Regular Paper

Evaluation of Autonomous Control of Server Relocation for Fog Computing Systems

Kouki Kamada[†], Hiroshi Inamura[‡], Yoshitaka Nakamura[‡]

[†]Graduate School of Systems Information Science, Future University Hakodate, Hakodate, Japan
[‡]School of Systems Information Science, Future University Hakodate, Hakodate, Japan

Abstract - Fog computing, which extends the paradigm of cloud computing to the edge of networking, has been proposed, and its research has been active. In the field of networking, research on Content Centric Networks (CCN) has been conducted. CCN have been shown to be able to handle cached content naturally within the network, reducing traffic and latency. However, in today's Internet, dynamic content with dynamic services is indispensable. A system that capable of handling dynamic services is desired by incorporating the way of handling computational resources in fog computing into CCN. In this paper, an autonomous control scheme of server relocation for fog computing systems is proposed. We study the optimizing quality of service by allowing services running on the network to be dynamically relocated. Our ns-3 simulations show the fairness between users achieved and the reduction of the average response time on non-uniform computer resources with three use cases.

Keywords: Contents Centric Network, Fog Computing, In-Network Caching, Server Relocation

1 INTRODUCTION

The number of IoT devices, which is 27.4 billion as of 2017, is expected to increase to about 40 billion by 2020[1]. For these large volumes of data generated by IoT devices, processing-intensive architectures such as cloud computing do not take advantage of the processing power of the edge and the latency from the point of data generation to the remote data centers cannot be ignored. Therefore, fog computing, which extends the paradigm of cloud computing to the edge of the network, has been proposed and actively studied[2].

In the field of networking, research on CCN (Content Centric Networks) such as NDN (Named Data Networking) has been carried out instead of the conventional IP address-based architecture[3]. It has been shown that CCN can naturally handle cached content in the network by using location- independent content as an identifier, which capable of reducing traffic and latency.

We proposed an autonomous control of server relocation for fog computing systems[4]. In addition, we improved the autonomous control of server relocation to transfer services on the fog network where the processing capacity is heterogeneous, so that the service transfer is commensurate with the required processing capacity[5].

In this paper, to optimize end-user QoS, we control server transfers in a fog computing environment to achieve both shortening of the average response time and fairness between users. For this purpose, we set up three use cases to examine the fairness between users in uniform computer resources, the realization of shortening the average response time in heterogeneous computer resources, and the realization of fairness between users and shortening the average response time in heterogeneous computer resources, respectively.

2 RELATED WORKS

2.1 Fog Computing

In fog computing, the delay time for execution is reduced by selecting and transporting the points necessary for the execution process. For example, in wireless sensor and actuator networking, simple processing should be performed at intermediate nodes, such as fog nodes, before the data collected by sensor nodes are transferred to the cloud. The appropriate placement of the processes at the intermediate nodes improves the command response time for actuation, compared with it is executed at cloud. There is another technique called code-offloading[6]–[9]. Code-offloading is the optimal use of resource-constrained mobile devices. This technology aims to improve the energy efficiency and execution speed of applications. In a mobile application, a part of the application code will be offloaded to the node on fog that has more computational resources to execute the code, rather than running on mobile devices. With the decision, it is possible to save resources such as batteries in mobile devices. In fog computing, the optimal allocation of computational resources is a focus, but there has been no discussion on the optimal placement of content.

Code Bubbling Offload System F. Berg et al.[9] focused on the fact that only two or relatively simple, restrictive system models consisting of three devices have been considered in previous code-offloading techniques. They argue that n-tier architectures become interrelated when multiple classes of various highly distributed resources take advantage of code offloading. For example, the smartwatch (tier 1) connects to the smartphone (tier 2) via Bluetooth. Its smartphone connects via Wi-Fi to a car (tier 3) connected to an edge server (tier 4). In addition, an edge server is connected to a server in a data center (tier 5) across 4G mobile communications and fixed networks. Heterogeneous devices in such an n-tier system differ greatly in terms of energy and computational resources. In this example, they have tiers 1 to 5, but the complexity of the tiers will increase typically from very limited to virtually unlimited. Therefore, they targeted an n-tier environment containing highly distributed heterogeneous resources with different performance characteristics and cost relationships code offload system proposed CoBOS (Code Bubbling Offload System). This proposal includes a concept called code bubbling.

Code bubbling[9] moves code dynamically and adaptively towards more powerful and more distant tiers, enabling an efficient and scalable code-offloading in n-tier environment. CoBOS decreases the energy consumption by 77% and the execution time by 83% for code-offloading in an n-tier environment.

This research aims to optimize the execution point in mobile applications by offloading the code components of the application to a stronger tier at runtime. Therefore, we thought that we could optimize the execution point of the services by considering an architecture that executes the services running in the cloud closer to the user.

2.2 CCN

V. Jacobson et al.[3] proposed a CCN that does not use the traditional IP addressing architecture and Two types of CCN messages, Interest and Data, are used in the CCN communication. This is done by a protocol that is based on the Messages can be sent and received through the FIB(Forwarding Information Base), CS(Content Store), PIT (Pending Interest Table) to send the data back to the requester, three main data structures are used. Using these data structures, CCN exchange messages between Interest and Data. The result retains the simplicity and scalability of IP but offers much better security, delivery efficiency, and disruption tolerance. In this way, CCN put content closer to the user, which allows static contents to be disseminated. However, to treat the running system, we need to care the internal state to continue the process, it is not possible to handle in the same way to provide dynamic content and services.

