WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 3, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v3n2.09
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Design and Implementation of SHMIOP Protocol for CORBA

Hongyi Liu *

China Electronics Technology Group Corporation Twentieth Research Institute, Xi'an, China

ABSTRACT

CORBA as a distributed object middleware application standard of the object management organization, the use of GIOP protocol makes cross-platform client-server object communication possible, and the use of shared memory combined with the GIOP protocol can improve the efficiency of transmission. In this paper, we propose a design method for a shared memory-based communication protocol, SHMIOP protocol, and test and verify it on e*ORB middleware.

KEYWORDS

CORBA; SHMIOP; Shared memory

1. INTRODUCTION

Object Management Group (OMG) [1] to strengthen the hardware and software collaboration capabilities in 1991 proposed a public object request broker architecture (Common Object Request Broker Architecture, CORBA).CORBA provides a suitable for the CORBA provides a flexible and rich architectural form for distributed system communication, through the use of orb in the object to establish the client-server object relationship, and will be linked.

The General Inter-ORB Protocol (GIOP) specifies a set of message formats and public data representations for communication between ORBs, allowing CORBA to interoperate with client and server objects under different operating systems and programming languages. It is because of this ability to communicate in heterogeneous environments that client and server objects need two encoding/decoding (marshal/demarshal) processes when sending requests and receiving results, which affects the efficiency of CORBA-based distributed applications, especially real-time systems and Internet applications. Many scholars have studied how to improve the transmission efficiency. Literature [2] tries to optimize the transmission delay from simplifying the function lateral call steps with the time overhead of function calls. Literature [3] proposed several methods to optimize CORBA performance including compression of byte streams and optimization of byte order exchange. However, the above optimization methods provide limited improvement in transmission efficiency. Literature [4] proposed an optimization method for transmission via shared memory and applied it on a variety of middleware for testing, however, the applied middleware has low transmission efficiency. Comparatively speaking, Prismtech's e*ORB [5] has higher real-time, versatility and scalability, and has greater application prospects and value in resource-constrained environments with higher real-time requirements.

In this paper, we will firstly introduce the GIOP protocol, and propose a design method of SHMIOP protocol, and finally test and verify it on e*ORB middleware.

2. CORBA TRANSPORT PROTOCOL

2.1. IIOP Protocol

In order to shield the underlying transport protocol on the CORBA object communication between the impact of the communication between the two sides can communicate across the protocol, CORBA specification defines a standard general interoperability protocol between ORBs - GIOP. GIOP protocol is an abstract protocol, does not provide for a specific transmission mode, only the data GIOP can be mapped to different specific transport protocols, commonly used IIOP protocol is mapped to the TCP/IP protocol generated. No matter what kind of underlying transport mechanism is used by the two communicating parties, as long as it is mapped to the GIOP protocol, then interconnection and interoperability can be realized, and cross-protocol operation can be truly realized.

However, IIOP protocol has the advantage of reliability. In order to ensure reliability, IIOP protocol adds a cumbersome process in the transmission process, which requires three handshakes before the client can send to guarantee the stable establishment of the connection. In the transmission process, there are also a variety of mechanisms, including packet loss detection, retransmission mechanism, checksum mechanism, sorting mechanism, etc. This ensures the correctness of the data transmission, but also transmits a large amount of additional data, which reduces the transmission efficiency. From the client sends a request to the server to receive the request, the transmission data will be copied four times: the client application will write the message to the write buffer, the client write buffer will be copied to the socket buffer, the server socket buffer will be copied to the server read buffer, the server application will be read out of the message. Four copies take more time and increase the transfer time.

2.2. SHMIOP Protocol

Located in the same processor within the components of the communication, in fact, the processor process communication, the operating system provides a variety of inter-process communication, which is based on shared memory is the most efficient way, other ways such as pipelines, named pipelines, message queues and so on are based on the shared memory to achieve. The basic idea of shared memory is to create a shared memory area, which allows multiple processes to access and exchange data, as if the data is located in the local address space of each process. When the server starts, create a shared memory area, when the client needs to communicate with the server, take the initiative to establish a connection with the server, the process of establishing a connection is to open the shared memory area, after which both sides of the communication will be able to exchange data through the shared memory area. After the end of communication, the client disconnects, releasing the right to operate the memory area. Mapping GIOP to shared memory produces the Shared Memory Inter-ORB Protocol (SHMIOP) protocol, which allows more efficient interconnection within the same device.

