Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

AN EFFICIENT ALGORITHM IN THE HLA TIME MANAGEMENT

Buquan Liu
Yiping Yao
Huaimin Wang

School of Computer
47 Yanwachi Street, National University of Defense Technology
Changsha, Hunan 410073, P. R. CHINA

ABSTRACT

The HLA time management is an important factor that limits the scalability of distributed simulations. An efficient algorithm of Greatest Available Logical Time (GALT) is thus much critical for the time management in an RTI to support large-scale simulations. The concept of GALT in IEEE 1516 was also called Lower Bound Time Stamp (LBTS) in HLA 1.3. The computation of GALT in the HLA time management is different from that of LBTS in traditional Parallel Discrete Event Simulation (PDES). In this paper, an algorithm about GALT is presented and its correctness is proved. Its efficiency is also explained by applying it to RTI1.3-NG. In fact, the algorithm has been implemented in our RTI to support thousands of federates in our cluster systems. In addition, a real-world example is introduced to explain the correctness of the algorithm proving, and the reason of our RTI supporting large-scale simulations.

1 INTRODUCTION

The HLA time management is an important factor for limiting the scales of distributed applications. The overhead for time synchronization is much higher for large-scale simulations. One important problem of using the HLA time management for large-scale simulations lies in the Greatest Available Logical Time (GALT) algorithm of RTI. GALT is the maximal logical time that a federate may advance securely, which is defined in IEEE 1516 and it is also called Lower Bound Time Stamp (LBTS) in HLA 1.3. Though many LBTS algorithms have been proposed in the Parallel Discrete Event Simulation (PDES) area, a GALT algorithm is virtually different from those LBTS algorithms in PDES. For example, the important concepts of Local Virtual Time (LVT) and Global Virtual Time (GVT) (Jefferson 1985) in PDES do not appear in the HLA time management. In HLA, a federate must schedule an event and advance its logical time under the control of RTI, while a process usually does without such an underlying infrastructure in PDES. For the optimistic mechanism, a process may roll back its virtual time even to its start in PDES, while a federate cannot roll back its logical time in HLA. A federate can only call the timeAdvanceRequest (TAR), timeAdvanceRequestAvailable (TARA), nextMessageRequest (NMR), nextMessageRequestAvailable (NMRA), and flushQueueRequest (FQR) services to advance its logical time. However, a federate should receive the 'RTI::LogicalTimeAlreadyPassed' exception if it tries to roll back its logical time according to the HLA standard.

Computing GALT is a complicated problem as stated in Fujimoto (1998). In the next section, some factors of GALT computation according to the IEEE 1516 standard are discussed. A GALT algorithm shall consider the TAR, TARA, NMR, NMRA, and FQR services. The earliest algorithms about the TAR service might be those presented in Fujimoto (1996) and Carothers et al. (1997), which were a little different from subsequent HLA standards (HLA 1.3 and IEEE 1516). A more detailed algorithm was discussed in a book by Kuhl, Weatherly, and Dahmann (1999). But the book was written for HLA beginners rather than RTI developers, and its algorithm was easily understood. The algorithm mentioned in the book was a recursive one and its lookahead was supposed to be unchangeable. In the third section, an efficient GALT algorithm is brought forward. In fact, our algorithm originally comes from the book and we have made much improvement on it. Now the algorithm can be practically used for RTI developers to implement the HLA time management. One important characteristic is that there is no recursion in our algorithm any more, and consequently it does not need to consider any deadlock caused by recursion. In addition, the lookahead in the algorithm may be changeable during the whole federation. In fact, the algorithm has been successfully applied to StarLink+ (Liu, Yao, and Wang 2005; Liu et al. 2006a). Nowadays, an RTI cannot efficiently support more than 100 federates in a federation very well. Millennium Challenge 2002 (MC02) was regarded as the largest, most

complex military experiment ever conducted, and the simulation included around 90 federates (Ceranowicz et al. 2002). However, StarLink+ is an RTI which can support thousands of federates in our self-made supercomputers (NUDT 2007). Thousands of federates can join to a federation within one minute in our experiments. In the fourth section, we apply the GALT algorithm to RTI1.3-NG, and discuss the efficiency of the algorithm. In the fifth section, the correctness of the new algorithm is proved. Finally, we explain the proving process of the GALT algorithm by using the example of pyramidal mountain climbing, and briefly discuss how the algorithm in StarLink+ can be used for simulations with thousands of federates.

2 COMPUTING FACTORS OF GALT

In this section, three important factors about GALT computing are discussed, which are advancing services, network messages, and underlying communication mechanisms.

