An Architecture of Thin Client in Internet of Things and Efficient Resource Allocation in Cloud for Data Distribution

Aymen Abdullah, Phuoc Phamhung, Eui Namhuh

An Architecture of Thin Client in Internet of Things and Efficient Resource Allocation in Cloud for Data Distribution

Aymen Abdullah A, Phuoc PhamHung, and Eui NamHuh
Department of Computer Science and Engineering, Kyung Hee University, South Korea

Abstract: *These days, Thin-client devices are continuously accessing the Internet to perform/receive diversity of services in the cloud. However these devices might either has lack in their capacity (e.g., processing, CPU, memory, storage, battery, resource allocation, etc) or in their network resources which is not sufficient to meet users satisfaction in using Thin-client services. Furthermore, transferring big size of Big Data over the network to centralized server might burden the network, cause poor quality of services, cause long respond delay, and inefficient use of network resources. To solve this issue, Thin-client devices such as smart mobile device should be connected to edge computing which is a localized near to user location and more powerful to perform computing or network resources. In this paper, we introduce a new method that constructs its architecture on Thin-client -edge computing collaboration. Furthermore, present our new strategy for optimizing big data distribution in cloud computing. Moreover, we propose algorithm to allocate resources to meet Service Level Agreement (SLA) and Quality of Service (QoS) requirements. Our simulation result shows that our proposed approach can improve resource allocation efficiently and shows better performance than other existing methods.*

Keywords: *Cloud computing, data distribution, edge computing, resource allocation, and thin client.*

Received January 19, 2015; accepted August 12, 2015

1. Introduction

Internet of Things (IoT) is a technology that enables many objects (e.g., smart mobile devices, tablets, home appliances, etc) which also known as Thin-client to connect to Internet to perform diversity of computing services (e.g., processing, memory, storage, virtualization, etc) as well as others (e.g., receive/send data, surf internet, access social websites, etc). As a result, mobile services are presence in almost every aspect of our daily life (e.g., education, health care, commerce, etc). In spite of mobile computing astonishing convenience and flexibility it offers, still it has deficiency in ability to perform heavy computing tasks/fast high data transmission due to restriction in mobile devices resources (memory, processing, battery life, CPU, storage, etc) as well as restriction in network bandwidth when we consider variety of devices. To overcome this issue, we use Mobile Cloud Computing (MCC) [10] and edge computing [15]. MCC leverages on the cloud technique for storage and process on mobile devices or collaborate with edge computing to acquire sufficient resources. edge computing can also be considered as MCC where it perform the same services as MCC. edge computing is localized which moves data and computation closer to user location where MCC is centralized. edge computing is an important method for delivering web data over the internet [15].

One of the ways to alleviate this issue is by using MCC [10], which leverages on cloud computing technique for storage and processing of data on mobile device, or collaborating with external devices to get more resources. This can be released with minimal management effort or service provider interaction. Connecting massive number of smart devices to MCC to perform computing might burden the network and the MCC as well. Therefore, edge computing is efficient solution where it provides better resource management, quick delivery of data, and fast access. In another word, we are moving all the service in MCC to be performed in edge computing based on the requested service/size of data that is need to be sent in order to be processed.

There is some research developed to minimize the shortcoming of MCC. In [6], the author introduces guidelines to create framework of virtual MCC provider. The framework advantages is being nearby thin-client to develop on-the-fly connection which avoid the need to connect to infrastructure based cloud. In spite of that, it has restriction in thin-client capacity and low bandwidth between thin-client and cloud because of the long distance. edge computing has higher capacities and fast strong connections with much higher bandwidth. Sufficient bandwidth is very critical issue where the higher bandwidth we have the higher quality of services is received [17].

Therefore, in this paper we introduce a new architecture that collaborate thin-client and edge computing which enhances its capacities. Furthermore, we introduce our new strategy for data distribution optimization such as big data. Moreover, we introduce an algorithm to perform resource allocation in order to satisfy Service Level Agreement (SLA) and Quality of Service (QoS). We also introduce new communication protocols between components on our architecture. Our simulation show that our approach can improves the efficiency of resource allocation and shows a better performance comparing with others.

The rest of the paper is organized as follows. In section 2, we introduce related work. In section 3, we present overview of edge computing. In section 4, we present our motivation scenario. In section, 5, we introduce our system architecture. In section, 6 we present our proposed communication protocol. In section 7, we present our implementation and analysis result. Finally, in section 8 we present our conclusions and future work.

2. Related Work

There are many searches attempting to resolve previously mentioned issues. In [19], the author proposed efficient cloud based synchronization for number of hierarchy distributed number of file system. They utilize the concept of master-slave architecture in order to propagate data to reduce traffics. Delgado *et al.* [2], is presenting resource scheduling methods which can be efficient in mitigating the impacts that can influence application time of respond and utilization of the system. Fan *et al.* [3] and Kwok [11] is present the impact of data transmission delay on the performance. In [9] the author introduce one way to make a parallel processing to big data which will increase the performance in federated cloud computing. In spite of that, these researches do not state how much resources should be used.

