Research on Real-time Publish/Subscribe System supported by Data-Integration

Lidong Zhai, Li Guo, Xiang Cui, Shuhao Li
Research Center of Information Security, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190
Email: nakusakula@gmail. com

***Abstract*—At present, distributed real-time applications flourish, including the military systems, telecommunications, factory automation, traffic control, financial trading, medical imaging, and so on, these applications will need to make real-time reliable data transmission. Publish/Subscribe interaction scheme is very suitable for distributed communications system applications. In this paper, based on the OMG DDS (Data Distributed Service) middleware, we study and design applied to the field of distributed real-time applications, to support real-time data integration of the Publish/Subscribe system RMOS (Real-time Message Oriented System).**

***Index Terms*—publish/subscribe; DDS; Distributed real-time applications; Database**

I. INTRODUCTION

Recent years, with the continuous development of computer technology and Internet and the continuous requirements of the calculation cost by business, computer application systems develop from centralized computing to distribute computing [9]. Software architectures also switch from C/S mode to a multi-layer application architecture which will inevitably need support by middleware. Distributed system is widely used. The fields which have been developed and put into use are e-commerce, automation technology, workflow control technology, traffic control systems and so on.

Various application systems architectures show a network-centric trend, which puts forward a higher demand to the real-time ability and dynamic flexibility of communication systems. Distributed publish/subscribe (Pub/Sub) communication model has the features such as asynchronism and Multi-point communications, which makes the participants in the communications completely decoupling in space, time and control flow [1]. A variety of distributed systems can share data efficiently in one network and perfectly meet the requirements of loose communications of large-scale distributed systems.

At present, there are many kinds of publish/subscribe middleware such as CORBA event service [2], JMS [3], TIB/Rendezvous [4], JEDI [5], SIENA [6], DREAM [12], S-ToPSS[13][14][15], A-ToPSS[16] and so on[17]. CORBA event service [2] is built on the remote method invocation or the design basis of "object-centered", which is not supported by application-level QoS, and is suitable for point-to-point communications, but not suitable for a number of distributed real-time applications. JMS [3] includes point-to-point and publish/subscribe two information models and provides reliable information transmission, the mechanisms of event and information filtering, which is suitable for large-scale data-centered network. TIB/Rendezvous [4] system is based on collaboration, and its critical idea of cooperation model is to search address according to topics. JEDI [5] is a publish/subscribe middleware based on Map. The customers' subscriptions conditions are consist of "and" operations of the various atoms bound conditions and only property is involved in each atom bound condition, which is often referred as the plane model [10]. SIENA [6] is similar to JEDI in communication model and difference is their support for mobility, which does not belong to the scope of the discussion.

Data Distributed Service (DDS)[7][11] standard, developed by OMG, which is the first wide and available standard as the center of "Publish/Subscribe". DDS can also reasonably and effectively control and deploy QoS parameters required by real-time system while providing advanced abstract interfaces for the developers of real-time applications. However, because DDS is the basic middleware only for the senior developers, its reliability is not considered. Mathematical models of a specific public/subscribe system has been given in [8]. Therefore, this paper considering supporting the database on the basis of DDS, rebuilt the DDS structure, developed a real-time publish/subscribe system RMOS (Real-time Message Oriented System) and designed a new three-layer structure of the Publish/Subscribe supported by Data-Integration, thus reliable and real-time transmission of publish/subscribe communication paradigm can be ensured.

The research progress of the publish/subscribe middleware is firstly introduced in this paper; RMOS system frame structure designed by this paper is outlined in the second part; Simulation testing toward RMOS is made in the third part; the conclusion that the reliability and real-time ability of RMOS system is good is drawn, and it can meet the field of large-scale distributed real-time applications.

II. THE FRAME STRUCTURE OF RMOS SYSTEM SUPPORTING DATABASE

doi:10.4304/jsw.6.6.1133-1139

A. DDS Architecture[7]

DDS specification unifies the interfaces and behaviours of data distributing, transmitting and receiving in the real-time system, defines the data-centric publish/subscribe mechanism, and provides an independent model not based on platform, which can be mapping into various actual platforms and programming languages. DDS specification is aimed at promoting the effective and reliable data distribution in distributed systems.

