EasyChair Preprint

№ 621

Research and Comparison of Efficiency between Multicast-based and Traditional SWIM

Ming Qi, Haining Sun and Xiling Luo

November 9, 2018

Research and Comparison of Efficiency between Multicast-based and Traditional SWIM

Ming Qi
Beijing Key Laboratory for Network-based Cooperative Air Traffic Management
School of Electronic and Information Engineering, Beihang University
Beijing, China
qiming@atmb.net.cn

Haining Sun
Beijing Key Laboratory for Network-based Cooperative Air Traffic Management
School of Electronic and Information Engineering, Beihang University
Beijing, China
buaashn@buaa.edu.cn

Xiling Luo
Beijing Key Laboratory for Network-based Cooperative Air Traffic Management
School of Electronic and Information Engineering, Beihang University
Beijing, China
luoxiling@buaa.edu.cn

***Abstract*—Increasingly high requirements are posed on the efficiency of information systems as the air transportation develops rapidly. At present, existing System Wide Information Management (SWIM) has dramatically changed the information architecture of point-to-point data exchange ones while little attention is placed on the efficiency of information exchange. This paper gives a proposition on the efficient information exchange method compatible with SWIM on the basis of UDP multicast. Theoretical analysis and simulation prove that reasonable packet size can take a strict control on the packet loss rate within an acceptable range. Additionally, comparison with the existing SWIM system presents that it also can lower down latency and server load ratio, especially at the time of the large number of users. It is worth to mention that this method has obvious advantages in transmitting including radar data, in terms of the features and requirements of different data.**

Keywords—SWIM, multicast, efficiency, styling

I. INTRODUCTION

EUROCONTROL first proposed the concept of System Wide Information Management (SWIM) to the Federal Aviation Administration in 1997 in order to improve the global interoperability between ATM service providers, enhance the information sharing among various departments, and break the bottleneck of information exchange between heterogeneous services. In 2005, ICAO used SWIM as an international aviation information dissemination system. The United States deployed the NextGen and Europe dispose SESAR in 2007, respectively, and both use SWIM as a framework for information communication and data sharing [1].

A lot of research in recent years of Flight Information Exchange Model (FIXM) and System Wide Information Management (SWIM) core services has theoretical and practical significance. But few studies have focused on the efficiency of information exchange in SWIM system. Kaltenhäuser [2] considers the need to support the standard air traffic control interface as an important quality for a flexible air traffic management simulation environment. Adelantado [3] solved the problem of using standard interfaces by implementing a distributed airport simulation environment for the French Aerospace Research Center ONERA using the Advanced Architecture (HLA). Shifeng and Danxia [4] developed a cost-effective approach to constructing a flight control tower simulation using commercial-of-the-shelf hardware. They also introduced the use of multicast to distribute simulation state encoded in "primitive commands" proprietary to their implementation. But unfortunately they did not quantify the efficiency of multicast transmissions.

At present, the research on multicast has achieved a lot of results, and has become a more mature technology. This paper proposes a multicast-based efficient information exchange method Combined with SWIM message service. The second section will describe the relationship between the typical SWIM service and the proposed method. The third part details the implementation of a multicast-based effective information exchange met hood. The fourth section describes how to apply and gives a simulation example. The simulation results and analysis are given in Section 5. Finally, summarize the full text work.

II. CONCEPTUAL ARCHITECTURE

A. Definitions of SWIM

According to Stal [6], SWIM should be loosely coupled, and its communication interface can have several implementations. According to Wilson et al. [7] SWIM does not rely on a specific implementation of the communication layer, that is SWIM accepts any available communication layer configuration. The SWIM service can implement a partial semantic model. Special implementations can also be applied to SWIM services when their properties can be mapped to the semantic information reference model.

The SWIM service-oriented architecture shall provide the infrastructure, data formats, and protocols to share information between all air traffic management subsystems in a scalable and interoperable way. SESAR defines the semantic layer of SWIM in the AIRM information reference model. In the communications layer it specifies the integration of air traffic management applications by message passing systems.

