Research on Time Management for Multiple Architectures Simulation by Using Gateways

Peng WANG*, Ge LI and Ke-di HUANG

College of Systems Engineering, National University of Defense Technology, Changsha, China

*Corresponding author

Keywords: Time management, Multiple architectures, Gateway, Time synchronization.

Abstract. Multiple architectures simulation means that more than one simulation architecture is used in the joint simulation environment. The time management problem of the multiple architectures simulation is compounded by the architectural differences. This paper gives a proposal for the time management of the multiple architectures simulation. In this paper, gateways are used as independent software applications that translate the protocol used by one simulation architecture to that of the different simulation architecture. A new synchronization algorithm for the machine time of all the simulation nodes is proposed. With time alignment and adjustment, the simulation time and machine time of the multiple architectures simulation is synchronized with the GPS time server. At last, it gives a brief analysis for the synchronization of machine time.

Introduction

With the rapid development of modern networking technology, simulation architectures have made the distributed simulation widely applied. There are various methods of interconnecting simulation applications. High Level Architecture (HLA) [1, 2], Distributed Interactive Simulation (DIS) [3, 4] and Test & Training Enabled Architecture (TENA) [5, 6] are architectures designed specifically for this task. They are the most commonly used architectures for this purpose [7].

Usually, the applications using different architectures couldn’t interact with each other and couldn’t be integrated directly because of the incompatibility of time advancement mechanisms, supported services and data format. Incompatibilities between DIS, HLA, and TENA require the development of point solutions to effectively integrate the different architectures into a single, unified simulation environment, namely the multiple architectures simulation environment [8]. Integration is achieved through gateway solutions, which can make the interaction and synchronization of different simulation architectures come true.

Apparently, HLA, DIS and TENA are heterogeneous architectures. While efforts are underway to address the interconnection of simulation applications using different architectures, we propose to use gateway solutions to support the integration of different simulation architectures. Such integrations require the sourcing of a suitable gateway application as well as the mapping of object models that may be represented in different formats [9]. A gateway provides a connection and translation between two simulation systems using different architectures [10]. A gateway needs to adapt to existing operational interfaces and technologies, using data and interfaces that are provided by architectures.

Time management in multiple architectures simulation should consider time in two distinct ways, one is logical time and the other is real time. For the logical time, simulation time is a variable that is set and advanced within the simulation according to the logic of the participating member applications. For the real time, simulation time passes at the same rate in the executing simulation as it does in the real world. Some simulation architectures, such as DIS and TENA, are designed to support primarily or only real-time execution, whereas some other architecture, such as HLA, offer explicit specialized protocol services to mediate and control simulated time and thereby support logical time.
Integrating multiple time management mechanisms is much more difficult in a multiple architectures simulation, for at least three main reasons. First, integrating multiple architectures needs to bring together member applications or simulation environments using different time management mechanisms, and the different architectures usually have the different time management capabilities. It is difficult to coordinate the different capabilities of heterogeneous architectures. Second, if different time management mechanisms are present in a multiple architectures simulation environment, the differences may have to be resolved or reconciled in the member applications at the architecture boundaries, for example, in the gateways used to connect the architectures. How to synchronize the different architectures using the gateways is also a big problem. Finally, reconciling different time management mechanisms across multiple architectures may involve architecture-specific details and design assumptions, such as time calibration across member applications, network message timestamps, and recovery procedures if processing load causes a member application to fall behind real time. These details and assumptions are potentially quite different across different architectures, and the differences affect the reconciliation of time management mechanisms [7].

This paper mainly talk about how to reconcile different time management mechanisms across architecture boundaries when one time management mechanism is in use in one of the architectures and another time management mechanism is in use in the other architecture.