Interoperable Object Reference (Interoperable Object Reference, IOR) is a data structure, CORBA uses IOR as a common means of identifying an object, the server in the start-up end of the shared memory address encapsulated into the object IOR, the client to get to the server object IOR, parse and from the IOR to get the When the client obtains the IOR of the server object, it parses and gets the shared memory address from the IOR, and links the shared memory address through the address, which is equivalent to establishing a connection with the server. When the client needs to call the server-side operation, it will write the request message into the shared memory address. When the server listens to the information written in the shared memory through the event processing mechanism, it will read the information from the memory address and process it, and finally write the processing result into the shared memory address to complete the communication.

3. SHMIOP PROTOCOL DESIGN APPROACH

3.1. Modules and Functions

In the process of realization, the thesis follows the basic structure of the pluggable protocol framework in e*ORB, and on the basis of the basic interfaces provided by it, the thesis designs and realizes each functional module for the SHMIOP protocol, which is used to complete the establishment of the connection and the transmission of the message in the whole communication process. The division of specific modules and their respective functions are shown in Table 1.

Table 1. Functional description of each module

Module Name	Function of the modules
SHMIOP_Listener	Listens for connection establishment request and connection closure request
SHMIOP_Connector	Realize connection establishment and connection closure, send and receive data
SHMIOP_Profile	Initialization of protocol information, matching between client and server
SHMIOP_Factory	Creates protocols, connectors, and listeners

In this paper, four modules, SHMIOP_Listener, SHMIOP_Connector, SHMIOP_Profile, SHMIOP_Factory, are designed. SHMIOP_Listener is mainly used by the server side, and its function is to listen to the connection information of the client side. SHMIOP_Listener is mainly used by the server side to listen to the client's connection information and process accordingly when it receives a request to establish or disconnect the connection. SHMIOP_Connector is the implementation of establishing and disconnecting the connection and the function of writing and reading to the shared memory area. SHMIOP_Profile is the initialization of the protocol information and determines whether the target matches the target when the connection is established. SHMIOP_Factory is the design of four modules: protocol, connector and supervisor. Factory function is to create protocols, connectors and listeners.

Table 2. Description of storage information in each area

Shared Memory Area	Information stored in the area
SHMIOP_CON Area	Connection request information of the client.
SHMIOP_CFG Area	Connection configuration information of the server.
SHMIOP_REQ Area	Client's request information
SHMIOP_ANS Area	Server's answer information

In the implementation process, the SHMIOP protocol will create a connection shared memory area (SHMIOP_CON), a configuration shared memory area (SHMIOP_CFG), a request shared memory area (SHMIOP_REQ), and an answer shared memory area (SHMIOP_ANS), and the specific functions are shown in Table 2. SHMIOP_CON records the client's connection information, and writes the connection information when the client needs to establish a connection to the server, and writes the shutdown information when it needs to close the connection. SHMIOP_CON records the client's connection information, which is written when the client needs to establish a connection to the server, and closes the connection when it needs to close the connection. SHMIOP_CFG records the address information of SHMIOP_REQ and SHMIOP_ANS, which is written by the server after responding to the client's request for connection. SHMIOP_REQ records the client's request information, which is written when the client needs to send data to the server. SHMIOP_ANS area records the answer information of the server, which is written when the server needs to send data to

the client. Among them, SHMIOP_CON area, SHMIOP_CFG area, SHMIOP_REQ area and SHMIOP_ANS area are created and closed by the server.

