Multi-network Communication Gateway of IoT for Complex Environment

Jinhong Li[1], Xiaoyan Pang[1(✉)], and Zhou Hu[2]

[1] Northwestern Polytechnical University, Xi'an, China
xypang@nwpu.edu.cn
[2] China Electronic Technology Group Corporation No. 20 Institute, Xi'an, China

Abstract. In the paper a multi-network communication gateway of Internet of Things (IoT) is designed for complex environments with poor mobile network. This design is developed based on ARM processor and Android operating system, which can forward the data of IoT to the Internet through two mobile networks (from different mobile operators) and a satellite network. Our experimental tests show that the system runs stably and can forward data of IoT accurately and instantly. This design of gateway can be applied to special scenarios, such as the tunnel construction and oil exploration.

Keywords: IoT · Multi-network · Mobile network · Satellite network

1 Introduction

With the development of the Internet era of big data, the concept of the Internet of Things (IoT) has been proposed and has attracted much attention [1]. The IoT is defined as a network that according to a specific protocol connects any item with the Internet and exchanges information with each other to realize intelligent identification, location, tracking, monitoring and management. It is a network that extends and expands on the Internet [2].

In most cases, the data collected in IoT need to be transmitted through a gateway device to the server in the Internet [3]. Usually, the gateway only needs to connect to mobile network in many applications like the agricultural standardization production monitoring, smart city, medical management [4], etc. However, in some complex environments, for instance the tunnel construction [5] and oil exploration [6], the mobile infrastructure is not constructed well, and the mobile network even does not exist. Considering the IoT applications in these special scenarios, in this paper we design a multi-network communication gateway of IoT and examine its performance. This gateway is developed based on ARM processor and Android operating system, which can forward the data of IoT to the Internet through two mobile networks (from different mobile operators) and a satellite network. Our proposal thus can work efficiently in the scenarios with poor mobile network and be applied to the complex environments.

B. Li et al. (Eds.): IoTaaS 2018, LNICST 271, pp. 120–126, 2019.
https://doi.org/10.1007/978-3-030-14657-3_13

2 Gateway Design

The main framework of this communication gateway is divided into four layers: hardware layer, operation systems layer, driver layer and application layer (see Fig. 1).

The hardware layer mainly includes Ethernet, two mobile modules, one satellite module and each hardware module is connected to the ARM mini PC. The operating system layer adopts the Android operating system which is based on a modified version of the Linux kernel and other open source software and designed initially for mobile devices. Hence this operating system can support ARM processor, flexible driver development and easily system tailoring. The driver layer is mainly responsible for driving the interface of each network hardware. The driver of the mobile network module is mainly related to Radio interface layer (RIL), which is mainly divided into two parts, RILJ and RILC. The RILJ runs in the framework layer, and the RILC runs in the hardware abstraction layer. The interaction between RILJ, RILC and Modem is based on the data interaction mode of network socket. The satellite module is connected to the ARM as a serial port device and is controlled by AT command. The Application layer is the place where the main functions are implemented by the communication gateway of IoT. It mainly includes: a data encryption, a SIM card status detection, a mobile network switching algorithm, and a satellite communication switching algorithm.

The configuring modules: mobile network driver, mobile switching algorithm, data forwarding configuration and data encryption are crucial parts in this gateway and in the following we will discuss these parts explicitly.

