A Two-tier Overlay Publish/Subscribe System for Sensor Data Stream Using Geographic Based Load Balancing

Tatsuya Fukui*, Satoshi Matsuura*†, Atsuo Inomata*, Kazutoshi Fujikawa*
*Graduate school of Information Science, Nara Institute of Science and Technology, Japan
Email: {tatsuya-fu, matsuura}@is.naist.jp, {atsuo, fujikawa}@itc.naist.jp
†National Institute of Information and Communications Technology, Japan

***Abstract*—Publish/Subscribe mechanism is one of the key technologies for handling a real-time data stream in decentralized environments. While recent studies have showed that publish/subscribe systems should support not only data filtering but also in-network data processing, the scalability issue is addressed here due to data concentrations on specific nodes. To solve the scalability issue, we propose a Publish/Subscribe system consisting of a two-tier overlay network: one overlay network is designed as a geographic overlay network, which is used to maintain fundamental information, and on the other hand, data transfer networks are constructed to perform in-network data processing and dynamic load balancing. By using these two networks, we were able to implement a geographic based dynamic subscription re-allocation mechanism. This mechanism enables achieving scalability and avoiding data/load concentrations on large-scale Publish/Subscribe systems. We present an evaluation of our approach through simulation. We compare our approach with another DHT based Publish/Subscribe system, focusing on data transfer during subscription. Our results show that our dynamic subscription re-allocation approach avoids data concentration on specific nodes.**

I. Introduction

Publish/Subscribe systems are based on content-based networking technologies, which have been developed as event notification systems. Publish/Subscribe systems consist of loosely coupled nodes. These nodes asynchronously transfer data to other nodes based on the rules of subscriptions, and realize push style communication. In recent studies, several Publish/Subscribe systems adopt overlay network technologies to manage ID space and to achieve scalability. Therefore Publish/Subscribe systems are useful for large-scale real-time data delivery in a decentralized manner.

However, traditional Publish/Subscribe systems only filter out data in the delivery networks. Subscribers have to carry out data processing (e.g., comparison, averaging and other calculations) after gathering data by themselves. However, data concentration occurs on particular nodes. In order to remove these concentrations and reduce transfer data, it is required that nodes collaborate in filtering and calculating data on Publish/Subscribe networks. On the other hand, this collaboration possibly increases load concentrations related to data processing. To achieve scalability and avoid data/load concentrations, filter/calculation processes should be divided and reassigned on appropriate nodes.

In this work, we adopt the architecture of geographical based overlay networks to construct a Publish/Subscribe system. Features of our system are efficient contents search, in-network processing, and a self-organized load balancing mechanism. To achieve scalability, we propose a multiple network consisting of two parts. One part is for information management and the other is for data transfer during subscription. The network of information management operates network stabilization, data searching data and node information management. The transfer network focuses on delivering data only during subscription and when constructing simple topologies. On the transfer network, nodes can perform in-network processing and easily divide/reassign calculation processes. In addition, our system has geographical based ID space, and we propose a mechanism of a dynamic calculation reassignments method by using this geographical ID space. These features of our system enable achieving scalability and avoid data/load concentrations on large-scale Publish/Subscribe systems.

The rest of this paper is structured as follows. Section II presents the related work, Section III provides the basic architecture of our publish/subscribe system, Section IV describes the experimental results, and Section V concludes the paper.

II. Related work

A Publish/Subscribe system is an event notification system which delivers data by communicating among processing nodes under distributed environments. There are two types of nodes on Publish/Subscribe systems: publisher and subscriber. Publishers generate data synchronously, and subscribers register subscriptions which describe the requirements of the subscribers.

Scalable Internet Event Notification Architectures (SIENA) [1] is one of the Publish/Subscribe systems. SIENA was developed to reduce network traffic. SIENA filters data close to the data ' s publisher. Through a broadcast subscription process, SIENA constructs a distribution tree and routing tables. Each server publishes data and assigns subscriptions according to routing tables. In addition, SIENA can adjust the transition path dynamically by overlapping publications among subscriptions. The delivery methods of SIENA are good, and these features contribute to reducing network traffic. However, SIENA only performs filtering and routing, and does not have a

processing mechanism (e.g., averaging, counting) in a delivery process. Routing tables based on broadcasting cause much network traffic. Therefore SIENA is not suited for handling stream data.

Domain space mapping and distributed 2-dimensional trees over DHT [2] is one of the overlay-based Publish/Subscribe systems. This study focuses on a particular attribute. It maps the value range (Min, Max) from a subscription's requirement to corrdinates on the X-Y axes in ghe logical space called the "Domain Space". Also, an event generated by the publisher is mapped to Domain Space as coordinates $(Value, Value)$. In this Domain Space, publication will be delivered to subscribers if subscription coordinates are in the upper-left side of the event's coordinates.

