Electronic Preprint for Journal of Information Processing Vol.29

Recommended Paper

Management and Network Orchestration for Edge/Fog-based Distributed Data Processing

Hiroki Watanabe[1,a)] Kazuki Hayashi[1,b)] Tomonori Sato[1,c)]
Takao Kondo[1,2,d)] Fumio Teraoka[3,e)]

Received: March 10, 2021, Accepted: July 7, 2021

Abstract: In the age of edge/fog computing, it is important to consider not only computing resources but also network resources when hosting services. Since service is composed of multiple small functions in the microservice architecture, we treat a service as a set of BFs (basic functions) that fulfill a single task. It is required to place BFs at edge/fog nodes considering the computing resources and network requirements within a practical time. This paper proposes a MANO (Management and Network Orchestration) for deploying services composed of multiple BFs with requirements to computing and network resources of distributed nodes. The proposed MANO considers the computing resources of edge/fog/cloud as well as the network delay and the bandwidth between them. This paper proposes an optimal method and a heuristic method for calculating the placement of BFs. The evaluation results show that the placement calculation time for a service composed of four BFs is about 10 seconds with the optimal method and about 20 seconds with the heuristic method. The calculation time is within the practical range.

Keywords: edge computing, fog computing, MANO

1. Introduction

Emerging of applications with delay sensitive feature and high volume content delivery, edge/fog computing is spreading to host applications in the vicinity of users in the network. It reduces the latency between users and application endpoints and the amount of traffic in the backbone network. Meanwhile, combining multiple simple functions to realize complex network services and application services has been spreading such as service chaining [1] and microservice architecture [2]. In such services, a common issue is where each function should be placed in the network focusing on the computing resources. It is expected that large ISPs (Internet Service Providers) and carriers will provide not only network services but also edge/fog/cloud computing services utilizing their own networks in the near future. In such an environment, the functions composing a service should appropriately be placed on edge/fog/cloud considering network resources as well as computing resources. Therefore, a scalable MANO (Management and Network Orchestration) mechanism in a geographically distributed environment is a challenge.

This paper assumes that a service is composed of one or more simple functions called *Basic Functions* (*BFs*) which have requirements to network and computing resources. BFs are appropriately distributed on edge/fog/cloud and compose a chain called a *Basic Function Chain* (*Chained-BF*). A Basic Function Chain may have various shapes such as linear chain and chain with branch and merge. Data are processed by BFs one after another along the Basic Function Chain.

This paper proposes a MANO architecture for Chained-BF. This paper also proposes a heuristic placement method of BFs in addition to an optimal placement method based on linear programming in a mathematical model. In the proposed MANO mechanism, resource information is managed at two levels: per machine in a computing site (e.g., a single data center) and per domain (e.g., a single ISP). The proposed MANO mechanism decides the machines on which BFs should be placed considering the requirements of each BF and the shape of the chain. This paper evaluates the time for calculating the placement of a Chained-BF as changing the shape of the Chained-BF and the number of BFs in the Chained-BF. In addition, this paper evaluates the end-to-end latency of packet delivery through a Chained-BF deployed with the optimal method and the heuristic method.

2. Basic Function based Distributed Data Processing Platform

Cloud computing services are centralized systems in a single administrative domain. Therefore, users have only a few choices

[1] Graduate School of Science and Technology, Keio University, Yokohama, Kanagawa 223–8522, Japan
[2] Computer Security Incident Response Team, Keio University, Minato, Tokyo 108–8345, Japan
[3] Faculty of Science and Technology, Keio University, Yokohama, Kanagawa 223–8522, Japan
[a)] nelio@inl.ics.keio.ac.jp
[b)] gordon@inl.ics.keio.ac.jp
[c)] glue@inl.ics.keio.ac.jp
[d)] latte@itc.keio.ac.jp
[e)] tera@keio.jp

The preliminary version of this paper was published at IPSJ SIGDPS Technical Reports, March 2020. The paper was recommended to be submitted to Journal of Information Processing (JIP) by the chief examiner of SIGDPS.

about data centers to which their service is deployed. *BF-DDPP* (*Basic Function based Distributed Data Processing Platform*) envisions an era in which network operators have medium-scale computing resources at the PoPs (point of presences), which can be handled transparently by service developers. **Figure 1** illustrates an overview of BF-DDPP and how a service (Chained-BF) is treated. Making computing resources at edges open to third parties, which have traditionally been reserved for network operators, it is expected that medium-scale data centers are available to users at, e.g., the prefectural level in Japan. Such a medium-scale data center is defined as a *Spot* in BF-DDPP. In order to execute a service using multiple Spots, an administrative domain that controls the Spots is necessary. In BF-DDPP, the administrative domain is defined as a *Zone*. When a user starts execution of a Chained-BF, the orchestrator decides on which Spot each BF in the Chained-BF is deployed according to requirements, such as the delay and the bandwidth. BF-DDPP also considers multi-Zone coordination.

2.1 Basic Function and Basic Function Chain

BF (*basic function*) is a program that should fulfill only a single task and work together with other BFs. Each BF has requirements for computing resources such as CPU and RAM usage, and requirements to network resources such as acceptable delay and bandwidth usage. A Chained-BF is composed of one or more BFs and terminated by *GNs* (*Gate Nodes*), which are located at borders of a Zone. A Chained-BF is treated as a service in BF-DDPP. **Figure 2** shows examples of various shapes of Chained-BFs such as branch-type, merge-type, and cycle-type.

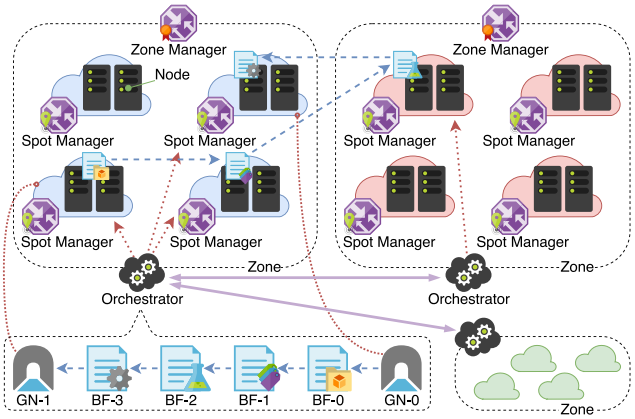
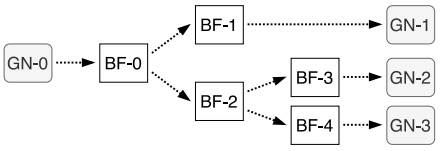
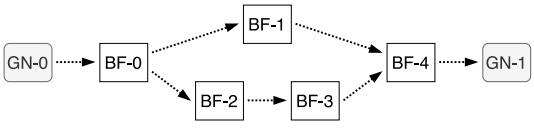


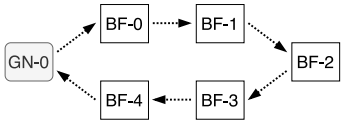
Fig. 1 An overview of Basic Function based distributed data processing platform and MANO.



