Electronic Preprint for Journal of Information Processing Vol.22 No.3

Invited Paper

Modeling of Dynamic Latency Variations for Simulation Study of Large-scale Distributed Network Systems

Hiroshi Yamamoto[1,a) Katsuyuki Yamazaki[1]

Received: November 1, 2013, Accepted: February 8, 2014

Abstract: A realistic evaluation model that simulates a realistic variation in latency between a large number of Internet nodes is attracting attention for use in simulation studies on large-scale distributed networks (e.g., CDN & on-line gaming). Therefore, we are attempting to build a new evaluation model by applying time series analysis using an ARIMA model and a Euclidean embedding technique to a dataset obtained from a global scale measurement service for this study. The proposed evaluation model not only generates accurate time series data of the latency of each path on the Internet, but also avoids unrealistic behaviors related to the spatial distribution of latency (i.e., triangle inequality violation). Furthermore, the performance of a representative distributed latency management and prediction system is evaluated through computer simulations as a use case of the proposed evaluation model. The evaluation results helped us to clarify whether or not the proposed model can correctly disclose the impact of the dynamic latency variation on large-scale distributed systems. An example program of the proposed evaluation model is found in an Appendix of this manuscript.

Keywords: evaluation model, time series analysis, central point model, network coordinate system, computer simulation

1. Introduction

Network services using large-scale distributed systems such as cloud computing have recently been widely available over the Internet. For example, a content distribution network [1] has been constructed of a large number of intermediate servers dispersed over the Internet, and distributed on-line resources have begun to be used to provide on-line gaming [2] to many users. In these services, each user is assigned to a server selected from a resource pool, and should be persistently served by that selected one because the state of a user on the service cannot be easily moved to the others. However, the network condition (especially, network latency) between the user and the selected server can dynamically change due to several factors (e.g., network congestion & mobility). For example, in the mobile environment, data transmission route between end-computers can be frequently changed, because a mobile node connects to a different access network (i.e., router) than before when handover or offloading occurs. As a result, the end-to-end latency on the Internet will also fluctuate in the mobile network. Therefore, the resource selection strategy in a large-scale system needs to be carefully considered by estimating the future quality of user's experiences when using the services.

Many researchers have performed simulation studies for evaluating their proposed systems or methods on the large-scale distributed network in order to improve the performance of their distributed system. However, the existing simulation studies have not used a realistic network where the network latency changes over time, and therefore, a fixed network environment has been assumed as the simulation model. Furthermore, several existing researches have focused on modeling the dynamic characteristics of the Internet [3], [4], [5], but have focused on analyzing only some fixed paths.

On the other hand, a type of Euclidean embedding that maps the positional relationship between Internet nodes onto a multi-dimensional geometric space in order to catch the spatial characteristics of a large-scale system has been studied by many researchers [6], [7], [8], [9]. The Euclidean embedding technique has been used for managing the network latency between a large number of Internet nodes and for predicting the unknown latency of paths that have not thus far been measurement targets. However, this technique has not been considered as a way to express the dynamics of a large-scale distributed network.

Therefore, in this study, we attempt to build a realistic network model that provides simulation studies of a large-scale distributed network with dynamic network latency characteristics between a large number of Internet nodes. For our proposed evaluation model, a mathematical model that can precisely represent the time variation of end-to-end latency is established by applying a time series analysis using an ARIMA (Auto-Regressive Integrated Moving Average) model [10] to the measurement results of a long-term latency observation. In addition, a Euclidean embedding technique is applied to the proposed model in order to express the latency variation between the nodes dispersed over the Internet and avoid simulating abnormal situations (i.e., too many triangle inequality violations of latency). Furthermore, a computer simulation was conducted to evaluate the performance of a representative distributed latency management and prediction system as a use case for the proposed model.

1 Nagaoka University of Technology, Nagaoka, Niigata 940–2188, Japan
a) hiroyama@nagaokaut.ac.jp

The rest of this paper is organized as follows. In Section 2, the existing studies related with the modeling of a dynamic network condition are introduced. Sections 3 and 4 propose a new method of modeling the temporal variation of network latency and the spatial distribution of Internet nodes in large-scale distributed networks. In Section 5, the practicality of the proposed model is evaluated using an example of a distributed system through computer simulation. Finally, the conclusion is presented in Section 6, and an example program of the proposed evaluation model can be found in an Appendix.

2. Related Works

Many researchers have evaluated the impact of network latency on the performance of their services in order to estimate the quality of user's experience using a large-scale distributed service. For example, the quality of on-line gaming is very sensitive to network latency, so an acceptable level of latency for comfortable gaming has been analyzed [11], [12]. In these researches, an average or a median value of the latencies between nodes (clients or servers) has mainly been considered, but the impact of the dynamic change in the latencies on the performance has not been evaluated. Furthermore, the researchers have focused on only some fixed paths that were extracted from the network topology of a distributed service. However, knowledge of the dynamic characteristics of the entire network topology is necessary so that the engineering method such as the resource selection can be studied through computer simulation.

Therefore, this study attempts to build a new evaluation model where both the time variation and spatial distribution of the network latencies can be represented for a large-scale simulation of the distributed network. The existing studies related with the modeling are introduced in the following parts of this section.

2.1 Modeling of Temporal Latency Variation

Some existing researches have analyzed a network latency measurement dataset, and have identified a type of distribution from which the realistic network latencies can be generated. For example, it has been concluded that the network latency can be approximated by using a Gamma-like distribution [13], [14]. This distribution unveils the statistical characteristics of the temporal latency variation, but does not include any knowledge of the time series of the latency.

A time series analysis technique was used to analyze the measurement dataset to model the time series data. In the existing studies, several techniques (e.g., ARIMA (Auto-Regressive Integrated Moving Average) model [10], Brownian motion [15]) have been used in the time series analysis. The ARIMA model has particularly been used for modeling the variations in the amount of data traffic on the Internet and for forecasting the future values in the time series [3], [4], [5]. A realistic latency dataset that includes a self similarity characteristic can be generated by using the ARIMA model [16].

The ARIMA-based modeling in the existing researches can be used to generate only a dataset where the latency of each path changes independent of the other paths, but the generated dataset includes some unrealistic conditions. For example, if a node experiences an increase in network latency, the other nodes close to it should receive almost the same experience. Therefore, a modeling method should be carefully designed so that it accurately expresses this spatial relationship of latency variation between network paths.

