PAPER Special Section on The Internet Architectures, Protocols, and Applications for Diversified Futures

Location-Aware Optimal Resource Selection Method for P2P-Based Contents Management and Real-Time Distribution

Hiroshi YAMAMOTO[†a)], ***Member and*** **Katsuyuki YAMAZAKI**[†], ***Fellow***

SUMMARY With the wide-spread use of high-speed network connections and high performance mobile/sensor terminals available, new interactive services based on real-time contents have become available over the Internet. In these services, end-nodes (e.g, smart phone, sensors), which are dispersed over the Internet, generates the real-time contents (e.g, live video, sensor data about human activity), and those contents are utilized to support many kinds of human activities seen in the real world. For the services, a new decentralized contents distribution system which can accommodate a large number of content distributions and which can minimize the end-to-end streaming delay between the content publisher and the subscribers is proposed. In order to satisfy the requirements, the proposed content distribution system is equipped with utilizing two distributed resource selection methods. The first method, distributed hash table (DHT)-based contents management, makes it possible for the system to efficiently decide and locate the server managing content distributions in completely decentralized manner. And, the second one, location-aware server selection, is utilized to quickly select the appropriate servers that distribute the streamed contents to all subscribers in real time. This paper considers the performance of the proposed resource selection methods using a realistic computer simulation and shows that the system with the proposed methods has scalability for a large-scale distributed system that attracts a very large number of users, and achieves real-time locating of the contents without degrading end-to-end streaming delay of content.

key words: *peer-to-peer, content distribution, DHT, location-aware node selection, network coordinate*

1. Introduction

With the increasing availability of high-speed and high-quality connections and high performance mobile/sensor terminals, the distribution of real-time human-centric contents has become readily available on the Internet. Human-centric content means information related to human activities on a real world. Examples include sensor data measured by a lot of sensors around a human, and live video/audio broadcasting generated by mobile terminals (for example, smart phone). These contents are transmitted to service providers, and are utilized to provide human-centric services which support human activities on the real world (for example, health care, life log, social communication, virtual/augmented reality). The human-centric services should be provided by a distributed and cooperative computing environment such as a cloud computing so as to quickly analyze a large amount of human-centric contents and to store a large amount of real-world data related with the services (for example, map and/or advertisement data for real-time navigation, 3D object data for virtual/augmented reality). In a health care or a navigation service, various sensor data (for example, accelerometer, gyroscope, GPS data) are sent from the users' mobile terminals (i.e., publishers) to appropriate resources (i.e., subscribers) on the cloud through the Internet, and then analyzed in order to specify a user's health condition or location. After that, the cloud resource determines the best reaction such as actuation of medical equipments or the best information such as navigation/advertisement data for the users, and responds to the users in real-time. The response causes new actions of both the user and the surroundings, hence generates new human-centric contents. Namely, the human-centric service is based on a highly-interactive and real-time communications between the content publisher and the subscribers.

In order to accommodate a large number of simultaneous content distributions, a management system which is responsible for storing distribution status is necessary. This distribution status includes the location of a content's publisher and attributes of content (for example, name, type, etc.). On the other hand, the users, who include human-centric service providers, express their interests through subscription and the system matches contents to related subscriptions. In the human-centric services, an appropriate computing resource on the cloud (i.e., subscriber of contents) for providing a service to a user (i.e., publisher of contents) can be frequently changed due to dynamic change of system load and user's condition (or suitable function for the user). Therefore, if it takes a long time for locating a corresponding content distribution to the subscription request, the start-up processes of the service are congested and delayed on the system, which degrades interactivity of the human-centric services. In other words, the system must minimize the start-up time needed to locate their desired content very quickly.

Furthermore, the management system is also responsible for distributing the human-centric contents to their subscribers simultaneously, because many users or service providers may subscribe the same content. Here, the main requirement of the real-time content distribution system is to minimize streaming delay between the publisher and the subscribers so as to accommodate the human-centric services which need interactive communications between the content publisher and the subscribers as mentioned above. Existing content distribution systems have attempted to minimize latencies between the distribution server and the sub-

Manuscript received June 5, 2012.
Manuscript revised October 1, 2012.

[†]The authors are with the Department of Electrical Engineering, Nagaoka University of Technology, Nagoya-shi. 466–8555 Japan.
a) E-mail: hiroyama@nagaokaut.ac.jp

DOI: 10.1587/transinf.E96.D.213

scribers, but have not considered the minimization of end-to-end latencies between users including publisher and subscribers.

To implement the management system, high performance computing and storage resource could be developed on the Internet. However, the centralized system requires costly resources for handling a large number of content distributions and cannot adapt to dynamic change in the number of users. Thus, the peer-to-peer (P2P) network technology is more appropriate for building the content management system because it is completely decentralized, scalable, and self-organizing.

In our preliminary study, a P2P-based content management and distribution system consisting of many servers deployed on the Internet has been proposed [1]. In contrast, in order to shorten a time interval needed for determining an optimal distribution server, this study proposes an extended version of the P2P-based system which utilizes new start-up time minimization methods (see Sect. 3.3).

The proposed system utilizes two P2P resource selection method, and each server belongs to two P2P networks. First, in order to manage a large amount of information relating to the contents distributions, and to shorten the start-up time needed for the subscribers to start receiving their requested content, the proposed system utilizes a structured P2P network technology such as Chord [2], [3]. The structured P2P network constructs a structured graph of servers, and makes it possible for the users to efficiently select and locate a server that is responsible for managing a distribution status of content that the user is interested in. Second, in order to prevent the subscribers from experiencing long delays for getting the real-time human-centric contents, the use of the network location-aware node selection framework is considered [4]. The framework builds another loosely-structured graph of servers, and decides the distribution server of each content based on distances from servers to users so as to minimize the end-to-end streaming delay between the users.

Furthermore, in order to shorten an additional start-up time required for locating the optimal distribution server, we also propose a novel method which minimizes the measurement time of the distances. The proposed method utilizes a network coordinate system [5], [7], [8] which decreases the measurement time by deriving end-to-end latencies from pre-produced network coordinates of computers. With the use of above mentioned methods, the proposed system not only achieves an excellent end-to-end streaming delay, but also markedly reduces the start-up time for contents distributions.

The rest of this paper is organized as follows. In Sect. 2, the existing peer-to-peer networking technologies related with this study are described. Section 3 presents the proposed content distribution system. In Sect. 4, an evaluation model and performance measures are introduced, and the performance of the proposed system is investigated using computer simulations in Sect. 5. Finally, the conclusions are presented in Sect. 6.

2. Related Works

This section presents existing P2P-based content distribution systems. Furthermore, we introduce P2P technologies that can be utilized to resolve main problems of the existing systems.

2.1 Existing Real-Time Content Distribution System

Many P2P-based content distribution systems have been proposed so far. Scribe is a famous group management algorithm for application-level multicast (ALM), and builds a tree-based overlay network for content distribution in a completely decentralized manner [12]. The tree structure enables an efficient contents distribution to multiple subscribers, and the decentralized algorithm can accommodate a very large number of content distributions. In addition, SplitStream adopts a multiple-tree topology in order to further improve subscribers' reception bandwidth of the contents [13]. Furthermore, a P2P-based live streaming applications such as PPLive [14] have been already released, and are now attracting a large number of users. On the other hand, a mesh-based topology for the content distribution has also been proposed by existing studies [15]–[17].