In addition, this research builds a 2-dimensional tree structure to search for all subscriptions in the Domain Space. The space is divided on the x-axis an the d y-axis alternatively as parts of the tree, and all the subscriptions are stored in leaf nodes of the tree. When a new event is generated, leaf nodes transfer data to a subscriber directly. If a leaf node becomes overloaded, the area of this node is divided into two half spaces and two new leaf nodes are assigned to the divided area. This method is efficient for scalable search and delivery in distributed environments. On the other hand, this method needs a lot of extra nodes for load balancing, because adding new nodes on leaf nodes is the only way for load distribution. This method is not suitable for managing sensor networks where many sensors constantly generate a large number of events.

A stream processing engine (SPE) is a system for handling stream data. Stream data is continuous and high-frequency data, such as sensor data or market stock data. Borealis [3] is one of the distributed style SPE. This system is based on Aurora [4] and Medusa [5].

Borealis generates an operator tree for each query. In this tree, contents and sequences of query processing are defined. After generating the operator tree, each operator is assigned to distributed computing resources. Borealis supports pre-defined operators located with fixed operators. This is inefficient operator placement, and a non-dynamic stream management style remains a problem for a large scale system with thousands of nodes and queries.

[6][7] are an extension of Borealis focusing on dynamic optimization for stream management. [6] proposes a stream-based overlay network (SBON). SBON has a layer between a stream processing system and the physical network. This layer manages operator placement for SPE. This research is based on a cost space, an abstract representation of the network and on-going streams, which permits decentralized, large-scale multi-query optimization decisions. SBON uses a spring relaxation algorithm to optimize operator allocation. This method requires that each node exchange a meta-message every time, so there is still a scalability issue. [7] solves distributed load shedding problems as a linear optimization problem. For this problem, they propose two alternative approaches to a solution : a solver-based centralized approach, and a distributed approach based on meta-data aggregation and propagation. They keep the CPU load lower by adjusting the input rate of the stream. The centralized approach takes too much time to solve each operator placement, and the state is changed during calculation. The distributed approach makes the tree adjust the input rate and send meta-data from the leave node to the root node. Scalability is not guaranteed if there are hundreds of nodes or more.

There is research on large scale Publish/Subscribe systems for a sensors network[8]. The system consists of 4 types of nodes, the first being the publisher which generates data. The second type of node is the subscriber which receives the data requested. The third type of node is the processing node, which performs delivery and calculation of data requested. The fourth type of node is an administration node which performs maintenance management of the system. This system builds a delivery tree. The root node of the tree connects to the subscribers, and leaf nodes connect to the publishers. This system can not only deliver raw data, This system can perform not only the delivery of raw data but can also calculate these data inside the Publish/Subscribe network. This system can also divide and reassign the processing operator based on subscriptions, if a processing node becomes over loaded. A subscription consists of a chain of filters and calculations. This structure for a subscription is useful when subscriptions are divided and merged. Processing nodes can divide and reassign subscriptions based on the subscriptions ' attributes (e.g., geographical location, sensor ID and etc.). From this research, we found that the main factor causing high CPU load is the data input/output rate during data transition in Publish/Subscribe networks. This research shows the direction of a load balancing method for in-network processing of the Publish/Subscribe system. It has not, however, considered the data transition topology and the dynamic optimizing method. In addition, the system depends on the administrator node for various processes, so that the administrator node is the one critical point of failure.

[9] investigated differences in the transfer performance focusing on the topologies balanced binary Tree, star, and three-step tree. According to this study, these topologies ware selected to compare topologies having a different number of joint nodes which have many edge nodes when filter operators are used to exclude unnecessary sensor data in an early phase of delivery. This research shows that a three-step tree was much better for in-network processing due to the flexibility of its methods of operator allocation and ease of constructing the tree.

III. System Architecture

As shown in Figure1, our system has three components: processing nodes, publisher and subscribers. Processing nodes are connected to each other. They operate forwarding and processing of sensor data according to subscriptions, which are determined by the target data type, the processing, and the requiring area. We define the publisher nodes as sensor devices such as weather sensor and river level sensor, that