(a) An example of branch-type Chained-BF.



(b) An example of merge-type Chained-BF.



(c) An example of cycle-type Chained-BF.

Fig. 2 Various shapes of Chained-BF.

3. Related Work

3.1 Distributed Data Processing

Service chaining and microservice architecture are typical examples of distributed data processing focused on in this paper. Service chaining is an approach for realizing a single service by chaining multiple small services. Data processing is achieved by passing data through a pre-defined chain. One of the service chaining technologies is SFC (Service Function Chaining) [1], which is used to provide network services such as firewall and load balancing. An overview of the SFC architecture is shown in **Fig. 3**. In SFC, the classifier tags the packet as to which SFP (Service Function Path) it applies when a packet arrives at an ingress node. The tagged packets pass through the SF (Service Function) according to the order defined in the SFP and finally reach the egress node. In a SFC-enabled domain, SFFs (Service Function Forwarders) on the way interpret the tag of the packet and forward the packet to the appropriate SF or forward the packet to the next node. In service chaining, each function is often connected with an input and an output like a pipe in a shell script.

Microservice architecture [2] is an approach for realizing services by invoking one function on another function. **Figure 4** shows a general application service overview with microservice architecture. In Fig. 4, the application is composed of four

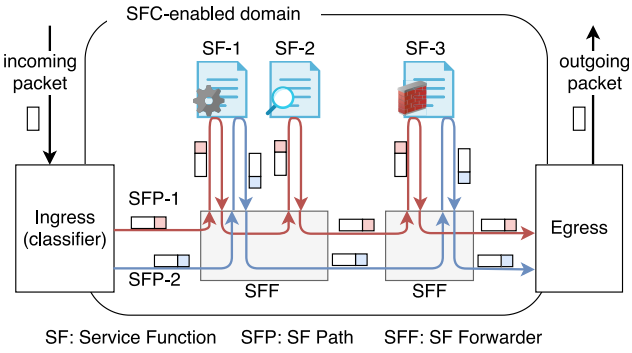


Fig. 3 SFC architecture [1].

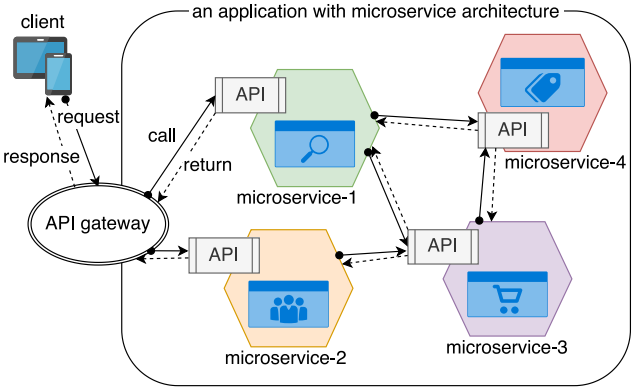


Fig. 4 General microservice architecture.

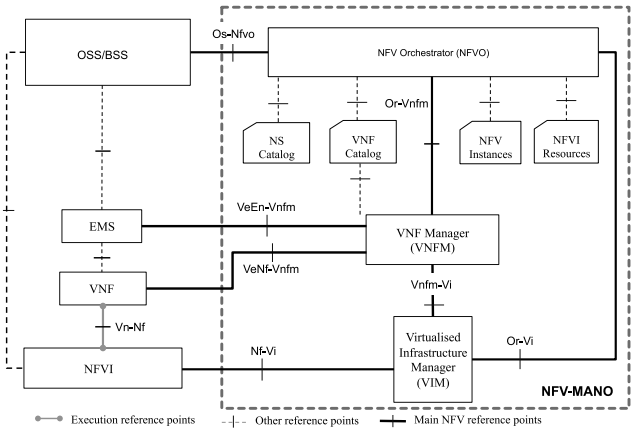


Fig. 5 NFV-MANO Architecture [7].

microservices. Each microservice uses another microservice through APIs such as REST and RPC. Recent cloud services are often provided with microservice architecture in mind, and application services are increasingly adopting microservice architecture. In addition, it is assumed that each microservice is deployed on edge/fog nodes [3], [4], [5] along with the expectation of edge/fog computing. Microservices are expected to be used not only in application services but also in the 5G Core network [6]. Related technologies of a standard resource management and resource management in distributed environments for service chaining and microservices are described in the next section.

3.2 ETSI NFV-MANO

NFV (Network Functions Virtualization) is an approach for realizing network functions on general purpose hardware instead of specialized hardware. The management and coordination functions required to build an NFV environment are called MANO (Management and Network Orchestration). **Figure 5** shows the architecture of NFV-MANO [7] proposed by ETSI (the European Telecommunications Standards Institute). The MANO mechanism is composed of NFVO (NFV Orchestrator), VNFM (Virtualized Network Functions Manager), and VIM (Virtualized Infrastructure Manager). ETSI adapts NFV-MANO to a multi-domain environment by standardizing the NFVO peer collaboration mechanism [8]. DASMO [9] has proposed an architecture which integrates ISM (In-Slice Management) with ETSI NFV-MANO. ISM introduces an interface to manage the inner workings of network slices, allowing third parties to manage slices. MEC-NFV [10] introduces the concept of VAF (Virtual Application Function) to integrate ETSI NFV with MEC (Multi-access Edge Computing/Mobile Edge Computing) [11], [12]. ETSI NFV-MANO and its extended methods manage resources virtualized by NFVI (NFV Infrastructure) with VIM. We introduce a mechanism for collecting resource information per machine in a distributed data center and per data center in a network provider, referring to the multi-domain architecture of ETSI NFV-MANO.

3.3 Resource Management in Edge/Fog Computing

FogTorchΠ [13] proposes a method for deploying complex applications in a heterogeneous fog environment. FogTorchΠ models QoS, infrastructure, and applications and outputs multiple suitable deployment locations through a two-stage search. Fog at Edge [14] is an edge computing platform. It proposes DNR (Distributed Node-RED), an extension of the flow-based programming tool Node-RED [15] for distributed environments. Both of them handle the location of users and network delay, but do not handle bandwidth.