Methods Available for Time Management

There are usually two kinds of basic methods for time management for the multiple architectures simulation. The first method is to modify the member applications within the multiple architectures simulation environment that are using different time management mechanisms to use the same time management mechanism, resulting in a single common time management mechanism in use throughout the multiple architectures simulation. The second method is to retain the different time management mechanisms and integrate them on the architecture boundary, is conceptually more difficult because it entails reconciling fundamentally different paradigms of time that are often deeply embedded in the basic assumptions under which the member applications were implemented. However, this approach also has a big advantage that it concentrates the changes on the architecture boundary, possibly in a gateway, bridge, or middleware, and thereby avoids the need to convert a number of member applications from one time management mechanism to another.

There is a difficult point for the second approach, which is the connection between logical time and real-time member applications. In such connection, the commonest method to resolve the time difference is to let only one of the two applications execute (or more precisely, to advance simulation time) at any given moment. If the logical time and real-time member applications are to execute concurrently, rather than in alternating fashion, their concurrent execution is most likely to be at real time, because anything other than real time would potentially produce logical inconsistencies or temporal anomalies in the real-time member applications. The connecting gateway will therefore make the time management services available in the logical time architecture, such as time advance requests, to constrain the passage of simulation time in the logical time portion of the simulation environment to match real time. In a multiple architectures simulation, the gateways can be regarded as the pace masters. In this approach the logical time member applications continue to use logical time as before, but that logical time passes at real time. This method is to make sure all the gateways and simulation nodes using different simulation architectures have the synchronized machine time.
Synchronization of Machine Time

The multiple architectures simulation belongs to the system-level hybrid simulation. Usually, the simulation subsystems to be integrated are already fashioned. And the simulation frameworks of the simulation subsystems are inconsistent because of the different real-time constraints [11]. Consequently, when we realize the synchronization of machine time for the multiple architectures simulation, we should solve the following two problems:

1. The discrepant simulation starting time. The simulation subsystems drive their simulation engines according to their own fixed logical timer, and use the starting point of the logical timer as their simulation starting time. If all the subsystems of the multiple architectures simulation use their own logical time that technically designed for themselves, it would result in the inconsistent starting point of all the subsystems.

2. The unsynchronized simulation time advancement. The real-time requirements and the simulation step sizes of different simulation subsystems are inconsistent. For example, the time step of some system-level simulations is considerably larger than that of the hardware-in-the-loop simulation. But sometimes they would exchange a large amount of simulation command or interactions. If the inconsistent timing intervals or errors of output time sequence exist, it will result in the unsynchronized time advancement of simulation subsystems, and the causal order of simulation events would be disordered.

General Framework for the Synchronization of Machine Time

In the WAN, we use the GPS time server to synchronize the gateways of all the subsystems that are using different architectures; and in the LAN, we use the gateways to synchronize all the simulation nodes that are using the same architecture.

Under normal conditions, the simulation systems using the architectures of DIS and TENA need to be running in the real time. Some real-time simulation equipment, such as simulation turntable or signal simulator, has more rigorous real-time requirements. In order to reduce the transmission delay, we choose to use the reflective memory networks which have higher transmission rate, to connect their gateways and the GPS time server. As HLA usually uses the logical time to drive its simulation engine and has its own RTI to schedule the simulation events, we use the Ethernet to connect its gateway and the GPS time server. The network structure is shown as Fig. 1.

Figure 1. The architecture on which the method is depending.

For the subsystems using different architectures, the synchronization of machine time is much more difficult. We propose a strategy to realize this purpose. The strategy is that we let all the subsystems participating in the simulation advance their simulation time basing on their own time...
advancement mechanisms. This strategy is similar to the independent time advancement, but we must ensure that the machine time of every subsystem is consistent with the global time or the real-world time. We realize the consistency of the machine time of all the nodes of the multiple architectures simulation by altering their machine time according to the global time.

We set a GPS time server in the multiple architectures simulation. The GPS time server chooses the Ethernet network or the reflective memory network to send the standard GPS second impulse to the gateways of all the subsystems basing on their real-time requirements. We regard the gateways as the time management nodes of the subsystems of which they are in charge. For the subsystems using HLA, we let their gateways communicate with their RTI directly by shared memory or reflective memory network. In this way, we can ensure that the gateways of all the subsystems are synchronized. And the gateways are the time server of the subsystems they belong to. Furthermore, we should ensure that the gateways are unilaterally restrained by the subsystems of which they are in charge. This strategy is a kind of time synchronization method combining the GPS time signals and the data-driven simulation.

