`International Journal of Database Theory and Application
Vol.9, No.11 (2016), pp.83-94
http://dx.doi.org/10.14257/ijdta.2016.9.11.08

An Improved Model of General Data Publish/Subscribe Based on Data Distribution Service

Shufen Liu, Xuejun Ma* and Xinyong Wang

College of Computer Science and Technology, Jilin University, Changchun 130012, China
jlmxj08@163.com

Abstract

Most existing data publish/subscribe systems applied in a particular field, the lack of generality. In order to satisfy the general support for interdisciplinary model, proposed an improved model of general data publish/subscribe. The model supports the configuration and modification of the underlying data types. In order to avoid the impact on the application layer while changing the underlying DDS product, proposed an encapsulation on DDS based on the abstract factory pattern. Finally, through simulation experiments to verify the feasibility of the proposed model, the simulation results show that the improved model can be well applied to various types of data publish/subscribe occasions, with high performance.

Keywords*: DDS, Data type, Interface encapsulation, RTI*

1. Introduction

Data Distribution Service (DDS) has been used for many years in military command information systems in foreign countries, the proposed thought of the data-centric coincides with the thought of the information-centric used by command information systems and new type of combat guiding ideology. Some characteristics of DDS itself can solve the problems that the business function of command information systems is difficult to realize. At the same time, DDS has the advantages of real-time and various QoS can be selected [1], the use of DDS in command information systems has significant significance.

However, current research related to this article is mostly concentrated in data distribution service with a particular area [2], the lack of universal. In this paper, the design of the improved model of general data publish/subscribe based on Data Distribution Service (IMGDPS) is the universal publish/subscribe model of DDS communication data for data communication service, and the underlying data types can be configured and modified, which has an independent function of publish/subscribe various types of data. Compared with the model proposed in reference [3], the function is more perfect and the structure is more reasonable.

OMG DDS specification uses the IDL language to describe the DDS implementation how to follow the interface and the semantics of each interface, but does not specify the mapping of specific language, this resulting in the concrete implementation of DDS appeared a variety of different ways of mapping. Currently, the companies provide DDS products have reached more than a dozen, the DDS products they provide support for the specification and the underlying technology used by each are different, such as RTI DDS [4], OpenSplice DDS [5], OpenDDS, *etc.* Application layer programming relies on APIs provided by the specific DDS implementation, although these APIs follow the same semantics, but due to the different forms of presentation, the need for adaptation transplant in multiple DDS implementations. If the underlying DDS-based code could not seamlessly replaced, will lead to portability issues.

ISSN: 2005-4270 IJDTA

This paper proposed the encapsulation of DDS for shielding interface differences among different DDS implementations, and high-level just for programming with the abstracted interface only, without concerning for the corresponding concrete class of DDS products. In this way, no matter what the underlying DDS product changes, the interface encapsulation layer will provide to the upper layer of the conformity and unified abstract interface. The high-level care only about the interface of the interface encapsulation layer, no longer makes the function of the application layer is not coupled with the underlying concrete DDS product APIs. The interface encapsulation layer realizes the function of the application layer and the underlying DDS can change independently.

The paper is organized as follows: section two, analyses the related theories; section three, designs the architecture of IMGDPS, elaborates the encapsulation method of DDS; section four, simulation tests for IMGDPS; section five, gives the conclusion of this paper.

2. The Related Theoretical Basis

2.1. DDS Specification

DDS is the data-centric specification based on publish/subscribe mechanism, developed by the Object Management Group in order to meet the demand for distributed real-time communication [6-8]. It provides a platform-independent data model, using Global Data Space [9] instead of central server, for managing the entire distributed system of the topic publication, topic subscription, the maintenance of the node information, and the relevance of the nodes. DDS has inherent QoS policy can be used to control the transmission quality of data distribution. DDS specification only defines its APIs through the platform independent language IDL, so that DDS is platform independent.

