Design and implementation of OBU terminal for Vehicle Ad-hoc Network

Yan Wu\(^*\), Pengkun Wang\(^2\) and Chao Sun\(^3\)
School of Computer and Information, Hefei University of Technology, Hefei, Anhui, China

\(^*\)Corresponding author’s e-mail: 2018170842@mail.hfut.edu.cn

Abstract. In order to solve the traffic problems and the needs of users' safe driving, according to the requirements of the function of the Vehicle Ad-hoc Network (VANET), this paper designs and implements the application function of VANET based on the onboard unit (OBU) terminal in Android system. The design selects ARM development board as OBU terminal and uses Android operating system to design UI interface according to functional modules to interact with users. The key issues and technologies are designed and improved, which implement the security application based on security message processing and expand the non-security application, so as to achieve the purpose of assisting users to drive safely.

1. Introduction
In recent years, with the acceleration of urbanization, the number of residents' car ownership shows a rapid growth mode [6]. At the same time, road safety problems are gradually exposed in the public's view. Due to road infrastructure and drivers and other reasons, the problem of traffic jam is becoming increasingly serious [8]. In order to solve the above problems, the National Transportation Department vigorously develops the policies of road construction and vehicle restrictions to ease the traffic pressure. The Vehicle Ad-hoc Network (VANET) technology can be a good solution. VANET is to connect vehicles with everything, that is V2X (vehicle to everything) [5][7], including V2V, which refers to the communication between vehicles through the OBU terminal. The OBU terminal can obtain the speed, driving direction, acceleration and other data information of vehicles and also can transmit text, voice, video and other information between vehicles. It can help users to establish a mobile interactive platform, which can exchange information all the time. VANET system diagram is shown in figure 1. Therefore, accelerating the development of VANET technology plays an important role in China's economic and technological development.
In the development of Chinese VANET technology, due to late start, the research of automobile manufacturers is still in the primary stage. In the domestic market, many automobile manufacturers have launched their own on-board system products. BYD’s system is built based on the Android kernel and the ecology of Android mobile phones can also be completely transplanted to the Di platform. With the rapid development of foreign markets, the car intelligent system, Car-play, released by Apple company, users can communicate with the car through Siri, interact through voice and operate through touch screen. The purpose is to let users enjoy the driving experience better. The On-Start system of General Motors mainly adopts wireless technology and global positioning system (GPS) satellite to provide perfect service to the automobile. But these system functions work independently, without creating a car-to-car connection.

Meanwhile, Android operating system has become the most popular operating system in the world. In recent years, it has occupied more than 80% of the global market. Due to the open-source characteristics of Android operating system, it is widely supported by individual developers and companies. Google company provides a large number of APIs for developers to facilitate the development of applications [1][3]. Therefore, the OBU terminal uses Android system.

This paper designs and implements the OBU terminal application of VANET based on Android. OBU terminal is the entry for interaction with users. It handles the message set and displays the processing results on the application interface and displays it to the user, which plays an early warning role. The rest of paper is organized as follows. Section II shows the application platform of this design, including hardware devices and software programming environment. Section III is divided into two parts, which are interface UI design and application improvement design. According to the design requirements, the program code is written to realize. Section IV is to debug, improve and solve the problems encountered in the process of programming, handle the causes of the problems and ensure the smooth operation of the program. Section V shows the function test and analysis, the last section is the summary.

2. Application platform
The hardware platform required for this design is two ARM development boards, the operating systems of which are Linux and Android systems and the entire software system design is divided into two parts: UI design and function realization, the purpose is to realize the application under the Android system information interaction with Linux operating system to achieve the effect of interaction with users.
2.1. Hardware platform
The core of the hardware platform is ARM development board, which uses the i.MX 6 development board, equipped with four core and dominant frequency 1 GHz Cortex-A9 processor. The board has onboard 4G, GPS, Gigabit Ethernet, HDMI, WIFI Bluetooth, CAN bus and other interfaces, supports 4.3-inch, 5-inch, 7-inch screens and supports HDMI displays. The processor has superior performance and the processing speed can achieve the expected effect. It can run Android operating system smoothly and support HDMI external display. In conclusion, the ARM board meets the design requirements and can be used as the core processing unit of OBU terminal.