There are also many researches done dealing with resource allocation. In [7] illustrate that shared allocation is superior to dedicated allocation. In spite of that, the author does not perform experiment with an arbitrary number of SLA and does not show how fast the server needs to be to guarantee QoS. In [13,14] the authors provide services to huge number of SLA even though it is difficult to obtain performance between shared allocation and reserved allocation. In [12] the author present model for securing resource allocation in cloud computing where it design fuzzy-logic based trust and reputation model.

Many researches have been done to provide better way for the integration of mobile devices and cloud computing. In [20] the author introduces an idea utilizing cloud to improve the capability of mobile devices. In [16] the author makes changes to Hyrax which enables mobile devices to use cloud computing platforms. The idea of utilizing mobile device as a provider of resources is introduce. However, the experiment is not integrated.

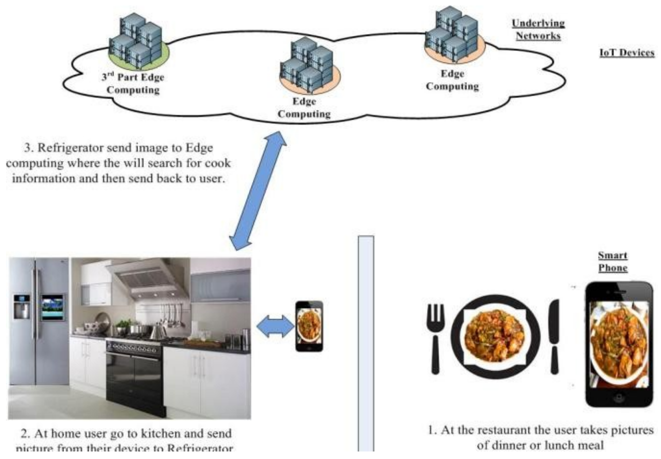
In [4], the authors just concentrate on using partition policies to hold the effect of application on mobile devices, but do not solve any other matter related to MCC. To the best of our knowledge, there are not so many researches considering collaboration of thin-client and edge computing to provide better way of managing data distribution and resource allocation in edge computing instead of MCC as well as creating protocol to show how these entities can communicate with each other.

3. Overview of Edge Computing

Edge computing was design to be located at the edge of network to provide scalability and availability of web services. It allocates the logic of application and the underlying data to network edges [8]. Some of edge computing advantages are 1) reduce down network latency, faster respond to end user, better user of resource, reduce the cost of scalability, and fast data delivery [8]. Edge computing consider as an extension of content delivery network as well as MCC because it offers all of mobile cloud capabilities. Edge computing can be helpful with applications that run database where it can distribute the section of database to edge servers for farther processing [8]. Therefore having edge server located closer to user location provides significant advantages.

4. Motivating Scenario

Figure 1 illustrates our scenario which reflects the benefits of the IoT-edge computing collaboration.



Underlying Networks
IoT Devices
3rd Part Edge Computing
Edge Computing
Edge Computing
3. Refrigerator send image to Edge computing where the will search for cook information and then send back to user.
Smart Phone
2. At home user go to kitchen and send picture from their device to Refrigerator
1. At the restaurant the user takes pictures of dinner or lunch meal

Figure 1. Motivating scenario architecture.

Our scenario start when user takes some pictures of food which they are eating and later on they want to cooking at home. The user decided to cook the same food in the picture at home. However, it is inconvenient and not safe to hold the smart phone in their hand while they are cook. Some of today home appliances such as refrigerator carry big screen and capable to connect to internet. The user sends the food picture to refrigerator to obtain food ingredient and cooking instructions. User can look at the screen or listen to the cooking instruction which is read by refrigerator system.

Unfortunately, the direct Internet connection of the refrigerator only has a restricted bandwidth and capacity to perform searches for food which might generate thousands of search result and required long respond time. Instead the refrigerator can connect to edge computer then requests the edge computer to access the internet to look for the information. After receiving information, the result will be returns to the refrigerator. Finally, the user can see or hear the cooking instructions.

Our scenario in using thin-client and edge computing introduce the potential benefit of their collaboration in cloud computing environment which increases the opportunity of using and managing resource efficiently. In spite of that, the issues here are:

1. How to optimize data distribution?
2. How to increase/better managing resources efficiently?
3. How to allocate sufficient resources to satisfy a diversity of SLA.

5. System Architecture

Our system architecture consists of three Layers which are illustrated in Figure 2.

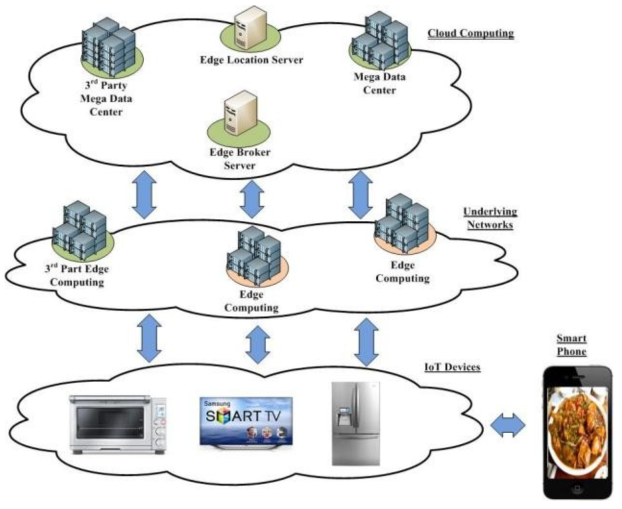


Figure 2. Our proposed system architecture.

