A method of sensor data processing on large scale publish/subscribe systems

MATSUURA Satoshi †‡ matsuura@is.naist.jp
MIYAGI Ryota † ryota-m@is.naist.jp
NOGUCHI Satoru † satoru-n@is.naist.jp

OCHIAI Hideya †‡ jo2lxq@hongo.wide.ad.jp
FUJIKAWA Kazutoshi † fujikawa@itc.naist.jp
SUNAHARA Hideki †† † su@is.naist.jp

† Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara 630-0192, Japan
†† The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
††† Graduate School of Media Design, Keio University
4-1-1 Hiyoshi, Kokohoku-ku, Yokohama-shi, Kanagawa 223-8526, Japan
† National Institute of Information and Communications Technology,
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan

Abstract: As sensor and wireless network technologies are developed, it is expected that we will be able to gather lots of sensor data from various sensors on globe. Combining/aggregating these sensor data, many useful applications could be developed. However, it is difficult to aggregate sensor data and secure scalability, because in some case too much sensor data concentrate on some specific nodes. Then, we add data processing components onto content-based networks, considering division and reassign of processing components. Content-based networks provide publish/subscribe system in distributed environment and appropriate for notifications and alerts. If content-based networks process published data and create meaningful information, this system enhance the application fields in ubiquitous sensing environment. In this paper, we model sensor data calculation process and describe how to add data processing components onto content-based networks. On the contrary, data processing components cause concentrations of sensor data. We also describe load distribution mechanism of these components. Performance evaluations of our implementation shows that load distribution mechanism works well, and proposed system secure the scalability by adding data components.

Key Words: large scale sensor networks, publish/subscribe system

1. INTRODUCTION

Content-based network[1,2] provides publish/subscribe system. This system is useful to provide event driven mechanisms for notifications and alert messages. Recently, various types of sensors are released and installed everywhere. If CBN (Content Based Network) could manage these huge data on large scale sensor network, event driven applications significantly enhance their utility values. Many large scale sensor networks are proposed, but almost existing systems only manage raw sensor data and these data are provided by pull technology.

In this paper, we use CBN to manage a large scale sensor networks and add a data processing component onto CBN. If a system filter/calculate sensor data on the way of publish/subscribe processes, users could receive more valuable data and application fields could be expanded. For example, if we receive alert messages related with integration values of rain/electricity on a particular area, we take flexible measures to cope with floods/power failures. If we could constantly receive short-range weather forecasts of a local area, these information is useful for businesses(e.g., construction works, road works) and our daily lives(e.g., washing clothes, shopping). On the contrary, a data processing component probably causes high-loaded, because many published data are concentrated on this component. On the proposed system, a data component divides calculation processes into small parts, and other data components cooperatively calculate these small parts. Our proposed system secure scalability because of these cooperative calculation processes.
The organization of this paper is as follows. Section 2 describes system architecture of the proposed system and its load distribution methodology. Section 3 shows performance evaluations of our implementation. Section 4 shows related work. In section 5, we summarize this paper.

2. DATA PROCESSING ON PUB/SUB SYSTEMS

A purpose of this work is to process published data on content-based network. On our proposed system, routers define routing paths from publishers to subscribers and create data components which have both publish and subscribe mechanism. Data components processes sensor data, so that our system provides data processing mechanism besides publish/subscribe mechanism. In this section, we describe a methodology of the proposed system and its features.

2.1. System Architecture

Figure 1 shows the system architecture. Our proposed system is based on CBN. CBN primarily has a function that manages message transfer from publishers to subscribers. Routing paths are defined by broadcast tree. Independent of this function, our proposed system manages assignment of data components. Data components have both a publishing function and a subscribing function. Data components subscribe several published data and process these data in order to republish more valuable data. For example, data components filter lots of rain data from publishers and alert heavy rain messages in a particular area.

2.2. Data Format and Subscription Rule

Data formats of raw sensor data and calculated data are the same. This is because, subscribers receive data from publishers, not considering differences between raw sensor data and calculated data. This abstraction is useful to filter or calculate data repeatedly through several subscription rules. Figure 2 is an example of data format. "dataID" means sequence number of sensor data. "dataID" is useful to manage the order of sensor data and we have plan to implement traceability mechanism into our system with "dataID". We determine the data format, referring the Live E! [3] and sMAP[4] data formats. Live E! data format mainly handle weather information and sMAP has a concept that data format should be simple and easy to use.