A MEC application in Ref. [16] adopts a microservice architecture and assumes that the application client is a mobile device. In Ref. [16], a mathematically optimal model and a Pareto optimal algorithm are proposed to select the migration destination of a microservice when the client is moving. The optimal model is difficult to implement because it requires prior knowledge of the client's travel path. In the Pareto optimal algorithm, the model is approximated as a shortest path problem. However, Ref. [16] considers the delay and the migration cost, but does not consider the bandwidth. Our proposed MANO considers the delay and the bandwidth as a network resource as well as a computing resource.

3.4 VNF Deployment Problem in NFV

Approaches to VNF placement in NFV can broadly be classified into two types. One is a method that uses a proprietary algorithm. The method for provisioning SFC using dynamic programming [17] calculates a VNF deployment that maximizes the provider's revenue. A recursive algorithm using divide-and-conquer and memorialization allows execution in polynomial time. Methods for managing VNF using genetic algorithms take account of both computing and network resources [18]. In this work, after a random selection with depth-first search, NF placement is determined by an algorithm that simulates GA (Genetic Algorithms) crossings and mutations. However, Ref. [18] does not assume multi-domain platform while our proposal does. Therefore, the method in Ref. [18] does not take account of the delay as a requirement to network. In contrast, our method takes account of not only the bandwidth but also the delay as a requirement to network.

The second is to model the problem appropriately and solve it with a solver. In the latency-aware VNF deployment method [19], the authors propose a method to determine VNF deployment that minimizes resource consumption, satisfies a certain amount of end-to-end latency, and does not violate SLAs (Service Level Agreements). A VNF placement method based on a mathematical optimization model formulates NF placement and chaining problems and solves it using ILP (Integer Linear Programming) [20]. In this work, a heuristic approach is also proposed to cope with the increased computation time with large infrastructures.

A MANO for a Basic Function based distributed data processing platform needs to take account of not only the delay and the bandwidth, but also policies of BFs and a Chained-BF. It is difficult to find the optimal solution in polynomial time, based on a proprietary algorithm. Our proposed MANO adopts a solver for the BF placement problem to meet a variety of requirements.

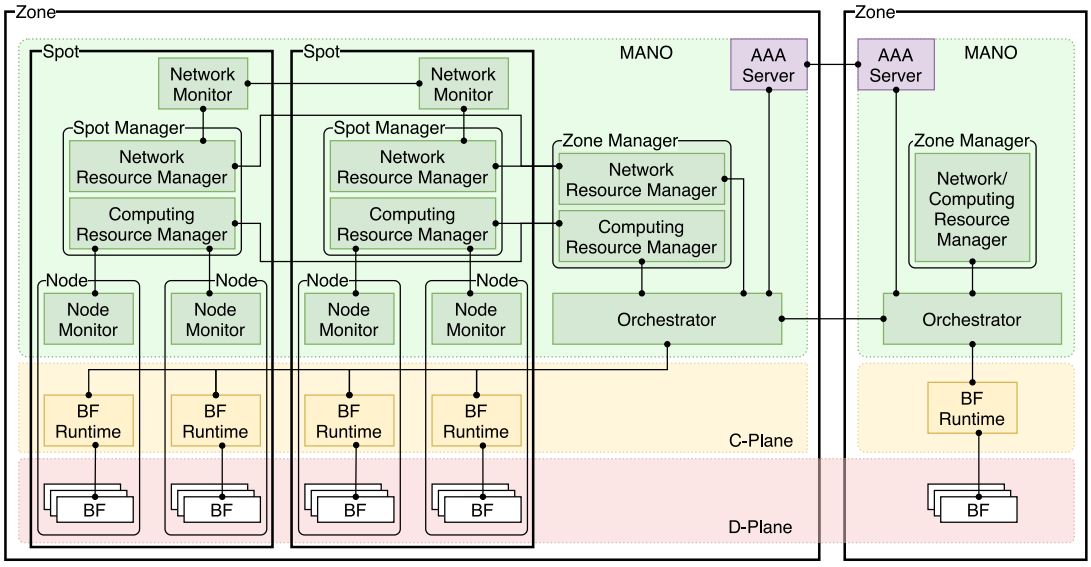
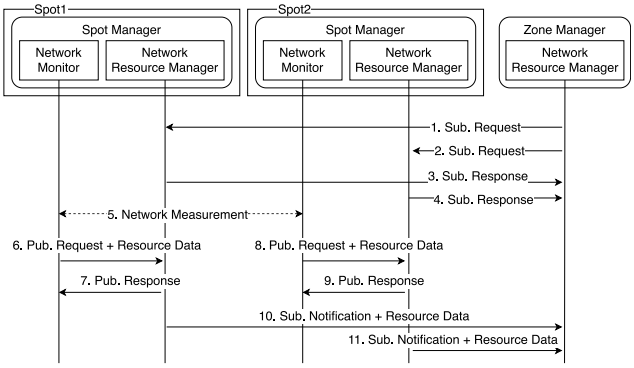


Fig. 6 Proposed MANO architecture.