DDS specification describes two levels of interfaces:

- A lower DCPS (Data-Centric Publish-Subscribe) level that is targeted to provide proper recipients the efficient delivery of the proper information
- An optional higher DLRL (Data Local Reconstruction Layer) level allows a simple integration of the Service to be integrated to the application layer.

DCPS layer is the core of DDS specification, providing the basic architecture of data distribution. This model is based on the concept of a “global data space” and all the data objects exist in the space. Typed interfaces are used by distributed nodes to simply “read” and “write” and to visit the data objects they are interested.

DCPS are divided into two subsections:

- The Platform Independent Model (PIM).
- The Platform Specific Model (PSM) for the OMG IDL platform based on the PIM. Figure 1 shows the PIM architecture:

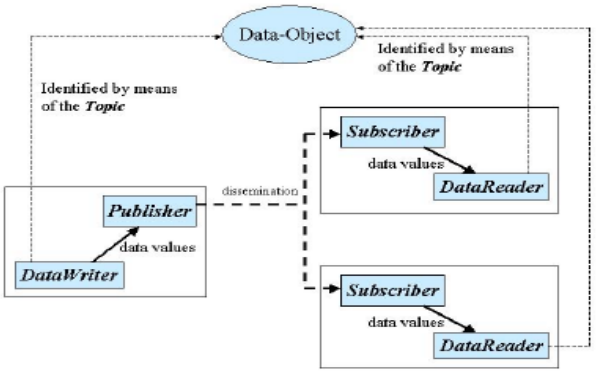

Figure 1: PIM Overview

PIM is mainly comprised of Domain, Publisher, Subscriber, Data Writer, Data Reader and Data Object. Domain is the basic unit of DCPS. Any entity belongs to a domain and can only interact with other entities in the same Domain. Publisher writes typed data through the relevant Data Writers. Subscriber receives and reads typed data through the relevant Data Readers. Topic is the connecting point between publishers and subscribers. The middleware links communicate data by topic, data type and QoS setting.

QoS Policy is one of the key concepts of DDS. Including Topic, Data Writer, Data Reader, Publisher and Domain Participant, all the entities have their own QoS Policies. QoS parameters can control communication mechanism, transmitting priority, reliability, bandwidth, renewing speed and time resource restriction, etc.

To guarantee the high performance and reliability of the communication, the QoS policies of the data publisher need to conform to that of the data reader. The subscriber provides an expected value of QoS, while the publisher provides a conforming value of QoS, and then the middleware checks if the two QoS are compatible. If they are compatible, communications between the two sides will be established. If not, system service will not establish communication and both publisher and subscriber will be notified. Thus QoS provides a great flexibility for communication.

B. The framework of RMOS system based on DDS

The basic framework of DDS standard proposed a "global data space" concept. The data can be read by other readers from the global data space. The units publishing information are called the "publishers"; the units that need to receive the designated data are called "subscribers. " Global data space is achieved by information repository (InfoRepo) which is responsible for matching publishing and subscription (according to the name of the topic and QoS parameters).

When the matching is successful, the data writer arranges the data from the application into a certain format and then gives it to the publisher. The publisher transmits the data to subscribers in accordance with the required transmission mode and transmission configuration. Subscribers receive data in accordance with the specific transmission method and transmission configuration, and give the data to the data readers who read out required information from the fixed data format and hand it to the application.

In the field of distributed real-time applications, the real-time ability and reliability of the system are mainly considered. And the structure of DDS ensures the real-time ability of the publish/subscribe communications model. DDS has been recognized, especially in the urgent tasks of aviation, national defense and telecommunications and distributed real-time systems.

DDS criterion has standardized the interfaces and behaviors of data distribution, transmission and reception in the distributed real-time system, defined the data-centered publish/subscribe mechanism, and provided a platform-independent data model, which can be mapped to a variety of specific platforms and programming languages. DDS is aimed at promoting efficient and reliable distribution of data in distributed system.