2.1 Advancing Services

Five time advancing services are provided according to the IEEE 1516 standard, which are TAR, TARA, NMR, NMRA, and FQR. Typically, the TAR and TARA services are used in time stepped simulations and the NMR and NMRA services are used in discrete event simulations. They are also called conservative advancing services. While the FQR service is used in optimistic simulations.

2.1.1 TAR/TARA

For the TAR and TARA services, only logical time and lookahead need to be considered when counting GALT. That is to say, it is not necessary to consider a certain Time Stamp Order (TSO) message in one federate's TSO queue if the federate calls the two services to advance. However, this is definitely different from the NMR and NMRA services.

2.1.2 NMR/NMRA

For convenience, a new concept called LETS is introduced in the paper which means Least Existent Time Stamp.

Definition 1 *A federate's LETS is the smallest time stamp of all TSO messages in its TSO message queue.*

LETS is different from the Least Incoming Time Stamp (LITS) that is defined in IEEE 1516. LITS is the smallest time stamp that the joined federate could (but not necessarily would) receive in the future in a TSO message queue. A federate's LITS can be set as the smaller of its GALT and LETS. LITS shall result in the recursion of GALT computing. The differences between both concepts of LETS and LITS show the main differences between our approach and that is currently used by the HLA specification.

For a federate using the NMR or NMRA service to advance to the specified logical time *T*, it may advance to the smaller of *T* and its LITS according to HLA standard. So the invoking federate may not be always granted to logical time *T*. Therefore, besides logical time and lookahead, network messages should be considered in a distributed simulation when counting GALT for the NMR and NMRA services.

2.1.3 FQR

There are two ways to compute GALT for the FQR service.

1. When a federate requests advancing to logical time *T* by invoking the FQR service, the advancing request is always granted by the RTI and the federate may arrive at the minimum of GALT, LETS and *T*, which is just the smaller of LITS and *T*. The logical time which the federate arrives at may be less than *T*. Thus when computing GALT, the FQR service can use the same algorithm as the NMR and NMRA services.
2. The FQR service can be regarded as an atomic operation because the FQR service is always granted in any case. As both states before and after the FQR service are granted, only logical time and lookahead should be considered. We adopt the second way in this paper. In addition, The Grant state is that when a federate's time advancing request is granted. A federate's initial state can also be thought as Grant. Consequently, only five states should be taken into account, which are Grant, TAR, TARA, NMR and NMRA.

2.2 Network Messages

An RTI should consider two types of messages on counting GALT, i.e. the messages that are already stored in each federate's TSO queue, and the transient messages that are being transferred in network and haven't arrived in the RTI. Of the two kinds of messages, transient messages are indeterminate for GALT computation because the RTI hasn't received them. In fact, this problem can be solved by using the pipeline mechanism, which means messages sent first from a federate should arrive at the RTI first, i.e. the FIFO (First In First Out) rule.

In Figure 1, after a federate has sent the messages *m1*, *m2* and *m3*, it invokes the TAR, TARA, NMR, NMRA or FQR service to request advancing logical time. It then sends messages *m4* and *m5* to the RTI. According to the IEEE 1516 standard, only the messages before invoking time advancing services (i.e. *m1*, *m2* and *m3*) should be

considered on counting GALT, and the latter messages (i.e. *m4* and *m5*) should not be considered. The correctness is confirmed by the following theorem, which is guaranteed by the IEEE 1516 standard.

Theorem 1 *A federate mustn't send any TSO messages less than T+L in time advancing state. T means the specified logical time that the federate requests to advance to, and L means the actual lookahead when the RTI grants the federate's request.*

The concept of time advancing state is defined in IEEE 1516, which means the interval between a federate's request to advance its logical time by invoking the TAR, TARA, NMR, NMRA or FQR service and the corresponding grant.

Though a federate requests the NMR, NMRA or FQR service to advance to logical time *T*, the RTI may grant it to the logical time which is less than *T*. So the federate may send the messages less than *T+L* after the foregoing request is granted. But the federate absolutely follows theorem 1 in time advancing state.

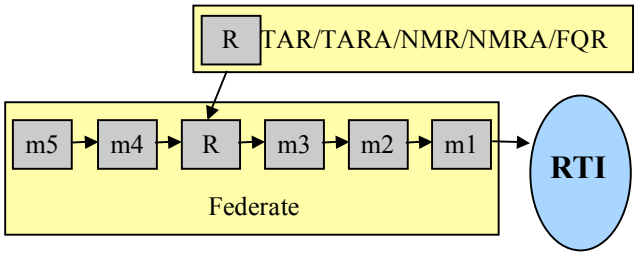

Figure 1: The rule of FIFO.