The lowest layer consists of user IoT devices such as refrigerator, smart TV, smart oven, smart phone, etc., which is capable to connect to each other through WiFi, 3G and LAN. The middle layer is the underlying network which consists of edge computing and 3rd party edge computing. We need 3rd party edge computing because 1) sometimes some requested services might not be offered by home edge computing and 2) due to the popularity of edge computing, countless number of IoT smart devices might be connected to edge computing requesting services which might be too much for it to handle, so some of the requested services can be redirected to 3rd party edge computing. In this case we can guarantee QoS. The upper layer is cloud computing environment and it consist of mega data center, edge location server, edge broker server which is purpose to receive new service requests and 3rd part mega data center. Most of the work will be accomplished by lower and middle layer.

Most of previously introduces approaches uses 1/m/1 model to resolve the above mentioned problem. However, our proposed utilize 1/m/m/1 model for resolving the problem. When the data is send to edge computing, it will be divided into multiple blocks. These blocks will be assigned to certain Virtual Machines (VMs) where each block is divided into multiple chunks which transferred to multiple processors for processing. After receiving the processed data, the processors join them into one data and send them to user IoT devices. In this case we do not burden the system to process big size data, ensure the availability of the server to process other request when they exists, and guarantee fast respond to ensure QoS.

The overall process is divided into two phases. Phase 1 will involve 1) determine VMs needed minimum number and the speed of that VMs, and 2) sorting, dividing and assigning data to VMs based on VMs current capacity. Phase 2 will involve 1) distribute data that has different capacities to processors, and 2) merging data and send to IoT devices. Table 1 describe our system component and their role.

Table 1. System component and their role.

Component	Role
IoT Smart Devices	Responsible of connecting to Internet through the network (WiFi, 3G, LTE, etc).
Edge Computing	Responsible of receiving user requests and providing service such as processing,storage, bandwidth, etc.
3rd Party Edge Computing	Responsible of providing other services and processing received service by other edge Compuitng server.
Mega Data Center	Responsible of providing services in the cloud.
3rd Party Mega Data Center	Responsible of providing other services which is not provided in the mega data center.
Edge Location Server	Responsible of which stores addresses edge computing server for fast requests respond and is used by edge computing to locate other nearby edge computing to request previous offered services.
Edge Broker Server	Responsible of receiving new services requested by the IoT device users.

5.1. Phase 1 of Our Proposed Method

5.1.1. Determine VMs Need Minimum Number

The purpose of algorithm 1 is to determine the minimum number of VMs depending on SLA. We utilize Cumulative Distribution Function (CDF) *F(x)* time respond which is available in [1]. Until the *F(x)* reaches the targeted probability, the minimum numberof VMs *m* keeps increases. Finally, we received the required *m* for SLA. Below is the description of F(x):

F(x) =Probability (time of response < x) =

$$\begin{cases} 1-e^{-\mu x}-k\mu e^{-\mu x}x & \text{for } \sigma=m_i-1 \\ 1-e^{-\mu x}-k\mu e^{-\mu x(m_i-\sigma)}\left[\frac{1-e^{-\mu x(1-m_i+\sigma)}}{1-m_i+\sigma}\right] & \text{for } \sigma\neq m_i-1 \end{cases} \quad (1)$$

Where $\sigma = \lambda/\mu$

$$k=p(0)\frac{\sigma^{m_i}-\mu}{m_i!}*\frac{m_i}{(m_i-\sigma)}$$

$$P(0)=\left(\sum_{n=0}^{m-1}\frac{\sigma^n}{n!}+\frac{mp^m}{m!(m-\sigma)}\right)^{-1} \quad (2)$$

λ is the arrival rate and μ is the service rate.

```
Algorithm 1: Determining the No. of VMs
Input:
1. λ                // rate of arrival
2. μ                // rate of service
3. SLA(x,z)     // x:time of response
// z: probability target
Output: m       // required minimum no. of VMs
4. Float σ=λ/μ
5. Function determineMinVM(σ,μ,x,z) {
6. If ( σ -- (int) σ) m-(int) σ;
7. Else m= (int)Math.floor (σ) + 1;
8. While F(x) <= z, m++;
9. Return m;// required minimum no. of VMs}
```

Usually, edge computing infrastructure may provide diversity of services to satisfy a large number of SLAs by utilizing First Comes First Server (FCFS) scheduling methods which is illustrated in Figure 3. Therefore, we recommend allocating the VMs into two groups where the first group will be used for Shared Allocation (SA) $m_{\text{sharedAllocation}}$ and the second group will be used for Reserved Allocation (RA) $m_{\text{reservedAllocation}}$.