However, because it is only a basic regulation, various problems in practical applications are not considered. For example, the architecture design of the actual application scenes is not taken into account. The most important problem in the performance is that the reliability of the system is not considered. In DDS structure, the InfoRepo providing exchange information of topic match is only online present. Once the InfoRepo providing information

match failed, the publish/subscribe communications would be forced to terminate, reducing the reliability of the system. What's more, the combination with data storage is not taken into consideration in DDS criterion either, but in order to resolve the issue of the reliability of the system, combining it to the database is a very effective way.

Therefore, DDS structure is reconstructed in RMOS system in this paper, and a new three-layer structure of data publish/subscribe supporting database is created, thus the reliable transmission of the communication mode of publish/ subscribe can be guaranteed. Specific network structure is shown in Figure 2.

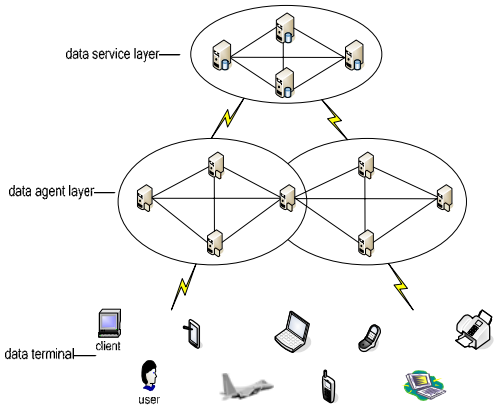

Figure 2: Network structure of RMOS system

The RMOS system that we will design based on DDS is mainly divided into three layers, the data service layer, the data agent layer and the data terminal layer. Oracle databases are added into traditional DDS structure in data service layer which combines publishing module, subscription module and the InfoRepo of DDS with large-scale databases; Agent local databases and fixed publish/subscribe agents are included in data agent layer; Data terminals include the public/subscribe terminals. The design of data service layer is concerned in this paper.

Data service layer is a fully distributed network structure of data exchange, in which each node mainly includes two parts, InfoRepo and database management system (DBMS) in DDS. We use DBMS to store the information in the InfoRepo and use the strategy of synchronous replication of databases to make data can be shared in the whole data service layer, so that the entire RMOS system can still provide data distribution services reliably even if some InfoRepo is cut off. The following figure 3 is the figure of data interaction of improved DDS.

In Figure 3, we assume that there are three InfoRepos, InfoRepo_A, InfoRepo_B, InfoRepo_C; the three InfoRepos interact with three databases respectively, whose main functions are to read and store. The three databases are DBMS_A, DBMS_B, DBMS_C.

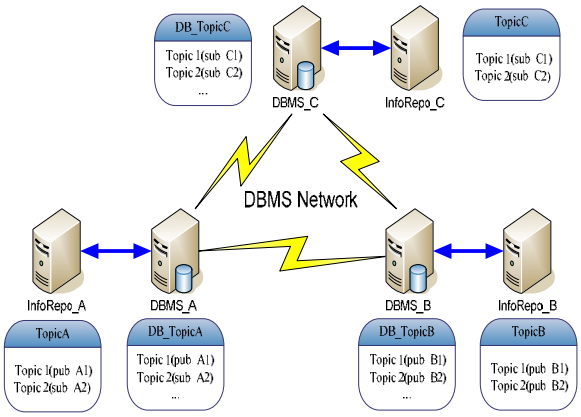

Figure 3: Information interaction in the data service layer of RMOS system

When each InfoRepo received publish/subscribe topics, they will be stored in their corresponding databases, and the publish/subscribe topics are copied in the interval of setting synchronous replication according to the specific scenes of distributed real-time application.

Here, we constructed the following two scenes to compare DDS and RMOS approach.

Scene 1:

Assuming that InfoRepo_C received published topic Topic 1, after a certain period of time the published topic Topic 1 was copied by the DBMS network, and then InfoRepo_A received subscribed topic Topic 1.

In the original DDS, because there is no data replication between InfoRepos, so when InfoRepo_A received subscribed topic Topic 1, it has to be waiting in the InfoRepo_A for the corresponding topic to be matched.