3.2. Specific Functional Implementation

The SHMIOP_CON area and SHMIOP_CFG area are created at runtime on the server side. When the server-side IOR is created, it writes the SHMIOP protocol information and includes the information of SHMIOP_CON area and SHMIOP_CFG area. After the client acquires the server-side IOR, it will parse the SHMIOP protocol of the server-side, and when the client chooses to use the SHMIOP protocol, it will obtain the information of the SHMIOP_CON area and the SHMIOP_CFG area. When establishing a connection with the server, the client writes a connection request to the SHMIOP_CON area, thus sending a SHMIOP connection request to the server. The server listens to the SHMIOP_CON area and receives the information, then creates the SHMIOP_REQ area and SHMIOP_ANS area and writes its information into the SHMIOP_CFG area, and the client reads the information in the SHMIOP_REQ area and the SHMIOP_ANS area and realizes the establishment of the connection. The client writes request information in the SHMIOP_REQ area, the server listens to the SHMIOP_REQ area and reads the request information, then it writes answer information in the SHMIOP_ANS area, the client listens to the SHMIOP_ANS area and reads the answer information. If the client does not write information to the SHMIOP_REQ area for more than a certain period of time or writes connection closure information to the SHMIOP_CON area, the server closes the SHMIOP_REQ area and the SHMIOP_ANS area, thus realizing the disconnection of the connection.

The shared memory created on the server side is used for information interaction between the client and the server side, so both sides will read and write to it, and sometimes there will be more than one client making requests to the server side at the same time. However, in order to ensure that memory usage is not chaotic, it is necessary to ensure that only one object has the right to use the memory at the same time, which requires the creation of signals and mutexes for each piece of shared memory.

In order to protect the integrity of the data in the shared memory, so that communication between the client and the server can be strictly synchronized, the server also needs to provide synchronization for the establishment of shared memory. The server establishes six semaphores with a maximum count value of 1, which are used to indicate the read and write permissions of the four areas. Because the same service object may have more than one client at the same time, in order to avoid more than one client at the same time with the server communication, but also need to add a mutual exclusion to ensure that at the same time only one client and the server communication.

Both the server and the client, after the completion of the read and write operations on the shared memory, the corresponding signal quantity value should be changed. In order not to cause memory leaks, but also to ensure that the memory is released after use, to ensure the robustness of the program.

4. TRANSMISSION PERFORMANCE EXPERIMENT

This paper builds a simple scenario that includes a client and a server, and the client sends test data to the server. The experiment was carried out using the SHMIOP protocol and the IIOP protocol to compare the transmission mechanism of the two. The two experiment environment, experiment tools, the interfaces and so on everything is exactly the same, the only difference is the underlying transport mechanism. The experiment environment is shown in Table 3.

Table 3. Environment used for experiment

Processor	ARM
Operating System	Linux
Operating system version	19.4

In the first comparison experiment, the client sends 1k byte size of data in a single transmission for a total of 10000 transmissions. The server receives the data and copies it out. Before sending the data, the client assigns a value to the sent data and ensures that the content of each packet is different. The server will copy out the received data and check the data content and packet number to verify the packet loss rate and the bit error rate of different protocol transmissions.

After the experiment, the transmission test using SHMIOP protocol and IIOP protocol respectively, the packet loss rate is 0% and the bit error rate is 0%. In the transmission of SHMIOP protocol and IIOP protocol have good stability, can ensure that in the test process, each packet of data from the client to the server at the same time, the content of each packet of data can realize the complete transmission, will not produce data errors.

In the second comparison experiment, the client respectively carries out several groups of transmission tests, and each group of transmission tests carries out 1000 packets of transmission, and different groups of tests respectively send different sizes of data. The client calculates the transmission delay for each packet and calculates the average transmission delay. The server side will only copy the received data data, will not check the data content and the number of packets, which ensures that the server side receives the data at the same time only calculates the time of the transmission process, blocking the other processes.

When using CORBA for communication, the delay of each call is on the order of microseconds, so it requires a highly accurate measurement tool to accurately measure it. Linux provides the clock_gettime function, which allows you to obtain nanoseconds of time, which is sufficiently accurate for calculating transmission delays on the order of microseconds. By calling this function at the beginning and end of a transfer function, you can get the time it takes to run the code, and thus accurately measure the time used for a single call.