2.2 Modeling of Spatial Latency Distribution

As explained in Section 2.1, the spatial relationship of latency between the Internet nodes should be accurately modeled in order to generate a realistic dataset of latency variations. Here, a Euclidean embedding that maps a matrix of network latencies between Internet nodes onto a multi-dimensional Euclidean space as illustrated in **Fig. 1** is a candidate technology [6]. The Euclidean space can be used to express the positional relationship of the Internet nodes because the physical entities of the nodes (i.e., client terminals & servers) are placed at physical places on the Earth, and therefore, the network latency is roughly proportional to physical distance between nodes. By appropriately embedding the nodes on the Euclidean space, the network latency can be derived as the virtual distance between the coordinates of end-nodes on the space.

Here, a network coordinate system (NCS) is a well-known algorithm for Euclidean embedding [7]. In the NCS, each node measures the network latency to the others, and the coordinates of the nodes on a multi-dimensional geometric space are optimized to minimize the difference between the actual latency and a virtual distance on the space. There can be two types of NCSs, landmark- and simulation-based systems, based on the embedding algorithm.

In the landmark-based system, the actual latencies among a small number of representative nodes (or landmarks) are measured first, and then their optimal coordinates are determined [8]. After that, the other nodes measure their latency to the predetermined landmarks, and calculate their own coordinates from the latencies and the coordinates of the landmarks according to the triangulation. In contrast, the simulation-based NCS adopts a completely distributed algorithm [9]. The simulation-based system does not assume any centralized point, but analyzes the network latencies of randomly selected paths on the distributed network.

Euclidean embedding has mainly been used to manage the network latency in a static environment, but has not yet been applied to expressing the temporal latency variation in a large-scale distributed system.

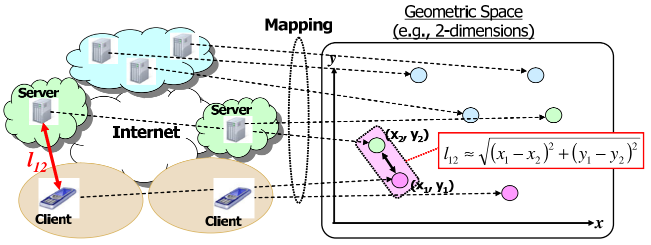

Fig. 1 Overview of Euclidean embedding.

3. Proposed Latency Variation Model

We discuss a mathematical model that can precisely represent the time variation of the end-to-end latency on the many paths used to construct a large-scale network topology in this section. A time series analysis using an ARIMA model is applied to the measurement results of a long-term latency observation in order to create a realistic model.

3.1 Targeted Dataset

Many kinds of measurement services for various network performances (e.g., throughput, latency, packet loss rate) have already been provided through the World Wide Web (WWW). Net Index [17] summarizes the measurement results continuously captured by global-scale measurement services (Speedtest.net [18] and Pingtest.net [19]), and distributes the dataset, including the statistics of the performance measures to the network researchers. In particular, the Pingtest.net measures the latency between a user terminal and a server set up by the service provider. The measurement results are summarized for the users contracting with the same Internet Service Provider (ISP), and the dataset includes the average latencies per day of each ISP during the period from October 2009 to May 2012.

In the following section, we extract the average latencies corresponding to 643 ISPs operated in the 74 countries from the dataset. The dataset contains only the daily averages of the end-to-end latencies on the Internet, and therefore, we attempt to clarify the fundamental characteristic of the temporal latency variation on a day-by-day basis. Note that the proposed analytical method does not depend on unit time of the dataset, and hence can be adapted to the other datasets that record hourly or minutely average of latencies.

3.2 Analysis Policy of Latency Variation

Figure 2 shows an example of the variation in the round trip time (RTT) corresponding to four ISPs in the United States. As illustrated in this figure, the latency between a client and a server dynamically changes day by day. Furthermore, **Fig. 3** depicts a histogram of the RTT. As mentioned in Ref. [13], the time variation of the end-to-end delay of many fixed paths produces a Gamma-like shape due to the self-similarity of the WAN traffic. However, as shown in Fig. 3, the long-term latency variation does not show the actual shape and includes several peaks. This is because several path variations may appear during the measurement interval and the number of intermediate routers may change several times.

On the other hand, Fig. 2 shows an important characteristic in which the latency is continuously increasing and gradually decreasing over time. The reason for this characteristic is related to the occurrences of network congestion on the Internet. Once the WAN traffic starts to concentrate on some specific part of the Internet, network congestion continuously worsens. The concentration of traffic gradually dissipates after the degree of the congestion peak. This characteristic should be applied to the time variation model of the latency.

We attempt to model the above-mentioned characteristics of the latency variation by using an ARIMA model in this research. This model is generally referred to as an ARIMA(p, d, q) model, where the parameters p, d, and q represent the order of the autoregressive, integrated, and moving average, respectively. When the time series data y_t, where t is an integer index, is analyzed, the ARIMA(p, d, q) model is given by the following equation.

$$\begin{aligned}\Delta^d y_t &= m + \phi_1 \Delta^d y_{t-1} + \cdots + \phi_p \Delta^d y_{t-p} \\ &\quad + a_t + \theta_1 a_{t-1} + \cdots + \theta_q a_{t-q}. \qquad (1)\end{aligned}$$

where a_t is the white noise at time t and m is the mean value of the time series data. In addition, Δ^d is an operator that represents the d-th order difference. For example, $\Delta^1 y_t$ and $\Delta^2 y_t$ are $y_t - y_{t-1}$ and $\Delta^1 y_t - \Delta^1 y_{t-1}$, respectively. The difference process is used to transform the linear non-stationary time series into a stationary time series. The p-th order autoregressive part and the q-th order moving average of the white noise are used to model the stationary time series data.

3.3 Decision of Order Parameters of ARIMA model

A realistic dataset that represents the temporal variation of the latency can be generated and used for the simulation by appropriately modeling the time series of the latency using the ARIMA model. Therefore, we first attempt to decide the appropriate parameters p, d, and q for the model. In this research, the three parameter patterns (1, 0, 1), (2, 0, 1), and (2, 0, 2) are considered in order to decide which is the best one. We found that a long-term average of latencies does not change with time by analyzing measurement results of many paths (ISPs), hence assume that the measurement results of the end-to-end latency are stationary time series (i.e., a parameter d is always set to zero). Furthermore, if the parameters p and q are set to large values, the ARIMA model should treat a large number of coefficients ($\phi_1, \cdots, \phi_p, \theta_1, \cdots, \theta_q$)

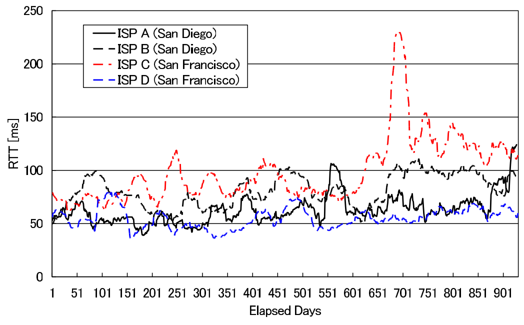

Fig. 2 Time variation of daily average of latency.