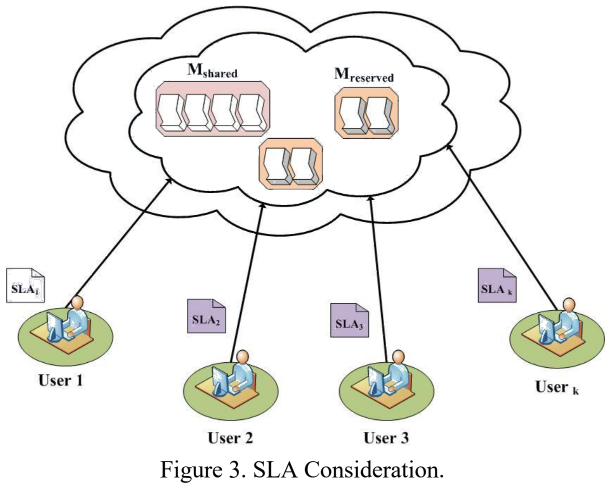


Figure 3. SLA Consideration.

For shared allocation, the arrival jobs of SLA are combined into a single steamed and served by $_m$ VMs. As for reserved allocation, we provide one VMs for each arriving job which illustrated in Figure 4.

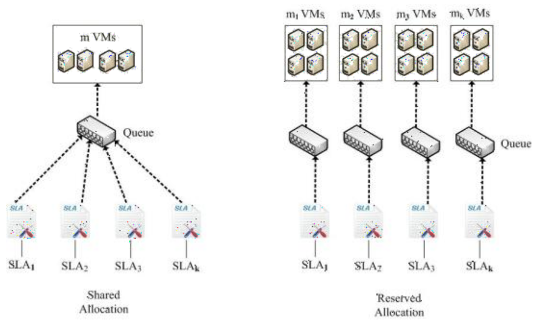


Figure 4. Our proposed strategy of resource allocation.

In shared Allocation, all of SLAs will have the same CDF of response time and arrival rate $\lambda=\sum_{i=1}^{k}\lambda_i$. As a result, the minimum number of VMs $m_{\text{SharedAllocation}}$ to meet k SLAs is given by:

$$m_{SharedAllocation}=\max(m_1,\ldots m_i,\ldots m_k) \quad (3)$$

m_i refer to the number of VMs required to satisfy SLA_i of $user_i$. Let $m_{sharedAllocation}$ become the smallest number of VMs which is required to meet k SLAs in Reserved Allocation. So $m_{ReservedAllocation}$ is given by:

$$m_{ReservedAllocation}=\sum_{i=1}^{k}m_i \quad (4)$$

As a result, when more than one requesters have the same SLAs, Shared Allocation will provide same or better performance than Reserved Allocation ($m_{sharedAllocation}<=m_{ReservedAllocation}$). But, if SLA_1, SLA_2 are different for Shared Allocation and Reserved Allocation, then it is difficult to determine which one is better than the other.Table 2 shows an example of both shared and reserved allocation.

In the first case, $m_{ReservedAllocation}$ is better than $m_{sharedAllocation}$ even though the reverse case is true in the Shard allocation.

Table 2. Proposed cases example.

Case	λ_1	x_1,y_1	λ_2	x_2,y_2	$m_{Reserved}$	m_{Shared}
1	3.9	3,0.7	3	10	10	11
2	3.9	3,0.85	2.9	12	12	10

In order to satisfy SLA_1,SLA_2 , we are trying to discover the best favourite strategy regarding shared allocation of reserved allocation. Furthermore, the VMs can ensure the QoS as well. Let $E(SLA)$ refer to the average number of VMs which is required to meet the given SLA over the considered arrival rate.

$$E(SLA)=\frac{1}{k}\sum_{0}^{k}f(k,x,y) \quad (5)$$

Let D be the SLA difference between both SLA_1 and SLA_2. D is given by:

$$D=\left|E(SLA_1)-E(SLA_2)\right| \quad (6)$$

In Algorithm 2, we state the allocation strategy to satisfy SLAs and QoS. In Table 3, we show the relationship between D and angle α. Every D is fixed by the change in arrival time λ_1,λ_2 in (0,30) and average angle of SLA difference for every range. We state angle α by the following formula which is presented in Figure 5:

$$\sin\alpha = \lambda_2 / sqrt(\lambda_1 * \lambda_1 + \lambda_2 * \lambda_2) \tag{7}$$

Table 3. Service Level Agreement Difference (SLA).

D	α
(0,20)	0
(20,40)	20
{40,66}	50
(66,88)	70

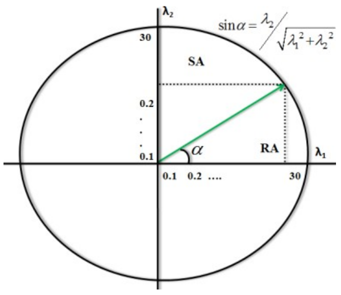


Figure 5. Allocation strategy.

