doi:10.3772/j.issn.1006-6748.2016.04.007

Integration of naval distributed tactical training simulation system based on advanced message queuing protocol[①]

Liu Yu (柳　玉)[②], Wen Jiayan, Ji Daqin
(The Scientific Research Department, Naval Marine Academy, Guangzhou 510430, P. R. China)

Abstract

Aiming at the problems of unreliable data transmission, poor steadiness, nonsupport of complex data types, direct couple between data transmission and exchange, a high-level method based on advanced message queuing protocol (AMQP) is proposed to integrate naval distributed tactical training simulation system after serious consideration with current information exchange features of military combat system. Transferring layer in traditional user datagram protocol is implemented by publishing and subscribing scheme of message middleware. By creating message model to standardize message structure, integration architecture is formulated to resolve potential information security risks from inconsistent data type and express data transmission. Meanwhile, a communication model is put forward based on AMQP, which is in the center position of the whole transmission framework and responsible for reliably transferring battlefield data among subsystems. Experiments show that the method can accurately post amounts of data to the subscriber without error and loss, and can get excellent real-time performance of data exchange.

Key words: tactical simulation, message middleware, advanced message queuing protocol (AMQP), software architecture, distributed interactive simulation

0 Introduction

To conduct naval tactical train simulation and exercises in modern warfare, personnel, weapon platforms, detection equipments, command systems and a large number of relevant information and data are integrated properly and placed at the same scenario region. By this way, different number of virtual troops and equipments can be trained under unknown fighting environment to improve the contract or cooperative engagement capability. Meanwhile, the scientific assessment of the training effects is also achieved. Since the naval tactical training simulation system is composed of different subsystems which are made by independent manufacturers, a unified technical standard, planning or guidance is lack and the communication between subsystems is difficult. Thus, the differences in data inputting, mode processing and the interfaces between subsystems and devices can lead to poor reliability and stability of information and data transfer. Distributed tactical training simulation systems can merge platforms with various manufacturers, heterogeneous data services into a new platform with unified management, information integration and resource share, so as to eliminate information islands, improve the system interoperability, and meet the needs of integrated joint pilot training.

Currently, the research between the academia and industry mainly focus on the distributed tactical training simulation system integration theory from two aspects: the integrated model and the integrated technology. The former focuses on establishing the uniform standard for the network system, while the latter is concerned about the efficiency, safety and reliability of the system[1]. For the existing problems, the academia has made a lot of research and achieves certain technological achievements in enhancing the system interoperability, sharing and adapting new platform such as context-aware middleware framework[2,3], middleware technology based real-time distributed systems[4,5], architecture design based on middleware[6-8], system adaptability[9,10], pervasive computing under middleware application background[11], multi-model of middleware application in the cloud computing[12] and so on. At the same time, the distributed system based on middleware technology has been widely used in medical treatment, traffic, military operations, e-government

[①] Supported by the National Natural Science Foundation of China (No. 61401496).

[②] To whom correspondence should be addressed. E-mail: game _ liuyu@ qq. com

Received on Nov. 9, 2015

and other fields. Hu, et al.[13] used messaging middleware to deliver all information and data of warfare resource information management systems for conventional submarine via message, which greatly improved the operational functions of the system, and received fruited operational results. But for the key technology of the distributed tactical training simulation system integration, some issues remain to be resolved, especially for emerging network computing environment and new equipment platforms.

The remainder of this paper is organized as follows: in Section 1, the architecture of the general distributed tactical training simulation system is overviewed and main problems and drawbacks are summarized. Section 2 introduces AMQP and six kinds of messaging models. Section 3 considerably analyzes the requirements of distributed tactical training simulation system integration, establishes the system platform architecture and designs their message and communication model. Section 4 conducts two groups of experiments to show the effectiveness and innovation of the proposed method. Finally a conclusion and an overview of the future work are given in Section 5.

1 Theoretical analysis of distributed tactical training simulation system

1.1 Architecture

The current distributed tactical training simulation system has been implemented for data exchange via standardized service interfaces, independent of the specific platforms, programming languages, operating systems and hardware architectures, as is shown in Fig. 1. Some of its applications as a service can be reused and shared. Compared with the traditional tree-like or hierarchic structure, it is more flexible and can quickly respond to the needs from business decisions.

