IPSJ Journal Vol. 49 No. 7 2408–2420 (July 2008)

Regular Paper

Implementation Methodology of Geographical Overlay Network Suitable for Ubiquitous Sensing Environment

Matsuura Satoshi,[†1] Fujikawa Kazutoshi[†1] and Sunahara Hideki[†1]

To handle ubiquitous sensor data, we have been developing a geographical-location-based P2P network called "Mill". In a ubiquitous sensing environment, data manipulation by users and sensing devices may have some characteristics. For example, sensor data are constantly generated and users keep getting these data in a particular region. When handling such an operation on existing overlay networks, relaying nodes have to process many queries issued by users. In this paper, we discuss an implementation methodology of "Mill" considering the characteristics of data manipulation. And we can also support event driven behaviors and consider the feasibility of the deployment through the implementation based on HTTP. The performance of the implementation shows that one Mill node can handle ten thousands users and sensing devices.

1. Introduction

Today's sensing devices such as weather sensors, cars, and other devices have become powerful. In addition, these devices have connectivity to the Internet and equip positioning devices such as GPS sensors. It is expected that the number of these devices keeps getting larger. In this environment, these sensing devices can readily collect and provide information anywhere and anytime. We call this environment *ubiquitous sensing environment*.

If we use a large number of information these devices provide, we can obtain detailed and up-to-date information. And these huge data can be applied to lots of applications. Gathering information based on the geographical location can be efficient for environmental problems, educational material, bussinesses and our daily lives. For example, if we could gather exact torrential rainfall information of some region, this information would be useful for evacuation instructions. And if we could examine the distribution of the temperature, we would apply this information to solve the problem of heat-island effect.

In a ubiquitous sensing environment, sensor data are generated anywhere and anytime. To manage and provide a large numer of sensor data, we have been deveploping a geographical location oriented overlay network called "Mill" [1].

In this paper, we discuss the implementation of "Mill", which is suitable for a ubiquitous sensing environment. In this environment, queries from users and sensing devices have some characteristic patterns. In recent years, we have installed weather sensors and shared these information on the Live E! project [2]. In Live E! networks, almost all users constantly get weather information of the same region, because users want to know weather conditions around their home, office or other particular places. And sensors constantly generate weather data of the same region, because all almost sensors are equipped on a particular location.

Mill provides a distributed environment based on the geographical location and supports a multiscale region search by $O(\log N)$ hops. In addition, the number of queries of users and sensors are minimazed by iterative routing in a Mill network. Mill also supports an event driven mechanism on geographical overlay networks.

The rest of this paper is structured as follows. Section 2 describes the requirements in a ubiquitous sensing environment. Section 3 shows the related work. Section 4 and 5 present the mechanism of Mill and explain its implementation. Section 6 describes evaluations of Mill's implementation and shows how many users and sensing devices Mill can handle. Finally, we summarize our contribution in Section 7.

2. Requirements for a Ubiquitous Sensing Environment

In a ubiquitous sensing environment, there are lots of devices including weather sensors, web cameras and other sensor devices. To manage this huge number of data, we have proposed an overlay network called "Mill", which supports "scalability", "multiscale region search" and "fast search". However, there are several requirements that we have to cope with.

- handling the repetition of similar queries
  Sensing devices such as weather sensors constantly generate data and users need to constantly get these data to know the current status. For example,

†1 Nara Institute of Science and Technology

users check the current status of the temperature, the wind speed or other weather information. On existing overlay networks, users need to search the overlay network each time they want to get data. In this case, relaying nodes should repeatedly pass the same or some similar queries from one node to another. This operation wastes time and network resources.

If overlay networks could manage location of data and enable users to access target sensor data directly, it is possible to reduce the almost all relaying queries in overlay networks.

Systems managing ubiquitous sensors should consider the characteristics of query patterns from users and sensing devices.

- supporting an event driven behavior

  There is a requirement that users want to receive an alert message. For example, users want to receive a message when it is starting to rain. To realize these alert services, a polling method is useful, but it might waste network resources. To save network resources, an event driven method is required.

  On existing overlay networks, users should install some special software to get data from them. If some nodes pushing alert messages knew some users information in advance, the special software would enable these users to receive alert messages when a particular event occurs. However, the setup cost of installing special software restricts the number of users. Sensor data, such as temperature and rainfall, are highly public data. In order to enable lots of users to use such public sensor data, overlay networks should support an event driven method without special software.

- considering the flexibility of system operation

  In overlay networks, using some special software restricts the number of users and makes the operation difficult. And in a ubiquitous sensing environment, the number of sensing devices can get very large. To accommodate this situation, overlay networks must consider the flexibility of system operation which includes an easy installation, an easy operation and the flexibility for user's environments.

3. Related Work

Japan Automobile Research Institute (JARI) and WIDE project experimented with IPcars (taxies have some sensors and connectivity to the Internet) [3]. This experiment showed that information provided from mobile devices is useful to know detailed weather information. In this experiment, a client-server approach was adopted. In the near future, it is expected that ubiquitous sensing environment expand and the number of queries for location-related information will much increase. Consequently servers will be much overloaded.

To decentralize information and queries, peer to peer (P2P) networks are widely studied. Especially, P2P networks with distributed hash table (DHT) have been proposed in lots of studies [4)–7)]. DHTs are scalable for the number of nodes and support fast search. DHTs can process queries even if the network is continuously changing. However, there is a serious disadvantage. DHTs support only exact-match lookups because of adopting a hash function. If users search a range consisting of consecutive IDs, lots of queries each whose the number is equal to one of the numbers in the consecutive IDs should be processed. Therefore, the exact-match mechanism is not suitable for searching a particular region.