However, these algorithms and applications do not consider a situation where a large number of end-nodes of publishers generate the real-time contents and distribute them to multiple subscribers, and where end-to-end streaming delay between the publisher and the subscribers should be minimized. Furthermore, when a new subscriber requests receiving a content, the topology should be adjusted as quickly as possible without utilizing global knowledge about proximity between nodes. Therefore, in order to achieve those requirements, we utilize two P2P network technologies. First, a structured P2P network presented in Sect. 2.2 is used to build a management system of a large number of contents distributions. Second, a location-aware node selection framework shown in Sect. 2.3 is utilized to locate an optimal distribution server which transmits a content between users in the shortest time.

2.2 Structured P2P Network

The structured P2P network is a distributed system constructed by a large number of computers (nodes) without any centralized control [2], [3]. In this network, the nodes are organized into a structured graph that maps data keys to a node. The structured graph enables the users to discover the data item corresponding to the given key in a short time.

There are many implementations of the structured P2P network [9]–[11], of which Chord [9] is a well-known implementation. Each node and each data are assigned unique ID and key by hashing their identifiers (for example, IP address, filename), respectively, and are mapped onto a one-dimensional identifier space. In the identifier space, the node with identifier id_i is responsible for managing the data

whose key is within id_{i-1} and id_i, and is referred to as a *successor node* of the data (id_i is the identifier of node having i-th smallest identifier in the space).

Each node maintains the routing table with up to $\log_2 N$ other nodes (N is the number of all nodes on the identifier space), and each routing table entry includes both the Chord identifier and the location (for example, IP address) of the relevant node. The routing table with node id_i contains the entry corresponding to the successor nodes of $id_i + 2^{j-1}$. By transmitting lookup queries to the node with the ID closest to the target key, Chord efficiently locates the successor node of the user's requesting data on average in $O(\log_2 N)$ hops of query forwarding.

2.3 Network Location-Aware Node Selection Framework

Network location-aware node selection framework selects the closest node to the given target based on network location in a large-scale distributed environment. Meridian [4] is a lightweight and scalable framework that forms a loosely-structured overlay network. The node keeps track of a small, fixed number of other nodes and organizes the neighbor list, recording the locations of the nodes. The neighbor list is updated using a scalable gossip protocol, which notifies other nodes of the memberships in the system.

The Meridian node periodically measures the latency between itself and each member in the neighbor list. When the node receives a "closest node discovery to the target T" query, it determines its latency d to T, and probes its neighbors, whose latency is within $(1-\beta) \times d$ to $(1+\beta) \times d$, to determine their distance to the target. β is an acceptance threshold, and determines the reduction in distance at each hop. The query is forwarded to the neighbor closest to the target, and the process will continue until no closer neighbor is discovered.

Meridian requires relatively modest state management and processing to the nodes as mentioned above and can efficiently discover the closest node to the target on average in $O(\log N)$ hops of query forwarding.

3. Proposed Content Distribution System Design

The goal of this study is to design a scalable live content distribution system that can handle a large number of users including both publishers and subscribers. In the system, the content publishers, who are completely unaware of the existence of the subscribers, publish their streamed contents through the system by specifying identifiers of the contents. On the other hand, the subscribers express their interest through subscription requests and wait to receive the desired contents. The content management system is responsible for matching the contents to related subscriptions and distributing the contents to interested subscribers. Therefore, the system must have enough resources to handle the matching of publications to subscriptions for a large number of contents distributions and to convey the streamed contents to a large number of interested subscribers in real time.

To satisfy these requirements, the proposed content distribution system is created by interconnecting a large number of servers dispersed over the Internet. The servers collaborate to match the content publications to the subscriptions and distribute the contents to interested subscribers. In this study, it is assumed that the management system is constructed by a large number of servers like Cloud Computing. In the future system, clients (e.g., desktop, laptop, and mobile terminals) may contribute their resources for the management system like peer-to-peer applications. Furthermore, the peer-to-peer paradigm is used to build a large-scale content distribution system because of its scalability and redundancy. The proposed system is equipped with the completely distributed network technology that locates the servers responsible for managing the subscriptions of desired content distributions and for distributing the streamed content to all subscribers.

In Sect. 3.1, a management infrastructure of information about contents distributions using a structured P2P network technology is presented. And then, Sect. 3.2 shows a proposed server selection which adjust the overlay topology by locating the optimal server for transmitting contents between the publisher and subscribers without utilizing global knowledge of network conditions. Finally, in Sect. 3.3, several methods which attempt to minimize an additional start-up time needed for optimizing the topology are proposed.

3.1 Assumed Network Configuration and Management of Information

Due to its simplicity, the popular P2P network technology, Chord, is selected for building the management infrastructure of matching the publications of contents to subscriptions [9]. Chord provides an exact-mapping function between the node identifier and the keys associated with data items as described in Sect. 2.2, that is, it finds the node storing the requested data in the shortest time. The advantages of Chord are leveraged to build a scalable infrastructure.

Figure 1 shows the network configuration of the content management system. As shown in this figure, each server is assigned a unique identifier (Server ID) by hashing its location information (for example, IP address), which is mapped to a large circular identifier space. Furthermore, each content distribution is also assigned a unique

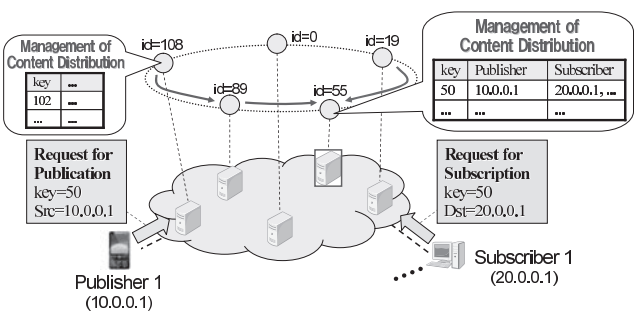

Fig. 1 Assumed network configuration.

key, which will be called the Content ID.

When starting to distribute the streamed content, the publisher chooses the key by hashing a name and/or other metadata information (for example, type of source node, format) of content. Then, the content publisher generates the publication query, and forwards it to the server on the circular identifier space. Each query includes the Content ID and identifier (and/or location) of the content publisher, as well as metadata information of the content. The server, which receives the publication query, chooses one server with the Server ID closest to the Content ID from the routing table, and forwards the query to the selected server. This process is repeated until the publication query arrives at the successor server of the Content ID.

On the other hand, the subscriber who is interested in the content first obtains the key by hashing the name and/or the metadata information. Note that the subscriber may obtain the key from the website of the content distribution service provider. The key is recorded to the subscription query, and the query is relayed to the successor server of the key over the circular identifier space.

The successor server of the Content ID is responsible for matching the publication to the subscriptions corresponding to the key and managing the information about the content, such as the location of the content publisher and the subscribers as shown in Fig. 1.

3.2 Proposed Location-Aware Distribution Server Selection

The successor server of the Content ID can work as a distribution server that delivers the content to all interested subscribers as shown in Fig. 2, because it knows the information about content distribution corresponding to the key. However, the use of the successor server may markedly increase the latency taken for propagating the content from the publisher to the subscribers, because the successor server is selected without considering any knowledge of locations of computers on the Internet.

Therefore, in order to achieve real-time content distribution, the proposed system is equipped with a new distribution server selection method. The method decides the distribution server so as to minimize the end-to-end latency between the content publisher and the subscribers, as shown in Fig. 3. The method is based on the existing network location-aware node selection framework, called Meridian [4]. From the membership management scheme of Meridian, each server keeps track of a small, fixed number of other servers selected randomly from a wide area of the Internet and organizes the neighbor list recording their locations. In addition, the server periodically measures the latency (for example, by using *ping*) between itself and each member in the neighbor list.