2.2. The Hierarchy of DDS

DDS specification is divided into two layers [6,10], they are Data Centric Publish/Subscribe (DCPS) layer and Data Local Reconstruction Layer (DLRL). DCPS layer provides the API interface, meta-information data types and event notification mechanism of the message publish/subscribe, settings and semantics of the QoS policies, the whole architecture of the middleware, *etc.*

DCPS layer can make data be transmitted from publish side to subscribe side more quickly and efficiently, if you want to integrate with the application level, it can be considered using a higher level of DLRL. DLRL is an optional part of DDS specification, which specifies DDS integrates with the application layer in a simple seamless way. Using DLRL, developers can define their own classes, make their own classes attached with some communication entities of DCPS layer, then through the operation of the class object to call the actual DCPS layer operation, thus give the user a better programming experience. The hierarchy of DDS is shown in Figure 1.

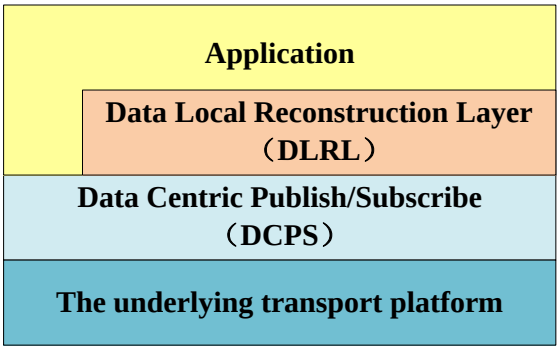


Figure 1. The Hierarchy of DDS

3. The Design of IMGDPS

3.1. Architecture Design of IMGDPS

The design of the system structure is usually according to the hierarchical approach on the system to do a very coarse-grained grouping about class, package or subsystem, having the responsibilities of the main aspects of the system to be cohesive [11-12]. IMGDPS require independent function of publish/subscribe various types of data, and the underlying data types can be configured and modified. According the requirements, we design the architecture of IMGDPS, it is divided into three levels: Technical Service layer, Domain layer and the User Interface layer. The architecture is shown in Figure 2.

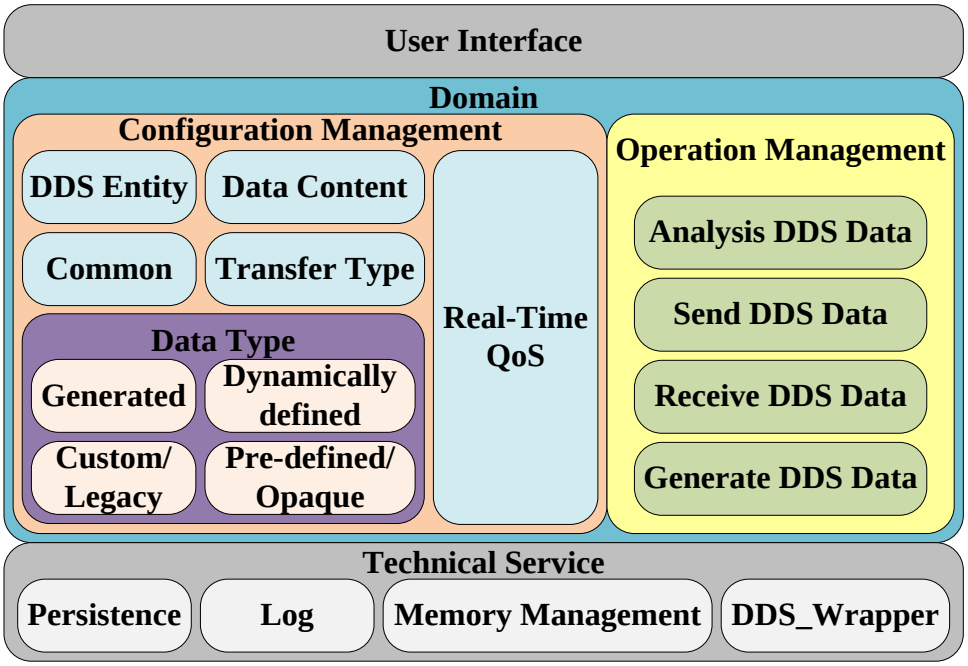


Figure 2. Architecture of IMGDPS

(1) Technical Service layer

Technical Service layer contains four modules: DDS_Wrapper, Memory Management, Log and the Persistence.