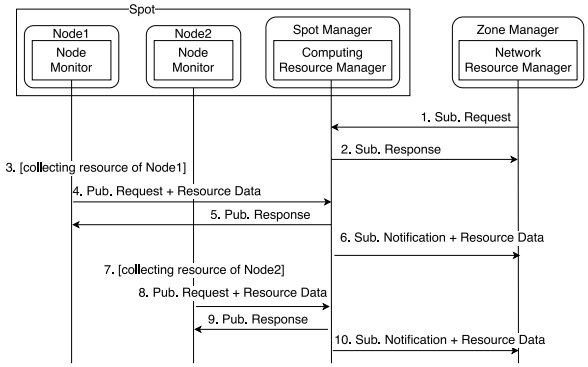
4. Design of MANO

4.1 Overview of a Basic Function based Distributed Data Processing Platform

Figure 6 shows the MANO architecture for BF-DDPP assumed in this paper. A BF-DDPP is composed of a single Zone. A Zone is an administrative domain of a network provider and is composed of multiple Spots, the AAA (Authentication, Authorization and Accounting) server, the *Zone Manager*, and the *Orchestrator*. The AAA server exists in a Zone for BF-DDPP coordination and accounts for the use of BFs and Chained-BFs. The Zone Manager is composed of the *Network Resource Manager* and the *Computing Resource Manager*. The Orchestrator calculates a placement of a Chained-BF satisfying computing and network resource requirements of each BF. A Spot is composed of a set of *Nodes*, the *Spot Manager*, and the *Network Monitor*. A Node is a physical or a virtual machine and is the smallest unit that manages computing resource information. It is assumed that each Spot is a medium-scale distributed data center of an ISP or a carrier. The Network Monitor observes network resources such as the delay between its Spot and another Spot and the utilization of each link in its Spot. The Spot Manager is composed of the Network Resource Manager and the Computing Resource Manager. All Spots are connected with each other through the network of the network provider. All Spots in a Zone share the same management policy. Two programs are running on each Node: the *Node Monitor* and the *BF Runtime*. The Node Monitor acquires information of the computing resources of a Node (CPU usage and RAM usage). The BF Runtime is responsible for launching BFs and for the reachability between BFs. The Orchestrator requests the BF Runtimes on the selected Nodes to launch the BF after it calculates the placement of each BF. It is very important for the BF placement to be aware of network resources such as the delay and the bandwidth as well as computing resources. Therefore, the BF-DDPP introduces a layer for the collection and management of resource information.



(a) Network resource collecting sequence.



(b) Computing resource collecting sequence.

Fig. 7 Resource collecting sequences.

4.2 Mechanism for Collecting Resource Information

Figures 7 (a) and (b) show sequences of collecting a network resource and a computing resource, respectively. Resource information is collected by the Publisher/Subscriber model. The Network Monitor measures the network status (bandwidth and delay) between the Spots and sends it to the Network Resource Manager in the Spot Manager. A Network Resource Manager collects the network resource information from Network Monitors. A Computing Resource Manager collects the computing resource information from Node Monitors. Both resource managers aggregate each resource information and send it to the Zone Man-

ager. The Zone Manager collects and stores the resource information from the Network Resource Manager and the Computing Resource Manager of Spot Managers. The Zone Manager provides resource information in the Zone to help the Orchestrator calculate the appropriate placement of BFs. The resource information is periodically updated by notifications from Spot Managers and is sent upon request from the Orchestrator.

5. Optimal Placement of BFs

5.1 Modeling BF-DDPP

Table 1 denotes the definitions of the symbols used in the model. The model defines the number of cores of vCPUs (virtual CPUs) and the capacity of RAM (main memory) as the computing resources of a Spot. The model also defines the maximum capacity that can be allocated to BFs and the capacity that has already been allocated to BFs. The unit price of vCPUs and RAM can be defined in each Spot. The model treats only the links between Spots as network resources. Furthermore, two types of links are considered. One is a closed network dedicated to BF-DDPP and the other is a public network. This is because some Chained-BFs may require guarantee of certain bandwidth and network delay while a best-effort network quality may be acceptable to some Chained-BFs depending on services provided by Chained-BFs. Assuming that QoS can be guaranteed in a private network, maximum bandwidth, reserved bandwidth, and network delay are defined as service parameters. The unit price of vCPU and RAM required to run a BF as well as the unit price of the BF itself can be defined at each Spot. The redundancy of a BF is defined as the value indicating how many redundant instances of the BF are created in a Chained-BF. The Spot in which a GN (Gate Node) is installed is determined by the location of the Chained-BF User. A *Chain* is defined as a connection between a BF and another BF or between a BF and a GN. A Chained-BF is composed of one or more Chains. $A^N_{m,n} \in \{0,1\}$ denotes the existence of a link between BFs or between a BF and a GN in which m and n are a BF and a GN, respectively. Similarly, $Q^{band}_{m,n}$ denotes bandwidth requirements between BFs or between a BF and a GN. $A^D_{r,m,n} \in \{0,1\}$ denotes delay requirements between BFs or between a BF and a GN. A matrix with elements of value 0 or 1 can be used to configure the shape of a Chained-BF as well as the requirements to bandwidth and delay for each Chain.

Table 1 Definition of the symbols in the proposed model.

Symbol	Definition
$i \in S$	Spot
C_i / C^{used}_i	Max / Used number of vCPUs in Spot i
P^{cpu}_i	Unit Price of vCPU in Spot i
M_i / M^{used}_i	Max / Used amount of RAM in Spot i
P^{ram}_i	Unit Price of RAM in Spot i
$(i,j) \in L$	Link between Spot i and j
$B_{i,j}$ / $B^{used}_{i,j}$	Max / Used bandwidth of Link (i,j)
$D_{i,j}$	Delay of Link (i,j) (Private)
$P^{band}_{i,j}$	Unit Price of leased Link (i,j)
$D^{pub}_{i,j}$	Delay of Link (i,j) (Public)
$m \in F$	BF (Basic Function)
F^{cpu}_m / F^{ram}_m	Requirement to vCPU / RAM of BF m
P^{bf}_m	Unit Price of BF m
F^{dup}_m	Redundancy of BF m
$n \in G$	GN (Gate Node)
$A^G_{n,i} \in \{0,1\}$	GN n is installed on Spot i or not
$N = F \cup G$	Union set of BFs and GNs
$A^N_{m,n} \in \{0,1\}$	Chain exists between BF and GN (m/n) or not
$Q^{band}_{m,n}$	Bandwidth requirement for Chain between m and n
$r \in D^{req}$	Delay requirement for Chain
$A^D_{r,m,n} \in \{0,1\}$	Pair of m and n satisfying r exists or not
Q^{delay}_r	Sum of network delay satisfying r
$x_{l,m,i} \in \{0,1\}$	BF m is placed at Spot i or not
$l \in \left\{1,2,...,\prod F^{dup}\right\}$	Variables for redundancy F^{dup}_m.

5.2 Problem of BF Placement to Spot

The problem of BF placement to Spot in the MANO is represented by a model that takes as input the Spot $A^G_{n,i} \in \{0,1\}$ to which the GN is placed and outputs the Spot $x_{l,m,i} \in \{0,1\}$ to which the BF is placed. When GN n is placed to Spot i, $A^G_{n,i} = 1$ is satisfied. Finally BF m is placed at Spot i such that $x_{l,m,i} = 1$.