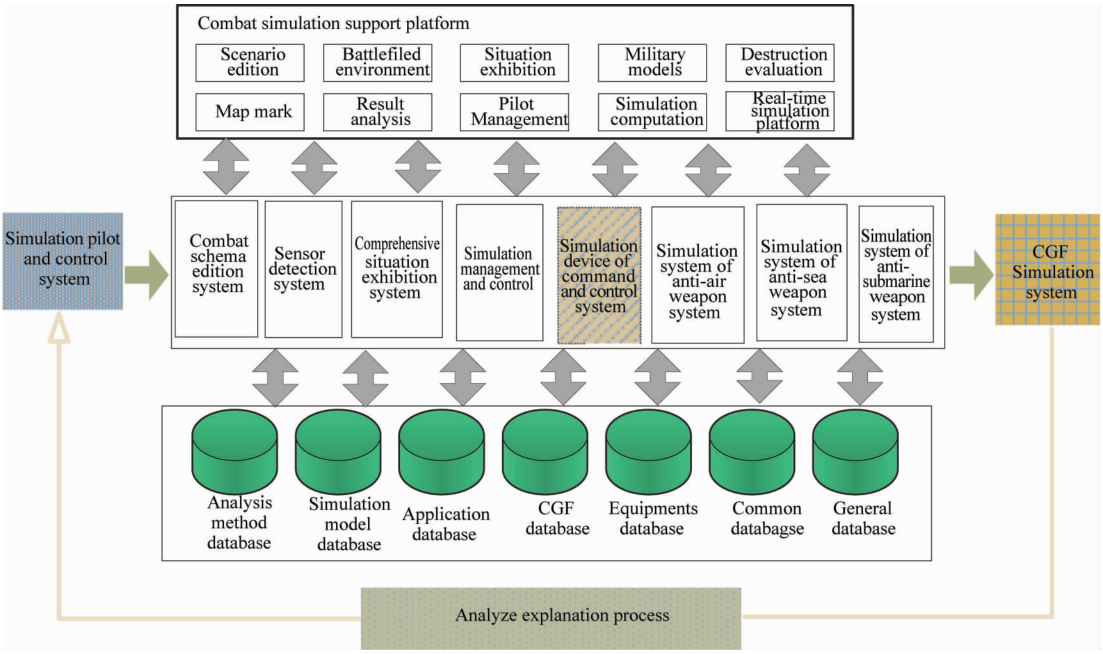

Combat simulation support platform
Scenario edition
Battlefiled environment
Situation exhibition
Military models
Destruction evaluation
Map mark
Result analysis
Pilot Management
Simulation computation
Real-time simulation platform
Simulation pilot and control system
Combat schema edition system
Sensor detection system
Comprehensive situation exhibition system
Simulation management and control
Simulation device of command and control system
Simulation system of anti-air weapon system
Simulation system of anti-sea weapon system
Simulation system of anti-submarine weapon system
CGF Simulation system
Analysis method database
Simulation model database
Application database
CGF database
Equipments database
Common databagse
General database
Analyze explanation process

Fig. 1 The architecture of distributed tactical training simulation system

According to function features, the current architecture can be divided into four layers: a data storage layer, a service component layer (combat simulation support platform), a business logic & implementation layer and a simulation director layer.

The data storage layer is the basis for all subsystems and services, which is used to store system resources and combat simulation models, including the analyzing method library, the simulation model library, military applications, databases, CGF (computer generation forces) library, equipment library, public library and other military data resources. The component services layer utilizes the data storage layer to provide data access interface for obtaining military resources, and achieving supported business functions including services from combat scenario requests to result feedback. The logic layer is the core of the system implemented by a series of corresponding business components, including the operational plan edition, the sensor detection simulation, integrated battlefield situation exhibition, simulation management and control as well as related equipment simulation systems. The simulation director layer controls cooperation and advance operation for the whole system.

During the process of actual operation, the collaboration of all subsystems is needed to finish the whole tactical training simulation task. First, using the operation plan subsystem to generate combat scenario, and outline the geographical scope of operations, participating troops, weapons and equipment, communication means and other information. Next, the simulation pilot and control subsystem release starting signals. Based on the supports of the data layer and the combat simulation platform, the military simulation models then are deployed and moved as planned. The sensor detection system provides location and movement of enemy targets, and reports to the simulation training equipment. Fourthly, the commander issues operational decision commands. The simulation systems can show actual combat procedure, and train commanders and operators to achieve good operational awareness and high strategy capability.

In the above process, the information exchange in the distributed systems includes objectives, environment, military equipments and other physical data which are necessary to complete the whole combat procedures, as well as piloting management commands for interoperability.

1.2 Problems in the existed architecture

1.2.1 Unreliable data transmission

Existing distributed tactical training simulation systems mainly use an unreliable UDP (user datagram protocol) to resolve data transfer between different nodes, which can result in data transmission error and even block in Fig. 2.

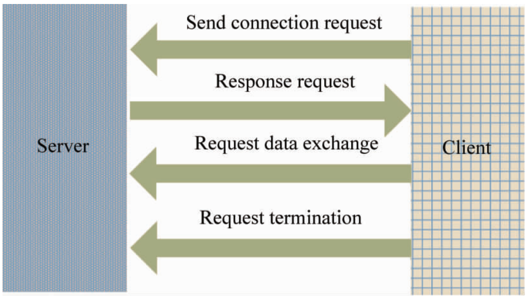


Fig. 2 The data transmission principle based on UDP

By analyzing the data exchange workflow of the distributed tactical training simulation system, the shortcoming can be found below:

(1) When a server sends data, any failure of client will cause the transmission to end;

(2) When the network is delay, the server will use a timeout retransmission mechanism to repeatedly send data to further accelerate the processing load of clients, seriously weaken their computing power.