Here we need to discover the speed of VM to guarantee QoS. In addition we are applying little law [18] which we describer below;

$$E[N] = \frac{p}{(1-p)} \quad \text{where} \; p = \frac{\lambda}{\mu} \tag{8}$$

$E[N]$ denotes the no. of jobs in the system. As a result, the processing time expectation is as follow:

$$E[T] = \frac{E[N]}{\lambda} = \frac{p}{\lambda(1-p)} = \frac{1}{\mu(1-p)} = \frac{1}{\mu-\lambda} \tag{9}$$

We set the bellow formula to satisfy QoS:

$$\mu >= \frac{1}{E[T]} + \lambda \tag{10}$$

Based on this formula , we can discover the VMs rate service. We present the bellow example to set it clear. Let's say for example we want E[T] <= 10 second, λ=1 job/sec, then the needed VM rate is as fellow:

$$\mu >= \frac{1}{10} + 1 = \frac{11}{10} \tag{11}$$

Algorithm 2: Determining the allocation strategy

```
Input:
1. λ1, λ2// rate of arrival
2. μ            // rate of service
3. SLA1, SLA2
4. E            // processing time expectation
Output:
5. SA, RA //shared and reserved allocation strategy
6. Function determineAllocStrategy (λ1,λ2,SLA1,
     SLA2,E,μ){
7. Calculate SLA difference D
8. Get the corresponding angle α from the SLA difference table
9. If (μ>= (1/E[T] +λ1) &&μ>=(1/E[T]+λ2))
10. If (Math.asin(λ2/sqrt(λ1*λ1 + λ2*λ2)) <=α)
11. Return RA // reserved allocation
12, Else
13. Return SA // sharedallocation
14. Else
15. Return false {
```

5.1.2. VMs Capacity

In this section we will sort, divide and assign data to VMs current capacity. In order to set data priority we utilize training data to sort out the data. The data with high priority will be transferred first and the data with low priority will be transferred last. The data can be in blocks {bl1, bl2,…,bln} which has different sizes. Then we uses Greedy algorithm to select the best VMs based on their capacities. Finally, we assign VMs with higher capacity to the block with big size. Figure 6 illustrate the proposed methods of assigning data to VMs.

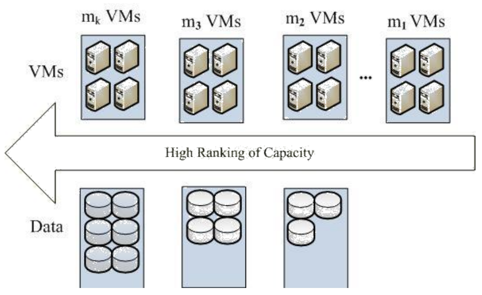


Figure 6. The assignment of data to VMs.

5.2. Phase 2 of Our Proposed Method

5.2.1. Distribute Data Block

In this section we distribute data block that has different capacities to processors. We start by dividing data to block where the blocks also will be divided to small size called chunks{chk_1, chk_2, …, chk_n} which has different size depending on the bandwidth strength. chk_i denote to chunk in each block. $w(ch_i)$ denote to the size of chunk. bw_i denote the bandwidth between VMs and processor. $w(ch_i)/b_i$ represent the time it takes to send chunk from VMs to processor. When we consider parallelization, then the time it takes to send chunks of data to processors should even.

$$\frac{w(chk_1)}{bw_1} = \frac{w(chk_2)}{bw_2} = \frac{w(chk_3)}{bw_3} = \dots \frac{w(chk_i)}{bw_i} = t \tag{12}$$

$$Set\ S = w(block) = \sum_{i=0}^{n} w(chk) = t \sum_{i=0}^{n} b_i \tag{13}$$

Therefore:

$$w(chk_i) = t * b_i = \frac{S}{\sum_{i=0}^{n} b_i} * b_i \tag{14}$$

Based in the above stated value, we are able to determine the size of every chunk to adapt it with the bandwidth. Then we sort out the processor based on their capacities. The bigger the chunk of data will be sent to processor with higher capacity to process it.

5.2.2. Merging Data

In this section we try to merge data and then send it to IoT devices. The use of peer-to-peer synchronization might generate complexity between processors. As a result, we make edge computing to act as master which will receive chunk of data from other processors to reduce the complexity which result from firewall between processors. Figure 7 illustrate four processor example as well as master-slave and all-to-all communication methods.

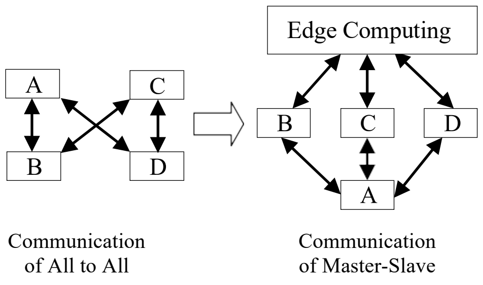


Figure 7. Communication strategy architecture.

6. Propose Our Communication Protocols

In this section we present our develop communication protocol. The communication protocol takes place between;

- Smart IoT devices and edge computing.
- Edge computing and other edge computing in inter/intra network area.
- Edge computing and 3rd party edge computing.

Due to the significant advantages of edge computing, most of the IoT devices requested service will be redirected to edge computing instead of cloud computing for the fact of being localized. This might lead to overhead, low performance and poor quality of services. As a result, we create communication protocol between edge computing's as well as 3rd party edge computing which enablesthese components to smoothly communicate with each other. Some of the requested service might not be available in user home edge computing, therefore we can request from 3rd part edge computing which will guarantee quality of services. Figures 8, 9, and 10 illustrate a sequence flow diagram of proposed communication protocols.