Equation (1) shows the objective function. The first term represents the sum of the network delays $x_{l,m,i} \cdot x_{l,n,j} \cdot A^N_{m,n} \cdot A^D_{r,m,n} \cdot D_{i,j}$ between BF m and BF n for all BFs, Spots, r, and l. The second term represents the sum of the network delays $x_{l,m,i} \cdot A^G_{n-|F|,j} \cdot A^N_{m,n} \cdot A^D_{r,m,n} \cdot D_{i,j}$ between BF m and GN n for all BFs, GNs, Spots, r, and l.

$$\sum_{\substack{m,n \in N \\ :m<n<|F|}} \sum_{\substack{i,j \in S \\ r \in Q,l}} x_{l,m,i} \cdot x_{l,n,j} \cdot A^N_{m,n} \cdot A^D_{r,m,n} \cdot D_{i,j} + \sum_{\substack{m,n \in N \\ :m<|F| \le n}} \sum_{\substack{i,j \in S \\ r \in Q,l}} x_{l,m,i} \cdot A^G_{n-|F|,j} \cdot A^N_{m,n} \cdot A^D_{r,m,n} \cdot D_{i,j}. \quad (1)$$

The required Spot placement $x_{l,m,i} \in \{0,1\}$ of a BF contains a subscript l. When $x_{l,m,i} = x_{k,m,i}$ is satisfied, the instances represented by $x_{l,m,i}$ and $x_{k,m,i}$ are identical. The constraints are denoted in Eq. (2)–(5). Equations (2) and (3) are the conditions under which the requirements to the total vCPUs or RAM of the BFs in a Chained-BF do not exceed the allowable amount at each Spot. Equation (4) is the condition that the bandwidth requirements of a Chained-BF do not exceed the allowable bandwidth of the Chains between Spots. The first term of the left-hand side represents the sum of the bandwidth requirements $x_{l,m,i} \cdot x_{l,n,j} \cdot A^N_{m,n} \cdot Q^{band}_{m,n}$ between BF m and BF n for all BFs. The second term represents the sum of the bandwidth requirements $x_{l,m,i} \cdot A^G_{n-|F|,j} \cdot A^N_{m,n} \cdot Q^{band}_{m,n}$ between BF m and GN n for all BFs and GNs. The right-hand side is the maximum bandwidth $B_{i,j}$ minus the reserved bandwidth $B^{used}_{i,j}$ for the link between Spot i and Spot j. Equation (5) is the condition for each BF in a Chained-BF to satisfy the redundancy level. Based on these objective functions and constraints, the Orchestrator calculates the BF placement to each Spot. As denoted in Eq. (4), the bandwidth requirement can be set for each Chain in a Chained-BF. When a public network and a private network coexist between Spots, a Chain which has a bandwidth requirement is automatically assigned to the private network if the available bandwidth of the corresponding link of the public network is set to 0. The same model can be used without changing the objective function or constraint equations.

$$\sum_{m \in F,l} x_{l,m,i} \cdot F^{cpu}_m \cdot \frac{F^{dup}_m}{\prod F^{dup}} \le C_i - C^{used}_i$$

$$\forall i \in S \tag{2}$$

$$\sum_{m \in F,l} x_{l,m,i} \cdot F_m^{ram} \cdot \frac{F_m^{dup}}{\prod F^{dup}} \leq M_i - M_i^{used} \quad \forall i \in S \tag{3}$$

$$\sum_{\substack{m,n \in N \\ :m<n<|F|}} x_{l,m,i} \cdot x_{l,n,j} \cdot A_{m,n}^N \cdot Q_{m,n}^{band} + \sum_{\substack{m,n \in N \\ :m<|F| \leq n}} x_{l,m,i} \cdot A_{n-|F|,j}^G \cdot A_{m,n}^N \cdot Q_{m,n}^{band} \leq B_{i,j} - B_{i,j}^{used} \quad \forall (i,j) \in L, \ \forall l \tag{4}$$

$$\sum_{i \in S} x_{l,m,i} = 1 \quad \forall m \in F, \ \forall l \tag{5}$$

6. Implementation

6.1 Mechanism for Collecting Resource Information

We implement the resource collecting sequences shown in Fig. 7 and the modules shown in Fig. 6: the Network Monitor, the Node Monitor, the Spot Manager, and the Zone Manager.

6.1.1 Network/Node Monitor

The Network Monitor and the Node Monitor in a Spot are implemented as a single program. After requesting a connection to the Spot Manager, the Network Monitor or the Node Monitor publishes network resource information or computing resource information to the Spot Manager.

6.1.2 Spot Manager

The Network Resource Manager and the Computing Resource Manager in the Spot Manager are implemented as a single program. It waits for connection requests from the Network/Node Monitors and the Zone Manager. The Spot Manager plays the role of a broker: it aggregates resource information published by the Network/Node Monitors for each Spot and notifies the Zone Manager. The resource information is stored in Redis [21], one of KVSs (Key-Value Stores).

6.1.3 Zone Manager

The Network Resource Manager and the Computing Resource Manager in the Zone Manager are implemented as a single program. It requests a connection to the Spot Manager specified in the configuration file. After establishing a connection, it subscribes to network resource information and computing resource information to the Spot Manager. When the subscribed information is published by the Network/Node Monitor, the Zone Manager is finally notified.

6.2 Calculation Methods in Orchestrator

6.2.1 Optimal Method

An optimal method uses the objective function and constraints defined by Eqs. (1)–(5) to find a solution using a solver. We use Pyomo [22] as a modeling language and a library to create mathematical optimization models in Python. We use IBM ILOG CPLEX Optimizer 12.9.0 [23] as a solver.

6.2.2 Heuristic Method

A heuristic method combines a solver calculation with some algorithm. Algorithm 1 shows the pseudo code of the heuristic method. In order to prevent the calculation time from increasing depending on the constraint conditions, the objective function is set as a constant in the heuristic method. The calculation is terminated when one solution satisfying the constraint conditions is obtained. A near-optimal solution is obtained by iteratively running the solver while narrowing the constraints on the network delay using binary search. The maximum network delay *MAX* is calculated from the shape of the Chained-BF and the maximum delay between Spots.