The DDS_Wrapper module implements the encapsulation and functional adaptation of DDS, this will be discussed in detail in Section 3.2.

In order to improve the reliability of the model, due to the continuous request or possible waste of resources, the Memory Management module mainly uses the Bridge Pattern to realize a different combination of storage mode and storage area types. By limiting the maximum number of available resources, combined with the logic cycle semantics to achieve the purpose of the model running for a long time.

The Persistence module implements the access to the configuration file.

(2) Domain layer

Each entity in the Domain can only communicate with entities in the same Domain, does not feel the existence of entities in other Domain, so the data publish/subscribe of different Domain can work independently. Domain layer contains two modules: Configuration Management and Operation Management.

Configuration Management completes the configuration operation, it includes six sub-modules: Data Type configuration, DDS Entity configuration, Data Content configuration, QoS configuration, Transfer Type and Common configuration. With Data Type configuration module, we can generate new data types; dynamically modify the supported data types of the model, set the field name and field types, as well as the name and key field information of data types; we can use custom/legacy or pre-defined/opaque data types in our model expediently.

Operation Management completes the operation management, it includes four sub-modules: Generate DDS Data, Receive DDS Data, Send DDS Data and Analysis DDS Data.

(3) User Interface layer

User Interface layer contains all the functionality of the user interface. The layer is used to display the specific publish/subscribe information of the configured subsystems, display the received data information, as well as the users interactive fill the data and trigger the completion of data publication.

3.2. Design of DDS_Wrapper

Although all of DDS products follow the same interface specification, due to the use of techniques are different, there are still some subtle differences in specific interface presentation, it hinders the use of the code written by DDS to be unchanged used in other DDS products. For this model, because the future choice of DDS products is uncertain, proposed the DDS_Wrapper layer, thus shielding the APIs differences between specific DDS products, has the vital significance to the system development, integration and deployment.

(1) The encapsulation thought

Abstract factory pattern provides an interface to create a series of related or mutually dependent objects, without specifying their concrete classes [13]. The pattern is used for shielding the underlying differences, is a construction mode of providing unified interface to the high-level. It is used to design a series of correlated objects and class libraries of some products, just want to show its interface to the high-level and hide the underlying implementation using this. The pattern is often combined with the singleton pattern in common use, it separates the specific product realization and the interface expression, so that implementation can evolve independently of the interface.

For DDS, we define an abstract factory class IEntityFactory, the class declares the interface used to create all other DDS entities. Each specific DDS entity abstracts an interface corresponding to it. IEntityFactory defines the creation method of the abstract interface to a series of specific entities, when using the specific implementation of DDS, only need to implement these interfaces to achieve the specialization of the specific DDS APIs, and then you can directly create the corresponding entity interface to complete high-level functions. High-level doesn't care about the underlying concrete class which is used, the underlying concrete class of DDS is transparent to the high-level.

(2) The encapsulation method

On the basis of DDS encapsulation, we make a specialization of RTI DDS. Including the use of DDS_DynamicData and DDS_TypeCode provide by RTI DDS to achieve the type bindings at the runtime, improves the flexibility. Relations of these interfaces are shown in Figure 3.