In RMOS system, because a data replication network has been built among the InfoRepos, the corresponding published topic Topic 1 can be matched. So communication links can be established between subscribed topics and published topics, and real-time publish/subscribe can be achieved more effectively.

Scene 2:

Assuming that InfoRepo_C received published topic Topic 1, after a certain period of time published topic Topic 1 was replicated in the DBMS network, when InfoRepo_C failed.

In the original DDS, because the topics stored in InfoRepos are recorded in the on-line memory, at the time the record of published topic Topic 1 disappeared. The publisher of published topic Topic 1 must wait and can't publish or subscribe information until InfoRepo_C restarts the service, or the publisher of published topic Topic 1 needs to find another InfoRepo to publish the information.

In RMOS system, because the data replication network is established among InfoRepos, when some InfoRepo InfoRepo_C disconnected and couldn't provide services, as InfoRepo_A and InfoRepo_B have stored the

information of Topic 1, the topic still can be matched in the data service layer. The ability of real-time publish/subscribe is enhanced.

These are our analysis of common scenes of real-time publish/subscribe. It can be seen that the scheme of data replication designed by RMOS system can enhance the real-time ability of publish/subscribe communication mode, and more effectively guarantee the applications of distributed real-time system.

The introduction of databases will involve the time of data storage. Below, we will analyze the typical publish/subscribe scenarios theoretically to prove the reliability and real-time ability of RMOS system.

C. Theoretical analysis

As to the above design of the system, we'll analyze it theoretically. We know that this analysis would usually be useful to publish / subscribe model. The following four points are mainly included:

1. The analysis of the wrong impact on the DDS nodes in network simulation
2. The estimate on the average number of the events that subscribers missed
3. The estimate on the size of the cache
4. The fixation of error level

Here, we have identified the following assumptions:

- If the link breaks in the process of publish/subscribe, data transmission is delayed until the link is joined. Hence the senders and receivers wait until the process has been fully restored.
- If subscribers disconnect, the lost events will be kept in the durable and stable records which database management system (DBMS) manages. Subscribers can use these durable and stable records to access the events which occurred during the disconnection time.
- If senders disconnect, another sender will replace it, and the data exchange will not be suspended.

In order to measure the impact of above errors on DDS architecture, the following parameters are identified:

λ_{pub}: Publication rate of publishers,

λ_{sub}: the rate of receiving the published events for subscribers,

λ_{fail}^{L}: the disconnection rate of the communication links,

t_p and t_s: the delay between publishing and subscribing, (t=tp+ts)

$\lambda_{\mathrm{Recov}}^{sub}$ and $\lambda_{\mathrm{Recov}}^{L}$: the recovery rate of subscription and links.

On subscribers' sides, we assume that there are i events occurred during failure time. When a DDS subscriber recovered from failure, it obtained data from DBMS preserving durable and stable data. The probability that i events occurred during the interval between disconnection and restoration is:

$$p=\left(\frac{\lambda_{pub}}{\lambda_{pub}+\lambda_{recov}^{sub}}\right)^{i}\frac{\lambda_{recov}^{sub}}{\lambda_{pub}+\lambda_{recov}^{sub}} \tag{1}$$

The maximum number of events that DBMS can save is denoted as NL. If i>NL, the event is missed. However, the average number of events that subscribers miss is

$$E(N)=\sum_{i=NL+1}^{\infty}\left(\frac{\lambda_{pub}}{\lambda_{pub}+\lambda_{recov}^{sub}}\right)^{i}\frac{\lambda_{recov}^{sub}}{\lambda_{pub}+\lambda_{recov}^{sub}}(i-NL)$$

$$\Rightarrow E(N)=\left(\frac{\lambda_{recov}^{sub}}{\lambda_{pub}+\lambda_{recov}^{sub}}\right)^{NL}\frac{\lambda_{pub}}{\lambda_{recov}^{sub}} \tag{2}$$

NL = 0 corresponds with the case that there is no DBMS. It can be seen from the formula (2) that the bigger NL is, the smaller the average number of missed events is.

