Fibocom
PERFECT WIRELESS EXPERIENCE

MTC_Dial-up Guide
Linux

V1.3

Disclaimer

Any actions you take while using this document are at your own risk. Fibocom will not be liable for any damages or losses of any nature under any circumstances. Due to product version update or other reasons, Fibocom reserves the rights to modify any information in this document at any time without prior notice and without any responsibility. Unless otherwise specified, all the statements, information and suggestions contained in the document do not constitute any explicit or implicit guarantee.

This document may contain third-party information, products, services, data or content (collectively known as "third-party content"). Fibocom does not control and assumes no responsibility for third-party content, including but not limited to its accuracy, compatibility, reliability, availability, legality, appropriateness, performance, non-infringement, and update status, unless otherwise expressly stated in this document. Mention or reference to any third-party content in this document does not constitute an endorsement or guarantee by Fibocom of the third-party content. If you need third-party permissions, you must obtain the third-party permissions in legal ways, unless otherwise expressly stated in this document.

Copyright Statement

Unless specifically authorized by Fibocom, the recipient of the document shall keep the received documents and information confidential and shall not use it for any purpose other than the implementation and development of this project. Without the prior written permission of the copyright holder, any company or individual is prohibited to excerpt, copy any part of or the entire document, or distribute the document in any form. For any violation of confidentiality obligations, unauthorized use or malicious use of the document and information in other illegal forms, Fibocom has the rights to pursue legal responsibility.

Trademark Statement

Fibocom The trademark is registered by Fibocom Wireless Inc.

Other trademarks, product names, service names, and company names appearing in this document are the properties of their respective owners.

Contact

Website: https://www.fibocom.com

Address: Floor 10-14, Block A, Building 6, Shenzhen International Innovation Valley, Dashiyi Road, Xili Community, Xili Street, Nanshan District, Shenzhen

Tel: 0755-26733555

Contents

Applicable Model

No.	Applicable Model	Description
1	NL66x Series	Qualcomm 9X07 platform
2	MG110 Series	Qualcomm 9X07 platform
3	MC116 Series	Qualcomm 9X07 platform
4	LC116 Series	Qualcomm 9X07 platform
5	MA510 series	Qualcomm 9205 platform
6	L61x&LC61x&LG61x&MC61x&MG61x series	UNISOC 8910 platform
7	MC66x&MG66x series	UNISOC 8850 platform
8	LE series	Eigencomm CAT 1 platform
9	FG132 Series	Qualcomm SDX35 platform
10	L716 Series	Sanechips platform
11	LQ2A0 Series	Qualcomm 9X07 platform

Change History

V1.3 (2025-03-24)	Add LQ2A0 Product
V1.2 (2024-09-28)	Modify the LE series Port Model Table Description, see section 2.1.4
V1.1 (2024-07-03)	FG132 series added GTUSBMODE 47/48/49 support
V1.0 (2024-05-26)	Initial version

1 Overview

This document mainly introduces how to modify Linux kernel code so that customer's systems (such as Android and Ubuntu) can properly load UNISOC and Qualcomm platform modules, as well as various dial-up methods of the module in the Linux system. This document is intended for developers of Linux drivers and applications. The supported systems are all based on the Linux kernel (hereinafter referred to as the Linux side). The drivers used by Fibocom modules on the Linux side are classified into two categories:

- Self-developed interfaces: Corresponding to the option kernel driver. These interfaces can be used normally only after the driver adaptation data of the Fibocom module is added to the driver.
- Universal interfaces: Including ECM and MBIM. For these interfaces, the Fibocom module directly adapts to the universal driver without modifying the code.

Kernel compilation configuration items should be set for the drivers of both self-developed interfaces and universal interfaces to ensure that the drivers are compiled into the kernel.

The following table describes the dial-up types supported by the Linux system on each platform.

Table 1. Dial-up types

	UNISOC	Qualcomm	Sanechips	Eigencomm
PPP	√	√	√	√
ECM	√	√	√	√
RNDIS	√	√	√	√
QMI_WWAN	--	√	--	--
MBIM	--	√	--	--
GobiNet	--	√	--	--

FG132 does not support PPP dial-up, nor does it support dial-up using AT$QCRMCALL on Linux host.

2 Integration of USB Serial Port

2.1 USB Port Information

The USB port of the module can include multiple USB interfaces, and each USB interface can implement different functions by loading different USB interface drivers. After the Linux system successfully loads the USB interface driver, it can generate device nodes that implement different module functions, such as Log interface, AT interface, and USB network adapter. The module supports multiple USB enumeration modes, which are controlled by the AT+GTUSBMODE command.

2.1.1 USB Port Information of UNISOC Platform

The specific USB port information is subject to the AT manual of each product. The following table describes commonly used USB port information of the L61x, LC61X, LG61X, MC61X, MG61X and other series modules on UNISOC platform.

Table 2. USB port information of L61x, LC61X, LG61X, MC61X, and MG61X series modules

GTUSBMODE: 31 (Default)		
Vendor ID: 0x1782 Product ID: 0x4d10		
Interface ID	Interface Name	Interface Function
0	modem	Modem Connector
1	NV	Device NV Interface
2	MOS	Device MOS Log Interface
3	DIAG	Device Modem Log Interface
4	LOG	Device Ap Log Interface
5	AT1 port	Device Application Interface
6	AT2 port	Device Application Interface
GTUSBMODE: 32		
Vendor ID: 0x1782 Product ID: 0x4d11		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2	modem	Modem Connector
3	NV	Device NV Interface
4	MOS	Device MOS Log Interface
5	DIAG	Device Modem Log Interface

6	LOG	Device Ap Log Interface
7	AT1 port	Device Application Interface
8	AT2 port	Device Application Interface
GTUSBMODE: 33		
Vendor ID: 0x1782 Product ID: 0x4d11		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	modem	Modem Connector
3	NV	Device NV Interface
4	MOS	Device MOS Log Interface
5	DIAG	Device Modem Log Interface
6	LOG	Device Ap Log Interface
7	AT1 port	Device Application Interface
8	AT2 port	Device Application Interface

The following table describes commonly used USB port information of the MC66X, MG66X and other series modules on UNISOC platform.

Table 3. USB port information of the MC66X and MG66X series modules

GTUSBMODE: 73 (Default)		
Vendor ID: 0x2cb7 Product ID: 0x0a0a		
Interface ID	Interface Name	Interface Function
0	modem	Modem Connector
1	AP Log port	Device Ap Log Interface
2	CP Log port	Device Cp Log Interface
3	AT1 port	Device Application Interface
4	AT2 port	Device Application Interface
GTUSBMODE: 74		
Vendor ID: 0x2cb7 Product ID: 0x0a0b		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2	modem	Modem Connector

3	AP Log port	Device Ap Log Interface
4	CP Log port	Device Cp Log Interface
5	AT1 port	Device Application Interface
6	AT2 port	Device Application Interface
GTUSBMODE: 75		
Vendor ID: 0x2cb7 Product ID: 0x0a0c		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	modem	Modem Connector
3	AP Log port	Device Ap Log Interface
4	CP Log port	Device Cp Log Interface
5	AT1 port	Device Application Interface
6	AT2 port	Device Application Interface

2.1.2 USB Port Information of Qualcomm Platform

The specific USB port information is subject to the AT manual of each product. The following table describes commonly used USB port information of the NL668, MG110, MC116, FG132, MA510 and other series modules.

Table 4. USB port information of the NL668, MG110, MC116 and LC116 series modules

GTUSBMODE: 17		
Vendor ID: 0x1508 Product ID: 0x1001		
Interface ID	Interface Name	Interface Function
0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	AT	Device Application Interface
3	Pipe	Device Pipe
4	RmNet	Wireless Data Device Ethernet Adapter
5	ADB	Android Composite ADB Interface
GTUSBMODE: 18		
Vendor ID: 0x1508 Product ID: 0x1001		
Interface ID	Interface Name	Interface Function

0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	AT	Device Application Interface
3	Pipe	Device Pipe
4	ECM	ECM communication class interface
5	ECM	ECM data class interface
6	ADB	Android Composite ADB Interface
GTUSBMODE: 19		
Vendor ID: 0x05C6 Product ID: 0x9025		
Interface ID	Interface Name	Interface Function
0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	Pipe	Device Pipe
3	RmNet	Wireless Data Device Ethernet Adapter
4	Mystorage	Device Storage Interface
5	ADB	Android Composite ADB Interface
GTUSBMODE: 20		
Vendor ID: 0x1508 Product ID: 0x1000		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
GTUSBMODE: 21		
Vendor ID: 0x1508 Product ID: 0x1000		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
1	AT	Device Application Interface
GTUSBMODE: 22		
Vendor ID: 0x1508 Product ID: 0x1000		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
1	AT	Device Application Interface
2	RmNet	Wireless Data Device Ethernet Adapter

GTUSBMODE: 23		
Vendor ID: 0x1508 Product ID: 0x1000		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
1	AT	Device Application Interface
2	ECM	ECM communication class interface
3	ECM	ECM data class interface
GTUSBMODE: 24		
Vendor ID: 0x05C6 Product ID: 0x90B6		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	Modem	Modem Connector
3	DIAG	Device diagnostic interface
4	ADB	Android Composite ADB Interface
GTUSBMODE: 25		
Vendor ID: 0x1508 Product ID: 0x1001		
Interface ID	Interface Name	Interface Function
0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	AT	Device Application Interface
3	Pipe	Device Pipe
4	RmNet	Wireless Data Device Ethernet Adapter
GTUSBMODE: 26		
Vendor ID: 0x1508 Product ID: 0x1001		
Interface ID	Interface Name	Interface Function
0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	AT	Device Application Interface
3	Pipe	Device Pipe
4	ECM	ECM communication class interface

5	ECM	ECM data class interface
GTUSBMODE: 27		
Vendor ID: 0x1508 Product ID: 0x1001		
Interface ID	Interface Name	Interface Function
0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	AT	Device Application Interface
3	Pipe	Device Pipe
4	ECM	ECM communication class interface
5	ECM	ECM data class interface
6	ADB	Android Composite ADB Interface
7	AUDIO	UAC Interface
GTUSBMODE: 29		
Vendor ID: 0x2CB7 Product ID: 0x0110		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	AT	Device Application Interface
3	DIAG	Device diagnostic interface
GTUSBMODE: 34		
Vendor ID: 0x2CB7 Product ID: 0x0110		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	AT	Device Application Interface
3	DIAG	Device diagnostic interface
4	AUDIO	UAC Interface
GTUSBMODE: 35		
Vendor ID: 0x2CB7 Product ID: 0x0110		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface

1	MBIM	MBIM data class interface
2	AT	Device Application Interface
3	DIAG	Device diagnostic interface
4	AUDIO	UAC Interface
5	ADB	Android Composite ADB Interface

When ECM/RNDIS/MBIM and ADB interfaces load system drivers, it is not required to modify files on the host side.

Table 5. USB port information of FG132 series modules

GTUSBMODE: 20		
Vendor ID: 0x2CB7 Product ID: 0x0107		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
GTUSBMODE: 21		
Vendor ID: 0x2CB7 Product ID: 0x0108		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
1	AT	Device Application Interface
GTUSBMODE: 22		
Vendor ID: 0x2CB7 Product ID: 0x0109		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
1	AT	Device Application Interface
2	RmNet	Wireless Data Device Ethernet Adapter
GTUSBMODE: 23		
Vendor ID: 0x2CB7 Product ID: 0x010A		
Interface ID	Interface Name	Interface Function
0	Modem	Modem Connector
1	AT	Device Application Interface
2	ECM	ECM communication class interface
3	ECM	ECM data class interface

GTUSBMODE: 24		
Vendor ID: 0x2CB7 Product ID: 0x010B		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	Modem	Modem Connector
3	Diag	Device diagnostic interface
4	ADB	Android Composite ADB Interface
GTUSBMODE: 29		
Vendor ID: 0x2CB7 Product ID: 0x0110		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	AT	Device Application Interface
3	DIAG	Device diagnostic interface
GTUSBMODE: 30		
Vendor ID: 0x2CB7 Product ID: 0x0111		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	Modem	Modem Connector
3	Diag	Device diagnostic interface
4	AT	Device Application Interface
GTUSBMODE: 36		
Vendor ID: 0x2CB7 Product ID: 0x0112		
Interface ID	Interface Name	Interface Function
0	RmNet	Wireless Data Device Ethernet Adapter
1	Diag	Device diagnostic interface
2	AT	Device Application Interface
3	NMEA (GNSS)	GNSS interfaces related to AT commands
GTUSBMODE: 37		

Vendor ID: 0x2CB7 Product ID: 0x0112		
Interface ID	Interface Name	Interface Function
0	RmNet	Wireless Data Device Ethernet Adapter
1	Diag	Device diagnostic interface
2	AT	Device Application Interface
3	NMEA (GNSS)	GNSS interfaces related to AT commands
4	ADB	Android Composite ADB Interface
GTUSBMODE: 38		
Vendor ID: 0x2CB7 Product ID: 0x0113		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2	Diag	Device diagnostic interface
3	AT	Device Application Interface
4	NMEA (GNSS)	GNSS interfaces related to AT commands
GTUSBMODE: 39		
Vendor ID: 0x2CB7 Product ID: 0x0113		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2	Diag	Device diagnostic interface
3	AT	Device Application Interface
4	NMEA (GNSS)	GNSS interfaces related to AT commands
5	ADB	Android Composite ADB Interface
GTUSBMODE: 40		
Vendor ID: 0x2CB7 Product ID: 0x0114		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	Diag	Device diagnostic interface
3	AT	Device Application Interface
4	NMEA (GNSS)	GNSS interfaces related to AT commands

GTUSBMODE: 41		
Vendor ID: 0x2CB7 Product ID: 0x0114		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	Diag	Device diagnostic interface
3	AT	Device Application Interface
4	NMEA (GNSS)	GNSS interfaces related to AT commands
5	ADB	Android Composite ADB Interface
GTUSBMODE: 42		
Vendor ID: 0x2CB7 Product ID: 0x0115		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	NMEA (GNSS)	GNSS interfaces related to AT commands
3	Diag	Device diagnostic interface
4	AT	Device Application Interface
GTUSBMODE: 43		
Vendor ID: 0x2CB7 Product ID: 0x0115		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	NMEA (GNSS)	GNSS interfaces related to AT commands
3	Diag	Device diagnostic interface
4	AT	Device Application Interface
5	ADB	Android Composite ADB Interface
GTUSBMODE: 47		
Vendor ID: 0x2CB7 Product ID: 0x0117		
Interface ID	Interface Name	Interface Function
0	RmNet	Wireless Data Device Ethernet Adapter
1	Diag	Device diagnostic interface
2	AT	Device Application Interface

3	NMEA (GNSS)	GNSS interfaces related to AT commands
4	DPL	Data Protocol Logging
GTUSBMODE: 48		
Vendor ID: 0x2CB7 Product ID: 0x0118		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2	Diag	Device diagnostic interface
3	AT	Device Application Interface
4	NMEA (GNSS)	GNSS interfaces related to AT commands
5	DPL	Data Protocol Logging
GTUSBMODE: 49		
Vendor ID: 0x2CB7 Product ID: 0x0119		
Interface ID	Interface Name	Interface Function
0	MBIM	MBIM communication class interface
1	MBIM	MBIM data class interface
2	NMEA (GNSS)	GNSS interfaces related to AT commands
3	Diag	Device diagnostic interface
4	AT	Device Application Interface
5	DPL	Data Protocol Logging

1. When ECM/RNDIS/MBIM and ADB interfaces load system drivers, it is not required to modify files on the host side.
2. Only USB composition containing DPL interfaces will include TCP/UDP logs in the log.

Table 6. USB port information of MA510 series modules

GTUSBMODE: 31 (Default)		
Vendor ID: 0x2CB7 Product ID: 0x0106		
Interface ID	Interface Name	Interface Function
0	DIAG	Device diagnostic interface
1	Modem	Modem Connector
2	AT	Device Application Interface
3	ECM	ECM communication class interface
4	ECM	ECM data class interface

GTUSBMODE: 32		
Vendor ID: 0x2CB7 Product ID: 0x010A		
Interface ID	Interface Name	Interface Function
0	Modem	Modem connector
1	AT	Device Application Interface
2	ECM	ECM communication class interface
3	ECM	ECM data class interface

2.1.3 USB Port Information of Sanechips Platform

The specific USB port information is subject to the AT manual of each product. The following table describes commonly used USB port information of the L716 and other series modules.