2.2. Software platform
This design uses two ARM development boards to use different operating systems. Android system is an open-source operating system based on the Linux kernel, mainly used in mobile device such as smart phones, tablets and ARM platforms. Like other operating systems, Android also uses a layered architecture design. From high to low are the system application layer (System Apps), the Java API framework layer (Java API Framework), the Android system runtime layer (including the Android Runtime, the native C/C++ library Native), the hardware abstraction layer (Hardware Abstraction Layer) and Linux Kernel [4].

2.3. Development environment
Android Studio is an Android integrated development tool launched by Google, based on IntelliJ Idea [2]. Similar to the Eclipse ADT, Android Studio provides integrated Android development tools for development and debugging. Compared with the previous Android development of Java, Android SDK and Eclipse, Android Studio has many advantages. It has a powerful UI editor and a complete plug-in management. You can download the required plug-ins directly in the plug-in management. It supports mainstream code management tools such as SVN and GITHUB. It can be imported directly and is convenient for developers to use. The IDE provides intelligent save, intelligent complement function which can greatly improve the code writing efficiency. This is the official Google specifically for Android application development tool, it is currently the only official Google recommendation and no longer support other IDEs. This design uses Android 4.4 version based on Linux kernel and Ubuntu system. Android application development uses Android Studio development environment, which is based on IntelliJ Idea's Android application integrated development environment, the early development of Android system using eclipse and ADT based on Eclipse to use, because of the slow start-up speed, large memory occupation and other shortcomings, Eclipse is gradually replaced by Android studio, which is supported by developers.

3. UI interface and function design
This design uses Android studio for design and development. The principle is that UI interface design and program logic are completely separated, which can be divided into two parts: UI interface design and function module design.

3.1. UI interface design
This project creates a new XML file, use the layout manager and native control provided by Android studio to make reasonable layout according to the function needs. The control is placed in the specified location in the container, such as Button, etc. At the same time, we can import picture resources into the project file, which can be used in XML files.

In addition to the design of the main interface layout file, we also need to design the interface layout of each function module, using Android fragment to design. Fragment is a kind of UI fragment that can be embedded in activity, which can make more reasonable and full use of screen space. Just like activity layout file, when you need to display fragment interface, you can switch the required fragment interface by calling the API function of fragment.
3.2. **OBU terminal function module design**

OBU terminal is the on-board unit of the VANET system, which directly interacts with users. According to the demand analysis, the functional module is divided into two parts: security application module and non-security application module. The system module diagram is shown in figure 2.

![OBU terminal System module diagram](image)

The security application function has realized two functions, which are giving way to special vehicles and fault parking. These two functions correspond to two function buttons on the interface. According to their own needs, users can click the button to operate, corresponding to two receive buttons, receive the messages sent by two send buttons and then judge the processing. The icon changes to remind users. The principle of these two functions is the same. Security application flow chart is shown in figure 3, the flow chart of special vehicle giving way function is taken as an example.

![Security application flow chart](image)

The design of non-security function module is determined according to the application requirements, including text, voice message, photo and short video, voice call and video call.
3.2.1. Text transmission
The user can input text in the text edit box, click the send button, start the sending thread and package the text. The thread uses UDP to send the data packet and the sent text will be displayed in the chat information display area of the interface. The chat record can be found by sliding up and down and the text information can be saved in SQLite database. The receiving thread monitors the port, receives the message, handle and judges the message type. If the received event text message is displayed in the display area, it is convenient for users to view.

3.2.2. Voice message
Voice message function first needs to record audio. According to the user's habits, the basic logic operation is to press the voice button for recording. After recording, we release the button to send. But if we want to cancel sending, slide left to cancel sending. We also can set the shortest and longest recording time. Similar to the text message, the sending thread is called to package the voice data and send it to the thread. The receiving thread will handle the received message and judge the message type. If it is a voice message, the message will be displayed in the chat information display. The user can click the voice message record to play the voice.

3.2.3. Photo and short video
The photos and short video are realized with the help of the third-party open-source package, which is designed and implemented in the mode of WeChat chat. Firstly, switching to the photo fragment and clicking the photo button, we can collect the picture frame. Then, starting the sending thread, inputting the picture frame and sending it out. As for shooting short video, long press the photo button to record the video. Meanwhile, we can set the shortest and longest recording time. After shooting the short video, click the send button to complete the process of recording and sending the short video. User can also choose to re shoot or exit. Click the short video data and short image data before sending and display the short image data before receiving.