However, according to actual scenes, in the network design of data services layer, because the users' demand amount that each InfoRepo can provide is different, the amount of stored events of the database corresponding to the InfoRepo can't be infinite. Otherwise it will affect the overall performance of entire data network. That is to say NL cannot be infinite.

Assume that te is the time spent to traverse each event. After each time that subscribers' connection is restored, the average time required for subscribers to access missed events from the database is:

$$E(t)=\sum_{i=1}^{N_L}p(i)*i*t_e=\sum_{i=1}^{N_L}\left(\frac{\lambda_{pub}}{\lambda_{pub}+\lambda_{recov}^{sub}}\right)^{i}\frac{\lambda_{recov}^{sub}}{\lambda_{pub}+\lambda_{recov}^{sub}}*i*t_e$$

$$=t_e\frac{\lambda_{pub}}{\lambda_{recov}^{sub}}\left[1-\left(\frac{\lambda_{pub}}{\lambda_{pub}+\lambda_{recov}^{sub}}\right)^{N_L}\frac{\lambda_{recov}^{sub}*N_L}{\lambda_{pub}+\lambda_{recov}^{sub}}\left(\frac{\lambda_{pub}}{\lambda_{pub}+\lambda_{recov}^{sub}}\right)^{N_L}\right] \tag{3}$$

We can see from the above formula that the bigger NL is, the bigger E(t) is.

III. RMOS SYSTEM PERFORMANCE TEST

A. Simulation test

We will carry out simulation tests toward the above deduction to support reliability and the real-time ability of the RMOS system supporting database.

We take $\lambda_{\text{Recov}}^{sub}$ =0. 5 in Matlab Simulation. NL was equivalent to 0, 5, 10, 20, 30, 40 respectively. The following is the figure from the simulation toward formula 1, in which the horizontal axis corresponding to λ_{pub} is an independent variable and vertical axis corresponding to $E(N)$ is a dependent variable.

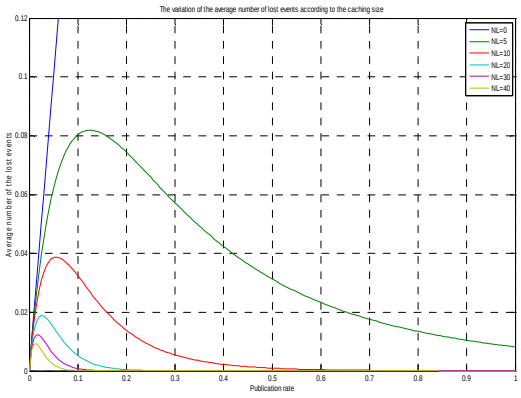

Figure 4: $(\lambda_{recov}^{sub} = 0.5)$ The variation of the average number of lost events

When $N_L = 0$, the curve depicts a publish/subscribe system without lasting databases. It can be seen from figure 4, supporting databases and replicating data synchronously in the publish/subscribe structure can effectively reduce the loss of information events enhancing the reliability of the RMOS system.

The following is the figure from the simulation toward formula 2, in which the horizontal axis corresponding to λ_{pub} is an independent variable and vertical axis corresponding to $E(N)$ is a dependent variable. In simulation, the time spent to traverse each event te is 0. 1s in. The gotten figure 5 is shown below:

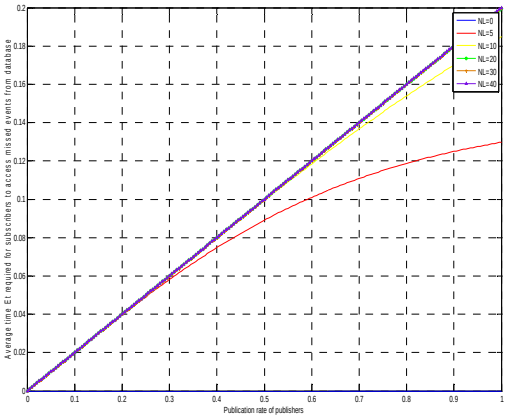

Figure 5: The event response time when NL increases

When NL was equivalent to 20, 30, 40, the curves nearly coincide. Tiny differences can be seen only after being enlarged. From this figure, we can see that the response time of increasing database caching size is within 200 ms, meaning that the delay of data transmission remains within 200 ms after the operations of increasing database caching size, which can effectively meet the real-time ability of the RMOS system.