1.2.2 Poor robustness

The tactical training simulation system generally uses RPC (remote procedure call) as the communication integration mean between applications distributed in the heterogeneous environment, where RPC utilizes software agents to communicate between programs in the different address space, whose data exchange process is shown in Fig. 3.

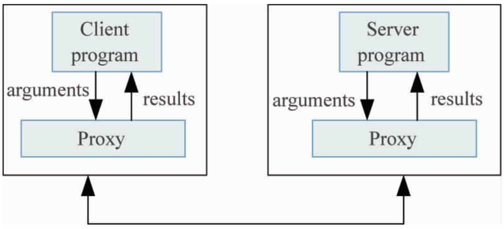


Fig. 3 Data exchange procedure based on RPC

Because the RPC generally uses synchronous communication mechanism, which needs full consideration of network and system reliability, flow control, processing synchronization and other complex issues, thus it is difficult to be implemented in practice.

1.2.3 Difficult to support complex data types

To save network bandwidth and improve data transfer speed, the combat system messaging protocol generally supports integer, floating point, character, Boolean and other simple data types, and unable to directly transmit string, vector, map and other complex types. When the sender needs to transfer complex elements, it must be converted to a simple type, the receiver then reintegrates data in accordance with pre-agreed rules, which will not only increase the development cost of the system, but also reduce the scalability of data exchange platform.

1.2.4 Directly coupled services between data transmission and exchange

To improve the deployment efficiency and reduce cost, the distributed tactical training simulation system usually couples tightly data transferring and processing module at the design stage. The data exchange interface between each two related subsystems must be remained. When the quantity of subsystem increases, interface will be doubled and the maintenance workload will be unbearable. In addition, when the network is in congestion, the high coupling degree of data exchange can cause collapse of the tire application system.

2 AMQP & messaging middleware

2.1 AMQP

AMQP[14], namely advanced message queuing protocol, is an open standard protocol of application layer to provide unified messaging services with advanced message queue, designed for message-oriented middleware, not restricted in specific products and development language. It can greatly reduce the cost of business and systems integration. AMQP can provide industrial-grade integration services to the public with such characters as multi-channel, asynchrony, security, cross-platform, neutrality, efficiency and so on, whose architecture is shown in Fig. 4.

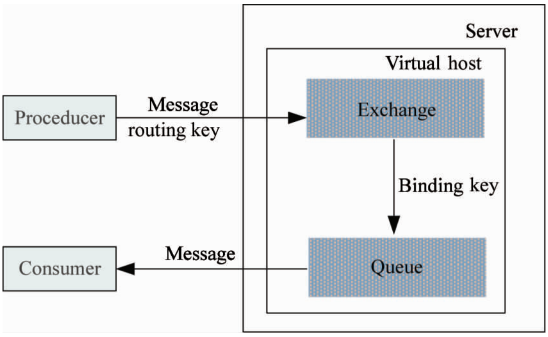


Fig. 4 Standard architecture of AMQP

In the AMQP model, a producer of the message sends a message to the switching node, which is responsible for switching/routing, properly forwarded to the appropriate message queue, and then the consumers read messages from that queue.

AMQP is usually divided into three layers:

- Model layer: defining a set of commands. The client application can use these commands to implement its business function.
- Session layer: responsible for transmitting commands from the client to the server, then the server responds, which provides reliability, synchronization mechanism, and error handling during the entire exchanging process.
- Transport layer: providing the processing frame, multiplexing channel, detection error, data representation and other capabilities. It can be replaced with any transport protocol in realization.

2.2 Communication mode

Middleware is independent of the operating system and database, usually classified by remote control, transaction, database and message. The messaging middleware can improve information and data transmission reliability, stability in distributed system with asynchronous mode, and can support smoothly expanding the system load by cluster. With expedite communication through messaging services, it is ensured that each subsystem is not directly coupled with other systems. When a subsystem fails, it will not have an effect on other systems, and not reducing the overall availability of the system, whose principle is shown in Fig. 5.

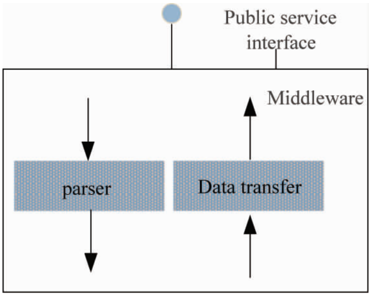


Fig. 5 Communication standard of middleware

AMQP supports the following six kinds of message exchange patterns:

(1) Store and forward (many message senders, a single message recipient)

(2) Distributed transaction (many message senders, many message recipients)

(3) Publish/subscribe mode (many message senders, many message recipients)[15]

(4) Content-based routing (many message senders, many message recipients)

(5) File transfer queue (many message senders, many message recipients)

(6) Point to point mode (single message sender, single message recipient)

3 System integration implementation

3.1 Integration requirement analysis

3.1.1 Architectural requirements

- Compatibility. Since the military equipments are developed as planned, the training simulation system is in flat structure, the interconnections and interoperability among the systems are more difficult. When a new system is built, the legacy systems need to be further considered, not only the future products.
- Loosely coupling. It makes service easier to be integrated or composed of each other, and can provide better application and service management capabilities.
- Independent unanimous protocol. It is not required to connect special system or network, so can greatly improve service reusability.

