Research and realization of message bus architecture for LAMOST control system

Lingzhe Xu[1,2]

1. National Astronomical Observatories / Nanjing Institute of Astronomical Optics & Technology, Chinese Academy of Sciences, Nanjing 210042

2. Graduate School of the Chinese Academy of Sciences, Beijing 100049

ABSTRACT

The LAMOST (Large sky Area Multi-Object fibre Spectroscopic Telescope) has now come to its final completion of R&D stage. Major functions of the telescope have successfully passed a serial site tests recently, and various kinds of applications integrated into the automation of the telescope chamber is being under vigorous tests too. The TCS (Telescope Control System) is built on multi-layer distributed network platform with many sub-systems at different levels. How to efficiently process the enormous amount of message with particular implications running in and out the TCS is one of the major issues of the TCS software programming. The paper describes the mechanism and methodology of the LAMOST message bus structure. The realisation of message bus architecture as a result of years of research and site test is presented in general, and dealing with the message priority and manipulating smallest piece of message in parallel or in serial sequence are elaborated in particular.

Keywords: LAMOST, TCS, message bus, communication, link

1. INTRODUCTION

The Large sky Area Multi-Object fiber Spectroscopic Telescope (LAMOST), one of the ongoing national large scientific and engineering projects, is approaching to its last stage and the final completion is due to the end of 2008. The telescope will be able to observe 4000 celestial objects simultaneously, thus become the world's most powerful ground astronomical optical survey instrument with 6m aperture[3]. The price of such a unique design philosophy is an extraordinary technical challenge. Of course, the fulfillment of such a task would put unprecedented challenge to our control people. Figure 1 shows the TCS（Telescope Control System） with its major function modules below and OCS (Observatory Control System) on its top. The OCS is at the top in the hierarchy responsible for supporting the on-site observer and system operator in their tasks and coordinates the activities of

Advanced Software and Control for Astronomy II, edited by Alan Bridger, Nicole M. Radziwill
Proc. of SPIE Vol. 7019, 701922, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.787059

the other three principal systems immediately at one level below the OCS in the hierarchy, namely TCS, Instrument Control System (ICS) and Data Handling System (DHS).

TCS is the centre of network control. A great amount of its routine work is involved in messages communication. Most messages are coming from OCS and the TCS's 8 subsystems. Message bus architecture has been taken to manage the communication. The main function of message bus architecture is outlined below [1].

1. Receive command messages from OCS
2. Decompose messages into atom messages
3. Send atom messages by priorities in parallel or in serial sequence
4. Receive status messages from subsystems
5. Send status messages to OCS
6. M&C (Monitor and Control) message life time