B. *Systems Test and Evaluation*

We analyze our system performance through testing the Publish/Subscribe system. The setting of environment parameters are shown in table 1, targeted at the 3 main components of the Publish/Subscribe system. We focus on the operating time of infoRepo, publish, and subscribe with different transmission protocols, the rate of information packet lost over UDP protocol and the system response test with multiple publishers and subscribers. We perform six groups of tests as follow.

Table 1 Environment Parameter Setting

System	Value	Parameter	Value
CPU	1. 8GHZ	length of message	300Byte
Memory	512M	Maximum of messages	3000
OS	Window XP	Maximum of publishers	50
Hard Disk	80G	Maximum of subscribers	50

Effects of the time-consuming of publisher and subscriber caused by the number of messages are as follows. We adopt TCP, and use the same simple message with single data writer, the number of messages is 10~3000, the time-consuming is shown as Figure 6. We can see that the response time of publishers or subscribers increases along with the increase of the number of messages, and the increasing speed decreases as time goes by.

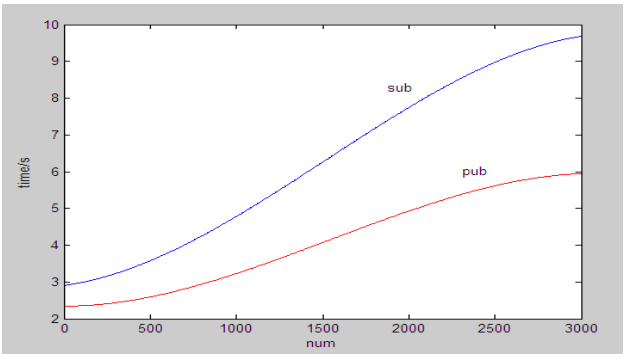

Figure 6: The response time of system along with the increase of messages

With different transmission protocols, effects of time-consuming on publishers caused by message numbers are as follows Figure 7. TCP and UDP are separately adopt, the same simple message is used, and the number of messages is 10~3000. The time-consuming on publishers is tested. We can arrive at a conclusion is Along with the increase of messages, the transmission

time of UDP decreases, while the speed of increase of TCP also decreases.

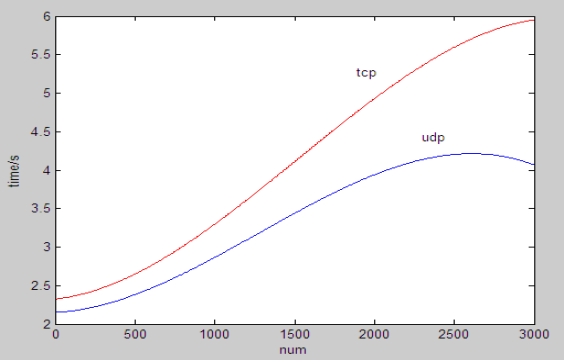

Figure 7: Response Time over different transmission protocols

Effects of packet loss rate caused by the number of messages over UDP are as follows. We adopt UDP, and use the same message (300 byte per message), the number of messages is 10~3000, the packet loss rate is shown as Figure 8. We can see that the packet loss rate tends to be stable, but the total amount of packet loss is still great, this is due to the UDP transmission mode.