Figure 11 illustrate the communication protocols between the above mentioned entities. We assign global address for each edge computing server which is generated by edge computing location server which will make it easy to discover/communication with other 3rd party edge computing as well as other edge computing. For the first time, edge computing need to connect to edge computing location server to discover surrounding other edge computing server. After that they just connect directly to them. We use the same approach in [5] to create the communication methods. This method will create tunnel between entities (MAG and LMA) in order to send data/ share data between each other. In our case we will use this method to create the communication of the entities mentioned above.

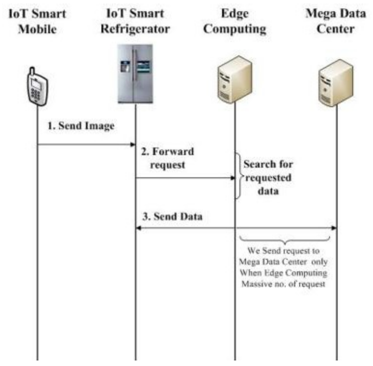


Figure 8. Sequence flow diagram between IoT device and edge computing.

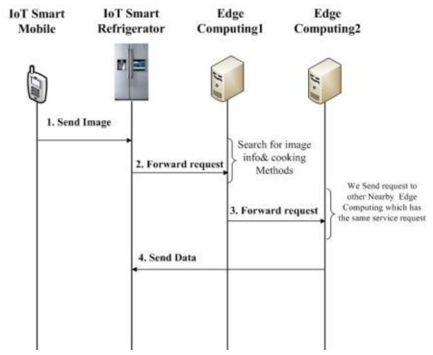


Figure 9. Sequence flow diagram between edge computing and other edge computing in inter/intra network area.

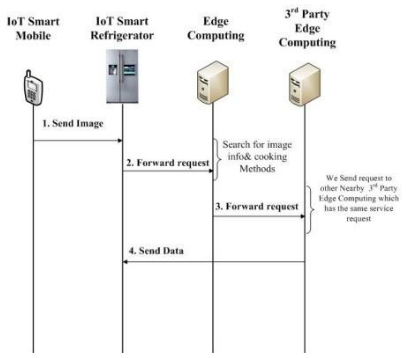
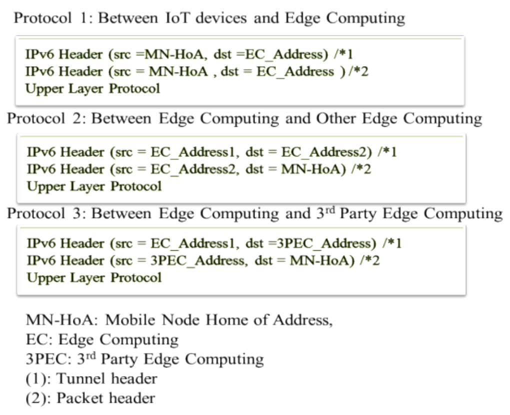


Figure 10. Sequence flow diagram between edge computing and 3rd party edge computing in inter/intra network area.



Protocol 1: Between IoT devices and Edge Computing

```
IPv6 Header (src =MN-HoA, dst =EC_Address) /*1
IPv6 Header (src = MN-HoA , dst = EC_Address ) /*2
Upper Layer Protocol
```

Protocol 2: Between Edge Computing and Other Edge Computing

```
IPv6 Header (src = EC_Address1, dst = EC_Address2) /*1
IPv6 Header (src = EC_Address2, dst = MN-HoA) /*2
Upper Layer Protocol
```

Protocol 3: Between Edge Computing and 3rd Party Edge Computing

```
IPv6 Header (src = EC_Address1, dst =3PEC_Address) /*1
IPv6 Header (src = 3PEC_Address, dst = MN-HoA) /*2
Upper Layer Protocol
```

MN-HoA: Mobile Node Home of Address,
EC: Edge Computing
3PEC: 3rd Party Edge Computing
(1): Tunnel header
(2): Packet header

Figure 11. Communication protocols.

7. Implementation and Analysis

In this section, we used numerical simulation to examine the efficiency of SA and RA. Furthermore, we compare the performance of our proposed approach with existing one. The parameter in our simulation consists of arrival rate (λ), response time (x), the targeted probability (y), and the proposed algorithms. We use Java (jdk-7u7-i586 and Netbeans-7.2) to generate our simulation. The result proves that the shared allocation and reserved allocation almost have the same impact when they have the same SLA with different arrival rate(λ), response time(x), and target probability (y). We also experimenting inthe same case but we used multiple SLA instead of single one.

Figure 12 illustrates shared allocation and reserved allocation with different response time. The result shows that the response time increases when the smallest number of VMs decreases. It also shows that when we set different respond time for shared and reserved allocation, the probability is almost the same for both of them.

Figure 13 illustrates SLA different target probability of shard allocation and reserved allocation. The result shows the minimum number of VMs which is needed to meet SLA satisfaction. For example, when the target probability to meet SLA is 0.2, then we need minimum of 5 VMs for shard and reserved allocation. Therefore we meet SLA different target probability for shared and reserved location.