As of now, three SWIM implementation profiles have been defined by SESAR as shown in Figure 1. The “profile 1” utilizes the Hypertext Transfer Protocol (HTTP) as communication protocol with XML message formats defined in the Web Services Description Language (WSDL). The “profile 2” utilizes the Data Distribution System (DDS) protocol with a binary message format. The “profile 3” utilizes the Advanced Message Queuing Protocol (AMQP) as communication protocol with message formats defined in the web services description language.

B. method for efficient information exchange

This paper proposes a multicast-based SWIM information exchange method (MSIX), which is a simplified implementation of SWIM for research and laboratory testing. We designed a mini implementation with similar information sharing capabilities to SESAR designed SWIM, but with different communication configuration protocols. Our implementation provides SWIM semantic compatibility, so it can be understood as a lab-scale research-based SWIM model.

The multicast-based SWIM information exchange method integrates air traffic information (such as radar data, weather data, etc.) sharing tools through a multicast messaging protocol. The data stored in the actual operating environment is injected into the multicast group for distribution in the simulated state. Each analog terminal can extract the required information from the public information pool after joining the multicast group.

III. METHOD FOR THE MULTICAST-BASED SWIM

The multicast-based SWIM information exchange method (MSIX) is a SWIM implementation for simulation. As stated in the SWIM definition, MSIX only implement a partial semantic model, but the method is compatible with the SWIM semantic layer definition. Therefore, the method can be understood as a SWIM practice with partial information transfer function, which can access operational data for simulation. The use of IP multicast has the following advantages: Multicast is a typical one-to-many message delivery method, which can improve the sharing efficiency of messages. The IP multicast protocol is supported locally, so no additional configuration is required. There is no need to re-set the network when changing networks.

The MSIX method implementation uses a layered architecture based on the OSI (Open System Interconnection) reference model and has been implemented in the Java programming language.

The application layer included in the implementation, implements the application interface between the simulation application and the MSIX. Different types of applications have different interface implementations. But no matter how the interface is implemented, the messages in the application layer conform to the SWIM unified information exchange model, which means that the application layer is compatible with the semantic layer.

The presentation layer contains an XML representation that is platform-independent. In the definition of the SWIM semantic layer, the concrete implementation of the XML message belongs to the physical layer of the exchange model. The exchange model, with FIXM as an example, has a data layer, a logical layer, and a physical layer. The data layer is a

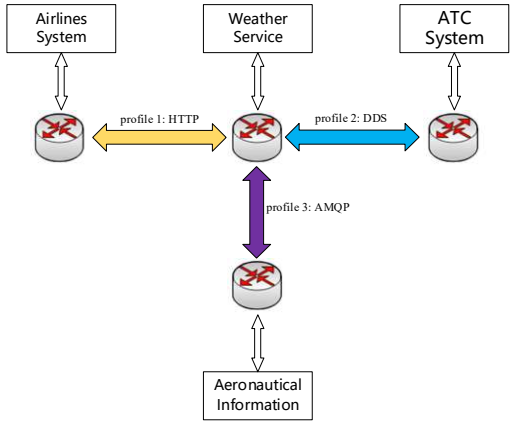

Figure 1: Message broker implementation of multiple communication profiles implemented by the SESAR SWIM infrastructure.

detailed description of all the data needed to exchange models. The logical layer represents the relationship between the data in the data layer. The physical layer is the physical implementation of the logical layer, which means that the data to be transmitted by the application is represented by the physical layer implementation.

The transport layer adopts the UDP (User Datagram Protocol) mode and encapsulates the upper layer XML information. The network layer implements IP multicast. The data link layer and physical layer below the network layer are beyond the scope of this paper, and network connections supporting IP multicast are adopted.

A. Blocking control

The root cause of congestion in the network load provided by the user or the calling system to the network is greater than the network resource capacity and processing capability. The performance is increased by packet delay, the probability of packet loss, and the performance of the upper application system. In the traditional AFTN network, a fixed format message is generally used as an information transmission carrier. This can reduce the amount of data in a single packet and avoid network congestion caused by excessive data usage and processing resources due to excessive data volume. According to the planning of the SWIM network, a more flexible message format will be adopted. At the same time, there is a larger amount of data, so it is necessary to consider ways to deal with network congestion.