Table 7. USB port information of L716 series modules

GTUSBMODE: 10 (Default)		
Vendor ID: 0x2CB7 Product ID: 0x0001		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2	AT	Device Application Interface
3	Modem	Modem connector
4	AT	Device Application Interface
5	Log	Device Log Interface
6	ADB	Android Composite ADB Interface
GTUSBMODE: 11		
Vendor ID: 0x2CB7 Product ID: 0x0001		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2	AT	Device Application Interface
3	Modem	Modem connector
4	AT	Device Application Interface

5	Log	Device Log Interface
6	ADB	Android Composite ADB Interface

2.1.4 USB Port Information of Eigencomm Platform

Table 8. USB port information of the Yixin series modules

GTUSBMODE: 30		
Vendor ID: 0x2CB7 Product ID: 0x0D01		
Interface ID	Interface Name	Interface Function
0/1	AT	Device Application Interface
2/3	Log	Device Log Interface
4/5	AT	Device Application Interface
GTUSBMODE: 31		
Vendor ID: 0x2CB7 Product ID: 0x0D01		
Interface ID	Interface Name	Interface Function
0	RNDIS	RNDIS communication class interface
1	RNDIS	RNDIS data class interface
2/3	AT	Device Application Interface
4/5	Log	Device Log Interface
6/7	AT	Device Application Interface
GTUSBMODE: 32		
Vendor ID: 0x2CB7 Product ID: 0x0D01		
Interface ID	Interface Name	Interface Function
0	ECM	ECM communication class interface
1	ECM	ECM data class interface
2/3	AT	Device Application Interface
4/5	Log	Device Log Interface
6/7	AT	Device Application Interface

As a ttyACM interface, the serial port on Eigencomm platform does not need to load the option driver, but needs to integrate the ACM driver. For details, see section 2.3.

2.2 Modifying and Configuring the Linux Kernel Option Driver

Linux option driver supports specific USB mobile broadband modems and mobile devices. The option driver is part of the USB serial driver of the Linux kernel. Compared with the universal USB serial port driver, the option driver provides a set of control messages based on ACM standards, such as DTR, RI and other control messages.

Some Linux kernels have integrated the VID and PID of some Fibocom products. When a module is identified, devices such as /dev/ttyUSB0 and ttyUSB1 will be automatically identified.

For most Linux kernels, you need to modify the option.c driver according to the instructions in this section, add Fibocom PID to identify the port of the module, and enumerate the ttyUSB device in /dev/device.

2.2.1 Modifying the Linux Kernel Option Driver

Modify the serial port driver file drivers/usb/serial/option.c. Refer to the following modification methods according to the specific kernel version.

If the kernel version is earlier than 4.16.8, modify as follows:

```
static const struct option_blacklist_info fibocom_b456_blacklist = {
    .reserved = BIT(4) | BIT(5) | BIT(6),
};

static const struct option_blacklist_info fibocom_b34_blacklist = {
    .reserved = BIT(3) | BIT(4),
};

static const struct option_blacklist_info fibocom_b345_blacklist = {
    .reserved = BIT(3) | BIT(4)| BIT(5),
};

static const struct option_blacklist_info fibocom_b2_blacklist = {
    .reserved = BIT(2),
};

static const struct option_blacklist_info fibocom_b23_blacklist = {
    .reserved = BIT(2) | BIT(3),
};
```

```
static const struct option_blacklist_info fibocom_b014_blacklist = {
    .reserved = BIT(0) | BIT(1) | BIT(4),
};

static const struct option_blacklist_info fibocom_b01_blacklist = {
    .reserved = BIT(0) | BIT(1),
};

static const struct option_blacklist_info fibocom_b04_blacklist = {
    .reserved = BIT(0) | BIT(4),
};

static const struct option_blacklist_info fibocom_b015_blacklist = {
    .reserved = BIT(0) | BIT(1) | BIT(5),
};

static const struct option_blacklist_info fibocom_b016_blacklist = {
    .reserved = BIT(0) | BIT(1) | BIT(6),
};

static const struct option_blacklist_info fibocom_b014567_blacklist = {
    .reserved = BIT(0) | BIT(1) | BIT(4) | BIT(5) | BIT(6) | BIT(7),
};

static const struct option_blacklist_info fibocom_b456789_blacklist = {
    .reserved = BIT(4) | BIT(5) | BIT(6) | BIT(7) | BIT(8) | BIT(9),
};
```

Add the PID/VID information of the module to option_ids[] and add the corresponding blacklist. When loading the driver, the system skips the interfaces specified by the blacklist. For details, refer to the following code.

```
static const struct usb_device_id option_ids[] = {
...
/*start add by fibocom*/
    { USB_DEVICE(0x1782, 0x4d10) },  /* L61x&LC61x&LG61x&MC61x&MG61x USBMODE = 31 */
    { USB_DEVICE(0x1782, 0x4d11),/* L61x&LC61x&LG61x&MC61x&MG61x USBMODE = 32 33 */
      .driver_info = (kernel_ulong_t)&fibocom_b01_blacklist },
    { USB_DEVICE(0x2CB7, 0x0A0A) },  /* MC66X&MG66X USBMODE = 73 */
    { USB_DEVICE(0x2CB7, 0x0A0B) ,   /* MC66X&MG66X USBMODE = 74 */
```

```
    .driver_info = (kernel_ulong_t)&fibocom_b01_blacklist },
    { USB_DEVICE(0x2CB7, 0x0A0C) ,   /* MC66X&MG66X USBMODE = 75 */
    .driver_info = (kernel_ulong_t)&fibocom_b01_blacklist },
    { USB_DEVICE(0x1508, 0x1001),/* NL668 USBMODE = 17 18 25 26 27 */
    .driver_info = (kernel_ulong_t)&fibocom_b456789_blacklist },
    { USB_DEVICE(0x05C6, 0x9025) , /*NL668 USBMODE = 19*/
    .driver_info = (kernel_ulong_t)&fibocom_b345_blacklist },
    { USB_DEVICE(0x1508, 0x1000),/* NL668 USBMODE = 20 21 22 23 */
    .driver_info = (kernel_ulong_t)&fibocom_b23_blacklist },
    { USB_DEVICE(0x05C6, 0x90B6) ,   /*NL668 USBMODE = 24*/
    .driver_info = (kernel_ulong_t)&fibocom_b014_blacklist },
    { USB_DEVICE(0x2CB7, 0x0110),/* NL668 FG132 USBMODE = 29 */
    .driver_info = (kernel_ulong_t)&fibocom_b014567_blacklist },
    { USB_DEVICE(0x2CA3, 0x4009),/* NL668 USBMODE = 30 31 */
    .driver_info = (kernel_ulong_t)&fibocom_b456_blacklist },
    { USB_DEVICE(0x2CB7, 0x0107) },  /* FG132 USBMODE = 20 */
    { USB_DEVICE(0x2CB7, 0x0108) },  /* FG132 USBMODE = 21 */
    { USB_DEVICE(0x2CB7, 0x0109) ,   /*FG132 USBMODE = 22*/
    .driver_info = (kernel_ulong_t)&fibocom_b2_blacklist },
    { USB_DEVICE(0x2CB7, 0x010A) ,   /*FG132 USBMODE = 23*/
    .driver_info = (kernel_ulong_t)&fibocom_b23_blacklist},
    { USB_DEVICE(0x2CB7, 0x010B) ,  /*FG132 USBMODE = 24*/
    .driver_info = (kernel_ulong_t)&fibocom_b014_blacklist },
    { USB_DEVICE(0x2CB7, 0x0111) ,  /*FG132 USBMODE = 30*/
    .driver_info = (kernel_ulong_t)&fibocom_b01_blacklist},
    { USB_DEVICE(0x2CB7, 0x0112), /* FG132 USBMODE = 36 37 */
    .driver_info = (kernel_ulong_t)&fibocom_b04_blacklist },
    { USB_DEVICE(0x2CB7, 0x0113), /* FG132 USBMODE = 38 39 */
    .driver_info = (kernel_ulong_t)&fibocom_b015_blacklist },
    { USB_DEVICE(0x2CB7, 0x0114), /* FG132 USBMODE = 40 41 */
    .driver_info = (kernel_ulong_t)&fibocom_b015_blacklist },
    { USB_DEVICE(0x2CB7, 0x0115), /* FG132 USBMODE = 42 43 */
    .driver_info = (kernel_ulong_t)&fibocom_b015_blacklist },
    { USB_DEVICE(0x2CB7, 0x0117),    /* FG132 USBMODE = 47 */
    .driver_info = (kernel_ulong_t)&fibocom_b04_blacklist },
    { USB_DEVICE(0x2CB7, 0x0118),    /* FG132 USBMODE = 48 */
    .driver_info = (kernel_ulong_t)&fibocom_b015_blacklist },
    { USB_DEVICE(0x2CB7, 0x0119),    /* FG132 USBMODE = 49 */
```

```
    .driver_info = (kernel_ulong_t)&fibocom_b015_blacklist },
    { USB_DEVICE(0x2CB7, 0x0106) ,  /*MA510 USBMODE = 31*/
    .driver_info = (kernel_ulong_t)&fibocom_b34_blacklist },
    { USB_DEVICE(0x05C6, 0x9008) },  /* qualcomm download mode */
    { USB_DEVICE(0x05C6, 0x900E) },  /* qualcomm crash mode */
    { USB_DEVICE(0x2CB7, 0x0001), /* L716 USBMODE = 10 11 */
    .driver_info = (kernel_ulong_t)&fibocom_b016_blacklist },
    { USB_DEVICE(0x19d2, 0x0256) },  /* L716 download mode */
    { USB_DEVICE(0x19d2, 0x0197) },  /* L716 crash mode */
    /* end add by fibocom */
{}
};
```

If the kernel version is 4.16.8 or later, modify as follows:

```
static const struct usb_device_id option_ids[] = {
...
    /* start add by fibocom */
    { USB_DEVICE(0x1782, 0x4d10) },  /* L61x&LC61x&LG61x&MC61x&MG61x USBMODE = 31 */
    { USB_DEVICE(0x1782, 0x4d11),
    .driver_info = RSVD(0) | RSVD(1) },  /* L61x&LC61x&LG61x&MC61x&MG61x USBMODE = 32
33 */
    { USB_DEVICE(0x2CB7, 0x0A0A) },  /* MC66X&MG66X USBMODE = 73 */
    { USB_DEVICE(0x2CB7, 0x0A0B),        /* MC66X&MG66X USBMODE = 74 */
    .driver_info = RSVD(0) | RSVD(1) },
    { USB_DEVICE(0x2CB7, 0x0A0C),    /* MC66X&MG66X USBMODE = 75 */
    .driver_info = RSVD(0) | RSVD(1) },
    { USB_DEVICE(0x1508, 0x1001), /* NL668 USBMODE = 17 18 25 26 27 */
    .driver_info = RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7) | RSVD(8) | RSVD(9)},
    { USB_DEVICE(0x05C6, 0x9025) ,  /*NL668 USBMODE = 19*/
    .driver_info = RSVD(3) | RSVD(4) | RSVD(5)},
    { USB_DEVICE(0x1508, 0x1000),/* NL668 USBMODE = 20 21 22 23 */
    .driver_info = RSVD(2) | RSVD(3)},
    { USB_DEVICE(0x05C6, 0x90B6) , /*NL668 USBMODE = 24*/
    .driver_info = RSVD(0) | RSVD(1) | RSVD(4)},
    { USB_DEVICE(0x2CB7, 0x0110) ,  /*NL668 USBMODE = 29*/
    .driver_info = RSVD(0) | RSVD(1) | RSVD(4) | RSVD(5) | RSVD(6) | RSVD(7)},
    { USB_DEVICE(0x2CA3, 0x4009),/* NL668 USBMODE = 30 31 */
```

```
    .driver_info = RSVD(4) | RSVD(5) | RSVD(6)},
    { USB_DEVICE(0x2CB7, 0x0107) },  /* FG132 USBMODE = 20 */
    { USB_DEVICE(0x2CB7, 0x0108) },  /* FG132 USBMODE = 21 */
    { USB_DEVICE(0x2CB7, 0x0109) ,   /*FG132 USBMODE = 22*/
    .driver_info = RSVD(2) },
    { USB_DEVICE(0x2CB7, 0x010A) ,   /*FG132 USBMODE = 23*/
    .driver_info = RSVD(2) | RSVD(3) },
    { USB_DEVICE(0x2CB7, 0x010B) ,    /*FG132 USBMODE = 24*/
    .driver_info = RSVD(0) | RSVD(1) | RSVD(4)},
    { USB_DEVICE(0x2CB7, 0x0111) ,    /*FG132 USBMODE = 30*/
    .driver_info = RSVD(0) | RSVD(1) },
    { USB_DEVICE(0x2CB7, 0x0112),    /* FG132 USBMODE = 36 37 */
    .driver_info = RSVD(0) | RSVD(4) },
    { USB_DEVICE(0x2CB7, 0x0113),   /* FG132 USBMODE = 38 39 */
    .driver_info = RSVD(0) | RSVD(1) | RSVD(5)},
    { USB_DEVICE(0x2CB7, 0x0114),  /* FG132 USBMODE = 40 41 */
    .driver_info = RSVD(0) | RSVD(1) | RSVD(5)},
    { USB_DEVICE(0x2CB7, 0x0115),  /* FG132 USBMODE = 42 43 */
    .driver_info = RSVD(0) | RSVD(1) | RSVD(5)},
    { USB_DEVICE(0x2CB7, 0x0117),  /* FG132 USBMODE = 47 */
    .driver_info = RSVD(0) | RSVD(4)},
    { USB_DEVICE(0x2CB7, 0x0118),  /* FG132 USBMODE = 48 */
    .driver_info = RSVD(0) | RSVD(1) | RSVD(5)},
    { USB_DEVICE(0x2CB7, 0x0119),  /* FG132 USBMODE = 49 */
    .driver_info = RSVD(0) | RSVD(1) | RSVD(5)},
    { USB_DEVICE(0x2CB7, 0x0106) ,  /*MA510 USBMODE = 31*/
    .driver_info = RSVD(3) | RSVD(4) },
    { USB_DEVICE(0x05C6, 0x9008) },  /* qualcomm download mode */
    { USB_DEVICE(0x05C6, 0x900E) },  /* qualcomm crash mode */
    { USB_DEVICE(0x2CB7, 0x0001),/* L716 USBMODE = 10 11 */
    .driver_info = RSVD(0) | RSVD(1) | RSVD(6)},
    { USB_DEVICE(0x19d2, 0x0256) },  /* L716 download mode */
    { USB_DEVICE(0x19d2, 0x0197) },  /* L716 crash mode */
    /* end add by fibocom */
{}
};
```

2.2.2 Configuring the Linux Kernel Option Driver

1. Run the **cd kernel** command to enter the root directory of the kernel.
2. Run the **make menuconfig** command.
3. On the pop-up window, select the following components in sequence:

```
-> Device Drivers

   -> USB support

      -> USB Serial Converter support

         [*]USB driver for GSM and CDMA modems
```

4. Select the "USB driver for GSM and CDMA modems" component as shown in the figure below, save the settings and exit the window.