SkipNet [8] is one of the structured peer-to-peer networks. SkipNet does not use hashed-IDs but consecutive IDs. Using consecutive IDs, SkipNet support range search on one dimension. Mercury [9] is a SkipNet based overlay network. Mercury manages several overlay networks to support range search by several attributes. For example there is one car, and this car has some attributes (color=#FFFFFF, price=$30,000, weight=1,500 kg). SkipNet can only manage one attribute. On the other hand, in a Mercury network users can search cars by color, price or weight. However, Mercury can not process the query consisting of two or more attributes at one time, because each attribute is independent from the others. If a user searches a geographical square region in a Mercury network, he has to search a large region, because a square region is determined by two attributes (latitude and longitude).

There are several P2P networks considering the location. However, these P2P networks have some defects in managing location-related information. GHT [10] is a location based P2P network. This P2P network defines an area as a square

region divided by latitude and longitude. GHT adopts a routing algorithm based on GPSR [11]. GHT supports region search. However the performance of the search is not good and this network is not scalable.

In DHTs networks, a large number of queries are generated for region search because of a hash function. In Mercury networks, users can search a particular range on one dimension. However, users have to search a large region to get information of a particular geographical region, because each attribute is independent from others in Mercury networks. It is difficult to handle sensor data based on the geographical location in these overlay networks.

In a GHT network, users can search a two-dimensional surface for target regions. However, the performance of the search is low ($O(\sqrt{N})$) and the fixed size of the grid restrains a multiscale geographical search. Moreover GHT does neither consider an event driven behavior nor the flexibility of the system operation.

4. Mill: A Geographical Location based P2P Network

Mill provides a distributed environment to handle sensor data. In a Mill network, sensor data are managed based on the geographical location. Mill supports a multiscale region search and the performance of search is $O(\log N)$. This section introduces the mechanism of Mill and describes its implementation.

4.1 Overview

If we want to use a large number of sensor data, a P2P network system must support a multiscale region search.

DHTs support only the exact-match queries because of adopting a hash function (e.g., SHA-1 [12]). If we search a particular region, we should search all points in the region. For example, if a DHT system expresses a region as 10 bits, we should search 1024 points. Exact-match causes a large number of queries on the search region.

In DHT based networks, each node is responsible for the management of a part of the whole hash table. On the other hand, in a Mill network, each node manages a part of the ID-space which is calculated by using "Z-ordering", which represents the square surface of the earth. This ID-space enables Mill to support a multiscale region search.

As **Fig. 1** shows, the architecture of Mill is similar to the DHTs, and it is a

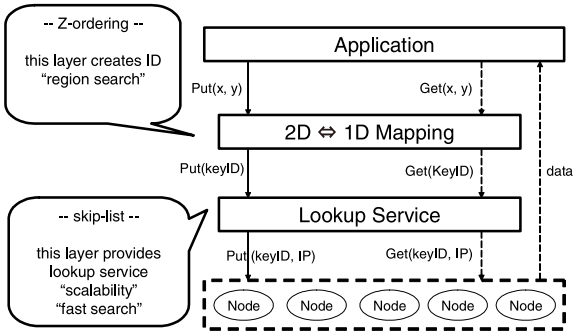


Fig. 1 Architecture of Mill.

hierarchical structure. In a Mill network, if an application stores or searches location-related information, the application just specifies the latitude (y) and the longitude (x). The 2D-1D mapping layer converts x and y into a key-ID. This layer corresponds to a hash function of DHTs. The lookup layer searches a particular node based on this key-ID. In case that there are N nodes, a query can be processed via $O(\log N)$ messages.

4.2 Z-ordering: Representing a 2D Surface as Consecutive IDs

Mill divides a two dimensional space into a grid cell by latitude and longitude. A grid cell is a very tiny square region. If each grid cell is represented by a 64 bit-ID over the surface of the earth, the size of a grid cell is in the order of the millimeter order. As **Fig. 2** shows, we suppose that each cell is assigned a 4 bit identifier. Mill manages these IDs as one dimensional circular IDs, and each Mill node is responsible for a portion of these circular IDs. The ID of each cell is generated by alternating x-bit and y-bit. For example, if an x-bit is '00' and a y-bit is '11', a cell ID is '0101'. This method is called "Z-ordering" [13]. Hilbert curve [14] is also one popular space filling curves. As Fig. 1 shows, Mill consists of several layers, and Mill can exchange an algorithm of "1D-2D mapping layer". It is important for ID creation to rely not on hash functions like DHTs but on space filling curves.

Here, the ID-space is very small, that is only 4-bits, to explain simply, however in real use Mill's ID space is represented by 64-bits. A particular region can be expressed as a range between "Start-ID" and "End-ID". For example, in Fig. 2 ID

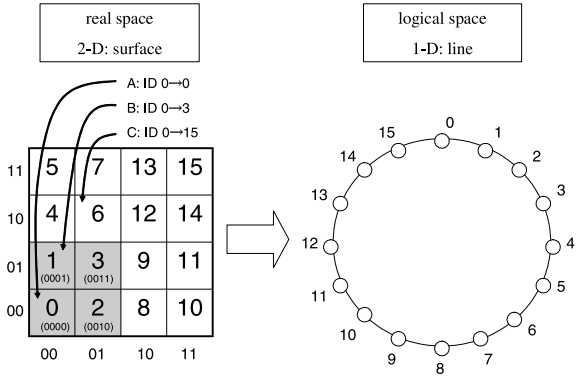


Fig. 2 2D-1D mapping method.

range (0, 0) corresponds to a square cell (region A), ID range (0, 3) corresponds to the quarter of the whole square (region B), and ID range (0, 15) corresponds to the whole square (region C). In fact, Mill expresses a particular square region as a succession of cell IDs and can search some information by range of IDs at one time. Mill searches location-related information by a few queries against an arbitrary size of the region. Because of this feature, Mill can reduce the number of search queries.

Here, we summarize the features of "Z-ordering".