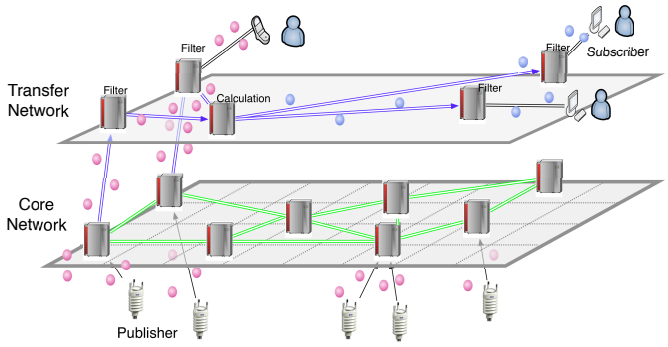

Fig. 1. System Overview

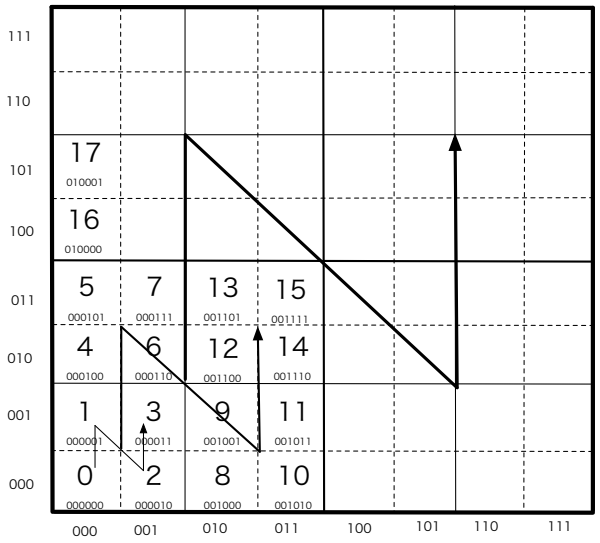

Fig. 2. Z-ordering ID Space

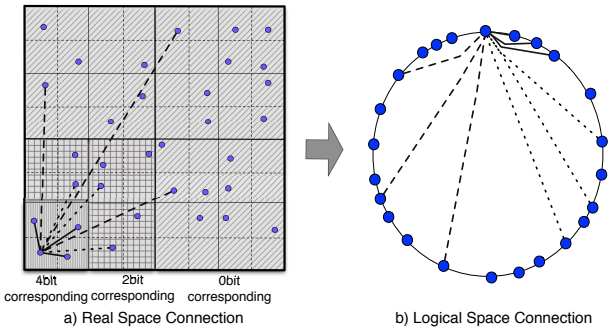

Fig. 3. Connection Image

are connected to the networking devices. Finally, we define subscriber nodes as those clients who require the data.

Our system construct is constructed with two types of networks: a *core network* and a *transfer network* (Figure1). The *core network* manages the information of processing nodes, publisher, and routing path. *Transfer networks* are built on the core network to deliver data to subscribers during the subscription period. Thanks to this multiple network architecture, we achieve flexibility and scalability in our system. The core network is useful for dealing with a huge number of publishers and processing nodes, so it should to be able to manage data with astute communication and quick searching. In contrast, the transfer network is specialized in the transfer of subscribed data, so this network focuses on only a small quantity of nodes which are related to the subscription. In addition, the transfer network needs to gather data to perform processing based on the subscription without too much data concentration, so readjustment of the delivery path and modification of the place of processing are needed to operate. Therefore the roles of these two networks are totally different.

A. Geolocation based Overlay Network

The core network is constructed as a geographic-based overlay using ID space with a Z-ordering function. Each processing node and publisher is assigned a geographical-location based ID calculated by the Z-ordering function. Z-ordering enables mapping multidimensional data (e.g., 2D surfaces) to one dimension without losing locality in the original dimension. The world is split into a grid cells based on latitude and longitude, as consecutive IDs, as depicted in Figure 2. Thanks to this ID space, we can look up data with range-based queries. In our system, subscribers request geographical related information, and therefore this feature contributes to reducing the number of search queries [10].

Processing nodes are connected to single-hop neighbors in the ID space. At the same time, they are also connected to relatively distant neighbors according to the skip graph algorithm [10]. This algorithm is based on matching ID length. This utilize the characteristic of the Z-ordering area division mechanism, such as the division of neighboring areas into consecutive IDs. Figure 3 shows how nodes are connected in the physical 2-D space and in the Z-ordering logical ID space. The first two bits of the ID of a node match the first two bits of the ID of the other nodes. For example, 001010 (10) has an initial four bits matching with 001011 (11) and 001000 (8), and an initial matching two bits with 000110 (6), 001110 (12). We focus on the last two bits of matching IDs. These two bits have four different patterns: 00, 01, 10 and 11. For example, 001010 (10) has different pattern areas: 0000xx, 0001xx, and 0011xx. With this algorithm, our system can find information in $O(\log(N))$ [10].

Processing nodes manage a part of ID-space. Each node has the responsibility to handle an ID-space which is larger than the node's own ID and smaller than next larger node's ID. When publishers are in a node's covering space, they send their data to the node, and the node manages the information (ID, data type) of publishers. Also, when a new processing node joins the network or an existing node leaves the network, the node's covering ID-space will be re-arranged in the same manner.