2.3 Underlying Communication

Another factor is the reliability of message transferring in the network. Underlying communication can be divided into reliable and best-effort models. The former can be implemented by TCP and the latter can be implemented by UDP. Of course, a best-effort message may also be transmitted as a reliable one by TCP.

For the reliable model, it is certain that the messages and time advancing requests must be received by the RTI. But it is not the case at all for the best-effort model implemented by UDP, and this makes the computation of GALT complicated. For the best-effort model, both RTI and federates are difficult to know whether a message has been received by the other side. In the PDES area, a paper by Riley, Fujimoto, and Ammar (2000) showed that the capabilities and reliability of various network models could be exploited in the design of a LBTS algorithm. Because the time management in PDES is different from that in HLA, the LBTS algorithms in PDES shall not be directly migrated to the RTI.

For simple discussion, the FIFO rule and the TCP communication model are used in the remainder of the paper. In addition, we also assume that all federates call the enableTimeRegulation and enableTimeConstrained services to be time regulating and time constrained.

3 ALGORITHM

Since Lamport presented the concept of a logical clock (Lamport 1978), the question about deadlock based on logical time has been a focus in distributed systems. The HLA time management is also faced the problem inevitably. Two deadlock cases exist in the HLA time management. One is caused by zero lookahead (Fujimoto 1997), which has a close connection with the application itself. Lookahead is introduced to resolve this deadlock essentially. The other deadlock is caused by a GALT algorithm.

We start from the following formula, which is suitable for the TAR and TARA services if a federate's lookahead is not modified during the federation.

$$GALT(i) = \min\{T(j) + L(j)\}, \quad i \neq j, \tag{1}$$

where:

- When federate *j* is in time granted state, *T(j)* means federate *j*'s current logical time; when federate *j* is in time advancing state, *T(j)* means the logical time that federate *j* requests advancing to.
- *L(j)* means federate *j*'s lookahead.

Next, we modify formula (1) twice, and the GALT algorithm comes out eventually. The first modifications aims at the changeable lookahead, and the second makes it suitable for the NMR and NMRA services.

3.1 Modifying Lookahead

Now a federate is permitted to modify its lookahead during the whole federation. The following theorem is still tenable.

Theorem 2 *For TAR and TARA, the sum of T(j) and L(j) in formula (1) mustn't be decreased.*

Proof According to the IEEE 1516 standard, *T(j)* can not be decreased. Federate *j* shall receive the 'RTI::LogicalTimeAlreadyPassed' exception if it tries to roll back its logical time. Now we know that only *L(j)* can be decreased. But the modifyLookahead service in the IEEE 1516 standard declares that "If the requested value is less than the joined federate's actual lookahead, the change shall take effect gradually as the joined federate advances its logical time and the actual lookahead is initially unchanged. Specifically, the joined federate's actual lookahead shall decrease by *T* units each time logical time advances *T* units until the requested lookahead is reached." A good algorithm of modifying lookahead can be found in Carothers et al. (1997). Therefore, *T*(*j*)+*L*(*j*) mustn't be decreased. □

Here, we emphasize two issues.

- The modifyLookahead service can only be successfully called when the calling federate is in time granted state in accordance with IEEE 1516. If the requested value is greater than or equal to the joined federate's actual lookahead, the modification shall take effect immediately and the requested lookahead shall become the actual lookahead. If the requested value is less than the joined federate's actual lookahead, the service shall be suspended by the RTI and the change shall take effect later gradually as the joined federate advances its logical time.
- The RTI::InTimeAdvancingState exception will occur when a federate wants to modify its lookahead in time advancing state.

Supposing that a federate is in time advancing state, we mark *L1* as its current lookahead and *L2* as the actual lookahead the federate would have if it is granted. If a federate wants to decrease its lookahead, we have $L1 \geq L2$. Hence, $T(j)+L1 \geq T(j)+L2$. From theorem 2, we know that *L(j)* in formula (1) should be *L2* rather than *L1*.

Now formula (1) may be modified into formula (2) as follows.

$$GALT(i) = \min\{T(j) + L(j)\}, \quad i \neq j, \qquad (2)$$

where:

- When federate *j* is in time granted state, *T(j)* means federate *j*'s current logical time and *L(j)* means federate *j*'s actual lookahead.
- When federate *j* is in time advancing state, *T(j)* means the logical time that federate *j* requests advancing to, and *L(j)* means the lookahead that federate *j* will have once it is granted.

A federate's lookahead shall be considered in order to compute GALT. But the lookahead to be considered is dependent on a federate's state. For convenience, anther definition is introduced.