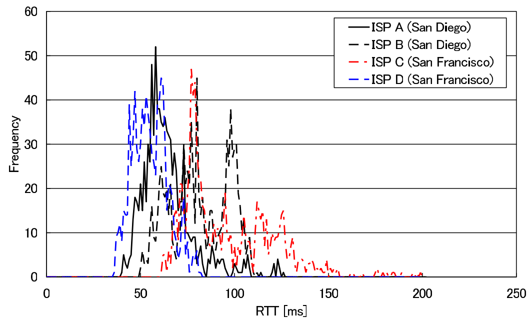

Fig. 3 Histogram of time variation of average latency.

to generate the time series data of each path. As a result, a computational load becomes so heavy for building a simulation model which decides a set of coefficients of each path by analyzing the relationship between the coefficients. Therefore, the parameter patterns with relatively small p and q have been considered as candidates.

For each parameter set, the time series data corresponding to each path is analyzed by using R-tool [20], and the coefficients $(\phi_1, \cdots, \phi_p, \theta_1, \cdots, \theta_q)$ in Eq. (1) and the standard deviation of white noises (a_t) are calculated by using a function *arima*() of the tool. After that, the time series data is generated from the ARIMA model with the coefficients by using a function *arima.sim*() of the same tool.

First, **Fig. 4** illustrates the time variation of both the actual latency corresponding to one path and the latency estimated from the ARIMA model. As shown in this figure, the time series data generated from the three ARIMA models catch the characteristics of the measurement results. In order to determine which one is the best model, **Figs. 5** and **6** depict the average and standard deviation of the latencies on 643 paths, respectively. The statistics (i.e., average or standard deviation) of both the actual RTTs and the estimated ones corresponding to the same path are plotted on the same x-axis in these figures, and the plots of the paths are sorted in increasing order of the statistic related to the actual latency. Therefore, a smaller "Rank" is assigned to a path with smaller actual latency statistics. As shown in these figures, the ARIMA model using any parameter achieves more accurately estimates the latency. In this study, we have decided to use the parameter $p = 2$ and $q = 1$ (or ARIMA(2, 0, 1) model (Eq. (2))) to estimate the time series of the latencies on the Internet because the model appears to accurately catch the steep change in actual latency.

$$y_t = m + \phi_1 y_{t-1} + \phi_2 y_{t-2} + a_t + \theta a_{t-1}. \qquad (2)$$

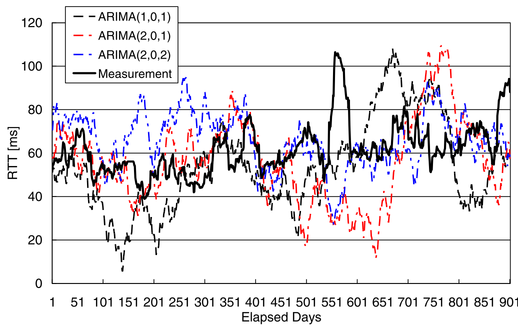

Fig. 4 Example of temporal latency variation.

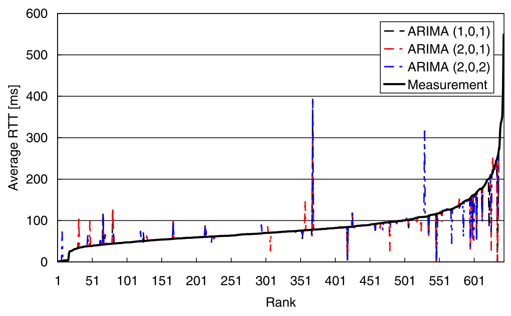

Fig. 5 Comparison of average latency in ARIMA model.

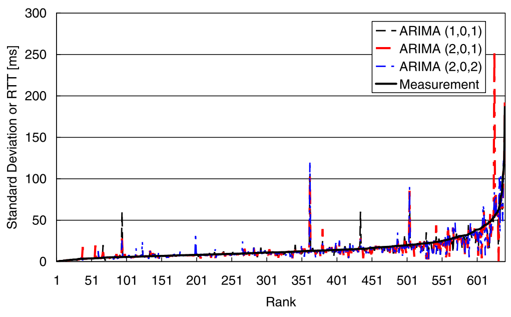

Fig. 6 Comparison of standard deviation in ARIMA model.

3.4 Construction of Simulation Model Using ARIMA Model

In order to build a realistic evaluation model where the network latency of all the paths changes day by day, the appropriate coefficients (i.e., ϕ_1, ϕ_2, and θ) in Eq. (2) and the standard deviation of white noise (a_t) should be decided for each path. Therefore, we attempt to establish a method for deciding these parameters by analyzing the dataset obtained from the Net Index.

First, the parameters of each path are estimated from the dataset of 643 paths over 74 countries by using the R-tool, and the correlation coefficients between ϕ_1 and the other parameters are evaluated. As listed in **Table 1**, ϕ_1 highly correlates with ϕ_2 and θ, therefore the ϕ_2 and θ of each path can be automatically decided when ϕ_1 is selected. Therefore, it was concluded that the derivation method for only two parameters (i.e., ϕ_1 and standard deviation of a_t) should be established for building the evaluation model. The following approximation equations can be determined by using the least squares method.

$$\phi_2 = -0.955 \times \phi_1 + 0.914, \qquad (3)$$

$$\theta = -0.971 \times \phi_1 + 1.02. \qquad (4)$$

Next, **Fig. 7** shows the probability distribution function of ϕ_1 among the 643 paths. As shown in this figure, the function can be approximated by using a linear expression, and the approximation equation is as follows.

$$F(x) = 0.425x + 0.0351. \qquad (5)$$

Furthermore, **Fig. 8** illustrates the probability density function of the variance in white noise (a_t). This figure shows that the function approximately follows the exponential distribution.

Table 1 Correlation coefficient between ARIMA parameters.

ϕ_1 - ϕ_2	ϕ_1 - θ	ϕ_1 - variance of a_t
−0.996	−0.978	0.00112

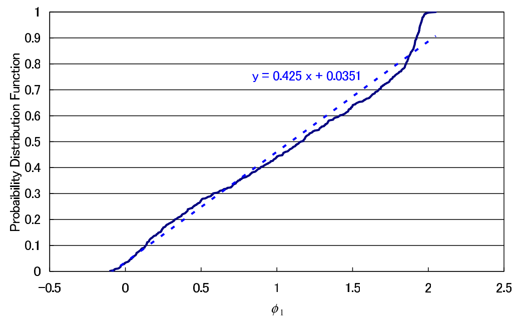

Fig. 7 Probability distribution function of ϕ_1.