Figure 4 presents the algorithm of the proposed distribution server selection. When the successor server of the content distribution accepts the subscription query for the new subscriber and if the distribution server has been already decided for the existing subscribers, the successor server requests the distribution server to start the algorithm. Otherwise, the successor server itself runs the algorithm so as to discover the appropriate distribution server for the first subscriber of the content.

The server, which received the request for the distri-

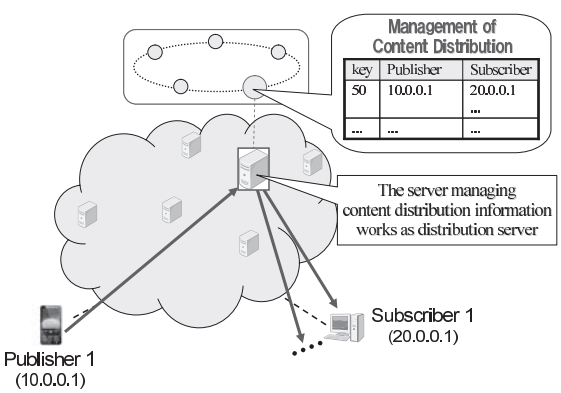

Fig. 2 Simple distribution server selection.

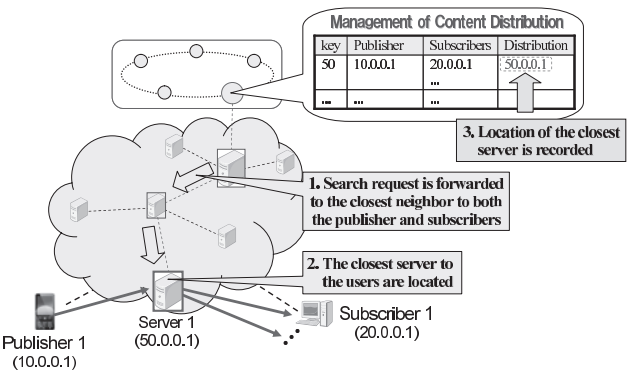

Fig. 3 Proposed distribution server selection.

```
1. Distribution_decision{
2.    min_latency = inf
3.    min_neighbor = null
4.    latency_guide = this.Distribution_point_measurement
5.    for (i in neighbor_list){
6.       if (latency_guide×(1-β) < this.Latency(i) < latency_guide×(1+β)){
7.          latency = i.Distribution_point_measurement
8.          if (latency < min_latency){
9.             min_latency = latency
10.            min_neighbor = i
11.         }
12.      }
13.   }
14.   if (latency_guide < min_latency){
15.      this.End_decision
16.   }
17.   else{
18.      min_neighbor.Distribution_decision
19.   }
20.}
```

Fig. 4 Algorithm of proposed distribution server selection.

```
1. Distribution_point_measurement{
2.    latency = latency_between_publisher_and_this
3.    for (i in subscriber_list){
4.       latency = latency + latency_between_this_and_subscriber;
5.    }
6.    return latency/(the_number_of_subscribers+1)
7. }
```

Fig. 5 Algorithm of average latency derivation.

bution server discovery, first measures its distances to the content publisher and the subscribers, and calculates the average latency *latency_guide* on line 4 in Fig. 4. The algorithm of average latency derivation is presented in Fig. 5. Next, as shown on lines 5-7 in Fig. 4, the neighbors whose latency to the server is within $(1 - \beta)\times$*latency_guide* to $(1 + \beta)\times$*latency_guide* are requested to measure their distances to the publisher/subscribers to derive the average latency. As a result, the server decides the closest neighbor to both the content publisher and the subscribers on lines 8-10. Finally, if no closer neighbor is detected, then the query forwarding stops, and the closest server currently discovered is chosen as a distribution server (lines 14-15). Otherwise, a request for the distribution server discovery is forwarded to the closest neighbor, and it will repeat the above-mentioned process (lines 17-18).

As mentioned in Sect. 2.3, the main objective of Meridian is to locate the closest node to the given target. In addition, an application of Meridian, Central Leader Election, has also been proposed in [4], which enables the users to locate centrally situated node with respect to a set of given targets. Each node in the application requests its neighbors to determine their average distances to the set of targets, and then selects the neighbor whose average distance is the smallest.

In order to locate the closest server to both the content publisher and the subscribers, our distribution server selection utilizes a procedure of the application. As the same as the application, each server requests its neighbors to measure their average distances to the publisher/subscribers, and selects the neighbor whose average distance is the smallest.

3.3 Proposed Start-Up Time Minimization Method

In the above-mentioned distribution server selection, the server must know latencies between computers corresponding to the content distribution when receiving the search request of the distribution server. The server can obtain the latencies by using an active measurement method such as *ping*, but takes a long time for receiving replies of the probe packet from the publisher and the subscribers. Due to the additional time of the active measurement, the total start-up time needed for newly-arrived subscribers to start receiving their requested contents may markedly increase. Therefore, in order to minimize the total start-up time, the method for minimizing the measurement time should be considered.

To complete the objective, we first propose two methods: *Method 1 - Cache-based Derivation* and *Method 2 - Network Coordinate-based Derivation*. The first method is a simple extension of the distribution server selection which allows the servers to cache the measurement results of distances and to reuse that. In the second method, each server derives its coordinate on a pre-defined geometric space by using network coordinate algorithms [5]–[8]. As a result, they can predict latencies between computers by calculating distances (e.g., Euclidean distance) between coordinates of computers. The second method markedly decreases the measurement time because the server does not require the additional time of the active measurement. However, it takes a long time for computers to determine their optimal coordinates, because the method mistakes a selection of the distribution server which is close to users when the coordinates of computers are not accurate. Therefore, we will propose the third method, *Method 3 - Hybrid Derivation*, by combining strong points of above-mentioned two methods.

3.3.1 Method 1 - Cache-Based Derivation

The server caches the measurement results after measuring a latency of a communication path between itself and other computer. And then, when the server is requested to measure the same path again, it reuses the cached result.

3.3.2 Method 2 - Network Coordinate-Based Derivation

All computers including both servers and user terminals of publisher/subscriber periodically measure their distances to others, and calculate their own coordinates on the predefined geometric space based on the measurement results. In order to derive accurate coordinates of computers, Vivaldi is chosen from simulation-based network coordinate algorithms because it is easy to implement [5]. The algorithm adjusts the coordinates so as to minimize the difference between an actual latency and a predicted one which is derived by calculating distance between a pair of coordinates. In our proposed system, the coordinate of computer has three dimensions (X, Y, Height) and the distance d_{ij} between computer i and j is calculated by a following equation.

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + h_i + h_j. \quad (1)$$

In this equation, (x_i, y_i, h_i) and (x_j, y_j, h_j) are coordinates of computer i and j, respectively. As in the existing research [5], [6], we use a virtual Height which represents the component of latency a computer experiences in all its paths due to its Internet access link.

