Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

DEVELOPMENT OF A RUNTIME INFRASTRUCTURE FOR LARGE-SCALE DISTRIBUTED SIMULATIONS

Buquan Liu
Yiping Yao
Jing Tao
Huaimin Wang

School of Computer
National University of Defense Technology
Changsha, Hunan 410073, CHINA

ABSTRACT

With the development of distributed modeling and simulation, it is necessary for the RTI to support large-scale applications. However, many RTIs can not support large-scale distributed simulations with more than 100 federates very well nowadays. StarLink+ is an RTI developed according to the IEEE 1516 standard, which can be used for large-scale simulations with thousands of federates. Great innovations are made in StarLink+, such as its architecture and inner implementation technologies. This paper presents the two-level architecture in StarLink+. The unique architecture has the advantages of both central architecture and distributed architecture. To improve the performance much more for large-scale simulations, two important technologies, i.e. multiple threads and data packing, are adopted in StarLink+. In addition, this paper explains the efficient advancing mechanism in time management and discusses the large-scale experiments with thousands of federates in StarLink+.

1 INTRODUCTION

The High Level Architecture (HLA) has been developed to provide a general framework for distributed modeling and simulation. The objective of HLA is to provide a common architecture applicable across a variety of simulations for interoperability and reuse. The Runtime Infrastructure (RTI) is the software that provides common interface services during a HLA federation execution for synchronization and data exchange. In September 2000, HLA was accepted as the IEEE 1516 standard.

Continuously supported by multiple National High-Tech Research and Development Programs, we have implemented an RTI named StarLink which supports the IEEE 1516 standard and which is based on CORBA (Liu, Wang and Yao 2004). StarLink is an RTI with central architecture, and it can be well applied from small to moderate applications. On the basis of StarLink, we have also successfully developed StarLink+ (Liu, Wang, and Yao 2005), which is a hierarchical RTI with two-level servers to meet the requirement of large-scale distributed simulations.

Presently, it is not very efficient for an RTI to support more than 100 federates, and the HLA time management is also a factor confining the scale of simulations. The HLA time management provides varieties of time synchronization mechanisms and can meet all kinds of distributed simulations theoretically. However, few large-scale applications with more than 100 federates in the HLA time management have been implemented efficiently in reality. The time management in RTI is meeting tremendous challenges for large-scale and efficient simulations. We have, however, successfully made a few large-scale experiments with thousands of federates which are time regulating and time constrained by calling the enableTimeRegulation and enableTimeConstrained services in StarLink+.

StarLink+ adopts a unique architecture and has particular characteristics, which this paper discusses in detail. In the Section 2, the two-level hierarchical architecture is presented and compared with other architectures. The unique architecture can greatly decrease the number of messages between RTI servers, and suitable for large-scale applications. In Section 3, two important technologies are described: multiple threads and data packing. The multiple threads approach can improve the RTI's parallel performance and the data packing method can resolve the bottleneck problem caused by large numbers of messages in StarLink+. In Section 4, the basic execution process in StarLink+ is introduced. From this section, we know that StarLink+ can greatly decrease network data and increase system efficiency. Section 5, explains the efficient implementation mechanism of time management in StarLink+, which can support large-scale applications. Finally, the large-scale experiments are also demonstrated.

2 TWO-LEVEL ARCHITECTURE

To support large-scale simulations, StarLink+ adopts the two-level architecture shown in Figure 1. The whole system is composed of a Central RTI server (CRTI), multiple Local RTI servers (LRTIs) and federates. The CRTI manages multiple LRTIs and each LRTI manages multiple federates. In addition, each federate belongs to one LRTI and each LRTI is also a federate of all other LRTIs.

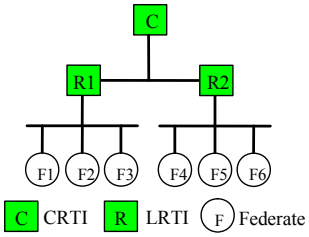

Figure 1: Architecture of StarLink+

2.1 Characteristics

The hierarchical architecture in StarLink+ has many unique characteristics.