- locality of ID
  - region search, load-balance
- consecutive ID
  - reduce search queries
- create one-dimension ID
  - simple management, fast & simple search

4.3 Join Protocol

Each node is responsible for the handling of a portion of the circular ID-space extending from his own ID to the next node ID. And each node communicates with two clockwise side nodes and two counterclockwise side nodes.

A new node joins Mill network by the following protocol. **Figure 3** shows an example of joining Mill network.

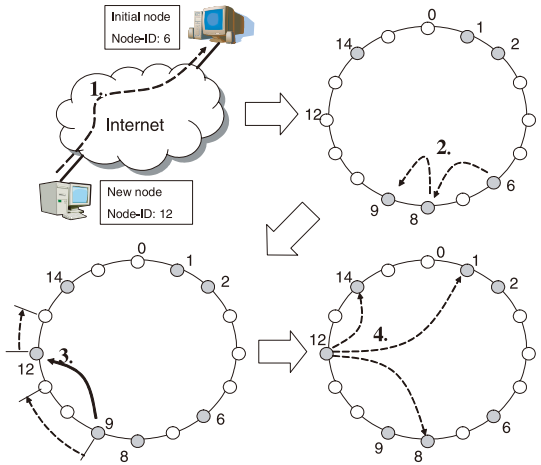


Fig. 3 Join protocol.

(1) A new node is assigned an ID according to the geographical locations (latitude and longitude). We define this ID as Node-ID. The new node knows the IP-address of at least one node in advance. We call this node "initial node". And the new node sends Node-ID and IP-address to the initial node. As Fig. 3 shows, the new node creates 12 as Node-ID from its geographical location. After that, the new node sends Node-ID (12) and IP-address to the initial node (Node-ID: 6).

(2) The initial node sends this message to the clockwise side node, and the clockwise side node sends this message in the same way. As several nodes send the message repeatedly, finally this message reaches a particular node which handles the ID-space including Node-ID. The initial node (Node-ID: 6) sends the message to the node (Node-ID: 8). And the node (Node-ID: 8) sends the message to the node (Node-ID: 9) which handles the ID-space including the new node's Node-ID (12).

(3) The node which handles ID-space including Node-ID assigns a part of the ID-space to the new node and reassigns his own ID-space. And this node also informs the new node about Node-ID and the IP-addresses of neighbor nodes. The node (Node-ID: 9) assigns the ID-space (12, 13) to the new node

and informs the new node about Node-ID and the IP-addresses of neighbor nodes (Node-ID: 8, 9, 14, 1). The node eventually (Node-ID: 9) reassigns the ID-space (9, 10, 11) by itself.

(4) The new node informs neighbor nodes about his own Node-ID and IP-address. Then neighbor nodes update their routing table. The new node (Node-ID: 12) informs neighbor nodes (Node-ID: 8, 14, 1) about his own Node-ID and IP-address.

Through the above join protocol, the new node can be assigned a particular ID-space and get to know neighbor nodes.

4.4 Leave Protocol

Each node constantly sends keep-alive messages to neighbor nodes on its routing table. If a link of a node is suddenly disconnected, a next node finds out the disconnection by keep-alive messages, and the next node tries to recover the overlay network. A node leaves the Mill network by the following protocol. **Figure 4** shows an example of leaving Mill network.

(1) Each node is responsible for a part of the ID-space, and constantly sends keep-alive messages to neighbor nodes. A node (Node-ID: 2) is responsible for the ID-space from ID-2 to ID-5, and this node sends a keep-alive message to a node (Node-ID: 6).

(2) If some nodes disconnect from the Mill network, a neighbor node finds out the disconnection by the absence of keep-alive message. The node (Node-ID:2) finds out the disconnection of the node (Node-ID: 6) by the failure of keep-alive.

(3) After detection of a disconnection, a next node tries to reconfigure its own ID-space including the ID-space of the disconnected node. After reconfiguration of the ID-space of the node (Node-ID:2), this node is responsible for the ID-space from ID-2 to ID-7. This ID-space includes the ID-space which the node (Node-ID: 6) had.

If a link of a node is disconnected, Mill network recovers the ID-space of the overlay network by itself through the above processes. If a node wants to leave, the leaving process is simpler than the case where a sudden disconnection occurs. Instead of a detection by the absence of keep-alive messages, the leaving node sends a leave-message to the next node. After accepting the leave-message, the next node reconfigures its ID-space as described in the second and third operations above.

4.5 Routing Algorithm Based on Skiplist

The clockwise liner search is simple and easy to implement. However, the clockwise liner search is not scalable, because a search query is sent through a sequence of $O(N)$ other nodes toward the destination. To reduce a search cost, each node manages information of nodes which are power of two hops away, as like 1, 2, 4, 8, 16 hops away. **Figure 5** shows the process of creating a routing

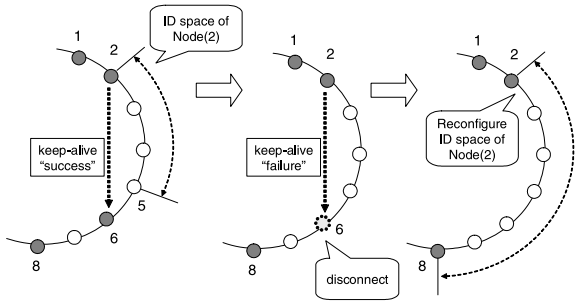


Fig. 4 Leave protocol.

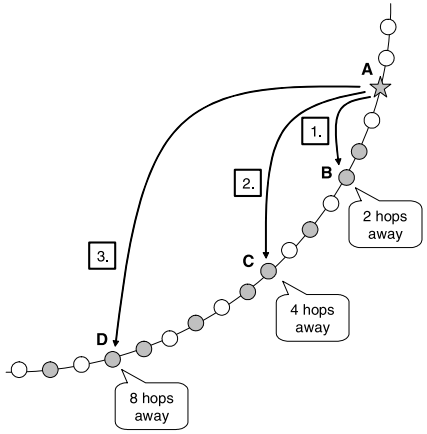


Fig. 5 Operation for creating routing table.

table.