- Unified infrastructure. It demands strict consistence with development and deployment across all the different applications to enhance system scalability.

3.1.2 Functional and non-functional requirements

- Automation in operating process and assembling streamline.
- Single processing operation can be manually or automatically executed. Flexibly trigger conditions can achieve data copied intelligently.
- Support heterogeneous data sources. A binding is created among the individual data sources, users do not need to know where the source is, and do not care how to transmit the data to the destination.
- Support UDP packets and message queues. These two kinds of data transfer modes can meet almost every data exchange needed for different system architectures.
- With excellent real-time performance. The interval between receiving and transferring data must be less than 50ms for urgent messages.
- With good robustness. The preferred system can flexibly adapts fresh subsystems and network protocols.

3.2 Integration architecture

As the route of data exchange, message middleware can achieve transfer transparently across multi-subsystems. Each member in distributed tactical training simulation system is a connection point to send combating data as messages to middleware when needing to upload data, while middleware server is responsible for exchanging/routing messages to the target. This mechanism can decrease the interface quantities and dependencies of the whole application system. The architecture is shown in Fig. 6.

The message middleware is located at the center and plays a key role in the training simulation system, while the top and the bottom components denote respectively the producers and subscribers of messages. All applications, equipments and middleware connected with interface components can finish their tasks independently. The middleware server can execute uniform schedule and cooperated work, and can resume abnormal transmission immediately to adapt poor communication environment, therefore avoid system thrashing when communication quality declines, and ensure reliable data transmission.

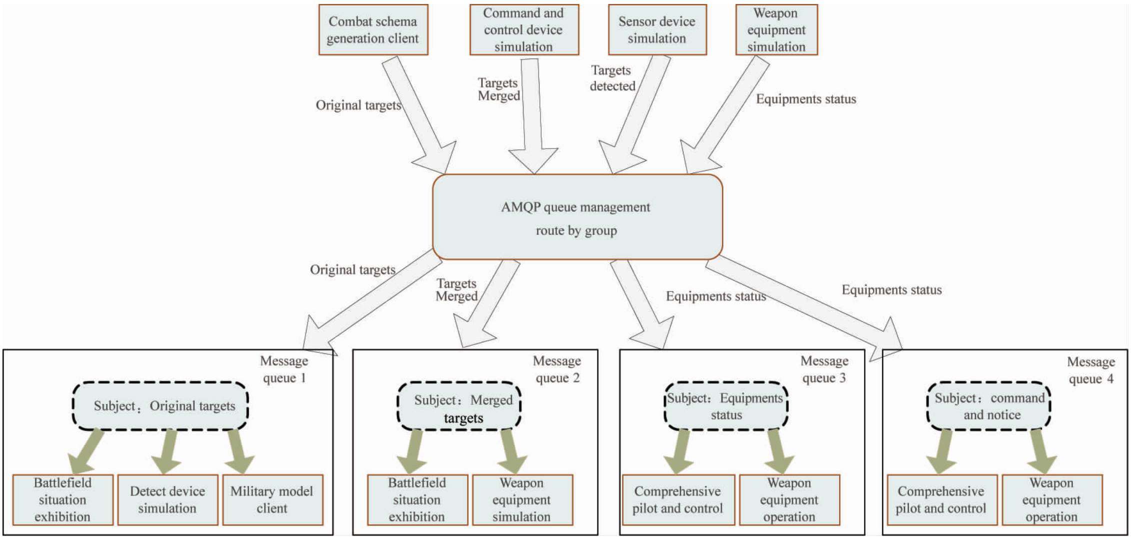

Combat schema generation client
Command and control device simulation
Sensor device simulation
Weapon equipment simulation
Original targets
Targets Merged
Targets detected
Equipments status
AMQP queue management
route by group
Original targets
Targets Merged
Equipments status
Equipments status
Message queue 1
Subject: Original targets
Battlefield situation exhibition
Detect device simulation
Military model client
Message queue 2
Subject: Merged targets
Battlefield situation exhibition
Weapon equipment simulation
Message queue 3
Subject: Equipments status
Comprehensive pilot and control
Weapon equipment operation
Message queue 4
Subject: command and notice
Comprehensive pilot and control
Weapon equipment operation

Fig. 6 Distributed tactical training simulation system integration architecture based on publish/subscribe model

3.3 Message model

Each displaying client in the system can acquire targets and return information from command and control terminals by subscribing corresponding messages. Every member is both supplier and consumer of combating data, located in different service points in order to achieve the information exchange smoothly among subsystems, the unified data protocol and the model designed by XML (extensible markup language) is used. The message middleware can shield the distribution and heterogeneity of data sources. The querying result requested by the local client is returned to middleware, which can attain uniform access to heterogeneous data sources.