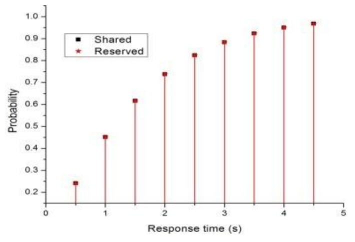


Figure 12. Different time response of shared and reserved allocation.

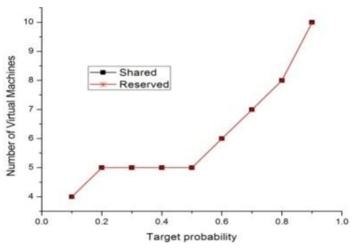


Figure 13. SLA different target probability of shared and reserved allocation.

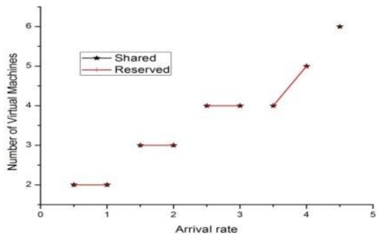


Figure 14. Different arrival rate of shared and reserved allocation.

Figure 14 illustrate Different arrival rate of Shard allocation and reserved allocation. The result shows the minimum number of VMs which is required to meet SLA which is equivalent to different arrival rate. For example, we need minimum number of 3 VMs when the arrival rate is 1.

When considering working with multiple SLAs, it is recommended that the strategy of shared allocation is more resource efficient than reserved allocation. Figure 15 illustrate the result of different SLAs of shard allocation and reserved allocation. The result shows that share allocation uses fewer VM than reserved allocation when the number of SLA increases.As a result, reserved allocation can provide guarantee rate due to the offering of resources.

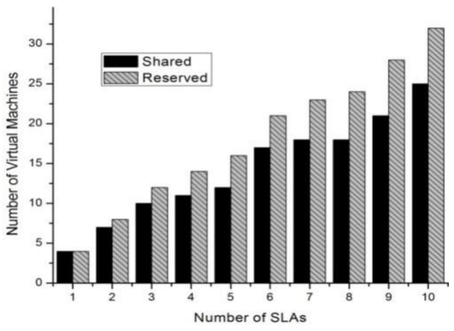


Figure 15. Different SLAs of shared and reserved allocation.

Furthermore, we compare the processing time of sending big size of data to destination for our proposed system with other approaches that uses only one single processor. Figure 16 illustrates a comparison of our proposed approach with other approaches that uses only one processor to process big file size. For example, the processing time for file size of 200Mb using our approach results in less processing time than other approaches that uses one processor. The result shows that our proposed approach results in a better performance than other approach (uses one processor).

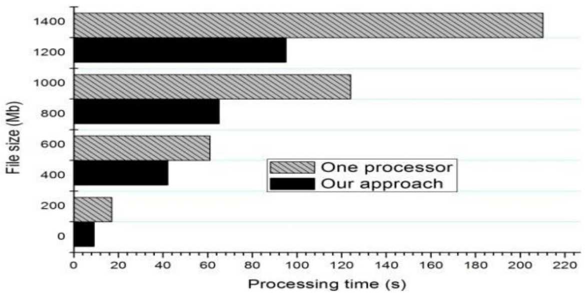


Figure 16. Comparison of our proposed approach with other approaches.

Figure 17 illustrate the result regarding the number of thin clients/edge computing with respect to thin clients' workload.We calculate the minimum number of thin client/edge computing which are able to satisfy thin client requirement with different workload. When the thin client work load increase, the number of edge computing increase in order to satisfy the requirement.

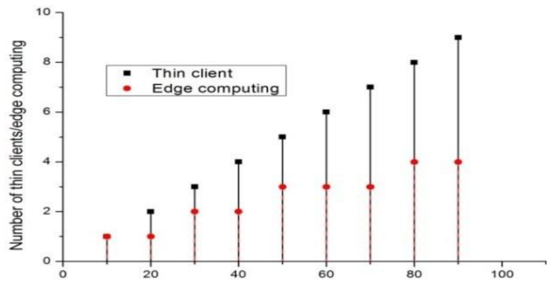


Figure 17. Thin clients workload.

8. Conclusions

In this paper, we have introduced a system architecture that utilizes the thin-client-edge computing collaboration to enhance thin client capacities. We introduce efficient strategy to optimize the data distribution in edge computing. In addition, we create algorithms to allocate resources to meet SLA and QoS. Furthermore, we propose a new communication protocol that allows entities in our system architecture to communication or share data. We simulated our proposed system to evaluate our method. Our proposed approach enhances resource allocation and shows better performance than other previous approaches.

Acknowledgments

This research was supported by the Ministry of Science, ICT and Future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2015-(H8501-15-1015) supervised by the Institute for Information and communications Technology Promotion (IITP)). The corresponding author is Eui-Nam Huh.

References

[1] Andreolini M., Casolari S., and Colajanni M., "Autonomic Request Management Algorithms for Geographically Distributed Internet-Based System," *in Proceeding of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, Venice, pp. 171-180, 20-24 , 2008.