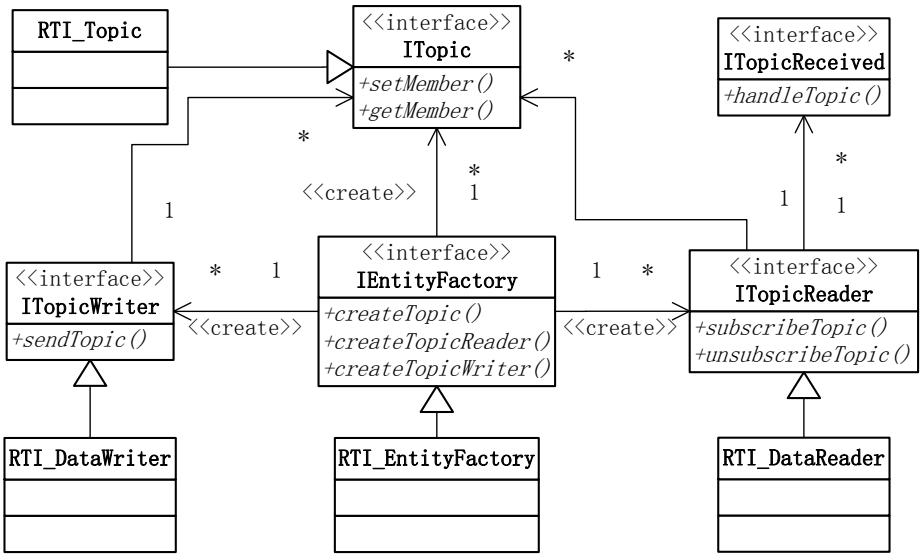


Figure 3. Related Interfaces of DDS Encapsulation

IEntityFactory is the role of the abstract factory, which is used to create a series of abstract products. IEntityFactory as the entry point to the wrapper of DDS application, will also participate in the related operations of DDS domain, including the maintenance and release of DDS resources, the discovery of DDS nodes, *etc.* IEntityFactory exists as a container for a series of entities it created, maintains the life cycle of entities it created, and provides its internal entity query work, *etc.* IEntityFactory is a pure virtual interface, in actual use needs specialized for specific DDS, as shown in Figure 3, we made a specialized version of RTI DDS, RTI_EntityFactory as the entry point of all RTI DDS related operations.

ITopic interface is the abstraction of topic and data samples associated with topic in DDS specification, provides the interface method for read/set the relevant field in data samples. ITopicWriter interface is the abstraction of datawriter and publisher entities in DDS specification, which can be used for publisher to write topic to the network. ITopicReader interface is the abstraction of datareader and subscriber entities in DDS specification, which can be used for subscriber to read topic from the network. ITopicReceived interface can be seen as the encapsulation of the listener mechanism of DDS.

Through the abstracted interface above, we have finished the call between the bottom DDS middleware and the upper application. The upper application depends only on the interface we abstract, and no longer depend on the specific implementation of DDS APIs, the concrete realization of the lower level DDS just need to achieve our interface. Here, our DDS_Wrapper layer is a glue layer, it lifting the coupling of the upper application and the concrete implementation of the underlying DDS, so that the seamless replacement of different DDS products can be realized.

4. Simulation and Evaluation

According to the structure of IMGDPS, the simulation system is constructed and simulated by using multiple computers under the laboratory environment, and complete the required function and performance evaluation.

4.1. Function Evaluation

Function evaluation is to explore IMGDPS for the current simulation system on the function support of data publish/subscribe, by selecting more representative communication scenarios. We select the typical scene of the lab, and the evaluation script is put forward according to the requirement. Scene is the system A and system B all messages interaction, the communication scenario covers all message types of this simulation system and the way the message transmission, Table 1 describes the evaluation script and execution results.

Table 1. Evaluation Script and Execution Results

Operation	Results
A subscribe M1_Info, M2_State, M3_Info, M6_Info B subscribe M1_Info, M4_Data, M5_State, M6_Info	Using RTI Analyzer to view the publication and subscription information of nodes
A send M4_Data	B received M4_Data, the data is consistent
B send M1_Info	A and B received M1_Info, the data is consistent
A cycle to send M6_Info	A and B continuously received M6_Info data update
B cycle to send M2_State	A continuously received M2_State data update
A update M5_State	B received M5_State data update
B update M3_Info	A received M3_Info data update

From Table 1 we can see that the simulation system completely covers the script of the scene, and the simulation results show that IMGDPS is fully adapted to the functional requirements of system A and system B for the data publish/subscribe in the scene.