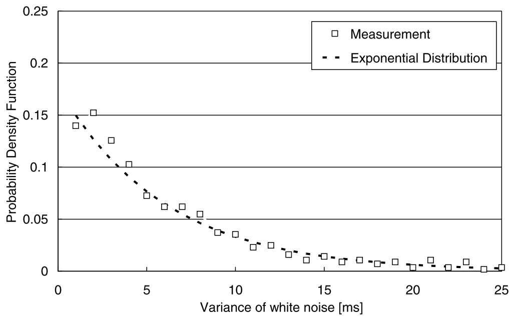

Fig. 8 Probability density function of variance of a_t.

$$f(x) = 0.177 \times e^{-0.168x}. \tag{6}$$

These equations have been derived by using the least squares method.

Finally, a method for deciding the parameters related to the latency variation on each path is summarized as follows.

(1) Two parameters (ϕ_1 and the variance of a_t) of each path are derived according to Eqs. (5) and (6).

(2) The other parameters (ϕ_2 and θ) are selected based on Eqs. (3) and (4).

(3) The time series data for white noise a_t is generated based on the variance.

In this study, the same parameter set ($p = 2$ and $q = 1$) of the ARIMA model can be applied to all paths, hence a set of coefficients (ϕ_1, ϕ_2, and θ_1) can be decided in a uniform manner as explained above. On the other hand, the accuracy of the estimated time series data may improve if each path adopts a different parameter set from others. However, a different derivation method of the coefficients should be considered for each parameter set, and a way to decide the parameter set of each path should also be proposed. Therefore, we adopt the simulation model using a uniform parameter set in order not to increase the computational complexity.

Furthermore, the proposed simulation model has been constructed based on the latency measurement dataset obtained from Net Index, but the construction method does not depend on a specific dataset. Therefore, if a dataset that includes time series data of latency between several end-nodes extracted from the target network could be prepared, the proposed method can generate the latency variation model that can be used for a large-scale simulation of the network.

4. Proposed Euclidean Embedding Method of Latency Variation

In this section, we start by explaining a problem that occurs when the time variation model proposed in Section 3 is simply applied to the network latencies of end-to-end paths in a large-scale distributed network. Furthermore, a new expression method for the latency variation that can resolve the problem by using the Euclidean embedding technique explained in Section 2.2 is proposed.

4.1 Problem with Simple Latency Variation Model

A simulation model where the network latency dynamically changes can be constructed by determining the parameters related to the ARIMA model of each path. However, in such a model where the latency of each path changes independently of the other paths, a realistic temporal variation of the network latency between Internet nodes cannot be expressed.

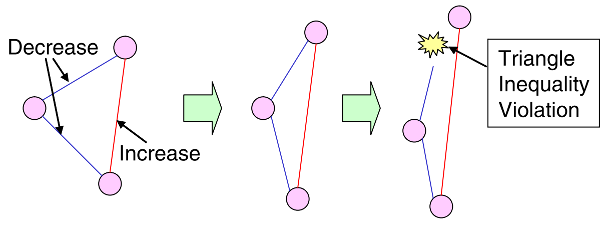

Fig. 9 Triangle inequality violation in simple evaluation model.

As explained in Section 2.2, the positional relationship between Internet nodes can be mapped onto a multi-dimensional geometric space. In the geometric space, the positional relationship among any three nodes should satisfy the triangle inequality, where the sum of the lengths of any two sides is greater than the length of the remaining side. However, if the latency of each path changes independently of the other paths, a triangle inequality violation frequently occurs on the evaluation model, as illustrated in **Fig. 9**.

Of course, there are some triangle inequality violations on the Internet, because the network topology cannot be completely mapped onto the Euclidean space. For example, the Internet nodes connect to the Internet through an Internet Service Provider (ISP), and the ISPs are complexly interconnected between each other according to their policies or contracts. Therefore, each node cannot transmit data packets towards the physical directions of the others, and the length of the communication path is distorted compared to the physical distance between nodes. However, if there are too many triangle inequality violations, the accuracy of the evaluation model, which should simulate characteristics of the Internet, degrades.

Therefore, in this study, we propose a new expression method for the realistic latency variation between Internet nodes that uses a Euclidean embedding technique. In the proposed method, the latency variation is expressed as a movement of the coordinates on the geometric space.

4.2 Decision of Initial Coordinates of Nodes

The proposed modeling method first decides on the initial coordinates of the nodes on a multi-dimensional geometric space by using the Euclidean embedding explained in Section 2.2 to express a realistic variation of the network latencies. The Euclidean embedding method analyzes the measurement results of the latencies between nodes, and decides on the optimal coordinates of the nodes to minimize the difference between the actual latency and the virtual distance on the geometric space.

In this study, we use a “King dataset” that contains the measurements results of the latencies between a set of DNS servers dispersed over the Internet [21]. The latencies are measured by using a King tool that estimates the RTT between two arbitrary DNS servers by using recursive DNS queries. An example of the King dataset that includes the latencies between all pairs of the 462 servers has been released from Ref. [22]. This dataset cap-

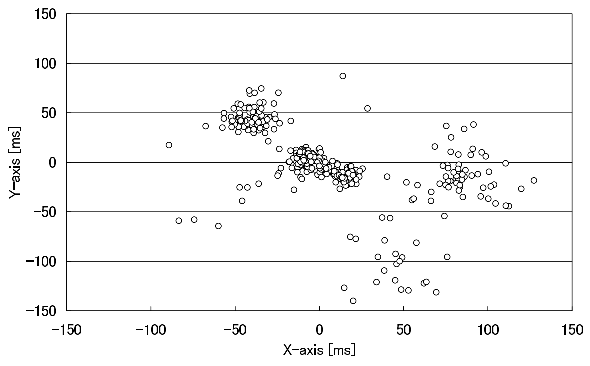

Fig. 10 Initial coordinates of nodes (King dataset).

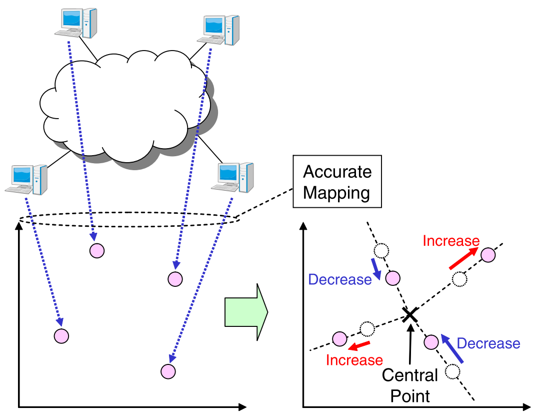

Fig. 11 Central point model of latency variation.

tures the realistic characteristics of the latencies on a large-scale distributed network, and therefore, has been used to evaluate the performance of the Euclidean embedding method by many researchers.