Algorithm 1 Pseudo code for the heuristic method.

```
Ensure: Spot Position of GN x_{l,m,i} ∈ {0, 1}
Require: BF placement to Spot A^G_{n,i} ∈ {0, 1}
  upper ← MAX                          ▷ Calculate the maximum delay in advance.
  lower ← 0
  while upper − lower < k do
    delay ← (upper + lower)/2
    s ← solveModel(delay)              ▷ Treat the obj. func. as a const.
    if s is ok then
      result ← s                       ▷ If a solution is found, save it.
      upper ← delay                    ▷ Make constraints tighter.
    else
      lower ← delay                    ▷ Make constraints looser.
    end if
  end while
  if result = ∅ then
    return error
  else
    return result
  end if
```

7. Evaluations

7.1 Mechanism for Collecting Resource Information

Figure 8 shows the experimental environment built on the five NOCs (Network Operation Centers) of the WIDE Cloud [24], which is an inter-university VM cloud infrastructure. One VM (Virtual Machine) is placed in each of the five NOCs (Yagami, Nezu, Komatsu, Dojima, and Nara). Each NOC is regarded as a Spot in the BF-DDPP. The value of the network delay between NOCs amounts to half of the RTT (round-trip time). It is assumed that all links between Spots are built on a private network. The VMs at each Spot are running Ubuntu 18.04 LTS with two virtual CPU cores of QEMU and 2 GB of RAM. One Spot Manager, one Network Monitor, and four Node Monitors are installed at each Spot, and one Zone Manager is installed only at the central Spot (Dojima). It is assumed that the clocks of all VMs used for the measurement are synchronized.

7.1.1 Subscribe Request & Subscribe Response

In this evaluation, we show that Subscribe Request messages and Subscribe Response messages are exchanged in parallel for each Spot. The Zone Manager sends a Subscribe Request message to all Spot Managers (Time t_1). Each Spot Manager sends back a Subscribe Response message individually as the response. The Zone Manager receives the last Subscribe Response message (Time t_2). We define latency as the difference between t_1 and t_2. The result shows a latency of 21.2 ms, which is smaller than the total RTT between the Zone Manager and each Spot Manager. This indicates that the Subscribe Request messages and Subscribe Response messages were exchanged in parallel for each Spot.

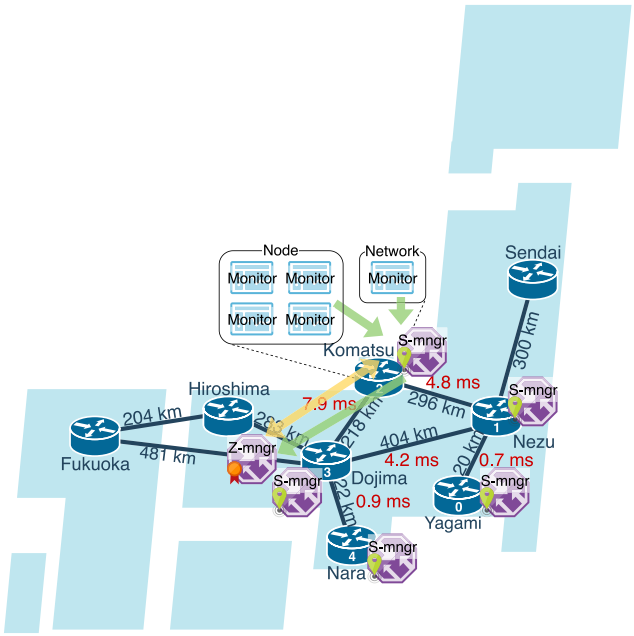


Fig. 8 Experimental environment using WIDE Cloud.

Table 2 Evaluation environment for the Orchestrator.

HV (Hypervisor)	
OS	VMware(R) ESXi(TM) 6.7.0
CPU	Intel(R) Xeon(TM) Gold 6248 2.50 GHz × 40
RAM	400 GB
VM (Virtual Machine)	
OS	Ubuntu Server 18.04 LTS
vCPU	Intel(R) Xeon(TM) Gold 6248 2.50 GHz × 32
RAM	16 GB

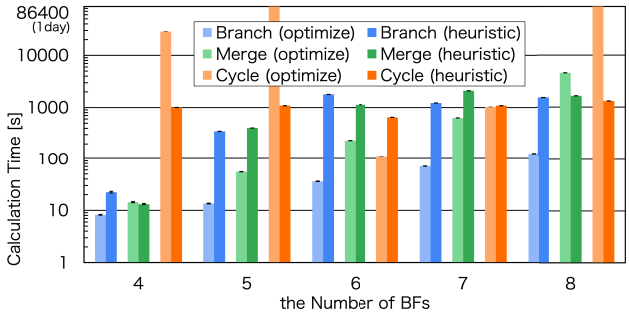


Fig. 9 Calculation time of the Orchestrator as changing the number of BFs and the shape of a Chained-BF.

Therefore, the latency is bound by the maximum RTT between the Zone Manager and the Spot Manager. As shown in Fig. 8, the maximum RTT is 15.8 ms between Dojima and Komatsu. The remaining 5.4 ms is considered to be the processing time in the Spot Manager and the Zone Manager.

7.1.2 Publish Request & Subscribe Notification

In this evaluation, we show that Publish Request messages and Subscribe Notification messages are forwarded in parallel for each Node. All Node Monitors simultaneously send a Publish Request message to the Spot Manager of the Spot to which they belong (Time t_1). Each Spot Manager sends back a Publish Response message individually as the response, and then sends a Subscribe Notification message to the Zone Manager. The Zone Manager receives the last Subscribe Notification message (Time t_2). We define latency as the difference between t_1 and t_2. The result shows a latency of 8.9 ms, which is smaller than the total delay between the Zone Manager and each Node Monitor. This indicates that Publish Request messages and Subscribe Notification messages were forwarded in parallel for each Node. Therefore, the latency is bound by the maximum delay between the Zone Manager and the Node Monitor. As shown in Fig. 8, the maximum delay is 7.9 ms between Dojima and Komatsu. The remaining 1.0 ms is considered to be the processing time in the Node Monitor, the Spot Manager, and the Zone Manager.

7.2 Calculation Time: Placement of BFs to Spots

We evaluated the time taken by the Orchestrator to calculate the placement of BFs and the results of the placement decision comparing the optimal method with the heuristic method. We use a part of the PoP level network topology [25] published by AT&T as the evaluation environment. We considered 18 PoPs distributed across the U.S. as Spots in the BF-DDPP. As the evaluation environment, we used VMs on the Hypervisor as shown in **Table 2**.