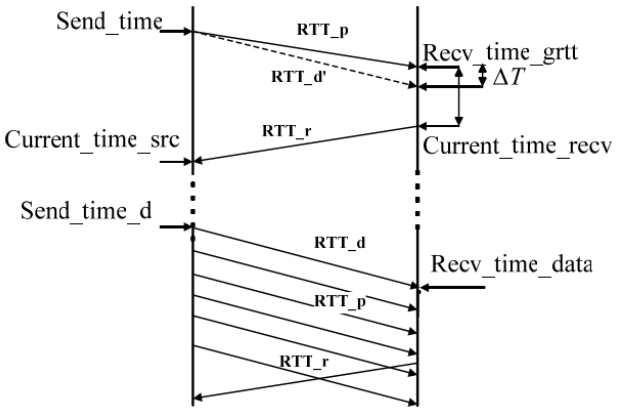

Figure2. calculate the delay and delay dynamics by timestamp.

If the multicast application cannot respond correctly to network congestion, it will bring more serious impact to the network than congestion caused by unicast. The main reason is that the receivers of the multicast data stream are heterogeneous, and each receiver has different processing capabilities. The multicast congestion control protocol is highly targeted, and most of the protocols are proposed for specific problems. For example, radar data does not require reliable transmission (a certain amount of packet loss can be accepted), but the delay is expected to be as short as possible. A multicast congestion control protocol cannot meet all the requirements.

This paper propose an actively adjusted multicast congestion control algorithm. The main design idea is that the receiver adaptively adjusts the receiving rate according to the packet transmission delay, thereby improving the throughput of the multicast session and reducing the delay. As shown in Figure 2, the information sent by the server contains the timestamp Send_time. The receiving end records the time Recv_time_grtt of the collected message and calculates the message delay. In the figure, T represents the dynamic change value that may be generated by the delay.

Messages to be delivered may be cached at the application layer due to network blocking or special implementations of the application. As shown in Figure 5, the application layer provides a message stack to cache messages. When the message is received, it will be pushed onto the stack. The thread that the application uses to process the message reads the message in the first-in, first-out order. Since some messages (such as radar data) are particularly time-sensitive, they should be delivered as quickly as possible, otherwise they will fail. Therefore, the stack will check the timeliness of the message. If the time limit of the data is exceeded, the data will be automatically popped up. The advantage of doing this is to clean up the discarded data on time and avoid wasting resources and time.

B. Reliable multicast

Use UDP at the transport layer. UDP is a connectionless and unreliable datagram protocol. From a resource perspective, UDP socket overhead is small compared to TCP. Because UDP does not need to maintain network connectivity and does not take time to establish a reliable connection, UDP sockets are also faster. Because no reliable connection is established, data may be lost, which is also an unreliable aspect of UDP. Care needs to be done to write a UDP implementation to check for errors and retransmit if necessary. Different messages have different requirements for data integrity. For example, losing a few of the hundreds of data is acceptable for radar information, but fatal problems can occur with flight information. Therefore, a method needs to be proposed to ensure the reliability and integrity of the data.

There are some potential problems in reliable multicast, mainly in three aspects:

Question 1. Distinguish between packet loss or congestion and bottleneck rate issues. If a single node has a high packet loss rate due to interference, it cannot be distinguished whether it is random packet loss or congestion. At this point, due to blocking control, all nodes will be affected, although the network is not crowded at this time.

Question 2. The scalability problem when the multicast size is too large.

Question 3. The dynamic nature of the network. Because the bottleneck receiving end may change constantly. In a dynamic network, the inability to quickly adjust the rate at the network bottleneck will result in more packet loss.

The biggest difference between congestion timeout and loss is that after congestion, the original information can finally be received by the receiver. If a transmission timestamp is added to each data information, and the uniqueness is marked with the sequence information, it is possible for the receiving end to judge whether the packet is congested or lost. By correcting the packet loss estimate p and feeding it back to the source, this can largely distinguish between packet loss and congestion. This solves the problem 1.

In questions 2 and 3, the congestion control scheme is mainly involved. While the blocking control algorithm is enabled, a representative node for the dynamic network needs to be selected for determining the network status. When the number of nodes selected is small, it may not represent the characteristics of the entire large network, resulting in false positives.