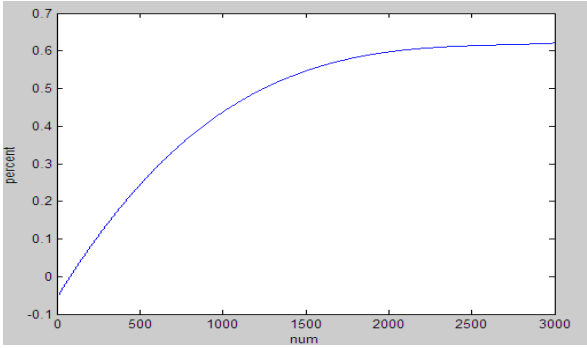

Figure 8: Packet loss rate over UDP

Effects of time-consuming on publishers caused by the number of instances on Publisher and Subscriber are shown as follows. We use TCP with single data writer, the number of messages is 1~1000, the time-consuming is shown as Figure 9. We can see that the response time of the InfoRepo and subscriber is longer than that of the publisher. But the total response time is conform to the system request.

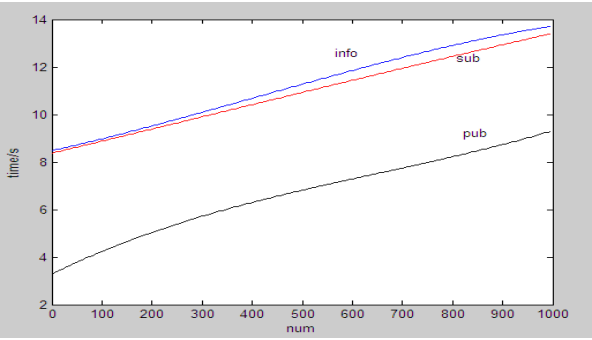

Figure 9: Effect of number of instances on system response time

Effects of time-consuming on InfoRepo, publisher and subscriber caused by the number of data writers are shown as follows. We use TCP, 1~1000 messages, and the time-consuming is shown as Figure 10. We can see that the response time of the InfoRepo decreases along with the increase of the data writers, thus the system efficiency increases.

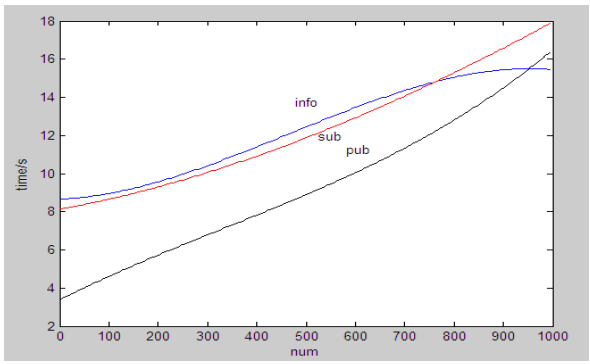

Figure 10: Effects of the number of data writers on system response time

Effects of response time on subscriber caused by the number of publishers are as follows. We use TCP, and the number of publisher are separately 1, 2 and 4, the number of subscribers is 1~50, the time-consuming is shown as Figure 11. We can see that the system response time decreases along with the increase of publisher and subscriber, thus we know that multiple messaging transmissions real-time are handled very well in our system.

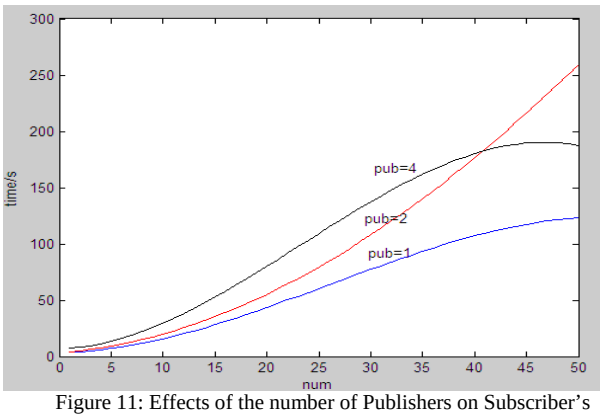

Figure 11: Effects of the number of Publishers on Subscriber's response time

From the tests, we can conclude that the response time of the real-time publish/subscribe system tends to be stable along with the increase of data amounts, slope tends to decrease. However, congestion may occur when great amount of data are transmitted, it will cause system response time to be longer, thus system performance needs to be improved. As for packet loss rate, our system performs well.