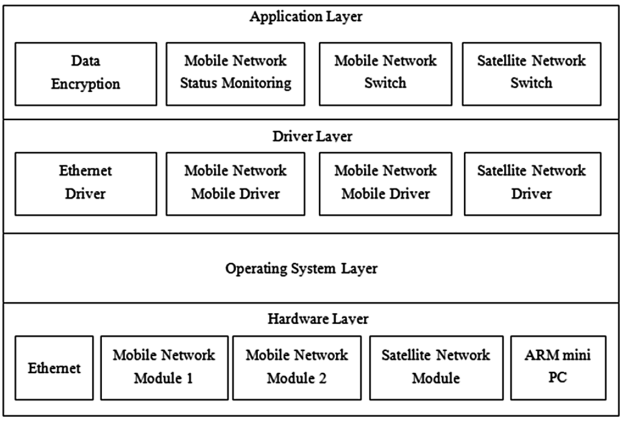

Fig. 1. System block diagram

2.1 Mobile Network Driver

Here the version of the both two mobile modules used in this multi network communication gateway is EC20. The mobile module is connected to the ARM mini PC via a Mini-PCIE interface which is essentially a USB interface, thus the 4G module can be

connected to ARM as a USB device. Based on USB to serial port drive, each mobile device is mapped to several virtual serial port devices, and the Android/Linux kernel sends AT commands and receive/send data to the mobile module through these virtual serial port devices.

In order to develop the mobile network driver, first we need to add the 'USB to serial port driver' and the 'PPP protocol support' into the Linux kernel, then also add the mobile module ID in the USB to serial port driver, as is shown in Fig. 2.

```
Linux Kernel Configure by "make menuconfig"
1.1 USB Driver Configure
Device Drivers  --->
   [*] USB support  --->
         <*> USB Serial Converter support
              [*] USB Generic Serail Driver
              [*] USB Driver for GSM and CMDA modems

EC20's Vender ID and Product ID in option_ids[] of drivers/usb/serial/option.c
       static const struct usb_device_id option_ids[] = {
                        { USB_DEVICE(0x05C6,0x9215) },   //for EC20

PPP Configure
       Device Drivers  --->
       [*] Network device support  --->
           <*> PPP (point- to- point protocol) support
           <*> PPP support for async serial ports
           <*> PPP support for sync tty ports
           <*> PPP Deflate compression
```

Fig. 2. Mobile module driver configuration

Secondly, we modify and add the source code provided by EC20 to the Android source code, and modify the Android launch configuration file to start the RILD service in, as is shown in Fig. 3. From this figure, it can be seen that the two mobile networks are set as two individual services and two different ttyUSB ports are assigned to these services, which is the basis for achieving the Dual SIM Dual Standby (DSDS).

```
service ril-daemon /system/bin/rild -l /system/lib/libreference-ril.so -c 0 -- -d /dev/ttyUSB2
class main
socket rild stream 660 root radio
socket rild-debug stream 660 radio system
user root
group radio cache inet misc audio log

service ril-daemon2 /system/bin/rild -l /system/lib/libreference-ril2.so -c 2 -- -d /dev/ttyUSB7
class main
socket rild2 stream 660 root radio
socket rild-debug2 stream 660 radio system
user root
group radio cache inet misc audio log
```

Fig. 3. RILD service configuration

2.2 Mobile Switching Algorithm

Since these two mobile networks of the communication gateway come from two different mobile operators and in the complex environments their signal strengths and qualities are also quite different from each other, it needs to monitor the signals from these two mobile networks and select the better one for data forwarding.

The TelephonyManager.listen function of the Android operating system is used to monitor the signal strength of the default mobile network. When there are two mobile

networks, the 'subId' of mobile network has to be determined. However, the 'subId' of the PhoneStateListener class is a hidden parameter that cannot be set or got directly by the application programs. Therefore, we have to use the java "reflection" method to make the 'subId' be accessible, and then the signal strength of the both mobile network can be monitored. After that the average historical signal strength of the two mobile networks can be calculated according to the following equation,

$$\text{ave} = 1.0 \times sigsum/signum \tag{1}$$

here sigsum represents the sum of the signal strengths and signum is the number of times of recording the signal strength. Then the mobile network with better average historical signal strength in a time window is chosen as the main mobile communication network, and the other is used as a standby mobile network.