1. The two-level other than multi-level architecture is adopted. In fact, we think that the multi-level architecture shown in Figure 2 is not efficient. In such an architecture, the communication between federates can only travel via ancestor nodes. For example, the communication between two federates *F1* and *F4* should at least cross multiple RTI servers which are *R3*, *R1*, *R2* and *R6*. Thus, the time delay between *F1* and *F4* is great. The architecture shown in Figure 2 also makes it difficult for time management to determine the time stamp order (TSO) message with least logical time when federates call the nextMessageRequest, nextMessageRequestAvailable and flushQueueRequest services to advance. In addition, the bottleneck phenomenon for transmitting a great deal of data shall not be avoided. The closer a node is to the central RTI server, the more disastrous the phenomenon is. However, Figure 2 is equivalent to Figure 1 if the CRTI and all LRTIs are allowed to communicate with each other. Therefore, *F1* shall communicate with *F4* via *R3* and *R6*. This shall greatly decrease the time delay between them.
2. There is no traditional Local RTI Component (LRC). In DMSO RTI1.3-NG (DMSO 2000), federates communicate with each other via their LRCs. But in StarLink+, a federate can only communicate with other federates via LRTIs. A federate is separated from its LRTI and their communication is automatically accomplished by the underlying CORBA middleware.
3. It only supports single federation. To simplify the implementation of StarLink+, we only support a single federation. Multiple federations can be executed by starting multiple CRTIs. Of course, we can also adopt the rtiexec process similar as RTI1.3-NG to start multiple CRTIs.
4. Compatible with a central RTI. StarLink+ is developed on the basis of StarLink with central architecture shown in Figure 3. StarLink+ can be degraded into StarLink by configuring an initialization file.
5. Compatible with a distributed RTI. Figure 4 shows a distributed architecture which is used by many RTIs such as RTI1.3-NG, MÄK RTI (MÄK Technologies 2006) and pRTI (Pitch Technologies 2006). However, Figure 5 is similar to Figure 4 and StarLink+ is degraded into a distributed RTI if each LRTI only manages a single federate. Then the CRTI is approximately equal to the CRC and a LRTI is approximately equal to a LRC. But there still exist some differences between them. In Figure 5, a federate and its LRTI are two independent processes; but in Figure 4, a LRC is bound to a federate in the form of a library.
6. A LRTI joins to another LRTI as a federate. When a federate joins to a LRTI, we say that this federate belongs to the LRTI. One federate can only join to one LRTI. In StarLink+, a LRTI can only know CRTI, other LRTIs and its federates. A LRTI doesn't see any federates that belong to other LRTIs. Of course, a federate can only see its LRTI, and it doesn't know CRTI and other LRTIs.
7. A large amount of messages are decreased. In Figure 1, the Local RTI server *R1* only sends a message to *R2* once rather than three times if the message sent by federate *F1* is subscribed by three federates *F4*, *F5* and *F6*.
8. Time management is suitable for large-scale simulations. In StarLink+, a federate has no its LRC. Only all LRTIs make decisions in time management rather than all federates in a federation.

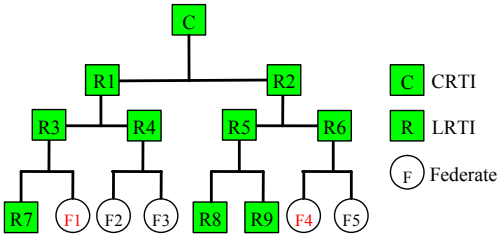

Figure 2: Architecture with Multiple Levels

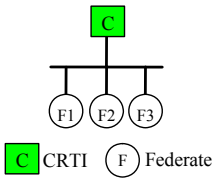

Figure 3: Architecture of StarLink

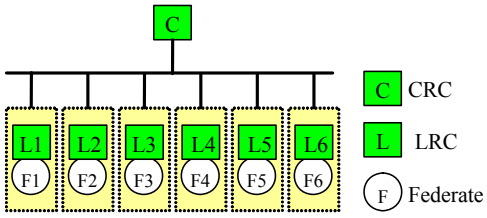

Figure 4: Architecture of Distributed RTI

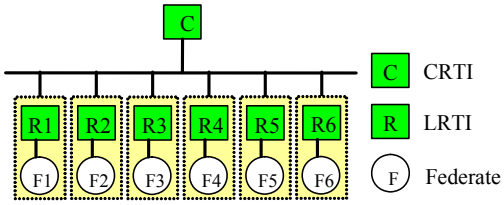

Figure 5: StarLink+ Used as a Distributed RTI

2.2 Typical Cases

StarLink+ has outstanding advantages. In StarLink+, the central RTI server is similar to the naming server in CORBA. All LRTIs can interconnect with each other by CRTI. The CRTI can only be used for synchronization point as well as save-and-restore in the HLA federation management. CRTI is useless for any other services. We ignore CRTI in the rest of this paper. We define the following symbols:

Definition 1 *T(FR) is the network time that a federate sends a message to its LRTI, or a LRTI sends a callback message to its federate.*

Definition 2 *T(RR) is the network time that a LRTI sends a message to another LRTI.*

Definition 3 *T(FL) is the network time that a federate sends a message to its LRC, or a LRC sends a callback message to its federate.* T(FL) is approximately equal to zero because a LRC is always bound to its federate.

Definition 4 *T(LL) is the network time that a LRC sends a message to another LRC.*

Definition 5 *T(FF) is the network time that a message sent by a federate arrives to another federate.*

In Figure 1, we have

$$T(F1F6) = T(F1R1) + T(R1R2) + T(F6R2). \quad (1)$$

In Figure 4, we also have

$$T(F1F6) = T(F1L1) + T(L1L6) + T(F6L6) \approx T(L1L6). \quad (2)$$

From the above formulas, we know the following conclusions shall be true if we only consider the delay time of a message and ignore the time that an RTI maintains global data consistency.

1. Assume that $T(FR) \approx T(RR) \approx T(LL)$, (1) should be three times of (2). Hence the performance of StarLink+ is worse than the distributed RTI in Figure 4; we should avoid this configuration when LRTIs in StarLink+ are deployed.
2. Suppose that $T(FR) << T(RR)$ and $T(RR) \approx T(LL)$. From (1), we have

$$T(F1F6) \approx T(R1R2). \quad (3)$$

From (3), we know that a message's delay time in StarLink+ is nearly equal to that in a distributed RTI.

On the other hand, assume that only the time about global data consistency is considered, StarLink+ is superior to a distributed RTI because only LRTIs are useful for decision-making in StarLink+, but all LRCs should take part in decision in a distributed RTI.

We conclude that the two deployments shown in Figure 6 are preponderant for StarLink+:

1. LRTIs connect with each other via Wide Area Network (WAN), while each LRTI and all its federates connect via Local Area Network (LAN) or Ethernet.
2. LRTIs connect with each other via Local Area Network or Ethernet, while each LRTI and all its federates run on the same machine. A cluster system is such an example.

The above deployments sound most reasonable for the federation with thousands of federates. On the other hand, we shall not deploy those federates into thousands of machines connected either by WAN, LAN or Ethernet.

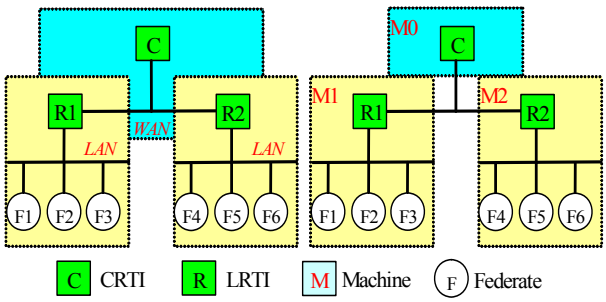

Figure 6: Two Typical Cases of StarLink+

3 KEY TECHNOLOGIES

Besides the unique architecture, StarLink+ has also adopted two important technologies to support large-scale simulations, i.e. the multiple threads approach and the data packing method. The former increases the parallel performance in StarLink+ and the tick service in HLA 1.3 does not appear in a federate of StarLink+, while the latter resolves the bottleneck of LRTIs for transferring a large amount of data.

3.1 Multiple Threads

The multiple threads approach can bring the following advantages.

