Shipborne Robust Message Queuing Service Prototype System

Yongguo Jiang, Qiang Liu, Qianqian Liu, Jian Su, Changshuai Qin

Department of Computer and Technology, Ocean University of China, Qingdao, P.R.China

***Abstract*—In view of the characteristics of multi-source data fusion requirements for shipborne information systems, this thesis improves ActiveMQ and builds a shipboard robust communication prototype system. The prototype system takes advantages of reliable data asynchronous transmission to provide flexible and reliable system communication for functional platforms of shipboard information system (SIS) and overcomes the difficulties of heterogeneous system integration in SIS. This system also improves the traditional message queue model by offering multiple message queues to serve a single data link, making the native Java Message Service (JMS) more reliable. On the other hand, the system provides a scalable clustering solution that utilizes a separate configuration center to implement the expansion, control, and monitoring of message server nodes, and dynamically distribute the load of the message server and message queue. The improved system effectively overcomes difficulties of single line and single point information source failure and becomes more interference and damage resistant.**

Keywords—Message-oriented middleware(MOM), ActiveMQ, Java Message Service, Ship information system(SIS), System integration

I. Introduction

With the continuous advancement of marine detection technology, a large amount of information to be obtained in the fields of ship navigation, fishery fishing [1], the scientific investigation [2], and maritime law enforcement [3] is provided through advanced detection equipment. At the same time, ships need to integrate a variety of detection and monitoring equipment, such as Sensor Network (SN), acoustic equipment, infrared imaging equipment, shipborne radar, Unmanned Surface Vehicles (USV), Unmanned Underwater Vehicles (UUV), Unmanned Aerial Vehicle (UAV), etc. However, most devices are manufactured and developed by independent vendors that use different operating systems as the operating environment and rely on different network architectures [8, 9] to make the ship information system (SIS) a distributed application system. In the traditional centralized shipborne information system (SIS), the direct communication between the various application systems, the connection relationship is complex, so the traditional centralized shipborne information system communication efficiency is relatively low [10]. On the other hand, this communication mode cannot satisfy the scalability and flexibility of the system. At the same time, according to the reliability requirements of the American Bureau of Shipping (ABS) for SIS, SIS design must use redundant design to achieve reliability [9, 11]. Therefore, a reliable, flexible, and scalable heterogeneous system communication solution is needed to meet the needs of SIS integration.

Currently, enterprise-class distributed application system integration solutions often use message middleware (MOM) to achieve integration of heterogeneous systems. MOM provides a cross-platform asynchronous messaging solution that shields the underlying complex operating system and network architecture to ensure reliability, cross-platform data exchange between applications in a distributed network environment [12]. Most of the existing MOM products can provide better communication services, but lack of Quality of Service (QoS) support in the harshest communication environment, and can't meet the flexibility and reliability requirements of the SIS integration for the communication module. For example, based on the "lightweight" communication protocol MQTT [13] based on the publish/subscribe model, MQTT is good at providing real-time and reliable data transmission in the service scenario of a bad network scenario but often causes memory congestion when transferring large files. Suitable for integration of application systems such as on-board video data and image data; ZeroMQ [14], a multi-threaded network library based on the message queue, can provide high-throughput and low-latency communication services due to its agentless communication mode. It is less reliable and is not suitable for SIS application requirements.

In order to meet the requirements of the SIS for real-time, reliability and scalability, this paper designs and develops a shipboard embedded robust message queue service prototype system based on Apache ActiveMQ(AMQ) [15]. AMQ is an open source implementation of the Java Message Service (JMS) standard [12]. Among them, JMS includes two communication models, peer-to-peer and publish/subscribe, and provides a reliable information transfer mechanism. On the other hand, AMQ uses the message broker to manage the message queue and provides a redundant design of full backup, which realizes the clustering of message brokers and ensures the reliability of information sources. However, given the limited performance of hardware devices in offshore environments, such data redundancy scale may be unacceptable. Therefore, we use a unified configuration management center to manage redundant message queues and message brokers and realize the reliability and scalability of the message transmission service under the premise of ensuring redundant backup of messages.

