Design of embedded inter-board communication middleware based on LCM

Qing Li¹, Xuehao Yin¹, Yang Wang¹, Kaihua Cui¹ and Tao Zhu¹

¹Department of Computing, Jiangsu Automation Research Institute, Lian Yungang, Jiangsu, 222000 China

Abstract. Aiming at the distributed architecture of embedded computer system, the network communication middleware is designed based on Lightweight Communication and Marshalling (LCM). In order to improve the reliability and performance of inter-board message interaction, middleware realizes the bidirectional named pipeline, message publication confirmation mechanism, subscription message multithreading processing mode and other improved design schemes, and provides the standardized programming interface while inheriting the topic of "publish-subscribe" based on LCM. Finally, the message transmission delay design was tested on the hardware (Loongson 2K1000 processor) and software (VxWorks operating system) platform. The results show that the message transmission delay is very close to that of traditional UDP by using LCM-based embedded inter-board network communication middleware.

1. Introduction

With the increasing complexity of equipment functional requirements and the development of network communication technology, more and more distributed architectures are used in equipment computer systems [1]. The embedded computer boards with different special functions are connected through the network to make them work together and meet the functional requirements of equipment. Compared with desktop computer systems, these embedded computer boards with special functions can flexibly configure software and hardware platforms according to their business characteristics. Because the system may have a large number of inter-board network data interaction [2], middleware technology must be used in the system design to shield the differences between software and hardware and reduces the workload of development [3].

Nowadays, network communication middleware based on data distribution service (DDS) is becoming more and more popular [4]. As a communication standard middleware based on the publish/subscribe model, it encapsulates the details of the underlying data network. The upper application is no longer concerned with the complex network programming technology, and establishes the communication relationship by creating data publishers and subscribers and relying on business-related "topics". Although DDS has many advantages, the tools used by DDS instances are highly correlated with each other and have poor portability. Therefore, it is difficult to be transplanted on the multi-processing real-time operating system, and DDS middleware is difficult to be deployed on the embedded system and ensure the running performance. For the inter-board network communication of embedded computer system based on distributed architecture, the best solution is to adopt the middleware with publish/subscribe function that is very light weight. LCM is a messaging and marshalling communication library and a Lightweight network messaging system, which was originally developed in 2006 for the MIT DARPA urban challenge team, and then becoming a...
gradually popular application, providing "publish-subscribe" function based on "topic" [5-6]. Since LCM focuses on the network communication services for embedded applications within LAN, it is light and simple enough to adapt to the low load and low latency messaging service requirements of embedded systems [7]. This paper designs a network communication middleware based on LCM to simplify the implementation of data exchange between boards in embedded computer system based on distributed architecture.

2. Overall scheme design

2.1. Architecture for LCM communication
LCM provides a local area network (LAN) publish/subscribe messaging system based on "push" on top of the UDP multicast protocol layer. As shown in Figure 1, after the publisher sends message “topic A”, it is available to both other participants in the communication domain, but is filtered so that only subscribers can process the message for topic A. Participants who do not subscribe to topic A will discard the message.

![Figure 1. LAN publish/subscribe messaging system.](image)

LCM provides multiple versions to suit different programming languages and different operating systems. The complete LCM itself contains four parts: type specification, marshalling, communication and tools. The type specification refers to the way and syntax of defining compound data types and the unique way in which LCM is used for data exchange between processes. Marshalling refers to the process of encoding and decoding messages into binary data for the communication system responsible for the actual transmission. Communication can be summarized as a publish-subscribe messaging system using UDP multicast as its underlying transport layer. LCM provides useful tools for logging, playback, and traffic checking [1]. For embedded computer system, LCMLite based on C language is more suitable for network communication middleware development. LCMLite is the slimmest version of LCM, designed to minimize external dependencies and memory usage. LCMLite has very low requirements for operating system components and only relies on the network protocol stack and POSIX standard Socket network programming interface, so it is easier to port to the new embedded computer hardware and software platform, with less real-time loss. However, because the LCMLite features are too simple and can only support LCM coding and basic messaging publish/subscribe functions, and the LCMLite based standardized application network programming needs to be further expanded to accommodate the use of complex scenarios.

2.2. Design of software framework
Since the embedded computer board card is used to complete a specific function, it will automatically start a business application software after the operating system is started. This software is coordinated by multiple functional modules. Network communication middleware can be embedded in the application software in the form of a "library", called by different functional modules, and then realize the network interaction with other computer boards.

The hierarchy of embedded inter-board network communication middleware based on LCM is shown in Figure 2. From bottom to top are three layers, including embedded operating system
containing UDP/IP, network communication middleware and software function module. The upper two layers constitute the complete business software.