The GPS time server provides the standard GPS second impulse and synchronized time information for the multiple architectures simulation. However, it is just not enough to satisfy the simulation step requirements of the real-time systems, so we must do the time-subsection basing on the standard GPS second impulse. The gateway of every subsystem is in charge of the time-subsection for the subsystem to which it is connected. By time-subsection, the gateway could send the frame synchronization signals to the simulation nodes of the subsystem as is shown by Fig. 2. During the running of the multiple architectures simulation, all the simulation nodes adjust their machine time according to the GPS frame synchronization signal received from the Ethernet or reflective memory networks.

Synchronization of Gateways

The gateways are in charge of the synchronization of the subsystems. Different gateways are connected to the subsystems that are using different architectures, and the transmission rate of the network they are using is also different. The gateways use the Ethernet interfaces or the reflective memory interfaces to receive the outside time synchronization signal. When they use the reflective memory interfaces to receive the outside time synchronization signal, the network transmission delay of the signal can be ignored. On this occasion, we should mainly consider the discrepant starting time of these subsystems. When they use the Ethernet interfaces to receive the outside time synchronization signal, we should consider both the discrepant starting time of simulation subsystems and the unsynchronized simulation advancement of simulation subsystems. Here we mainly talk about how to solve these problems, and we will propose a method to make the gateways’ machine time synchronized with the GPS time server.

The modified model of the machine time for the gateways is as follows:
Here \( T_m \) is the nominal time cycle of the machine time, \( T_m' \) is the actual time cycle, \( T_{m0} \) is the initial time migration, and \( c(t) \) is the compensation factor of the machine time migration. We name the process of making certain the initial time migration \( T_{m0} \) as time alignment. We name the process of making certain the compensating factor of the machine time migration \( c(t) \) as time adjustment. Now we will study the realization of them.

**Time Alignment**

The time alignment of gateways and the GPS time server is realized by a software alignment algorithm called low-frequency alignment. The alignment is implemented by the subsystem’s gateway at the moment of the GPS frame synchronization signal arrives. We calculate \( T_{m0} \) by using the following formula:

\[
T_{m0} = f_{MT}(T_G) - f_{GT}(T_G)
\]  

(2)

Here \( T_G \) is the time point of the gateway asking for time service. This means we only do the time alignment at specific time points. With this method we can eliminate the error of time synchronization when the simulation is running. The operation of time alignment has two stages, one is called starting time alignment, while the other one is called running time alignment.

**a) Starting time alignment:** All the simulation nodes do not have interaction with the others at the beginning of the multiple architectures simulation, so the network load is low at this moment. We use the master-slave probabilistic algorithm to realize the starting time alignment \[12\]. By multiple times of time data packets sending and receiving, we can eliminate the influence of the variability of the network transmission time.

We measure the average transmission time (as represented as \( \overline{d} \) ) of the time data packets. The GPS time server sends packets of time synchronization data with the time interval \( r \) and every data packet includes the time information (as represented as \( T_i^M \) ) of the GPS time server. We record the machine time \( T_i^S \) when the gateway receives the ith time packet. The initial time migration of this gateway can be calculated by

\[
T_{m0} = \overline{T^S} - (\overline{T^M} + \overline{d})
\]  

(3)

\( \overline{T^S} \) is the average value of \( T_i^M \), and \( \overline{T^S} \) is the average value of \( T_i^S \). The modified machine time of the gateway is as follows:

\[
\hat{T} = T_i^S - \overline{T^S} + \overline{T^M} + \overline{d}
\]  

**b) Running time alignment:** When the machine time of the gateway has a deviation (as represented by \( \tau \) ) from the true global time, we can eliminate the deviation by modifying the initial time migration by

\[
T_{m0} = T_{m0} + \tau.
\]

As is shown in the Fig. 3, we use the deterministic real time synchronization algorithm (also named by S-C-S double direction query) to realize the running time alignment. The concrete steps of this algorithm are shown as follows.