Users (subscribers) make subscription rules consisting of mainly "publishers", "filter" and "calculation". Figure 3 is an example of subscription rules. Users define a observation area by "publishers" tag. The structure of "publishers" tag is array and users select publishers by location, ID and sensor types. If users need data processing, they use "filter" tag and "calculation" tag. In "filter" tag, users define ranges and sensor types. In "calculation" tag, users select calculation operators. Proposed system provides 5 operators (average, max, min, count, range). On our system, data processing is represented by combination of "filter" tags.
and "calculation" tags. "filter" / "calculation" tag determines just one next process by "next" tag. In other words, data processing never branches out more than one "filter" / "calculation". Because combinations of "filter" and "calculation" does not have ramification, it is easy to divide these combinational processes into several parts. If subscription rules are creates, data components manage these rules and transfer or calculate sensor data based on rules.

### 2.3. Division of Data Component

Data components enhance utility value of sensor data on CBN. On the contrary, it is possible that data components cause high-loaded situations, because many published data gather on a data components and the data components process/republish these data. In addition, routers around the data components have to transfer many messages. We should consider load distribution on data components besides optimization of data components assignment.

If a data component is high-loaded, the data component is divided based on 'publishers' and combination of 'filter', 'calculation' tags. The procedure of this division is shown as follows.

1. high-loaded data component(DC-A) divides own subscription rule
2. DC-A sends divided these rules to around components
3. around components process a part of data
4. around components process a part of data
5. DC-A publishes the same data and DC-A's load drops

For example, if a high-loaded data component subscribes a square region, the data component divides this region into 4 regions by dividing ranges of latitude/longitude into equals halves. Another way of division is that a data component divides a subscription region by its address. Address has a hierarchical structure (e.g., a city has some towns). Data components use this hierarchical structure for divisions.

### 3. Evaluations

This section shows performance evaluations about divisions of data components. Table 1 shows the environments of evaluations. The machine-A consisting of CPU-A, Memory-A and OS-A is used as publishers (data generator). The machine-B is used as data components.

#### 3.1. Division of Data Components

Data components can create more valuable data than raw data by processing data from many publishers. On the contrary, it is possible that data components are easily high-loaded. Load distribution of data components is one of the most important mechanism on our proposed system.

Figure 4, 5 and 6 show the result of message loss rate and response time. The x-axis means the number of queries per one second. On this experiment, data components constantly subscribe temperature data. After subscribing data, these components calculate the average of these data and republish calculated data.

As shown in Figure 4 and 5, one data component can process approximately up to 300 published data per one second without message loss and delays, and divided two data components can process approximately 600 published data. We define these max values without message loss as the upper limit. If the load is over the

<table>
<thead>
<tr>
<th>CPU-A</th>
<th>Intel Xeon L5520 2.26GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-B</td>
<td>Intel Core 2 Duo 1.20GHz</td>
</tr>
<tr>
<td>Memory-A</td>
<td>12.0GB</td>
</tr>
<tr>
<td>Memory-B</td>
<td>1.0GB</td>
</tr>
<tr>
<td>OS-A</td>
<td>Linux 2.6.21(ubuntu 9.10 Server)</td>
</tr>
<tr>
<td>OS-B</td>
<td>Windows Vista</td>
</tr>
<tr>
<td>NIC</td>
<td>Gigabit Ethernet(PCI-X)</td>
</tr>
<tr>
<td>Language</td>
<td>Java6</td>
</tr>
</tbody>
</table>

Table 1. environments of evaluations
server support multi-attribute search as well as an index table to manage sensor profiles and sensor data, and environments such as DNS. Live E! servers separately implement an authentication mechanism in distributed environments. This tree topology is suited for managing sensor data. Owners can control data access by the servers. The performance upper limit is proportional to the number of division, the higher the upper limit of processing data is. The performance upper limit is proportional to the division number.

