The Architecture of Middleware for Real-time Sensor Data Sharing

Dong-lai JIAO¹ and Tao-tao CAO²

¹College of Geographic and Biologic Information, Nanjing University of Posts and Telecommunications, Nanjing 210046, China
²College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

Keywords: IoT, Netty, Middleware, Real-time, Sharing.

Abstract. With the development of Internet of Things technology, the era of the Internet of Everything is getting closer and closer to us. People use a variety of applications combined with sensors to greatly improve social productivity. There is almost no interaction between these sensor applications and the utilization of sensor information is not maximized. Due to the massive, real-time, and heterogeneous nature of sensor information, these tasks are difficult to develop. This paper builds a middleware based on Netty to realize real-time sharing of sensor information and analyze its performance through comparative experiments.

Introduction

The concept of Internet of Things was proposed by British technical expert Kevin Ashton at the end of the twentieth century[1]. With the rapid development of Internet of Things technology, the decline of sensor manufacturing costs and the maturity of sensor network technology, More and more industries are beginning to use IoT technology to provide real-time information services to people. For example, the power industry widely uses power real-time information integration systems to monitor and manage production lines. The urban traffic control department uses the intelligent digital bus stop system to provide people with real-time information services from the bus to the station. Environmental monitoring department uses various sensors to monitor real-time environmental information, etc[2,3]. These sensing devices informatize and digitize various life scenarios. Enable people to master their status information anytime, anywhere via the network. Great convenience for people's lives and greatly improved social productivity[4,5,6]. However, most of the applications in the field of Internet of Things currently only serve their own industries[7]. Most of the sensor data information is only for their respective applications[8]. For example, environmental monitoring sensors only serve the environmental monitoring department, and power sensing devices only serve the power system department, forming information islands[9]. Third-party users or applications (hereinafter referred to as user) can only get sensor information from other applications in two ways: One way is to obtain access to the open platform of the app. The other is to directly access the database to obtain information when the administrator of other applications opens the database access rights. However, the data obtained by these two methods is mostly static data, and it is not real-time. With the advent of the era of the Internet of Everything, Real-time sharing of sensor data between different IoT applications to meet the needs of some social production. For example, the construction of modern smart cities requires a wider range of interconnections. Real-time monitoring of environmental and business conditions by connecting, interacting, and sharing multiple system application data[10,11]. Analyze the situation from a global perspective and solve problems in real time so that work can be done through multiple parties and change the way the city works. Therefore, real-time sharing of sensor information in IoT applications is a problem worth exploring.

With the development of Internet of Things technology, more and more sensors are used in various production and life scenarios, so the number of sensors is massive. To achieve the sharing of sensor data, users must be able to quickly locate the target sensor according to their needs in massive information. Metadata is a structured information used to describe, interpret, and locate a
This paper establishes a global directory service by accurately describing the sensors based on metadata of different IoT applications, enabling users to efficiently locate the corresponding sensors according to their needs.

Sensor messages are heterogeneous and real-time. This paper realizes real-time sharing of sensor information of different IoT applications by constructing a sensor information real-time sharing middleware. Middleware belongs to the category of reusable software. Provides a homogeneous, distributed software development platform for users and developers by shielding hardware devices, services, and other heterogeneities and complexities. Widely used in a variety of distributed architectures and heterogeneous environments. Different from the traditional middleware application environment: the information processed by the sensor information sharing middleware has higher real-time performance. The sensor information sharing middleware (hereinafter referred to as middleware) realizes real-time transmission of information by establishing a long-connection communication service between the producer and the consumer of the information. Sharing is the core function of middleware and publish/subscribe is a way to share. In this paper, the sensor publishes a message, and the message consumer subscribes to the sensor to realize the sharing of the sensor message.