3.2.4. Voice call
The voice call function first needs to send voice request packets and the other party must accept the voice request before the two-way voice call can be carried out. The user clicks the voice call button to send the UDP voice request packet. When the other party receives the packet, a dialog box pops up. The user can choose to reject or accept the packet. If the voice call request is accepted, a TCP connection is established to transmit voice data. If the voice call request is rejected, a reject packet is fed back to remind the user that the other party has rejected.

3.2.5. Video call
The implementation of video call is based on Netty framework. Before programming, the Netty package must be imported. Just like voice call, the first step is to send a video call request packet and wait for the other party to confirm its acceptance or rejection. The principle is similar to voice call. However, in the process of transmission, the video call uses the Netty framework. It only needs to input data, bind the port number, set the channel and buffer, then it can carry out efficient video stream data transmission. In order to ensure the timeliness of transmission, the data is encoded and compressed before transmission, the x264 technology is used for encoding and compression and the data is decoded and played after receiving.

4. Technical implementation and difficulty solution
According to the design of function module, the program is designed and implemented, but in the process of implementation, there are lots of technical difficulties, such as the realization of inter layer communication, thread opening, data storage, data transmission and processing. Through continuous
debugging of code and consulting official code data, the relevant functions are modified and implemented.

### 4.1. SQLite database
For the data storage of non-security applications, SQLite, the relational lightweight database of Android SDK, is adopted, which fully complies with the rules of SQL language, just like MySQL and other common databases. The received messages are stored in the database, which is convenient for users to query and display in the chat information display area. Users can achieve it through the drop-down box.

As shown in figure 4, the service layer accesses the database through the implementation class of Dao layer without modifying, adding or querying statements. We create a new class, ChatMessageDao.java, in Dao layer and write the method of adding information and querying. The service layer only needs to call the method of Dao layer class to add or query data, which makes the code logic clear and easy to modify.

![Figure 4. Working diagram of SQLite database](image)

### 4.2. Interface callback
In the VANET system, there is a certain call relationship between different modules of application function and the most common one is synchronous call. In the function method body, another function is called by the function name and the execution of the calling function body is continued after the execution. When programming to realize the function, we write the function method body. When it needs to be called, pass in the parameter or no parameter through the function name. However, such a call relationship is too coupling and it is a blocking call. Once an error occurs in a function body, the whole application will crash. The solution is the idea of call-back.

As shown in figure 5, we define the call-back interface, ReceiveMessageCallback.java, and interface method in the project file. The main thread is the call-back object. The receiving thread, ReceiveMessage.java, is called by parameter passing. The receiving thread, ReceiveMessage.java, is the controller object. After receiving the data, the controller object calls the interface method to call the data back to MainActivity.java.

![Figure 5. Schematic diagram of interface call-back mechanism](image)

### 4.3. Encapsulation, analysis and transmission of data
In the implementation of application function, it is necessary to send request packets and distinguish different message types. The receiver needs to handle the data after receiving the data, then judge the
type of the message and make different data processing. Therefore, this design uses JavaBean to achieve
message encapsulation, parsing and uses JSON data format for data transmission. The data format
definition of the security message is shown in figure 6.

| Flag | Type | Priority | Num | EgoVehicleId | Time | UserData | Datalength |
|------|------|----------|-----|--------------|------|----------|------------|

Figure 6. Data encapsulation format

Before sending data, the data is encapsulated into Java objects, but TCP transmission is in the form
of byte stream, so we need to use Java serialization and deserialization technology to optimize the
implementation of this function.

Java serialization refers to the process of converting Java objects into byte sequences. Deserialization
is the opposite process. Its most important role is to ensure the integrity and transitivity of objects when
transferring and saving objects. Objects are transformed into byte streams to facilitate transmission on
the network or saving to local files, while deserialization reconstructs objects and saves object States. It
is implemented by Fastjson launched by Alibaba company, which is an excellent JSON framework used
by many enterprises. It's very easy to use. We don't need to add other class libraries and just need to
import Fastjson dependency into the configuration file of the project. After the project is recompiled,
the jar package will be automatically downloaded without manual operation by the programmer.

