access from Cloud environment the user's data does not retain in the server side rather they retain in the client side so that end-user can have control over their data. This solution in fact increase the trust over the system as no data are transferred to the server side except the data related to user authentication both for Edge and Cloud environment.

7. Performance Evaluation

To perform performance evaluation, we began to implement JYAGUCHI applications as services in two different environment i.e. Edge & Cloud environment and tested whether we can utilize the deployment environment without having difficulties. We did not have much difficulty to implement JYAGUCHI services and expose them via look up service and browser for end users.

A. Latency Evaluation

As the latency of a network is the time it takes for a data packet to be transferred from its source to the destination. In the Figure 3 'A', we can observed that after server started, delays in transmission are small, it's referred to as a low-latency network. This is the latency for download the Micro service.

On the other hand, in the Figure 4 'A' we can observed larger delay when application is started to run Mega JYAGUCHI service. This high latency is due to Mega service download from JIN server to user's computer. But after download is completed to local PC of user's the latency gradually decrease.

B. Bandwidth Evaluation

The JYAGUCHI services access at the socket or stream level. Below this level, the data is handled on the network, using the appropriate protocol. Depending on the Internet speed of user the bandwidth may be varies. The Internet download and upload speed was 38.36 Mbps and 43.14 Mbps respectively during the evaluation process in our working environment. We can observe in the Figure 3 'B' as bandwidth Report of JYAGUCHI services at Cloud environment that with the minimum internet speed the Micro service is accessible to the user's computer via Internet.

In addition, in the Figure 4 'B', to access the Mega service via Internet there is a little higher peak of bandwidth. This is because the application download in the user's computer via Internet.

8. Conclusion and Future Works

In this work, we have successfully presented a service-oriented development approach in JYAGUCHI platform which allows the user to develop, publish and utilize the services as per the requirement and security constraint. For example, if the communication infrastructure of the client is not reliable, users are

recommended to utilize the edge computing platform provided by JYAGUCHI. In this case, mostly the mega and macro services are developed and delivered to the end users within the edge environment. In this research, we enhanced JYAGUCHI platform by proposing a new architectural component that augments the services as per the granularity of the applications and also ensure the access control for secured service delivery. We evaluated our system by testing the performance of each component in terms of resource consumption, latency and network bandwidth metrics within and outside the edge. In our previous architecture, distributed services were not able to pass via internet due to the restriction in NAT. However, this constrain is solved by designing a NAT traversal module in the architecture. In the current architecture, this module is integrated with WebSwing and the services can be accessed by using web browser.

Reference

[1] G. Gautam B.P, Wasaki K, Batajoo A.:Encapsulation of Micro Engineering Tools in a Co- Operative Jyaguchi Computing Infrastructure DOI:10.15613/sijrs/2014/v1i1/53851 , Vol 1, No 1 (2014), Pagination: 34-41

[2] Gautam, B. P. (2009). An Architectural Model for Legacy Resource Management in a Jini Based Service Cloud over Secured Environment. SIG Technical Reports, 53(EIP-43), 55–62.

[3] Bishnu Prasad Gautam, Shree Krishna Shrestha and Dambar Raj Paudel. Thesis Report (2009), "Utilization of Jyaguchi Architecture for development of Jini Based Service Cloud."

[4] Gautam B. P., "An architectural model for time based resource utilization and optimized, [Master Thesis], Shinshu University, 2009

[5] Shrestha, S. K., Kudo, Y., Gautam, B. P., & Shrestha, D. (2013). Multidimensional Service Weight Sequence Mining based on Cloud Service Utilization in Jyaguchi, I.

[6] Gautam, B. P., & Shrestha, D. (2010). A Model for the Development of Universal Browser for Proper Utilization of Computer Resources Available in Service Cloud over Secured Environment. Lecture Notes in Engineering and Computer Science, 2180, 638–643. Retrieved from http://www.doaj.org/doaj?func=abstract&id=550261

[7] B. P. Gautam, H. Asami, A. Batajoo and T. Fujisaki, "Regional Revival through IoT Enabled Smart Tourism Process Framework (STPF): A Proposal," 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS) and 17th International Symposium on Advanced Intelligent Systems (ISIS), 2016, pp. 743-748, doi: 10.1109/SCIS-ISIS.2016.0162.

[8] B. P. Gautam, K. Wasaki, A. Batajoo, S. Shrestha and S. Kazuhiko, "Multi-master Replication of Enhanced Learning Assistant System in IoT Cluster," 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), 2016, pp. 1006-1012, doi: 10.1109/AINA.2016.110.

[9] Gautam Bishnu Prasad, Batajoo Amit, Wasaki Katsumi "Fogging Jyaguchi Services in Tensai Gothalo". International Journal of Computer Trends and Technology (IJCTT) V28(3):119-125, October 2015. ISSN:2231-2803. www.ijcttjournal.org. Published by Seventh Sense Research Group.

[10] M. N. Manas, C. K. Nagalakshmi, and G. Shobha, ―Cloud Computing Security Issues And Methods to Overcome,‖ Int. J. Adv.

[11] Res. Comput. Commun. Eng., vol. 3, no. 4, pp. 6306–6310, 2014.

[12] H. M. H. El-hoby, M. A. F. Salah, P. Mohd, and A. Suhaimi, Aligning Cloud Computing Security with Business Strategy,‖ Int. J. Comput. Technol., vol. 7, no. 1, pp. 52–60, 2014.

[13] Newmarch J., "Foundations of Jini 2 programming".Available: http://jan.newmarch.name/java/jini/tutorial/Jeri.html#Jeri

[14] Apache River Project. Available: http://river.apache.org/doc/specs/html/lookup-spec.html

[15] Gautam B. P., Sharma N., and Wasaki K., "Using a solar powered robotic vehicle to monitor and manage unstable networks", ICFN 2014

[16] Terlouw L., "Modularization and specification of service oriented systems", Ph.D Thesis, 2011.

[17] Brax S. A., and Toivonen M., "Modularization in businessservice innovations", Available: www.imi.tkk.fi/publications/ download/207/

[18] Gautam B. P., Paudel D. R., and Shrestha K., "A study and site survey in himalayan region for proper utilization of wireless community networks: an assessment of community wireless implementation in heterogeneous topography", WAKHOK Journal, vol. 11 Japan Disaster Statistics, Natural Disasters from 1980–2010, International Disaster Database. Available: http://www.preventionweb.net/english/countries/ statistics/?cid=87

[19] Chen M. X., Sung F., and Lin B. Y., "Service discover protocol for network mobility environment", ICIC International Journal 2012, Available: http://www.ijicic.org/ijicic-11-05025.pdf

[20] Perianu R. M., Hartel P., and Scholten H., "A classification of service discovery protocols". Available: http://doc.utwente.nl/54527/1/classification_of_service.pdf

[21] JiniTM Discovery & Join pecification"https://river.apache.org/release-doc/current/specs/html/discovery-spec.html

Acknowledgments: This work was supported in part by the Japan Society for the Promotion of Science (JSPS) under Grant 19H04101 and Grant 18K11273, the Cooperative Research Project Program of the Research Institute of Electrical Communication, Tohoku University, Japan