There are three types of system messages: target messages, working status messages of equipments and returned messages from the command and control terminal. The first class mainly contains physical parameters such as identifier, location, structure, while status messages are issued by the participating device or subsystem, and can describe their work status and mode. The last one can issue all combating commands needed to announce directors. The message is composed of three parts, the header, the properties and the body. The header includes identification information like message type, subject, priority and so on. The second part describes the message sender, destination, priority, sending date, reply indication, etc. The message body contains the actual transmitted data described by XML.

The target message is different from file transmission, which is always coded by binary form or pure text, and plays a role as the carrier to transmit participated troops, which emphasizes more on the design of message structure. Its generally includes data generation time, sending cycle, source and destination, and message content, etc. Table 1 shows an example of the definition of radar target message in combating protocol. The message structure is standardized to resolve application and information security risks from inconsistent data type and express data transmission.

3.4 Communication model

Operational data transmission is based on the combat network agent implemented by AMQP middleware. The procedures are shown as follows: the combat system director implements combating command program in accordance with real tactical operation regulations. Then the combat network packet agent of tactical training simulation system is responsible for monitoring real-time network data, and collecting the data related with training simulation to upload message server based on the defined message model and expressed by XML. At last the information can be reviewed by all subsystems in Fig. 7.

Table 1 Testing data collection of publishing and subscribing function in tested demo system

ID	Field name	Field length (byte)	memo	ID	Field name	Field length (byte)	memo
1	Unit sequence	1	Filling in packing	13	Relative course	2	
2	Length	2	0x37H	14	Absolute course	2	Invalid
3	Time tick	4	Data generation time	15	Relative speed	2	
4	Spare	2		16	Absolute speed	2	Invalid
5	Controlling word	4		17	Azimuth velocity	2	Invalid
6	Data cycle	2		18	High and low angular velocity	2	Invalid
7	Distance	4	In geographic coordinate system, the length from target point to coordinate origin.	19	X-axis speed	2	In geographic coordinate system, the component of the target speed in three coordinates.
8	Azimuth	2	Range 0° ~360°	20	Y-axis speed	2	
9	Elevation	2	Range 0° ~90°.	21	Z-axis speed	2	
10	X-axis dimension	4		22	Batch number	2	Range 7000 ~7999
11	Y-axis dimension	4	Dimensions in geographic coordinate system.				
12	Z-axis dimension	4					