(1) After joining the Mill network, each node has routing information of neighbor nodes. Node-A (star shape) has routing information of a node which is two hops away (Node-B). First, Node-A communicates with Node-B to get a routing information of a node which is 4 hops away (Node-C).

(2) Secondly, Node-A communicates with Node-C to get a routing information of a node which is 8 hops away (Node-D). Node-C can probably reply with information of Node-D, because Node-C also adds routing information as does Node-A.

(3) Thirdly, Node-A communicates with Node-D to get a routing information of a node which is 16 hops away. If Node-D does not have routing information of a target node, Node-A tries to communicate with Node-D again, after a while.

By repeating this operation, the size of a routing table becomes larger. Through the above processes, the maximum size of a routing table is as much as $O(\log N)$. This routing table is likely to have more entries for the closer nodes and fewer entries for the distant nodes. This list structure is called "skip-list" [15].

This routing table reduces the search cost to $O(\log N)$. This routing table also enhances the stability of a Mill network. A Mill network can recover itself to find nodes alive by using this routing table even if several nodes are disconnected at the same time.

The search method of Mill is similar to the one of Chord. Chord also adopts a skip-list search. Mill creates a routing table based on the existence of nodes. On the other hand, Chord creates a routing table based on the ID-space. In DHTs networks, a system does not need to consider the density of nodes, because a hash function assigns random IDs to nodes. On the contrary in real space, density of nodes changes by the location. Mill responds to the difference of node density by creating a routing table based on the existence of nodes. This routing table and the separation method of ID-space in a Mill network is effective for load balance.

In a Mill network, each node is responsible for a part of the ID-space. The size of IDs each node manages is determined by the distance between one node and next node. In areas where the density of nodes is high, the distance between two nodes is short. In such an area, lots of information is generated, however the size of IDs each node has is small. The size of IDs is inversely proportional to the density.

The amount of information each node has is not affected by the density of nodes because of locality of Z-ordering. Every node manages almost the same size of information, and load balance is realized in a Mill network.

On the other hand, in SkipNet and Mercury network, the load of nodes is affected by the density of nodes, because each node has a random ID and should be responsible to manage a part of the ID-space defined by this random ID. In SkipNet and Mercury network, if the density of nodes is high, a particular node needs intensively processing many queries.

Several previous works [16),17)] construct an overlay network by adopting Z-ordering and skip graphs. However, some nodes have to process much more queries than others do, because the density of data or queries depends on a portion of an overlay network and skip graphs can not control the unfairness. And these previous works integrate several axes of attributes into one axis by Z-ordering and can support multiple-attributes queries. However, all data are not distributed equally in each attribute. Consequently this integration causes another unfairness.

On a Mill network, each node has an ID related to the geographical location (not random), and unlike the original skip list a routing table is created based on the location of nodes (not the ID). Even on a real environment where the density of data at each area is different, Mill can achieve load balancing by considering the location of nodes and data. In a ubiquitous sensing environment, data management systems should support multiple- attributes queries. Mill resolves an attribute of the geographical location on its overlay network and resolves other attributes of queries by each node of Mill, in order to support multiple-attributes queries. Even though queries should be processed by geographical location at first, this restriction practically does not cause problems, because queries of sensing devices and users are distributed by geographical locations.

5. An Implementation toward Characteristics of Ubiquitous Sensor Data

To meet the requirements in Section 2, we implement Mill based on HTTP,

because HTTP meets these requirements and additionally has some good features for a ubiquitous sensing environments. This section describes the implementation of Mill.

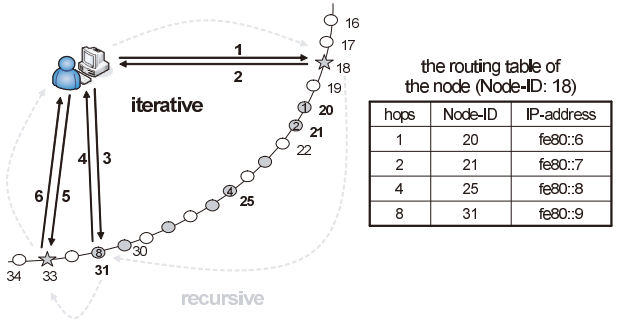
5.1 Handling the Repetition of Similar Queries

We have operated the Live E! network, in which weather information mainly is managed [18]. In this sensor network, users tend to search the same place to get constantly current status of that place, since people are interested in the situation of their home or office. In addition, sensing devices constantly measure data of the same place, because most sensors are equipped on a particular location. Consequently similar or same queries for uploading and downloading sensor data are frequently generated. In this case, relaying nodes should pass many similar or same queries from one node to another on an overlay network. This operation wastes time and network resources.

To handle these queries, we adopt skiplist with iterative routing on our overlay network. **Figure 6** shows a process of iterative routing. Unlike recursive routing, by using iterative routing, users communicate with several relaying nodes until a query reaches upto a target node. After reaching a target node, users can cache routing information (e.g., IP-address, ID-space) of the target node and its neighbor nodes, because of features of skiplist. The routing table made by skiplist enables a querying node to especially communicate with neighbor nodes of a target node.

Figure 7 shows that users and sensing devices communicate with nodes of Mill. In a Mill network, sensor data are managed based on the geographical location by nodes. Consequently, users constantly communicate with the same node, if they want to know the current status of a particular region. Caching information created by skiplist with iterative routing enables queries and sensor data to dynamically being clustered in a particular local area, and nodes and sensing devices can directly connect to a target node. In addition, this cache is effective to search the adjacent area of the target node. And sensing devices also directly upload data to the target node in the same way. A caching mechanism by iterative routing optimizes the path of queries from users and sensing devices. For this reason, the number of relayed queries can be significantly reduced.