**Middleware Overall Architecture**

The entire middleware is divided into five modules as Figure 1 shows: global directory service, data communication service, message queue service, message push service and data persistence service. Register the metadata information of the sensor in the global directory service when the sensor establishes a connection with the middleware for the first time. The metadata information is unique throughout the middleware; The data communication service establishes and maintains a long connection with the sensor and resolves the message from the sensor. Then post the parsed message to the message queue service; The Message Queuing service establishes a unique subject for each sensor, and manages each sensor message separately. Decoupling data communication service and message push service, making the entire architecture highly scalable and also serving as a messaging station; The message push service listens on the sensor(Listen to the subject of the sensor in the message queue service). The message push service pushes the corresponding sensor message in real time according to the sensor information subscribed by the user. (The user first obtains the metadata information of the sensor according to his own needs in the global directory service. Then establish a connection with the message push service and send the metadata message to the message push service) Therafter achieving the purpose of real-time sharing of sensor information; The persistence service is used to persist metadata such as sensor description information and sensor history messages to improve the reliability and stability of the entire middleware.

![Figure 1. Middleware overall architecture.](image-url)
Middleware Design

Global Directory Service. Global Directory Service provides registration services for sensors accessing middleware. Establish a global directory service based on the unique description of the metadata level for each sensor through functions, work areas, job locations, and other elements. Through the global directory service, the user can quickly locate the sensor according to his own needs and then obtain the corresponding sensor metadata information. When the user establishes a connection with the message push service, the obtained sensor information is sent to the message push service. The message push service adds user to the corresponding subscription group based on the information and complete the subscription of sensor.

Data Communication Service. Sensor information is real-time and massive. In order to obtain sensor messages in real time, the data communication service needs to establish a long connection with the sensor. And in the case of such a large number of sensors, the load of a single data communication server should be increased as much as possible. The traditional model is a connection to create a Socket thread. When the thread reaches a certain amount, the server can't bear a lot of thread load pressure and it is easy to crash. Netty is an efficient client/server communication framework based on the NIO model, using the reactor design pattern. A Reactor aggregates a selector. Thousands or even more clients can be connected, listened, polled by server with very few threads. Greatly reduce the performance loss caused by server creation threads and thread context switching. Therefore, this paper uses Netty as the basic framework of data communication service, which can solve the real-time acquisition of sensor data and the stress caused by multithreading. At the same time, Netty supports multi-protocol parsing and customization, so it can solve the heterogeneity of sensor information very well.

This article implements Netty's data communication service through a custom protocol. The protocol structure is as shown in Table 1: Where the frame header and the end of the frame are used to detect data validity and integrity; The frame length indicates the message type and the length of the message data; There are three types of messages. 0 is the heartbeat message, 1 is the sensor's own status message, and 2 is the collection message. Message data represents a specific message.

<table>
<thead>
<tr>
<th>Frame header</th>
<th>Frame length</th>
<th>Message type</th>
<th>Message data</th>
<th>Frame tail</th>
</tr>
</thead>
</table>

Netty provides message codec and business processing based on the chain of responsibility model. When receiving a message, first deal with the sticky and unpacking of TCP frame data. Then parse the data frame according to the custom protocol parsing method. Finally, the business processing of the obtained data. The process is shown in Figure 2.

![Netty processing flow](image)

Figure 2. Netty processing flow.

Message Queue Service. The number of sensors is massive and the recipient of the message is
dynamically changing. If the message is sent directly to the receiver in the data communication service, the data communication service not only needs to resolve a large number of sensor messages, but also needs to maintain complex dynamic message receiver relationships. As a result, the entire data communication service is very complicated and difficult to expand. So this paper decouples message producers (data communication service) and message consumers (message push service) through message queue service. The data communication service publishes the processed message to the message queue service. The message push service does not need to establish any connection with the data communication service, and directly obtains the message from the message queue service, thereby decoupling the two services. By combining two message models of sensor and two working modes of message queue, fine-grained message management is achieved. The process is shown in Figure 3.

Figure 3. Message queue service processing flow.