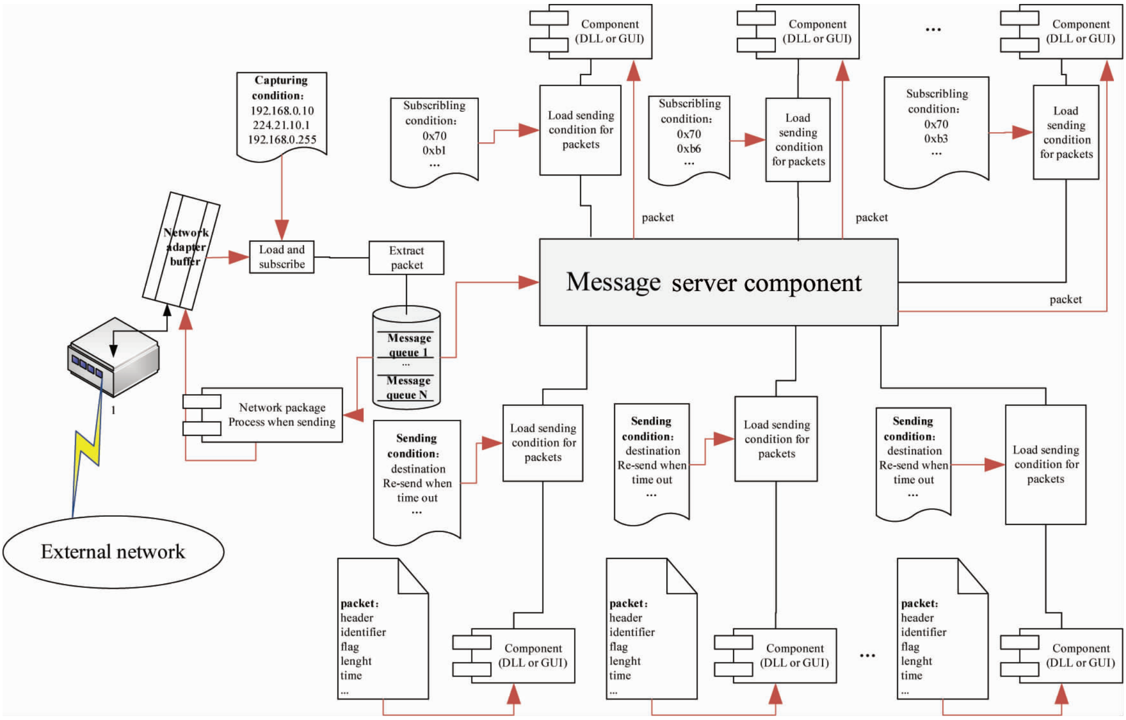

Component (DLL or GUI)
Component (DLL or GUI)
...
Component (DLL or GUI)
Capturing condition: 192.168.0.10 224.21.10.1 192.168.0.255
Subscribling condition: 0x70 0xb1 ...
Load sending condition for packets
Subscribling condition: 0x70 0xb6 ...
Load sending condition for packets
Subscribling condition: 0x70 0xb3 ...
Load sending condition for packets
packet
packet
Network adapter buffer
Load and subscribe
Extract packet
Message server component
packet
Message queue 1
...
Message queue N
1
Network package Process when sending
Sending condition: destination Re-send when time out ...
Load sending condition for packets
Sending condition: destination Re-send when time out ...
Load sending condition for packets
Sending condition: destination Re-send when time out ...
Load sending condition for packets
External network
packet: header identifier flag lenght time ...
Component (DLL or GUI)
packet: header identifier flag lenght time ...
Component (DLL or GUI)
...
packet: header identifier flag lenght time ...
Component (DLL or GUI)

Fig. 7 Communication model of the distributed tactical training simulation system

The combating system acts as a role of producing commanding and controlling messages, while the tactical training simulation system is a consumer. All messages are stored by queues, and the consumer receives messages from them. This method can achieve asynchronous operation and ensure integrity. The operational network client publishes various data acquisition as services. The data listener is a conduit for data processing to bridge communication between external application and messaging middleware. The data collection service in the combating network agent is called according to serving code of training mission configuration, and finishes data processing and acquiring procedure. The messaging middleware is in the core position of the architecture, and can guarantee the reliable data transmission processed by listeners between nodes.

4 Integration experiments and performance verification

In order to verify feasibility, effectiveness of the integration solution, C++ language is used to develop a network service agent, choose Apache QPID 2.0 software based on AMQP as the messaging middleware to build demo system. The experiments are done both from the function and performance.

There are two testing tasks as follows:

- Whether the operational data can be accurately published to subscribers.
- The real-time performance of messaging middleware integration.

The experimenting environment is created by the following configuration: the network agent and messaging middleware server are installed on a same machine, whose processor is the Intel quad-core product i7-3770 3.40G, the memory size is 4G, the operating system is 32-bit Windows 7 Ultimate, and all subsystems, weapons and equipments use thin Lenovo terminals and local area networks.

To simulate the process of real data transmission, a specific operational plan with large targets is selected to generate a certain number of participating troops, then pause at a fix interval and regenerate the same amount of targets until it reaches the maximum. At last, statistics is done on the average results.

4.1 Functional testing and result analysis

To verify the main function of the publishing and subscribing module, the testing user is constructed successively, then original targets are uploaded and returned targets, equipment status from the command & control client are produced. Meanwhile, each display and control terminal are configured to subscribe different topic messages. A program is prepared to check whether the received results of network data is consistent with the sent according to the relationship between the publishing and subscribing clients, which is shown in Fig. 8.

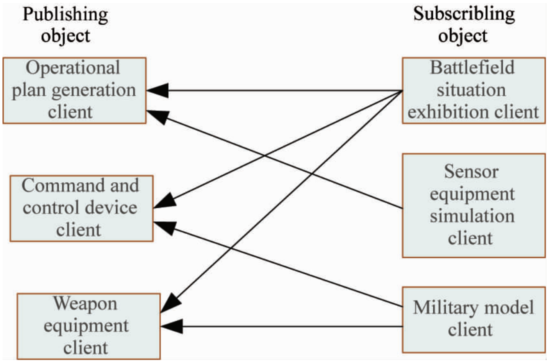

Publishing object
Subscribling object
Operational plan generation client
Battlefield situation exhibition client
Command and control device client
Sensor equipment simulation client
Weapon equipment client
Military model client

Fig. 8 The publish & subscribe relationship in demo system

According to the experimenting procedure, the director generates an operational plan and begins to implement tactical training simulation. Then each client sends and receives messages as planned according to the configuration scheme. The program is used to count the number of each subsystem sent and received within 30 minutes shown in Table 2.

It shows in Table 2 that the battlefield situation exhibition client subscribed simultaneously three kinds of combating data, including operational plan generation, command and control device, and weapon equipments. In theory it can receive all planned messages, which is in accordance with experimental results. Similarly, the sensor equipment simulation client only described the specific message from operational plan generation client, therefore it can receive only one topic message.

Table 2 Testing data collection of the publishing and subscribing function in tested demo system

Subscribing object Publishing object	Operational planning generation client sending message: 4000 group	Command and control device client sending message: 6500 group	Weapon equipment client sending message: 3200 group
Battlefield situation exhibition client	4000	6500	3200
Sensor equipment simulation client	4000	/	/
Military model control client	/	6500	3200

The experimental results illustrate that using AMQP messaging middleware integrates a distributed tactical training simulation system. The messages can be accurately passed to the subscribers and can ensure the integrity of the data without any data loss in multi-to-multi communication mode. At the same time, no overflow of message queues occurs when there is a large number of reporting operational data. This experiment shows this method is effective and reliable.