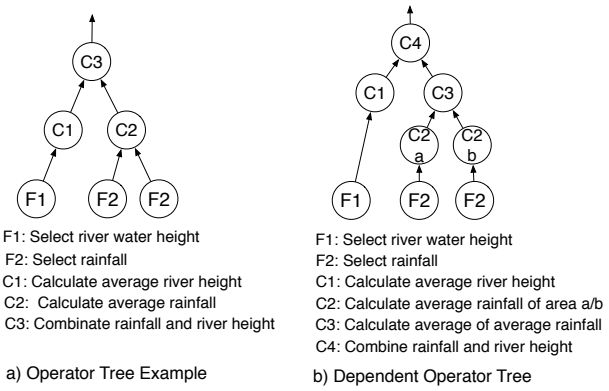

Fig. 4. Example of Operator Trees

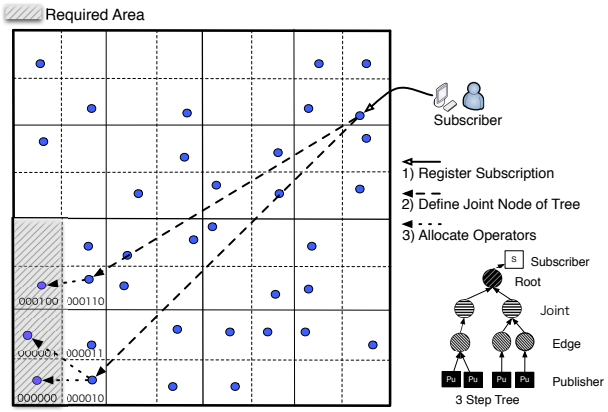

Fig. 5. Initial Operator Placement Algorithm

B. Transfer Network with Dynamic Optimization

We constructed a network which specialized for data transfer during the time of subscription. The purpose of this network is to facilitate readjustment of the delivery path and modification of the place of processing.

When we think about construction of transfer networks, we need to consider the composition of the subscription. The subscription is composed of operators, of which there are two kinds: *Filter and Calculation*. A filter operator defines the passage of data according to a predicate (e.g., data type, data size, and value size). While the calculation operator does averaging, aggregation and a general calculations. The transfer tree is constructed combining these operators. For example, a alarm system for the rise of river-water can predict danger by combining operators.

One of the operators operate the averages the rain precipitation measured at the river upstream, and another calculates the average height of the river-water level. The operator tree in this case is represented as (Figure 4(a)). Some operators have an order or dependency. The prerequisite of the dependent process is a follows: the processing operator has both types of data inputs, which are the output result of calculation operators that include other data types of streams. All patterns of dependent operators are a combination of these conditions. For example, one is the average rise of river-water, and another is the average of precipitation in area A and the average precipitation of the area B (Figure 4(b)). In this case, we need to consider the operators allocation patterns.

The main difference between the transfer network and the core network is the routing path. We make a routing path only for data transfer while the subscription is in the transfer network. This is in contrast to SIENA [1] and other research [11] [12]. This is because it is obviously more efficient to transfer data from edge nodes to the root of the trees node directly without broker nodes. However, in this case the root node may be overloaded due to too much data input. Therefore, we provide joint nodes as a buffer for data merging and processing. This Edge-Joint-Root style tree we call a three-steps tree [9].

The research [8] constructs a multistage tree as a data transfer topology, but their method is not maintained once the tree has been constructed. If some nodes are overloaded during data transfer, their system moves operators to the leaf side nodes. This movement of operators has the effect of decreasing the granularity of the data input. A multistage tree is useful to deal with complex operator allocation because it has many conjunctions for operator placement, but the tree has a scalability issue. When a node of our 3-step tree is overloaded, a new joint node is simply added and the overloaded node moves operators to the new joint node. This load shedding approach is easily performed.

Another advantage of th three-steps tree approach is that we can adjust the combination of operators easily. For example, when we need to monitor a moving typhoon, we need to change the monitoring area to follow the typhoon. In this case, the three-steps tree can be modified quickly by simply cutting the no longer necessary joint of the tree which is no longer necessary. Then, the system finds a new joint, which covers the new area's edges. On the other hand, multiple trees of complex construction cannot cope with this typhoon, because the topology of such trees cannot be readjusted, and too much time is required to construct a new tree.