1. Avoid deadlock. Single thread adopted in StarLink+ may lead to the deadlock. The basic client/server mode is used in CORBA. In StarLink+, either a LRTI or a federate is not only a server but also a client. When a federate calls its LRTI, the federate is in state of waiting before its LRTI returns. On the other hand, when a LRTI calls back its federate, the LRTI shall wait for what the federate returns. Thus, one or more federates and their LRTIs may wait for each other and the deadlock occurs. Another advantage brought by the multiple threads method is that the tick service defined in HLA 1.3 shall not be implemented. For one service call, a federate shall waste abundant of CPU time because it would call the tick service for many times in using those RTIs.
2. Substitute multicast. The multicast technology is used in many RTIs. As many HLA services are reliable including ownership and time management services, but multicast is an unreliable mechanism based on the UDP protocol. It shall make the development of an RTI more difficult and complicated when an unreliable mechanism is used to resolve reliable problems.
3. Improve parallelism. Different data can simultaneously be sent and received by multiple LRTIs and federates. In StarLink+, when a federate joins to a LRTI, the LRTI creates two threads for the joining federate. One thread is responsible for receiving data and the other is responsible for sending data. This is true for a LRTI to join to the CRTI and to another LRTI.

3.2 Data Packing

Multiple threads shall exist in a LRTI, but the scheduling time of a thread assigned by the operating system is limited. If a callback thread can not send data to a federate in time, numerous messages will pile up in the LRTI. The memory may be not enough and a LRTI shall collapse if excessive messages store in the LRTI. To resolve the bottleneck phenomenon, the data-packing method is brought forth. Data-packing means that all messages storing in a thread are bound into a single packet and the thread only sends one packet to a receiver when the thread is invoked by the operating system. According to our experimental results, the approach can greatly improve the efficiency of StarLink+.

Suppose that the size of a message is 100 bytes. Figure 7 shows the delay time for message transmitting between two LRTIs in different machines in 100M network. In the experiment, the delay time is 0.31ms for one message with 100 bytes sent from one LRTI to another LRTI. From Figure 7, we know that sending one packet with multiple messages is more efficient than sending these messages in sequence. For example, sending 100 messages will take 31.60ms, but one packet covering these 100 messages will only take 1.440ms. The efficiency ratio is *31.60/1.440=21.93*.

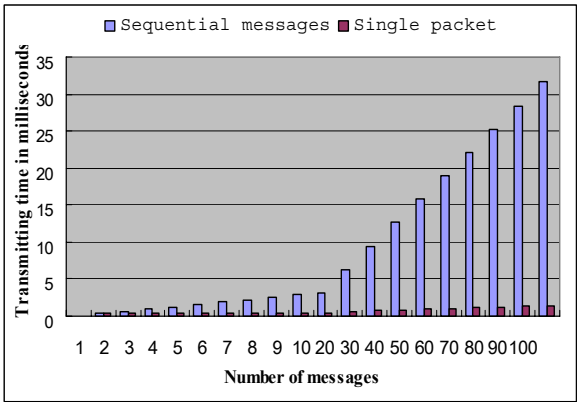

Figure 7: Message Transmitting Time between Two LRTIs

4 DATA EXCHANGE PROCEDURE

One of the basic concepts in the HLA is the publication and the subscription mechanisms for data transmissions. To better comprehend the rationale of StarLink+, we present the process of data exchange for object class attributes. During the process of developing StarLink+, a number of services extended from normal HLA services are adopted in StarLink+ for the internal communication between CRTI and LRTI. These services are user transparent.

4.1 Create Federation Execution

The central RTI server is always started firstly, and all Local RTI servers started afterwards. During a federation is created, the CRTI is similar to the naming server in CORBA. When a LRTI starts, the LRTI firstly joins to the CRTI by calling the extended HLA service joinCentralFederationExecution, then the CRTI calls the lRTIjoinAnotherLRTI service to make the LRTI join to all other LRTIs which have joined the CRTI. Finally, the CRTI also

calls the lRTIjoinAnotherLRTI service to make all other LRTIs which have joined CRTI join to the LRTI. Thus, CRTI and all LRTIs can communicate with each other.

4.2 Publish and Subscribe

When a federate publishes or subscribes object class attributes, its LRTI performs two things.

1. Recording the publication and subscription information from the federate.
2. If there exists at least one attribute which was not published or subscribed by local federates before, the LRTI forwards the publishing or subscribing service to any other LRTIs, and these LRTIs then record the publication and subscription information from the LRTI.