Network communication middleware consists of four parts: UDP multicast data transceiver, LCM, message queue management, and message publish and subscribe programming interface. 1) the sending and receiving part of UDP multicast data USES Socket network programming interface to realize the transmission of message data within the network multicast group; 2) LCM part is LCMLite, which can realize LCM format message encoding/decoding, group package/unpacking, "publish-subscribe" based on "topic" and other basic functions; 3) the message queue management part USES the message queue mechanism to provide a caching and queuing mechanism for the publication and subscription of the topic to improve the reliability of transmission. Also Posix interface programming is adopted to improve portability; 4) the publish-subscribe programming interface part provides a standardized C language programming interface for software programming. Among the four parts of middleware, message queue management is the core of this design.

![Software hierarchy diagram.](image)

**Figure 2. Software hierarchy diagram.**

3. Framework for software

At design time, according to the distributed architecture of embedded system board communication between actual demand, from news to improve system capacity and improve the transmission reliability and the communication pipeline, release confirmation, subscription processing threading and other aspects of design, and finally through the analysis of publish-subscribe interaction process, and put forward the standardized programming interface.

3.1. Message directed transmission

The LCM-based message communication mechanism requires the creation of an LCM instance when the business application is initialized. During creation, an embedded computer is added to a multicast group at a specified address. After multiple embedded computers in the system create an LCM instance and join this "group", the LCM instance USES this "group" for data transmission. After a computer board card publishes a topic message through an LCM communication instance, any other LCM instance in the communication domain can receive the message. But not every LCM instance subscribes to this message, and to reduce unnecessary processing, a named channel is created for each LCM instance in the communication domain. After one participant publishes data on the native named channel, another participant can receive packets by "subscribing" to the channel's data, as shown in Figure 3.
In order to enhance the data interaction between the computer boards in the system, especially to adapt to the application scenarios such as read-back results and confirmations, a bi-directional pipeline strategy is adopted in the middleware design. Pipelines are also used to effectively manage the flow of each message packet within the communication domain.

The name of each pipe in a communication domain should be determined at system planning time and should be unique. When you publish a message group package, you can make the message properties contain information about the name of the source channel and one or more target channels. After receiving multicast, the middleware can automatically parse and match the name of the channel of this board and the target channel of the message. If they are consistent, the subsequent processing of the message can be carried out and the directional release of the message can be realized. At the same time, because of carrying the source pipeline information, it is easy to realize the return of processing results and receive confirmation and other functions.

3.2. Release confirmation

Due to the use of multicast protocol based on UDP as the basis of network message communication protocol, while UDP multicast is not reliable, considering the embedded systems usually have high reliability and application requirements. On the basis of the LCM message coding mechanism, increasing the timeout to release mechanism no answer and republishing one theme in the case of no answer for important messages between two pieces of computer interface card are highly reliable directional release of information.

For important topic messages, the receive reply property needs to be added to the properties option of the message. After the message is published, the publisher middleware automatically checks whether the message packet parameter configuration contains the declaration that requires the receiver to reply. If so, the message packet is inserted into the message topic corresponding resend queue and the timeout timer is started and the flow of the message publishing process is shown in Figure 4(a). When a subscriber receives a message packet to reply, it replies to the source pipe with the message packet number information. After the original message publisher receives the reply packet, it retrieves the resend queue and deletes the corresponding ordinal message packet. The receiving process of the reply packet is shown in Figure 4(b). If the timer is timed out, it will enter the timeout processing thread, read out the message packet at the head of the queue, resend, and publish the timeout processing thread as shown in Figure 4(c).
3.3. Threaded subscription processing

Business software is composed of multiple functional modules, each of which has a large number of inter-board data interaction requirements. Especially in the aspect of message subscription, it is highly possible for multiple topic messages to arrive in one set. To improve message processing performance, subscription message processing needs to be threaded. Each functional module in business application software needs module registration before using network middleware. The middleware creates a "dedicated" subscription message processing thread and a message processing queue for the module. Next, the function module can register the topic to be subscribed, specify the topic subscription information such as the topic message processing hook function, and save it in the topic subscription information list. The subject name, like the address of a mailbox, must be unique in the communication domain.

The topic message processing hook function is the function module carries on the topic message processing procedure, may have the different processing function for the different topic. After the LCM instance receives an identified topic message, it first retrieves the topic subscription information linked list. When a matching topic is found, the functional module subscribing to the topic can be read from the linked list, and the message is distributed to the corresponding thread of the functional module for processing, as shown in Figure 5. Assuming the topic message subscribed by function module 1 is received, the handle to the subscription message processing thread of function module 1 is found, and this new message is added to the end of the message queue of function module 1. The message processing thread of function module 1 continuously processes messages in the message queue and executes the topic matching callback function.

(a) Message releasing  (b) Responding packet receiving and processing  (c) Processing threads of publishing timeout

Figure 4. Process of messaging release.
3.4. Publishing and subscribing exchange process
When the business software on the embedded computer board is initialized, the LCM communication instance is created and the handle of the LCM communication instance is returned. Publishers and subscribers are actually a functional module of business software deployed on different single boards. They both send and receive topic messages through the same network middleware. The interaction process of publish-subscribe is described in Figure 6.