The three-steps tree is constructed as follows:

1) **Initial Operator Placement**
   As a first step of the transfer, operators (filter and calculation) are placed on processing nodes. This step is composed of the following 3 sub-steps; register the subscription, define the joint node and allocate operators, as shown in Figure 5. After these steps, the proposed system adjusts the topology of the transfer network and the operator's location to avoid an overload of the processing nodes.
   a) *Subscription Registration*: The subscriber sends subscriptions to the geographically neighbor processing node. The selected processing node is assigned as the root node of the transfer tree for this subscription. The root node converts the requested geographic area to a range of IDs in the Z-ordering

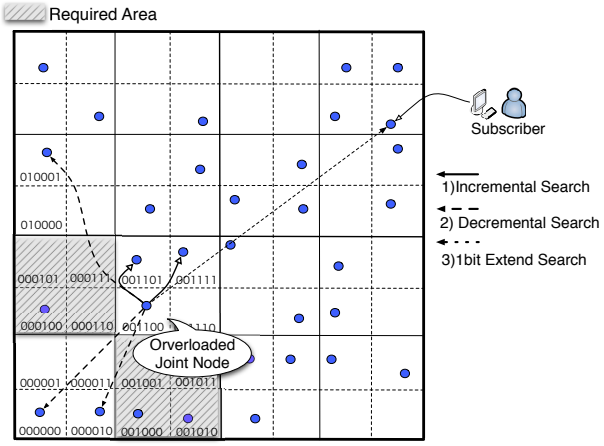

Fig. 6. New Joint Search Algorithm

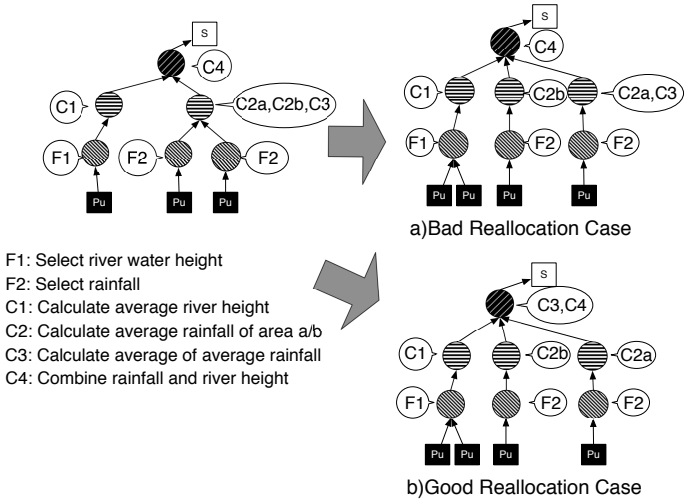

Fig. 7. Operator Reallocation Pattern

ID space. After resolving the IDs, the root node looks up a joint node, as described in the next sub-step.

b) *Joint node assignment*: In the *joint node allocation* step, a joint node is allocated to each requested geographical area (i.e., cells in Figure 2). A joint node is selected in the group of nodes with IDs larger than request range of IDs. If among the candidates with the smallest ID is capable of working as a joint node in terms of CPU load, the root node specifies that node as the joint. The root node repeats this step until the joint node is specified. If the resolved ID-space form latitude and longitude is one, the joint node is only one. If there are multiple ID-space groups, there will many joint nodes as ID-spaces.

c) *Operator Allocation*: In this step, the root node sends the subscription to the joint node after parsing the subscription into a set of separated operators. Then, the joint node re-allocates operators to the other nodes as an initial allocation. Filter operators are mainly allocated to the edge nodes of the transfer trees to reduce redundant data forwarding [8] [9]. On the other hand, calculation operators are assigned to joint nodes. When the joint node cannot cover the required data or when summarization of the processing result is needed, operators are allocated to the root node.

2) **Operator Reallocation**

When a joint node is overloaded, it node searches nodes around itself for re-allocation of operators. To re-allocate operators, the overloaded joint node selects the node, which has an ID larger than the ID of the over-loaded node. The selected candidate nodes checks its own CPU time to answer whether re-allocation of the operator is acceptable or not. If acceptable, the candidate becomes a new joint of the tree. Then, the overloaded joint node sends the operator's information to the new joint node, transmits the new destination to the edge node, adds a new source to the root node, and finally the overloaded node sends operators to the new joint node. Incidentally, the reallocated operator's covering area must be different from, or having no dependency on, operators which are assigned to the overloaded node. If the candidate node cannot allocate the new operator, the overloaded joint node asks the processing node of the next larger ID. This process is repeated until a new joint node is defined. When the matching ID length of a candidate node differs from that of the overloaded joint node during the incremental candidate searching process, the over-loaded joint node searches the candidate node from smaller ID-space than over-loaded node's ID. If the new joint node is not defined in a smaller ID-space, the overloaded node searches for a new joint node in a wider ID area in which the length of the ID is decreased one bit (Figure6).

When operators have some order relations or dependencies, combinations of operators must be taken into account for proper results.

Figure 7 shows a good case and a poor case of operator reallocation. Here, overloaded joint nodes have operators $C2a, C2b, C3$. Operator $C3$ requires calculating the results of $C2a$ and $C2b$. In Figure 7a), the new joint node has $C2b, C3$, so $C3$ cannot get the result from $C2a$. This is a bad example of operator re-allocation. In contrast, $C3$ can get the results from both $C2a$ and $C2b$ in figure 7b).