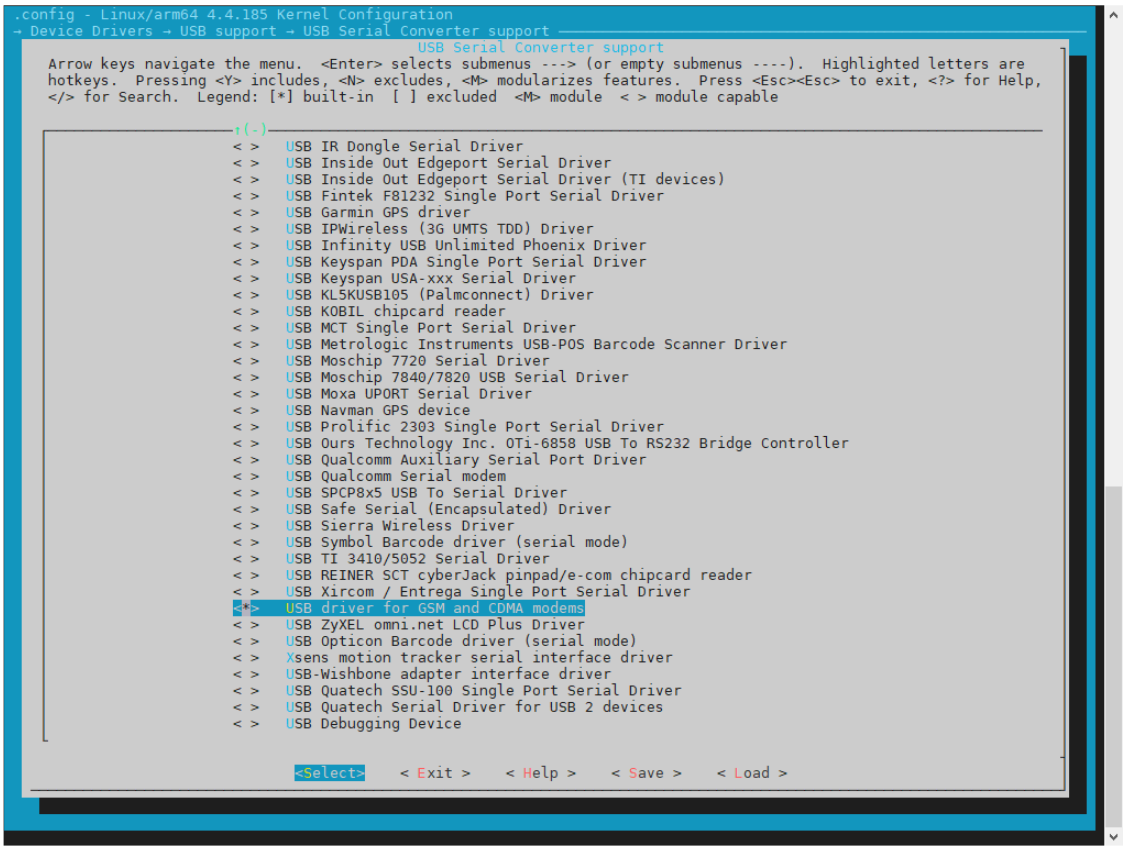

Figure 1. CONFIG_USB_SERIAL_OPTION configuration

2.3 Configuring the Linux Kernel ACM Driver

The Linux Abstract Control Model (ACM) driver supports universal serial USB devices such as modems, data terminal devices, and virtual serial port devices. The ACM driver allows the Linux system to identify and communicate with these devices, enabling serial data transfer and control. When the ACM driver identifies a port, the module will enumerate the port as /dev/ttyACM*.

1. Run the cd kernel command to enter the root directory of the kernel.
2. Run the make menuconfig command.
3. On the pop-up window, select the following components in sequence:

```
-> Device Drivers
    -> USB support
        [*]USB Modem (CDC ACM) support
```

4. Select the "USB Modem (CDC ACM) support" component as shown in the figure below, save the settings and exit the window.

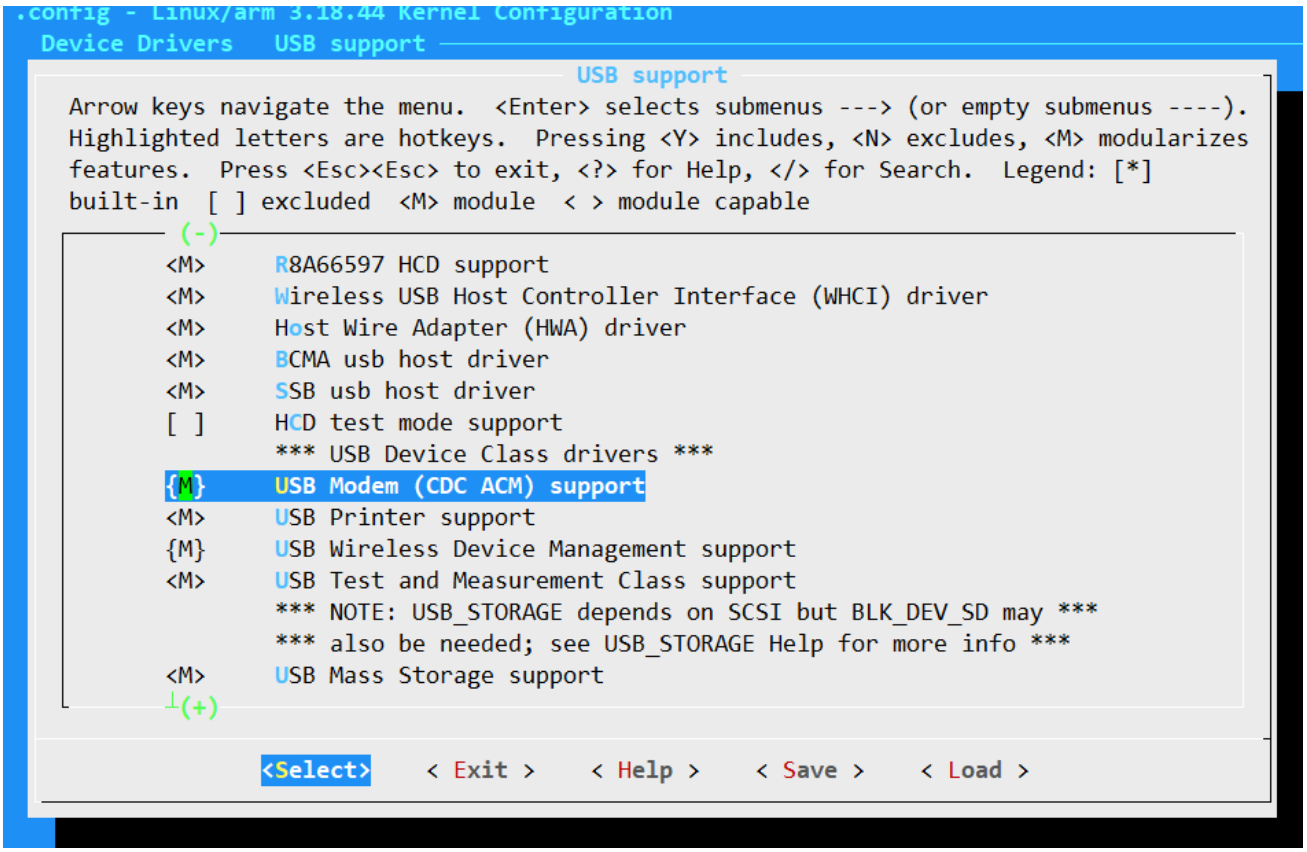

Figure 2. CONFIG_USB_SERIAL_ACM configuration

2.4 Adding the Zero-length Packet Mechanism

According to the requirements of the USB protocol, you need to add a zero-length packet processing mechanism for bulk transfer.

1. For the Linux kernel in version 2.6.35 or later, modify as follows in the drivers/usb/serial/usb_wwan.c file:

```
static struct urb *usb_wwan_setup_urb(struct usb_serial_port *port,
                                      int endpoint,
                                      int dir, void *ctx, char *buf, int len,
                                      void (*callback) (struct urb *))
{
```

```
    struct usb_serial *serial = port->serial;
    struct urb *urb;
    urb = usb_alloc_urb(0, GFP_KERNEL); /* No ISO */
    if (!urb)
        return NULL;
    usb_fill_bulk_urb(urb, serial->dev,
    usb_sndbulkpipe(serial->dev, endpoint) | dir,
    buf, len, callback, ctx);
    /*start add by fibocom*/
    if (dir == USB_DIR_OUT) {
        struct usb_device_descriptor *desc = &serial->dev->descriptor;
        if (desc->idVendor == cpu_to_le16(0x2cb7)
              || desc->idVendor == cpu_to_le16(0x1508))
            urb->transfer_flags |= URB_ZERO_PACKET;
    }
    /*end add by fibocom*/
    return urb;
}
```

2. For the Linux kernel in version 2.6.34 or earlier, modify as follows in the drivers/usb/serial/option.c file:

```
static struct urb *option_setup_urb(struct usb_serial *serial,
int endpoint,
int dir, void *ctx, char *buf, int len,
void (*callback)(struct urb *))
{
struct urb *urb;
if (endpoint == -1)
return NULL; /* endpoint not needed */
urb = usb_alloc_urb(0, GFP_KERNEL); /* No ISO */
if (urb == NULL) {
dbg("%s: alloc for endpoint %d failed.", __func__, endpoint);
return NULL;
}
/* Fill URB using supplied data. */
usb_fill_bulk_urb(urb, serial->dev,
usb_sndbulkpipe(serial->dev, endpoint) | dir,
buf, len, callback, ctx);
/*start add by fibocom*/
if (dir == USB_DIR_OUT) {
```

```
struct usb_device_descriptor *desc = &serial->dev->descriptor;
if (desc->idVendor == cpu_to_le16(0x1508)
|| desc->idVendor==cpu_to_le16(0x2CB7)
|| desc->idVendor==cpu_to_le16(0x2CA3)) {
urb->transfer_flags |= URB_ZERO_PACKET;
}
}
/*end add by fibocom*/
return urb;
}
```

2.5 Adding the reset-resume Mechanism

Some USB host controllers or USB hubs may lose power or reset when the SOC enters suspend/sleep mode.

In addition, the module cannot be restored after the SOC exits the suspend/sleep mode. In this case, you can add the following statements to enable the reset-resume mechanism.

Modify the code as follows:

```
static struct usb_serial_driver option_1port_device = {
    …
    .attach            = option_attach,
    .release           = option_release,
    .port_probe        = usb_wwan_port_probe,
    .port_remove      = usb_wwan_port_remove,
    .read_int_callback = option_instat_callback,
#ifdef CONFIG_PM
    .suspend           = usb_wwan_suspend,
    .resume            = usb_wwan_resume,
    .reset_resume      = usb_wwan_resume, //add by fibocom
#endif
};
```

If the host SOC does not sleep or USB selective suspend is not enabled on the USB bus, there is no need to modify the code as described in this section.

2.6 FAQs

2.6.1 How to Check Whether the USB Driver Exists in the Device

You can check whether the USB driver exists in the /sys/bus/usb/drivers directory. For example:

```
# ls /sys/bus/usb/drivers
cdc_acm  cdc_ether  cdc_mbim  cdc_ncm  cdc_wdm   ftdi_sio  hub  option    qmi_wwan
rndis_host  rtsx_usb  usb  usbfs  usbhid
```

If you need a USB-to-serial option driver, ensure that the option exists in the /sys/bus/usb/drivers directory.

Likewise, if you need the GobiNet driver, ensure that GobiNet exists. If you need the QMI_WWAN driver, ensure that qmi_wwan or qmi_wwan_f exists. The rest can be deducted by the same analog.

2.6.2 How to Check Whether the Module USB Is Connected to the Host Normally

When the module starts the USB and the host SOC is connected normally, the following log will be output:

```
root@ght-20H3A004CD:/home/ght# dmesg -c
[194684.154256] usb 1-2: new high-speed USB device number 92 using xhci_hcd
[194684.304560] usb 1-2: New USB device found, idVendor=2cb7, idProduct=0112, bcdDevice= 5.15
[194684.304585] usb 1-2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[194684.304597] usb 1-2: Product: Fibocom Module
[194684.304607] usb 1-2: Manufacturer: Fibocom Wireless Inc.
[194684.304616] usb 1-2: SerialNumber: 14dff31f
[194684.335848] qmi_wwan 1-2:1.0: cdc-wdm0: USB WDM device
[194684.336446] qmi_wwan 1-2:1.0 wwan0: register 'qmi_wwan' at usb-0000:00:14.0-2, WWAN/QMI device, 1a:4c:fd:13:eb:a5
[194684.337020] option 1-2:1.1: GSM modem (1-port) converter detected
[194684.337251] usb 1-2: GSM modem (1-port) converter now attached to ttyUSB0
[194684.337971] option 1-2:1.2: GSM modem (1-port) converter detected
[194684.338235] usb 1-2: GSM modem (1-port) converter now attached to ttyUSB1
[194684.338952] option 1-2:1.3: GSM modem (1-port) converter detected
[194684.339183] usb 1-2: GSM modem (1-port) converter now attached to ttyUSB2
```

Figure 3. FG132 USB enumeration log

If the above log does not exist, check whether the module is powered normally, whether there is a normal boot sequence, and whether VBUS is input to the module at a high level.

2.6.3 How to Check Installed USB Drivers

You can check the USB driver to which the Fibocom module's USB interface is connected. The name of the USB driver contains keyword "Driver=". If the system displays "Driver=none", the corresponding configuration item is not enabled in the kernel configuration, or the VID and PID of the Fibocom module are not added to the corresponding USB driver source file.

The USB interface and driver of FG132 module are as follows:

```
root@ght-20H3A004CD:/home/ght# cat /sys/kernel/debug/usb/devices

T:  Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#=  1 Spd=480  MxCh=12
B:  Alloc=  0/800 us ( 0%), #Int=  0, #Iso=  0
D:  Ver= 2.00 Cls=09(hub  ) Sub=00 Prot=01 MxPS=64 #Cfgs=  1
P:  Vendor=1d6b ProdID=0002 Rev= 6.02
S:  Manufacturer=Linux 6.2.0-39-generic xhci-hcd
S:  Product=xHCI Host Controller
S:  SerialNumber=0000:00:14.0
C:* #Ifs= 1 Cfg#= 1 Atr=e0 MxPwr=  0mA
I:* If#= 0 Alt= 0 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=00 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   4 Ivl=256ms

T:  Bus=01 Lev=01 Prnt=01 Port=01 Cnt=01 Dev#= 91 Spd=480  MxCh= 0
D:  Ver= 2.01 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=2cb7 ProdID=0112 Rev= 5.15
S:  Manufacturer=Fibocom Wireless Inc.
S:  Product=Fibocom Module
S:  SerialNumber=14dff31f
C:* #Ifs= 4 Cfg#= 1 Atr=a0 MxPwr=500mA
I:* If#= 0 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=50 Driver=qmi_wwan
E:  Ad=82(I) Atr=03(Int.) MxPS=   8 Ivl=32ms
E:  Ad=81(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=01(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 1 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=30 Driver=option
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=83(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=40 Driver=option
E:  Ad=85(I) Atr=03(Int.) MxPS=  10 Ivl=32ms
E:  Ad=84(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=03(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
I:* If#= 3 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
E:  Ad=86(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=04(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
```

Figure 4. FG132 interface and driver

3 ECM Dial-up

3.1 ECM Overview

The Ethernet Networking Control Model (ECM) is used to transfer Ethernet packets between the device and the host. For the operating system, the CDC_ECM device serves as a virtual Ethernet card that provides the MAC address and IP address required by a standard NIC. The CDC_ECM device is usually an Ethernet card for connecting to a LAN or WLAN. When a host initiates a request for ECM dial-up, the module (as a router) calls the corresponding service internally to implement WWAN dial-up. After the dial-up succeeds, the module internally starts the DHCP server and other functions. The client application system obtains an IP address assigned by the DHCP server of the module through the DHCP client service. The host calls DHCP and other scripts to configure the IP address and DNS obtained by the module from the network side to the local machine, so as to implement the Internet access.

3.2 Configuring the Linux Kernel ECM Driver

1. Run the **cd kernel** command to enter the root directory of the kernel.
2. Run the **make menuconfig** command.
3. On the pop-up window, select the following components in sequence:

```
-> Device Drivers
   -> Network device support
      -> USB Network Adapters
         ->Multi-purpose USB Networking Framework
            -*-     CDC Ethernet support (smart devices such as cable modems)
```

4. Select the "CDC Ethernet support (smart devices such as cable modems)" component as shown in the figure below, save the settings and exit the window.