4.2. Performance Evaluation

Through measuring the message transmission delay, jitter, maximum transmission rate and throughput to evaluate the performance of IMGDPS, analyze the change trend of the performance, to check whether they are able to meet the performance requirement of the existing simulation system [14-15].

(1)Delay

Select the one-to-one and one-to-many (test using 4 units) for DDS messaging, according the different length of messages, measure the message delay respectively. Using 100Mbps network card for different sizes of messages transmission, at least about 1000 samples were collected from the transmission interval of 10ms, take the average delay. Results as shown in Figure 4, abscissa is the message size (byte), ordinate for the delay time (microseconds).

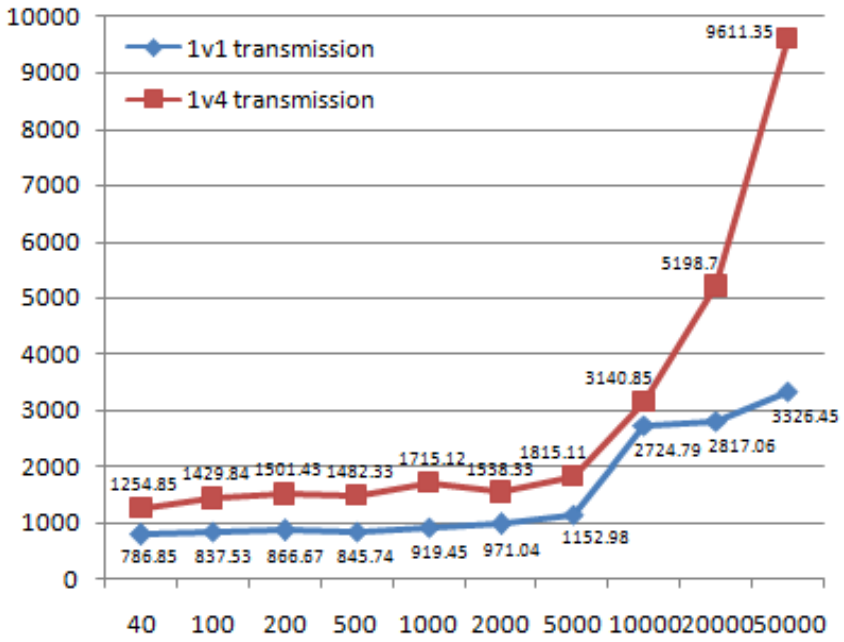


Figure 4. Delay

When the sample size is less than 5000 bytes, the delay is similar in below 2000 microseconds and the increase is not obvious; but when the sample size rise to 10000 bytes, whether it is one-to-one or one-to-many transmission, the delay will have a more substantial increase, and with the increase of sample size, the trend of the delay also increases linearly. One-to-many, using the network analysis tool Wireshark for analysis, in the case without changing the default QoS policies, each packet will be delivered in the form of point-to-point. So from the analysis of results shown above, when increasing the number of subscribers, delay will increase.

From the perspective of practical application, the actual size of sending and receiving packets for the application layer is usually less than 4k (typically less than 1k), in this case using IMGDPS the delay is smooth enough to meet the requirements of real-time communication.

(2)Jitter

According to the delay sample values, the jitter of the system delay is calculated. Results as shown in Figure 5, abscissa is the message size (byte), ordinate for the delay jitter rate (ratio).