There is research on cache efficiency in CCN and how to route Interest packets efficiently[10]. These studies have been discussing the treatment of static content and how efficiently distributed content can be considered as transparent, and there is no discussion on how dynamic services can be distributed and deployed on the network.

2.2.1 Service over Content-Centric Routing

In host-oriented communication, technologies such as replication of content and services, caching services, load balancing, and routing of content requests have been enabled by the introduction of applications (e.g., CDN and P2P), but the cost of managing and operating the network is high. Two paradigms, CCN and SCN (Service Centric Network)[11], are used to solve these problems.

If content and services are treated separately, it is not possible to show the relationship between content and services. In practice, services generate new content or perform various functions on existing content, just as some services perform various functions on existing content, but despite the deep relationship between content and services, there is a problem of creating a content-service divide and the gap between CCN and SCN needs to be bridged. Therefore, S. Shanbhag et al. proposed Services over Content-Centric Routing (SoCCeR)[10]. The SoCCeR has added a new ant colony optimization[12] based service routing control layer on top of the CCN to manipulate the routing table for Interest messages. Without affecting the content request and retrieval capabilities of the CCN, they show that they can add SCN capabilities, service requests selectively to service instances with lighter loads and is highly responsive to network and service state changes. However, there is no discussion of how to deploy distributed service instances on the network.

2.2.2 PiGeon

A platform for service orchestration is proposed that addresses the challenges of CCNs in delivering dynamic content and services. M. Selimi et al. propose a Docker container-based PiGeon platform that extends CCN to seamlessly deliver, cache, and deploy at the network edge[13]. PiGeon consists of three types of components: a service controller that monitors the network topology and resource consumption of nodes for service deployment, a service execution gateway for executing service instances at the network edge, and a forwarding node that forwards user requests to nearby caches and other content. The service provider uploads the service to the service controller as a prelude, and the service controller uses the decision engine to decide when and where to place the service in a service algorithm based on the monitoring data.

The PiGeon platform has a service controller as a single point of failure. If a service controller is isolated from the network, the service is not relocated. It is necessary to register the service execution gateway with the service controller in advance, and the monitoring information must be sent periodically from the service execution gateway to the service controller. In medium-sized networks, such as the 80-node experiment in their previous work[14] and the 32-node network in their study, the effect is more than the traffic of monitoring information, but in large networks, the traffic of monitoring information is rapidly increasing and the scalability to the network is low.

2.2.3 Necessity of Fog Computing and CCN Integration

In order to solve the problems we have seen from the research mentioned so far, it would be useful to consider an architecture that allows us to control the deployment of services running in the cloud and dynamically redeploy them as needed. For example, it may be possible to optimize the point of execution of services by running them closer to the user.

Since the CCN is based on the idea of replacing the current TCP / IP with the CCN, there are several discussions on static content caching schemes. However, in today's Internet, which is created by real-world TCP / IP, dynamic content with dynamic services is essential. For example, there is a web page that authenticates the user and displays information appropriate for the user. We wondered if a static content caching scheme is not enough to replace the current Internet with a CCN because of the large amount of these dynamic contents. In the study of fog computing, there is little discussion on the issue of how to place data on a fog network. Therefore, we believe that the problems in the research fields of fog computing and CCN can be mutually resolved by incorporating the way computational resources are handled in the CCN, as introduced in 2.1, into the CCN.

3 CHALLENGE

We consider optimizing quality of service by allowing services running on the network to be dynamically relocated. In this paper, we focus on response time as seen by the client as quality of service. We assume that the response time is expressed as the sum of the network transmission delay and the processing time at the server. When considering the response time, we need to optimize the system in two ways: fairness of the response time among clients and minimization of the processing time. An example is shown and discussed below.

3.1 Use Case that Require Fairness in Delay Times

There is a need for autonomous resource allocation that satisfies the fairness of delay times for participants. For example, in an Internet conferencing system, media quality for all participants may not be maintained if the server is in a single location for clients distributed in different locations on the network with different latency. Therefore, there is a need for autonomous resource allocation that satisfies the equity of delay time for participants.

3.2 Use Case that Require Minimize Processing Time

We assume that the system is available with a dedicated purpose-specific unit, such as TPU (Tensor Processing Unit)[15], to enable machine learning at the edge. Installing a purpose-specific unit for every node may be cost-prohibitive. Therefore, in order to utilize the sparsely placed purpose-specific unit, it is necessary to discover the node with the unit from the topology and to transfer the execution point to the node.

3.3 Assumptions and Requirements for Autonomous Control of Server Relocation System

We need a system that aims at fairness in average response time and shortening of service execution time among users simultaneously. We define the service response time which is the sum of the network latency between the client and server and the processing time of the service.

In addition, the system should reduce the average response time on non-uniform computer resources. For machine learning applications, where the execution time varies greatly depending on the processing performance, the processing time of the service becomes a bottleneck due to the processing performance. Therefore, by monitoring the processing performance of each node and transferring services based on the predicted service response time, services can be transferred to nodes with appropriate processing capacity.

In this system, we assume that the client has a fixed position in the network because it communicates directly with the sensor and user. On the other hand, since servers providing services are arbitrarily located in the fog/cloud, we assume that it is possible to relocate the server by transferring the state of service execution to obtain the necessary resources to execute a process.

4 PROPOSAL FOR AUTONOMOUS CONTROL OF SERVER RELOCATION SYSTEM

In order to optimize QoS for end users, we propose the autonomous control of server relocation for fog computing to reduce both the average response time and the fairness between users.

4.1 The Functions Required by the System

This system collects information about the computing environment of the surrounding nodes, searches for a candidate node that can minimize the service response time, and transfers the server to the selected node.