Definition 2 *Modified Lookahead (ML). When a federate is in time granted state, Modified Lookahead is defined as its actual lookahead. When a federate is in time advancing state, Modified Lookahead is defined as the lookahead the federate will have once it is granted to advance.*

From the definition, we know that a federate's Modified Lookahead is equal to its actual lookahead if it doesn't call the modifyLookahead service. In addition, the lookahead in formula (2) is just a federate's Modified Lookahead.

3.2 GALT Algorithm

For NMR/NMRA, we further modify formula (2) to obtain our GALT algorithm. For any federate *i*, we note *S(i)* as federate *i*'s stature. A federate's stature is different from the output time defined in the book by Kuhl, Weatherly, and Dahmann (1999). It is only a metaphor for numeric comparison.

Definition 3 *A federate i's stature is defined as*

$$s(i) = \begin{cases} T(i) + L(i); \text{if } i \text{ in Grant / TAR / TARA state} \\ \min\{T(i), LETS(i)\} + L(i); \text{if } i \text{ in NMR / NMRA state.} \end{cases}$$

Where:

- *S(i)* is federate *i*'s stature.
- When federate *i* is in time granted state, *T(i)* is federate *i*'s current logical time. When federate *i* is in time advancing state, *T(i)* is the logical time that federate *i* requests to advance to.
- *L(i)* is federate *i*'s Modified Lookahead.
- *LETS(i)* is the smallest time stamp of TSO messages in federate *i*'s TSO queue.

The following algorithm is the GALT algorithm.

GALT Algorithm *For any federate i, its GALT is determined by other federates' statures. GALT(i) is computed as*

$$GALT(i) = \min\{S(j)\}, \quad i \neq j.$$

The algorithm is simple and efficient, and it does not result in deadlock.

Theorem 3 *For the GALT algorithm, if a federate with minimal stature is in time advancing state and its Modified Lookahead is greater than zero, the federate's advancing request must be granted.*

Proof. Suppose that federate *a* has minimal stature, thus *S(a)* is minimal. From the GALT algorithm, we know that $GALT(a)=min\{S(i)\} \geq S(a), i \neq a$.

If *a* is in TAR/TARA state, we know that $GALT(a) \geq T(a)+L(a) > T(a)$. Federate *a*'s request can be granted.

IF *a* is in NMR/NMRA, we get that $GALT(a) \geq min\{T(a),LETS(a)\}+L(a) > min\{T(a),LETS(a)\}$. According to IEEE 1516, the RTI can grant federate *a* to $min\{T(a), LETS(a)\}$.

If *a* is in FQR, the request can always be granted. □

Theorem 4 *If each federate's Modified Lookahead is greater than zero, the GALT algorithm won't result in deadlock.*

Proof. Suppose that the GALT algorithm result in a deadlock. Then all federates must be in time advancing state. For all federates, the set Φ can be defined as $\{S(1), S(2), S(3), ..., S(n)\}$. In the finite set Φ, there must exist a minimal number noted as *S(a)*. By theorem 3, we know

that federate *a*'s request must be granted. This conflicts with the assumption of deadlock. □

4 APPLYING TO RTI1.3-NG

In this section, we apply the GALT algorithm to RTI1.3-NG. The RTI is composed of a Central RTI Component (CRC) and multiple Local RTI Components (LRCs). Here is a possible procedure when a federate calls TAR/TARA/NMR/NMRA/FQR to advance simulation. For simplicity, we do not consider other services such as the enableTimeRegulation, disableTimeRegulation, modifyLookahead, and even resignFederationExecution services. These services may change a federate's stature, and have effects on the advancement of the federation.

1. Each LRC contains an array to save all federates' statures, which are initialized to infinity.
2. When federate *i* calls TAR/TARA/NMR/NMRA/FQR to advance logical time, do following steps. Let's note LRC_i as federate *i*'s LRC.
   (a) LRC_i computes *i*'s stature by definition 3. If the new value is different from (larger or smaller than) *i*'s old stature, LRC_i saves the value in its array and sends it to all other LRCs.
   (b) LRC_i computes *i*'s GALT by our GALT algorithm. The GALT can be easily computed because LRC_i can obtain all federates' stature from its array.
   (c) If *i*'s request can be granted, LRC_i calls the timeAdvanceGrant service to notify federate *i*.
3. When a LRC receives a federate's new stature, the LRC saves the value in its array. If the LRC's federate is in time advancing state, do the same things as step 2.

From the procedure, we know that our GALT algorithm has following characteristics, which show that the algorithm is highly efficient.