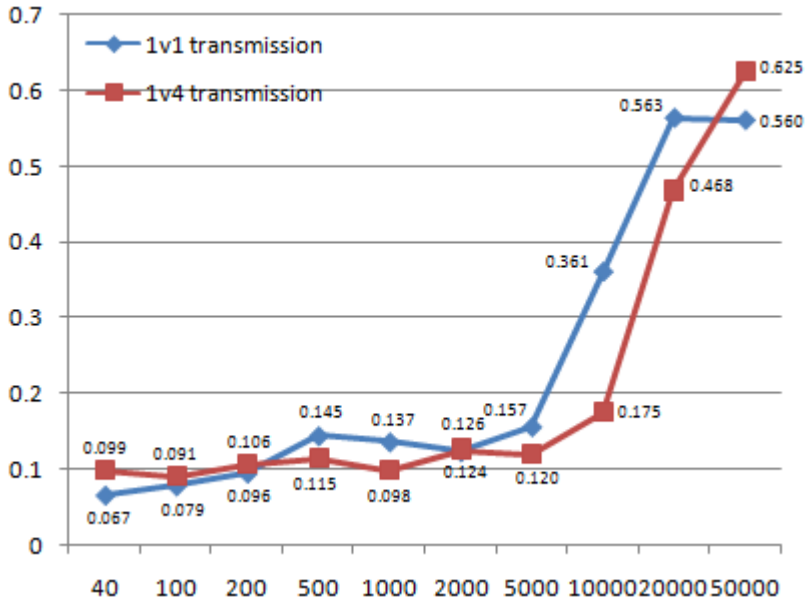


Figure 5. Delay Jitter

From Figure 5 we can see that with the increase of packet size, whether it is one-to-one or one-to-many transmission, when the package size is less than 5000 bytes, delay jitter rate steady at around 10%, when more than 5000 bytes, delay jitter rate increased rapidly. Because as packet size increases, the corresponding IP packets will be more and more when the network layer is transmit. More delay will cause the instability of the delay, which means the delay jitter rate will increase significantly.

From the practical application point of view, the message packet size of the application layer is generally less than 5000 bytes, in this case, the delay jitter rate steady at about 10%. This provides a basis for forecasting the worst case of the system, it proves that the system under the worst case delay is not big, indicating IMGDPS is sufficient to meet the requirements of the delay stability.

(3)Maximum transmission rate

Select the one-to-one for DDS messaging, just to send and receive one topic. On sending machine sends DDS messages continuous uninterrupted, ensure the subscriber can receive the messages. Constantly changing the length of the messages, recording the number of messages that can be sent per second is the maximum transmission rate. The sender uses two kinds of network cards for measurement: 100Mbps and 1000Mbps. Results as shown in Figure 6, abscissa is the message size (byte), ordinate for the message rate (n/s).

From Figure 6, the maximum transmission rate is almost constant and at a high level when the sample size is no more than 500 bytes. When more than 1000 bytes, the two kinds of network cards shown different message transmission rate. Using 100Mbps network card, the sample size less than 500 bytes exhibited a similar message transmission rate; samples of 500 to 5000 bytes in size have the consistent message transmission rate, but is lower than samples less than 500 bytes in size; when more than 10000 bytes, message transmission rate reduced significantly. Using 1000Mbps network card, in addition to the sample size is less than 500 bytes show a consistent message transmission rate, with the increase of sample size, the message transmission rate declines linearly.

When the data timeliness requirements are high, the smaller data format should be used to transmit DDS packets to improve the message transmission rate, meet the real-time requirements of the specific communications.