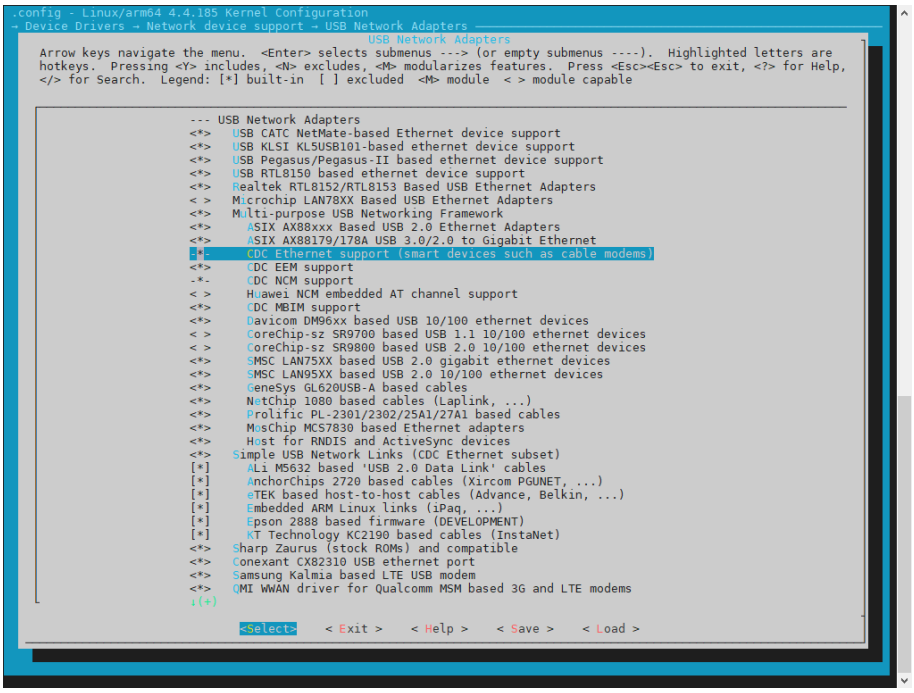

Figure 5. ECM kernel configuration

5. It is recommended to modify **strcpy (net->name, "eth%d")** in the **int usbnet_probe(struct usb_interface *udev, const struct usb_device_id *prod)** function under the **drivers/net/usb/usbnet.c** code, and change the NIC name from "eth*" to "usb*" to avoid the risk that both the network port enumerated by the module and the Ethernet port that may already exist in the Linux system are all "eth*" ports.
6. After completing the above configuration, run the **make** command to compile the modified kernel.

> In ECM mode, check whether the relevant code of Linux system has been modified by the customer. In particular, the following items should not be deleted from array products[] in **/kernel/drivers/net/usb/cdc_ether.c**:
>
> {USB_INTERFACE_INFO(USB_CLASS_COMM, USB_CDC_SUBCLASS_ETHERNET, USB_CDC_PROTO_NONE), .driver_info = (unsigned long) &cdc_info,}

3.3 Driver Loading Detection

After connecting the module on a Linux device configured with ECM, such as the NL668 module, send **AT+GTUSBMODE=18** to switch the USB port to the ECM mode and run the **modprobe cdc_ether** command to load the ECM driver. Alternatively, enter the **/lib/modules/`uname -r`/kernel/drivers/net/usb/** directory, find the **cdc_ether.ko** file and run **insmod cdc_ether.ko** to load the ECM driver.

Then, run **usb-devices** to view the information about the driver loaded on the port. As shown in the following figure, ports 4 and 5 are loaded with the cdc_ether driver.

```
If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=30 Driver=option
If#= 1 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=40 Driver=option
If#= 2 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=40 Driver=option
If#= 3 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=00 Prot=40 Driver=option
If#= 4 Alt= 0 #EPs= 1 Cls=02(commc) Sub=06 Prot=00 Driver=cdc_ether
If#= 5 Alt= 1 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=cdc_ether
If#= 6 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=42 Prot=01 Driver=(none)
```

Figure 6. ECM driver loading detection

After the ECM driver is successfully loaded, run the ifconfig -a command to view the NIC information of the module. If the driver is correctly loaded, the following information will be returned:

```
root@ubuntu:/lib/modules/4.15.0-142-generic/kernel/drivers/net/usb# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:cf:c4:03
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1020 (1.0 KB)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:4844 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4844 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:360104 (360.1 KB)  TX bytes:360104 (360.1 KB)

usb0      Link encap:Ethernet  HWaddr 9e:a5:ce:ed:ca:5a
          inet6 addr: fe80::c731:e615:8ebd:63e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Figure 7. NIC information detection

3.4 ECM Dial-up Process

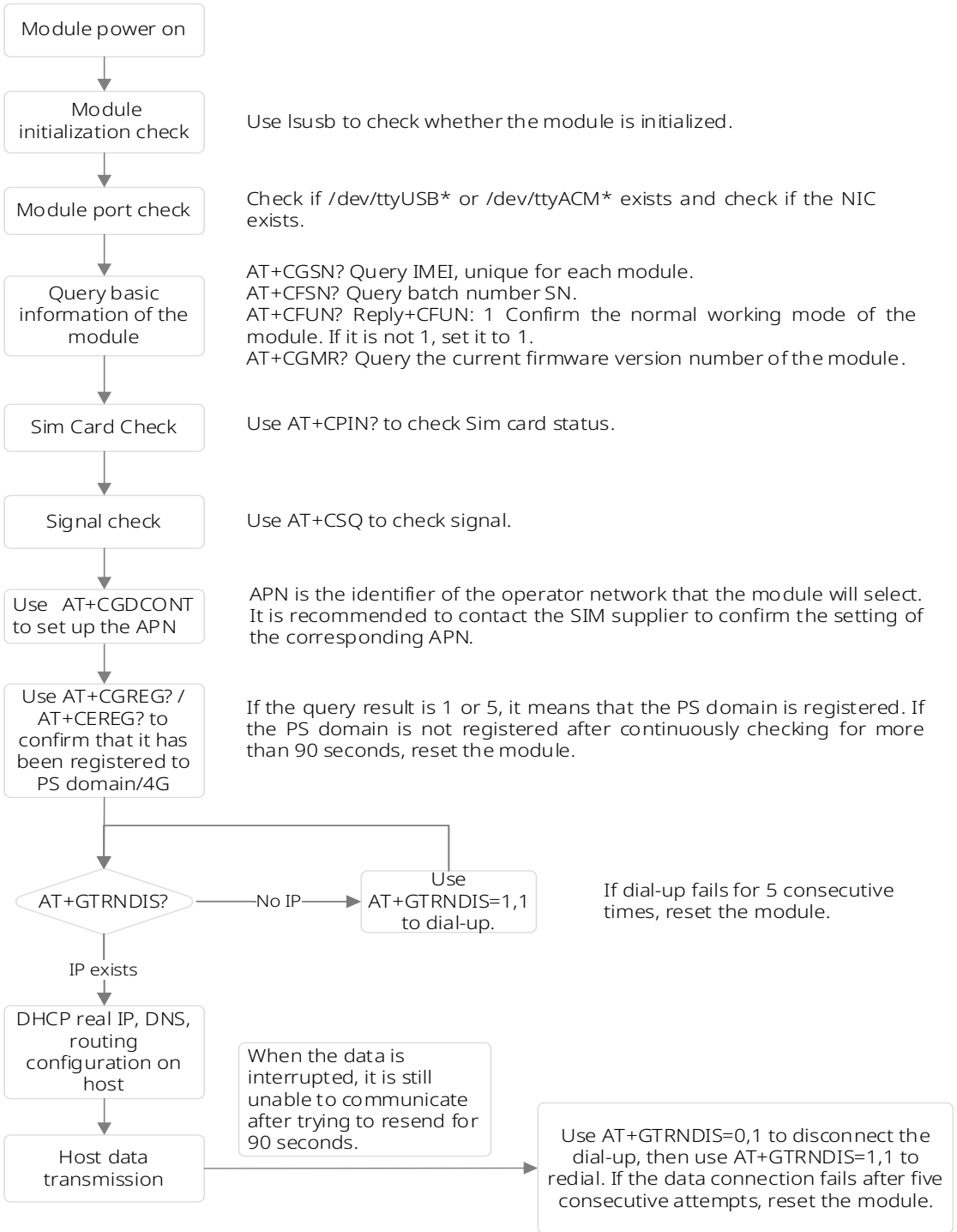

Figure 8. ECM dial-up flowchart

3.5 ECM Dial-up Example

This section describes the dial-up operation by taking the NL668 module and a SIM card as an example.

```
AT+GTUSBMODE?
+GTUSBMODE: 18          //Ensure that the USB port is in ECM mode. If not, run AT+GTUSBMODE=18 AT+CFUN=15.
OK
AT+CPIN?
+CPIN: READY                                                    //Ensure that the SIM card is ready.
OK
AT+CGDCONT=1,"IP","CTNET"                                      //Set the access point information.
OK
AT+CGDCONT?
+CGDCONT: 1,"IP","CTNET","0.0.0.0",0,0,0,0                       //Check whether the setting is successful.
OK
AT+COPS?
+COPS: 0,0,"CHN-CT",7                                          //Ensure that the module has been registered with the carrier’s network.
OK
AT+CGREG?
+CGREG: 0,1                                 //The module attaches to the data network.
OK
AT+CEREG?
+CEREG: 0,1                                                                 //The module attaches to the LTE network.
OK
AT+GTRNDIS=1,1
OK
AT+GTRNDIS?
+GTRNDIS: 1,1,"100.72.107.45","61.134.1.6","218.20.19.40"   //Query the module state and check whether it obtains an IP address.
OK
```

After OK is returned for the AT+GTRNDIS command, send AT+GTRNDIS? to query whether the dial-up is successful. If yes, the obtained IP address is returned.

After the dial-up, run ifconfig on Ubuntu16.04 Linux host to check whether the NIC has obtained the IP address and whether the data service can be used through this NIC. If yes, the dial-up is successful.

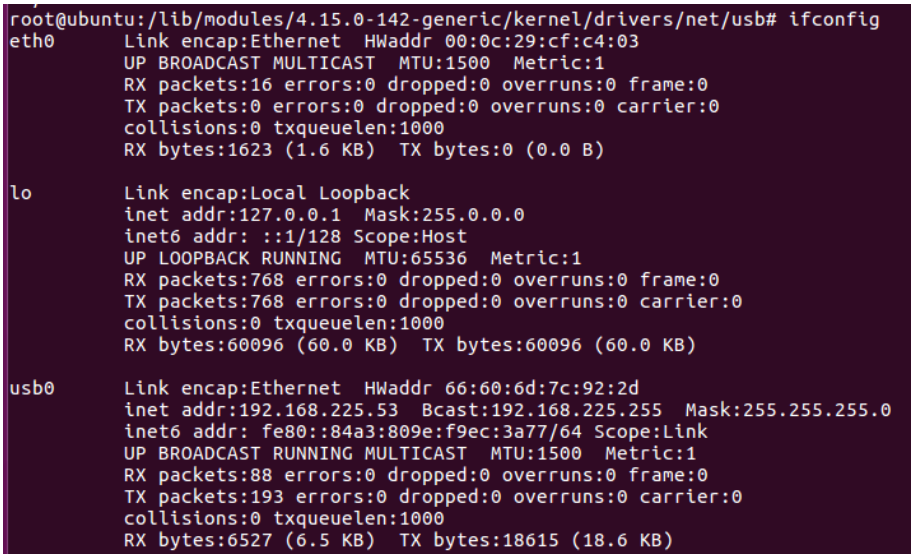

Figure 9. NIC state after ECM dial-up

After the dial-up is successful, run **ping www.baidu.com –I usb0** to send ping packets.

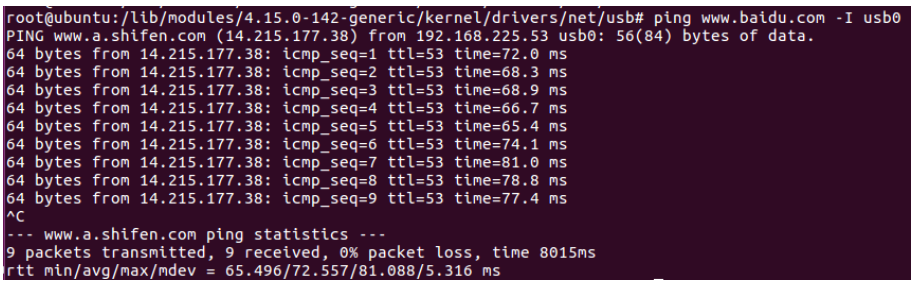

Figure 10. Sending ping packets after ECM dial-up

If you need to disconnect the network, run the following command:

```
AT+GTRNDIS=0,1
OK
AT+GTRNDIS?
+GTRNDIS: 0
OK
```

To disconnect dial-up, do not directly remove the USB cable, disconnect the power supply, or restart the module. Run **AT+GTRNDIS=0,1** in the serial port tool to disconnect the dial-up connection.

4 MBIM Dial-up

4.1 MBIM Overview

Mobile Broadband Interface Model (MBIM) is a new interface standard proposed by Intel, Microsoft and other USB/IF members as mobile broadband is widely used in mobile devices such as notebooks/Ultrabooks, Tablets, and Pads. You can download the specific standards from the official website of USB IF. It unifies the interface standards between mobile broadband devices (USB data cards/WAN cards, NGFF data cards, etc.) and PCs. Modem manufacturers do not need to provide drivers, as the driver function is already supported by the Win8 system. The Linux system supports the driver function in version 3.8.

4.2 Configuring the Linux Kernel MBIM Driver

1. Run the cd kernel command to enter the root directory of the kernel.
2. Run the make menuconfig command.
3. On the pop-up window, select the following components in sequence:

```
-> Device Drivers
   -> Network device support
      -> USB Network Adapters
         ->Multi-purpose USB Networking Framework
            <*>     CDC MBIM support
```

4. Select the "CDC MBIM support" component as shown in the figure below, save the settings and exit the window.

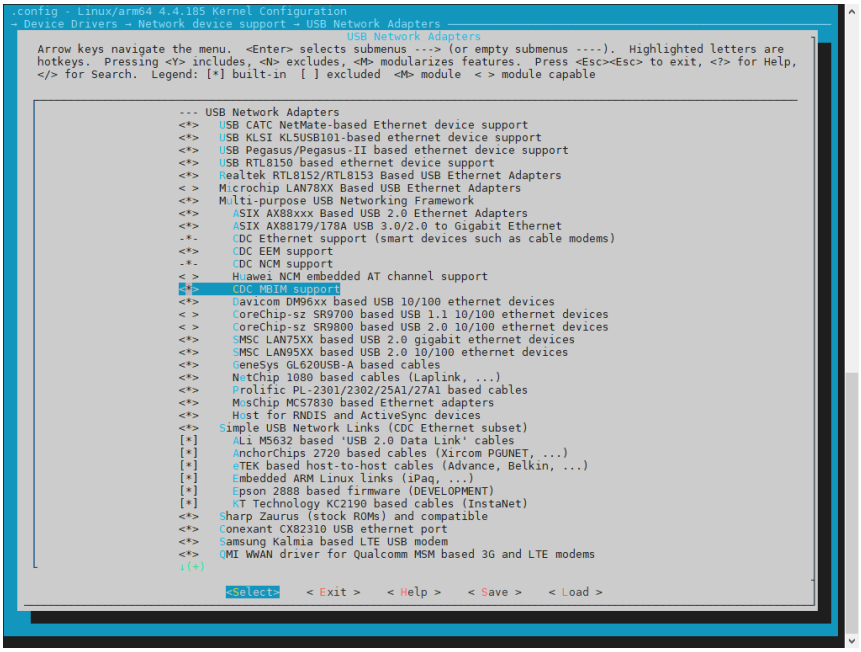

Figure 11. MBIM kernel configuration

4.3 Driver Loading Detection

1. After connecting the module on a Linux device configured with MBIM, such as the NL668 module, send **AT+GTUSBMODE=29** to switch the USB port to the MBIM mode and run the **modprobe cdc_mbim** command to load the MBIM driver. Alternatively, enter the **/lib/modules/`uname -r`/kernel/drivers/net/usb/** directory, find the **cdc_mbim.ko** file and run **insmod cdc_mbim.ko** to load the MBIM driver. Open the command line interface (CLI), and enter **lsmod | grep cdc_mbim** to query MBIM driver loading information. The MBIM driver is loaded successfully, as shown in the following figure.

```
root@ubuntu:~# lsmod | grep cdc_mbim
cdc_mbim               16384  0
cdc_wdm                20480  1 cdc_mbim
cdc_ncm                36864  1 cdc_mbim
usbnet                 45056  3 cdc_mbim,cdc_ncm,cdc_ether
```

Figure 12. Querying the loading of the MBIM driver

2. You can view the driver loading information of the port using the **usb-devices** command. As shown in the following figure, If #=0 and If #=1 have loaded the cdc_mbim drivers, among which one is the communication class interface of the MBIM driver and the other is the data class interface.