This is the ideal operator re-allocation example of this operator set. In this operator set, the overloaded joint node needs to send the operators which is requiring multiple source type to root node, and send single source type operator to new joint node. The joint searching algorithm is the same as initial joint selection algorithm. Basically, the processing load of a root node does not

increase because the root node receives a processing result. The processing result is aggregated as usual. Moreover, the root nodes only send a result to the subscriber , this operation is not so high loaded.
When a root node is overloaded, the root node sends a new aggregation operator to the joint node that is sending data at a high frequency. This new aggregation operator makes the joint node buffer sending data, which reduces the output rateby half.

IV. Evaluation

In this section we present the results of a simulation study of our system. We measured the effect of our load balancing mechanism and system scalability. We focus on transfer or messages during subscription delivery, because it is the primary factor in the CPU load. Maintenance messages are also important, but these have a lower data frequency than the sensor data stream because sensors continuously generate data more than once a second. Therefore we focus on only data transfer according to the subscription. In addition, we compare the results with another Publish/Subscribe system, which is based on an overlay network based on *Domain Space Mapping* (DSM-Pub/Sub) [2]. Our system and DSM-Pub/Sub have such similarities such as an overlay based Publish/Subscribe system with a dynamic load balancing mechanism. But there are differences between our system and DSM-Pub/Sub in data transfer topology and in the load balancing mechanism. Basically, we use the Filter-Calculation-Merge flow subscription in our system, but DMS-Pub/Sub transmits data directly to the subscribers after filtering data in the edges.

A. Simulation Details

Our simulation consisted of three stages. First, we checked the effect of the publisher ' s population change. Then, we evaluated the performances of load balancing. Finally, we install an in-network mechanism in to DSM-Pub/Sub and investigate the difference between these systems.

In this simulation, we assumed the use case was weather monitoring covering some cities. The parameters in table I show the number of processing nodes (num_nodes), the number of publishers (num_pubs), the publishing rate (pub_rate), and the data size rate remaining after processing the filter operator ($rate\_fil$) and the calculation operator ($rate\_calc$). These parameters are sufficient to get detailed information on a range of cities. Processing nodes and publishers are arranged at equal intervals. We use the geographical information as attributes of domain space in DSM-Pub/Sub. Also, we use our core network as the base DHT for their system. The geo-location based ID fits in their mechanism because it needs some kind of continuous space to manage information. Therefore DSM-Pub/Sub can use the same ID-space as our system without any changes. Each of the subscription comes from a different subscriber and requires a different areas, but the systems use the same subscriptions.

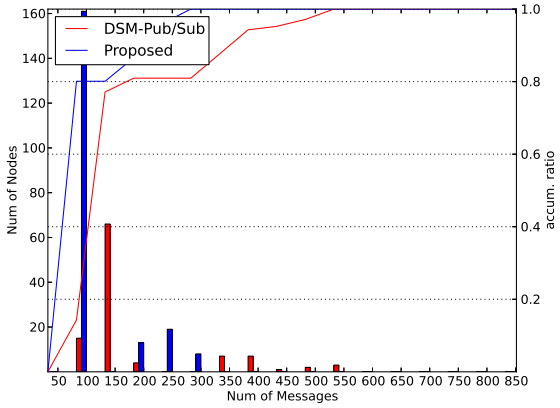

Fig. 8. Distribution of I/O messages (without load balancing, $pub\_rate = 1$)

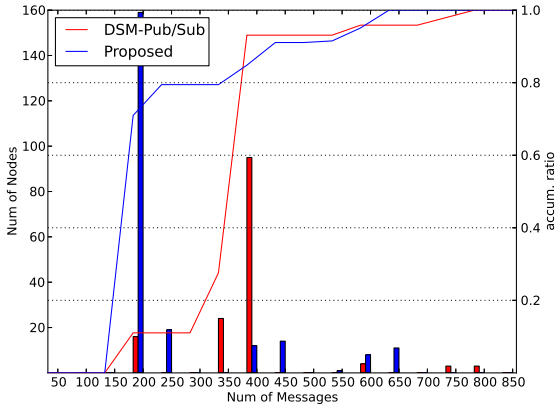

Fig. 9. Distribution of I/O messages (without load balancing, $pub\_rate = 3$)

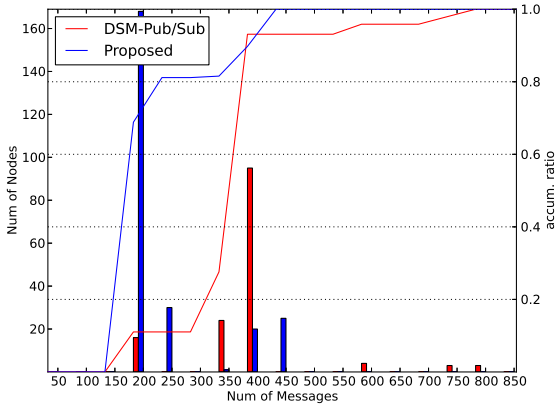

Fig. 10. Distribution of I/O messages (with load balancing, $pub\_rate = 3$)