4.2 Performance testing and result analysis

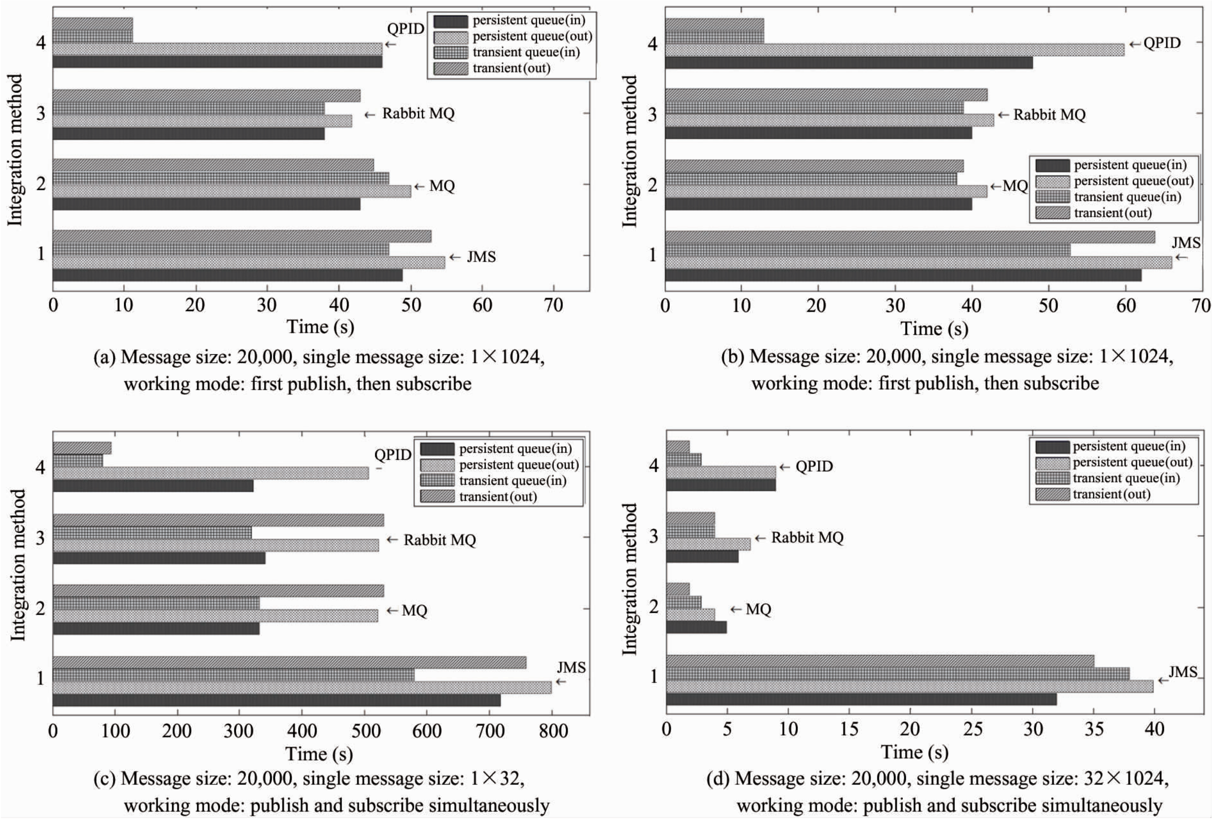
To further validate the real-time processing performance of the method, especially in a small and large number of data packets, the following four groups of datasets are set in Table 3, using two kinds of common message acquisition methods.

Three kinds of messaging middleware integration systems are constructed as a comparative reference respectively based on JMS, MQ[16], and RabbitMQ[17], whose architectures, message subscription relations are consistent with the demo system and the only difference is in the message queue and routing management products used.

Table 3 Testing results of the publishing and subscribing function in tested demo system

Dataset identifier	Size (entry)	Single message size (byte)	Message acquisition method
C1	20,000	1 × 1024	Firstly publish, then subscribe
C2	20,000	1 × 1024	Publish and subscribe at the same time
C3	200,000	32	Publish and subscribe at the same time
C4	200	32 × 1024	Publish and subscribe at the same time

In order to objectively reflect experimental results and minimize error generated by the system, experiments are carried out on the above four groups of test cases one by one. The middleware server is configured respectively as transiently and persistently working mode, sending data packets continuously without interruption, and doing five times for each mode. The specific program is used to calculate the average time from sender to middleware server, or vice versa, and the results are shown in Fig. 9.



(a) Message size: 20,000, single message size: 1×1024, working mode: first publish, then subscribe

(b) Message size: 20,000, single message size: 1×1024, working mode: first publish, then subscribe

(c) Message size: 20,000, single message size: 1×32, working mode: publish and subscribe simultaneously

(d) Message size: 20,000, single message size: 32×1024, working mode: publish and subscribe simultaneously

Fig. 9 Experimental results of real-time performance in tested demo system

The following conclusions can be drawn from the experiments:

(1) Under the same hardware and network environment, large message is used to get better data exchange performance than small ones in each method. Therefore, the integration of distributed tactical training simulation system should pack fragmented messages to group sending, and then the client may further identify messages by topic or keyword in the message body field.