By applying the Euclidean embedding to the King dataset, the proposed modeling method decides the initial coordinates of the computers on the multi-dimensional geometric space, as shown in **Fig. 10**.

4.3 Central Point Model of Latency Variation

In order to represent the latency variation on the evaluation model, the coordinates of the nodes on the multi-dimensional Euclidean space should be adjusted whenever the network latency of each path is updated according to the ARIMA model, as proposed in Section 3. Therefore, in this study, we propose a new expression method where the latency variation derived from the ARIMA model can be easily translated to the movement of the coordinates on the geometric space.

In the proposed method, a center point coordinate of all the nodes is defined as a central point of the geometric space, and the network latency between each node and the central point is derived based on the proposed ARIMA model at each time. The coordinate of each node moves in the direction or opposite direction of the central point so that the distance on the geometric space has the same value as the derived latency, as illustrated in **Fig. 11**.

As explained in Section 3.1, the Net Index dataset includes the measurement results of the latency between a user terminal and a server prepared by the service provider. The measurement server should be located on the central point of the Internet in order to provide the measurement services to all the Internet nodes. Therefore, the assumption of the central point of the geometric space is reasonable for modeling the latency variation. By using the central point model, the latency variation can be expressed on the multi-dimensional geometric space, and therefore, the occurrence of triangle inequality violations in the evaluation model can be avoided.

Finally, we summarize the proposed model for expressing the latency variation on a multi-dimensional geometric space as follows.

(1) The initial coordinates of the nodes are decided by applying the Euclidean embedding technique to the latency dataset as mentioned in Section 4.2.
(2) A center point coordinate from all the nodes is calculated, and a distance between each node and the central point is defined as the initial latency of an ARIMA model.
(3) The parameters related with the ARIMA model of each edge between each node and the central point are selected as explained in Section 3.
(4) The time series data of the latency of the edges is generated according to the ARIMA model.

5. Performance Evaluation of Network Coordinate System Using Proposed Evaluation Model

In order to clearly evaluate the practicality of our proposed evaluation model, we should prepare a large amount dataset which includes long-term measurement results of latencies between all end-nodes on the target network. And then, simulations of a large-scale distributed system should be performed on both the dataset of the measured latency and that generated by the proposed model. However, such a large-scale and realistic dataset has not been distributed from any organization, and of course cannot be easily obtained. Therefore, we will confirm a simulation study for clarifying whether or not a main problem of the dynamic latency variation model (i.e., triangle inequality violation) can be solved by our proposed evaluation model. In this simulation, the network coordinate system (NCS) explained in Section 2.2 is used as an example of the distributed system because the performance (i.e., accuracy of the estimated latency) of the system is markedly affected by the existence of the triangle inequality violation. In particular, the necessity of the central point model proposed in Section 4 is unveiled by evaluating the performance of NCS on two types of evaluation models.

5.1 Network Coordinate System: Vivaldi

As described in Section 2.2, the NCS is a well-known Euclidean embedding method, but is attracting attention as a low-cost management and prediction system of network latencies on a large-scale distributed system. In the case of Euclidean embedding, the NCS should obtain the latency data on as many of the paths as possible, and should optimize the coordinates of the nodes to minimize the difference between an actual latency and a virtual distance on the space. However, even if only the measurement results of the latency on a limited number of paths are gathered, the coordinates of the nodes on the geometric space can be decided by using the NCS algorithm, and the unknown latency of the remaining path can be predicted by calculating the virtual distance between the coordinates of both ends of the path. Of course, the more measurement results can be obtained, the more accurate latency prediction can be achieved.

Vivaldi is a well-known NCS algorithm and can be easily used in a decentralized system [9]. Each node in the system transmits probe packets to pre-selected neighbor nodes in order to measure the latency and to obtain their coordinates on the geometric space. If the latency estimated from the coordinates is different from the actual one, the coordinate of the node on the geometric space moves toward an appropriate point so as to reduce the difference between the actual and estimated ones. The following equation represents the condition where node A adjusts its coordinate based on the measured latency between nodes A and B.

$$\begin{aligned}
&\vec{x}_A \leftarrow \vec{x}_A + c_c w_{AB}(\|\vec{x}_A - \vec{x}_B\| - l_{AB})u(\vec{x}_A - \vec{x}_B) \qquad (7)\\
&w_{AB} \leftarrow \frac{w_A}{w_A + w_B}\\
&\epsilon \leftarrow \frac{(\|\vec{x}_A - \vec{x}_B\| - l_{AB})}{l_{AB}}\\
&w_A \leftarrow c_e w_{AB}\epsilon + (1 - c_e w_{AB})w_A
\end{aligned}$$

where $\vec{x}_n$ is the coordinate vector of node n, l_{nm} is the measured latency between nodes n and m, $u(\vec{x}_n - \vec{x}_m)$ is the unit vector in the direction from node m to n, and w_n is the uncertainty of node n's coordinate. c_c and c_e are algorithmic constants, and the authors of Ref. [9] claimed that both parameters should be set to 0.25. By repeatedly adjusting the coordinates of the nodes as mentioned above, the coordinates will converge towards the accurate points, as shown in **Fig. 12**.

Furthermore, in the Vivaldi, the coordinate for a node has three dimensions (X, Y, Height) and the distance d_{ij} between nodes i and j is calculated by using the following equation.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + h_i + h_j. \qquad (8)$$

In this equation, (x_i, y_i, h_i) and (x_j, y_j, h_j) are the coordinates of nodes i and j. A virtual Height is used to represent the component

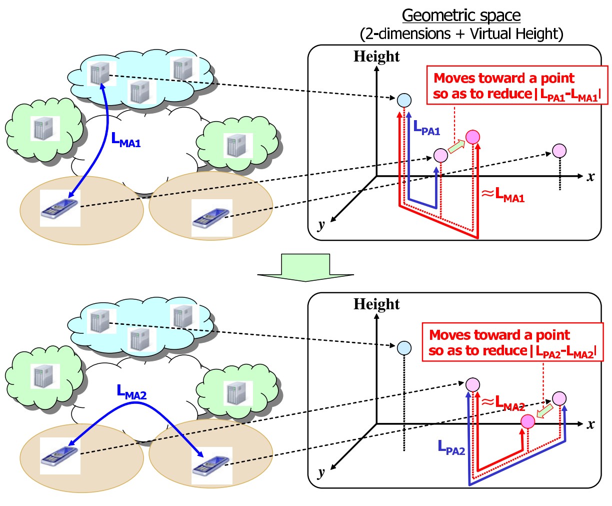
Geometric space
(2-dimensions + Virtual Height)
Height
Moves toward a point so as to reduce |LPA1-LMA1|
LMA1
LPA1
≈LMA1
x
y
Height
Moves toward a point so as to reduce |LPA2-LMA2|
LMA2
≈LMA2
LPA2
x
y

Fig. 12 Overview of coordinate adjustment in Vivaldi.

of latency that a node experiences in all its paths due to its Internet access link.