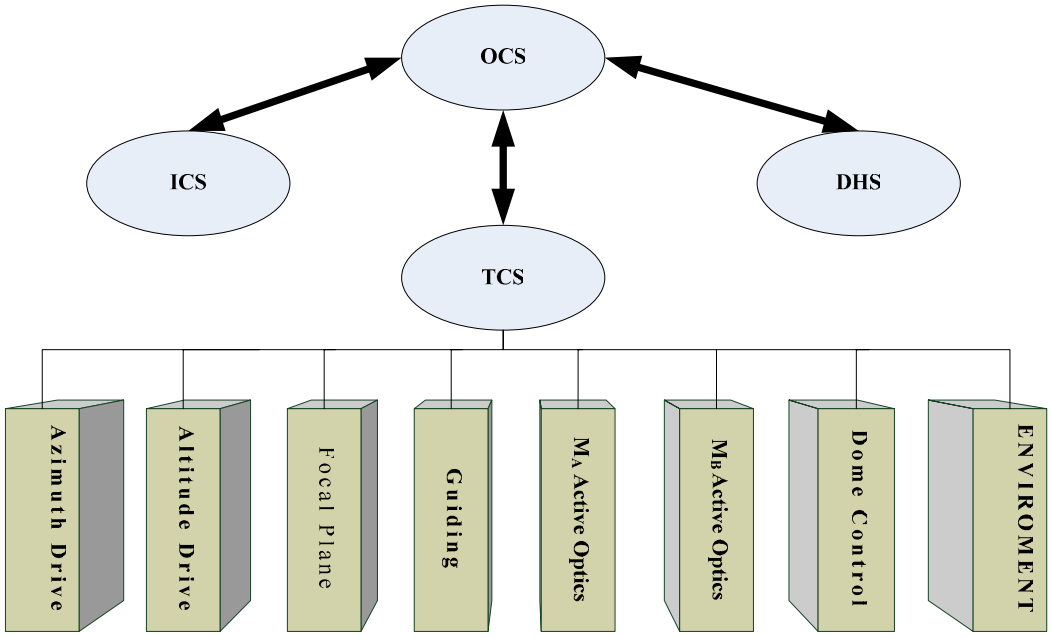

Figure 1: LAMOST high-level control and interfaces

2. Outline of message bus architecture

2.1 Modules of message bus architecture

Nowadays most telescopes' OCSs are built on Linux OS, one of favorite OSs for astronomers worldwide, and so is the LAMOST's OCS. The choice of Linux as OCS platform is because Linux features powerful tools for astronomical data processing and it is free. [4]. TCSs on the other hand usually need real time platforms for time critical applications. Particularly in the case of LAMOST for

manipulating the messages from OCS and 8 subsystems QNX OS has been chosen as the corner stone in network real time application. The QNX is cost effective, a well known distributed real time OS. In addition, with QNX's FLEET networking protocol it is possible to form a switching platform by scaling to as many subsystems (nodes) as are required in a software-transparent manner [2]. It also provides multitasking, priority-driven preemptive scheduling and fast context switching, all are essential ingredients of a real time system. Socket communication technique with TCP/IP protocol is adopted between the LAMOST's TCS and OCS on Ethernet network. Socket communication, database based ODBC communication and Rs232 communication are adopted between TCS and subsystems because of various kinds of subsystem hardwires [5]. So, the message bus architecture is divided into 3 parts, message bus control module, command message bus module and status message bus module. Message bus control module is in charge of sending command to subsystems and receiving command from command message bus module. Command message bus module is in charge of receiving command from OCS and decomposing the command into atom messages. Status message bus module is in charge of receiving status from subsystems and sending the message to message bus control module. Message bus control module is the headquarters of message bus architecture, which decides the transmission of messages. By this way, the message bus control module is independent of the change of OCS or subsystems.

2.2 Message process flow

A typical message process flow is shown in figure 2, which is elaborated in following steps.

1. OCS sends a command message
2. Command message bus module receives the message and decomposes it into atom messages
3. Command message bus sends the atom message to the message bus control module
4. Message bus control module puts the atom messages into message bus by its priority in parallel or serial sequence
5. Message bus control module sends the current message to TCS and saves it in database
6. TCS sends the message to subsystems through command generator
7. Status message bus module receives a status message from subsystems
8. Status message bus module analyses the status message
9. Status message bus module sends the disposal message to message bus control module
10. Message bus control module updates message bus
11. Message bus control module sends the status message to TCS and save it in database
12. TCS sends the status message to OCS through status generator