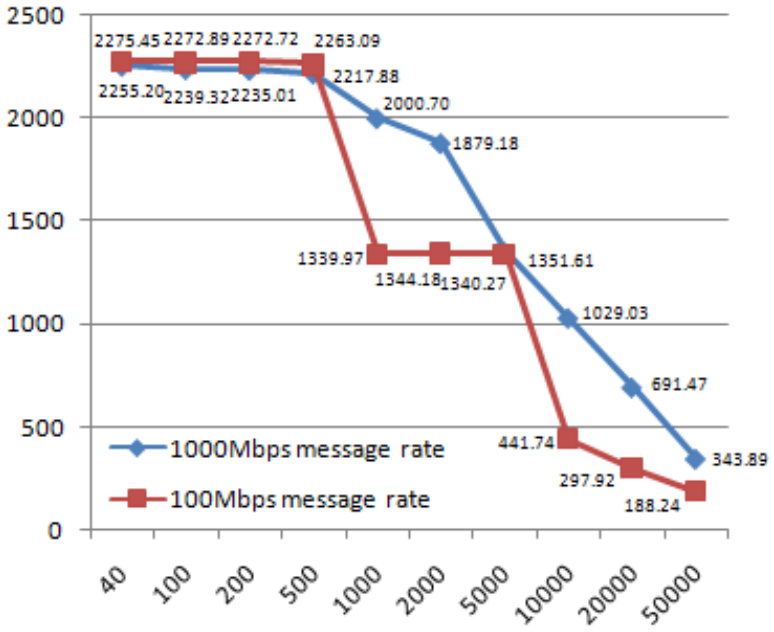


Figure 6. Maximum Transmission Rate

(4)Throughput

Select the one-to-one for DDS messaging, just to send and receive one topic. On sending machine changing the sending frequency, send 10000 messages, ensure the subscriber can receive the messages, until get a critical maximum transmission rate. Constantly changing the length of the messages, recording the total time used for sending 10000 messages. The sender uses two kinds of network cards for measurement: 100Mbps and 1000Mbps. Results as shown in Figure 7, abscissa is the message size (byte), ordinate for the bandwidth (MB/s).

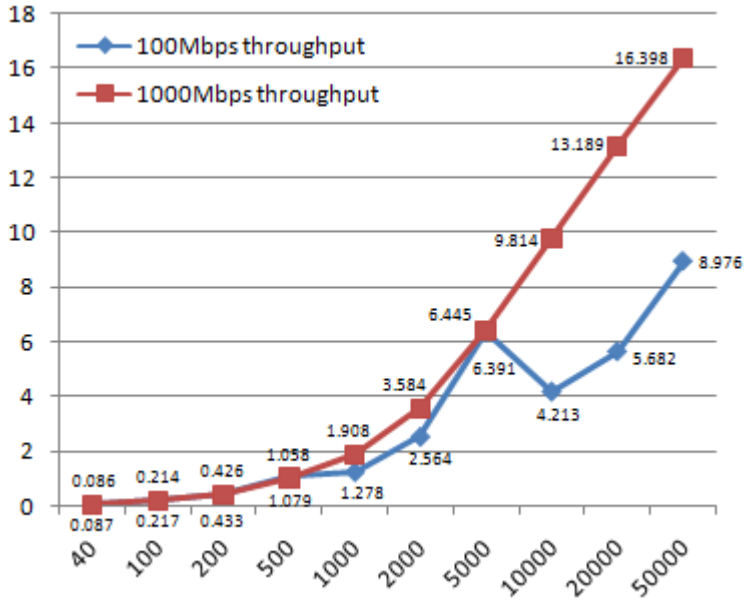


Figure 7. Throughput

From Figure 7 we can see that the throughput of 100Mbps and 1000Mbps network cards is similar changes, when transmitting larger bytes messages will have larger bandwidth occupancy rate, increase the sending time. With 100Mbps network card, when

the packet size of 10000 bytes, the throughput will decline. This is because the limited bandwidth of the network card, when the packet size is 10000 bytes, the amount of packets sent per unit time decrease, and packets are not big, thus caused the decline of sending bytes, resulting in the decline of the throughput. This in Figure 6, when the packet size reaches 10000 bytes, the maximum transmission rate will be drastically reduced, which is also illustrated this situation by the other hand. But 1000Mbps network card does not have this problem.

When data timeliness requirements is not high and the amount of data is large, suitable for larger data formats to send data packets using DDS, in order to increase the bandwidth utilization ratio. Because larger packets, the overall trend of throughput is better.