As shown in Fig. 6, each server manages its own coordinate, and periodically exchanges it with the neighbors of overlay network built for the distribution server selection. In addition, when a publisher/subscriber transmits publication/subscription request of the content distribution, it registers its coordinate to a server managing the information of the corresponding content. When the server accepts the subscription request, it generates the search request including coordinates of both the publisher and the subscribers,

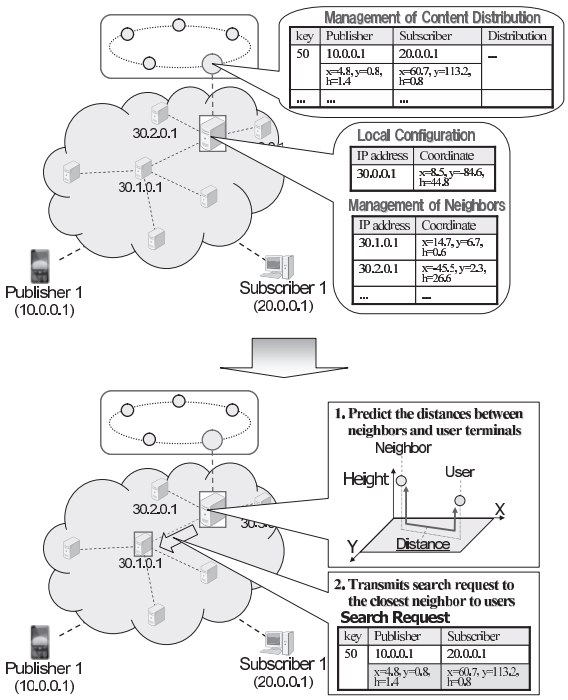

Fig. 6 Network coordinate-based derivation.

and starts to locate the optimal server for the content distribution. As a result, when receiving the search query, the server can derive the average distances, which are needed to select the closest neighbor to the users, from the coordinates of servers and that of the users recorded on the search query.

3.3.3 Method 3 - Hybrid Derivation

In order to always locate the optimal distribution server but to keep the measurement time within a lower level, the servers select better method from above-mentioned two methods according to the accuracy of their coordinates.

To complete the objective, each server manages a metric of accuracy of its coordinate. As the same as Method 2, the server periodically adjusts its coordinate by comparing the predicted latency which is derived from coordinates with the actual one which is obtained by the active measurement. When adjusting the coordinate, the server calculates a relative error according to Eq. (2).

$$Relative\_Error = \frac{|Latency_M - Latency_P|}{Latency_M}. \quad (2)$$

where $Latency_P$ and $Latency_M$ are defined as the predicted latency and the actual one, respectively. Note that when the predicted latency $Latency_P$ is more than twice as large as the actual latency $Latency_P$, the error becomes larger than one. And then, the server calculates a moving average of relative errors among ten adjustments, and treats the average error as the accuracy metric of its coordinate.

Figure 7 presents the algorithm which is used by the server to select an appropriate one from two start-up time minimization methods. When the server receives the search request of optimal distribution server, it compares the accuracy metric of coordinate (*Relative_Error*) with the predetermined threshold (*THRESH*) on line 2. If *Relative_Error* is smaller than or equal to *THRESH*, the server judges that the coordinates of computers on the network are accurate enough to predict the latencies, and decides to use the *Method 2* as shown on lines 2-4. Otherwise, it utilizes the active measurements or reuses cached results (if available) to derive the latencies (*Method 1*) on lines 5-7. Here, the same threshold is set to all servers composing the system. Finally, the search request is transmitted to the neighbor which is decided by the selected method on line 8.

```
1. Hybrid_derivation{
2.   if (Relative_Error ≤ THRESH){
3.     min_neighbor = Method2
4.   }
5.   else{
6.     min_neighbor = Method1
7.   }
8.   min_neighbor.Hybrid_derivation
9. }
```

Fig. 7 Algorithm of hybrid derivation.

4. Evaluation Model and Performance Measures

The performance of the proposed content distribution system is investigated using computer simulations that assume a realistic overlay network.

4.1 Evaluation Model for Proposed Content Distribution System

In order to evaluate the effectiveness of the proposed system, latencies between a large number of computers should be prepared. As mentioned above, we assume the management system which is constructed by servers, and positions of all computers are fixed in the evaluation model. Therefore, in this study, we examined latencies obtained from the "King data set" that contains measurements of latencies between a set of DNS servers [20]. These measurements were taken by "King tool" that estimates round-trip time (RTT) between arbitrary two DNS servers by using recursive DNS queries. First, the tool sends the query to the first DNS server for resolving a name of node that the server is managing. Next, another query is sent to the first server but is routed to the second DNS server by requesting a lookup of a name that the second server is responsible for. The RTT between two DNS servers can be found as a difference of RTTs of these two queries.

The data set captures more realistic characteristics of latencies between computers on the Internet compared with a data set used in our preliminarily study [1]. In addition, any researcher can easily download an example of the King data set including latencies between all pairs of 462 servers from [21], and can utilize that for the realistic simulations.

When Method 2 or 3 is utilized, each computer peri-

odically measure the distance to a randomly-selected other computer and adjusts its coordinate according to the Vivaldi algorithm [5]. In this study, we use a measure of "elapsed time" to express a running time of the proposed system. "1 [unit time]" is defined as the elapsed time needed for measuring latencies between all pairs of computers after the system is started, and 10 [unit time] is the elapsed time needed until every computer finishes measuring the distances to all others ten times. As the elapsed time increases, the coordinates of computers become more accurate, and thus it is possible for the proposed system with Method 2 and 3 to more accurately predict the end-to-end latency between computers from the coordinates of computers. Note that the elapsed time depends on frequency by which each computer adjusts the coordinates, so that a common elapsed time of the environment cannot be defined.

Figure 9 shows the average relative error of predicted latencies between all pairs of computers as a function of the elapsed time, where the error is calculated by Eq. (2). When the elapsed time is 0, the average relative error becomes 1 because the initial coordinates of all computers are set to $(x_i, y_i, h_i) = (0, 0, 0)$, namely $Latency_p$ in Eq. (2) is 0. With the increase in the elapsed time, accuracy of coordinates of all computers improves and the average error converges to about 0.12 as shown in this figure.

In the realistic environments, each computer changes the state (online or offline) at an individual timing, and may change the positions dynamically, hence the accuracy of coordinates fluctuates with time (especially in mobile environments). Therefore, in order to evaluate the effectiveness of the proposed system on both the transient and the stable environments, we clarify the performance measures as a function of the elapsed time. Through the evaluation, we identify the proposed method which always works well by being adapted to the state of the environment.

4.2 Evaluation Model for Scalability Study

For the P2P-based system, evaluation of scalability for a large number of computers is an important topic. However, the network model consisting of only 462 computers presented in Sect. 4.1 is not sufficient for evaluating the scalability of the system.

To expand the network model, we utilize a network coordinate algorithm. The computers are mapped to a 2-D with height network coordinate space by using the Vivaldi algorithm as shown in Fig. 8. The algorithm repeats an adjustment of the coordinates of computers until the average relative error of the coordinates (see Eq. (2)) for all computers is minimized. Specifically, we run the Vivaldi algorithm during elapsed time of 10 [unit time] to obtain the accurate coordinates.

After that, the distributions of the three sub-coordinates (X, Y, Height) were calculated and, using the distributions, up to 100,000 Vivaldi coordinates were generated. From the coordinates, the latency can be modeled between any pair of a very large number of computers.

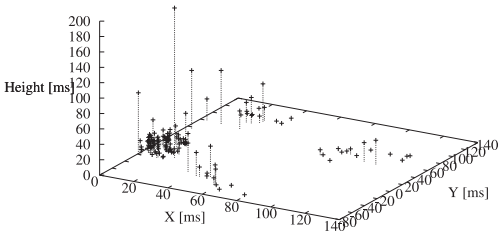

Fig. 8 2-D + height coordinates of 462 DNS servers.