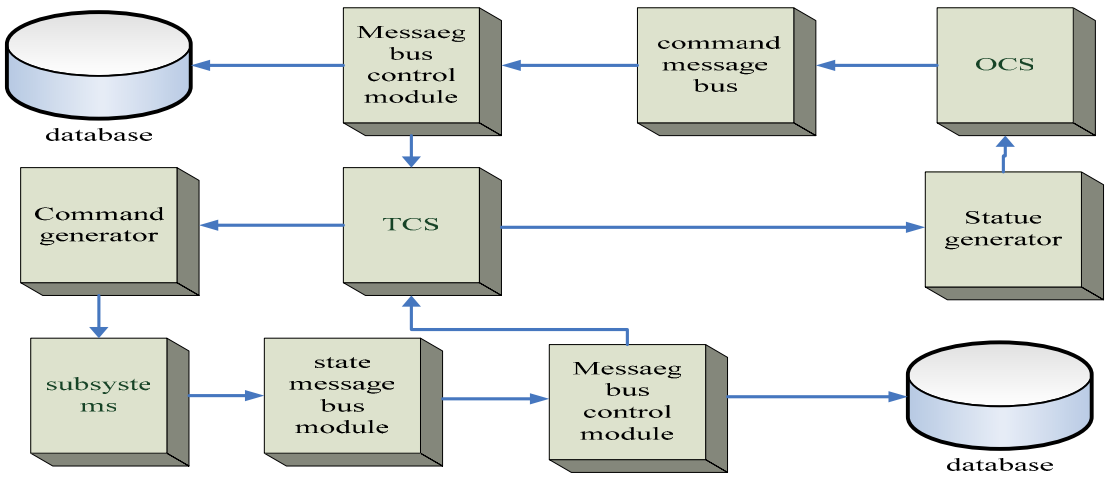

Figure 2: Message process flow in message bus architecture

3. Design Highlights of message bus architecture

3.1 Message components

A command which OCS sends to command message bus module is either atom message or combined message. Command message bus module decomposes the combined message into atom messages by querying the database. There is a "message bus table" in database which defines the relation between every combined message and atom messages. In addition the "message bus table" also defines priority and sequence, either parallel or serial, for all the atom messages. Again the priority and sequence for all atom messages are defined according to their real time application in the telescope observation flow of celestial objects. The "message bus table" consists of two items, "item id" and "message". The "item id" defines the priority of message. The lower the value is the higher the priority is. The higher priority message must be processed before the lower priority message except for the "item id" equals 1, meaning the message is a global, which must be sent immediately despite of its priority. The item "message" defines the commands of all the messages and their maximal life time. The maximal life time indicates the deadline of the command to be done.

With same "item id" the corresponding commands could be different. The priority can be changed based on the real time application without changing the program. Table 1 is a simplified sample of the "message bus table".

item id	message
1	refrigeration，environment monitor

2	ventilation
3	Self check
4	mount ready
5	open dome
6	open focus door
7	open focus
8	MB focus
9	close focus
10	track
11	SH track
12	close focus door
13	close dome

Table 1: Simplified "message bus table"

3.2 Data structure in message bus

Message bus is designed as a chain bintree. Every chain is an atom message. The TCS working team describes the structure of the chain bintree by C program.

```
Struct st_message
{
    int i_degree;    //priority
    char message[512]; //the atom message
    Struct st_message *pst_child ;//the child chain
    Struct st_message *pst_brother; //the brother chain
    int i_flag; //flag
    int i_time; //the maximal lifetime
    char s_time ;//begin time
}
```

The character of the data structure is described below:

1. the root chain is the ancestor of all other chain
2. the chain is the child chain of the current chain when it is pointed by the current chain's

child chain

3. the current chain is the father chain of its child chain
4. the chain is the right brother chain of the current chain when it is pointed by the current chain's brother chain
5. the current chain is the left chain of its left brother chain
6. all the chain is the direct offspring of root chain when it can be visited through root chain's child chain
7. if there is no right brother chain or child chain then the current chain points to empty