We measured the calculation time of the optimal and the heuristic methods for three different shapes of Chained-BFs shown in Fig. 2. For each shape, the number of BFs is varied from 4 to 8. At the beginning of each calculation, computing and network resources of each Spot are not used at all. **Figure 9** shows the results. Note that the calculation time in Fig. 9 shows only the time from when a Chained-BF configuration is entered into the Orchestrator to the time when target Nodes are derived, and does not include the deployment time to each Node. In the optimal method, it seems that the calculation time is highly dependent on the shape of the Chained-BF. The constraint conditions are extremely relaxed in the optimal calculation of cycle-type Chained-BF placement. This is because a cycle-type Chained-BF specifies only a single Gate Node as both an ingress and an egress of the chain while other types specify two or more Gate Nodes as an ingress and an egress, respectively. Under extremely relaxed constraints, the number of candidates satisfying the conditions increases. It leads to an increase of the number of times to find which one is optimal. As a result, the calculation time increases. In other words, tightening the constraints leads to a reduction in the calculation time. The reason why the calculation time of the cycle-type increases is that the constrains are extremely relaxed.

Increasing the number of BFs works to increase the calculation time and tighten the constraints regardless of the shape of the Chained-BF. In the optimal calculation, MANO checks whether the placement of a pair of BFs on each pair of Spots satisfies the requirement for every pair of BFs in a Chained-BF. As the number of BFs increases, the number of BF pairs that must be checked to see if they satisfy the bandwidth and delay requirements also increases. Thus, an increase in the number of BFs contributes to an increase in the calculation time (Factor 1). This property is also true for other shapes as shown in Fig. 9. In addition, an increase in the number of BFs also leads to tighter constraints since the number of candidates satisfying the conditions decreases (Factor 2).

As described above, increasing the number of BFs in a cycle-type Chained-BF results in both an increase in calculation time

Table 3 End-to-End delay ratio [heuristic]/[optimal].

No. BFs	4	5	6	7	8
Branch-type	1.5	1.0	1.1	0.8	1.1
Merge-type	1.0	1.0	1.0	1.1	1.5

due to Factor 1 and a decrease in calculation time due to Factor 2. When the number of BFs was increased from 5 to 6, the calculation time to optimally place a cycle-type Chained-BF decreased because it seemed that Factor 2 had a larger impact than Factor 1. On the other hand, when the number of BFs was increased from 6 to 8, the calculation time increased because it seemed that Factor 1 had a larger impact than Factor 2.

In contrast, in the heuristic method, the calculation time does not change much as the number of BFs increases more than 6. When the number of BFs composing a branch-type Chained-BF increases from 6 to 7 in the heuristic method, the calculation time decreases from 1,795 seconds to 1,226 seconds. The reason why the calculation time does not change much as the number of BFs increases is that the heuristic method sets a certain timeout during the repeated calculations by the solver. The calculation time with the solver increases when the constraints are extremely relaxed. In the heuristic method, the calculation with the solver times out when the calculation time becomes too long due to extremely relaxed constrains. The constrains are updated to more strict ones and the solver is called again after the timeout. In most cases, the optimal method completes the calculations in less time than the heuristic method, but the placement calculation time tends to increase gradually as the number of BFs increases.

Next, we calculated the ratio of the end-to-end network delay using the results of the BF placement. **Table 3** shows the results. The values in the table are calculated in such a way that the network delay with the heuristic method is divided by that with the optimal method except for the cycle-type case because some calculations are not completed due to extremely relaxed constrains. The delay in the heuristic method is approximately 1.0 to 1.5 times longer than that in the optimal method. The ratio value is 0.8 when the number of BFs is 7 in the branch-type. It seems that the solver is stuck in a local optimal solution in the optimal method because the model is nonlinear.

7.3 Applying Calculation Methods

In real-world usage scenarios, it is assumed that an operator of a Zone decides which calculation method to apply according to its operational policy. For example, we illustrate two patterns. In the first pattern, a user specifies the calculation method at the same time when the user deploys a Chained-BF. This is because it is assumed that some users want to optimize the placement even if it takes a long time while others want to get the calculation result as soon as possible. In the second pattern, the decision is made agnostic to users by a Zone operator. From a given Chained-BF deployment request, the MANO programmatically applies either the optimal placement calculation or the heuristic calculation method based on the shape and the number of BFs.

8. Conclusion

With the spread of edge/fog computing, it has become more important to place functions of services in appropriate locations. This paper assumes that a service is composed of one or more simple functions called BFs. In the era of edge/fog computing, it is increasingly important for network providers to consider not only computing resources but also network resources when deploying BFs in geographically distributed locations. This paper proposed a MANO architecture and placement calculation methods for a BF-based distributed data processing platform (BF-DDPP). We defined the BF-DDPP with a mathematical model and proposed an optimal placement calculation method and a heuristic placement calculation method. In the MANO architecture, we proposed a resource information collection mechanism divided into two levels: Spot and Zone. The evaluation results showed that the mechanism for collecting resource information had a sufficiently small overhead in terms of execution time. The calculation time for deploying four BFs with branch-type or merge-type was small enough, about 10 seconds with the optimal method and about 20 seconds with the heuristic method. With the heuristic method, the placement calculation time did not increase regardless of the shape of Chained-BF when the number of BFs exceeded 6. In most cases, the optimal method completed the calculations in less time than the heuristic method, but the placement calculation time tended to increase slowly as the number of BFs increases. Future work includes reducing the calculation time of cycle-type Chained-BF and evaluating the performance of MANO in a more realistic environment. For more realistic evaluations, we can change the load on the Orchestrator, such as the number of requests per second for launching Chained-BF. We will also evaluate the calculation time of MANO and the utilization dynamics of platform resources by giving the lifetime of a Chained-BF.

References

[1] Halpern, J. and Pignataro, C.: Service Function Chaining (SFC) Architecture, RFC 7665 (2015).

[2] Cerny, T., Donahoo, M.J. and Trnka, M.: Contextual Understanding of Microservice Architecture: Current and Future Directions, *SIGAPP Appl. Comput. Rev.*, Vol.17, No.4, pp.29–45 (2018).

[3] Tato, G., Bertier, M., Rivière, E. and Tedeschi, C.: ShareLatex on the Edge: Evaluation of the Hybrid Core/Edge Deployment of a Microservices-based Application, *MECC 2018 - 3rd Workshop on Middleware for Edge Clouds & Cloudlets*, pp.1–6 (2018) (online), available from ⟨https://hal.inria.fr/hal-01942807⟩.

[4] Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A. and Shen, X.: Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach, *IEEE Trans. Mobile Computing*, Vol.20, No.3, pp.939–951 (online), DOI: 10.1109/TMC.2019.2957804 (2021).

[5] Zhou, A., Wang, S., Wan, S. and Qi, L.: LMM: Latency-aware microservice mashup in mobile edge computing environment, *Neural Computing and Applications*, Vol.32, No.19, pp.15411–15425 (2020).