```
#Ifs= 5 Cfg#= 1 Atr=a0 MxPwr=500mA
If#= 0 Alt= 0 #EPs= 1 Cls=02(commc) Sub=0e Prot=00 Driver=cdc_mbim
If#= 1 Alt= 1 #EPs= 2 Cls=0a(data ) Sub=00 Prot=02 Driver=cdc_mbim
If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=40 Driver=option
If#= 3 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=30 Driver=option
If#= 4 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=40 Driver=option
```

Figure 13. MBIM driver loading detection

3. After the MBIM driver is successfully loaded, run the **ifconfig -a** command to view the NIC information of the module. If the driver is correctly loaded, the following information will be

returned.

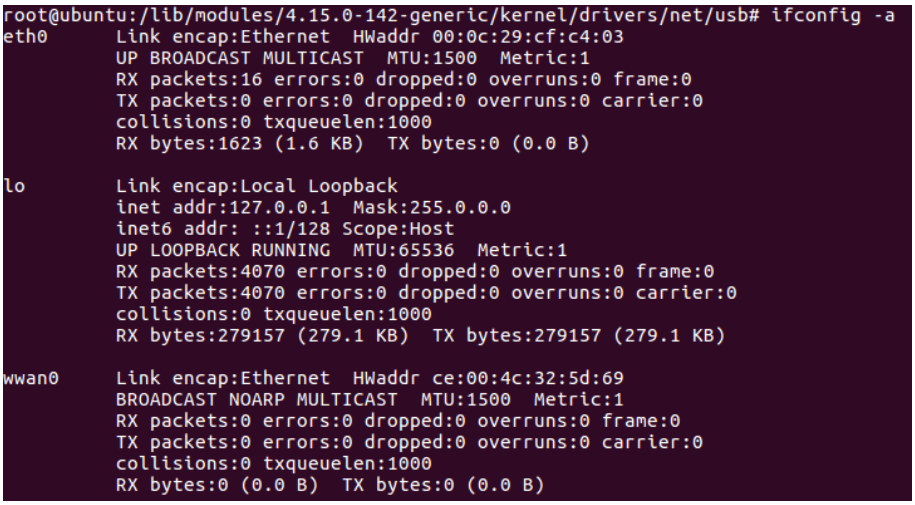

Figure 14. NIC information detection

4.4 Installing and Configuring the MBIM Dial-up Environment

Before the dial-up, you need to install the MBIM dependent environment.

Install the MBIM library.

```
apt install libmbim-utils
```

The correspondence between the mbimcli library version and the Ubuntu version is as follows:

- Ubuntu 16.04 : mbimcli 1.14
- Ubuntu 18.04 : mbimcli 1.18
- Ubuntu 20.04 : mbimcli 1.22

After the installation is completed, you can run the **mbimcli –help** command to view the MBIM commands supported by the current version. The MBIM dial-up process in this section is based on mbimcli version 1.14 of Ubuntu 16.04.

4.5 MBIM Single-channel Dial-up Process

1. Initialize the MBIM: mbimcli -p -d /dev/cdc-wdm0 --query-subscriber-ready-status.

```
root@ubuntu:/home/ght# mbimcli -p -d /dev/cdc-wdm0 --query-subscriber-ready-status
[/dev/cdc-wdm0] Subscriber ready status retrieved:
        Ready state: 'initialized'
        Subscriber ID: '460028091813606'
        SIM ICCID: '89860010261881586751'
        Ready info: 'none'
Telephone numbers: (0) 'unknown'
```

2. Perform MBIM dial-up that carries the APN information.

```
root@ubuntu:/home/ght# mbimcli -p -d /dev/cdc-wdm0 --connect=session-id=0,apn=3gnet
[/dev/cdc-wdm0] Successfully connected
[/dev/cdc-wdm0] Connection status:
                    Session ID: '0'
                    Activation state: 'activated'
                    Voice call state: 'none'
                    IP type: 'ipv4v6'
                    Context type: 'internet'
                    Network error: 'unknown'
 [/dev/cdc-wdm0] IPv4 configuration available: 'address, gateway, dns, mtu'
                     IP [0]: '172.19.42.119/28'
                     Gateway: '172.19.42.120'
                     DNS [0]: '211.137.130.18'
                     DNS [1]: '211.137.130.2'
                     MTU: '1500'
[/dev/cdc-wdm0] IPv6 configuration available: 'address, gateway, dns, mtu'
IP [0]: '2409:8970:1080:4e3a:68a1:a16a:1802:e564/64'
                     Gateway: '2409:8970:1080:4e3a:5808:d2b3:fbb8:8162'
                    DNS [0]: '2409:8070:2000:f110::1'
                    DNS [1]: '2409:8070:2000:f010::1'
                    MTU: '1500'
```

3. After the dial-up is completed, query the dial-up state and the obtained IP address, DNS information, and other information.

```
root@ubuntu:/home/ght# mbimcli -p -d /dev/cdc-wdm0 --query-connection-state
[/dev/cdc-wdm0] Connection status:
     Session ID: '0'
     Activation state: 'activated'
    Voice call state: 'none'
    IP type: 'ipv4v6'
    Context type: 'internet'
            Network error: 'unknown'
root@ubuntu:/home/ght# mbimcli -p -d /dev/cdc-wdm0 --query-ip-configuration
[/dev/cdc-wdm0] IPv4 configuration available: 'address, gateway, dns, mtu'
                  IP [0]: '172.19.42.119/28'
                 Gateway: '172.19.42.120'
                 DNS [0]: '211.137.130.18'
                 DNS [1]: '211.137.130.2'
                 MTU: '1500'
[/dev/cdc-wdm0] IPv6 configuration available: 'address, gateway, dns, mtu'
```

```
                IP [0]: '2409:8970:1080:4e3a:6daf:c536:d26a:c8ff/64'
                Gateway: '2409:8970:1080:4e3a:5808:d2b3:fbb8:8162'
                DNS [0]: '2409:8070:2000:f110::1'
                DNS [1]: '2409:8070:2000:f010::1'
                MTU: '1500'
```

4. After the dial-up is completed, the IP information obtained through the dial-up needs to be added to the MBIM NIC of Ubuntu before data services can be carried out. Please contact Fibocom technical personnel to obtain the corresponding mbim-set-ip.zip script.

After ensuring that the script has executable permissions, run the ls /sys/class/net command to check whether the NIC exists.

```
root@ubuntu:/home/ght# ./mbim-set-ip /dev/cdc-wdm0 wwan0
Requesting IPv4 and IPv6 information through mbimcli proxy:
[/dev/cdc-wdm0] IPv4 configuration available: 'address, gateway, dns, mtu'
                 IP [0]: '172.19.42.119/28'
                 Gateway: '172.19.42.120'
                 DNS [0]: '211.137.130.18'
                 DNS [1]: '211.137.130.2'
                 MTU: '1500'
[/dev/cdc-wdm0] IPv6 configuration available: 'address, gateway, dns, mtu'
                IP [0]: '2409:8970:1080:4e3a:e0d1:a4f1:d918:7f1/64'
                Gateway: '2409:8970:1080:4e3a:5808:d2b3:fbb8:8162'
                DNS [0]: '2409:8070:2000:f110::1'
                DNS [1]: '2409:8070:2000:f010::1'
                MTU: '1500'
Network interface configurations completed.
```

5. Run the ifconfig command to query the NIC information.

```
ght@ubuntu:~$ ifconfig
ens33: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:0c:29:fc:6f:5e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2791  bytes 200440 (200.4 KB)
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2791  bytes 200440 (200.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wwan0: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 172.19.42.119  netmask 255.255.255.240  broadcast 172.19.42.127
        inet6  2409:8970:1080:4e3a:248c:daff:fe34:8b75     prefixlen  64     scopeid
0x0<global>
        inet6  2409:8970:1080:4e3a:5d77:72e5:507b:78e     prefixlen  64     scopeid
0x0<global>
        inet6 fe80::248c:daff:fe34:8b75  prefixlen 64  scopeid 0x20<link>
        inet6  2409:8970:1080:4e3a:60ad:93b3:f22e:bb88     prefixlen  64     scopeid
0x0<global>
        ether 26:8c:da:34:8b:75  txqueuelen 1000  (Ethernet)
        RX packets 12  bytes 1581 (1.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 7418 (7.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The above information shows that the MBIM NIC of wwan0 has been added with the correct IP address, and data operations such as sending ping packets can be performed.

6. Disconnect the MBIM dial-up connection: mbimcli -d /dev/cdc-wdm0 --disconnect="0".

0 indicates the session-id and can be omitted when there is only id0.

7. Others related commands:

In addition to the most commonly used dial-up commands, MBIM also has other additional commands. You can use these commands to implement various functions such as querying module information, adding unlock PIN code, switching on/off the airplane mode, performing network authentication, and querying signals.

Run the **mbimcli --help** command to query the commands supported by the mbimcli version. An earlier version of mbimcli may not support all commands. This has nothing to do with the module. You can upgrade the mbimcli version to address this issue.

```
root@ubuntu:/home/ght# mbimcli --help
Usage:
  mbimcli [OPTION…] - Control MBIM devices

Help Options:
  -h, --help                                    Show help options
  --help-all                                    Show all help options
```

```
  --help-basic-connect                                Show Basic Connect Service
options
  --help-phonebook                                  Show Phonebook Service options
  --help-dss                                         Show Device Service Stream
options
  --help-ms-firmware-id                              Show Microsoft Firmware ID
Service options
  --help-ms-host-shutdown                           Show Microsoft Host Shutdown
Service options
  --help-atds                                       Show AT&T Device Service options
  --help-intel-firmware-update                       Show Intel Firmware Update
Service options
  --help-ms-basic-connect-extensions                Show Microsoft Basic Connect
Extensions Service options

Application Options:
  -d, --device=[PATH]                               Specify device path
  -p, --device-open-proxy                            Request to use the 'mbim-proxy'
proxy
  --no-open=[Transaction ID]                         Do not explicitly open the MBIM
device before running the command
  --no-close                                        Do not close the MBIM device
after running the command
  --noop                                            Don't run any command
  -v, --verbose                                     Run action with verbose logs,
including the debug ones
  --silent                                          Run action with no logs; not
even the error/warning ones
  -V, --version                                     Print version
```

4.6 MBIM Multi-PDN Dial-up Process

1. MBIM multi-PDN dial-up is based on single-channel dial-up. According to the steps in sections 4.2 and 4.3, configure the Linux kernel and load the cdc_mbim driver.
2. Establish MBIM multi-PDN dial-up links. Use different session-ids for multi-PDN dial-up. Multiple links need to use different APNs. For example, session-id=1 uses APN=ctnet, and session-id=2 uses APN=cmnet.

```
root@ubuntu:/home/zhang/Desktop/mbim_tool#   mbimcli   -p   -d   /dev/cdc-wdm0   --
connect=session-id=1,apn=ctnet
[/dev/cdc-wdm0] Successfully connected

[/dev/cdc-wdm0] Connection status:
```

```
          Session ID: '1'
    Activation state: 'activated'
    Voice call state: 'none'
             IP type: 'ipv4v6'
        Context type: 'internet'
       Network error: 'unknown'

[/dev/cdc-wdm0] IPv4 configuration available: 'address, gateway, dns, mtu'
     IP [0]: '10.40.78.250/30'
    Gateway: '10.40.78.249'
    DNS [0]: '211.137.130.2'
    DNS [1]: '211.137.130.18'
        MTU: '1500'

[/dev/cdc-wdm0] IPv6 configuration available: 'address, gateway, dns, mtu'
     IP [0]: '2409:8970:1124:2528:74a8:b6b9:23d9:d543/64'
    Gateway: '2409:8970:1124:2528:551d:4ecb:7c7b:ace0'
    DNS [0]: '2409:8070:2000:f110::1'
    DNS [1]: '2409:8070:2000:f010::1'
        MTU: '1500'
root@ubuntu:/home/zhang/Desktop/mbim_tool#    mbimcli    -p    -d    /dev/cdc-wdm0    --
connect=session-id=2,apn=3gnet
[/dev/cdc-wdm0] Successfully connected

[/dev/cdc-wdm0] Connection status:
          Session ID: '2'
    Activation state: 'activated'
    Voice call state: 'none'
             IP type: 'ipv4v6'
        Context type: 'internet'
       Network error: 'unknown'

[/dev/cdc-wdm0] IPv4 configuration available: 'address, gateway, dns, mtu'
     IP [0]: '10.9.37.142/30'
    Gateway: '10.9.37.141'
    DNS [0]: '211.137.130.2'
    DNS [1]: '211.137.130.18'
        MTU: '1500'
```

```
[/dev/cdc-wdm0] IPv6 configuration available: 'address, gateway, dns, mtu'
     IP [0]: '2409:8970:1114:3432:8497:1e6:c0e8:d40c/64'
    Gateway: '2409:8970:1114:3432:3134:1c14:705e:5847'
    DNS [0]: '2409:8070:2000:f110::1'
    DNS [1]: '2409:8070:2000:f010::1'
        MTU: '1500'
```

3. After multi-PDN dial-up is successfully implemented, you need to configure the obtained IP addresses for different NICs. You can use the same **mbim-set-ip** script described in section 4.5 to configure the IP addresses and routes. For multi-PDN dial-up, you can add a third parameter **SID="$3"** to the script to adapt to multiple dial-up links established with different session-ids, thereby adding the corresponding VLANs to different session-id links.

```
#CONTROL-IFACE
CONTROLDEV="$1"
#WWAN-IFACE
DEV="$2"
#session-id
SID="$3"

if [ -z "$SID" ];then
    SID="0"
    SESSIONID="0"
elif [ "${SID:0:11}" = "session-id=" ];then
    SESSIONID=${SID:11}
else
echo "operation failure session-id"
exit 0
fi


set_vlan()
{
    ip link add link $DEV name $DEV.$SESSIONID type vlan id $SESSIONID
}

if [ $SESSIONID -gt 0 ]
then
    set_vlan
    DEV=$DEV.$SESSIONID
fi

echo "Requesting IPv4 and IPv6 information through mbimcli proxy:"
mbimcli -d $CONTROLDEV -p --query-ip-configuration=$SESSIONID
IPDATA=$(mbimcli -d $CONTROLDEV -p --query-ip-configuration=$SESSIONID)
```

Figure 15. Modified mbim-set-ip script

The modified **mbim-set-ip** script is also applicable to single-channel dial-up, and the third parameter can be omitted.

4. You can run the following commands to configure IP addresses of NICs for multi-PDN dial-up.

```
./mbim-set-ip /dev/cdc-wdm0 wwan0 session-id=1
./mbim-set-ip /dev/cdc-wdm0 wwan0 session-id=2
```

5. After running the above commands, run **ifconfig –a** to query the NIC information. As shown in the following figure, the NICs wwan0.1 and wwan0.2 have obtained IP addresses.