In searching for candidate nodes, it is not realistic to assume global knowledge across different computing environments such as fog and cloud. As a reasonable scope of search, we assume a routing topology of CCN interest messages when the server is regarded as a resource. It collects PCEL (Available Processing Capacity and Estimated Latency) management information for each node on the path where a message arrives and determines the server transfer based on this information.

Compared with the related studies mentioned in Section 2.2.2, the advantages of our system are as follows. From this system collects information via service communication and the service execution point makes the relocation decision, the system has high availability, including the fact that the single point of failure for each service moves through the fog network and does not affect other services. By this system loads monitoring information on a node via service communication, it is not necessary to register the system at a single location simply by installing it on the node. Also, since the monitoring information is superimposed on the communication of the service, there is no increase in traffic to monitor the network status and the network is scalable.

The following sections describe the main components of the proposal, the estimation of the service processing time, the collection of PCEL information on the message arrival route, and the algorithm for selecting candidate nodes.

Service processing time

C which is the processing power of a node and L which is the amount of processing of the requested service executed by the node are represented as a two-dimensional vector to decompose the processing power of the node into CPU C

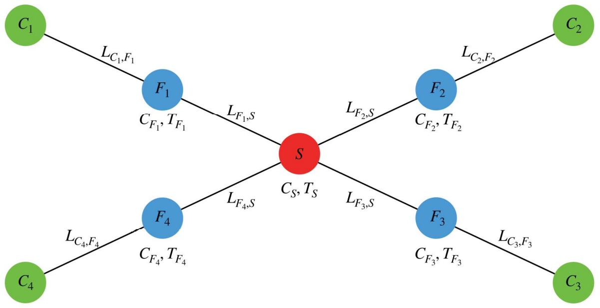

Figure 1: System Configuration Example

and the purpose-specific unit T, respectively. The processing capacity of a fog node is represented by (C_C, C_T), and the amount of processing required for service execution is defined as (L_C, L_T). Based on these processing capacity and processing volume, the service processing time T_{est} is defined as follows.

$$T_{est}(C_C, C_T, L_C, L_T, \alpha) = \begin{cases} \frac{1}{C_C}(L_C + \frac{L_T}{\alpha}) & (C_T = 0) \\ \max(\frac{L_C}{C_C}, \frac{L_T}{C_T}) & (otherwise) \end{cases}$$

However, the CPU can perform the processing required for the purpose specific unit. The α that express ratio is set to 5 for use in later evaluation.

Collecting PCEL information on the message arrival path

In order to treat all the nodes on the path from each client node to the server as candidates for transfer, it is necessary to collect information on the delays between the links traversed and the processing capacity of the nodes traversed. These are called the route PCEL information. In our system, PCEL information is added to the request message at the node that passes by the time the request message reaches the server node, and it is transmitted to the server.

For example, when our system is used as shown in Fig. 1, the following parameters are added to the request message of C_1. In Fig. 1, S is the server node, F is the fog node, and C is the client node.

- L_{C_1,F_1}, which is the communication delay of the link from C_1 to fog node F when a request message from client node C_1 goes through each fog node
- $L_{F_1,S}$, which is communication delay of the link from server node S to F_1
- C_{F_1}, which is the CPU processing power of F_1
- T_{F_1}, which is the processing power of the purpose-specific unit of F_1

This added PCEL information can be acquired by S from the request message. Similarly, S is able to obtain information on the route to and from all clients from the PCEL information attached to the request messages from C_1 to C_4, which are all participating client nodes.

Candidate node selection algorithm

The server selects candidate nodes for transfer from the PCEL information appended to the request message received from the client By using the algorithm shown in Algorithm 1. Algorithm 1 calculates the average and standard deviation of the service response time of the participating clients based on the information that the server shown in Fig. 1 can obtain from the request message. The value is the sum of the service response time multiplied by $1 - R_{std}$ and the standard deviation multiplied by R_{std}, based on R_{std}, which specifies how much importance is placed on the fairness between clients and users. Then, we find the node whose evaluation value is at a minimum.

Algorithm 1 Find Candidate Node

Require: L_{All}:List of L on the Path
Require: $L_{(F_i,C_j)}$:List of L on the Path between F_i to C_j
Require: F_{All}:List of F on the Path
Require: C_{All}:Clients connected to the service
Require: L_C:CPU processing capacity during service execution
Require: L_T:Purpose specific unit throughput during service execution
Require: $node.C_C$: CPU processing capacity of the node
Require: $node.C_T$: Purpose specific unit capacity of the node
Require: R_{std}:Ratio of importance to the standard deviation
Require: $S_{F_i,C_{All}}$:Standard deviation of service response time between F_i to C_{All}
Require: α:Coefficient that represents the ratio when the CPU can handle the amount of processing of the target-specific unit
Ensure: $MinNode$ is candidate node

```
MinCost ← ∞
for all node in F_All do
    for all client in C_All do
        Latency_{node,client} ← Σ L_{node,client} * 2
        Latency_sum ← Latency_sum + Latency_{node,client}
    end for
    Latency_ave ← Latency_sum / C_All.length
    Latency_var ← 1/C_All.length Σ_{i=1}^{C_All.length} (Latency_Ave − Latency_{node,C_i})^2
    ServiceRTT ← T_est(node.C_C, node.C_T, L_C, L_T, α) + Latency_ave
    R_RTT ← 1 − R_std
    COST ← ServiceRTT * R_RTT + S_{node,C_All} * R_std
    if MinCost > Cost then
        MinCost ← Cost
        MinNode ← node
    end if
end for
return MinNode
```