[6] 3GPP: 5G System; Principles and Guidelines for Services Definition; Stage 3, 3GPP TS 29.501, version 16.5.0 Release 16 (2020).

[7] ETSI: Network Functions Virtualisation (NFV); Management and orchestration, ETSI GS NFV-MAN 001, V1.1.1 (2014).

[8] ETSI: Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on architecture options to support multiple administrative domains, ETSI GS NFV-IFA 028, V3.1.1 (2018).

[9] Kukliński, S. and Tomaszewski, L.: DASMO: A Scalable Approach to Network Slices Management and Orchestration, *Proc. 2018 IEEE/IFIP Network Operations and Management Symposium* (*NOMS 2018*), pp.1–6 (2018).

[10] Sciancalepore, V., Giust, F., Samdanis, K. and Yousaf, Z.: A double-tier MEC-NFV Architecture: Design and Optimisation, *Proc. 2016 IEEE Conference on Standards for Communications and Networking*

(*CSCN*), pp.1–6 (2016).
[11] ETSI: Multi-access Edge Computing (MEC); Framework and Reference Architecture, GS MEC 003 V2.1.1, pp.1–21 (2019).
[12] ETSI: Mobile Edge Computing (MEC); Framework and Reference Architecture, GS MEC 003 V1.1.1, pp.1–18 (2016).
[13] Brogi, A., Forti, S. and Ibrahim, A.: How to Best Deploy Your Fog Applications, Probably, *Proc. 2017 IEEE 1st International Conference on Fog and Edge Computing* (*ICFEC*), pp.105–114 (2017).
[14] Giang, N.K., Lea, R., Blackstock, M. and Leung, V.C.M.: Fog at the Edge: Experiences Building an Edge Computing Platform, *Proc. 2018 IEEE International Conference on Edge Computing* (*EDGE*), pp.9–16 (2018).
[15] OpenJS Foundation: Node-RED, available from ⟨https://nodered.org⟩.
[16] Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A. and Shen, X.: Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach, *IEEE Trans. Mobile Computing*, Vol.20, No.3, pp.939–951 (online), DOI: 10.1109/TMC.2019.2957804 (2021).
[17] Ghribi, C., Mechtri, M., Soualah, O. and Zeghlache, D.: SFC Provisioning over NFV Enabled Clouds, *Proc. 2017 IEEE 10th International Conference on Cloud Computing* (*CLOUD*), pp.423–430 (2017).
[18] Rankothge, W., Ma, J., Le, F., Russo, A. and Lobo, J.: Towards making network function virtualization a cloud computing service, *Proc. 2015 IFIP/IEEE International Symposium on Integrated Network Management* (*IM*), pp.89–97 (2015).
[19] Alleg, A., Ahmed, T., Mosbah, M., Riggio, R. and Boutaba, R.: Delay-aware VNF Placement and Chaining based on a Flexible Resource Allocation Approach, *Proc. 2017 13th International Conference on Network and Service Management* (*CNSM*), pp.1–7 (2017).
[20] Luizelli, M.C., Bays, L.R., Buriol, L.S., Barcellos, M.P. and Gaspary, L.P.: Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions, *Proc. 2015 IFIP/IEEE International Symposium on Integrated Network Management* (*IM*), pp.98–106 (2015).
[21] Redis Labs: Redis, available from ⟨https://redis.io⟩.
[22] COIN-OR: Pyomo, available from ⟨http://www.pyomo.org⟩.
[23] IBM: ILOG CPLEX Optimization Studio, available from ⟨https://www.ibm.com/products/ilog-cplex-optimization-studio⟩.
[24] WIDE Project: WIDE, available from ⟨http://www.wide.ad.jp⟩.
[25] Ciavattone, L., Morton, A. and Ramachandran, G.: Standardized active measurements on a tier 1 IP backbone, *IEEE Communications Magazine*, Vol.41, No.6, pp.90–97 (2003).

Editor's Recommendation

This paper extended AFC (Application Function Chaining) for mixing computing infrastructure provided by various companies. The authors formulate an optimization problem that minimizes the network latency due to the concatenation of application processing, with redundancy and network bandwidth as constraints. And they proposed a framework to manage the entire infrastructure. In addition, this paper showed its usefulness with experiments in an actual network environment. The paper gives insights to readers in this research field and thus is selected as a recommended paper.

(Chief examiner of SIGDPS Atsushi Tagami)

Hiroki Watanabe is a Ph.D. student and a project researcher at Graduate School of Science and Technology, Keio University. He received a master degree in engineering from Keio University in 2018. His research interest covers future Internet architecture, protocol layering, network modeling, and edge computing. He is a member of IEEE, ACM, IPSJ, and IEICE.

Kazuki Hayashi received a master degree in engineering from Keio University in 2020. His research interest covers cellular network protocol, Internet architecture, and network slicing.

Tomonori Sato received a master degree in engineering from Keio University in 2021. His research interest covers edge computing and application platforms.

Takao Kondo is an assistant professor in Computer Security Incident Response Team and a researcher in Cyber Security Research Center, Keio University. He received master degrees in engineering and medicine from Keio University, in 2015 and 2016, respectively. From 2013 to 2017, he was a project researcher in Graduate School of Science and Technology, Keio University. From 2017 to 2020, he was an assistant professor in Information Technology Center, Keio University. His research interest covers cyber security (network traffic analysis, anomaly detection, authentication and authorization etc.) and Future Internet architecture. He is a researcher in Industrial Cyber Security Center of Excellence, IPA. He is a member of ACM, IEEE, IEICE, IPSJ and a board member of WIDE Project.

Fumio Teraoka received a master degree in electrical engineering and a Ph.D. in computer science from Keio University in 1984 and 1993, respectively. He joined Canon Inc. in 1984 and then moved to Sony Computer Science Labs., Inc. (Sony CSL) in 1988. Since April 2001, he is a professor of Faculty of Science and Technology, Keio University. He received the Takahashi Award of JSSST (Japan Society for Software Science and Technology) and the Motooka Award in 1991 and 1993, respectively. He also received the Best Paper Award in 2000 from IPSJ (Information Processing Society Japan). His research interest covers computer network, operating system, and distributed system. He was a board member of the WIDE Project from 1991 to 2010. He was a board member of IPSJ from 2000 to 2002. He was a board member of JSSST from 2005 to 2009. He is a member of ACM, IEEE, IPSJ, and IEICE.