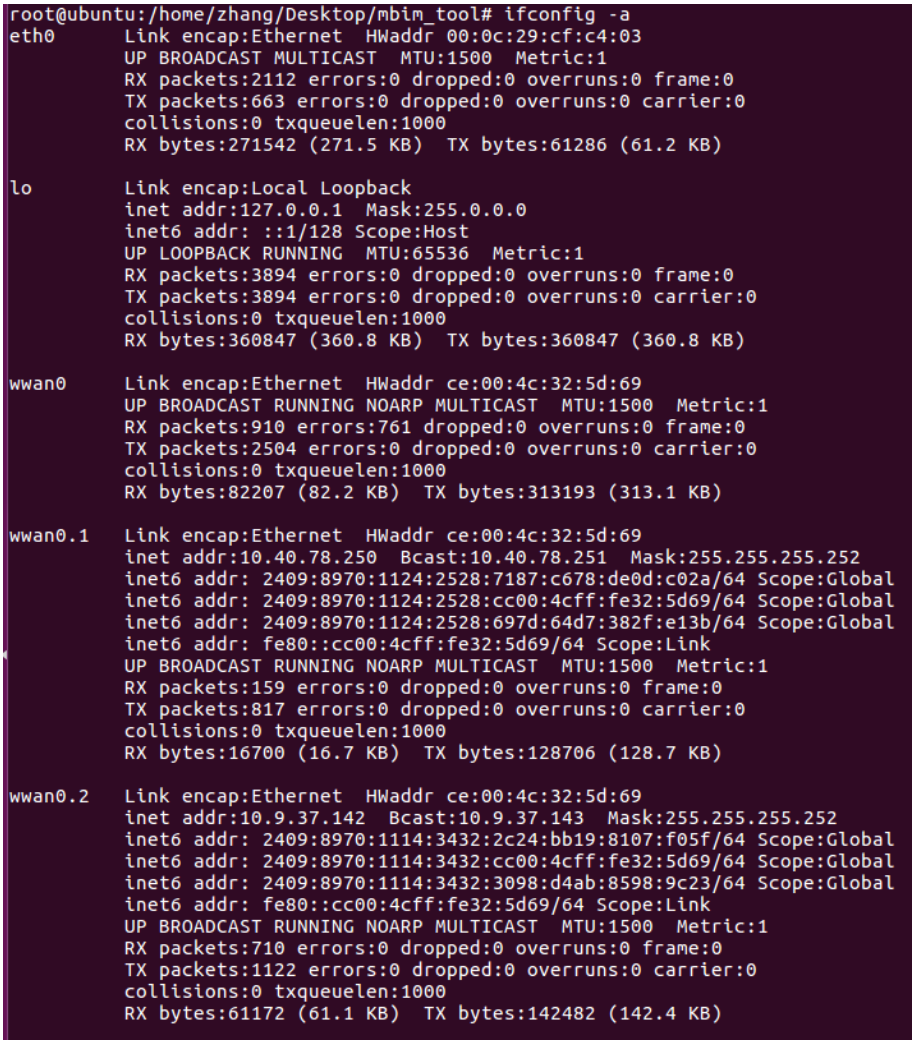

Figure 16. Querying the NIC information

6. After NICs wwan0.1 and wwan0.2 obtain the IP addresses, you can perform data operations such as sending ping packets.

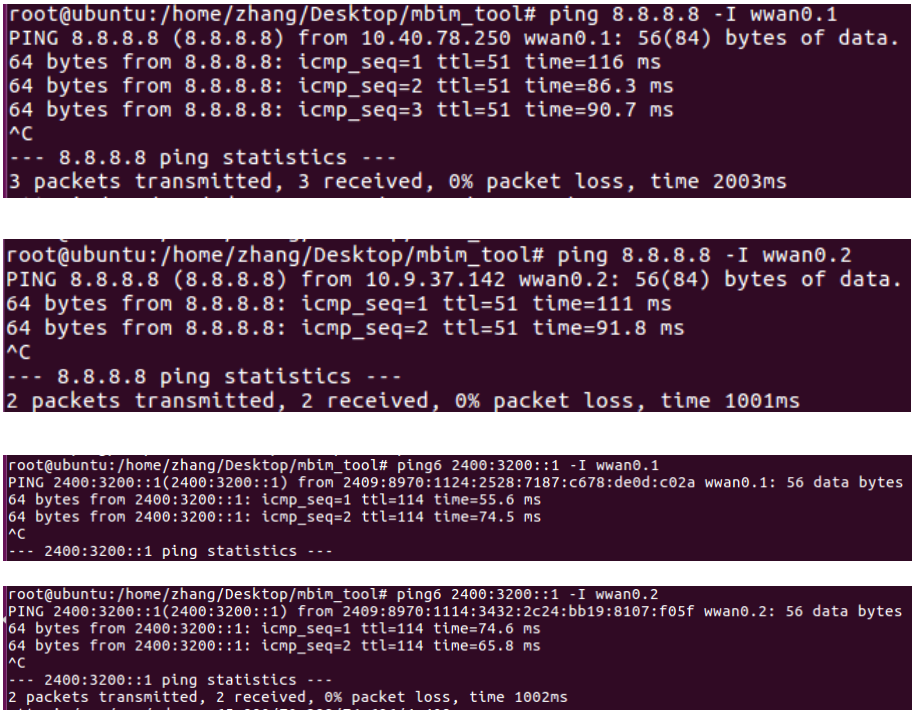

Figure 17. Ping packet sending test

4.7 FAQs

4.7.1 Failed to Send Ping Packets in Multi-PDN Dial-up

Run the following commands to disable reverse routing:

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/wwan0.1/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/wwan0.2/rp_filter
```

5 RNDIS Dial-up

5.1 RNDIS Overview

Remote Network Driver Interface Specification (RNDIS) is a bus-independent specification for Ethernet (802.3) network devices on dynamic Plug and Play (PnP) buses such as USB, 1394, Bluetooth, and InfiniBand. RNDIS defines a bus-independent messaging protocol between a host computer and RNDIS devices through abstract control and data channels.

5.2 Configuring the Linux Kernel RNDIS Driver

1. Run the **cd kernel** command to enter the root directory of the kernel.
2. Run the **make menuconfig** command.
3. On the pop-up window, select the following components in sequence:

```
-> Device Drivers
    -> Network device support
        -> USB Network Adapters
            ->Multi-purpose USB Networking Framework
                <*>     Host for RNDIS and ActiveSync devices
```

4. Select the "Host for RNDIS and ActiveSync devices" component as shown in the figure below, save the settings and exit the window.

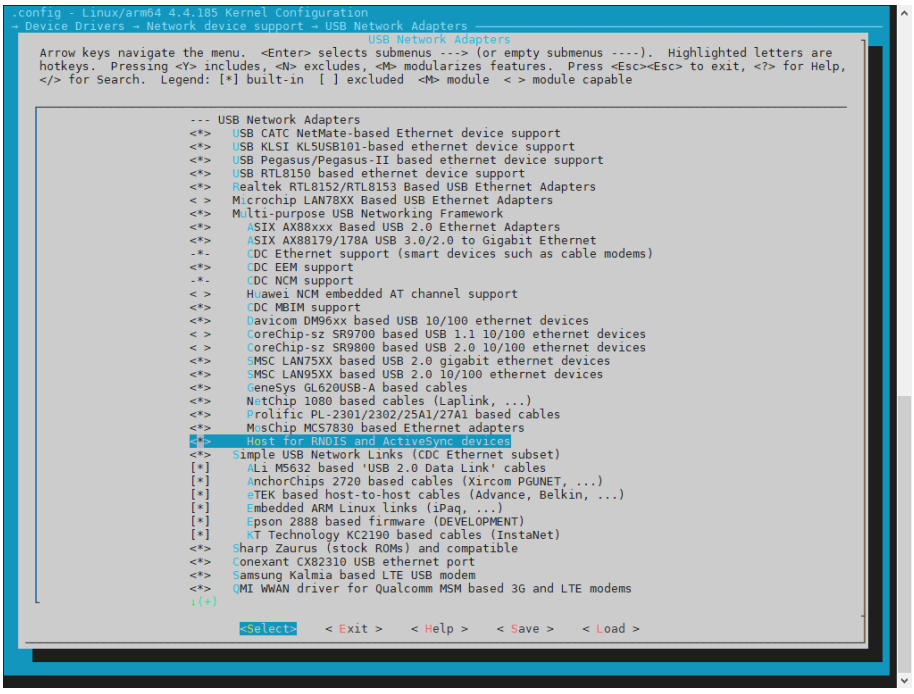

Figure 18. RNDIS kernel configuration

5.3 Driver Loading Detection

1. After connecting the module on a Linux device configured with RNIDS, such as the NL668 module, send **AT+GTUSBMODE=24** to switch the USB port to the RNDIS mode and run the **modprobe rnids_host** command to load the RNDIS driver. Alternatively, enter the **/lib/modules/`uname -r`/kernel/drivers/net/usb/** directory, find the **rndis_host.ko** file and run **insmod rndis_host.ko** to load the RNDIS driver.
2. Run the **usb-devices** command to view the information about the driver loaded on the port. As shown in the following figure, the rndis_host driver has been successfully loaded.

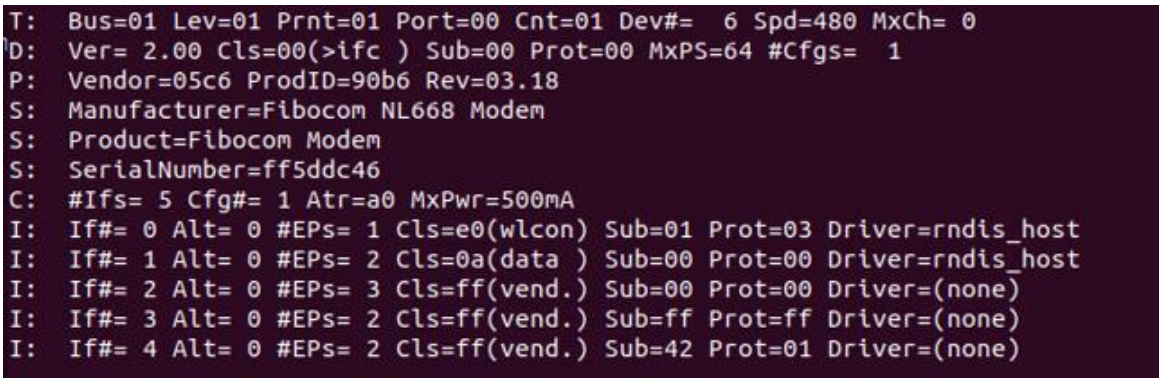

Figure 19. Driver loading detection

3. After the RNDIS driver is successfully loaded, run the **ifconfig -a** command to view the NIC information of the module. If the driver is correctly loaded, the following information will be returned:

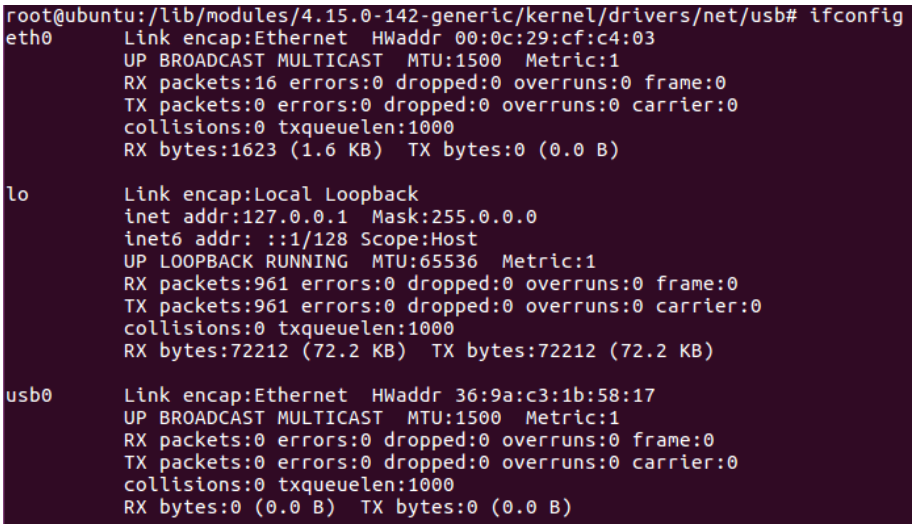

Figure 20. NIC information detection

5.4 RNDIS Dial-up Process

The RNDIS dial-up process and commands are the same as those of ECM dial-up process. You just need to switch the port to RNDIS mode. For example, switch the NL668 port to 24. For details, see section 3.4.

5.5 RNDIS Dial-up Example

The RNDIS dial-up process and commands are the same as those of ECM dial-up process. For details, see section 3.5.

6 GobiNet Dial-up

6.1 GobiNet Overview

The GobiNet driver is Qualcomm CDC/ECM NDIS NIC driver (also known as RmNet driver) for Qualcomm products. The RmNet interface evolves based on ECM on Qualcomm platform and also belongs to CDC-ECM. The specific difference between them lies in the encapsulation of USB commands and the different USB interface and endpoint definitions.

6.2 GobiNet Driver Integration

6.2.1 GobiNet Driver Integrated in Linux Kernel

GobiNet needs usbnet driver's support of the Linux kernel. Therefore, you need to configure the Linux kernel as follows:

1. Run **cd kernel** to enter the kernel root directory.
2. Type in **make menuconfig** and press **Enter**.
3. In the pop-up window, select the following components in sequence:

```
-> Device Drivers
   -> Network device support
       -> USB Network Adapters
           ->Multi-purpose USB Networking Framework
```

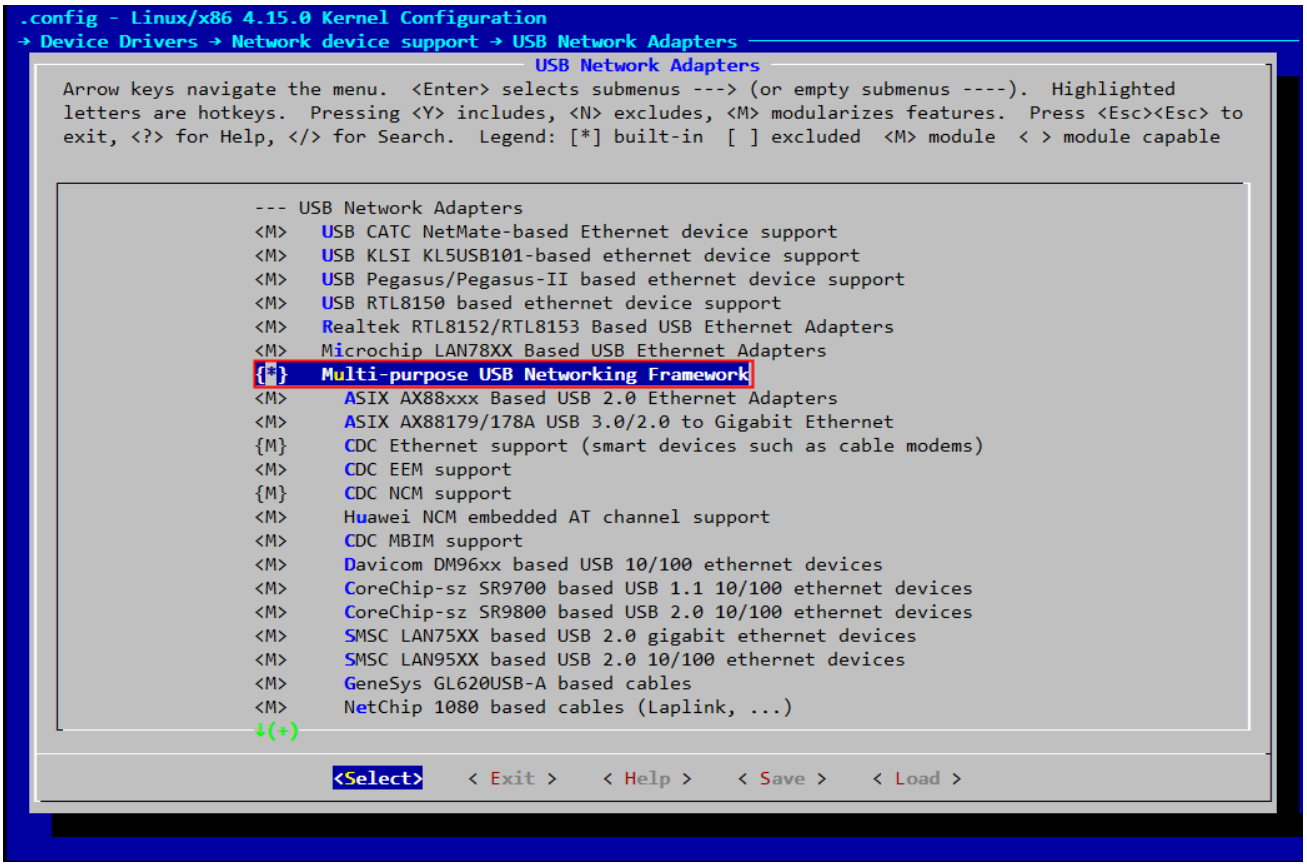

Figure 21.Selecting Multi-purpose USB Networking Framework

For the method of compiling and configuring the Linux kernel, the configuration rules of the customer's built-in Linux system shall prevail. The method described in this document is for reference only.

6.2.1.1 GobiNet Driver Code Structure

As shown below, the driver is provided in the form of source code, and it is independently compiled in the customer's system.