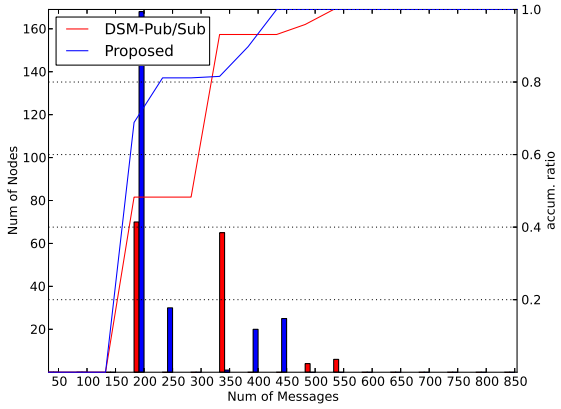

Fig. 11. Distribution of I/O messages (with load balancing, $pub\_rate = 3, fil\_rate = 0.25$)

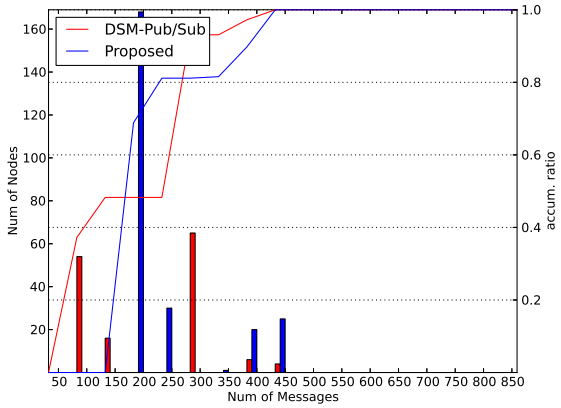

Fig. 12. Distribution of I/O messages (with load balancing, $pub\_rate = 3$, DMS-Pub/Sub with caluclation)

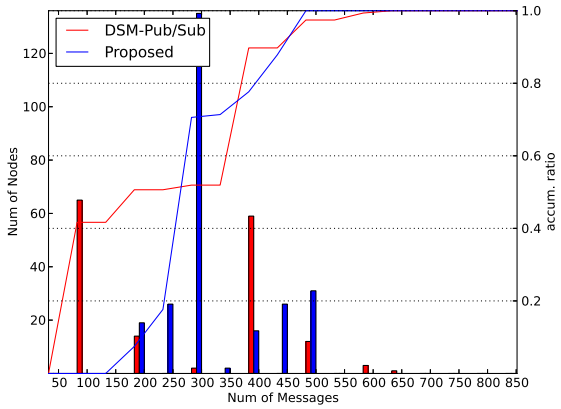

Fig. 13. Distribution of I/O messages (with load balancing, $pub\_rate = 4$, DMS-Pub/Sub with caluclation)

TABLE I
SIMULATION PARAMETERS

num_nodes	500 (num)
num_pubs	20000 (num)
pub_rate	1, 3, 4 (data/sec)
$num\_subscribers$	20 (num)
$rate\_fil$	0.5, 0.25 (%)
$rate\_calc$	0.1 (%)

B. Simulation Results

Figure 8 is the result of 20 Subscriptions. This figure is one of the criteria for this simulation. We also measured the number of messages at two or three times of the publishing rate case. Figure 9 equals three times the publishing rate. Figures 8 and figure 9 show the same tendency. The message size of DSM-Pub/Sub at the lower x-axis side is twice that of our system. This is because DSM-Pub/Sub uses binary tree type topology as the data managing topology. This means that their system can use at most only half of a processing node as the edge node. Therefore edge nodes of DSM-Pub/Sub have to manage twice the number of publishers compared to the edge nodes in our system. This difference becomes remarkable when the data volume is increased.

Figure 9 does not include the load-balancing mechanism. In figure 10, we use the load-balancing mechanism with the same publishing rate and subscriptions as shown in Figure 9. We define the threshold of load balancing as 500 messages. In figure 9, the numbers of messages increases in proportion to the publishing rate. In contrast, figure 10 shows that no processing Node exceeds 500 in our system. This is thanks to the functioning of our load balancing method and the dynamic finding of operator re-allocation nodes. DSM-Pub/Sub also has a load balancing mechanism so that edge nodes can reduce by half the dealing with data with every load balancing operation, but it only works at the edge nodes. Therefore the root node receives the same data size as it did before the load balancing, because the load balancing mechanism of DSM-Pub/Sub only divides the covering area of the edge nodes. In addition, their mechanism requires the addition of a new node, because most nodes are already used for constructing the binary tree. Adding a new node for each load balancing operation is clearly inefficient. Furthermore, one of their load balancing operations can save only a narrow area by adding a new node to the specific point. Our load balancing mechanism, on the other hand, can collaborate with nodes around an overloaded node and avoid a sudden heavy load on a specific node.