Suppose that all three federates *F1*, *F2* and *F3* publish the same attributes of an object class in Figure 1, the LRTI *R1* only informs *R2* once and *R2* records the publication.

4.3 Register Object Instance

When a federate calls the registerObjectInstance service to register an object instance, its LRTI does the following things.

1. If there exist any local federates which have subscribed any attributes of corresponding object class, the LRTI informs these federates to discover an object instance.
2. The LRTI calls the extended lRTIregisterObjectInstance service to inform those remote LRTIs which have subscribed any attributes of the object class, and these LRTIs inform their federates which have subscribed any attributes of the object class to discover an object instance. The object instance is successfully registered in remote LRTIs.

From the above procedure, we know that multiple federates from different LRTIs can register object instances simultaneously.

4.4 Update Attribute Values

This is similar to the registration process of an object instance. When a federate calls the updateAttributeValues service, its LRTI does the following steps.

1. If there exist any federates which have subscribed corresponding information, the LRTI calls the reflectAttributeValues service to inform these federates to receive information.
2. If there exist any remote LRTIs which have subscribed corresponding information, the LRTI forwards the updateAttributeValues service to these LRTIs, and these remote LRTIs call the reflectAttributeValues service to inform their subscription federates to receive information.

We emphasize that even if multiple remote federates belonging to a LRTI *R2* have subscribed corresponding information, a LRTI *R1* should only send the updateAttributeValues service once to the remote LRTI *R2*. Therefore, StarLink+ is able to decrease the messages among LRTIs greatly.

5 TIME MANAGEMENT

Time management in HLA is virtually the time synchronization for federates. The hierarchical architecture in StarLink+ improves the decision-making efficiency in time management. Here the decision-making means the computation of Greatest Available Logical Time (GALT). GALT is the term in the IEEE 1516 standard which is also called Lower Bound Time Stamp (LBTS) in HLA 1.3. In a distributed RTI, all LRCs shall participate in the computation of GALT. While in StarLink+, only all LRTIs participate in the computation. As an example of a federation with 1,024 federates, we deploy 64 LRTIs and each LRTI sees after 16 federates. In such a federation, only 64 LRTIs rather than 1,024 federates participate in the computation. But for RTI1.3-NG with the same federation, 1,024 LRCs shall take part in the computation.

The computation of GALT is a complicated problem as discussed in the papers (Fujimoto 1996; Kuhl, Weatherly and Dahmann 1999; Riley, Fujimoto and Ammar 1999). The detailed introduction about the algorithm is out scope of this paper. We only simply discuss an example in Figure 8 about the timeAdvanceRequest (TAR) service, and other time advance services have similar results because of our GALT algorithm. The example includes two LRTIs, i.e. *R1* and *R2*. In the federation, each LRTI includes 16 federates. CRTI doesn't appear in the figure because it is useless for GALT computing. Suppose that the initial logical time of each federate is *T* and lookahead (Fujimoto 1988; Fujimoto 2000) is 0.1. For any federate *Fi*, $S(Fi)=T+0.1$. The symbol *S* is the value of one federate or one LRTI server described in our GALT algorithm (Liu, Wang and Yao 2005). Now we have $S(R1) = S(R2) = T+0.1$. The example is nearly same as the model that introduced in the next section.

The advance procedure of the federation is described as follows.

1. When *F1* calls the TAR service to advance to $T+1$, the LRTI *R1* computes *F1*'s GALT.

$$GALT(F1) = \min\{S(F2), S(F3), ..., S(F16), S(R2)\}$$
$$= T + 0.1 < T + 1.$$

2. *F1* shall be in time advancing state and wait for *R1* to grant its request. Similarly, when *F2*, *F3*, … , *F15* call the TAR service to advance to *T+1*, they shall also be in waiting state. Now we have

$$S(F1) = S(F2) = ... = S(F15) = (T + 1) + 0.1 = T + 1.1.$$

When *F16* requests to advance to *T+1*, *F16* shall be in waiting state. But now *S(F16)* and *S(R1)* are changed into *T+1.1*. Thus *R1* must send *S(R1)* to the LRTI *R2* to modify the value.