5. Conclusion

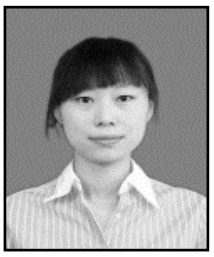
IMGDPS achieves the function of data recording and data transmission using DDS communication service middleware. The model supports the configuration and modification of the underlying data types, its communication flexibility is very well. We made an encapsulation on DDS based on the abstract factory pattern to avoid the impact on the application layer while changing the underlying DDS product. Through the analysis of the evaluation, IMGDPS can be well applied to various types of data publish/subscribe occasions, can completely satisfy the data publish/subscribe function and the demand for performance, can provide the support for software development, testing and debugging of the next generation equipment.

References

[1] M. Mazouzi, S. Hasnaoui and M. Abid, "Challenges and solutions in configuring, rapid developing and deploying of a QoS-enabled component middleware", 2008 3rd International Design and Test Workshop, **(2008)**, pp. 221-224.
[2] J. Ma, T. Huang, J. Wang, G. Xu and D. Ye, "Underlying Techniques for Large-Scale Distributed Computing Oriented Publish/Subscribe System", Journal of Software, vol. 17, no. 1, **(2006)**, pp. 134−147.
[3] X. Ma and S. Liu, "Research on General Data Publish/Subscribe Model Based on DDS", Applied Mechanics and Materials, **(2013)**, pp. 2491-2494.
[4] RTI Inc, "RTI Data Distribution Service: The Real-Time Publish Subscribe Middleware User's Manual v4.5", http://www.rti.com/.
[5] J. H. V. Hag, "Data-centric to the max, the SPLICE architecture experience", 23rd International Conference on Distributed Computing Systems Workshops, **(2003)**, pp. 207-212.
[6] G. P. Castellote, "OMG data-distribution service: Architectural overview", 23rd International Conference on Distributed Computing Systems Workshops, **(2003)**, pp. 200-206.
[7] N. Wang, D. C. Schmidt, H. van't Hag and A. Corsaro, "Toward an adaptive data distribution service for dynamic large-scale network-centric operation and warfare (NCOW) systems", IEEE Military Communications Conference MILCOM, **(2008)**, pp. 1-7.
[8] J. M. Schlesselman, G. P. Castellote and B. Farabaugh, "OMG data-distribution service (DDS): architectural update", IEEE Military Communications Conference MILCOM, vol. 2, no. 2, **(2004)**, pp. 961-967.
[9] OMG, "Data Distribution Service for Real-Time Systems", Version1.2, http://www.omg.org/docs/formal/07-01-01, **(2007)**.
[10] OMG, "The Real-time Publish-Subscribe Wire Protocol DDS Interoperability Wire Protocol Specification", Version2.1, http://www.omg.org/spec/DDSI/2.1/PDF/, **(2009)**.
[11] OCI, "OpenDDS Developer's Guide", Version3.4, **(2013)**.
[12] OMG, "The Common Object Request Broker: Architecture and Specification", **(2002)**.
[13] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Longman. Inc., **(1995)**.
[14] O. Jun, C. Zhiming and W. Ximin, "Performance Test Based on DDS Middleware", Ship Electronic Engineering, vol. 31, no. 11, **(2011)**, pp. 136-139.
[15] S. B. Moon, P. Skelly and D. Towsley, "Estimation and removal of clock skew from network delay measurements", IEEE INFOCOM, vol. 1, no. 1, **(1999)**, pp. 227-234.

Authors

Shufen Liu, currently she is a professor in College of Computer Science and Technology of Jilin University. She has been conducting research for many years on computer network and security technology, computer supported cooperative work, computer simulation technology, software programming method based on model-driven, *etc.*



Xuejun Ma, she is a Ph.D. candidate in College of Computer Science and Technology of Jilin University. Her research area covers software engineering, computer network and cooperative computing.