Many enhanced versions of the Vivaldi [23], [24] exist, but we use the standard Vivaldi in the following sections in order to evaluate the fundamental impacts of the temporal variation of latency of the proposed evaluation model on the prediction accuracy of the NCS.

5.2 Evaluation Model

For the evaluation of the performance of NCS, the initial latencies between the end-nodes on the distributed network were obtained from the "King dataset" that contains the measurement results of the latencies between a set of DNS servers dispersed over the Internet as mentioned in Section 4.2. By analyzing the dataset, the network latencies between all pairs of the 462 servers can be obtained. Two types of evaluation models were built for the simulation based on the latency dataset.

The first model is a *simple latency variation model*, where the latency between the end-nodes in the King dataset is directly changed day by day according to an ARIMA model. In this evaluation model, the parameters related with the ARIMA model are set to each pair of nodes, and the time series data for the latency of each path is generated independently of the others as explained in Section 3.4.

The second model is a *central point-based latency variation model*, which uses the proposed central point model. As explained in Section 4.3, this model first decides the initial coordinates of the end-nodes by using the Euclidean embedding technique on the King dataset. In this evaluation, the positional relationship between the nodes is expressed on a two dimensional + virtual height space, which is the same as the Vivaldi. After that, the latency between each node and the center point coordinate of all the nodes is dynamically changed by the ARIMA model. In the geometric space, the latency between nodes can be derived by calculating the virtual distance (Eq. (8)) between their coordinates.

By evaluating performance of the NCS on these two models, we can clarify the practicality of the proposed central point model explained in Section 4.

5.3 Performance Measure

We use a *Relative Error* that represents the accuracy of the predicted latency as a performance measure for the NCS. The relative error is calculated using the following equation.

$$Relative\ Error = \frac{|Latency_M - Latency_P|}{Latency_M}. \qquad (9)$$

where $Latency_P$ and $Latency_M$ indicate the predicted latency (i.e., virtual distance on the geometric space) and the actual one. The performance metric has been used by many existing studies on the NCS, and the Vivaldi algorithm adjusts the coordinates of the nodes based on the uncertainty parameter (w_n in Eq. (7)) derived by the equation. The relative error of each path is calculated, and the average relative error among all the paths is used as a performance measure.

5.4 Evaluation on Simple Latency Variation Model

The impact of the time variation of latency on the prediction of the Vivaldi is evaluated on a simple evaluation model where the network latency between the end-nodes changes according to the ARIMA model. In this evaluation, the number of neighbors for each node is set to 5 or 10% of all the nodes, and each node obtains the latency and coordinates for them by sending out probe packets in order to adjust their coordinates. Here, *Ratio* means the ratio of the neighbors that each node communicates with per day. For example, when the *Ratio* is set to 2, each node attempts to find the better coordinate by checking the positional relationship of all its neighbors twice each day.

Figures 13 and **14** show the average relative error of all the paths as a function of the elapsed days when the number of neighbors is set to 5 and 10% of all the nodes. In addition, the data plots with $Static$ show the simulation results of the static environment, where the latency between the nodes does not change from the initial state, and the data plots *Dynamic* indicate the performance on the dynamic latency variation model. When the elapsed days is 0, the average error rate becomes 1 because the initial coordinates of all the nodes are set to $(x_i, y_i, h_i) = (0, 0, 0)$, namely $Latency_p$ in Eq. (9) is 0.

As shown in Figures 13 and 14, in a static environment where the network latency does not change over time, the appropriate coordinates for the nodes can be determined, so the average error rate converges to about 0.2 when the number of neighbors is set to 10% of all the nodes. In addition, the prediction accuracy improves more rapidly with the increase in the number of adjustments of the coordinates for each day. The evaluation results in the static environment show this type of realistic behavior, but unrealistic situations in the dynamic environment still exist.

Even when the number of neighbors is set to large and the coordinates are frequently adjusted per day in a dynamic environment, the prediction accuracy gradually degrades. This is because the latency of each path between the end-nodes changes independent of the other paths, and therefore, too many triangle inequality violations occur in the evaluation model, as explained in Section 4.1. Therefore, it is concluded that the simple latency variation model cannot simulate a realistic variation in latency in a large-scale distributed network.

5.5 Evaluation on Central Point-based Latency Variation Model

Next, we evaluate the performance of the Vivaldi on the proposed center point-based latency variation model. As explained in Section 4, the proposed model avoids the occurrence of triangle inequality violations in the distributed network by mapping the positional relationship between Internet nodes onto a Euclidean space.

Figures 15 and **16** show the average relative error as a function of the elapsed days when the number of neighbors is set to 5 and 10% of all the nodes. As shown in these figures, the average error rate converges to a smaller value over time than that in the simple evaluation model. Furthermore, unlike in the simple evaluation model, the prediction accuracy of the Vivaldi stabilized by continuously adjusting the coordinates of nodes, and degradation of the prediction accuracy due to latency variations can be clearly confirmed. Therefore, it is concluded that the proposed Euclidean embedding technique helps to create a realistic evaluation model where the fundamental characteristic of the distributed network never changes even if the network latency is dynamic.

However, the prediction accuracy markedly improves in the

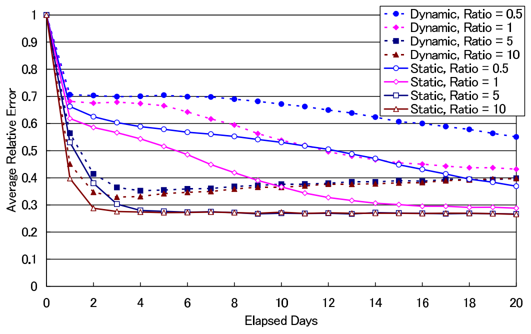

Fig. 13 Average relative error vs. elapsed days on simple latency variation model (Neighbor = 5%).