IV. APPLICATION AND SIMULATION

This chapter describes the implementation method and simulation test results of the SWIM information exchange method based on multicast.

A. Demonstration Setup

This paper uses Java language to write a multicast-based SWIM information exchange method. The implementation is divided into a server and a client. Different from the traditional civil aviation business system and the SWIM platform in Europe and America, the simulation platform written in this paper provides a multicast interface for data.

The simulation platform server can realize the access and standardization of different flight data, and convert the data format and transmission protocol. As shown in Figure X, based on the public flight data model, the flight data multicast service uses the adapter bidirectional mapping XML Schema model and heterogeneous data to convert the flight data into an XML standard format for encapsulation. The encapsulated standardized data is passed to the data exchange engine to select the corresponding multicast tree. Finally, multicast transmission of standard data is implemented through the message transmission module.

The emulation platform client can join the relevant multicast tree to receive the required flight data as needed. After receiving the standardized data sent by the server, the client will decapsulate the data through the two-way mapping XML Schema model. Finally, the data is presented in the form of a chart. It should be noted here that a series of operations such as receiving a multicast tree message and processing data are performed on different threads. This can avoid packet loss caused by processing data occupying system resources.

B. Demonstration Scenarios

The server will be deployed on a computer with the configuration shown in TABLE I. . In order to test the cross-

platform features of the client, we selected four computers with different systems for testing.

TABLE I. SIMULATION SETUP

Name	Configuration		
	CPU	*Memory*	*System*
Server	Intel i5-3470, dual-core four-thread, 3.2GHz main frequency	16GB	Windows 10 Professional
Client No. 1	Intel i5-3470, dual-core four-thread, 3.2GHz main frequency	16GB	Windows 10 Professional
Client No. 2	Intel i5-3470, dual-core four-thread, 3.2GHz main frequency	4GB	Windows 7 Professional
Client No. 3	Intel i5-3470, dual-core four-thread, 3.2GHz main frequency	4GB	Ubuntu 16.04
Client No. 4	Intel i5-5257U, dual-core four-thread, 2.7GHz main frequency	8GB	macOS High Sierra 10.13.6

In order to test the cross-platform features of the client, we selected three computers with different systems for testing.

In order to test the performance of the system during its actual operation, we wrote the test program in Java language. The test program will make a TCP connection with the server and three clients. After the server sends the flight data to the multicast tree, it will send it to the test program over the TCP connection. After the three clients receive the flight data through the multicast tree, they are also sent to the test program through the TCP connection. The test program compares the flight data to obtain parameters such as packet loss rate, network delay, and throughput.

The test data is derived from real data from the actual operating environment. Operational data for three days from January 4th to January 6th, 2018 was used. The total number of data is 23,957,494.

C. Results

The simulation results include an intuitive presentation and some data. On the one hand, it can be proved that the method of this paper is feasible and can realize the exchange of data. In addition, compared with the traditional system, the method of this paper proves that it has great advantages in exchanging data such as radar and meteorology.

a) The interface that the client runs is shown in Figure 3.

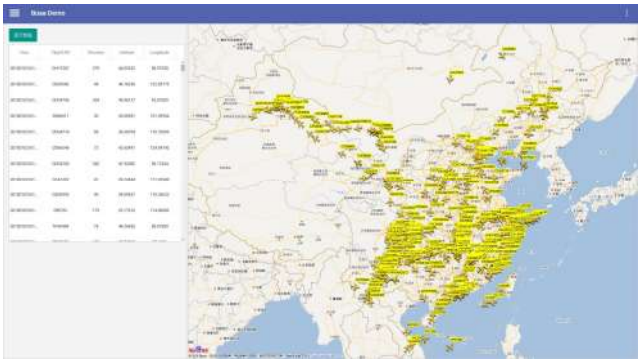

Figure 3: SIWM client. The client selects the page function via the popup button in the upper left corner. Currently selected to enter the radar data display function.