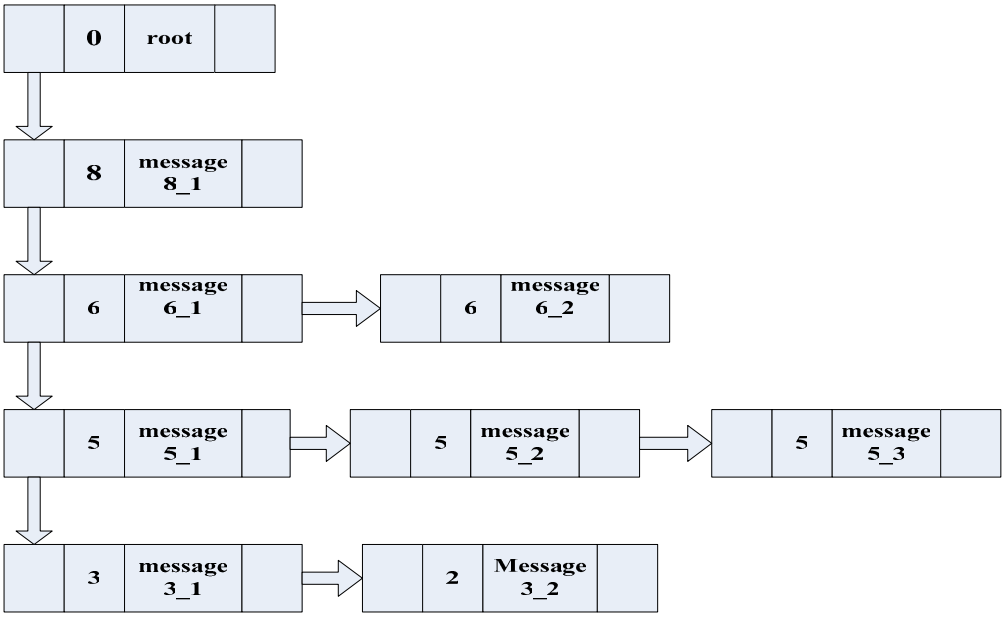

Figure 3.Data structure in message bus

For example, figure 3 shows the data structure of the message bus. The root chain points to the 8_1 chain. 8_1 chain is the lowest priority chain in message bus. The 3_1 chain is the highest priority chain in message bus. The 3_2 chain is the right brother of 3_1 chain. They have the same priority. The 3_1 chain is the child chain of 5_1 chain. The 5_1 chain is the father chain of 3_1 chain. The 3_1 chain and 3_2 chain are the current chains; they must be done before the 5_1 chain.

3.3 Function and variable design

e: e is the chain of the message bus

e1: e1 is the chain of the message bus

root: root is the root chain of the message bus

Visit(e) : visit the chain e of the message bus

Child(e): get the chain e's child chain. If the chain e has no child chain then return empty

Brother(e):get the chain e's right brother chain. If the chain e has no right brother chain then return empty

Delete(e): delete the chain e from message bus

3.4 Process for receiving messages

The algorithm for inserting a new message into message bus is programmed below. When a new message is sent to message bus, it is set to "unsent".

e_new: is the chain to be inserted into message bus

```
e1 = root;
while(Visit(Child(e1))!=empty)
{
    if(e_new->i_degree > Child(e1)->i_degree)
    {
        e_new->pst_child = Child(e1);
        e_new->pst_brother= NULL;
        e1->pst_child = e_new;
        return;
    }
    if(e_new->i_degree == Child(e1)->i_degree)
    {
        e1 = Child(e1);
        while(Visit(Brother(e1)!=empty) e1=Brother(e1);
        e_new->pst_child = NULL;
        e_new->pst_brother= NULL;
        e1-> pst_brother = e_new;
        return;
    }
    e1 = Child(e1);
    if(Visit(Child(e1))=empty)
    {
```

```
            e_new->pst_child =NULL;
            e_new->pst_brother= NULL;
            e1->pst_child = e_new;
           return;
        }
    }
    e_new->pst_child =NULL;
    e_new->pst_brother= NULL;
    e1->pst_child = e_new;
    return;
```

3.5 Process for sending messages

The message which is set to "unsent" and it is at lowest level of message bus will be sent to TCS, and it will be set to "sent". Depending on the level of message priority the implement of inserting a message is shown below in 3 different ways, figure 4 to figure 6.