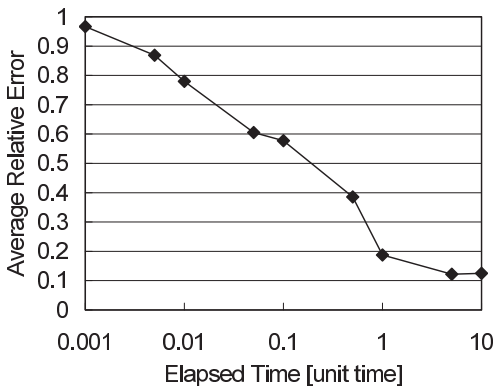

Fig. 9 Average relative error vs. elapsed time.

In this evaluation model, we model actual latencies by the coordinates of computers, which is used by start-up time minimization methods proposed in Sect. 3.3, from the actual latencies of King data set. Therefore, when clarifying the scalability, we evaluate the fundamental scalability of only the proposed system with no start-up time minimization methods.

4.3 Performance Measures

In this research, the effectiveness of handling users' subscription requests to content distribution on large-scale distributed systems is considered. As a measure of the effectiveness, we consider the start-up time needed for the newly arrived subscriber to locate the server managing the location information of the requested content distribution and to decide the distribution server. In order to minimize the end-to-end latency between the content publisher and the subscribers, the proposed system uses the network location-aware distribution server selection framework. Therefore, the streaming delay between the publisher and the subscriber is measured to determine whether the framework selects an appropriate distribution server that can transmit the streamed content to all subscribers.

Here, the subscribers of the human-centric contents are human-centric service providers, and should be able to receive the contents faster than that of one-way streaming such as video broadcasting. This is because the human-centric service is an interactive service which quickly collects events about human actions and actuates surroundings of users in real time. For example, a health care ser-

vice (one of the human-centric service) detects a user's condition getting worse by analyzing the human-centric content, and rapidly actuates medical equipments in order to recover her/his condition. And then, the actuation causes new actions of both the user and the surroundings, which generates new human-centric contents. The interactive services include many kinds of applications (for example, virtual/augmented reality, voice interaction, interactive groupware, etc.), and a maximum one-way delay of 100 [ms] is required by the interactive application with the highest sensitivity [18].

Furthermore, in traditional web-based services (e.g., file transfer, video streaming), the objective of the start-up time for locating the user's requested contents has been eight seconds (Recently, the objective has become short (e.g., 3 seconds)) [23]. However, the human-centric service should accommodate highly-interactive communications of sensor data or real-time contents. The start-up time of the human-centric services should also be minimized less than one second in order to achieve completely interactive services, but a trade-off between the end-to-end streaming delay and the start-up time exists. This is because the additional start-up time is needed for the subscriber to locate the appropriate distribution server which can shorten the streaming delay of the desired content.

Therefore, we attempt to minimize the the end-to-end streaming delay between the publisher and the subscriber with reducing the start-up time to the smallest possible value.

5. Simulation Results

First, the effectiveness and the practicality on the realistic environment of the proposed system are evaluated by using the evaluation model of Sect. 4.1. Next, we clarify the fundamental scalability of the system without adopting any start-up time minimization method on the evaluation model of Sect. 4.2. Finally, we evaluate the proposed system in mobile environments where the users are connecting to the Internet through a mobile network.

In figures of the following evaluations, *Active Measurement* indicates performance of the basic system consisting of methods explained in Sects. 3.1 and 3.2. Namely, the *Active Measurement* shows a basic method which has been proposed in [1]. On the other hand, *Simple Server Selection* in Sect. 5.3 indicates performance of a simple Chord-based management system explained in Sect. 3.1 without a location-aware distribution server selection and any start-up time minimization method.

5.1 Evaluation of Proposed Content Distribution System

The performance of the proposed system with each start-up time minimization algorithm is evaluated as a function of an elapsed time from starting to run Vivaldi algorithm. Here, 10 percent of computers serve as servers to create the content distribution system, and the others are user terminals. As well, a half of the users are content publishers, and the others are subscribers. Each publisher posts one content to the system. And then, each subscriber randomly subscribes one content from all available contents posted on the system. Furthermore, for the network location-aware distribution server selection, each server keeps track of 10 other servers and sets the parameter of β to 0.5. In the performance evaluation, the average streaming delay and start-up time in the $1,000$ experiments are used as performance measures of the proposed system.

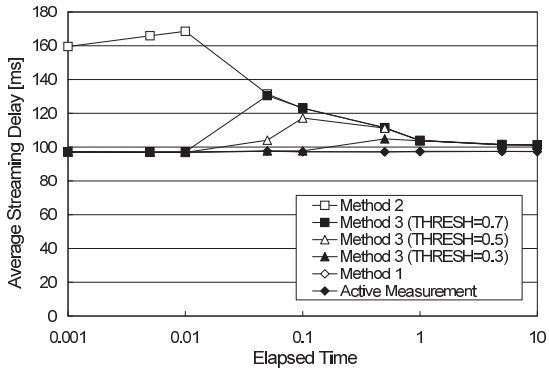

Fig. 10 Streaming delay as a function of elapsed time.

Figure 10 shows the relationship between the end-to-end streaming delay and the elapsed time. As shown in this figure, all methods reduce the end-to-end streaming delay close to 100 [ms]. The reduction of the streaming delay means that our proposed server selection method enables the content distribution system to have potential of accommodating a large number of human-centric services which require a maximum one-way delay of 100 [ms] with the highest sensitivity.

However, Method 2 degrades the performance at an earlier stage when the coordinates of computers are not accurate. On the other hand, Method 3 always achieves the streaming delay close to the targeted value by selecting the method which is appropriate for the accuracy of coordinates especially when the threshold $THRESH$ in Fig. 7 is set to less than or equal to 0.5.

Next, Fig. 11 shows the average start-up time as a function of the elapsed time. As shown in this figure, Method 2 outperforms other algorithms. Furthermore, Method 3 achieves the excellent start-up time when the coordinates of computers are accurate. And, by setting the threshold $THRESH$ to 0.5 or larger, the interval when the additional start-up time is required becomes very short.

As a result, we conclude that Method 3 with the appropriate threshold is the best start-up time minimization method because it minimized the start-up time at the condition where the streaming delay is close to 100 [ms].

5.2 Effect of Realistic Pattern of Content Subscriptions

Next, we evaluate the performance of the proposed system with Method 3 on a more realistic pattern of content sub-

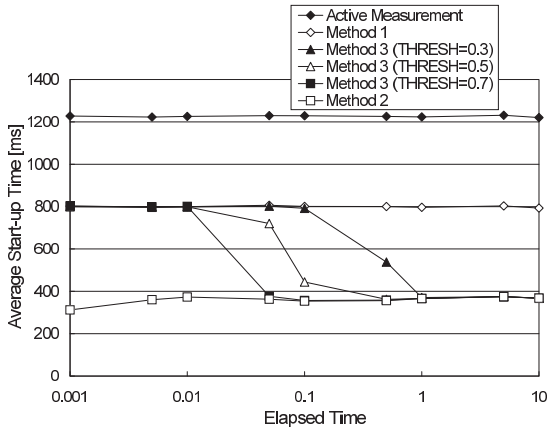

Fig. 11 Start-up time as a function of elapsed time.