Users or sensing devices communicate with nodes on a Mill network through HTTP. HTTP is suitable for an iterative routing, because HTTP basically consists of one request and one response. The request and the response correspond to the query and the reply of an iterative routing. Users easily get sensor data on a Mill network by some web browsers or other HTTP clients, without installing any special software. In addition, iterative routing is efficient for avoiding performance reduction caused by a time-out. Recursive routing is fast, but querying clients can not know relaying nodes. For this reason, if time-out happens, a querying node can not do anything. On the contrary, by using iterative routing, clients can communicate with all relaying nodes. A querying node therefore can



the routing table of the node (Node-ID: 18)

hops	Node-ID	IP-address
1	20	fe80::6
2	21	fe80::7
4	25	fe80::8
8	31	fe80::9

Fig. 6 Iterative routing.

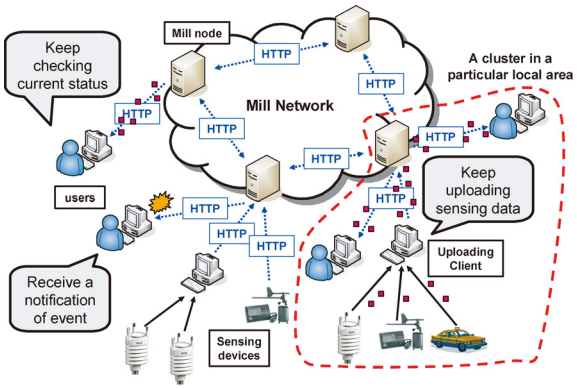


Fig. 7 Mill network based on HTTP.

find out nodes causing the time-out and change the routing path, by picking up another node in its own routing table.

5.2 Supporting an Event Driven Behavior

There is a requirement that users often want to receive alert messages. For example, they might want to receive a message when it is starting to rain or when then temperature gets over 35 degrees Celsius. To realize these alert services a polling method is useful, but this method might waste network resources. To save network resources, an event driven method is required.

To realize an event driven method, we apply a data pushing mechanism called "Comet" on an HTTP server. A server-side application does not respond to the requests from client-side applications, and the connections between a server and clients remain until the event occurs. When the event occurs, the server pushes the data to the clients through the already-established connection.

On existing overlay networks, users should install some special software to get data from overlay networks. This setup cost restricts the number of users. On a Mill network, users easily can receive event driven messages as well as get sensor data without installing any particular software.

5.3 Considering the Flexibility of System Operation

In a ubiquitous sensing environments, the number of sensing devices keeps increasing. To adapt this situation, overlay networks must consider the flexibility of system operation which includes easy installation, easy operation and the flexibility for user's environments.

To realize these requirements, we adopt HTTP as an underlying communication protocol of "Mill". Using HTTP, we can define services for responding to users and sensing devices with URLs. Mill nodes have their own software version, and URLs are managed by the software of Mill in each version. As the software of Mill is upgraded, URLs basically increase with new services. Accessing to a node of Mill, Users get lists of services (URLs) and their explanations with some web browsers. If users want to receive some services, they just specify the URLs. For this reason, users can receive services without any special software. HTTP is one of the most popular protocol and its interconnectivity is high between different sites. Adopting HTTP enables an easy operation of overlay networks. In addition, we use lots of software assets for HTTP. For examples, SSL can be used for encrypted communication and redundancy technology for web sites can be used for improving the application availability of each node.

Moreover, adopting HTTP also provides flexibility for developer's environments. For example, if we want to improve the performance of a database or add new functions, we can easily exchange a RDBMS or write additional code. The reason is that an interface is represented by a URL and URLs can be handled independently from the internal implementation. In other words, the URL hides the internal implementation of a software. For this reason, adopting HTTP provides flexibility of the developer's environment. However, it is a little difficult to practically exchange databases, because the way of accessing data is different on each system. We therefore implement an O/R mapper of Mill in order to enable almost RDBMSs to be operable on a Mill network. Mill considering easy installation, expandability and flexibility of development, and as the result Mill provides flexibility of system operation.

6. Evaluation

In this section, we evaluate the performance of Mill. We have implemented nodes of Mill by Perl. **Table 1** shows the specification of the implementation and the experimental environment. We measure the performance of downloading data, uploading data and notifying of events. **Figure 8** shows the environment of evaluations. One node on the Mill network consisting of a total 10 nodes processes lots of queries from sensing devices or users. Through this experiment, we clarify the performance limitations.

As shown in the above section, Mill realizes localities of processing sensor data besides the accessibility to the whole geographical region. On a Mill network, some clusters related to particular local areas are dynamically created. One

Table 1 Specification of Mill node.

CPU	AMD Opteron252 x 2
Memory	DDR400 4 GB
Hard Disk	Ultra320 SCSI 10000 rpm 150 GB
NIC	Gigabit Ethernet (PCI-X)
OS	Linux 2.6.17-10 (ubuntu 6.10)
Language	Perl 5.8.8
Data Base	SQLite 3.3.5

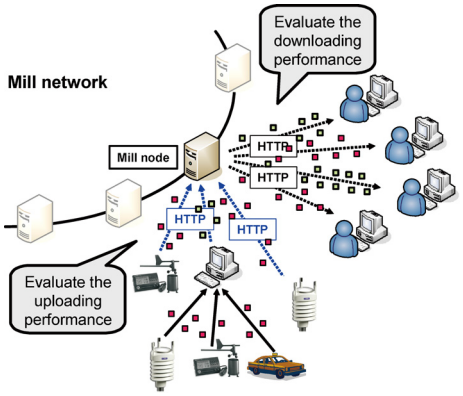


Fig. 8 Environment of evaluations.