![Figure 3. The process of running time alignment](image-url)
1) The GPS time server sends the command packet P1 which includes the command of starting running time alignment to the gateway. Then it turns its state to network querying and detects circularly the arrival of the network data packets.

2) After receiving the command packet P1, the gateway queries its machine time $t_s$ at proper time, and then sends the time data packet P2 which includes the machine time $t_s$ to the GPS time server.

3) After receiving the time data packet P2, the GPS time server reads the packet and calculates the deviation $\epsilon$ between the machine time of the gateway and the standard time $t_m$ of the GPS time server. The deviation is $\epsilon = t_s + \hat{d} - t_m$, where $\hat{d}$ is the estimated value of the network delay time. Then the GPS time server sends the command data packet P3 which includes the deviation $\epsilon$ to the gateway.

4) After receiving the command data packet P3, the gateway adjusts its machine time according to the formula $T_{m0} = T_{m0} - \epsilon$.

**Time Adjustment**

As most of the computers in the simulation system are using the electronic impulse to generate the time pulse, the error between the nominal cycle of the clock unit and the $i$th impulse cycle is the error of machine time. Here the nominal cycle of the clock unit is represented as $T_m$, the $i$th impulse cycle which counts from the computer’s starting up is represented as $T_{im}$. The relationship between the $i$th impulse cycle and the nominal cycle is as follows.

$$ T_{im} = \frac{c^i}{p_{im}} = \frac{c^i}{(p_m + p_n + \epsilon_i)} = T_m \times \frac{c^i}{(1 + T_m \times (p_n + \epsilon_i))} = T_m \times c^i $$

(4)

Here $c^i$ is the initial frequency error of the computer’s time unit. $p_m$ is the nominal frequency of the computer’s time unit. $p_n$ is the frequency drift error. $\epsilon_i$ is the stochastic disturbance. $c^i$ is the frequency error of the clock unit’s actual cycle $T_{im}$.

From Eq. 4 we can see that there is an error between the nominal cycle of the clock unit $T_m$ and the clock unit’s actual cycle $T_{im}$ caused by the frequency deviation. When we use the computer’s machine time to simulate the natural time without compensating the frequency deviation, if the simulation duration is short, the error would be small and can be neglected. But if the simulation is long, the error would be big and we can’t neglect it. For example, if the computer’s time unit’s frequency accuracy ranges from $10^{-4}$ to $10^{-6}$, it means that there would be an error range from 1 microsecond to 100 microseconds every second. If the simulation duration is 1 hour, the accumulative error would be ranging from 3.6 milliseconds to 360 milliseconds. For the real-time simulation, this error would be intolerable, and we usually use the compensation factor of the machine time migration $c(t)$ to adjust the error.

If the simulation duration is only a few hours, the initial frequency error of the computer’s time unit is the main factor and we can neglect the influence of the frequency drift error $p_n$. Under this occasion, $c^i$ would turn to be a constant parameter, and the compensation factor of the machine time migration $c(t)$ satisfies the formula as $c(t) = c^i$. Then we can have the model of the machine time we need. The basic measurement method is that the GPS time server broadcasts two time data packets with the time interval $T$ at the beginning of real-time simulation. The gateway receives these two time data packets and measures the time interval $T'$ of the two data packets basing on its own machine time and then calculates the time migration adjustment factor $C = T' / T$. As $C$ is a scaling factor, it is unlike the starting time alignment. The error of starting time alignment is fixed and would not diffuse. While the time migration factor is very sensitive to the error, we must ensure that the measurement of
the time migration factor is sufficiently precise. In order to guarantee the accuracy of measurement, we usually want the time interval \( T \) as large as possible. For example, if the time accuracy of the GPS time server is \( dT=1\mu s \), the time interval needs to satisfy \( T>7.2s \).