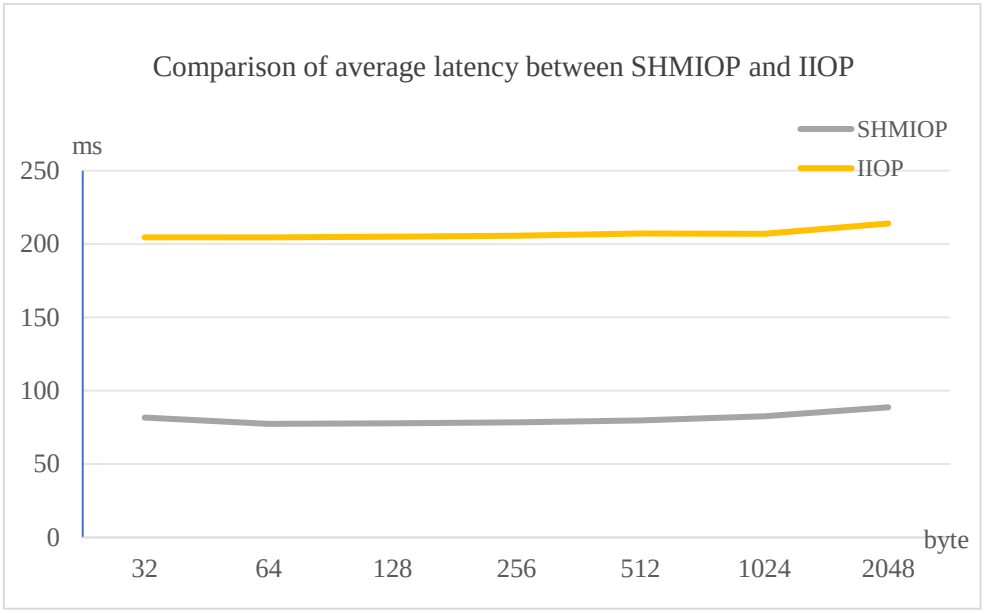

Figure 1. Comparison of average latency between SHMIOP and IIOP

The results of the average delay are shown in the Fig. 1, comparing the average delay of transmission using SHMIOP protocol with that of transmission using IIOP protocol, whether it is SHMIOP protocol or IIOP protocol, as the size of a single packet of data increases, the delay required for transmission of a single packet increases, and therefore the average delay of transmission increases, this is due to the fact that as the size of the message increases, it takes more time for coding and decoding as well as for transmission. This is due to the fact that as the message size increases, more time is taken up in coding, decoding and transmission. Whether using SHMIOP or IIOP, the results for a single packet size of 32 are larger than those for a single packet size of 64, due to the need to

link the client to the server and initialize the relevant data at the beginning of the test, which takes up some of the time and is counted in the transmission time.

Comparing the SHMIOP protocol with the IIOP protocol, it can be found that the average delay of the SHMIOP protocol implemented in the paper is less than that of the IIOP protocol under the same experiment conditions, and the transmission efficiency of the SHMIOP protocol is higher. Moreover, with the increase of single packet data size, the average delay increase of SHMIOP protocol is smaller, and the transmission efficiency improvement is more obvious.