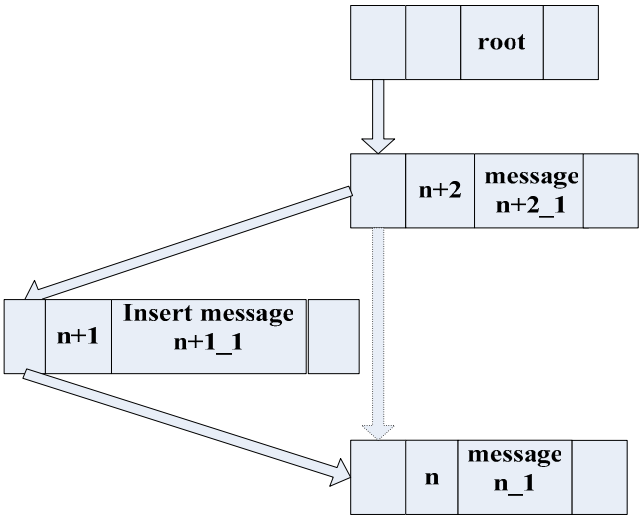

Figure 4.The priority of message to be inserted is higher than the current message's

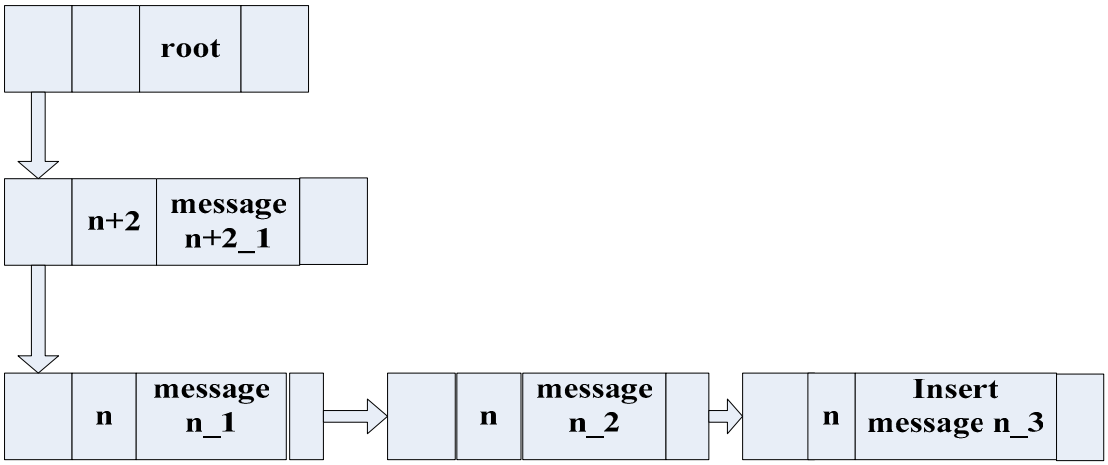

Figure 5.The priority of message to be inserted is equal with current message's

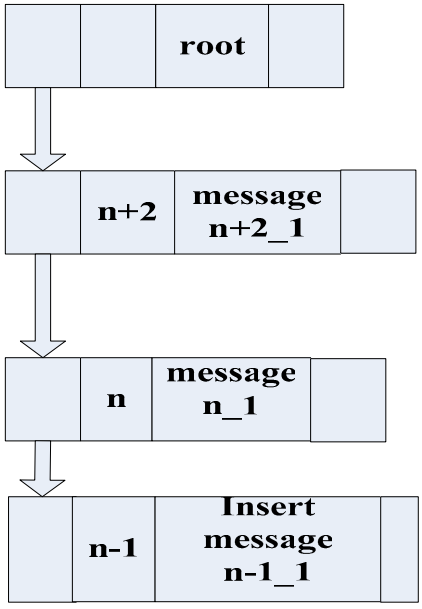

Figure 6.The priority of message to be inserted is less than current message's

3.6 Process for deleting messages

When a message expires, it should be deleted from message bus. The programming for deleting message from message bus is shown below, from figure 7 to figure 9.