4.2 Functions of the Node

The movement of the proposed system is shown in Fig. 2. This system is assumed to operate at the session layer of all

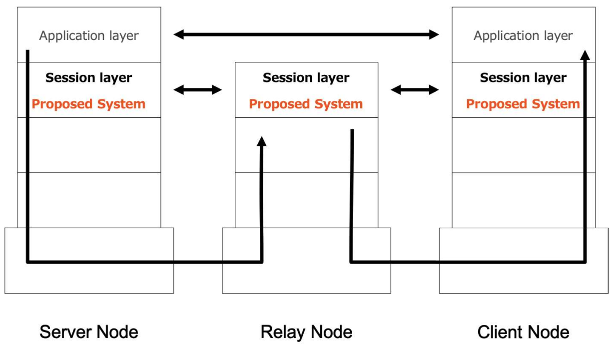

Figure 2: Autonomous Control of Server Relocation System

participating fog nodes. It is preferable that the change in the point of service execution is done transparently to the client and server. In order to implement the collection of PCEL information and transparent addition to the request message, it is convenient to work at the session layer in the seven-layer model. Since management actions such as service transport are operated by resources below the transport layer, they must be located in the upper layer where they are visible, and the session layer is lower than the application layer where clients and servers are running. By relaying communication between server nodes and client nodes at the application layer, the server relocation system at the session layer of server nodes, relay nodes and client nodes share information about resources available at each node. Based on the collected information, it realizes the selection of service execution points and resource allocation.

The proposed system consists of three types of nodes. The proposed system consists of multiple connections: a cloud node that has the contents necessary for service execution and has high processing power, a middle-class fog node that has medium processing power and can communicate with end devices with relatively low latency, and a client node that is an end device such as a smartphone that participates in the server. Based on the client-server communication model, cloud nodes and fog nodes play the role of servers, and leaf nodes play the role of clients. The server monitors the communication status of participating clients and decides whether it should autonomously play the role of the server or delegate the role of the server to other fog nodes based on the communication status. The delegated fog node takes over the role of the server. In this way, we try to optimize the server relocation of the server's role. This is an attempt to reduce the service response time.

There is each fog node has an in-network resource monitoring function, a candidate selection function and a service transfer function. In this section, each function is explained. The autonomous control of server relocation system at each node operates at the session layer to superimpose management information on the communication messages between the server and the client to realize the resource monitoring function in the network. At the same time, it constantly monitors changes in the resources in the network, and if a change is observed, it executes the candidate selection function and, if it is judged to be necessary, it executes the service transfer function to the selected node to optimize the server relocation.

4.2.1 Network Resource Monitoring Function

The in-network resource monitoring function monitors the currently used network resources and collects information to decide whether or not to transfer the service. Specifically, this function monitors and records network information, such as the delay between client nodes participating in the service and the network information, as well as the CPU and memory resources of the fog nodes themselves, while the fog nodes on which this system is installed are deploying the service. As described in the previous section, the PCEL(available Processing Capacity and Estimated Latency) management information described below is included in the normal communication message exchange between the client and server, and information on relay nodes that exist on the route between the client and server is collected.

4.2.2 Candidate Selection Function

The candidate selection function is called when there is a change in the information collected by the in-network resource monitoring function. This function determines whether the service should be transferred. In addition, if it is to be transferred, it will be based on the information collected by the in-network resource monitoring function. Select the appropriate candidate nodes. After the consignee is determined by this function, the server is consigned by the service transfer function.

4.2.3 Service Transfer Function

This function transfers services to the nodes selected by the candidate selection function. The node that is the current server is the candidate server for the new candidate fog node specified by the candidate selection function. The information about the data required for the service is sent with a message informing the user that the service is being entrusted. The server candidate fog nodes are now in the process of gathering all the necessary data and are ready to take on the server. It sends a message informing the user that it can be entrusted to a fog node that is a server of When the fog node, which is currently the server, receives it, it sends a message to a client node that has joined the server It sends a message notifying the new destination to In this way, the service is transferred.

4.2.4 Overall Flow to Optimize Server Placement

We summarize the optimization process explained so far. The client sends a request to the server . The server stores information associated with the request by means of an in-network monitoring function. Using the candidate selection function, we select candidate nodes from the accumulated information. After a candidate node is selected, it sends a message to the candidate node that it will transfer the service. A fog node that receives a message to transport a service confirms that no other service is established at its own node and starts collecting data for service execution, while at the same time sending a message to the node from which the service is being transported to inform it that the service is being prepared. When a

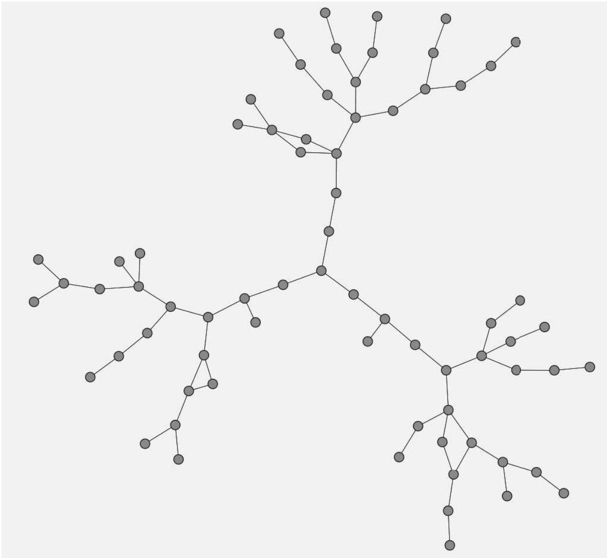

Figure 3: The network topology used in the experiment

fog node completes its data collection, it starts the service and sends a message to the source node telling it that it is ready. The server node that receives the ready message announces the new server to its current clients. The client receives information about the new server and changes the destination of the request to the new server. A client that joins from the middle of the process first sends a request to the original server node, receives information on the current server, and joins the service based on that information.