- A LRC can compute its federate's GALT immediately without communicating with other LRCs. In fact, a LRC can determine if its federate is granted to advance only according to its local information, i.e. the array for saving statures. Therefore, all LRCs can compute GALTs simultaneously and the lock mechanism for consistency is not necessary although a federate's stature may not be consistent in all LRCs.
- A TSO message has nothing to do with the procedure of GALT computing above. From definition 3, we know that a federate's stature may decrease when it is in NMR/NMRA and receives a smaller TSO message. But the federate's LRC needn't notify the new value to other LRCs. This is guaranteed by theorem 8. An example for decreasing stature is given in Liu et al. (2006b), and Figure 3 in this paper is another one.
- The algorithm is not recursive. This is very important because an RTI needn't detect, resolve, or avoid deadlock resulted from recursion.

5 CORRECTNESS

The conservative mechanism in time management requires that a federate shouldn't receive an outdated TSO message, i.e., it must follow the local causality constraint rule (Fujimoto 2000). For an RTI developer, a good GALT algorithm must adhere to this rule. Now the question is: Will our GALT algorithm result in an outdated TSO message during the federation execution? Theorem 12 declares the correctness of the algorithm.

Theorem 5 *A federate's logical time that the RTI would grant mustn't be greater than its stature.*

The theorem follows from definition 3.

Theorem 6 *For any state, a federate mustn't send any TSO messages with time stamps less than its stature.*

Proof In this paper, the states can be defined as Grant, TAR, TARA, NMR and NMRA. In time granted state, federate *i* mustn't send any TSO messages with time stamps less than the sum of its current time and lookahead. By definition 3, we know that the theorem is correct when federate is in time granted state. In time advancing state, federate *i* mustn't send any TSO messages with time stamps $\leq T(i)+L(i)$ by theorem 1. Here $T(i)$ is the logical time that federate *i* requests to advance to, and $L(i)$ is its Modified Lookahead. By definition 3, we know that $S(i) \leq T(i)+L(i)$. Thus, a federate cannot send any TSO messages less than its stature. □

We reemphasize that a federate in NMR/NMRA advancing state cannot send any messages with time stamps less than its request advancing logical time plus its Modified Lookahead according to the HLA time management. But this does not mean that it will not send such messages forever. An example of decreasing stature is given in Liu et al. (2006b). The concept of stature defined in this paper is different from that of output time defined in the book by authors Kuhl, Weatherly, and Dahmann (1999). Because there is no recursion in the definition of stature, GALT can be computed easily and efficiently.

The following corollary is attained from theorem 6.

Corollary 1 *For any state, a federate mustn't send any TSO messages with time stamps less than the minimal stature in the federation.*

Theorem 7 *If the federate with minimal stature is in time advancing state, the stature after the grant by the RTI mustn't be less than the stature before the grant.*

Proof Suppose that federate *i* is in time advancing state, the requested advancing logical time is represented by the variable *requestTime*, the current actual lookahead is represented by *LT1*, the logical time after the grant by the RTI is represented by *grantTime*, and the corresponding actual lookahead is represented by *LT2*. From definition 2, we know that the Modified Lookahead defined in the GALT algorithm is *LT2* rather than *LT1*. For TAR/TARA, *requestTime* = *grantTime*, and *requestTime+LT2* = *grantTime+LT2*. From definition 3, *requestTime+LT2* is the stature before the grant, and *grantTime+LT2* is the stature after the grant. Hence, the stature after the grant is equal to the stature before the grant. For NMR/NMRA, the stature before the grant is represented by *S1*, and *S2* means the stature after the grant. From corollary 1, we know that other federates mustn't send federate *i* any TSO messages with time stamps less than *S1*. Therefore, the messages that federate *i* receives later don't work on the computation of *S1*, and *S1* shouldn't be decreased by receiving later messages. Now we have that *S1=min{requestTime, LETS}* + *LT2* = *grantTime* + *LT2* = *S2*. □

Theorem 8 *If a federate's stature is not minimal, its stature may be decreased. But the federate's stature mustn't be decreased to be less than the minimal stature.*

Proof When a federate is in time granted state, it can only successfully increase its current lookahead. Otherwise the modifyLookahead service will be suspended when it attempts to decrease its lookahead. In addition, a federate cannot change its lookahead when it is in time advancing state. Hence, a federate's stature mustn't be decreased when it is in Grant/TAR/TARA. Thus a federate's stature can only be decreased when the federate receives the TSO messages with smaller time stamps when it is in NMR/NMRA. From corollary 1, we know that no federate can send any TSO messages with time stamps less than the minimal stature in the federation. From the definition of stature, we know that the NMR/NMRA federate's stature can't be decreased to be less than the minimal stature. □

Theorem 9 *The federate with minimal stature mustn't receive any outdated messages.*

From theorem 8, theorem 7, corollary 1 and theorem 5, we know that theorem 9 is correct apparently.