4.4. Use and improvement of multithreading

According to the module description of the above function design, a large number of threads are needed
for the realization of function programming. However, the time and resource cost of opening and closing
threads are huge, even cause too many threads and the system memory consumption is finished. The
function of the non-security function module in this design needs to send data messages by sending
threads. The idea of thread pool is used to improve. Thread pool provides a means of resource
management and restriction, which provides better performance for performing a large number of
asynchronous tasks. Relevant parameters can be set to set the number of core threads and the maximum
number of threads.

Multithreading working diagram is shown in figure 7. In the main interface, users click the button to
trigger message sending, get thread from thread pool and call sending thread to send data. The receive
thread receives message, call data back to main thread and display data to users in the main interface
after processing.

Figure 7. Multithreading working diagram

5. Function test and analysis

The project is created and programmed under Android studio IDE, compiled and downloaded to ARM
development board for operation. The interface diagram can be seen on the display screen, which is
divided into three areas: chat information display area, security message display area and function
application display area. OBU terminal interface diagram is shown in figure 8.
The main function of chat information display area is to display the recording contents of non-safety messages. As shown in figure 9 is security application test, after data processing, the icon on the interface changes and flickers continuously to remind users.

Besides, communication records of other non-security applications, such as photo taking, recorded short video, voice and video call record, are displayed. The function application display area occupies most of the interface area, mainly used to display the camera picture, such as taking photos and short video. After the operation test of OBU terminal application, the function of safety application module has a good effect and can meet the basic needs of users. When the vehicle breaks down, the user can click the button to end messages to other users periodically. After receiving the message, it can be processed quickly and feed them back to users. The disadvantage is that only two functions are realized and the other three functions need to be implemented by algorithm.
Non-security application module can run normally. Text, voice message and picture transmission are simple and fast without processing. Once the message is received, the message is directly displayed in the chat interface area and users can view it manually. For the voice call and video transmission function, although there is delay in the test process, it can meet the requirements and ensure the voice call quality within the allowable range. The disadvantages are that the beauty is not good enough and the program needs to continue to be optimized to improve the application fluency. Finally, the functions need to be improved, such as system settings, file management and other applications.

6. Conclusion
This paper introduces the hardware and software development environment of OBU terminal and Android operating system. On this basis, it puts forward the design idea of terminal application function and the function effect it wants to achieve. Based on the original programming, it continuously debugs and improves the program, we use more convenient, fast and practical technology to improve the readability and logicality of the program. Finally, the function of the test result shows that each functional application can run well, it can be used as OBU terminal application.

References
[1] Almomani, M. A. Khayer, A. (2020) A Comprehensive Analysis of the Android Permissions System. IEEE Access, vol. 8: pp. 216671-216688.
[2] Android Studio. (2021) Meet Android Studio. https://developer.android.com/studio/intro.
[3] Bai et al, G. (2018) Towards Model Checking Android Applications. IEEE Transactions on Software Engineering, vol. 44: pp. 595-612.
[4] Android Studio. (2021) Android Developer Platform Technology: Platform Architecture. https://developer.android.google.cn/guide/platform.
[5] Wang et al, X. (2019) Optimizing Content Dissemination for Real-Time Traffic Management in Large-Scale Internet of Vehicle Systems. IEEE Transactions on Vehicular Technology, vol. 68: pp. 1093-1105.
[6] Wang, J. Jiang, C. Han, Z. Ren, Y. Hanzo, L. (2018) Internet of Vehicles: Sensing-Aided Transportation Information Collection and Diffusion. IEEE Transactions on Vehicular Technology, vol. 67: pp. 3813-3825.
[7] Backfrieder, C. Ostermayer, G. F. Mecklenbräüker, C. (2017) Increased Traffic Flow Through Node-Based Bottleneck Prediction and V2X Communication. IEEE Transactions on Intelligent Transportation Systems: pp. 349-363.
[8] Nishimura, E. Guo, H. (2020) Vehicle Arrangement Problem for an Automobile Carrier Terminal. In: 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). Singapore. pp. 761-765.