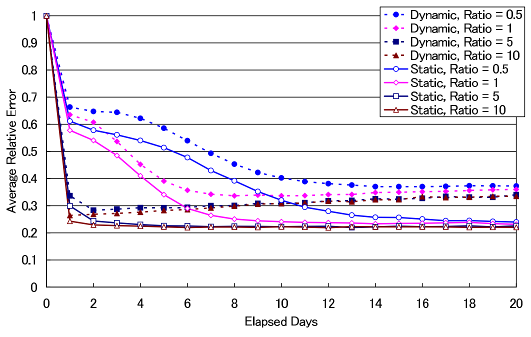

Fig. 14 Average relative error vs. elapsed days on simple latency variation model (Neighbor = 10%).

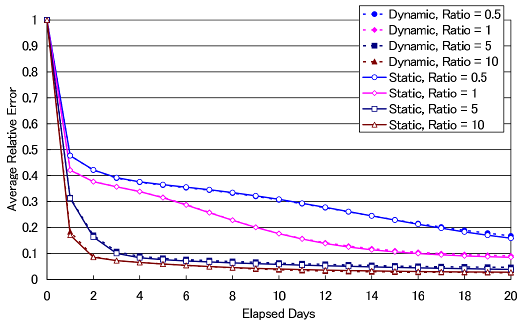

Fig. 15 Average relative error vs. elapsed days on central point-based latency variation model (Neighbor = 5%).

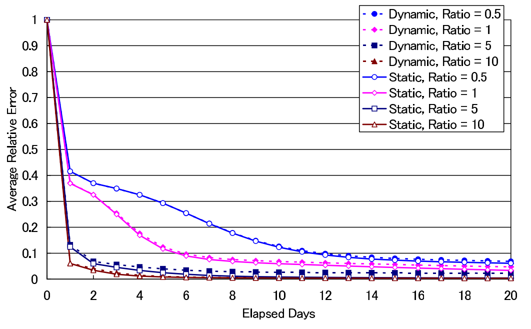

Fig. 16 Average relative error vs. elapsed days on central point-based latency variation model (Neighbor = 10%).

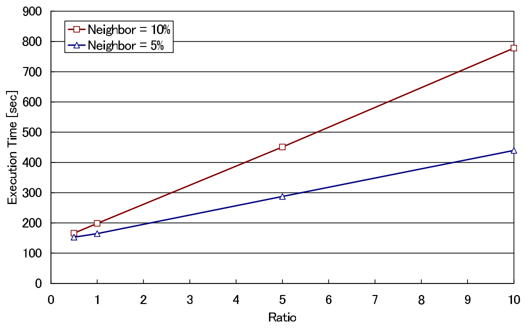

Fig. 17 Execution time of proposed evaluation model.

Table 2 Specifications of computer used for simulation.

CPU	Intel Core i3 2.4 GHz
Memory	4 GB
Operating system	Windows 7 Pro.

proposed model when compared to the simple evaluation model, because triangle inequality violations do not occur in the proposed model. In order to completely emulate a realistic network environment, a realistic amount of triangle inequality violations should be represented in the evaluation model. Improvement of the evaluation model will be considered in a future study.

Finally, **Fig. 17** shows the execution time of a simulation where the experiment of 100 days is repeated 100 times on the proposed evaluation model. The simulation program was written in Java with R Java Interface (JRI) [25], and was executed on the personal computer whose specifications are listed in **Table 2**. As shown in this figure, the simulation takes longer time as the Ratio and the number of neighbors increase, because it should simulate more times of coordinate adjustments of end-nodes. However, the simulation can be finished in 800 seconds, even when the *Ratio* and the number of neighbors are set to 10 and 10%. Therefore, we conclude that the proposed evaluation model helps researchers of a distributed system to finish evaluating their proposals in reasonable time.

6. Conclusions

We have modeled temporal latency variation by applying time series analysis using an ARIMA(2, 0, 1) model and a Euclidean embedding technique to a dataset obtained from a measurement service called Net Index in order to build a new evaluation model that is useful for evaluating the performance of a large-scale distributed system on a realistic dynamic network. In addition, the performance of a representative distributed latency management and prediction system, Vivaldi, has been evaluated through computer simulations consisting of 462 end-nodes as a use case of the proposed evaluation model. It has been concluded from the evaluation results that the proposed evaluation model can avoid 10% of unrealistic degradation of the prediction accuracy due to too many triangle inequality violations of latencies between Internet nodes, and therefore, can disclose the impact of the dynamic latency variation on the large-scale distributed system. Furthermore, the simulation where the experiment of 100 days is repeated 100 times on the proposed evaluation model has finished in only 800 seconds, even when the *Ratio* and the number of neighbors for each end-node are set to 10 and 10%. In the future, we will further improve our proposed evaluation model to help more realistically express the amount of triangle inequality violation in a dynamic network environment.

Acknowledgments This study was partly supported by JSPS KAKENHI Grant Number 25730055.

References

[1] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R. and Weihl, B.: Globally Distributed Content Delivery, *IEEE Internet Computing*, Vol.6, No.5, pp.50–58 (2002).

[2] Cloud Computing Solutions from NVIDIA, available from ⟨http://www.nvidia.com/object/cloud-computing.html⟩.

[3] Dhand, R., Lee, G. and Cole, G.: Communication Delay Modeling and its Impact on Real-Time Distributed Control Systems, *Proc. 4th International Conference on Advanced Engineering Computing and Applications in Science*, pp.39–46 (2010).

[4] Basu, S., Mukherjee, A. and Klivansky, S.: Time Series Models for Internet Traffic, *Proc. IEEE INFOCOM '96*, Vol.2, pp.611–620 (1996).

[5] Karagiannis, T., Molle, M. and Faloutsos, M.: Long-Range Dependence - Ten Years of Internet Traffic Modeling, *IEEE Internet Computing*, Vol.8, No.5 (2004).

[6] Abrahao, B. and Kleinberg, R.: On the Internet Delay Space Dimensionality, *Proc. IMC '08*, pp.157–168 (2008).

[7] Donnet, B., Gueye, B. and Kaafar, M.A.: A Survey on Network Coordinates Systems, Design, and Security, *IEEE Communications Surveys and Tutorials*, Vol.12, No.4 (2010).

[8] Ng, T.S.E. and Zhang, H.: Towards Global Network Positioning, *Proc. 1st ACM SIGCOMM Workshop on Internet Measurement*, pp.25–29 (2001).

[9] Dabek, F., Cox, R., Kaashoek, F. and Morris, R.: Vivaldi: A Decentralized Network Coordinate System, *Proc. ACM SIGCOMM 2004*, pp.15–26 (2004).

[10] Box, G.E.P. and Jenkins, G.M.: Time Series Analysis: Forecasting and Control, Holden-Day Inc. (1976).

[11] Claypool, M. and Claypool, K.: Latency and Player Actions in Online Games, *Comm. ACM - Entertainment Networking*, Vol.49, No.11 (2006).