(2) When the middleware is configured in transient working mode, the integrated system based on QPID has better real-time exchange performance than the other three integrated approaches. According to message size, the efficiency can be increased by 60%-80%. Therefore, when the message model is designed, messages from distributed tactical training simulation system should be filtered, and persistent ones should be compressed.

(3) When the middleware is configured in persistent working mode, though the performance of the integrated system based QPID is not the absolutely best, with the increase of the size of a single message, its weaknesses will gradually disappear.

Comprehensive consideration upon various factors, selecting QPID as distributed tactical training simulation system can satisfy performance requirements better. With the increasing of the system communication scale, their real-time data exchange performance can be fully guaranteed.

5 Conclusions

Due to the disadvantages existed in traditional distributed tactical training simulation system, such as unreliable data transmission, poor robustness, not supporting complex data types, direct couple between data transfer and exchange and so on, a fresh method to integrate naval distributed tactical training simulation system is proposed. The integration architecture, functional and non-functional requirements, message and communication model are discussed. By constructing a prototype system, the feasibility and effectiveness of the method is verified. To further improve the real time performance and adaptability of distributed tactical training system, a possible direction for future work is to characterize the efficient message filtering mechanism and middleware awareness ability.

References

[1] Ntanos C, Botsikas C, Rovis G, et al. A context awareness framework for cross-platform distributed applications. *The Journal of Systems and Software*, 2014, 88(11): 138-146

[2] Stjepan S, Juraj G H, Tomislav D. A device-level service-oriented middleware platform for self-manageable DC microgrid applications utilizing semantic-enabled distributed energy resources. *Electrical Power and Energy Systems*, 2014, 54(8): 576-588

[3] Sara H, Animesh P, Valerie I. Service-oriented middleware for large-scale mobile participatory sensing. *Pervasive and Mobile Computing*, 2014, 10(10): 66-82

[4] Isidro C, Luis A, Adrian N. A flexible time-triggered service for real-time CORBA. *Computer Standards & Interfaces*, 2014, 36(11): 531-544

[5] Toral S L, Barrero F, Cortes D, et al. Analysis of embedded CORBA middleware performance on urban distributed transportation equipments. *Computer Standards & Interfaces*, 2013, 35(7): 150-157

[6] Danh L P, Hoan N M, Josiane X P, et al. A middleware framework for scalable management of linked streams. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2012, 16(6):42-51

[7] Alexandre S, Noemi R, Roberto I. A survey of support for structured communication in concurrency control models. *Journal of Parallel Distributed Computation*, 2014, 74(6): 2266-2285

[8] Marisol G V, Pablo B V. Comparative analysis of two different middleware approaches for reconfiguration of distributed real-time systems. *Journal of Systems Architecture*, 2014, 60(8): 221-233

[9] Ren Y F, Li T, Yu D T, et al. Design and testbed evaluation of RDMA-based middleware for high-performance data transfer applications. *The Journal of Systems and Software*, 2013, 86(3): 1850-1863

[10] Janne T, Teemu R, Jouni M, et al. Interoperability of remote handling control system software modules at divertor test platform using middleware. *Fusion Engineering and Design*, 2013, 88(2): 2177-2180

[11] Vaskar R, Jiannong C M. Middleware for pervasive computing: a survey. *Pervasive and Mobile Computing*, 2013, 9(9): 177-200

[12] Lloyd W, Pallickara S, David O. et al. Performance implications of multi-tier application deployments on infrastructure-as-a-service clouds: towards performance modeling. *Future Generation Computer Systems*, 2013, 29(11): 1254-1264

[13] Hu T, Wang S, Hu Z, et al. A study of normal submarine warfare resource information system based on message middleware. *Journal of WUT (Information & Management Engineering)*, 2007, 29(4), 39-49

[14] Organization for the Advancement of Structured Information Standards. ISO/IEC19464 Advanced message queuing protocol 1.0. Boston, USA, 2012

[15] Subramoni H, Marsh G, Narravula S, et al. Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (AMQP) over infiniband. In: Proceedings of the Workshop on High Performance Computational Finance, Heidelberg, Germany, 2008. 1-8

[16] Eike S R, Michael K, Saad L K, et al. Performance simulation of a context provisioning middleware based on empirical measurements. *Simulation Modeling Practice and Theory*, 2013, 34(4): 208-220

[17] Sun L K, Wu J G. Schedule and control co-design for networked control systems with bandwidth constraints. *Journal of the Franklin Institute*, 2014, 351(10): 1042-1056

Liu Yu, born in 1982. He received his philosophy doctor from Naval Engineering University in 2011. Then he received his M. S degree from Naval Aeronautical Engineering Institute in 2008. His research focuses on military modeling and simulation, operational system and high performance computing technology.