[2] Delgado J., Sadjadi S., Fong L., Yanbin L., Bobroff N., and Seelam S., "Efficiency Assessment of Parallel Workloads on Virtualized Resources," *in Proceeding of 4th IEEE International Conference on Utility and Cloud Computing*, Melbourne, pp. 89-96, 2011.

[3] Fan P., Wang J., Zheng Z., and Lyu M., "Toward Optimal Deployment Of Communication-Intensive Cloud Applications," *in Proceeding of IEEE International Conference on Cloud Computing*, Washington, pp. 460-467, 2011.

[4] Giurgiu I., Riva O., Juric D., Krivulev I., and Alonso G., " Calling the Cloud: Enabling Mobile Phones as Interfaces to Cloud Applications," *in Proceeding of ACM/IFIP/USENIX 10th international conference on Middleware*, vol. 5896, Urbana, pp. 83-102. 2009.

[5] Gundavelli S., Leung K., Devarapalli V., Chowdhury K., and Patil B.," Proxy Mobile IPv6," Technical Report Network Working Group, 2008.

[6] Huerta-Canepa G. and Lee D., "A Virtual Cloud Computing Provider for Mobile Devices," *in Proceeding of 1st ACM Workshop on Mobile Cloud Computing and Services: Social network and Beyond*, San Francisco, pp. 1-24, 2010.

[7] Hu Y., Wong J., Iszlai G., and Litoiu M., "Resource Provisioning for Cloud Computing," *in Proceeding of the Conference of the center for advanced studies on Collaborative Research*, Ontario, pp. 101-111, 2009.

[8] Pang H. and Tan K., "Authentication Query Results in Edge Computing," *in Proceeding of 20th Conference on Data Engineering*, Washington, pp. 560-571, 2004.

[9] Jung G., Gnanasambandam N., and Mukherjee T., "Synchronous Parallel Processing of Big-Data Analytics Services to Optimize Performance in Federated Clouds," *in Proceeding of IEEE 5th International Conference on Cloud Computing*, Honolulu, pp. 811-818, 2012.

[10] Kumar K. and Yung-Hsian L., "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," *IEEE Computer*, vol. 43, no. 4, pp. 51-56, 2010.

[11] Tadapaneni, N. R. (2017). Different Types of Cloud Service Models. Available at SSRN 3614630.

[12] Chandran K., Shanmugasudaram V., and Subramani K., "Designing a Fuzzy-Logic Based Trust and Reputation Model for Resource Allocation in Cloud Computing," *The International Arab Journal of Information Technology*, vol. 13, no. 1, pp. 30-37, 2013.

[13] Li J., Chinneck J., Woodside M., and Litoiu M., "Fast Scalable Optimization to Configure Service Systems Having Cost and Quality of Service Constraints," *in Preceeding of the 6th International Conference on Autonomic Computing*, Barcelona, pp. 159-168, 2009.

[14] Lenk A., Klems M., Nimis J., Tai S., and Sandholm T., "What's inside the Cloud? An Architectural Map of the Cloud Landscape," *in Proceeding of ICSE Workshop on Software Engineering Challenges of Cloud Computing*, Washington, pp. 23-31, 2009.

[15] Tadapaneni, N. R. (2016). Overview and Opportunities of Edge Computing. *Social Science Research Network*.

[16] Marinelli E., "Hyrax: Cloud Computing on Mobile Devices using MapReduce," Master Thesis draft, 2009.

[17] Nguyen T., Nguyen M., and Huh E., "Service Image Placement for Thin Client in Mobile Cloud Computing," *in Proceeding of IEEE 5th International Conference on Cloud Computing*, Honolulu, pp. 416-422, 2012.

[18] Sheldon R., *Introduction to Probability Models*, Elsevier, 2010.

[19] Uppoor S., Flouris M., and Bilas A.,"Cloud-Based Synchronization of Distributed File System Hierarchies," *in Proceeding of International Conference on Cluster Computing Workshops and Poster*, pp.1-4, 2010.

[20] Luo X., "From Augmented Reality to Augmented Computing: A Look at Cloud-Mobile Convergence," *in Proceeding of International Symposium on Ubiquitous Virtual Reality*, pp. 29-32, 2009.

[21] Lin Y., Kemme B., Patino-Martinez M., and Jimenez-Peris R., "Enhancing Edge Computing with Database Replication," *in Proceeding of 26th IEEE Symposium on Reliable Distributed System*, Beijing, pp. 45-54, 2007.

[22] Lin Y., Kemme B., Patino-Martinez M., and Jimenez-Peris R., "Enhancing Edge Computing with Database Replication," *in Proceeding of 26th IEEE Symposium on Reliable Distributed System*, Beijing, pp. 45-54, 2007.

[23] Kwok M., "Performance Analysis of Distributed Virtual Environments," PhD Thesis University of Waterloo, Ontario, 2006.

[24] PanimalarS, A., Dharani, N., Aiswarya, R., & Shailesh, P. (2017). Cloud Data Security Using Elliptic Curve Cryptography.

[25] Pvandana, C., & Chikkamannur, A. (2016). Internet of Things future in Edge Computing.