e_delete: is the chain to be deleted from message bus

e1 = root;

while(Visit(Child(Child(e1)))!=empty)

{

```
        e1 = Child(e1);
}
If(Child(e1)==e_delete)
{
        e1-> pst_child = Child(e1)->pst_brother;
        Delete(Child(e1));
        return;
}
e1 = Child(e1);
while(Visit(Brother(e1))!=empty)
{
        if(Brother(e1) == e_delete)
        {
                e1-> pst_brother = e1-> pst_brother->pst_brother;
                Delete(e_delete);
                return;
        }
        if(Brother(Brother(e1))=empty)
        {
                e1->pst_brother = NULL;
                Delete(e_delete);
                return;
        }
        e1 = Brother(e1);
}
```

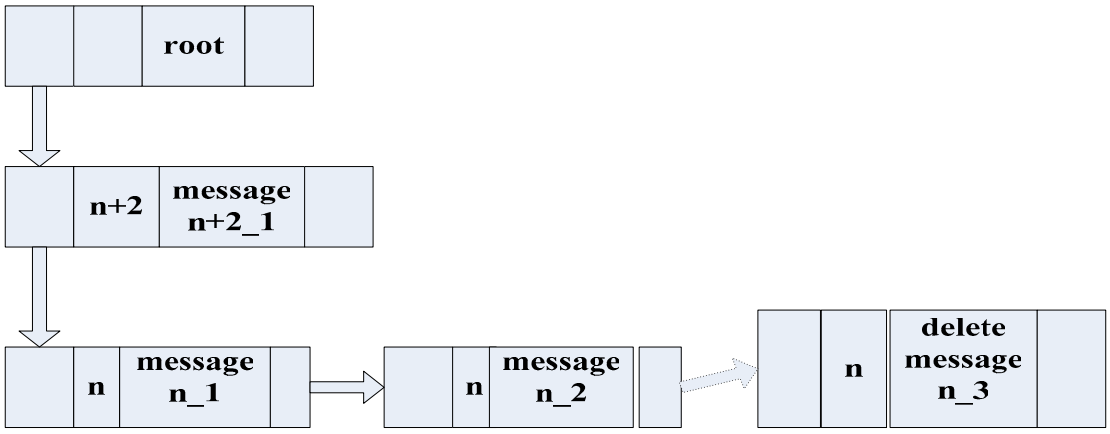

Figure 7.The method 1 of delete message

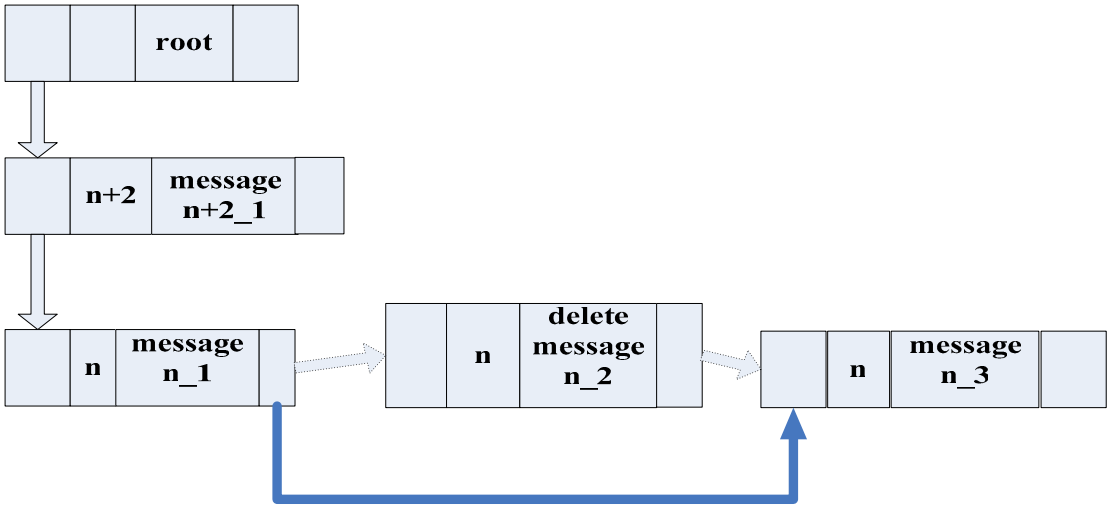

Figure 8.The method 2 of delete message

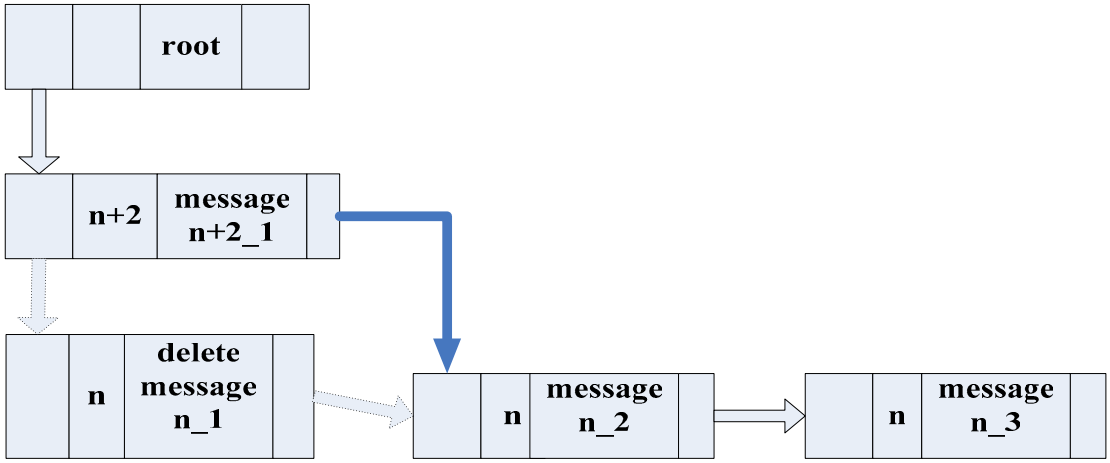

Figure 9.The method 3 of delete message

4. The conclusion

The realization of message bus architecture for the control system of LAMOST telescope is elaborated above. The mechanism and technique tips involved in the message structure, message manipulation and its programming are all illustrated in detail. The research work has passed a critical review and evaluation, and undergone a vigorous on site test. Such an approach for message process in LAMOST's high level software programming has been proved practical, effective and robust. Above all the advantage of such a strategy makes it possible that TCS application is independent of the sequence order in which the telescope is under operation for getting ready to observe a celestial object. The reason is quite obvious because the user is able to update the database to adjust the message's attributes so that to modify the observation preparation steps.

REFERENCES

1. Xinqi Xu, "Control system and technical requirements - preliminary design", LAMOST Internal Technical Report, 1998.
2. Xinqi Xu, "Study of QNX real time operation system and exploration on its application in large astronomical telescopes", ASTRONOMICAL INSTRUMENT AND TECHNOLOGY, 1999.
3. Shou-guan Wang, Ding-qiang Su, Yao-quan Chu, Xiangqun Cui, and Ya-nan Wang, "Special configuration of a very large Schmidt telescope for extensive astronomical spectroscopic observation", Appl. Opt. 35, pp. 5155-5161, 1996.
4. Xinqi Xu, Lingzhe Xu, Gangping Jin, "Overview of LAMOST control system", SPIE Volume 4837, 2002.
5. Lingzhe Xu, Xinqi Xu, "Application of real time database to LAMOST Control System", SPIE Volume 5496, 2004.