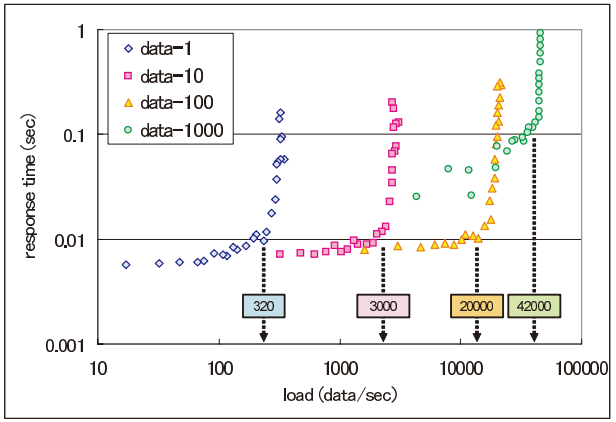


Fig. 9 Downloading sensor data.

cluster consists of one Mill node and a large number of sensing devices and users, and almost all communications are processed within each cluster. For this reason, if we evaluate the performance of one Mill node, the performance of the whole of Mill network is almost clarified.

We have evaluated the performance [1] of the routing algorithm, the messaging cost and the robustness in a Mill network on HAKONIWA [19]. HAKONIWA is a traffic simulator on which lots of cars are moving around and generating sensor data (e.g., velocity, direction) in a particular city (e.g., Nagoya, Tokyo). On this environment being close to the real world, a Mill network consisting of thousands of nodes is able to achieve its goals of logarithmic-hop routing, scalability and robustness as an overlay network. In other words, we have evaluated the performance of the overlay network (Mill) consisting of lots of nodes on the simulator. This time we therefore implement nodes of Mill, and evaluate the performance of one Mill node in order to clarify the performance of the whole of Mill network in a real environment.

6.1 Downloading Sensor Data

We evaluate the performance of downloading data. As we describe in the above section, users tend to get constantly sensor data of the same place. In most cases, users directly communicate with target nodes. For this reason, we evaluate performance limits of downloading data per node. The limitation of downloading performance at one node clarifies performance limits of the whole Mill network.

Figure 9 shows the result of the downloading performance. The "response time" means the time from when a user sends a query to a target node to when the user receives the answer of the query. One "data" consists of *ID* of sensor, *sensor type*, *time* when the data is generated, *Mill-ID* (made from latitude and longitude) where data is generated and *value* of a sensor. The example format (JSON) of one data is as follows. Users can get this data, if users access to URL (http://192.168.0.1/get/data?last=1&sensor_type=Pressure) of a target node assigned "192.168.0.1" as IP-address.

The size of one data is approximately 150 bytes.

```
{
  "id":1,
  "mill_id":"0xed01944ec1361d9e",
  "sensor_type":"Pressure",
  "time":"2007-09-24T11:06:44+09:00",
  "value":"985.2"
}
```

In Fig. 9, "data-1" means that users get one sensor data per query. One Mill

node can respond to approximately 320 queries per one second. And "data-10" means that users get 10 sensor data per query. Then, one Mill node can respond to approximately 300 queries per one second. This result shows that one Mill node can process almost 9.3 times the amount of data traffic by merging sensor data, compared with "data-1".

As users attempt to get larger data, the performance of downloading data gets better. On the contrary, response time gets worse and the efficacy also gets worse. As Fig. 9 shows, "data-100" provides better performance, considering both the downloaded data and the response time.

In the Live E! [2] network, weather sensors upload current data once a minute. Therefore, if all users want to keep getting current 10 kinds of data, one Mill node can handle 18,000 ($= 3000$(data/sec) $\div$ 10(data) $\times$ 60(sec)) users on performance limits.

Compared with other overlay networks, Mill reduces almost all relaying queries, because users can directly access to nodes of Mill. If users get certain data on a 1000-nodes network, the cost of search is approximately 10 hops on existing overlay networks. The relaying cost is almost equal to the cost of getting data, because the size of sensor data is very tiny. Mill improves the cost of search by approximately 90% on a 1000-nodes network. This improvement is also efficient if sensing devices upload data.

6.2 Uploading Sensor Data

First, we evaluate the performance of uploading data. Sensor devices directly communicate with a target node, because almost all sensors are equipped on a particular location. For this reason, we evaluate performance limits of uploading data per node.

Figure 10 shows the result of the uploading performance. One "data" consists of *the ID*, *the Mill-ID*, *the sensor type*, *the time*, and *the value*. Uploading clients use *the latitude* and *the longitude* instead of *the Mill-ID*. As sensing devices (uploading clients) put larger data, the performance of uploading gets better. In addition, poor sensing devices can save their energy, uploading several data at once.

However, "data-1000" is too large to improve the performance. In order to upload sensor data, uploading clients have to register before uploading data.

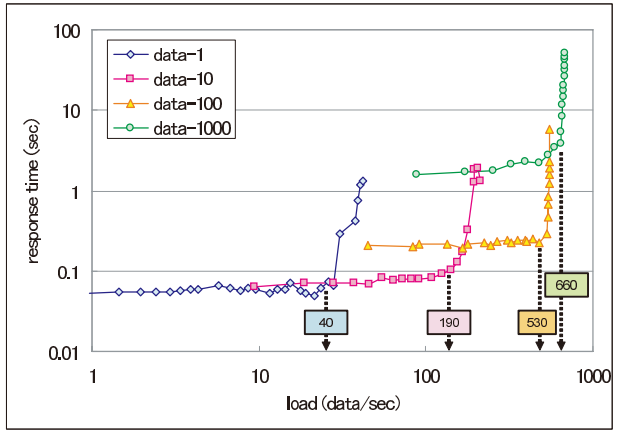


Fig. 10 Uploading sensor data.