Theorem 10 *Any federate's current logical time mustn't be larger than any other federates' statures.*

Proof If the RTI grants federate *i* to advance to logical time *T* at any wall clock time, we must have *T≤GALT(i)*. From the GALT algorithm, we know that *GALT(i)* = *min{S(j)}, i≠j*. Hence, *T≤min{S(j)}*. □

The following corollary can be directly obtained from theorem 10.

Corollary 2 *Any federate's current logical time mustn't be larger than the minimal stature in the federation.*

Theorem 11 *If a federate's stature is not minimal, it mustn't receive any outdated messages.*

Proof Suppose that federate *i* is not a federate with minimal stature. From theorem 10, we know that the current logical time of federate *i* mustn't be larger than the minimal stature. From corollary 1, we have that federate *i* mustn't receive any outdated messages. □

Theorem 12 *The GALT algorithm mustn't result in a federate to receive any outdated messages.*

From theorem 9 and theorem 11, we know that the theorem is correct.

6 EXAMPLE ANALYSIS

6.1 Explanation for Algorithm Proving

Figure 2 shows the procedure that nine people climb a pyramidal mountain together from three directions. Here we use this example to explain the correctness in the previous section. Each person is modeled by a black (hollow) image and a red (solid) image. The position of a black image shows the stages where the person arrives at, which is modeled as a federate's current logical time. In Figure 2, *a1*'s logical time is *1*. The values of logical time for *a2, a3, b1, b2, b3, c1, c2, c3* are *2, 3, 1, 3, 2, 2, 2, 1* respectively. Now suppose the lookahead of each person is *3*, and they are all time regulating and time constrained by calling the enableTimeRegulation and enableTimeConstrained services. This means that the farthest distance between the fastest person and the slowest is not more than *3* stages. It may be also the maximal stages for each person to stride for one step but everyone cannot go too far from the others.

The red image is the stature we defined in this paper, and it can be thought as the farthest stage that a person can arrive in the following step. In Figure 2, the statures for *a1, a2, a3, b1, b2, b3, c1, c2, c3* are *4, 5, 6, 4, 6, 5, 5, 5, 4* respectively. A TSO message is modeled by what a person says to another for waiting. For example, If *a1* says to *a3* "please wait for me in the *5th* stage", *a3* will receive the TSO message with logical time *5*.

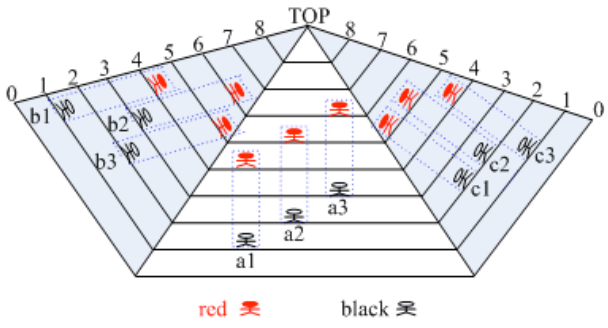

Figure 2: Pyramidal mountain climbing.

- The red image mustn't be under its corresponding black image, i.e. any person shall not walk down at any time. This is what theorem 5 means.
- Anyone cannot ask others to wait for him at the stages under the speaker's red image. For example, *a1* shall not ask *a3* to wait at the *2nd* or *3rd* stage, which is under *a1*'s red image. This is what theorem 6 means.
- Corollary 1 means that anyone shall not ask another to wait for him under the red image with minimal value. In Figure 2, *a1*, *b1* and *a3* have the minimal red images with value of *4*.
- Suppose *a1* wants to arrive at the *4th* stage, He can stride to the stage only when he is granted by all others. Now he is in time advancing state, and his stature is *4+3=7*. Here the value of *4* is the time he requests to arrive at. When he arrives at the *4th* stage, his stature is also *4+3=7*. Here the value of *4* is the time he has arrived at. The procedure is complicated if *a1* wants to arrive at the *5th* stage and he may receive one message with time stamp of *4th* which is less than that he wants to get to. But at that moment of granting, the stature after the grant is equal to (also not less than) that before the grant. This is what theorem 7 says.
- If *a1* wants *a3* to wait for him at the *5th* stage, *a3* will receive the TSO message with logical time *5*. Now *a3*'s stature is decreased to *5* in Figure 3, which is not *6* in Figure 2. Although *a3*'s stature is decreased to *5* from *6*, but it still above *a1*'s stature. Here *a1* is the federate with minimal stature. This is what theorem 8 says. Of course, this is suitable for the NMR and NMRA services, but not for TAR or TARA.
- Theorem 9 is apparent for the cases in Figure 2 and Figure 3. For example, *a1* is one of the slowest, and no one else shall tell him to walk down to wait for the teller.
- All red images are above (not under) any black images in our example, and this is what theorem 10 means.
- All red images are above (not under) the slowest *a1*, *b1* and *c3*'s black images in our example, and this is what the corollary 2 means.
- The *a3*'s stature is not minimal as shown in Figure 2, and he may receive a message with logical time less than his stature as shown in Figure 3. However, he shall not receive any messages that ask him to walk down for waiting, i.e. *a3*'s red image can't be under his black image. This is what theorem 11 means.
- Theorem 12 says that everyone shall not receive any messages that ask him to walk down.