The client can display flight data such as radar data, flight information, airport information, etc. through charts. The data shown in the figure is radar data from the actual operating environment. Store data in the database in a running business system. During simulation, real-time playback is performed according to the timestamp of the data, and the ability of the client to receive and display data is tested.

b) Experiment with the most appropriate packet size.

As shown in Figure 4, the test packet sizes are 500B, 1KB, 8KB, and 10KB, respectively. Test the packet loss rate when sending data with different size packets. Under normal circumstances, when the speed at which the client processes information cannot keep up with the speed of data transmission, large-scale packet loss occurs. It was found that the data packet was the best when it was 8 KB.

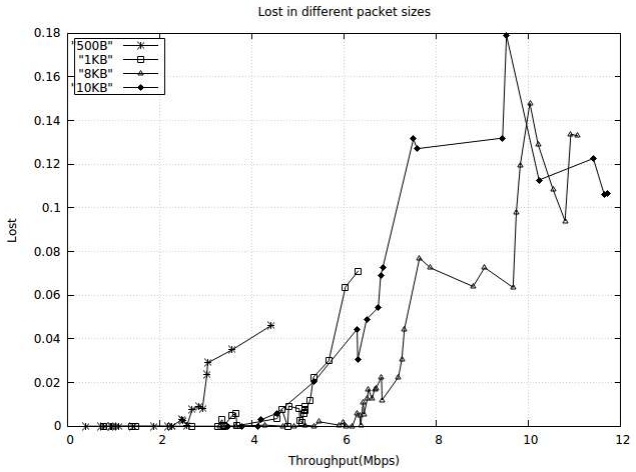

Figure 4: Packet loss rate when sending packets of different sizes. An 8KB packet is an ideal situation.

Data such as radar data and flight information are generally much smaller than 8KB. Radar data is generally around 100B. In the application layer, multiple radar data can be combined into a packet of nearly 8 KB for optimal transmission.

c) Contrast with traditional methods.

The object of comparison in this paper chooses the traditional SWIM information sharing method using Web Services Description Language (WSDL).

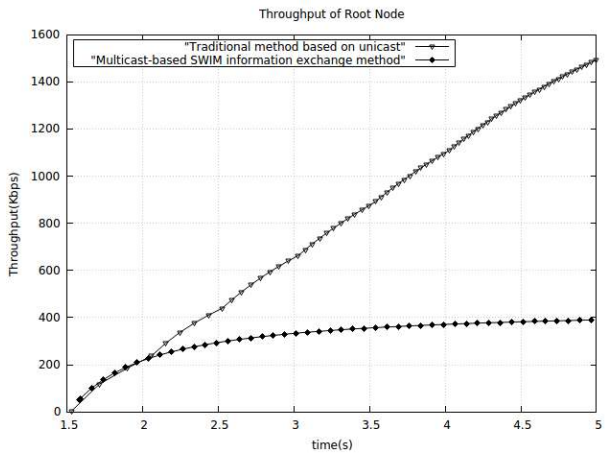

Figure 5: The multicast-based SWIM messaging method can significantly improve the load rate of the source.

As shown in Figure 5, you can see the amount of transmission of the sink node in the case of different methods. The change in the amount of transmission is small in the case of the multicast-based SWIM information transfer method. However, if the conventional method is used, the transmission amount is in a straight rising state. This can also be easily understood. Because every time a node is added in the traditional method, a new data is sent from the source to send.

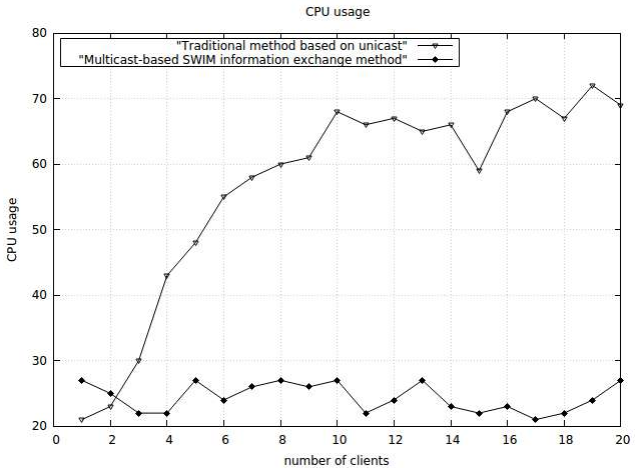

Figure 6: The use of a multicast-based SWIM information exchange method can save computer resources at the source of the message.

