Research and realization of RTI data transmission method based on attribute reflection

Long Wang, Juan Fang, Qiang Xiao and Liang Cai*

Jiuquan Satellite Launch Centre, Jiuquan, Gansu, China

Keywords: HLA, Attribute Reflection, Call-back function.

Abstract. In order to solve the error of simulation result because of data loss in VR-Link traditional simulation framework, in this paper data transfer mechanism of instance attribute on VR-Link object class is researched in detail. Registration method of call-back function is given, and the key codes of relevant steps are published. The problem of data loss is solved by the call-back function of VR-Link.

1 Introduction

In HLA simulation applications developed based on MAK RTI, VR-Link is generally used to develop HLA framework code to achieve network communication for simulation applications. Simulation applications (federal members) are generally implemented with a dual-thread structure of "interface thread + simulation thread". Interface threads are used to implement human-computer interaction and visualization of simulation data. Simulation threads are used to establish simulation loops, send and receive network data, and implement specific Simulation applications.

In the simulation loop, before the simulation thread updates the HLA object instance properties owned by the member, the other object class information needed is obtained by traversing the reflection object list, and the reflection object list is automatically generated by the VR-Link framework code according to the RTI working mechanism. According to the time advancement mechanism of RTI and the difference in advancement speed of each HLA member, when the HLA member traverses the reflection object list, the reflection object list may or may not be updated. The traversal process cannot ensure that the traversed data is the simulation the cycle is up-to-date, so there are situations where simulation members use outdated data and discard valid data, that is, data is lost. In other words, even under the same input conditions, the simulation output or result will not be exactly the same, and there will always be a certain difference. In a sense, this means that the simulation results are wrong, not errors.

In order to solve the problem of "lost counts" in the HLA framework code based on VR-Link and realize the passive notification process of the simulated members, this article

* Corresponding author: 362052456@qq.com

studies the attribute callback mechanism of VR-Link and gives the registration of various Callback function method and key code.

2 General simulation structure based on Mak RTI

When developing simulation applications based on Mak RTI, the general structure of "interface thread + simulation thread" is generally used. The concise general simulation structure diagram of members based on VC MDI is shown in Figure 1. The interface thread is generally implemented in the form of SDI, MDI, or dialog based on VC. The simulation thread is generally created and started by the interface thread. The communication between the two can adopt multiple communication methods, and generally uses the method of sharing global data.

Fig. 1. Concise general simulation structure diagram of members based on VC MDI.

3 VR-Link attribute transfer mechanism

According to the HLA specification, VR-Link's mechanism for transmitting data through RTI can be simply summarized as "one party announces and one party receives" or "must announce before receiving." As far as the object class is concerned, the specific process is as follows: First, the object class instance is announced to the RTI through the publishing process, the registration application of the object class instance in the RTI is completed, and the tick operation is used to request the RTI to update the instance attributes to implement object class data release. Second, the object class is ordered from the RTI through the ordering process, and the object instance attributes are automatically received by reflecting the object list. When users need data, they only need to traverse the list of reflection objects to know the number of object class instances and the attributes of each object instance.

The following uses an object class named Class A as an example to analyze the process of publishing data and receiving data in detail.

3.1 Analysis of the mechanism of data released by the publisher

To publish data, the publisher needs to go through several steps, such as publishing the object class, registering the object instance, and updating the instance attributes.

3.1.1 Publication of object classes and registration examples

Announcing the object class and registered instance is actually declaring the following information to the RTI: An HLA object is added to the simulation federation represented by the DtExerciseConn class instance, which initializes an object class instance to the RTI.

The process of publishing the object class completes the following tasks: First, the global exercise pointer and object class instance name "ClassA" represented by the DtExerciseConn class instance are passed to the base class DtObjectPublisher of the publishing object class, so that the base class gets the publishing object class The basic information required; second, the local state repository and the remote state repository are specified for the base class; third, the setHlaObject function of the published base class is used to indicate which instance of the object class is to be published, and the encoding used for the object instance And decoder.

It can be seen that the object class publishing process interconnects the attribute warehouse, encoder, and decoder of the object class.

3.1.2 Update instance properties

When updating the attribute value of an instance, the attribute value setting function of the "property warehouse" instance specified for the object instance through the "publication link" of the object class is used to set the value of each attribute, and then the object attribute update function tick of the object publication class is completed The new attribute value is sent to the RTI.

3.2 Analysis of the data receiving mechanism of the purchaser

In simple terms, the orderer receives the data of the object instance through the reflection object list of the object class.

3.2.1 Order object class and its function

The reflection object list class of the object class ClassA is ClassAReflectedList. This list class inherits from VR-Link's reflection object list base class DtReflectedObjectList, to conveniently use the functions provided by the base class reflection object list.