For most simulations, we usually regard the compensating factor of the machine time migration \( c(t) \) as \( c(t) = c' \). But if the simulation duration is much longer, the simulation requires higher precision, or the quality of the crystal oscillator is bad, we must take the error of the pulse period of the machine time into account. Under this condition, the error of the pulse period of the machine time is \( c' = c' \times (1 + p_n) \), here the frequency drift error \( p_n \) is time-varying. The compensation factor of the machine time migration \( c(t) \) wouldn’t be a constant parameter, and it should be the average value of all the \( c'_i, (i=1, \ldots, n) \) belonging to \( (0, t) \), so we can get \( c(t) = \sum_{i=0}^{n} c'_i / n \). In order to get the accurate machine time model, we must adjust the machine time during the running of the simulation. One way is that we regard the compensating factor of the machine time migration \( c(t) \) as a constant parameter, and adjust the initial time migration \( T_{m0} \) during the running of the simulation. Another way is that we regard the initial time migration \( T_{m0} \) as a constant parameter and adjust the time migration adjustment factor \( C \) during the running of the simulation.

**Algorithm Analysis**

Firstly, we analyze the algorithm of time alignment. The error of time synchronization (as is represented as \( \varepsilon \)) is mainly caused by the error between the network’s actual latency \( d \) and the network’s estimated latency \( \hat{d} \), and it can be calculated by \( \varepsilon = d - \hat{d} \).

We have known that \( \hat{d} = d \), so we have \( \varepsilon = d - \left(1/n\right) \sum_{i=1}^{n} d_i \). The main value of the error of time synchronization can be calculated by \( E(\varepsilon) = d - E\left(\left(1/n\right) \sum_{i=1}^{n} d_i \right) \). If \( n \rightarrow \infty \) which means the number of the measurements of \( d \) tends to infinity, we can have \( \varepsilon = d - E(d) \). The variance of the error of time synchronization is \( D(\varepsilon) = \left(1/n^2\right) \sum_{i=1}^{n} D(d_i) = \sigma^2/n \).

By observing the mean value and variance of the error of time synchronization, we can see that if the number of times of synchronization is enough, the error of time synchronization can reach arbitrary accuracy. But we must consider the cost of the realization of the algorithm. The algorithm needs to measure the average value of the network’s latency beforehand, and during the process of synchronization, we need to send data packet for \( n \) times. In order to assure the irrelevance of the latency, the interval of sending the data packets need to be bigger than 1 second. So the cost of this algorithm can’t be neglected. If the number \( n \) is bigger, the cost of time would be bigger, and the variance of the error of time synchronization would be smaller. So we should maintain a balance between the accuracy of time synchronization and the cost of realization.

Secondly, we analyze the algorithm of time adjustment. From Eq. 4, we can have follows:

\[
dc = dc/dT' + dc/dT = dT' / T - T' \times dT / T^2 \approx dT' / T - dT / T
\]  

(5)

From Eq. 5 we can see that the measurement error of the initial frequency error \( c' \) is not only related to the computer’s time accuracy, but also related to the interval of sending the data packets. For instance, if the simulation duration is 1 hour and we want the time error to be smaller than 1 millisecond (the initial error is not included). So we can have \( dc/c \approx dc < 1/36000000 \) and \( |dT/T| < 1/(36000000 \times 2) \). The interval of sending the data packets \( T \) must satisfy \( T > 7.2 \times 10^6 \times dT' \). If \( dT' \) is 1 millisecond, the interval of sending the data packets \( T > 7200s \). If \( dT' \) is 1 microsecond, the interval of sending the data packets \( T > 7.2s \). So we can see that the accuracy of the computer’s time unit is very important.
Summary
In this paper we give a proposal for the time management of the multiple architectures simulation. The basic idea is the synchronization of the machine time for all the nodes of the multiple architectures simulation. We focus on the time alignment and time adjustment for the synchronization of gateways. In the future we plan to extend our proposal to the other kinds of multiple architectures simulations.

Acknowledgement
This research is supported by National Natural Science Foundation of China under Grant No.61374185.

References