Figure 11 is a case of a high filtering rate at the edges. This figure is also three times the publishing rate of the criterion pattern. In this case, DSM-Pub/Sub compares favorably with our system.

Finally, we installed an in-network calculation mechanism in their system. Figure 12 and figure 13 are the results of in-network calculation of both systems. Figure 12 is three times the publishing rate, showing a reduction in the large message size nodes of DSM-Pub/Sub. The peak message size of their

system is less than that of in our system. Figure 13 is four times the publish rate. In this result, the peak message of their system was increased, but our system kept the proper data size range. However, their peak node was also still lower than the non-calculation version of the peak load node. However their load balancing is only for the edge node, and the root node's message size will be increased straightforward if the publish message size increases. In addition, we performed simple cases of subscription such as calculating the single data average. Their system gives good results in for simple subscriptions, but they cannot adapt to complex dependent subscriptions because their processing node only covers a static area and cannot collaborate between processing nodes. This complex subscription durability shows one of differences between our system and theirs.

V. Conclusion

In this paper, we introduced a scalable geographic Publish/Subscribe system. The system provides a mechanism of in-network processing and dynamic load balancing.

Our system uses two types of networks. One network focuses on managing fundamental information such as publisher and processing nodes, while the other network focuses on transfering data for subscriptions. Using geographical information, the system can search contents and divide or reassign overload operators among neighboring nodes. The transfer network in this system is based on a three-steps tree. Edge nodes are suitable for allocating filtering operators, while joint nodes are the proper place to allocate the calculation operators. With this topology and optimization based on geographical locations, our system can avoid overloading processing nodes.

Simulation results showed that our system can transfer sensor data while avoiding load concentration. This indicates that the load-balancing mechanism was effective in the simulation scenarios. The next step in our research will be to investigate various query patterns to improve the dynamic operator mapping mechanism to avoid overload cases.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 24700068

References

[1] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, pp. 332–383, August 2001.

[2] W. Li and S. Vuong, "Towards a Scalable Content-Based Publish/Subscribe Service over DHT," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE, 2010, pp. 1–6.

[3] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. B. Zdonik, "The Design of the Borealis Stream Processing Engine," in *Second Biennial Conference on Innovative Data Systems Research*, January 2005, pp. 277–289.

[4] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik, "Monitoring streams: a new class of data management applications," in *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002, pp. 215–226.

[5] S. B. Zdonik, M. Stonebraker, M. Cherniack, U. Cetintemel, M. Balazinska, and H. Balakrishnan, "The aurora and medusa projects," *Proceedings of IEEE Data Eng. Bull.*, pp. 3–10, 2003.

[6] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer, "Network-aware operator placement for stream-processing systems," in *Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on*, april 2006, p. 49.

[7] N. Tatbul, U. Çetintemel, and S. Zdonik, "Staying fit: efficient load shedding techniques for distributed stream processing," in *Proceedings of the 33rd international conference on Very large data bases*, ser. VLDB '07. VLDB Endowment, 2007, pp. 159–170.

[8] R. Miyagi, S. Nguchi, S. Matsuura, A. Inomata, and K. Fujikawa, "A divide and merge method for sensor data processing on large-scale publish/subscribe systems," in *The 3rd Workshop on Enablers for Ubiquitous Computing and Smart Services (EUCASS)*, jul 2012.

[9] T. Fukui, S. Noguchi, S. Matsuura, A. Inomata, and K. Fujikawa, "Analysis of transmission topology and operator location strategy of publish/subscribe system for sensor network," in *Multimedia, Distributed, Cooperative, and Mobile Symposium*, no. 4A-4, 2012.

[10] S. Matsuura, K. Fujikawa, and H. Suahara, "Mill: A geographical location oriented overlay network managing data of ubiquitous sensors," *IEICE TRANSACTIONS on Communications*, vol. E90-B, no. 10, pp. 2720–2728, 2007.

[11] M. Matos, A. Nunes, R. Oliveira, and J. Pereira, "Stan: exploiting shared interests without disclosing them in gossip-based publish/subscribe," in *Proceedings of the 9th international conference on Peer-to-peer systems*, ser. IPTPS'10. USENIX Association, 2010, pp. 9–9.

[12] S. K. Gero Mühl, Arnd Schröter, Helge Parzyjegla and J. Richling, "Stochastic analysis of hierarchical publish/subscribe systems," vol. 5704/2009, 2009, pp. 97–109.