Design of Data Transmission System Based on Smart City Integrated Data Center Construction

Changping Gong\textsuperscript{1,2}, Wei Wang\textsuperscript{1,2}, Jian Cao\textsuperscript{1,2}, Guangrui Peng\textsuperscript{1,2} and Jianglong Gao\textsuperscript{1,2}

\textsuperscript{1}Nari Group Corporation/State Grid Electric Power Research Institute, Nanjing 211106, China.
\textsuperscript{2}China Realtime Database Co., Ltd., Nanjing 211106, China.
Contact email: 1198894081@qq.com

Abstract: The construction of integrated data center provides data support services for energy smart city, which is inseparable from the aggregation of various data, such as electricity consumption information and power related data all over the city. Therefore, a stable and efficient data transmission system is needed to complete the data transmission work. But in reality, there are some problems such as the complexity of the public network environment, the diversity of data sources, and the security of source data. In order to solve the above problems, the data transmission system needs to have the following core functions: message forwarding, routing exploration, concurrent transmission, encrypted transmission, breakpoint continuation, error retransmission, etc. Based on socket communication, the system constructs a routing path from the source to the data center by exploring the routing nodes. By encapsulating the messages, the messages exchange and forward the information between the paths. For the source data with high security requirements, it encrypts the transmission. At the same time, it designs the transmission failure protection mechanism, which leads to connection problems in the transmission process, Connection recovery can continue transmission; at the same time, it can continue transmission for some unfinished tasks; it can automatically resend the failed data.

1. Introduction

The core of smart city integrated data center is data, the application and system services of various smart cities need continuous massive data as support, and the data is distributed all over the city, so a stable and efficient data transmission system is urgently needed to gather all kinds of scattered data to the integrated data center. Because of the diversity of data sources, the complexity of public network environment, and the security of source data, we should design a non-direct network data transmission system to differentiate traditional point-to-point data transmission system, isolate the data source from the integrated data center, and ensure the security of the integrated data center through routing. In view of the diversity of data sources, this data transmission system only deals with file data, for non-file type data source, files can be generated through ETL tools.

The design of data transmission system based on smart city integrated data center normalizes the data transmission process through independently designed application protocol, adopts distributed architecture, supports horizontal expansion, ensures the stability of file transmission system, ensures the efficiency of data transmission with multithreading and streaming forwarding technology, and combines the common needs of data transmission, such as breakpoint continuation and error retransmission, encrypted transmission and other functions are designed to form a complete data transmission system based on the construction of smart city integrated data center.
2. Architecture Design
The data transmission system is mainly divided into API and broker modules.

**API**: responsible for file processing, including file read and write operations, file chunking, file stream sending and other functions.

**Broker**: responsible for building task routing and file flow forwarding. Routing refers to the process that determines the network scope of the side-to-side path when the packet is from the source to the destination.[1]

3. Module Design

3.1. **API Module**

**ConnectionFactory**: connection factory class;

**Connection**: TCP socket connection;
**Producer**: sender object, including all sending logic;  
**Consumer**: receiver object, including all receiving logic;  
**Sendfile()**: send file main method;  
**Recvfile()**: receive file master method.

### 3.2. Broker Module

![Broker Module Diagram]

**Server**: the main program, which is responsible for receiving all connections and messages and controlling the use of resources;  
**MessageHandler**: process corresponding messages according to message type; encapsulate all types of messages;  
**ConnectionManager**: responsible for managing the connection relationship with API;  
**BrokerManager**: responsible for managing the connection relationship with neighbor broker;  
**UserManager**: responsible for managing the usage of sender and consumer objects.

### 4. Application Protocol Design

In order to ensure the stability of data transmission, we design the application protocol of data transmission – message. All communication interactions, such as connecting heartbeat interaction, routing information, status information, file flow, etc., are packaged into messages, through the encapsulation and analysis of messages, we can accurately control the data transmission process, at the same time, we realize of breakpoint continuation, error retransmission and other functions conveniently by message.

The message design is as follows:
Table 1.

| Name                 | Type             | Length |
|----------------------|------------------|--------|
| Protocol magic number| MAGIC            | 2 bytes|
| Message type         | Message-Type     | 1 byte |
| Hops                 | Max-Forwards     | 1 byte |
| Message sequence number| Message-Seq     | 4 bytes|
| Origination address  | Original-IP      | 4 bytes|
| Originating port     | Original-Port    | 2 bytes|
| Task ID              | Task-Id          | 2 bytes|
| token                | Token            | 4 bytes|
| Source account       | Producer         | 64 bytes|
| Purpose account      | Consumer         | 64 bytes|
| Message body length  | Body-Length      | 4 bytes|

Because there are multiple message types in the application protocol, the example of the FileDataReq type message body design is used to illustrate:

Table 2.

| Name                 | Type            | Length        |
|----------------------|-----------------|---------------|
| Document identification| 0x0301          | 16 bytes      |
| Block number         | 0x0302          | 4 bytes       |
| Segment number       | 0x0303          | 4 bytes       |
| bitmap data          | 0x0304          | variable      |
| Summary data         | 0x0305          | 16 bytes      |
| Fragmentation data   | 0x0306          | <=1024*1024 bytes |
| Fragment length remainder| 0x0307        | <1024 bytes   |

A complete message includes a message header and a message body. The header is fixed to 152 bytes, and the type and length of the message body are marked in the message header. For different message types, the corresponding message processing is carried out according to the content of the message body, when all messages of a task are processed, the transmission task is completed.