**Message Push Service.** In the field of Internet of things, sensor messages are mostly real-time dynamic data. Usually, the acquisition of dynamic data is updated by means of long polling. Set a certain polling interval to periodically poll the database. However, when the user volume is large and the polling period is set unreasonably, the server may be crashed due to limited load capacity of the server.

This paper establishes a long connection between the user and the message push service. When the sensor message is updated, the server actively pushes the sensor message to the user. Also based on the Netty framework to build a server connected to the user. The basic processing flow is similar to data communication service. First, the user finds the sensor metadata information that needs to be subscribed through the global directory service, and then establishes a long connection with the message push server and send the obtained sensor metadata information. The server parses the metadata information through the corresponding protocol and add the user to the message subscription group of the corresponding topic, and the above steps are completed to indicate that the consumer subscribes successfully. The server listens to the corresponding topic in the message queue. When the topic has a message update, the server pushes the message to each subscriber in the topic subscription group in real time. The process is shown in Figure 4.
Persistence Service. In order to improve the reliability and maintainability of the entire middleware, it is necessary to persistently store important information of the entire middleware service. For example, sensor metadata information in global directory services, user information in message push services, message subscription groups, and so on. Relationship is shown in Figure 5:

![Message push service process](image)

**Implementation and Testing**

Simulate the sensor terminal through the client program, and send data to the data communication service of the middleware at regular intervals. Simulate a third-party user through a client program and subscribe to the sensor concurrently. The server configuration is shown as Table 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel(R) Core(TM) i5-337U CPU @1.8GHZ 1.8 GHZ</td>
</tr>
<tr>
<td>RAM</td>
<td>4GB</td>
</tr>
<tr>
<td>HD</td>
<td>500GB</td>
</tr>
<tr>
<td>OS</td>
<td>Windows64</td>
</tr>
<tr>
<td>Network Adapter</td>
<td>100Mbps</td>
</tr>
</tbody>
</table>

Experiment with the design method in the paper and the traditional periodic polling method, change the number of concurrent subscription users, statistical message arrival delay, message arrival rate and CPU load. The experimental results are as follows:

<table>
<thead>
<tr>
<th>Number of connections</th>
<th>CPU consumption</th>
<th>Average delay(s)</th>
<th>Message arrival rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>8%</td>
<td>&lt;1</td>
<td>100%</td>
</tr>
<tr>
<td>1000</td>
<td>14%</td>
<td>&lt;1</td>
<td>100%</td>
</tr>
<tr>
<td>1800</td>
<td>23%</td>
<td>&lt;1</td>
<td>100%</td>
</tr>
</tbody>
</table>
Table 4. Test results by traditional long polling method.

<table>
<thead>
<tr>
<th>Number of connections</th>
<th>CPU consumption</th>
<th>Average delay(s)</th>
<th>Message arrival rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>10%</td>
<td>&lt;1</td>
<td>100%</td>
</tr>
<tr>
<td>100</td>
<td>22%</td>
<td>&gt;1</td>
<td>100%</td>
</tr>
<tr>
<td>200</td>
<td>45%</td>
<td>&gt;2</td>
<td>80%</td>
</tr>
</tbody>
</table>

It is relatively simple to update messages using the traditional long polling method. However, it can be seen from the experimental results that the CPU load capacity of the method designed in the paper is greatly improved, and the delay of the message is relatively small, and the message arrival rate is more stable.

**Summary**

This paper discusses the design and implementation of sensor information real-time sharing middleware based on Netty. Firstly, the significance of this paper is analyzed by analyzing the current situation of Internet of Things applications and the actual production needs. Then combined with the massive, real-time and heterogeneous sensor information, the specific design scheme is proposed. Finally realize the middleware function and analyze the performance of the middleware through experiments. The next step is to optimize the functionality and explore the clustering model to further improve the stability and usability of the entire middleware.

**Acknowledgements**

We gratefully acknowledge financial support from the National Natural Science Foundation of China (Grant No. 41471329, 41101358).

**References**


[9] Internet politics beyond the digital divide. Calderaro A. Social Media in Politics. 2014.