Our main work includes the following two aspects:

A. The JMS message queue model has been improved to integrate multiple queue entities into one queue service group, each of which serves a group of producer-to-consumer data transfers.

B. On the other hand, the system provides a scalable clustering method, which uses the configuration center to manage the message queue entity and the message server in a unified manner, realizing the dynamic adjustment of the message queue entity in the message server and the queue service group, and guarantees the message. Reliability.

II. SYSTEM DESIGN

A. System integration architecture

Message-oriented middleware is an intersystem communication technique that uses the message queue mode. In a distributed system, the application (producer) that generated the message sends the specified message to the message queue. On the other hand, the application (consumer) concerned with this message accepts the message from the message queue. The delivery of the message is handled by the message queue. The producer and the consumer do not know each other's existence, and can effectively implement loosely coupled system integration.

Based on AMQ and combined with the requirements of ship information system integration and application, this paper provides a fully integrated and highly survivable universal network for all mission-critical functions of SIS in the form of message queue group, including control, navigation, monitoring, and detection information.

B. Message Queue Prototype System Framework

The overall framework of the shipboard embedded robust message queue service prototype system is shown in Figure 2, including the configuration center, AMQ cluster, and user API. The configuration center is the core component of the prototype system. It is primarily responsible for managing and monitoring message queues, managing AMQ clusters, monitoring AMQ node status, providing message queue connection information for producers and consumers, and dynamically adjusting a load of AMQ nodes and message queues. An AMQ cluster consists of multiple physical servers, each with a different AMQ proxy node. The cluster provides information backup and messaging services for the SIS. The user API provides a simple programming interface for applications that can be rapidly integrated into the SIS. The process of using the prototype system for message service is roughly divided into seven parts, as shown in figure 1.

First, the message producer asks the configuration center if the message service it needs exists. If the service is available, the configuration center returns the message queue information used by the message service. After the message producer receives the information of message service returned by the configuration center, it connects to the AMQ cluster according to the information and sends messages to the specified node and message queue in the message service information, as shown in steps 3 and 4. The consumer also needs to perform steps 1, 2, and 3 to request the message service information from the configuration center and connect to the AMQ cluster according to the information. Then the AMQ node pushes information from the specified message queue to the consumer.

If the service requested by the producer does not exist, the configuration center registers the new service and performs step 6 to control the AMQ cluster and open up the message queue groups used by the new service in the cluster. Meanwhile, the configuration center monitors the cluster in real time and determines whether the message service is in a healthy state, according to the node and information of message queue returned in step 7. If the message service is in an unhealthy state, step 6 is performed to dynamically adjust the cluster nodes and message queues used by the message service.