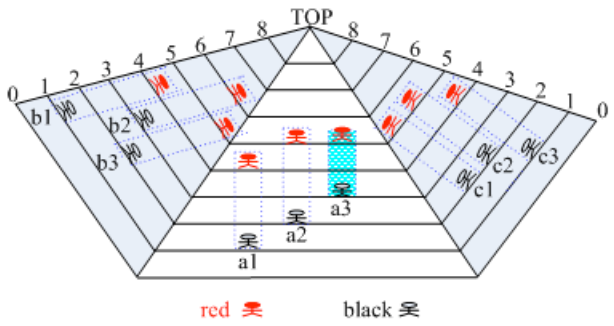

Figure 3: a3 receives a smaller TSO message and its stature is decreased.

6.2 Rationale for Experiments with Thousands of Federates

Now it is time for us to see why the HLA time management in StarLink+ can be used for large-scale simulations with thousands of federates. Suppose that *1,000* people climb the pyramidal mountain from *25* directions in Figure 2; thus, each *40* people climb the mountain from the same direction. In an RTI as RTI1.3-NG, each federate has its LRC. For a simulation with *1,000* federates, all LRCs shall participate in the computation of GALT in time management. If a LRC is modeled as one's mobile telephone, it is nearly impossible today for *1,000* people to walk together up to the mountain because everyone shall connect with other *999* people to know whether he can stride the next step. Note that the distance between the fastest and the slowest is *3* stages, which is the supposed lookahead of each person.

However, the procedure of climbing mountain is more efficient if a team leader is selected in each direction. Now only *25* team leaders communicate with each other by their telephones, while anyone else have not got a telephone if he is not a team leader. When a person wants to advance, he only requests to his team leader. The team leader can know if the person can be granted according to the locations of his all members and all other *24* team leaders. A team leader knows the locations of his all members because each member shall tell the leader his position. In addition, a team leader shall communicate with all other team leaders only when his slowest member moves ahead. Thus, an interesting thing is that a team leader doesn't communicate with another team leader if the member who is not the slowest moves ahead. Similarly, StarLink+ is made up of one Central RTI Server (CRTI) and multiple Local RTI Servers (LRTIs). Each LRTI manages multiple federates, but a federate has no its LRC. A federate communicates with its LRTI via underlying CORBA middleware. The CRTI is useless for time management in StarLink+. Specially, it is highly efficient if one LRTI and its federates are deployed in the same node of a cluster system. In such a cluster system, only all LRTIs participate in the computa-

tion of GALT in StarLink+. From the mountain-climbing example, we know that a LRTI in StarLink+ is approximately equal to a team leader, and a federate is approximately equal to a member. Now we know that it is really not strange for StarLink+ to support thousands of federates in our experiments. In Liu et al. (2006a), we presented the unique hierarchical architecture in StarLink+, which is definitely different from that in RTI1.3-NG and has nothing to do with the concept of hierarchical federations presented in Myjak and Sharp (1999). The RTI1.3-NG software is comprised of the RTI Executive process (RtiExec) and the Federation Executive process (FedExec), and a federate may simultaneously participate in more than one federation (DMSO 2000). However, these FedExec processes are independent, and they don't communicate with each other. HLA is the standard about one federation, and the software must not be an RTI if it can interconnect with different federations. In fact, it is unreasonable to ask an RTI to support the interconnection between multiple federations, not even to compute GALTs subject to multiple federations. However, different federations may be connected by the upper layer application, which is also a federate belonging to multiple federations and is usually called 'bridge', 'gateway', or something similar.