3. When *F17*, *F18*, … , *F31* call the TAR service to advance to *T+1*, they shall be in waiting state as their GALTs are *T+0.1*. Now we have

$$S(F17) = S(F18) = ... = S(F31) = (T + 1) + 0.1 = T + 1.1.$$

When *F32* wants to advance to *T+1*, *S(F32)* and *S(R2)* are changed into *T+1.1*. Thus *R2* sends *S(R2)* to *R1* to inform its new value. Now each federate's GALT is changed into *T+1.1* which is larger than the federate's request time *T+1*. Therefore, *R1* and *R2* call the timeAdvanceGrant (TAG) service to grant their federates to logical time *T+1* respectively. Note that each federate is informed by a thread in its LRTI. Now all federates have advanced to *T+1*. If the 32 federates request to advance simultaneously step by step, the federation can advance simultaneously step by step.

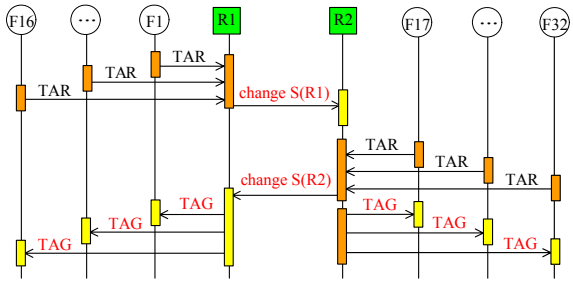

Figure 8: TAR Parallel Advancing Mechanism

In Figure 8, only two calls occur between two LRTIs when 32 federates advance one step. If each LRTI and its 16 federates run in the same machine, the HLA time management can be implemented efficiently because there is rather a little communication about GALT computation among different machines.

6 EXPERIMENTS

To compare the capability of national RTIs to integrate countrywide military simulations which may consist of hundreds of programs in the future in China, a famous modeling and simulation expert brought forward a well-known population problem for voting. Suppose that there are 20,000,000 populations in a kingdom. Each person can select one from three choices to decide whether or not to support the king's new policy. The three choices are "yes", "no" and "neutral". But each person is affected by all other persons and this means that each person will send a message to the others. The voting example can be used to analyze and make political decisions on some sensitive and hot spots in the world. In this system, the activity of an individual is useless and the aggregate activity is what we care about. The example is a typical complex system, which features a large number of interacting components (agents, processes, etc.) whose aggregate activity is nonlinear (not derivable from the summations of the activity of individual components) and typically exhibits hierarchical self-organization (Rocha 2003). The research area can be applied to social networks, gene and protein networks, knowledge networks, infrastructure networks, etc.

The population voting question is also an example closely relating to network communication. Highly frequent network communication is one important characteristic for many simulation applications. Network bandwidth has great effects on those simulations and their underlying runtime infrastructure. For example, many experimental results are given in Clark, Capella, Bailey and Steinman (2002). The HPC-RTI based on SPEEDES is designed for high performance computers, and it is not strange for large amount of objects over the RTI to take more than 100 seconds to advance one step for some experiments.

In the voting example, the whole federation shall be composed of 2,000 federates if 10,000 persons are modeled as a federate. The computational process for each federate is complicated and this is out scope of this paper. Besides these voting federates, the simulation consists of three federates, i.e. a control federate, a situation display federate, and an environment federate. The control federate is used to control the start and end of the simulation. The situation display federate supports various results in the forms of figures, curves, and cylinders. The environment federate issues some particular events subscribed by all voting federates. An event is a stimulus which has effects on the result of voting. For instance, more people shall no longer support their king if the news about the king's scandal is disseminated. In the federation with 2,000 federates, each federate is modeled to use the TAR service to advance logical time, then call the updateAttributeValues service to send a reliable TSO message subscribed by all other federates. The logical time interval for each step is 40 and each federate's lookahead is 1 (1 is far less than 40). The time-

stamp of each TSO message is the logical time of next step. However, as far as the voting example and the TAR service, the exact value of interval logical time is not important and it can be equal to any value so far as the lookahead is less than it. Virtually, the lookahead can also be set to zero if we use the timeAdvanceRequestAvailable (TARA) service. Thus, a federate can not advance beyond all other federates, and they must simultaneously advance to the same logical time together step by step.