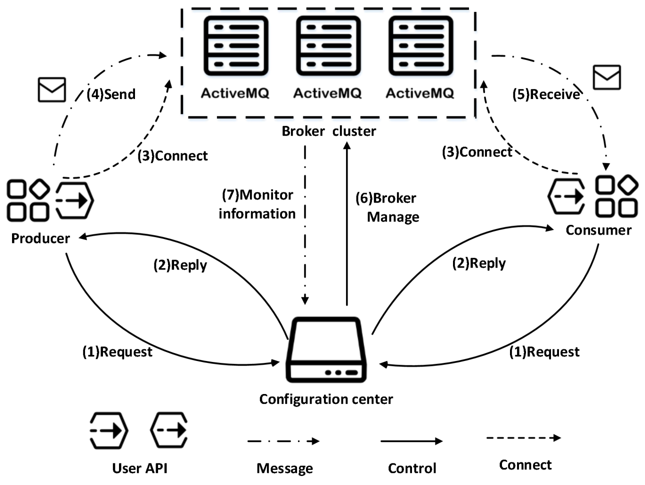


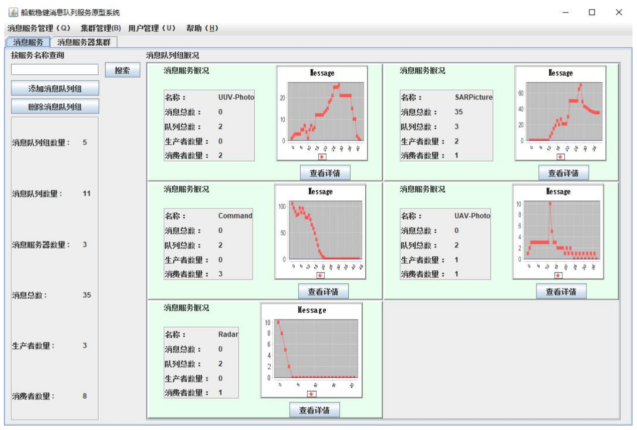
Fig. 1. Message Queue Prototype System Framework

III. SERVER DESIGN

A. Message queue group model

For a certain message producer, JMS uses a message queue entity to provide both peer-to-peer and publish/subscribe messaging services. A message queue entity is usually stored on a fixed message server, and producers and consumers can only send and receive messages through the message server. Even if multiple message servers are used to provide message services, the message queues of each message server are not associated. Once a message server fails, all message queues it hosts will not be available. In order to improve the reliability of the message service and meet the redundancy requirements in the SIS design, we have designed a new message queue group model. A message queue group is an abstraction of multiple message queue entities, containing multiple message queue entities in different message servers. When a message producer generates a message, it sends a message to the message queue group. Messages are sent to different message queue entities of different message servers. It means that the message server where the message is stored is undefined. Thus, when a single message server fails, producers and consumers can perform messaging services through other message queues and servers in the message queue group. It can effectively solve the problem of single line failure. The message queue group model is shown in Figure 2.

runtime environment can use desktop applications developed in Swing. The display module needs to display the whole function of the prototype system intuitively and provide user-friendly operation mode. According to the overall design of the system, the display interface function module is divided into message service management and cluster management, as shown in figure 5. Due to operational security requirements, the prototype system has different permissions for different users. Administrators can manage users in the integrated interface.



消息队列组数量： 5
消息队列数量： 11
消息服务器数量： 3
消息总数： 35
生产者数量： 3
消费者数量： 8

名称： UUV-Photo
消息总数： 0
队列总数： 2
生产者数量： 0
消费者数量： 2

名称： SARPicture
消息总数： 35
队列总数： 3
生产者数量： 2
消费者数量： 1

名称： Command
消息总数： 0
队列总数： 2
生产者数量： 0
消费者数量： 3

名称： UAV-Photo
消息总数： 0
队列总数： 2
生产者数量： 1
消费者数量： 1

名称： Radar
消息总数： 0
队列总数： 2
生产者数量： 0
消费者数量： 1

Fig. 2. Integrated display interface

Message service management provides the monitoring, registration, modification and other functions of message queue groups. The main interface provides an overview of basic information about each message queue group, including name, number of messages, number of producers, number of consumers, number of message queues, etc. The message service detail interface provides the detailed information of the message service corresponding to the message queue entity, as well as the operation of sending a message, deleting a message, suspending dispatch and so on to the message queue.

The cluster management module takes the message server as the operation and monitoring object and provides the monitoring information and basic control functions of the message server. Monitoring information includes AMQ Broker monitoring information and server hardware monitoring information. On the other hand, the management module also provides remote control functions of AMQ Broker, including adding message queues, deleting message queues and so on.