As shown in Figure 6, the CPU usage of the source of the message is shown. When using the multicast-based SWIM information transfer method, the CPU usage fluctuates within a small range. However, with traditional methods, CPU usage will rise rapidly and remain at a high level. This is because traditional methods require computers to maintain existing connections. Maintaining a connection will consume a lot of CPU and memory resources when the number of message receiving nodes is very large.

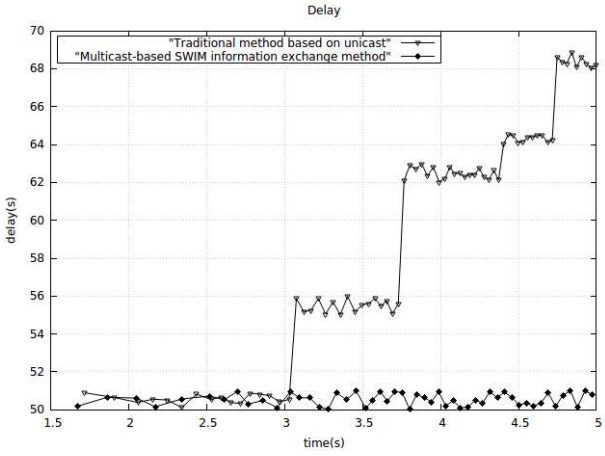

Figure 7: A more stable data delay can be obtained by using the multicast-based SWIM message exchange method.

As shown in Figure 7, the message delay variation of the message receiving node in the network is shown. It can be seen from the figure that with the traditional method, as the number of nodes in the access system increases, the network delay will also increase sharply. There are two problems with excessive network latency. First, it may cause a large number of messages to be retransmitted, resulting in network congestion. Second, it may cause the message to be read and processed by the message consumption node before the end of the life cycle.

V. Conclusion

According to the simulation result b), it can be determined that the performance of a single data packet is about 8 KB or so. In fact, the theoretical analysis is also like this. Due to limited resources, the larger the amount of data sent, the higher the packet loss rate. Therefore, the data can be compressed into as large a packet as possible and transmitted. The maximum packet size in the UDP protocol is 64 KB, and the maximum Transmission Unit (MTU) is 1460B. In fact, in order to improve efficiency (especially in high-speed networks), jumbo frames are defined. When the transmitted data packet is too large, it will directly drop packets due to buffer overflow. When the package is a jumbo frame, it is the best size for both efficiency and packet loss.

According to the simulation result c), comparison between the multicast-based SWIM messaging method and the traditional information exchange method can be obtained. The multicast-based SWIM messaging method is more efficient. And it greatly reduces the pressure on the server. It can realize that the server can serve more clients without increasing the cost. It has a very low latency when there are many connected users. Data that guarantees high timeliness can be delivered to consumers as soon as possible.

References

[1] Di Crescenzo D, Strano A, Trausmuth G. SWIM: A next generation atm information bus-The swim-suit prototype[C]//Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010 14th IEEE International. IEEE, 2010: 41-46.

[2] Kaltenhäuser S. Tower and airport simulation: flexibility as a premise for successful research[J]. Simulation Modelling Practice and Theory, 2003, 11(3-4): 187-196..

[3] Adelantado M. Rapid prototyping of airport advanced operational systems and procedures through distributed simulation[J]. Simulation, 2004, 80(1): 5-20.

[4] Shifeng M, Danxia W. Implementation of a flight control tower simulator using commercial off-the-shelf hardware[J]. Simulation, 2010, 86(2): 127-135.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Stal M. Using architectural patterns and blueprints for service-oriented architecture[J]. IEEE software, 2006, 23(2): 54-61.

[7] S. Wilson, R. Suzic, and S. van der Stricht, “The SESAR ATM information reference model within the new ATM system,” in Proceedings of the Integrated Communications, Navigation and Surveillance Conference (ICNS '14), pp. L3-1–L3-13, IEEE, Herndon, VA, USA, April 2014.