### 3.2. Prototype Application and Simulator

Figure 7 shows web user interface and GUIs of the proposed system. On this web user interface, users register sensors as publishers, setting their attributes (e.g., locations, sensor types, and others). Users also describe subscription rules on this web. After selecting and inputting some information, this web system generates subscription rules as XML documents and sends these rules to routers.

Operators and developers use the GUIs applications made by Java. This applications can create routers and measure message traffic. First, operators create routers. Second they register routers on web as the same of sensors registrations. On this GUIs applications, operators can easily divide data components by manual, using mouses. Of course, data components are divided automatically based on their load.

Figure 7 shows the simulator of the proposed system. On this simulator, users define sensors distributions (sensor density), subscription rules. On this Figure, we assume that a sensor is installed in each 100 meter square. Simulator calculates the load of data components and shows these load. For example, red characters means that a particular data component is high-loaded and fails to process almost messages. Using this simulator, users visually know the relations among load of routers/data component, the number of publishers, and subscription rules. This simulator is useful to make a guideline for operating a particular application. For example, we use past rainfall data on June 2009 and easily know that how many data components (computers) we should prepare to operate this application notifying rainfall information.

### 4. Related work

Live E! distributed servers use a tree topology. Owners of sensors add Live E! servers to manage sensor data. Owners also can control data accesses by the servers. This tree topology is suited for implementing an authentication mechanism in distributed environments such as DNS. Live E! servers separately manage profiles of sensor devices and sensor data and periodically receive these profiles from children sites. Each Live E! server creates an index table from these profiles. Using this index table as a routing table, Live E! servers support multi-attribute search as well as an authentication mechanism in distributed environments. However, Live E! servers only provide raw data and do not provide pub/sub system or calculated data.

IrisNet is one of large-scale sensor networks [5,6]. IrisNet uses a two-tier architecture, comprised of two different agents. These agents are called sensing agents (SAs) and organizing agents (OAs). OAs are organized into groups and one service consists of one OA group. A group of OAs creates the DNS like distributed database and provides query processing infrastructure for a specific service (e.g., parking space finder, person finder). An OA manages a part of a tree topology database and this tree structure is suitable for a XML document tree. Users send XPath queries to a web servers. A web server can process XPath queries by following the path tree of OAs. IrisNet has a load distribution mechanism based on structures of addresses. However, IrisNet does not have data processing mechanism and also does not provide pushing mechanism.

These large scale sensor networks do not consider supporting push mechanisms. On the other hand, we adopt push mechanism based on Content-Based Network (CBN). One of the major studies on CBN field is Scalable Internet Event Notification Architecture(Siena) [7]. Siena provides Pub/Sub systems consisting of distributed servers. Subscribers describe a filter or a pattern (combination of filters) as a subscription rule and send it to a particular server. The server broadcast this subscription rule and all servers know this rule. Through this broadcast process, a distribution tree is created. Each server checks data from publishers and sends matched data with subscription rules through distribution trees. If Siena handles simple raw data on simple server toporogy, it works well, but Siena does not consider handles complex data or data processing. As the same of GSN, Siena does not have a load distribution mechanism of data processing.

### 5. Conclusion

Recently, many sensor devices are installed everywhere and it is required that large sensor networks managing these sensor data. However almost existing systems only manage raw sensor data and these data are provided by pull technology. Then, we add data processing components onto content-based networks. Content-based networks provide publish/subscribe system in distributed environment and appropriate for notifications and alerts. Publish/subscribe mechanism and data processing components enhance the application fields in ubiquitous sensing environment.

In this paper, we describe how to add data processing components onto content-based networks. It is possible that data processing components causes concentrations of sensor data, however we distribute the load of data components by dividing subscription rules based on geographical locations or address structures. Performance evaluations of our implementation shows that load distribution mechanism works well, and proposed system secure the scalability by adding data components.

As the future works, we should optimize data components assignment, considering not only path lengths but message traffic, RTT and other criteria.
Acknowledgement

This work was supported by The Ministry of Education, Culture, Sports, Science and Technology (MEXT) KAKENHI 22700072.

Reference