The method for implementing the order object class ClassA is to call the order function of the ClassA object class in the framework code function of the simulation thread. This function takes a pointer to the DtExerciseConn class as a parameter, uses the constructor of the object class to reflect the object list, and instantiates the ClassA object class to reflect the object list.

The instantiation process accomplishes two things. First, the DtExerciseConn class instance is passed by reference to the base class of the reflection object list. The base class of the reflection object list can obtain all the information of the federation according to the global exercise pointer. This process is automatically implemented by VR-Link. 2. Call the addObjectClassByName function of the reflection object list base class to notify the reflection object list base class that the object class "ObjectRoot.ClassA" has been ordered.

3.2.2 Response of the purchaser after the target instance is announced

When the publisher performs the operation of publishing the object class ClassA, the function newReflectedObject of the ordering object class ClassA's reflection object list ClassAReflectedList is automatically called by VR-Link to notify the reflection object list that there is a new object that needs to be reflected. The reflection object list uses this information to automatically construct the reflection object and implement the attribute reflection process of the object instance published by the publisher

VR-Link only passes one message to the newReflectedObject function, which is the HLA object. The function newReflectedObject uses the passed information DtHlaObject * obj and the exercise pointer myExConn of the reflection class list base class of the object class ClassA to create an instance of the reflection class ClassAReflected. This operation associates the reflection class with the reflection list class.

In the process of constructing the reflection object class, by initializing the base class DtReflectedObject of the reflection object class, the constructor specifies the object class attribute warehouse, which is used to store the reflected attribute values. In the constructor body, by calling the newReflectedObject function of the reflection object base class, the decoding class of the object class ClassA is specified.

3.2.3 Order response after updating data

When the publisher updates the data, the decoder of the reflection object class will be automatically called by VR-Link to complete the instance attribute decoding. After the decoding is completed, the attribute value will be stored in the object class attribute warehouse ClassARepository.

3.2.4 Extract the attributes of objects in the reflection object list

When the user needs the data of the reflection object instance, he can search or traverse to the "reflection object instance" corresponding to the "publish object instance" through the reflection object list, and obtain the corresponding attribute data through the function of "reflection object instance".

4 Implementation method of property update callback function

The HLA framework refers to various interaction classes and object classes implemented using VR-Link to complete interaction with the RTI, as well as reflective object list classes, reflective object classes, object encoding classes, object decoding classes, interactive encoding classes, and interactions. Implementation code framework of decoding class, object class attribute warehouse, etc.

4.1 Register "Find Object Instance" and "Delete Object Instance" callback functions

The reflection object list class provides support for registering / unregistering "discover object instances" and "delete object instances" callback functions, including addObjectAdditionCallback, removeObjectAdditionCallback, addObjectRemovalCallback, removeObjectRemovalCallback, etc.

Through the "Find Object Instance" and "Delete Object Instance" callback functions, users can know the existence of the object instance in the RTI, and can make some logical

judgments, such as setting the instance existence flag to true when the instance is found. When deleting an instance, set the instance presence flag to false. Assume the following logic. If the instance existence flag is true and the attribute update flag is true, use the instance data, otherwise wait for the data to be updated. This can ensure that the used data is up-to-date, and the program must Wait until the property is updated before performing subsequent work, otherwise it will always wait for data update, so the implementation of these two callback functions is still necessary for some application requirements.

4.2 Register "Attribute Update" callback function

If you want to use the data after the property data of the object instance is updated, it is best for VR-Link to call back a user-defined function after the data is updated to notify the user that the property has been updated. VR-Link provides the function `baseAddPostUpdateCallback` in the base class `DtReflectedObject` of the reflection object class, which is used to add and delete callback functions after the property is updated. The callback function can be registered in the constructor of the reflection object class.

4.3 Implementation of Property Update Function

The "Attribute Update" callback function is called by VR-Link after decoding is completed. In this function, you can use the attribute warehouse class to extract data without manual decoding. The specific implementation method is the same as the original framework.

4.4 Summary

The registration method based on VR-Link framework code realizes the reuse of the original framework code. Without destroying the original framework code, the registration of the attribute update callback function was implemented, and the decoding module of the original framework code was used to avoid the tedious manual decoding.

5 Conclusions

In order to solve the problem of missing data in the simulation thread, the attribute update callback mechanism and the implementation method of the attribute update callback function of Mak VR-Link are studied in this paper. Application research shows that by registering the attribute update callback function, it becomes "active to traverse the instance data". Using the method, the data interaction between the data publisher and the data orderer can be reliably realized, and the loss of data is avoided.

References

1. Translated by Fu Zhengjun, etc. HLA technology in computer simulation. National Defense Industry Press,2003.06
2. Zhou Yan et al. HLA simulation program design. Electronic Industry Press,,2002.06
3. VR-Link 3.9.6 HLA Class Documentation, MAK Technologies, Inc, 2005.
4. VR-Link 3.9.6 Developer's Guide, MAK Technologies, Inc, 2005.