5 EVALUATION

We defined three use cases to show three aspects: fairness between users on uniform computer resources, reduction of average response time on heterogeneous computer resources, and fairness and reduction of average response time between users on heterogeneous computer resources.

5.1 Simulation Environment

For the network topology, we used the topology generated by BRITE[16], which is a topology generator as shown in Fig. 3. A county of three AS-equivalent nodes was prepared, with about 20 Fog nodes in each AS, and in each simulation, one AS was treated as a cloud environment and two AS were treated as a fog network with clients connected to it. Table 1 shows the parameters used in the simulation. The amount of content cache space owned by each node is also determined by the This was done assuming that the system has enough space to cache all the necessary data.

5.2 Use Case 1: Internet Conference

Use case 1 assumes a multi-point Internet conferencing system to show fairness among users with uniform computer resources. In the Internet conferencing system, the server mixes media data received from all connected terminals and distributes them as a single stream to all terminals. The goal is to keep media quality fair in situations where geographically distributed participants connect to the system. Simulate the behavior of a server moving to the optimal location for clients distributed in different locations on the network with different latency times.

Table 1: Simulation Parameters

Parameters	Value
Cache Algorithm	LRU
Data Rate	10Mbps
Delay	1ms
Simulation time	100s
Server's C_C	100.0
Server's C_T	100.0
Dedicated Unit's C_C	20.0
Dedicated Unit's C_T	50.0
Regular Unit's C_C	20.0
Regular Unit's C_T	0

Simulation Scenario Multiple meetings were defined and the participants of each meeting were placed in the same AS, and the servers of all the meetings were placed together in a different AS than the AS in which the clients were participating. The processing capacity for conducting the conference and the processing capacity of each node were assumed to be constant. The results were compared with the case in which no transfer was performed.

5.3 Use Case 2: Machine Learning

In Use Case 2, we assume a service that uses machine learning on a fog network to demonstrate the realization of shortening the average response time on heterogeneous computational resources. For efficient processing of machine learning, it is better to have a dedicated purpose specific unit such as a TPU (Tensor Processing Unit). However, installing a purpose-specific unit for every fog node may be cost-prohibitive. Therefore, the cost problem is solved by using nodes with dedicated units as part of the fog network. In the fog network, nodes with normal CPU only and nodes with specialized units are mixed together, and the processing power is not uniform in the vicinity of a point. In this situation, we aim to minimize the service response time by transferring the service execution point from the processing accelerated by a purpose-specific unit, such as machine learning, to nodes with appropriate processing power. Experiments in this use case show that a certain percentage of nodes with purpose-specific units such as TPU are randomly placed on the fog network, and the average service response time can be reduced as expected in the entire network.

Simulation Scenario The following simulation scenarios were prepared to achieve the purpose of the experiment. Multiple clients connected to the network are available with servers in the distant cloud and fog nodes with various processing capabilities. Clients make service requests that be able to take advantage of the processing power of purpose-specific units such as the TPU. In this experiment, we show that the service response time can be minimized by arranging about % of all fog nodes. For that purpose, the ratio of nodes equipped with a dedicated unit in the fog network was changed from 0% to 100% in 20% increments, and an experiment was performed 100 times in which the arrangement was randomly changed.

5.4 Use Case 3: Video Delivery Service

In the third use case, we assumed a video delivery service to demonstrate the realization of fairness among users and reduction of average response time on heterogeneous computational resources. In some cases, you may want to cut out noteworthy parts of the video, overlay the board, or overlay chroma-keyless CG. When we want to perform such advanced video delivery, we need computer resources for video delivery. In addition, it is required that the communication delay must be low for the participants to comfortably watch videos. As in Use Case 2, it is not practical to deploy computational resources for video delivery to all nodes due to cost issues. Therefore, optimizing the execution point requires both processing time and communication delay, which are affected by processing performance. The results are compared with the case where simple average service response times are taken into account.

Simulation Scenario The video distributor sends the video to the server. The client receives the delivery stream processed by the server. While transferring to nodes with processing power suitable for video processing, the delay in video delivery to participants is reduced to such a node. When new participants are added from a different AS than the one that attracts video distributors and initial participants, the processing power and participants were observed transferring based on the fairness of the communication delay. New participants are timed to join the video feed at 50 seconds from the start of the simulation, such as Simulations were performed.

6 RESULTS AND DISCUSSION

The results and discussion of the experiments conducted for each use case are described, respectively.

6.1 Use Case 1: Internet Conference

In this use case experiment, we achieved fairness between users on a uniform computational resource. The experimental results for Use Case 1 are shown in Fig. 6 and Fig. 7. Figure 6 shows the change in service response time when the proposed system is not used, and Fig. 7 plots the change in service response time against time when the proposed system is used. The users of the proposed system are gradually transferred to the one with less network latency.

At the timing of the start of the simulation, the two conference streams are shown in Fig. 4. It is sent from different AS to a single AS, and the network traffic is aggregated. In the network after the transfer, the servers are transferred within each AS, as shown in Fig. 5, and the two conference avoids the aggregation of meeting traffic.