2.3 Data Forwarding Configuration

The data from IoT are usually sent to the gateway through Ethernet and will be forwarded to the server through mobile network or satellite network. However, when the amount of data is too large or the data packet is very big, using the traditional Data forwarding configuration will lead to losing some packets. In order to avoid this lost, here another method–using iptables to forward data directly from the driver layer is adopted. Iptables is a packet filtering management tool based on Netfilter architecture. The most important function is to build firewall or transparent proxy. To use iptables, we first change the firewall to a forwarding mode, and then set the relevant parameters according to its command rule, as is shown in Fig. 4.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -F
iptables -t nat -F
iptables -t mangle -F
iptables -X
iptables -t nat -X
iptables -t mangle -X
iptables -A INPUT -i eth0 -j ACCEPT
iptables -A INPUT -i ppp0 -j ACCEPT
iptables -A INPUT -i ppp1 -j ACCEPT
iptables -A OUTPUT -o eth0 -j ACCEPT
iptables -A OUTPUT -o ppp0 -j ACCEPT
iptables -A OUTPUT -o ppp1 -j ACCEPT
iptables -A FORWARD -i eth0 -j ACCEPT
iptables -A FORWARD -i ppp0 -j ACCEPT
iptables -A FORWARD -i ppp1 -j ACCEPT
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -A PREROUTING -p udp --dport PORT -i eth0 -j DNAT --to-destination REMOTE_MOBILE_IP:REMOTE_PORT
iptables -t nat -A POSTROUTING -d REMOTE_MOBILE_IP -j SNAT --to LOCAL_MOBILE_IP
```

Fig. 4. Iptables parameter settings

In Fig. 4, PORT is the port where the gateway receives data of IoT; REMOTE_MOBILE_IP and REMOTE_PORT are the IP address and port of the server, and LOCAL_MOBILE_IP is the port of the local mobile network. With these commands, iptables can transmit packets directly to the server through the driver layer.

2.4 Data Encryption

In the complex environment the data of IoT need to be encrypted. In the design of the gateway, we use end-to-end encryption, that is, the packet is automatically encrypted by the gateway, while it is decrypted at the receiving end. The encryption algorithm used here is RSA encryption algorithm which is the first algorithm that can be used both for data encryption and digital signature. RSA is a kind of asymmetric encryption algorithm with a changeable key length. The principle of key generation is as follows:

1. Randomly selecting two large prime numbers p and q, where p is not equal to q, then to calculate $N = p * q$;
2. Selecting a natural number e which is greater than 1 less than N, and e must be compatible with $(p - 1) * (q - 1)$.
3. Using the Euclid algorithm to calculate the decryption key d and also to make it satisfies $e * d = 1[\text{mod}(p - 1) * (q - 1)]$ (where n, d are also mutually prime).
4. Getting the public key which is (n, e), and the private key which is (n, d).

In the process of encryption (or decryption) of information, the message m (or c) which is going to be encrypted (or decrypted) is first divided into equal length data blocks before they are encrypted (or decrypted). The corresponding formula are shown as follows:

$$c = m^e (\text{mod}\, N) \tag{2}$$

$$m = c^d (\text{mod}\, N) \tag{3}$$

3 Results and Discussions

We implement the communication gateway of IoT in an ARM mini PC and test its system performance by using Android Studio software. In the following we show some main results from the debug window of the Android Studio.

3.1 Mobile Network Test Results

As shown in Fig. 5, after the communication gateway is powered and both the mobile modules work normally, the system can obtain the IP address of both mobile network (denoted as ppp0 and ppp1), and get the signal strength and the operator's information of the mobile network. This illuminates that the mobile networks of the communication gateway work well.