When a node of Mill receives sensor data from uploading clients, the node checks the weather sensor data are sent by a registered sensor. This process of checking data makes the performance worse.

Secondly, we optimize the format for uploading sensor data. And we compare the performances of normal uploading and optimized uploading. Ordinary weather sensors have some functions. For example, these sensors can measure the temperature, the humidity, the pressure and several other items of the weather. Aggregating sensor data sent from the same sensor device, nodes of Mill reduces the number of verifications on sensor data.

Figure 11 shows the result of the comparison. The graph legend of "opti" means that nodes of Mill can check data blocks at once by optimized format. As the data becomes larger, the efficacy of optimization becomes better.

If a sensing device measures 10 kinds of data and send data to a node once per minute, one Mill node can handle 13,000 (≈ 220(data/sec) $\times$ 60(sec) $= 13200$) sensors on performance limits. If an uploading client aggregates 10 sensors, that means sending 100 "data" to a node per minute, one Mill node can handle 52,000 (≈ 870(data/sec) $\times$ 60(sec) $= 52200$) sensors per node.

6.3 Event Driven Messages

Mill supports an event driven method by the mechanism of "Comet". Here we

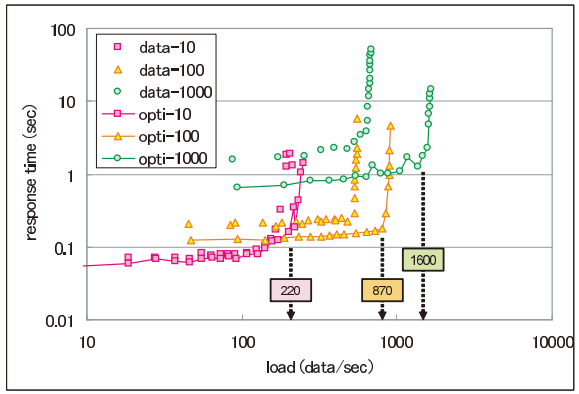


Fig. 11 Efficacy of optimized format.

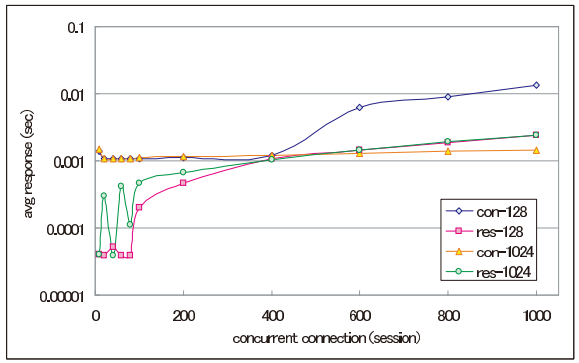


Fig. 12 Average response time.

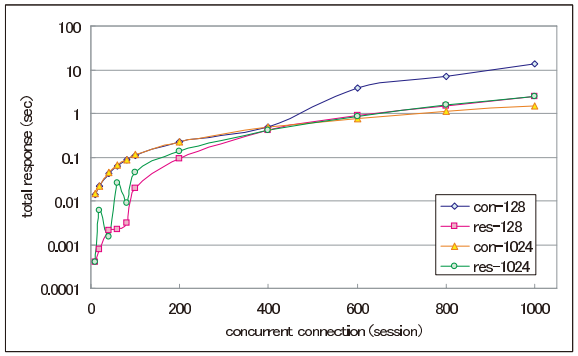


Fig. 13 Total response time.

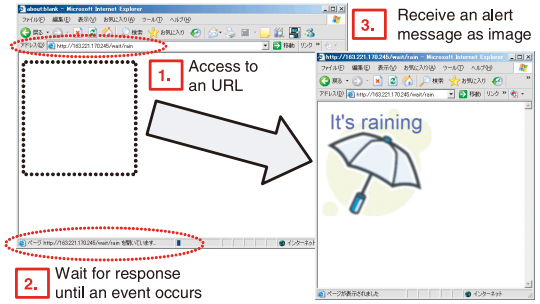


Fig. 14 Example of event driven bevavior.

evaluate the performance of the response time when a particular event occurs.

Figures 12, **13** shows the performance result if a Mill node handles concurrent connections. The "response time" means the time from when a particular event occurs to when users receives the notification of the event. In these Figures, graph legend of "con" means that users connect to a Mill node and "res" means that a Mill node responds to users. And "128" means that the maximal value of "SOMAXCONN". The legend of "1024" is also the maximal value of "SOMAXCONN". Unix operating systems defines the number of listen queue by "SOMAXCONN".

As Fig. 12 shows, the times of connection and response are almost constant and it is milliseconds. As the number of concurrent connections becomes larger, the total response time linearly becomes longer. Because Mill adopts an asynchronous IO system at handling HTTP requests. Figure 13 shows that if the number of concurrent connections is 1024, the response time is approximately 2 seconds. This result means that users can receive an alert message at least in 2 seconds. This response time is useful to lots of applications, which alert start of a rain, exceeding 35 Celsius and others.

Figure 14 shows that a user receives an alert message of a rainfall by an

ordinary web browser. Without installing any special software, users can easily receive some services.

7. Concluding Remarks

In the ubiquitous sensing environment, sensing devices can measure lots of kinds of data anywhere and anytime. To share these huge sensor data through the Internet, we implement a geographical location based P2P network called "Mill". Mill provides a distributed environment based on the geographical location and supports a multiscale region search by $O(\log N)$ messages.

Our implementation of "Mill" handles *repetition of similar queries*, supports *event driven behavior* and considers *flexibility of system operation* in a ubiquitous sensing environment. We adopt HTTP as a communication protocol and iterative routing as a routing algorithm to realize the above three requirements.

The results of our evaluation show that one Mill node can handle more than 10,000 users and 10,000 sensing devices. If we can install Mill nodes on different places on the Internet, we will be able to manage the huge amount of data in a ubiquitous sensing environment.