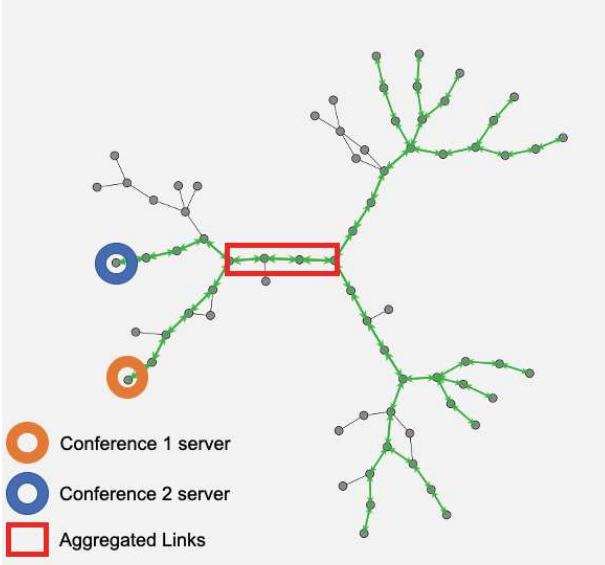

Figure 4: Use case 1: Network status before the transfer begins

In Figure 5, the fairness between users depends on which node on each communication path the server will be transferred to. It is possible to achieve the fairness required by each application by adjusting the R_{std} used in the algorithm 1 for destination determination. Figure 8 shows the trend of the mean and standard deviation of the service response time for a single conference among the results of the selecting candidate nodes for transfer, where the value of R_{std} is set to 0 and only the mean of the service response time is important. Figure 9 sets the value of R_{std} as 0.7 and the node selection with a weighted standard deviation of 70% of the response time, showed the average service response time for the same meeting as in Fig. 8. Comparing the two figures, we can confirm that the result of Fig. 9, weighted at 70%, is fairer (i.e. smaller deviation) than the result of Fig. 9, which shows the fairness of the transfer between users. We are able to confirm that the fairness of the transfer between users is maintained. In this way, we achieved fairness among users with uniform computer resources by performing transfers to shorten the service response time and adjusting the parameters to meet the requirements of the application.

6.2 Use Case 2: Machine Learning

In this use case, we have shown the realization of reduced average response time on heterogeneous computational resources. Figure 10 shows the average service response time of 100 simulations with the TPU placement probability varying from 0% to 100% in 20 percent increments. In the worst case with the TPU node placement rate set to 0%, transfers do not occur because the service response time is shorter if the service continues to be processed at the initial server node than if it is transferred to the fog node. When the placement rate of TPU nodes is increased, transfers occur to TPU nodes and the processing capacity is uneven The service response in Fig. 10 shows that the optimization of the execution point on the fog network is correctly done This can be seen by looking

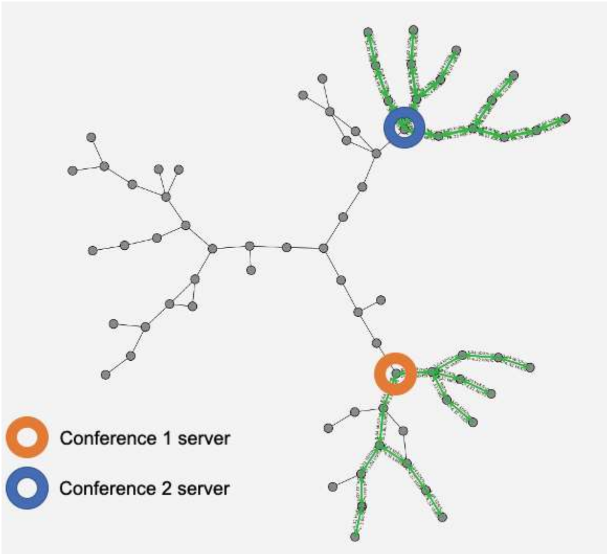

Figure 5: Use case 1: Network status after transfer

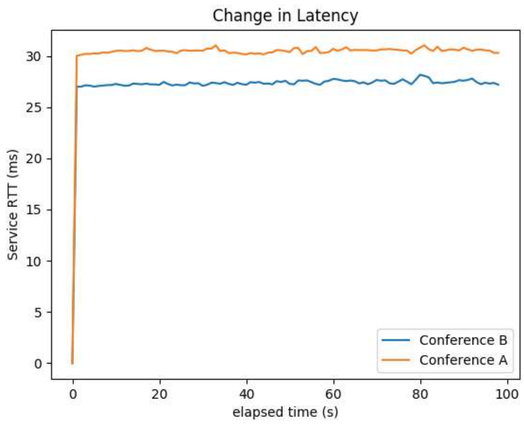

Figure 6: Use case 1: Changes in service response time if the proposed system was not used

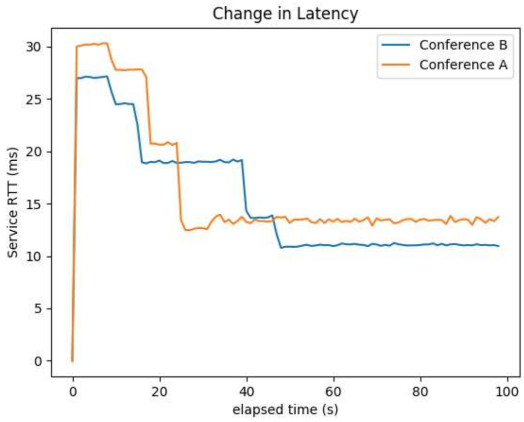

Figure 7: Use case 1: Changes in service response time when using the proposed system

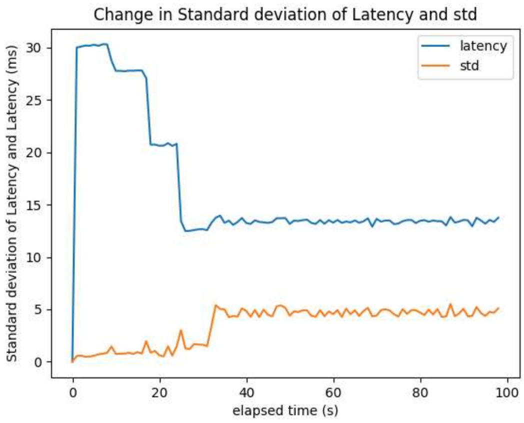

Figure 8: Mean and standard deviation of service response time for conference A when R_{std} is set to 0%

at the changes in time. We have shown that this reduces the average response time for non-uniform computer resources.