5. Transmission Process Sequence Diagram
The data transmission sequence is as follows:
The sequence diagram of data transmission completely shows the whole process of file transmission, including route exploration, file preprocessing, file flow and other core operations. The processing of LoginReq, FileInfoReq, QueryRouteReq, FileDataReq requests and their responses constitutes the communication process of data transmission.

6. Core Technology

6.1. Breakpoint Resume
The file sending process is interrupted, some data has been written into the temporary file at the target side, in order to ensure the overall efficiency of the file transmission, the breakpoint resume function is designed here. During the file sending process, a bitmap file is maintained at the target side, in which the block information that has been successfully written is recorded. When the file transmission task is interrupted, the file is transferred again, the bitmap file information can be obtained at the target side. After the route path is established, it is forwarded to the source side through the status report request, according to the bitmap file information, the source side can continue to transmit which the block file is not received, so that the overall file transmission efficiency is guaranteed, avoiding the situation that all files need to be retransmitted.

6.2. Error Retransmission
The file transmission process is based on messages, each message has a unique message sequence number, this is a message identifier. During the file transmission process, we can use this message sequence number: Message-Sec to locate whether the message is consumed. If the message is not consumed within the specified time, the message will be retransmitted through the sending side, and there is a limit on the number of retransmissions, so as to avoid problems such as delay in message processing due to network reasons, and improve the fault tolerance of the file transmission process.
6.3. Route Discovery

The purpose of broker routing is to solve the problem that the sender can accurately transfer the file data to the destination user when the destination user login location is unknown. Broker routing includes configuration reason, dynamic routing, query routing, record routing and other core technologies. The configuration route is a static route, each broker server will be configured with 0-n neighbors. After the broker starts, it will actively try to connect the neighbors. The directed connection graph composed of neighbor relationship determines the accessibility of the target user. For various reasons, the configuration of routing may cause the following problems: multiple paths, loopback paths and destination are not reachable, so the broker must maintain a dynamic routing table internally. Dynamic routing maintains the path from local to destination, and maintains its validity by querying and recording the path. When a server in the directed graph goes down or shuts down normally, its upstream neighbors will send a node failure status report after detecting the connection interruption; the server receiving this report will delete the routing table items related to this node. When a transmission task starts, because the destination is unknown, the starting server will send a request to query the route. The request message will be forwarded in multiple ways, and the server address will be recorded along the way. It will be forwarded step by step through the neighbors, and finally reach the endpoint server. At this time, the forwarding will be stopped and a routing notice will be sent to the starting server. Route notifications are routed according to the records in the query route request. The original route is forwarded level by level until it reaches the starting server. After the starting server sends out the route query, it may receive multiple route notifications. At this time, it should calculate the shortest path through the algorithm and eliminate the loop in the path. In order to avoid the routing dead cycle in extreme cases, all messages have Max-forwards attribute. The initial value is 10. After each node passes by one, it is decreased by 1. When the value is 0, forwarding stops. Large file transmission is usually completed in a long time, and the route is stable. In order to improve the performance, the starting server will cache the path to avoid too many queries on the route.

7. Summary

The design of data transmission system based on the construction of smart city integrated data center aims to solve the problem of data concentration from multiple sources; the core point of system design is to build a non-direct transmission path through routing algorithm, isolate the source side and the target side, and ensure that the two levels of data do not interfere with each other; the core module broker clustering design ensures that the data transmission system is highly available and fault-tolerant divided into hardware and software failure; through the design of message protocol to ensure the traceability of data transmission process, to achieve two core functions: breakpoint resume and error retransmission, to ensure the stability and transmission efficiency of data transmission process.

8. Acknowledgements

This paper is supported by the project of State Grid Electric Power Research Institute (based on the key application technology research of ubiquitous power Internet of things data center 524623190007).

9. References

[1] Calvert, Kenneth L, 1988, TCP/IP Sockets in Java [M], Morgan KaufmannPublishers.
[2] James F.Kurose, Keith W.Ross, Chen Ming Translate, 2009, Computer Network Engineering[M],CHINA MACHINE PRESS.
[3] Cay S. Horstmann, Cray Cornell. Zhou Lixin, ChenBo Translate, 2013, Core Java Volume I-Fundamentals (Ninth Edition) [M]. CHINA MACHINE PRESS, P620.
[4] Burg, 2003, Java in Distributed Systems [M], CHINA MACHINE PRESS.
[5] Shi Wei, 1997, Windows Sockets Specification and Application [M], Publishing House of Electronics Industry.
[6] Yang Yanshuang, 2007, TCP/IP Protocol Analysis and Application [M], CHINA MACHINE PRESS, China.