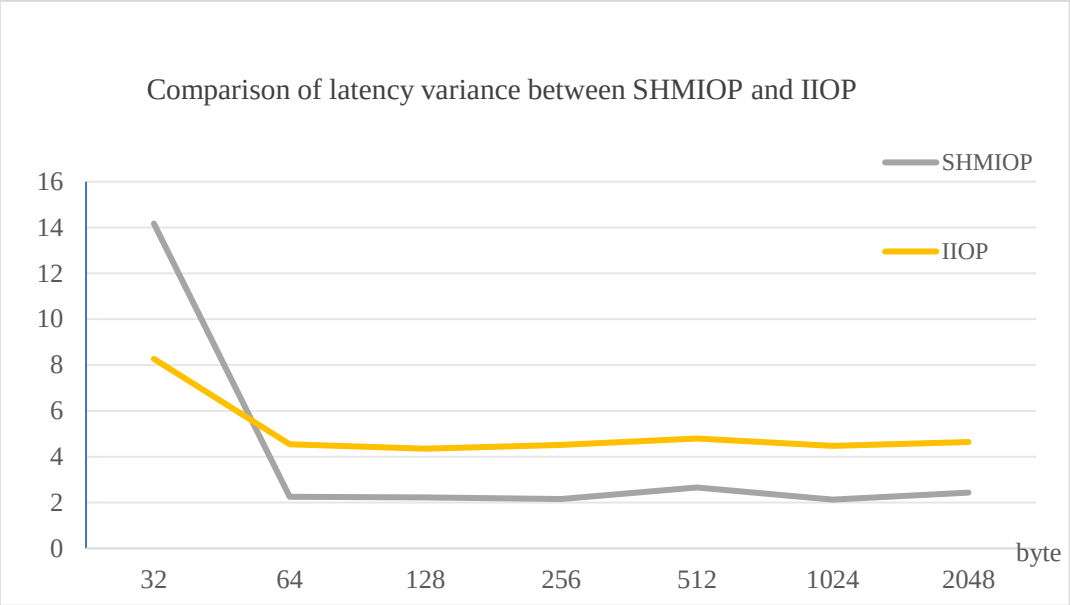

Figure 2. Comparison of latency variance between SHMIOP and IIOP

The experiment results of transmission variance are shown in Fig. 2. Comparing the transmission variance of transmission using SHMIOP protocol with that of transmission using IIOP protocol, there is no significant change in the transmission variance with the increase of single packet data size for both SHMIOP and IIOP protocols. However, the results for a single packet size of 32 are all larger than the results for a single packet size of 64, which is due to the fact that more time is taken up at the initial stage of the test, and the results are in line with the theoretical analysis above.

Comparing the SHMIOP protocol with the IIOP protocol, it can be seen that the transmission variance of the thesis implementation of the SHMIOP protocol is less than that of the IIOP protocol during stable operation and under the same test conditions. However, the transmission variance of the thesis implementation of the SHMIOP protocol is larger than that of the IIOP protocol for a single packet size of 32, which indicates that the thesis implementation of the SHMIOP protocol has a better transmission stability, however, it takes a longer time for the establishment of the connection.

5. SUMMARY

Distributed CORBA provides scalable and flexible solutions for client/server environments as well as Internet applications, and this high degree of compatibility indirectly leads to performance loss. In this paper, based on analyzing the OMG GIOP protocol, we propose a design methodology for the shared memory-based SHMIOP protocol, which is tested and verified on the e*ORB middleware. However, the protocol requires more time for client-server connection and has a large impact on the transmission variance at the beginning of the transmission. Further analytical studies will be done based on this aspect in the future.

REFERENCES

[1] July R. The Common Object Request Broker: Architecture and Specification [J]. Computer Integrated Manufacturing Systems, 1991, 64(11):401. DOI:10.1007/978-0-387-39940-9_2227.

[2] Lei Pengbin, Wang Ling, Wu Yu, et al. Optimization Design and Implementation of CORBA Middleware in Software Defined Radio System [J]. Computer Engineering, 2016, 42(6):43-47, 54.

[3] Li Fang, Zhang Hong. GIOP and How to Improve the Performance of Distributed CORBA Applications [J]. Microcomputer Information, 2006(07X):4. DOI:10.3969/j.issn.1008-0570.2006.21.003.

[4] Jang J H, Lee D G, Choi W, et al. Design and Implementation of CORBA Inter-ORB Protocol Based on Shared Memory for Communication Systems [J]. Kips Transactions Parta, 2003, 10A(3). DOI:10.3745/KIPSTA.2003.10A.3.231.

[5] Fedak G, Germain C, Neri V, et al. Xtrem Web: A Generic Global Computing System[C]//Preceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid. Washington D.C., USA: IEEE Press, 2001:582-587.