References

1) Matsuura, S., Fujikawa, K. and Sunahara, H.: Mill: A geographical location oriented overlay network managing data of ubiquitous sensors, *IEICE Trans. Comm.* Vol.E90-B, pp.2720–2728 (Oct. 2007).
2) Live E!: http://www.live-e.org
3) Sunahara, H., Sato, M., Uehara, K., Aoki, K. and Murai, J.: IPCar: Building the Probe Car System with the Internet, *IEICE Trans. Inf. Comm. Engineers*, Vol.J85-B, pp.431–437 (Apr. 2002).
4) Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A scalable peer-to-peer lockup service for internet applications, *ACM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp.149–160 (June 2001).
5) Zhou, B., Joseph, D.A. and Kubiatowicz, J.: Tapestry: A fault tolerant wide area network infrastructure, *Sigcomm*, Tech. Report UCB/CSD.01.1141 (2001).
6) Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Schenker, S.: A Scalable Content-Addressable Network, *ACM SIGCOMM*, pp.161–172 (2001).
7) Rowstron, A. and Druschel, P.: Pastry: Scalable, decentralized object location and routing for largescale peer-to-peer systems, *IFIP/ACM Int'l Conf. Distributed Systems Platforms* (*Middleware*), pp.329–350 (2001).
8) Harvey, N.J.A., Jones, M.B., Saroiu, S., Theimer, M. and Wolman, A.: SkipNet: A Scalable Overlay Network with Practical Locality Properties, *Proc. 4th USENIX Symposium on Internet Technologies and Systems* (*USITS '03*) (Mar. 2003).
9) Bharambe, Ashwin R., Agrawal, M. and Seshan, S.: Mercury: Supporting scalable multi-attribute range queries, *ACM SIGCOMM Computer Communication Review*, pp.353–366 (2004).
10) Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R. and Shenker, S.: GHT: A Geographic Hash Table for Data-Centric Storage, *Proc. ACM International Workshop on Wireless Sensor Networks and Applications* (*WSNA2002*), pp.78–87 (2002).
11) Karp, B. and Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, *Proc. ACM/IEEE International Conference on Mobile Computing and Networking* (*Mobicom2000*), pp.243–254 (Aug. 2000).
12) Eastlake, D. and Jones, P.: SHA-1: Us secure hash algorithm 1, RFC3174 (2001).
13) Orenstein, J.A. and Merrett, T.H.: A class of data structures for associative searching, *Proc. 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp.181–190 (1984).
14) Moon, B., Jagadish, H.V., Faloutsos, C. and Saltz, J.H.: Analysis of the clustering properties of the Hilbert space-fillingcurve, *Transactions on Knowledge and Data Engineering*, Vol.13, Issue 1, pp.124–141 (2001).
15) Pugh, W.: Skip Lists: A Probabilistic Alternative to Balanced Trees, *Comm. ACM*, Vol.33, pp.668–676 (June 1990).
16) Shu, Y., Ooi, B.C., Tan, K.L. and Zhou, A.: Supporting multi-dimensional range queries in peer-to-peer systems, *Proc. 5th IEEE International Conference on Peer-to-peer Computing* (*P2P 2005*), pp.173–180 (2005).
17) Brault, G.J., Augeri, C.J., Mullins, B.E., Baldwin, R.O. and Mayer, C.B.: Enabling Skip Graphs to Process K-Dimensional Range Queries in a Mobile Sensor Network, *Proc. 6th IEEE International Symposium on Network Computing and Applications* (*NCA 2007*). pp.273–282 (2007).
18) Nakayama, M., Matsuura, S., Esaki, H. and Sunahara, H.: Live E! Project: Sensing the Earth, *Second Asian Internet Engineering Conference AINTEC 2006*, Lecture Notes in Computer Science, pp.61–74, Springer Berlin/Heidelberg (Nov. 2006).
19) Tetsuji, H., Masaaki, S., Keisuke, U., Hitoshi, H., Kei, I., Ryota, H., Hirokazu, A. and Sunahara, H.: Hakoniwa — application developement environments for internet car systems, October 2004, *11th World Congress on ITS Nagoya, Aichi2004* (2004).

(Received October 10, 2007)
(Accepted April 8, 2008)
(Original version of this article can be found in the Journal of Information Processing Vol.16, pp.80–92.)

Matsuura Satoshi received the B.S. degree in Physics from Ritsumeikan University in 2003, and the M.E. degree and Ph.D. from Nara Institute Science and Technology in 2005 and 2008, respectively. Currently, he is an assistant professor in Graduate School of Information Science, Nara Institute of Science and Technology. His research interest includes overlay networks and large scale sensor networks.

Fujikawa Kazutoshi is an Associate Professor in Graduate School of Information Science, Nara Institude of Science and Technology since 2002. He was a visiting researcher of the Multimedia Communications Laboratory at Boston University from March 1996 through January 1997. He has been engaged in the project of bio/IT related R&D called "BioGrid Project," which consists of several institudes in Kansai area of Japan. His research interests widely cover distributed systems and multimedia systems. Now he is very interested in Grid systems for scientific simulations. He received the M.E. and Ph.D. degree in information computer sciences from Osaka University in 1990 and 1993, respectively. He is a member of IEICE, IEEE and ACM.

Sunahara Hideki received the B.S. and M.S. degrees in electrical engineering from Keio University in 1983 and 1985, respectively. He received the Ph.D. in computer science from Keio University in 1989. Currently, he is a professor in the Information Technology Center, Nara Institute of Science and Technology, Ikoma, Japan. His research focuses on multimedia communication systems, digital libraries, computer architecture, parallel processing, distributed systems, operating systems, and computer networks. He is a member of the ACM, IEEE, Internet Society, JSSST, IPSJ and IEICE. He is also a board member of the WIDE project of Japan.