7 CONCLUSION

The computation of GALT is a complicated problem and it is very important for RTI developers to implement the HLA time management. After analyzing some factors about GALT computation such as time advance services, network services and underlying communication mechanisms, we present a GALT algorithm without deadlock nor recursion. The algorithm has been successfully implemented in StarLink+, which can support thousands of federates in high performance computers. Then we analyze the efficiency of the algorithm by applying it to RTI1.3-NG. The algorithm's correctness is also proven. In addition, the proving procedure is explained by an example of mountain climbing. The rationale about the algorithm to be used for thousands of federates within StarLink+ is also introduced. The algorithm in this paper can be practically applied to the implementation of time management in various RTIs.

ACKNOWLEDGMENTS

We thank *the National High Technology Research and Development Program of China* (863 program) under the grant of No. 2006AA01Z330 for its support. We also appreciate *the National Natural Science Foundation of China* under the grant of No. 90412011.

REFERENCES

Carothers, C. D., R. M. Weatherly, R. M. Fujimoto, and A. L. Wilson. 1997. Design and implementation of HLA time management in the RTI version F.0. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, D. H. Withers, and B. L. Nelson, 373-380. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via <http://www.informs-sim.org/wsc97papers/0373.PDF> [accessed March 20, 2007].

Ceranowicz, A. Z., M. Torpey, W. Hellfinstine, J. Evans, and J. Hines. 2002. Reflections on building the joint experimental federation, *Proceedings of the 2002 I/ITSEC Conference*, Orlando, Florida.

DMSO. 2000. RTI 1.3-Next Generation Programmer's Guide Version 6. Available online via <http://hla.dmso.mil> [accessed April 8, 2002].

Fujimoto R. M. 1996. HLA time management: design document. Available online via <http://www.cc.gatech.edu/computing/pads/papers.html> [accessed March 20, 2007].

Fujimoto, R. M. 1997. Zero Lookahead and repeatability in the high level architecture. *1997 Spring Simulation Interoperability Workshop*. Available online via <http://www.cc.gatech.edu/computing/pads/papers.html> [accessed March 20, 2007].

Fujimoto, R. M. 1998. Time Management in the High Level Architecture. *Simulation* 71(6): 388-400. Available online via <http://www.cc.gatech.edu/computing/pads/papers.html> [accessed March 20, 2007].

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

Jefferson, D. R. 1985. Virtual time. *ACM Transactions On Programming Languages and Systems* 7(3): 404-425.

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

Lamport, L. 1978. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM* 21(7): 558-565.

Liu, B. Q., H. M. Wang, and Y. P. Yao. 2005. Data Consistency in a large-scale runtime infrastructure. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 1787-1794. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via <http://www.informs-sim.org/wsc05papers/221.pdf> [accessed March 20, 2007].

Liu, B. Q., Y. P. Yao, J. Tao, and H. M. Wang. 2006a. Development of a runtime infrastructure for large-scale distributed simulations. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1036-1043. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via <http://www.informs-sim.org/wsc06papers/131.pdf> [accessed March 20, 2007].

Liu, B. Q., Y. P. Yao, J. Tao, and H. M. Wang. 2006b. Implementation of time management in a runtime infrastructure. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1044-1052. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via <http://www.informs-sim.org/wsc06papers/132.pdf> [accessed March 20, 2007].

Myjak, M. D., and S. T. Sharp. 1999. Implementations of hierarchical federations. In *Proceedings of 1999 Fall Simulation Interoperability Workshop*, 99F-SIW-180.

NUDT. 2007. Available online via <http://www.nudt.edu.cn/newweb/research/achievement.htm> [accessed March 20, 2007].

Riley, G. F., R. Fujimoto, and M. H. Ammar. 2000. Network aware time management and event distribution. *The Workshop on Parallel and Distributed Simulations*. Available online via <http://www.cc.gatech.edu/computing/pads/papers.html> [accessed March 20, 2007].

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 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 in the school and his research interests include distributed simulation and high performance computing. His e-mail address is <bqliu@nudt.edu.cn>.

YIPING YAO is a professor in the School of Computer at 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. More than 40 papers and 3 monographs were also published. His research areas are distributed simulation and virtual reality. His e-mail address is <ypyao@nudt.edu.cn>.

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). Since 1990, he has chaired more than 10 research projects under the grants of *the National Natural Science Foundation of China*, 863 Program, and *the National Basic Research Program of China* (973 Program), etc. In 2003, he was awarded one 2nd class National Science and Technology Advance Award. Up to now, he has published more than 100 papers and directed 20 graduate students. His research focuses on distributed object, agent technology, grid computing and network security. His e-mail address is <whm_w@163.com>.