```
GobiNetDriver/
├── driverLoader.sh
├── GobiUSBNet.c
├── makefile
├── QMI.c
├── QMI.h
├── QMIDevice.c
├── QMIDevice.h
├── Structs.h
```

6.2.1.2 GobiNet Driver Configuration

In GobiNet driver versions later than 2.0.0, in order to optimize driver performance and enhance driver stability, it is necessary to configure the multi-channel dial-up in AT and QMI modes separately. Therefore, the driver parameters are configured during the driver loading phase to set the current driver mode. The following describes the use methods and application scenarios of the two modes.

6.2.1.2.1 AT Mode

The AT mode is mainly used for AT dial-up of Fibocom modules as described in section 6.3. You can set this mode in two ways:

1. Modifying source code.

Before compiling the driver, check the **GobiUSBNet.c** file in the GobiNet driver source code and add the macro definition "__QCRMCALL_MODE", as shown in the following figure:

```
119: //setup data call by "AT$QCRMCALL=1,1"
120: #define __QCRMCALL_MODE 1
121: #ifdef __QCRMCALL_MODE
122: static uint __read_mostly qcrmcall_mode = 1;
123: #else
124: static uint __read_mostly qcrmcall_mode = 0;
125: #endif
126: module_param( qcrmcall_mode, uint, S_IRUGO | S_IWUSR );
```

Figure 22. Adding macro definition "__QCRMCALL_MODE"

If you do not add the macro definition, you can set **qcrmcall_mode** to **1**, as shown in the following figure:

```
119: //setup data call by "AT$QCRMCALL=1,1"
120: #ifdef __QCRMCALL_MODE
121: static uint __read_mostly qcrmcall_mode = 1;
122: #else
123: static uint __read_mostly qcrmcall_mode = 1;
124: #endif
125: module_param( qcrmcall_mode, uint, S_IRUGO | S_IWUSR );
```

Figure 23. Setting qcrmcall_mode to 1

2. Modifying driver loading parameters.

If the **GobiUSBNet.c** file is not modified before the driver is compiled, you can set the following driver parameters when the driver is loaded.

```
insmod GobiNet_f.ko qcrmcall_mode=1
```

6.2.1.2.2 QMI Mode

The QMI mode is mainly used for IP aggregation and multi-channel dial-up. AT dial-up is not supported in this mode. Dial-up for Internet access requires Fibocom-dial tool. For details, see section 6.4.1.1.

IP aggregation can significantly reduce the CPU usage of the host computer and improve the efficiency of data transmission and reception. IP aggregation is recommended when the CPU space or memory space of the host computer is small.

You can set this mode in two ways:

1. Modifying source code

Before compiling the driver, check that **qcrmcall_mode** in the **GobiUSBNet.c** file in the GobiNet driver source code is set to **0**. The source code supports the QMI mode by default. Therefore, there is no need to modify the source code here. Just ensure that the setting here is consistent with the source code.

```
119: //setup data call by "AT$QCRMCALL=1,1"
120: #ifdef __QCRMCALL_MODE
121: static uint __read_mostly qcrmcall_mode = 1;
122: #else
123: static uint __read_mostly qcrmcall_mode = 0;
124: #endif
125: module_param( qcrmcall_mode, uint, S_IRUGO | S_IWUSR )
```

Figure 24. Setting qcrmcall_mode to 0

If IP aggregation is used, set the macro definition **__QMAP_NUM** in line 188 to **1** (for multiple channels, set it to the corresponding number of channels. Currently, a maximum of 5 channels are supported), as shown in the following figure. The **__QMAP_NUM** in the driver source code is **0**, indicating that IP aggregation is not applicable.

```
188: #define __QMAP_NUM 1
189: #ifndef __QMAP_NUM
190: static uint __read_mostly qmap_mode = 0;
191: #else
192: static uint __read_mostly qmap_mode = __QMAP_NUM;
193: #endif
```

Figure 25. Setting macro definition __QMAP_NUM

2. Modifying driver loading parameters

If the **GobiUSBNet.c** file is not modified before the driver is compiled, you can set the following parameters when the driver is loaded.

```
insmod GobiNet_f.ko qcrmcall_mode=0 qmap_mode=1          // IP aggregation
insmod GobiNet_f.ko qcrmcall_mode=0 qmap_mode=5         // 5-channel dial-up
```

6.2.1.3 GobiNet Driver Compilation

Copy the GobiNet source code file to the **kernel/drivers/net/usb** (except the **makefile** of the GobiNet

driver) directory of the image to be compiled.

6.2.1.3.1 Compilation in builtin mode

Add the following content to kernel/drivers/net/usb/Makefile: After that, the GobiNet driver will be automatically compiled each time the kernel is compiled.

```
obj-y += GobiNet_f.o
GobiNet_f-objs: = GobiUSBNet.o QMIDevice.o QMI.o
```

6.2.1.3.2 Compilation in .ko mode

Add the following content to kernel/drivers/net/usb/Makefile: After that, the GobiNet driver will be automatically compiled each time the kernel is compiled.

```
obj-m := GobiNet_f.o
GobiNet_f-objs := GobiUSBNet.o QMIDevice.o QMI.o
```

6.2.1.4 GobiNet Driver Loading

1. If the GobiNet driver is compiled in .ko mode, GobiNet_f.ko will be added to the kernel as a module.
2. If the driver is not loaded, you can load it manually in two ways:

   a. Use the insmod command to load the GobiNet driver:

```
sudo insmod GobiNet_f.ko
```

   b. Use the modprobe command to load the GobiNet driver:

```
cp -f GobiNet_f.ko /lib/modules/`uname -r`/kernel/drivers/net/usb/
modprobe GobiNet_f
```

   c. Use the ifconfig command to check NIC information. If usb0 is available, the driver is loaded successfully, as shown below.

```
usb0      Link encap:Ethernet  HWaddr 52:f4:dc:d6:ce:0a
          inet6 addr: fe80::50f4:dcff:fed6:ce0a/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```


The NIC name may vary with the specific Linux version (maybe other than usb0).

6.2.1.5 GobiNet Driver Log Output

In the current GobiNet driver code, different levels are set for log output in different scenarios.

For example, in the QMI. h file, the output level ranges from 1 to 3. Using the fibo_debug parameter value as the identification standard, only the log with a value smaller than or equal to the fibo_debug value will be output.

```
extern int fibo_debug;
// DBG macro
enum G_LOG_LEVEL {
    G_LOG_ERR = 1,
    G_LOG_INFO,
    G_LOG_DEBUG,
};

#define G_ERR( format, arg... ) do { \
    if (fibo_debug >= G_LOG_ERR)\
    { \
        printk( KERN_ERR "GobiNet::%d::%s " format, __LINE__,__FUNCTION__, ## arg ); \
    }}while(0)

#define G_INFO( format, arg... ) do { \
    if (fibo_debug >= G_LOG_INFO)\
    { \
        printk( KERN_INFO "GobiNet::%d::%s " format,__LINE__,__FUNCTION__, ## arg ); \
    }}while(0)

#define G_DEBUG( format, arg... ) do { \
    if (fibo_debug >= G_LOG_DEBUG)\
    { \
        printk( KERN_DEBUG "GobiNet::%d::%s " format, __LINE__,__FUNCTION__, ## arg ); \
    }}while(0)
```

Figure 26. Log output settings

In the GobiUSBNet.c file, fibo_debug is used as a module parameter and is initialized to G_LOG_ERR by default, which means that G_ERR and related logs will be output by default when the driver is loaded.

```
int fibo_debug = G_LOG_ERR;
module_param( fibo_debug, int, S_IRUGO | S_IWUSR );
MODULE_PARM_DESC( fibo_debug, "Debuging enabled or not" );
```

In the process of driver debugging, if you need the log in a certain scenario, but it is not output by default, you can send a command to dynamically set the debug parameters:

Disable log output:

```
echo 0 > /sys/module/GobiNet/parameters/fibo_debug
```

Dynamically set log output (n ranges from 1 to 3):

```
echo n > /sys/module/GobiNet/parameters/fibo_debug
```

6.3 GobiNet Dial-up with AT Commands

6.3.1 Process of GobiNet Dial-up with AT Commands

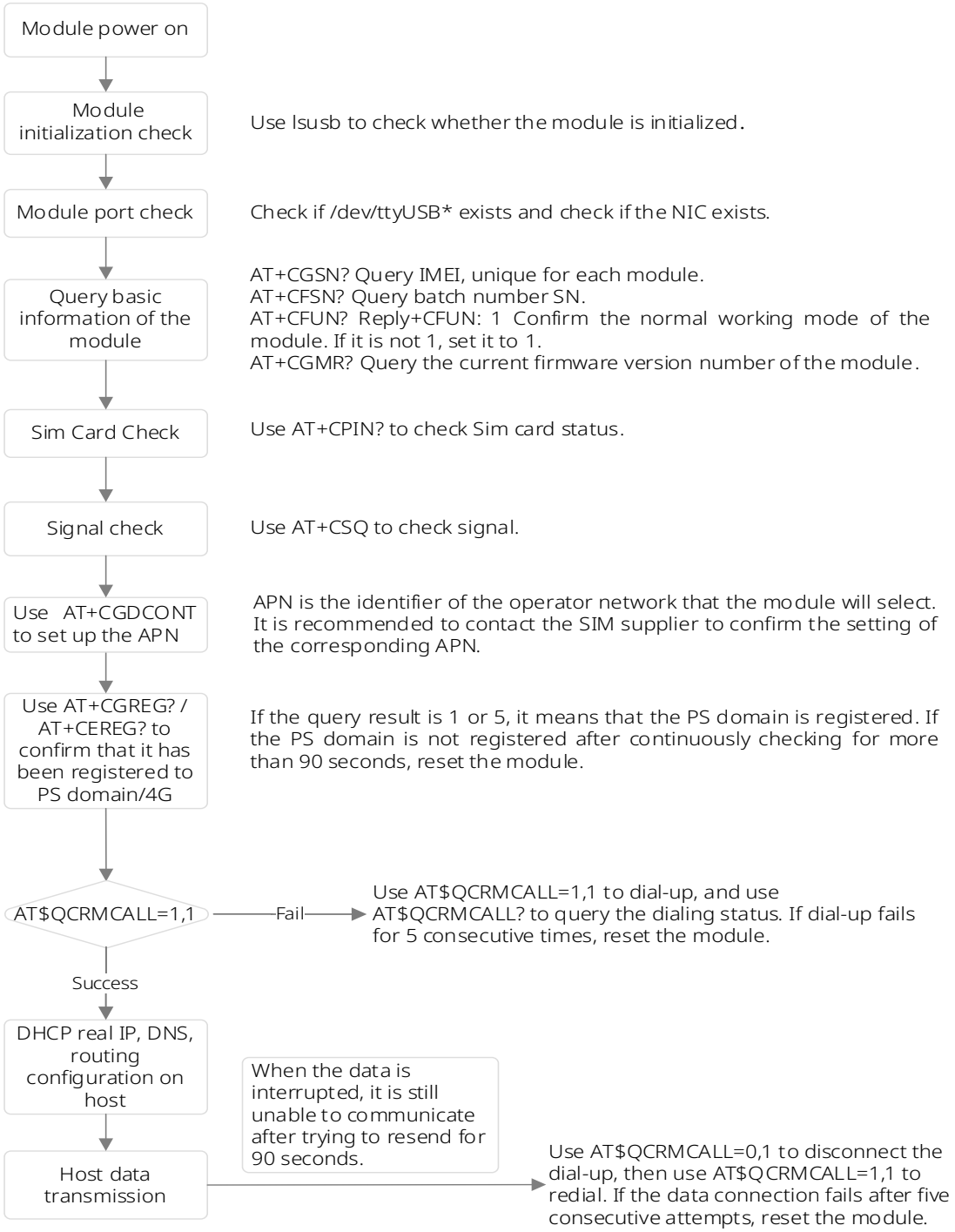

Figure 27. Flowchart of GobiNet dial-up with AT commands

6.3.2 Description of GobiNet AT Commands

The following table shows the format of AT$QCRMCALL commands:

Table 9. AT$QCRMCALL command format

Type	Command	Response
Setting command	$QCRMCALL=<Action>,<Instance>[,<IP Type> [,<Tech Pref >[,<umts profile number> [,<cdma profile number > [,<APN>]]]]]	Response 1： $QCRMCALL: <Instance>,<IP Type> OK Response 2： ERROR/+CMS ERROR: <err> Response 3： NO CARRIER
Reading the current state	$QCRMCALL?	Response 1： $QCRMCALL: <Instance>,<IP Type> OK Response 2： ERROR/+CMS ERROR: <err>
Querying the command parameter range	$QCRMCALL=?	$QCRMCALL: (list of supported <Action>s),(list of supported <Instance>s),(list of supported <IP Type>s),(list of supported <Tech Pref>s),(list of supported <umts profile number>s), (list of supported <cdma profile number>s),(max length of supported <APN>) OK

Table 10. AT$QCRMCALL parameters

Name	Description	Value
<Action>	Activation/Deactivation	Type: Integer 0 – Stop

		1 – Start
<Instance>	Example	Type: Integer 1
<IP Type>	PDP type	Type: Integer 1 – Ipv4 2 – Ipv6 3 – Ipv4v6
<Tech Pref>	Tech type	Type: Integer 1 – 3GPP2 2 – 3GPP
<umts_profile>	3GPP Profile ID	Type: integer Specific PDP context definition (see the +CGDCONT command)
<cdma profile number >	3GPP2 Profile ID	Type: integer Value range: 100–179
<APN>	APN	Integer; maximum length: 100

6.3.3 Related AT Logs and Descriptions

If the Internet access is needed, the following dial-up process is recommended (taking China Telecom card as an example):

```
AT+CPIN?
+CPIN: READY                              //Ensure that the SIM card is ready.
OK
AT+CSQ
+CSQ: 21,99                                  //Ensure that the module can receive
the signal.
OK
AT+COPS?
+COPS: 0,0,"CHN-TELECOM",7              //Ensure that the module has been registered
to the operator’s network.
OK
AT+CGREG?
+CGREG: 0,1                               //The module attaches to the data network.
OK
AT+CEREG?  //The module is attached to the LTE network.
```

```
+CEREG: 0, 1
OK
AT+CGDCONT=1,"IPV4V6","CTNET"                                    //Set the access point
information.
OK
AT+CFUN=4                                                       //Disable the RF.
OK
AT+CFUN=1                                                       //Enable the RF.
OK
AT+CGDCONT?                                                      //Query whether the setting
is successful.
+CGDCONT: 1,"IPV4V6","CTNET","10.187.169.152,36.14.4.87.255.16.120.160.23.151.10
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0,36.14.5.84.129.240.93.92.23.151.191.204.1200
+CGDCONT: 3,"IPV4V6","sos","0.0.0.0,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1
+CGDCONT: 4,"IPV4V6","CTWEB","0.0.0.0,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0

OK
AT$QCRMCALL=1,1                                                    //Initiate the dial-up.
$QCRMCALL: 1, V4
OK
…
AT$QCRMCALL?
$QCRMCALL: 1, V4  //Regularly query dial-up status. If it is disconnected, initiate
dial-up again
OK
…
```

After the dial-up, run **ifconfig** in the terminal window. If the NIC has obtained the IP address and data services can be conducted through the NIC, the dial-up is successful:

```
usb0      Link encap:以太网  硬件地址 ce:9d:05:89:61:9c
          inet 地址:10.187.169.152  广播:10.187.169.159  掩码:255.255.255.240
          inet6 地址: fe80::ccbc:c42:f9ad:4a50/64 Scope:Link
          UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  跃点数:1
          接收数据包:7 错误:7 丢弃:0 过载:0 帧数:0
          发送数据包:67 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:2162 (2.1 KB)  发送字节:10428 (10.4 KB)
```

Figure 28. Checking the NIC information

If the built-in Linux system cannot obtain the IP address, run **udhcpc -i usb0** to configure the IP address for the system.

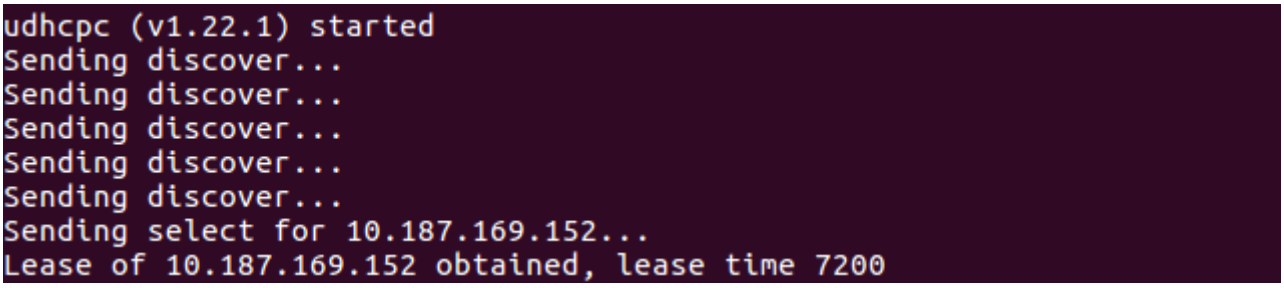

Figure 29. Manually configuring the IP address

If you need to disconnect from the network, run the following command:

```
AT$QCRMCALL=0,1
OK
AT$QCRMCALL?
OK
```

6.4 GobiNet QMI Dial-up

Qualcomm chips later than 9x07 support only RmNet interfaces. To enable user terminals to support multi-PDN links, Qualcomm chips adopt the QMAP protocol to support multiple IP channels over one RMNet interface. The GobiNet driver can virtualize multiple network interfaces on the host computer side. On the modem side, the data format is set to qmap through QMI, and multiple links are also virtualized. The host computer and modem use qmux_id to distinguish the links. The block diagram of the implementation in the system is shown below.