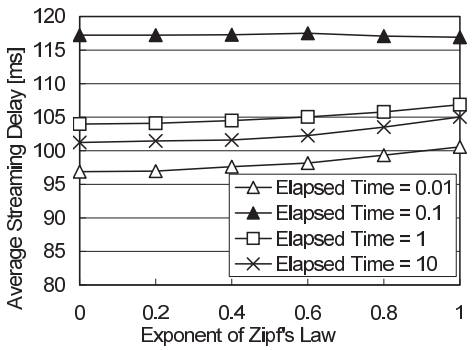

Fig. 12 Streaming delay as a function of exponent of Zipf's law.

scriptions (i.e., there are a few widely-subscribed contents and a large number of less-subscribed contents). In this evaluation, the threshold of Method 3 is set to 0.5. Here, the performance metrics are presented as a function of the exponent s of the Zipf's law shown in Eq. (3).

$$f(k, s, N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}. \quad (3)$$

where f is the frequency at which the content of rank k is selected, and N is the number of contents. With the increase in the exponent s, the highly ranked contents become more popular. Other settings of this evaluation are the same as Sect. 5.1.

Figure 12 shows the end-to-end streaming delay as a function of the exponent of Zipf's law, with elapsed time as a parameter. As shown in this figure, the proposed system achieves the excellent streaming delay even when only highly ranked contents are subscribed. Furthermore, the relationship between the start-up time and the exponent of Zipf's law is presented in Fig. 13. When a smaller number of contents are subscribed by a large number of subscribers, the server is requested to measure the distance to the same publishers/subscribers, and reuses cached measurement results frequently. Therefore, as shown in Fig. 13, the start-up time of the proposed system decreases with the increase in the exponent s especially when the coordinates of computers are not accurate.

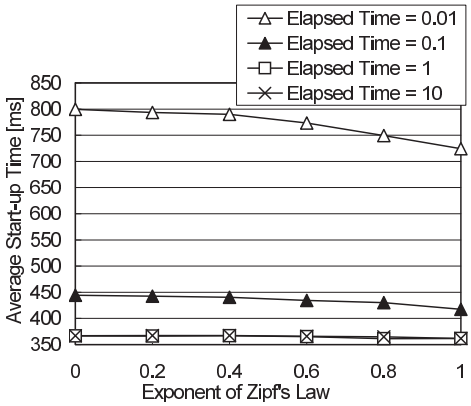

Fig. 13 Start-up time as a function of exponent of Zipf's law.

From these results, we can conclude that the proposed system with Method 3 works well in the realistic environment where a small number of contents are more popular than others.

5.3 Scalability Study of Proposed System

Next, the fundamental scalability of the proposed system with no start-up time minimization method is obtained as a function of the number of computers, which ranges from 1000 to 100,000. One percent of all computers serve as servers to build the content distribution system, and the others are user terminals, like in Skype. Furthermore, for the network location-aware distribution server selection, each server keeps track of 50 other servers and sets the parameter of β to 0.5. In the performance evaluation, the average streaming delay and start-up time in the 10 experiments are used as performance measures of the proposed system.

In order to determine the effect of the distribution server selection, two different versions of the system are compared. The first is the content distribution system with the server selection which utilizes only active measurements (*Active Measurement*), and the second is the same system but without the selection (*Simple Server Selection*).

First, the effect of a changing number of computers on the average streaming delay between the content publisher and the subscriber is considered. Figure 14 shows the relationship between the number of computers and the average end-to-end latency between the content publisher and the subscriber. As shown in this figure, the proposed system with the distribution server selection outperforms that without selection and achieves a delay less than 100 [ms]. This is because the selection effectively chooses the servers so as to shorten their distance to both the content publisher and the subscribers.

Next, Fig. 15 shows the effect of the number of computers on the average start-up time needed for the subscribers to start receiving the content. As shown in this figure, the average start-up time in the proposed system with the distribution server selection increases from that of the system

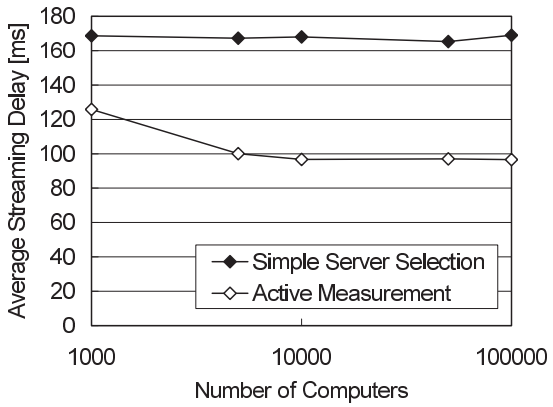

Fig. 14 Streaming delay as a function of the number of computers.

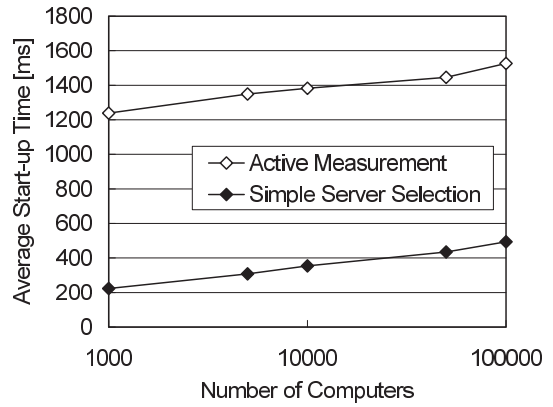

Fig. 15 Start-up time as a function of the number of computers.

without the selection. This is because both systems make the subscribers wait as the successor server managing the location information of content distribution is located, but the use of the server selection requires an additional time interval until the server closest to the users is decided.

However, the increase in the start-up time is insensitive to the increase in the number of computers, and the average time interval does not exceed 1.6 [s] even when the number of computers is 100,000. Furthermore, as shown in Sect. 5.1, by using the start-up time minimization method, the proposed system may halve the start-up time. Scalability study of the proposed system with the start-up time minimization method is our future work.

On the other hand, our proposed system may concentrate a larger number of content distributions on a small number of servers than the existing system (i.e., simple DHT-based system). This is because our proposed system decides a distribution server of each content based on proximity among computers while the DHT does not consider any information except the Content ID. The load concentration may reduce scalability of the content distribution system, but can be easily mitigated by transferring a part of load on the selected distribution server to its neighbor with the lowest load.

Here, we propose the straightforward load-balancing

```
1. Distribution_LoadBalancing{
2.   min_load = this.Streaming_load
3.   min_neighbor = null
4.   latency_guide = this.Distribution_point_measurement
5.   for (i in neighbor_list){
6.     if (latency_guide×(1-β) < this.Latency(i) < latency_guide×(1+β)){
7.       latency = i.Distribution_point_measurement
8.       if (latency < latency_guide×(1+r)){
9.         load = i.Streaming_load
10.        if (load < min_load){
11.          min_load = load
12.          min_neighbor = i
13.        }
14.      }
15.    }
16.  }
17.  if (min_neighbor == null){
18.    this.End_decision
19.  }
20.  else{
21.    min_neighbor.End_decision
22.  }
23.}
```

Fig. 16 Algorithm of proposed load balancing.

method for our content distribution system, and evaluate the trade-off between the streaming delay and the degree of load concentration. Figure 16 presents an algorithm of the load-balancing method. After the appropriate distribution server is decided (i.e., line 15 in Fig. 4), it extracts the candidate neighbor whose average latency to both publisher and subscribers (see Fig. 5) is less than $(1 + r) \times$ that of the distribution server on line 8 in Fig. 16, where the r indicates an acceptance rate of increase in the streaming delay. And then, the distribution server selects the candidate whose streaming load (i.e., how many subscribers are accommodated) is the smallest on lines 9-12 in Fig. 16, and transfers the requested content distribution to the selected neighbor. In this algorithm, the average streaming delay (i.Distribution_point_measurement in Fig. 16) has already been measured in the procedure of the proposed distribution server selection (see Fig. 4), and the server can exchange information of the streaming load with the neighbors anytime. Therefore, the proposed load-balancing method does not increase the start-up time except for the communication time needed for transferring the streaming load from the distribution server to the selected neighbor.