During topic publishing, publishers can directly call publishing functions to publish messages. The publisher specifies the message properties, passing the topic properties and data that need to be published to the LCMLite publish function, and calls the Socket interface to send the message directly.

During the topic subscription process, the subscriber registers both the subscriber and the topic to subscribe to in the middleware. Middleware will always listen to the message in the multicast group on the network. Once the message is received, it will decode it, match the target pipeline, and retrieve the list of topic subscription information. After the match is successful, the topic message processing function will be automatically called.

Through the analysis of the above interaction process, the number of interaction interfaces between publishers and subscribers and middleware is not large, so the publish and subscribe programming interface generally only needs to include the creation and cancellation of LCM instance, message publishing, subscriber registration and cancellation, topic subscription registration and cancellation, etc., which is relatively concise.
Testing and validation

According to the characteristic of message communication, the performance of network communication middleware pays more attention to the transmission delay. For middleware with topic-based publish/subscribe capabilities, it makes more sense to measure the elapsed time from the time a publisher issues a topic message to the time a subscriber's topic handler is called.

We built the test environment based on Loongson 2K1000 single board computer. The test hardware environment is composed of two network-based 2K1000 mono-board computers, both of which are deployed with VxWorks operating system. These two computers act as communication participants to send data to each other. Here, we can assume that the transmission delay of these two participants is the same. For comparison purposes, we also tested the data transfer delay of UDP without middleware. However, this comparison is not fair, because in the case of middleware, after this transmission delay, the application can be directly used, and after receiving the UDP message, the application software needs to analyze the message and schedule the processing function by itself. Therefore, we also simulated the general use of UDP to obtain the transmission delay index closer to the application.

The specific test method includes three conditions that are in the use of middleware, simulation application UDP, and pure UDP. The computer A directly sends valid 64 bytes data to the computer B. Then B received the message packet, and sent the data back to the computer A repeatedly. Calculate the consumed time that computer A sends the first packet data to receive 1000th package data, and the average transmission delay is:

\[ t_{\text{delay}} = \frac{t}{1000} / 2 \]  

(1)

The test results are shown in Table 1. It can be seen from the test results that compared with the pure UDP data transmission, the time delay of UDP data transmission with packet parsing and scheduling processing increases greatly after the use of middleware, but it is close to that of UDP data transmission with packet parsing and scheduling processing, indicating that the performance loss for the actual application is very low.
Table 1. Test results for average transmission delay

| Conditions          | Middleware used | UDP simulation | Pure UDP |
|---------------------|-----------------|----------------|----------|
| Average transmission delay (us) | 292             | 263            | 105      |

5. Conclusion
Aiming at the characteristics of inter-board network communication of embedded computer system with distributed architecture, an embedded inter-board network communication middleware is designed based on LCM. Considering the cross-platform characteristics of the embedded system, based on the "topic" "publishing-subscribe" function provided by LCMLite, the paper improves the design of the pipeline directional transmission of LCM instance, the confirmation of message publication, and the threading of message subscription, and proposes a standardized programming interface. The inter-board interaction of middleware is easier, and the transmission reliability of important messages is improved. Additionally, the processing efficiency of subscription messages is enhanced. Through test and verification, the average transmission delay of the middleware of inter-board network communication is very close to the actual application performance of UDP, but there is still a lot of space for optimization, which can be further improved as a following work.

References
[1] Zastavniy O M, Krytskevych N D, Nykolaychuk Y M. Architecture and System Characteristic of Distributed Computer Network with Autonomous Sensors Equipment[C]// International Conference-modern Problems of Radio Engineering. 2007.
[2] Chuang, Chen, Bor-Sen, et al. Interspecies protein-protein interaction network construction for characterization of host-pathogen interactions: a Candida albicans-zebrafish interaction study[J]. Bmc Systems Biology, 2013, 7(1):79.
[3] Wang X, Bian F, Yong H. The Approach of Land Administration Information System with Middleware Technology[J]. Journal of Wuhan Technical University of Surveying & Mapping, 1999.
[4] Feiock D, Hill J H. Using Component-based Middleware to Design and Implement Data Distribution Service (DDS) Systems[C]// Software Engineering & Advanced Applications. 2013.
[5] Huang A S , Olson E , Moore D C . LCM: Lightweight Communications and Marshalling[C]// IEEE/RSJ International Conference on Intelligent Robots & Systems. IEEE, 2010.
[6] Bingham B S, Walls J M, Eustice R M. Development of a Flexible Command and Control Software Architecture for Marine Robotic Applications[J]. Marine Technology Society Journal, 2011, 45(3):25-36.
[7] Thomas A, Menassa C C, Kamat V R. Distributed simulation framework to analyze the energy effects of adaptive thermal comfort behavior of building occupants[C]// Winter Simulation Conference. 2016.