Design of transceiver based on functional programming

Zhuomao Zhao ¹, Yujie Yang ², Xue Long ³, Hongwei Cui ⁴
¹, ³, ⁴ Beijing Institute of Structure and Environment Engineering, Beijing, 100876, China
² Beijing Institute of Precision Mechatronics and Controls, Beijing, 100876, China
*Corresponding author’s e-mail: Wicky0607@163.com

Abstract: This paper discusses the setting of transceiver based on functional programming, and decouples the differences between communication protocol and network transmission protocol from the specific business implementation. In the actual application, the service implementation is set up. When the communication protocol changes, the function can be realized only by modifying the communication protocol, which improves the readability of the program and reduces its maintenance overhead.

1. Introduction
In all kinds of information and automation applications, the information sending and receiving mechanism is an essential part. The conventional information sending and receiving mechanism will set the corresponding communication protocol, send information according to the protocol content, and analyze the received data packets.

There are four parts in this process: 1、 data packaging and analysis; 2、 Packet cache; 3、 sending and receiving information; 4、 Network communication. With the increase of applications, we will find in the actual scientific research and production the above contents are always designed and implemented repeatedly for unused applications, as well as the protocol contents of the same application will be continuously adjusted with the deepening of design, Which will lead to further adjust the program, along with a lot of repetitive workload to the development and design work.

Based on functional programming (LF), a more flexible mechanism suitable for the design of all kinds of transceivers is proposed in this paper.

2. Conceptual model based on functional programming
Function is parameterized in functional programming, and function is also used to describe the specific expression of a certain kind of behavior. In functional programming, functions have the same status as conventional variables and can be defined anywhere. They can be used as parameters and return values of functions inside or outside functions, and can be combined with functions[2].

For example, there is a science and technology conference to be held, and only those who meet specific conditions can participate. It is described in program language as: Isqualified (fun (A)); Here, fun (A) is used to express whether A meets certain conditions. Fun(A) can have different expressions according to different situations. It is parameterized to describe whether it meets specific conditions. With this expression, the judgment of meeting specific conditions is decoupled from the original
business logic, simultaneous interpreting different parameters (functions) according to the different circumstances, without affecting the original business logic expression.

The sending and receiving mechanism is shown in the figure below[1]. The sending data is packaged according to the communication protocol and put into the sending data cache list <sendpack>, then the data packets in the cache are sent to the specified IP and port number on the network at a certain frequency through the timer; Regularly the data packets transmitted from the network are received and put into the received data cache list <recvpack>, the received data packets are analyzed according to the communication protocol.

![Figure 1 Data transform](image)

For different applications, the transceiver mechanism conforms to the above process, but the communication protocol is different from the network transmission protocol. In order to design a universal transceiver mechanism which can flexibly change the communication protocol and network transmission protocol, it is necessary to parameterize the packaging and parsing behavior function in the communication protocol.

In this situation, ‘senpack’ and ‘recvpack’ represent a structure that contains the main information of the data, including the actions accompanied with sending or receiving data, which is parameterized as parameters so as to meet the programming structure.

3. Function realization

3.1. Data packaging and analysis

The conventional and relatively complete data packet format is as follows:

![Figure 2 Data package](image)

The first part is the identification bit, which identifies the function and function of the record; The second part is the data length bit, which records the data length of the data package; The third part is data bits, which have different packaging and parsing methods according to different instructions; The fourth part is the check bit. Generally, parity check and data length bit can be used to judge whether there is any abnormality in the data.