6.3 Use Case 3: Video Delivery Services

In Use Case 3, we showed how to achieve fairness between users and reduce the average response time on non-uniform computational resources. A simulation with a 20% probability of deploying a dedicated unit to process the video. Figure 11 shows the mean and standard deviation of the service response time after 100 trials. In the period from 0 to 50 seconds in Fig. 11, when only the clients stuck in one AS are connected to the video delivery server, we are able to see that transfer reduces the standard deviation value, i.e., the fairness among the participating clients, but the mean of the service response time can be reduced significantly, and therefore transfer is used. In addition, after 50 seconds of simulation time, a new client of another AS began to connect to the video delivery server. Temporarily, the values of the mean and standard deviation of the service response time are increasing. However, we found a fog node where both the standard deviation and service response time values can be reduced by about 75 seconds of simulation time, and we are capable of seeing a gradual change in the transfer.

6.4 Discussion

In this section, we discuss the validity and constraint of the design of the system in the view point of flexibility to scale and response to changes in available resources.

For scalability to the size of network, the proposed system does not require aggregation of information for the entire network nodes. The server node executing the service determines the service transfer based only on the PCEL information of each client and the relaying node on their path. In this design, we limit the discovery of the available resource in node on the message path. Our evaluation shows the effectiveness, however the result of the service transfer may be sub-optimal.

In terms of service scalability, the proposed system improved fairness and response time by controlling the placement of a single server instance. We have not considered the case to utilize multiple instances to scale.

We discuss the network change issues from the perspective

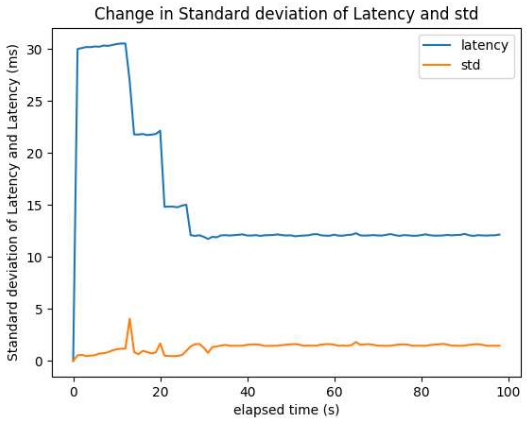

Figure 9: Mean and standard deviation of service response time for conference A when R_{std} is set to 70%

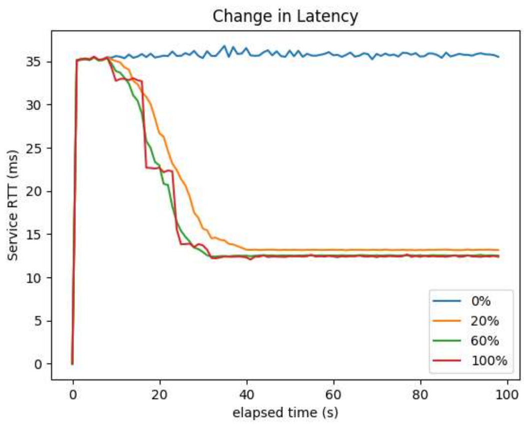

Figure 10: Changes in the average service response time as a percentage of TPU nodes deployed

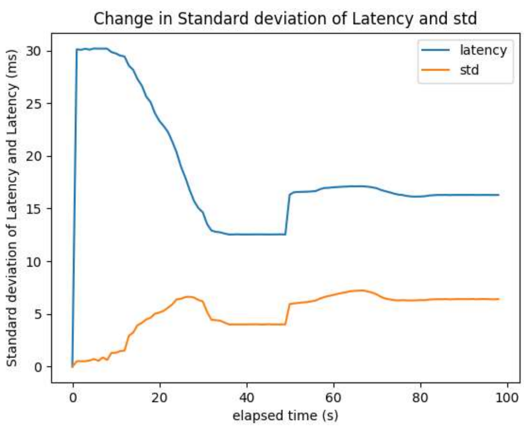

Figure 11: Use case 3: Mean and Standard Deviation of Service Response Time

of adapting to changes in paths and available resources.

Regarding the change of the path due to the reflection of the network topology etc., proposed system assumes that the interest message of the base CCN mechanism follows, and has no other mechanism. To estimate the available resources, PCEL information is piggybacked on the interest message and collected at the server node. From these, it can be said that the possibility of immediate adaptability to changes in the path of this system and changes in available resources depends on the frequency of exchange of interest messages.

If there is high frequency of exchange, it is possible to estimate precisely than when there are few. The evaluation scenarios discussed in this paper, target applications inherently assume stable frequency of sending and receiving messages. In contrast, if the use case only interacts with clients infrequently, they may not be able to keep up with changes in paths and available resources. It may be necessary to increase the volume of message flow for the resource monitoring purpose.