IV. CONCLUSION

In this paper, new real-time publish/subscribe system RMOS (Real-time Message Oriented System) which is applicable for the field of distributed real-time

applications and support database has been studied and designed based on the middleware of OMG DDS (Data Distributed Service). Tests has proved that the reliability and real-time ability are good, which can meet the field of large-scale distributed real-time applications. In future work, we will further refine the three-layer network structure built by RMOS system and solve complex scenes in which data service layer responds to the theme of publish/subscribe.

ACKNOWLEDGMENT

This work was supported by Basic Research Program of China (973 Program) under Grant No. 2007CB311100 and "The National Natural Science Foundation of China" (No. 61003261).

REFERENCES

[1] Eugster PT, Felber PA, Guerraoui R, Kcrmarrcc AM. The many faces of publish/subscribe. ACM Computing Surveys, 2003, 35(2): ll4-131

[2] Object Management Group. Real-time CORBA Specification[R]. Version1. 2, Jan. 2005

[3] Sun. JAVA Message Service[R]. Version1. 1, April. 2002

[4] TIBCO Coro. TIB/Rendezvous White Paper, 2000. URL. http: //www. tibco. com/software/enterprise_backbone/rendezvous. jsp

[5] Cugola G, Nitto ED, Fuggetta A. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Trans. on Software Engineering, 2001, 27(9): 827-850.

[6] Carzaniga A, Rosenblum DS, WolfAL. Design and evaluation of a wide-area event notification service. ACM Trans. On Computer Systems, 2001, 19(3): 332—383.

[7] Object Management Group. Data Distribution Service for Real-time Systems Specification[R]. Version1. 1, Nov. 2005

[8] Lidong Zhai, Fan Wu, Tiantian Tang, Yuan'an Liu. Research on the Publish-Subscribe System Based on DDS. Journal of Computational Information Systems, v3, n6, December, 2007, p 2487-2493.

[9] HUANG Gang, WANG Qian-Xiang, MEI Hong, YANG Fu-Qing. Research on Architecture-Based Reflective Middleware. Journal of Software, 2003, 14(11): 1819-1826

[10] MA Jian-Gang, HUANG Tao WANG Jin-Ling XU Gang YE Dan Underlying Techniques for Large-Scale Distributed Computing Oriented Publish/Subscribe System. Journal of Software, Vo1. 17, No. 1, January 2006, PP. 134-147

[11] Hennin M Vinoski S. Advanced CORBA Programming with c++[M]. Addison Wesley I 999

[12] Buchmann A, Bornh6vd C. Cilia M, Fiege L, Gartner F, Liebig C, Meixner M, Miihl G. DREAM: Distributed reliable event-based application management. In: Levene M, Poulovassilis A, eds. Web Dynamics. Springer-Verlag, 2004. 319-352.

[13] Petrovic M, Burcea I, Jacobsen HA. S-ToPSS: Semantic Toronto publish, subscribe system. In: Proc. of the 29th Int'l Conf on Very Large Databases. Berlin: Morgan Kaufmann Publishers. 2003. 1101-1104.

[14] Wang JL, Jin BH, Li J. An ontology-based publish/subscribe system. In: Jacobsen HA, ed. Proc. of the 5th ACM/IFIP, USENIX Int'l Middleware Conf. LNCS 323 1, Toronto: Springer-Verlag, 2004. 232-253.

[15] Burcea I, Petrovic M, Jacobsen HA. I know what you mean: semantic issues in Internet-scale publish/subscribe systems. In: Cruz IF, Kashyap V, Decker S, Eckstein R, eds. Proc. of the SWDB 2003. Berlin: Morgan Kaufmann Publishers, 2003. 5l-62.

[16] Liu H, Jacobsen HA. A-ToPSS: A pub1ish/subscribe system supporting imperfect information processing. In: Proc. of the 30th Int'l Conf. on Very Large Databases. Toronto: Morgan Kaufmann Publishers. 2004. 1281-1284.

[17] Cugola G, Picco GP, Murphy AL. Towards dynamic reconfiguration of distributed publish-subscribe middleware. In: Proc. of the 3rd Int'l Workshop on Software Engineering and Middleware. Orland: Springer-Verlag. 2002. 1 87-202.