The example of communication protocol is as follows:

| Flag | identification | data type | Parameter meaning | data sources  |
|------|----------------|-----------|-------------------|---------------|
| 1    | upstream       | String    | Master angle      | Manual entry  |
| 2    | upstream       | Int       | Number of channels| Equipment setting |
| 3    | down           | Double[]  | Parameter value   | Automatic acquisition |
To pack the data of communication, first we should confirm the data type according to the identification of the data; second the data should be conversed into binary data corresponding to the data type; third the identification bit, data length bit, data bits and check bit is spliced together according to the data packet format, and finally the binary data packet sent in the process of network communication is obtained; Data parsing is the reverse process of data packaging. The core of these processes is to transfer the data packaging and parsing method as a parameter, the packaging or parsing methods of different data types can be quickly retrieved through the data dictionary. And then these methods can be called as a parameter in the program framework. This process is shown in the following figure:

3.2. Transceiver thread

Based on the idea of functional programming, the establishment of a general transceiver framework boils down to the design of a standardized data structure and general process, which can cover most scenes, in which different operation parts transfer parameters with parameterized functions.

Through summary, when sending data, its data structure ‘send datapack’ is shown in the figure. It describes (1) sending data identification number (tag): indicating what kind of data is sent (2) before sending action: including judging whether the current condition determines to send the message, Acquisition of message data, etc. (3) whether to confirm sending (isenable): it can be judged in the operation before sending (4) data packet (rawdata): it can obtain the visual data value and package it
before sending (5) sending operation (sendaction). At this time, it can be determined according to the type of communication protocol (TCP / UDP / RDP) (6) after sending action: update the status.

Based on the above data structure, the general process of the sending thread is determined, as shown in the figure above: read the data from the data cache, identify the data identification number, and call the 'beforesendaction', 'sendaction' and 'aftersendaction' functions in turn. The implementation of the functions can be implemented according to different businesses.

The receiving thread also adopts a similar idea, and its diagram is as follows:

Once a general transceiver framework is established through the above data structure and general process, the advantages of functional programming can be brought into play, that is, the internal implementation is no longer coupled with the framework itself. When the business changes, programmers do not need to pay attention to communication, packet sending and receiving, but only the business itself, encapsulate the implementation of the business into the functions of the above data structure, and then call the function in the form of parameters to realize the relevant business.

3.3. overall design

As shown in the figure above, the sending and receiving process includes four parts: underlying network communication, data cache, data sending and receiving, data packaging and analysis. The network communication can be implemented in different ways according to the different communication protocols of the selected data link layer; The data analysis module can be used as a general module for divination; The data cache and sending and receiving thread can be adjusted according to different services and managed centrally without affecting other structures of the program. Based on the way of functional programming, this design encapsulates the specific implementation in one function and passes it as parameters, and realizes a framework which is generally suitable for most of the data communication transceiver[3].

In addition, in the process of data processing, factors such as sending and receiving speed and CPU processing speed will affect the receiving and parsing of data. If the rates do not match, it is easy to
cause packet loss or memory overflow. Caching is a good solution. Although caching can solve the above problems to a certain extent, we still need to pay attention to the matching of communication speed and information processing speed.

4. Conclusion and Application
This paper discusses the setting of transceiver based on functional programming, and decouples the differences between communication protocol and network transmission protocol from the specific business implementation. In the actual application, the service implementation is set up. When the communication protocol changes, the function can be realized only by modifying the communication protocol, which improves the readability of the program and reduces its maintenance overhead.

References
[1] YanPing Chen, Zehua Ding, Qinghua Zhen, “Protocol programming: A layered Programming Paradigm,” ICEBE, 12-14 Oct. 2018, 306-311.
[2] Sara Buse Chian, Vladimir Ivanov, Alan Rogers, “Understanding the impact of pair programming on the minds of Developers,” 2018 IEEE/ACM 40th International Conference on Software Engineering. 27 May – 3 June 2018, 85-88.
[3] Kire Trivodaliev, Biljana Risteska Stojkoska, “Teaching computer programming: The macedonian case study of functional programming” EDUCON, 2017 IEEE, April 2017, 1282-1289.
[4] Abdullah Khanfor, Ye Yang, “An overview of practical Impacts of Functional Programming” APSECW, Dec, 2017, 50-54.