In summary, the situations in which this system is considered applicable are follows. This system makes it possible to relocate servers by utilizing frequent exchanges of messages between clients and servers in response to changes in the network. While this system is scalable to the size of the network because it does not require aggregation of information for the entire network, there is a limitation to the service scalability.

7 CONCLUSION

Fog computing, which extends the cloud computing paradigm to the edge of the network, has been proposed and is being actively researched. In the field of networking, there is research on CCN that use location-independent content as identifiers instead of the traditional IP address-based architecture. So far, we have proposed a system that aims at fairness in response time and shortening of service execution time between users, respectively. Therefore, in this study, we proposed the autonomous control of server relocation for fog computing systems to optimize QoS for end users, which achieves both shortening the average response time and fairness between users. To this end, the system achieves inter-user fairness on uniform computer resources, reduction of average response time on non-uniform computer resources, and we tested the fairness between users and the reduction of the average response time on non-uniform computer resources by setting up three use cases for each.

In the future, we will consider further expansion of the proposed system based on the use cases mentioned in this article, and work to enhance the usefulness of the proposed system.

REFERENCES

[1] Ministry of Internal Affairs and Communications, Japan: *The 2018 White Paper on Information and Communications in Japan*, Japanese Government (2018).

[2] Bonomi, F., Milito, R., Zhu, J. and Addepalli, S.: Fog Computing and Its Role in the Internet of Things, *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pp. 13–16 (2012).

[3] Jacobson, V., Smetters, D. K., Thornton, J. D. et al.: Networking Named Content, *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pp. 1–12 (2009).
[4] Kamada, K., Inamura, H. and Nakamura, Y.: A Proposal of Autonomous Control of Server Relocation for Fog Computing Systems(In Japanese), *IPSJ SIG Technical Report*, Vol. 2018-MBL-89, No. 1, pp. 1–5 (2018).
[5] Kamada, K., Inamura, H. and Nakamura, Y.: Autonomous Control Scheme of Server Relocation for Non-Uniform Computing Capacity in Fog Networks(In Japanese), *DICOMO2019*, Vol. 2019, pp. 1204–1211 (2019).
[6] Cuervo, E., Balasubramanian, A., Cho, D.-k. et al.: MAUI: making smartphones last longer with code offload, ACM Press, p. 49 (2010).
[7] Chun, B.-G., Ihm, S., Maniatis, P. et al.: CloneCloud: elastic execution between mobile device and cloud, ACM Press, p. 301 (2011).
[8] Kosta, S., Aucinas, A., Hui, P. et al.: ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, *2012 Proceedings IEEE INFOCOM*, pp. 945–953 (2012).
[9] Berg, F., Dürr, F. and Rothermel, K.: Increasing the Efficiency of Code Offloading in N-tier Environments with Code Bubbling, *Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS 2016, ACM, pp. 170–179 (2019).
[10] Shanbhag, S., Schwan, N., Rimac, I. and Varvello, M.: SoCCeR: services over content-centric routing, *Proceedings of the ACM SIGCOMM workshop on Information-centric networking - ICN '11*, ACM Press, p. 62 (2011).
[11] Wolf, T.: Service-Centric End-to-End Abstractions in Next-Generation Networks, *Proceedings of 15th International Conference on Computer Communications and Networks*, IEEE, pp. 79–86 (2006).
[12] Dorigo, M. and Birattari, M.: Ant Colony Optimization, *Encyclopedia of Machine Learning* (Sammut, C. and Webb, G. I., eds.), Springer US, pp. 36–39 (online), https://doi.org/10.1007/978-0-387-30164-8_22 (2010).
[13] Selimi, M., Navarro, L., Braem, B., Freitag, F. and Lertsinsrubtavee, A.: Towards Information-Centric Edge Platform for Mesh Networks: The Case of City-Lab Testbed, *2020 IEEE International Conference on Fog Computing (ICFC)*, IEEE, pp. 50–55 (online), doi10.1109/ICFC49376.2020.00016.
[14] Selimi, M., Lertsinsrubtavee, A., Sathiaseelan, A., Cerdà-Alabern, L. and Navarro, L.: PiCasso: Enabling information-centric multi-tenancy at the edge of community mesh networks, Vol. 164, p. 106897 (online), doi10.1016/j.comnet.2019.106897.
[15] Jouppi, N. P., Borchers, A., Boyle, R. et al.: In-Datacenter Performance Analysis of a Tensor Processing Unit, *Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17*, ACM Press, pp. 1–12 (2017).
[16] Medina, A., Lakhina, A., Matta, I. and Byers, J.: BRITE: An approach to universal topology generation, *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE, pp. 346–353 (2001).

(Received November 14, 2021)
(Accepted October 1, 2021)

Kouki Kamada received his B.E. and M.E. degrees in information science from Future University Hakodate, Japan in 2019 and 2021. His research interests include cloud infrastructure and mobile computing. He currently works in Cybozu, Inc.

Hiroshi Inamura is a professor of School of Systems Information Science, Future University Hakodate, since 2016. His current research interests include mobile computing, system software for smart devices, mobile and sensor network and their security. He was an executive research engineer in NTT docomo, Inc. He received B.E., M.E. and D.E. degree in Keio University, Japan. He is a member of IPSJ, IEICE, ACM and IEEE.

Yoshitaka Nakamura received B.E., M.S., and Ph.D. degrees from Osaka Univer-sity in 2002, 2004 and 2007, respectively. He is currently an associate professor at the Faculty of Engineering, Kyoto Tachibana University. His research interest includes infor-mation security and ubiquitous computing. He is a member of IEEE, IEICE, and IPSJ.