```
D/MainService: SIM card status changes, SIM card 1 is true, SIM card 2 is true
D/IPUtils: getppp0IPAddress() enter...
D/IPUtils: getppp0IPAddress() success 10.172.199.87
D/IPUtils: getppp1IPAddress() enter...
D/IPUtils: getppp1IPAddress() success 10.36.82.252
D/MainService: The 4G network IP address is normal, create a mobile network send thread for: 10.172.199.87
D/MainService: Started 4G network UDP thread
D/MainService: The signal strength of the SIM card changes: Card: SIM_CARD_2 Signal strength: 4
V/MainService: Mobile operator 1: ChinaMobile
V/MainService: Mobile operator 2: CHN-CT
V/MainService: Data network type 1:13
V/MainService: Data network type 2:13
```

Fig. 5. Mobile networks test results

3.2 Mobile Networks Switching Test Results

After the system is powered, the mobile network 1 (denoted by SIM card 1 in Fig. 6) is selected as the main mobile communication network, and the average historical signal strength of the two mobile networks is monitored at the same time. If the signal strength of the mobile network 2 (denoted by SIM card 2 in Fig. 6) is better than the he mobile network 1 (denoted by SIM card 1) in a time window, the mobile network 2 (denoted by SIM card 2) is automatically switched as the main mobile communication network. In addition, manual switching is also possible. The debugging information is shown in Fig. 6.

```
D/MainService: The current network is SIM card 1, signal strength: 2
D/MainService: SIM card 2 signal strength changes, signal strength: 4
D/MainService: Automatically switch to SIM card 2
D/MainService: Switched to SIM card 2 successfully
```

Fig. 6. Mobile networks switching test results

3.3 Data Forwarding Test Results

After the data forwarding configuration is done, the data packets from IoT in the Ethernet interface are directly forwarded to the server through mobile network or satellite network. In Fig. 7 the data forwarding process in the debug window of Android Studio is given.

```
D/EthUdpServerNio: Received data, length: 25
D/DataBroadcastReceiver: Data received from Ethernet, command word: text message
I/Mobile UDP client process: Send packets to the mobile server
```

Fig. 7. Data forwarding test results

3.4 Data Encryption Test Results

The data encrypting process is shown in Fig. 8, from which we can see that the original data is 'Message from IoT' and the encrypted data cannot be read directly.

```
I/Mobile UDP client process: Unencrypted data: Message from IoT
I/Mobile UDP client process: Encrypted data: Wad2qs[]syw
```

Fig. 8. Data encryption test results

4 Conclusion

For some complex scenarios of IoT applications lacking of good mobile networks, a multi-network communication gateway is proposed and tested. The gateway is developed based on ARM processor and Android operating system and it can forward the data of IoT to the Internet through two mobile networks (from different mobile operators) and one satellite network. The working principle and performance of four crucial configuring modules including the mobile network driver, the mobile switching algorithm, the data forwarding configuration and the data encryption in this gateway have been discussed separately in detail. Our results have proved that this gateway system runs steadily and can forward the data of IoT accurately and safely. The combination of the two mobile networks and one satellite network ensures the communication in a complex environment effectively.

References

1. Breur, T.: Big data and the internet of things. J. Mark. Analytics **3**(1), 1–4 (2015)
2. Al-Fuqaha, A., Guizani, M., Mohammadi, M.: Internet of Things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutorials **17**(4), 2347–2376 (2015)
3. Saxena, N., Roy, A., Sahu, B.J.R.: Efficient IoT gateway over 5G wireless: a new design with prototype and implementation results. IEEE Commun. Mag. **55**(2), 97–205 (2017)
4. Zanella, A., Bui, N., Castellani, A.: Intenet of things for smart cities. IEEE IoT J. **1**(1), 22–32 (2014)
5. Cheng, H., Wu, N., Lian, J.: The management and monitor system of tunnel construction based on internet of things. In: Wang, W. (ed.) Proceedings of the Second International Conference on Mechatronics and Automatic Control. LNEE, vol. 334, pp. 1019–1026. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-13707-0_112
6. Khan, W.Z., Aalsalem, M.Y., Khan, M.K., et al.: A reliable Internet of Things based architecture for oil and gas industry. In: International Conference on Advanced Communication Technology, pp. 705–710 (2017)