Figure 17 shows the effect of the acceptance rate r of the load-balancing method (Active Measurement - with Load Balancing) on the performance metric related with load balancing when the number of computers on the system is set to 100,000. As a comparative value, we also illustrate a performance of simple DHT-based System (Simple Server Selection) and that of the proposed system without load-balancing method (Active Measurement - without Load Balancing) in this figure. In order to express the degree of the load concentration, we utilize *Balance Index* (Jain's Fairness Index) shown in Eq. (4) [22].

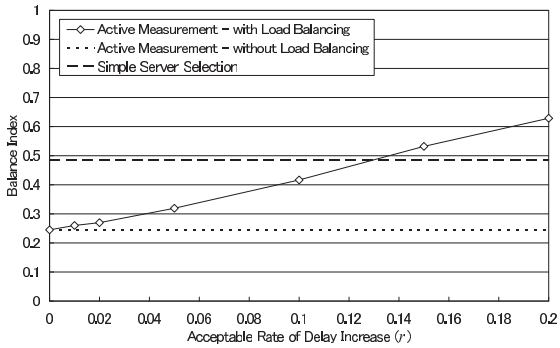

Fig. 17 Balance index of streaming load as a function of acceptance rate of delay increase.

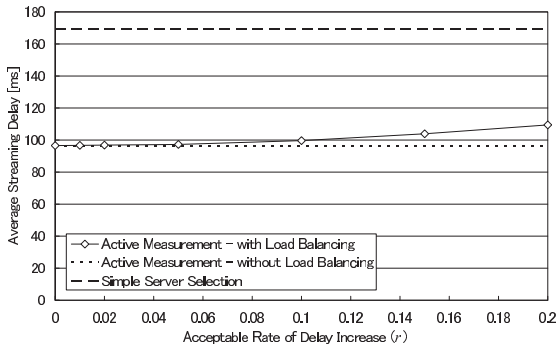

Fig. 18 Streaming delay as a function of acceptance rate of delay increase.

$$Balance\ Index = \frac{\left(\sum_{i=0}^{N-1} x_i\right)^2}{N \times \sum_{i=0}^{N-1} {x_i}^2} \tag{4}$$

where N and x_i indicate the number of all servers on the system and the number of subscribers accommodated by i-th server, respectively. When the Balance Index is closer to one the streaming load is evenly distributed to all servers, and when it is close to zero the load is concentrated on the small number of servers. As shown in Fig. 17, the load concentration can be mitigated by increasing the acceptance rate r, and achieves a better Balance Index than the simple DHT-based method (i.e., Simple Server Selection) when r is set to 0.15.

In addition, Fig. 18 presents a relationship between the acceptance rate and the average streaming delay. This figure shows that the average streaming delay does not exceed 105 [ms] even when the r is set to 0.15. Therefore, it can be concluded that the use of the simple load-balancing method achieves better load-balancing performance without largely increasing the streaming delay than the DHT-based system.

5.4 Evaluation of Proposed System in Mobile Environments

Finally, we evaluate the effectiveness of the proposed system in mobile environments. This is because the users of the content distribution system may be mobile terminals or sensors/actuators having wireless interfaces. In the mobile environment, addition of wireless links to network paths increases end-to-end latency between nodes, and the latency largely fluctuates compared with the fixed environment. Therefore, it should be clarified whether the degradation of communication quality in the mobile environment prevents the management system from accommodating human-centric services which require higher interactivity.

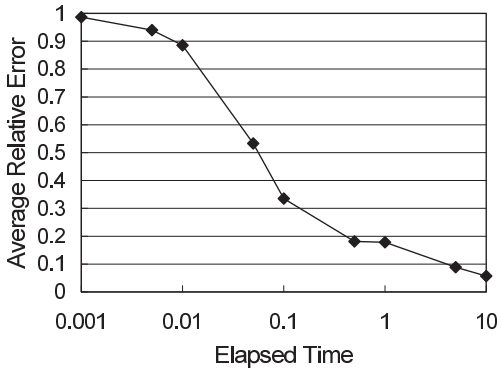

Fig. 19 Average relative error vs. elapsed time (Mobile).

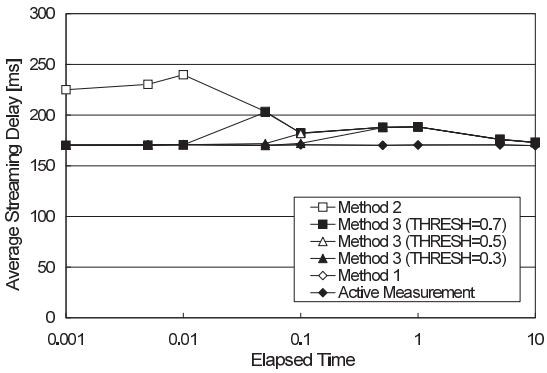

Fig. 20 Streaming delay as a function of elapsed time (Mobile).

In the evaluation, we assume that all subscribers and publishers are connecting to the Internet through HSDPA (High Speed Downlink Packet Access). The evaluation model presented in Sect. 4.1 is used for this evaluation. However, when receiving or sending a packet, the user terminal requires an additional one-way delay due to a propagation delay of HSDPA. Here, the delay of HSDPA is decided according to a uniform distribution between 25 [ms] and 40 [ms] as described in [19]. Other settings are the same as that in Sect. 5.1 Evaluation of the impact of the time variation of the end-to-end latency on the performance of our proposed method is our future work.

Figure 19 shows the average relative error of predicted latencies between all pairs of computers as a function of the elapsed time in the mobile environments. As shown in this figure, with the increase in the elapsed time, the average relative error decreases to less than 0.1.

Figure 20 shows a relationship between the end-to-end streaming delay and the elapsed time in the mobile environments. As shown in this figure, the end-to-end streaming de-

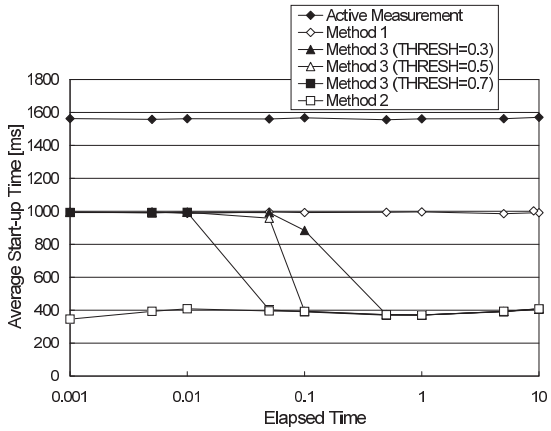

Fig. 21 Start-up time as a function of elapsed time (Mobile).

lay in the mobile environments is larger than that in the fixed environments (Fig. 10) due to the propagation delay (i.e., about 70 [ms]) of HSDPA. However, Method 3 achieves almost the same streaming delay as Method 1 or Active Measurement by setting a threshold $THRESH$ to less than or equal to 0.5 even in the mobile environments.