In the example, suppose that the variable *n* denotes the number of federates, and also suppose that each federate sends a TSO message to all other federates. If the size of one message is 100 bytes, the size of total TSO messages for one step is *(n-1) * n* 100 * 8* bits. Over the 1,000Mbps network, the theoretically minimal execution time for n federates to advance one step is

$$((n-1)*n*100*8)/(1024*1024*1024)$$

seconds. For 1,024 federates, the minimal time is 0.78 seconds. Figure 9 and Figure 10 are the experimental results of StarLink+ for one step, Figure 9 is the average execution time for 128 to 1,024 federates, and Figure 10 is for 1,024 to 5,120 federates. For 1,024 federates, the average execution time is 1.798 seconds. These experiments were made in a cluster system with 128 CPUs, which was composed of 64 independent nodes connected by two 1,000Mbps switches. In fact, the experiments were equivalent to perform in 64 independent personal computers connected by 1,000Mbps Ethernet. Here is the configuration of each node.

- CPU: Intel(R) Xeon(TM) CPU 3.40GHz (Dual Core).
- Memory: 4GB.
- Network: 1,000Mbps.
- Operating system: Linux server2 2.4.21-32.EL.
- Programming language: GNU C++.

In our experiments with Figure 9 and Figure 10, we did not start the graphical situation display federate and the environment federate, which should bring on inaccurate results. The whole process of starting the voting simulation was as follows.

1. Start the central RTI server in the first node of the cluster system.
2. Start 64 local RTI servers and each local RTI server ran in one node.
3. Start the control federate in the first node of the cluster system.
4. Start the voting federates in all nodes. All federates were averagely deployed into each node. As far as the simulation of 1,024 federates, each node consisted of 16 (*1024/64*) federates. During the federation execution, only the first joined federate outputted and recorded its execution time for each step in each node.
5. The control federate issued the start and end commands of federation execution.

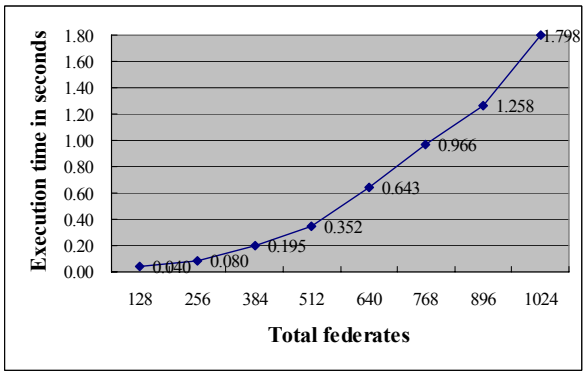

Figure 9: Large-Scale Experimental Results

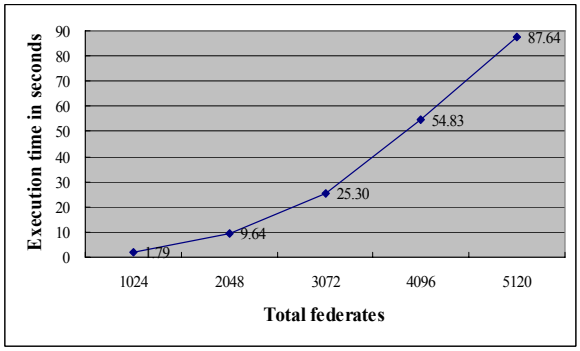

Figure 10: Ultra Large-Scale Experimental Results

7 CONCLUSION

StarLink+ is a unique RTI with two-level hierarchical architecture, which possesses the advantages of the central architecture and distributed architecture. To promote the ability for large-scale simulations, two key technologies are applied into StarLink+. The multiple threads approach is used to increase the parallel performance and the data packing method is used to improve the efficiency of message transferring in StarLink+. This paper also explains the efficient time management in StarLink+. The HLA time management is virtually the synchronization mechanism for messages which makes large-scale distributed simulations more challengeable. In StarLink+, a federate has no LRC and it communicates with its LRTI server via underlying CORBA middleware; accordingly, time synchronization is efficiently coordinated only by all Local RTI servers. Finally, large-scale experiments with thousands of federates in StarLink+ are introduced.

ACKNOWLEDGMENTS

This work was funded by *the National Grand Fundamental Research 973 Program of China* under grant number 2005CB321804, and *the National Natural Science Foundation of China* under the grant number 60373024.