[12] Armitage, G.: An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3, *Proc. 11th IEEE International Conference on Networks* (*ICOIN 2003*), pp.137–141 (2003).

[13] Bovy, C.J., Mertodimedjo, H.T., Hooghiemstra, G., et al.: Analysis of End-to-end Delay Measurements in Internet, *Proc. Passive and Active Measurement Workshop* (2002).

[14] Zhang, Y. and Duffield, N.: On the Constancy of Internet Path Properties, *Proc. 1st ACM SIGCOMM Workshop on Internet Measurement* (*IMW 2001*), pp.197–211 (2001).

[15] Norros, I.: On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks, *IEEE Journal on Selected Areas in Communications*, Vol.13, No.6, pp.953–962 (1995).

[16] Yamamoto, H. and Yamazaki, K.: Analysis of Temporal Latency Variation on Network Coordinate System for Host Selection in Large-Scale Distributed Network, *Proc. 37th IEEE Annual International Computers, Software and Applications Conference* (*COMPSAC 2013*), pp.604–609 (July 2013).

[17] Net Index, available from ⟨http://www.netindex.com⟩.

[18] Speedtest.net, available from ⟨http://www.speedtest.net⟩.

[19] Pingtest.net, available from ⟨http://www.pingtest.net⟩.

[20] The R Project, available from ⟨http://www.r-project.org/⟩.

[21] Gummadi, K.P., Saroiu, S. and Gribble, S.D.: King: Estimating Latency between Arbitrary Internet End Hosts, *Proc. SIGCOMM IMW 2002*, pp.5–18 (2002).

[22] King dataset, available from ⟨http://pdos.csail.mit.edu/p2psim/kingdata/⟩.

[23] Chen, Y., Xiong, Y., Shi, X., Zhu, J., Deng, B. and Li, X.: Pharos: Accurate and Decentralised Network Coordinate System, *IET Communications*, Vol.3, No.4, pp.539–548 (2009).

[24] Agarwal, S. and Lorch, J.R.: Matchmaking for Online Games and Other Latency-Sensitive P2P Systems, *Proc. ACM SIGCOMM 2009*, pp.315–326 (2009).

[25] JRI - Java/R Interface, available from ⟨http://www.rforge.net/JRI/⟩.

Appendix

A.1 Derivation of Time Series Data of Network Latency

Derivation process of time series data of network latency between each end-node and a center point coordinate from all the nodes is illustrated in **Fig. A·1**. In this figure, an example program is written in Java with Java R Interface (JRI), and *NumNodes* and *ElapsedDays* indicate the number of end-nodes in the evaluation model and a simulation interval.

The derivation process consists of three phases. In the first phase, a JRI engine is initialized so that functions of R-tool can be invoked from a Java program. This is shown in Fig. A·1, line 2. In the second phase, we decide parameters related with the ARIMA (2, 0, 1) model of each path between each end-node and the central point. These parameters are derived according to Eqs. (3), (4), (5), and (6), and then are set to the JRI engine as shown on line 8–17. Finally, time series data of network latency of each path can be obtained from an *arima.sim*() function of R-tool through the JRI engine on line 19. The result of the function is converted to a double array on line 20 in order for the Java program to be able to treat the time series data.

```
 1: // Initialization of JRI engine
 2: Rengine engine = new Rengine(new String[]{"--no-save"},false,null);
 3:
 4: // Time series data of each path between each node
    // and the central point is derived
 5: for(int i = 0; i < NumNodes; i++){
 6:
 7: // Parameters related with an ARIMA(2,0,1) model are decided
 8:   phi[0] = DecidePhi1(); // phi_1
 9:   sig2[0] = DecideSig2(); // variation of while noise (a_t)
10:   phi[1] = DecidePhi2(phi[0]); // phi_2
11:   theta[0] = DecideTheta(phi[0]); // theta
12:
13: // Parameters are set to JRI engine
14:   engine.assign("phi", phi);
15:   engine.assign("theta", theta);
16:   engine.assign("sig2", sig2);
17:   engine.assign("num", ElapsedDays);

18: // Time series data is generated from the ARIMA model
19:   result = engine.eval("arima.sim(n=num,
                      model=list(order=c(2,0,1),
                      ar=phi, ma=theta), sd=sqrt(sig2))");
20:   ChangedLatency[i] = result.asDoubleArray();
21: }
```

Fig. A·1 Example program of derivation of time series data.

```
 1: DeriveUpdatedCoordinate (int n, int t) {
 2:   // The Unit vector and the latency between the coordinate
      // of node and the center point coordinate is derived
 3:   unitVector = CalculateUnitVector(initialCoordinate[n],
                                       centerCoordinate);
 4:   latency = CalculateLatency(initialCoordinate[n],centerCoordinate)
                     + ChangedLatency[n][t];
 5:
 6:   // The Updated coordinate of node is calculated
 7:   coordinate[0] = unitVector[0]*latency + centerCoordinate[0];
 8:   coordinate[1] = unitVector[1]*latency + centerCoordinate[1];
 9:   coordinate[2] = unitVector[2]*latency + centerCoordinate[2];
10:   return coordinate;
11: }
```

Fig. A·2 Example program of derivation of updated coordinate.

A.2 Derivation of Updated Coordinate of Node at Each Time

Derivation process of updated coordinate of node at each time is illustrated in **Fig. A·2**. The function *DeriveCoordinate* derives the coordinate of node n at time t.

The derivation process consists of two phases. In the first phase, the unit vector and latency between the initial coordinate of node n and the center point coordinate is derived, as shown in Fig. A·2, line 2–4. In the second phase, the coordinate of node n at time t is calculated from the unit vector, latency, and center point coordinate on line 7–9. Finally, the updated coordinate of node n is returned on line 10.

Hiroshi Yamamoto received his M.E. and D.E. degrees from Kyushu Institute of Technology, Iizuka, Japan in 2003 and 2006. He worked at FUJITSU LABORATORIES LTD., Kawasaki, Japan from April 2006 to March 2010. Since April 2010, he has been an Assistant Professor in the Department of Electrical Engineering at Nagaoka University of Technology. His research interests include computer networks, distributed applications, and networked services. He is a member of IEICE and IEEE.

Katsuyuki Yamazaki received his B.E. and D.E degrees from University of Electro-communications and Kyushu Institute of Technology in 1980 and 2001. At KDD Co. Ltd., he had been engaged in the R&D and international standardization of ISDN, S.S. No.7, ATM networks, L2 networks, IP networks, mobile and ubiquitous networks, etc., and was responsible for the R&D strategy of KDDI R&D Labs. He is currently a Professor at Nagaoka University of Technology.