Next, Fig. 21 shows the average start-up time as a function of the elapsed time in the mobile environments. As shown in this figure, the start-up time of Active Measurement and Method 1 in the mobile environments is larger than that in the fixed environment (Fig. 11) because the server requires an additional propagation delay of HSDPA when measuring the distances to user terminals. On the other hand, Method 2 and Method 3 can reduce the start-up time to 400 [ms] by enabling the servers to predict latencies between computers without active measurements. Furthermore, by setting the threshold $THRESH$ to 0.5 or larger, Method 3 can reduce the time interval when the additional start-up time is required for the active measurement even in the mobile environment.

As mentioned above, the wireless link added to network paths increases both the streaming delay and the start-up time, but our proposed method can mitigate the impact of these degradation. Therefore, it is concluded that our proposed method helps the content distribution system to accommodate the human-centric services in the mobile environments.

5.5 Discussion of Effectiveness of Proposed System in Mobile Environments

In the performance evaluations, we have assumed environments where positions of all computers are fixed. With the increase in running time of the proposed system, the coordinates of computers in the environments become more accurate, which enables the servers to obtain accurate latencies without active measurements. Therefore, the start-up time minimization method of Method 3 can markedly reduce a start-up time needed for determining the distribution server when the elapsed time is large.

In mobile environments where there are a lot of mobile terminals, when the computers change their positions, the accuracy of the coordinates of computers decreases. As shown in Figs. 10 and 11, when the accuracy of the coordinates is low (namely, when the elapsed time is small), the start-up time increases because the server in Method 3 uses active measurements to obtain accurate values of latencies. In addition, the server caches the measurement results after measuring latencies, but should frequently discard the cached results because accuracy of the cached results decreases with time in the mobile environments. Therefore, in the mobile environments, the start-up time frequently increases due to movements of the mobile terminals although the streaming delay can decrease.

On the other hand, in order for Method 3 to always achieve the excellent start-up time in the mobile environment, the computers should derive their accurate coordinates quickly by frequently measuring latencies among them. However, the frequent measurements may cause a heavy load on communications over the Internet, hence the measurement interval should be carefully determined.

Furthermore, our proposed system aims at optimizing performance metrics when the subscription request of the content is generated by the subscriber. In the condition where the mobile terminals move around after deciding the distribution server, the streaming delay may increase gradually from the optimal value. Therefore, a derived method which periodically reselects the appropriate distribution server based on the latest coordinates of computers should be considered for the mobile environments.

6. Conclusions

In this paper, a new content distribution service infrastructure consisting of many servers deployed over the Internet has been developed. The advantage of peer-to-peer technology has been used to build a completely distributed management system that is scalable for a large number of users.

The well-known implementation of a structured P2P network, called Chord, has been used to manage the information about the content distributions. The proposed system has also used a network location-aware distribution server selection to determine the server that will distribute the streamed content to the interested subscribers. Furthermore, we have proposed a method which reduces the additional time needed for the distribution server selection to obtain the distances between computers by utilizing the network coordinate system.

The performance of the management system with the proposed methods has been investigated using a realistic computer simulation. In the evaluation, the subscribers in the proposed system were able to efficiently select and locate the servers managing the status of their desired contents distributions, and have discovered appropriate servers for the distributions very quickly in both fixed and mobile environments. Furthermore, the proposed system markedly reduced the end-to-end streaming delay between the content

publisher and the subscriber.

Acknowledgments

This study was supported in part by KDDI Foundation, Research Grant Program.

References

[1] H. Yamamoto and K. Yamazaki, "Decentralized live video broadcasting system using location-aware P2P network technology," Proc. IEEE Consumer Communications and Networking Conference (CCNC2011), pp.1092–1096, Jan. 2011.
[2] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," IEEE Commun. Surveys Tutorials, vol.7, no.2, pp.72–93, 2005.
[3] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," ACM Comput. Surv, vol.36, no.4, pp.335–371, 2004.
[4] B. Wong, A. Slivkins, and E.G. Sirer, "Meridian: A lightweight network location service without virtual coordinates," ACM SIGCOMM Computer Communication Review, vol.35, no.4, pp.85–96, 2005.
[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," Proc. ACM SIGCOMM 2004, pp.15–26, Aug. 2004.
[6] S. Agarwal and J.R. Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," Proc. ACM SIGCOMM 2009, pp.315–326, Aug. 2009.
[7] Y. Shavitt and T. Tankel, "Big-bang simulation for embedding network distances in Euclidean space," IEEE/ACM Trans. Netw., vol.12, no.6, pp.993–1006, 2004.
[8] Y. Chen, Y. Xiong, X. Shi, J. Zhu, B. Deng, and X. Li, "Pharos: Accurate and decentralised network coordinate system," IET Commun., vol.3, no.4, pp.539–548, 2009.
[9] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," ACM/IEEE Trans. Netw., vol.11, no.1, pp.17–32, 2003.
[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," Proc. ACM SIGCOMM 2001, pp.161–172, Aug. 2001.
[11] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," Proc. 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), pp.329–350, Nov. 2001.
[12] M. Castro, P. Druschel, A.-M. Kermarrec, and A.I.T. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," IEEE J. Sel. Areas Commun., vol.20, no.8, pp.1489–1499, 2002.
[13] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," Proc. ACM Symposium on Operating Systems Principles (SOSP '03), pp.298–313, Oct. 2003.
[14] "PPTV - The most popular net TV in the world," http://www.pptv.com/en/index.html
[15] C. Gkantsidis and P.R. Rodriguez, "Network coding for large scale content distribution," Proc. IEEE Inforcom 2005, March, 2005.
[16] D. Ren, Y.T.H. Li, and S.H.G. Chan, "On reducing mesh delay for peer-to-peer live streaming," Proc. IEEE Infocom 2008, April, 2008.
[17] X. Zhang, J. Liu, B. Li, and T.S.P. Yum, "CoolStreaming/DONet: A data driven overlay network for peer-to-peer live media streaming," Proc. IEEE Inforcom 2005, March, 2005.
[18] D. Delaney, T. Ward, and S. McLoone, "On consistency and network latency in distributed interactive applications: A survey - Part I," MIT Press Journals, vol.15, no.2, pp.218–234, May 2006.
[19] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perala, "HSDPA performance in live networks," Proc. IEEE International Conference on Communications (ICC '07), pp.467–471, June 2007.
[20] K.P. Gummadi, S. Saroiu, and S.D. Gribble, "King: Estimating latency between arbitrary Internet end hosts," Proc. SIGCOMM IMW 2002, pp.5–18, Nov. 2002.
[21] King data set, http://pdos.csail.mit.edu/p2psim/kingdata/, 2004.
[22] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC Research Report TR-301, Sept. 1984.
[23] D.A. Menasce, "A reference model for designing an E-commerce curriculum," IEEE Concurrency, vol.8, no.1, pp.82–85, Jan.-March 2000.

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

Katsuyuki Yamazaki received B.E. and D.E degrees from the University of Electro-communications and Kyushu Institute of Technology in '80 and '01, respectively. At KDD Co. Ltd., he had been engaged in 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 R&D strategy of KDDI R&D Labs. He is currently a Professor of Nagaoka University of Technology.