IV. Conclusion

The shipborne robust message queue service prototype system provides a reliable and flexible solution for SIS to integrate multi-source data. At the same time, the prototype system USES the message queue group model and scalable clustering scheme to reduce the hardware requirements of the message service system and provide a reference for the distributed system integration under harsh environment. However, the prototype system still needs further improvement. For example, the monitoring center and real-time warning are combined to provide users with alarm information. In addition, in a real work environment, it is not very reliable to use a single configuration center node to provide management for the entire message service, so a configuration center clustering scheme needs to be provided.

Acknowledgment

This work is supported by the National Key R&D Program of China (Grant No. 2017YFC1405200)

References

[1] M. Doray et al., "The PELGAS survey: ship-based integrated monitoring of the Bay of Biscay pelagic ecosystem," vol. 166, pp. 15-29, 2018.

[2] Y. Xie, X.-h. Yang, F.-f. Xun, and L.-y. Wang, "Integrated Data Acquisition Terminal Used on Board," in 2018 18th International Symposium on Communications and Information Technologies (ISCIT), 2018, pp. 127-130: IEEE.

[3] Z. L. Szpak and J. R. J. E. s. w. a. Tapamo, "Maritime surveillance: Tracking ships inside a dynamic background using a fast level-set," vol. 38, no. 6, pp. 6669-6680, 2011.

[4] P. S. Vincent, C. P. Gardiner, A. R. Wilson, D. Ellery, and T. Armstrong, "Installation of a sensor network on an RAN Armidale Class Patrol Boat," in Materials Forum, 2008, vol. 33, pp. 307-316.

[5] C. Jun-Hong, K. Jiejun, M. Gerla, and Z. J. N. Shengli, IEEE, "The challenges of building mobile underwater wireless networks for aquatic applications," vol. 20, no. 3, pp. 12-18, 2006.

[6] H. Ferreira et al., "Autonomous bathymetry for risk assessment with ROAZ robotic surface vehicle," in Oceans 2009-Europe, 2009, pp. 1-6: Ieee.

[7] J. Sánchez-García, J. M. García-Campos, M. Arzamendia, D. G. Reina, S. L. Toral, and D. Gregor, "A survey on unmanned aerial and aquatic vehicle multi-hop networks: Wireless communications, evaluation tools and applications," Computer Communications, vol. 119, pp. 43-65, 2018.

[8] P. A. Bernstein, "Middleware: a model for distributed system services," Communications of the ACM, vol. 39, no. 2, pp. 86-98, 1996.

[9] M. Roa, "ABS naval vessel rules (NVR) for mission critical networks, software development, and safety critical control systems," in 2007 IEEE Electric Ship Technologies Symposium, 2007, pp. 138-144: IEEE.

[10] L. Chuang, Z. Gang, and G. Q. J. C. Engineering, "Research on Data Distribution Service for Ship Information System," vol. 39, no. 9, pp. 94-97, 2013.

[11] S. Liu, B. Xing, B. Li, and M. Gu, "Ship information system: overview and research trends," International Journal of Naval Architecture and Ocean Engineering, vol. 6, no. 3, pp. 670-684, 2014.

[12] M. Hapner, R. Burridge, R. Sharma, J. Fialli, and K. J. S. M. I. Stout, Santa Clara, CA, "Java message service," vol. 9, 2002.

[13] A. Banks and R. J. O. s. Gupta, "MQTT Version 3.1. 1," vol. 29, p. 89, 2014.

[14] P. Hintjens, ZeroMQ: messaging for many applications. " O'Reilly Media, Inc.", 2013.

[15] B. Snyder, D. Bosnanac, and R. Davies, ActiveMQ in action. Manning Greenwich Conn., 2011.

[16] A. Videla and J. Williams, "RabbitMQ in Action," 2012.

[17] A. Toshniwal et al., "Storm@ twitter," in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 2014, pp. 147-156: ACM.

[18] VMware. vRealize Hyperic. Available: https://www.vmware.com/products/vrealize-hyperic.html

[19] Alibaba. Dubbo Development guide. Available: http://dubbo.io/Developer+Guide-zh.htm.