REFERENCES

DMSO. 2000. RTI 1.3-next generation programmer's guide version 6. Available via <http://hla.dmso.mil> [accessed April 8, 2002].

Fujimoto, R. M. 1988. Lookahead in parallel discrete event simulation. In *Proceedings of the 1988 International Conference on Parallel Processing 3*: 34-41.

Fujimoto, R. M. 1996. HLA time management: design document. College of Computing Georgia Institute of Technology Atlanta. Available via <http://www.cc.gatech.edu/computing/pads/papers.html> [accessed March 18, 2006].

Fujimoto, R. M. 2000. *Parallel and distributed simulation systems*. New York: John Wiley & Sons.

Clark, J., S. Capella, C. Bailey, and J. Steinman. 2002. The development of an HLA compliant high performance computing run time infrastructure. *Proceedings of the 2002 Spring Simulation Interoperability Workshop*, 02S-SIW-016.

Kuhl, F., R. Weatherly, and J. Dahmann. 1999. *Creating computer simulation systems: an introduction to the high level architecture*. New Jersey: Prentice Hall PTR.

MÄK Technologies. 2006. Available via <http://www.mak.com/rti.htm> [accessed March 18, 2006].

Liu, B. Q., H. M. Wang, and Y. P. Yao. 2004. Key techniques of a hierarchical simulation runtime infrastructure-StarLink. *Journal of Software* 14 (1): 9-16. Available via <http://www.jos.org.cn/paper/detail.asp?id=1765> [accessed June 10, 2006].

Liu, B., H. Wang, and Y. Yao. 2005. Data consistency in a large-scale runtime infrastructure. *Proceedings of the 2005 Winter Simulation Conference*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available via <http://www.informs-sim.org/wsc05papers/221.pdf> [accessed March 18, 2006].

Pitch Technologies. 2006. Available via <http://www.pitch.se/prti> [accessed March 18, 2006].

Riley, G. F., R. Fujimoto, and M. H. Ammar. 1999. Network aware time management and event distribution, College of Computing Georgia Institute of Technology Atlanta [online]. Available via <http://www.cc.gatech.edu/computing/pads/papers.html> [accessed March 18, 2006].

Rocha, L. 2003. Complex systems modeling: using metaphors from nature in simulation and scientific models. Los Alamos National Laboratory. Available via <http://informatics.indiana.edu/rocha/complex/csm.html> [accessed June 8, 2006].

AUTHOR BIOGRAPHIES

BUQUAN LIU received his B.S. degree in computer science from Nanjing University in 1991. His M.S. and Ph.D. degrees were received in the School of Computer from National University of Defense Technology (NUDT) in 1998 and 2004 respectively. He has achieved 2 Provincial Science and Technology Advance Awards and 1 patent of *the design of hierarchical RTI servers based on interoperability protocol*. Now he is an associate professor of the school and his interests are distributed simulation and high performance computing. His e-mail address is <bqliu@nudt.edu.cn>.

YIPING YAO is a professor of School of Computer in National University of Defense Technology. In this school, he received his M.S. and Ph.D. degrees in 1987 and 2004 respectively. He received his B.S. degree in computer science from Huazhong University of Science and Technology in 1985. At present, he has achieved 2 second-class National Science and Technology Advance Awards and 8 Provincial Science and Technology Advance Awards. His research areas are distributed simulation and virtual reality. His e-mail address is <ypyao@nudt.edu.cn>.

JING TAO is an associate professor in the School of Computer at the National University of Defense Technology. She received her B.S. and M.S. degrees in the school in 1992 and 1998 respectively. She has won 3 Provincial Science and Technology Advance Awards. Her interests are distributed simulation and high performance computing. Her e-mail address is <ellen5702@tom.com>.

HUAIMIN WANG is a professor in the School of Computer at the National University of Defense Technology. He received his Ph.D. degree in computer science in 1992. He is a member of the Editorial Board of *Chinese Journal of Computers* and *Journal of Computer Science and Technology*. Dr. Wang has served as a member of the Expert Committee for *Computer Software and Hardware of the National High Technology Research and Development Program of China* (863 Program). In 2003, he was awarded one 2nd class National Science and Technology Advance Award. His research focuses on distributed object, agent technology, grid computing and network security. His e-mail address is <whm_w@163.com>.