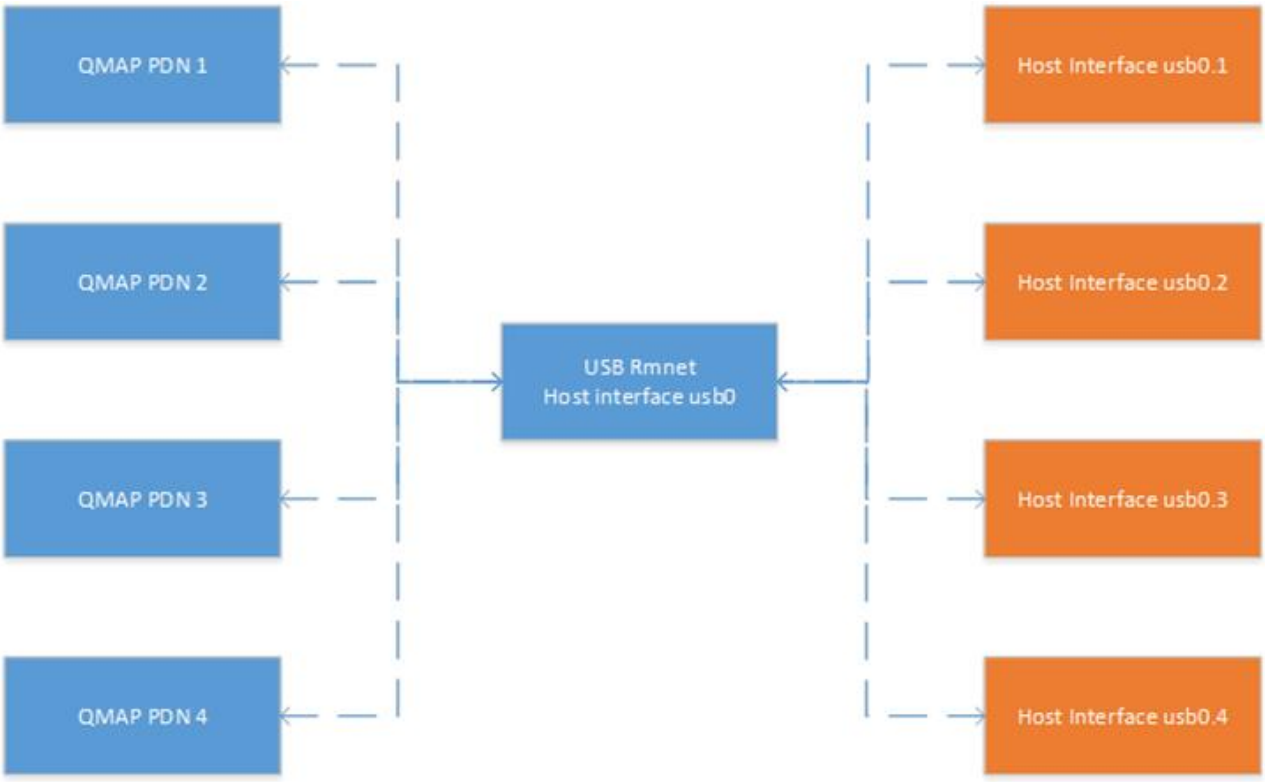

Figure 30. Block diagram of single-channel RmNet supporting multiple IP channels

6.4.1 Fibocom-dial Management

6.4.1.1 Fibocom-dial Application

The Fibocom-dial can be used for single-channel or multi-channel dial-up, and supports the specified interface during application.

The following table lists all parameters of Fibocom-dial, used in the format of "./fibocom-dial [options]".

Table 11. Fibocom-dial parameters and applications

Parameter	Description
-s [apn [user password auth]]	When the **-s** parameter is configured, the PDP profile saved in the module will be modified.
-p pincode	If the SIM card is locked, check the SIM card pin.
-f logfilename	Save the log information of this application to a file.
-i interface	Specify the network port (automatically detected by default).
-4	IPv4 dial-up.
-6	IPv6 dial-up.
-g	IPv6 private gateway.

-m PDP ID	Specify the PDP and APN for multi-PDN data connection (this ID corresponds to CIDx in the cgdcont query result).
-n channelID	Specify the virtual network interface ID when setting up a multi-PDN data connection (1 by default).
-N Number of channel	Specify the total number of channels when setting up a multi-PDN data connection (1 by default).
-k pid	Terminate the specified process according to pid.

You need to set **-N** once only. To ensure the stability of the data channel, Fibocom-dial will kill other Fibocom-dial processes that are previously used to establish links and delete the network interface to recreate it, if you set different numbers of qmaps supported for multiple times without restarting the module.

Example 1:

```
./fibocom-dial
```

Example 2:

```
./fibocom-dial -s ctnet
```

Example 3:

```
./fibocom-dial -s ctnet carl 0000 0 -p 0000 -f gobinet_log.txt
```

6.4.1.2 Fibocom-dial Dial-up Example

6.4.1.2.1 Fibocom-dial Single-channel Dial-up Example

1. After the module is inserted into the system, run the **lsusb** command to check the identified USB devices.

```
root@ubuntu:/home/kaibo/Fibocom_Linux_GobiNet_multi_driver# lsusb
Bus 004 Device 002: ID 2cb7:0104 Fibocom Fibocom Modem_SN:17641255
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 003 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 002: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Figure 31. Devices normally reported

2. Check that the driver is installed correctly (see section 6.2.1).

3. If the __QMAP_NUM in the GobiUSBNet.c file of the driver in 2.x version is not modified, the __QMAP_NUM defaults to 1, indicating single-channel dial-up. Run Ifconfig -a to check that there is 1 PDN channel and the network interface is named usb0.

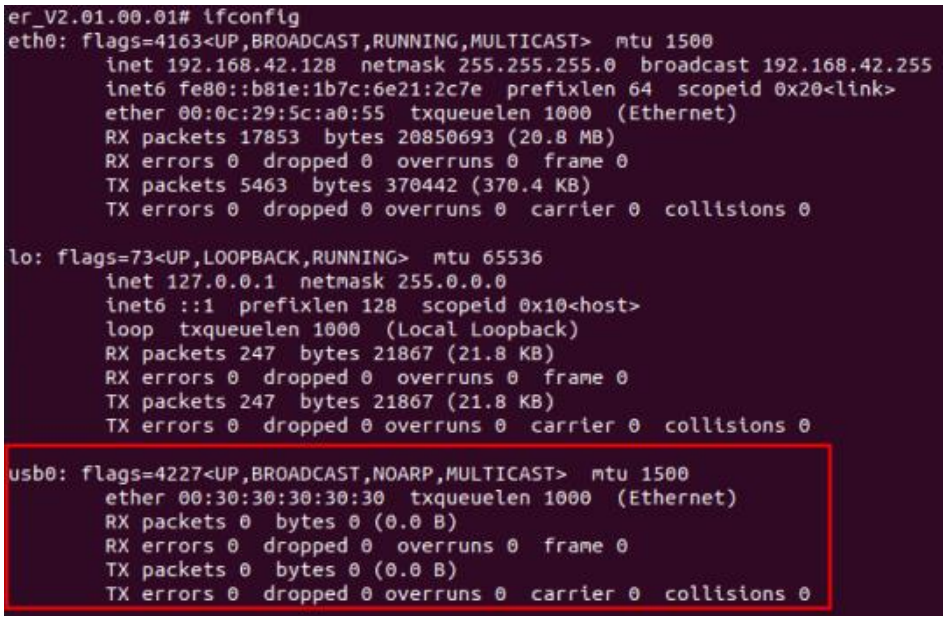

Figure 32. Information about usb0 of single-channel NIC

4. Enter the fibocom-dial/code/src directory and run make to generate binary file fibocom-dial. Run ./fibocom-dial to implement single-channel dial-up. The following figure shows the dial-up log.

```
root@ubuntu:/home/mousse/Desktop/fibocom-dial_2in1/src# ./fibocom-dial
[03-25_17:28:14:443] Fibocom-dial_Linux_Tool_V2.0.11
[03-25_17:28:14:443] ./fibocom-dial profile[1] = (null)/(null)/(null)/0, pincode = (null)
[03-25_17:28:14:443] socket[3] successfuly!
[03-25_17:28:14:443] Waiting client to connect.....
[03-25_17:28:14:444] Find /sys/bus/usb/devices/4-1 idVendor=2cb7 idProduct=0104
[03-25_17:28:14:444] Find /sys/bus/usb/devices/4-1:1.4/net/usb0
[03-25_17:28:14:444] Find usbnet_adapter = usb0
[03-25_17:28:14:445] Find /sys/bus/usb/devices/4-1:1.4/GobiQMI/qcqmi5
[03-25_17:28:14:445] Find qmichannel = /dev/qcqmi5
[03-25_17:28:14:445] qmichannel(/dev/qcqmi5) usbnet_adapter(usb0)
[03-25_17:28:14:445] usb rmnet mode
[03-25_17:28:14:445] qmap_mode = 1, muxid = 0x00, qmap_netcard = usb0
[03-25_17:28:14:445] qmap_mode=1
[03-25_17:28:14:452] GobiNetThread 130
[03-25_17:28:14:462] GobiNetGetClientID: QMIType = 1 clientid 8
[03-25_17:28:14:462] Get clientWDS = 8
[03-25_17:28:14:466] GobiNetGetClientID: QMIType = 2 clientid 10
[03-25_17:28:14:466] Get clientDMS = 10
[03-25_17:28:14:474] GobiNetGetClientID: QMIType = 3 clientid 11
[03-25_17:28:14:477] Get clientNAS = 11
[03-25_17:28:14:486] GobiNetGetClientID: QMIType = 11 clientid 12
[03-25_17:28:14:486] Get clientUIM = 12
[03-25_17:28:14:487] write triger_event: 4098 to qmidevice_control_fd
[03-25_17:28:14:494] requestBaseBandVersion 19101.1000.00.00.30.10
[03-25_17:28:14:511] dev: /dev/ttyUSB1
sendbuffer:AT+GTDUALSIMEN?
```

```
[03-25_17:28:17:523] requestGetSIMStatus SIMStatus: SIM_READY
[03-25_17:28:17:530] requestGetICCID DeviceICCID: 89860121801361187764
[03-25_17:28:17:534] requestGetIMSI DeviceIMSI: 460017548066241
[03-25_17:28:17:546] requestGetProfile[1] ///0
[03-25_17:28:17:554] requestRegistrationState2 MCC: 460, MNC: 1, PS: Attached, DataCap: LTE
[03-25_17:28:17:554] write signo: 12 to signal_control_fd
[03-25_17:28:17:554] epoll fd = 5, events = 0x0001
[03-25_17:28:17:554] get signo: 12
[03-25_17:28:17:562] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[03-25_17:28:17:562] usbnet_link_change :link:0
[03-25_17:28:17:562] write signo: 10 to signal_control_fd
[03-25_17:28:17:562] epoll fd = 6, events = 0x0000
[03-25_17:28:17:562] epoll fd = 5, events = 0x0001
[03-25_17:28:17:562] get signo: 10
[03-25_17:28:17:562] usbnet_link_change :link:0
[03-25_17:28:17:570] requestRegistrationState2 MCC: 460, MNC: 1, PS: Attached, DataCap: LTE
[03-25_17:28:17:594] requestSetupDataCall WdsConnectionIPv4Handle: 0x0ec57370
[03-25_17:28:17:598] write triger_event: 4101 to qmidevice_control_fd
[03-25_17:28:17:619] epoll fd = 6, events = 0x0000
[03-25_17:28:17:619] epoll fd = 5, events = 0x0000
[03-25_17:28:17:619] epoll fd = 6, events = 0x0001
[03-25_17:28:17:619] write signo: 12 to signal_control_fd
[03-25_17:28:17:619] epoll fd = 5, events = 0x0001
[03-25_17:28:17:619] get signo: 12
[03-25_17:28:17:632] usbnet_link_change :link:1
[03-25_17:28:17:638] enter fibo_set_driver_link_state
[03-25_17:28:17:639] if_link_up usb0
[03-25_17:28:17:639] IPv4 MTU: 1500
[03-25_17:28:17:639] IPv4 Address: 11.169.68.250
[03-25_17:28:17:639] IPv4 Netmask: 30
[03-25_17:28:17:639] IPv4 Gateway: 11.169.68.249
[03-25_17:28:17:639] IPv4 DNS1: 221.11.1.67
[03-25_17:28:17:639] IPv4 DNS2: 221.11.1.68
[03-25_17:28:17:639] if_link_up usb0
[03-25_17:28:19:640] The IPv6 Address in profile is NULL
[03-25_17:28:19:640] epoll fd = 6, events = 0x0000
[03-25_17:28:34:655] write signo: 12 to signal_control_fd
[03-25_17:28:34:655] epoll fd = 5, events = 0x0001
[03-25_17:28:34:655] get signo: 12
[03-25_17:28:34:664] usbnet_link_change :link:1
[03-25_17:28:34:664] epoll fd = 6, events = 0x0000
```

Figure 33. Single-channel dial-up log

5. According to the figure above, the single-channel dial-up is successful. Run the **ifconfig** command. It is found that usb0 of NIC successfully obtains the IP address.

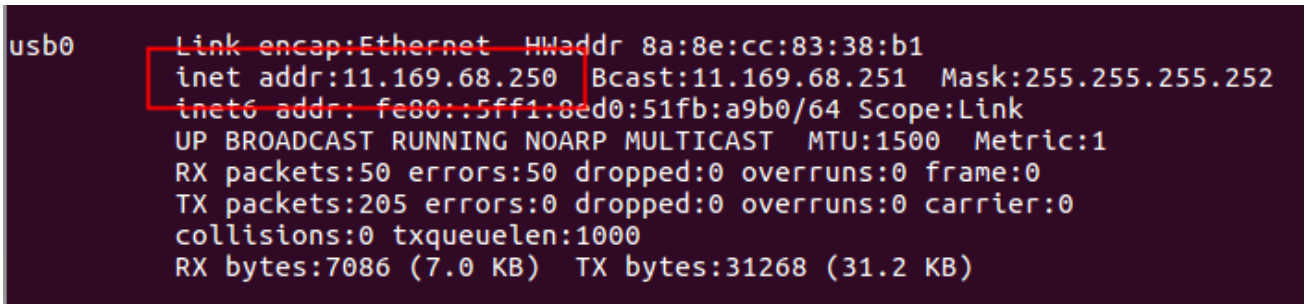

Figure 34. usb0 of NIC successfully obtained the IP address in single-channel dial-up

6. Run the **ping** command over usb0.

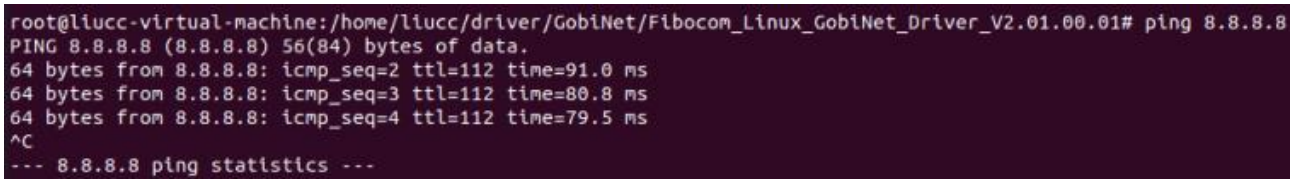

Figure 35. Successfully pinging the device using usb0 of NIC

6.4.1.2.2 Fibocom-dial Multi-channel Dial-up Example

The following example shows how to use two of four channels for configuration and dial-up:

1. Set the number of PDNs to 4 when loading the GobiNet driver.

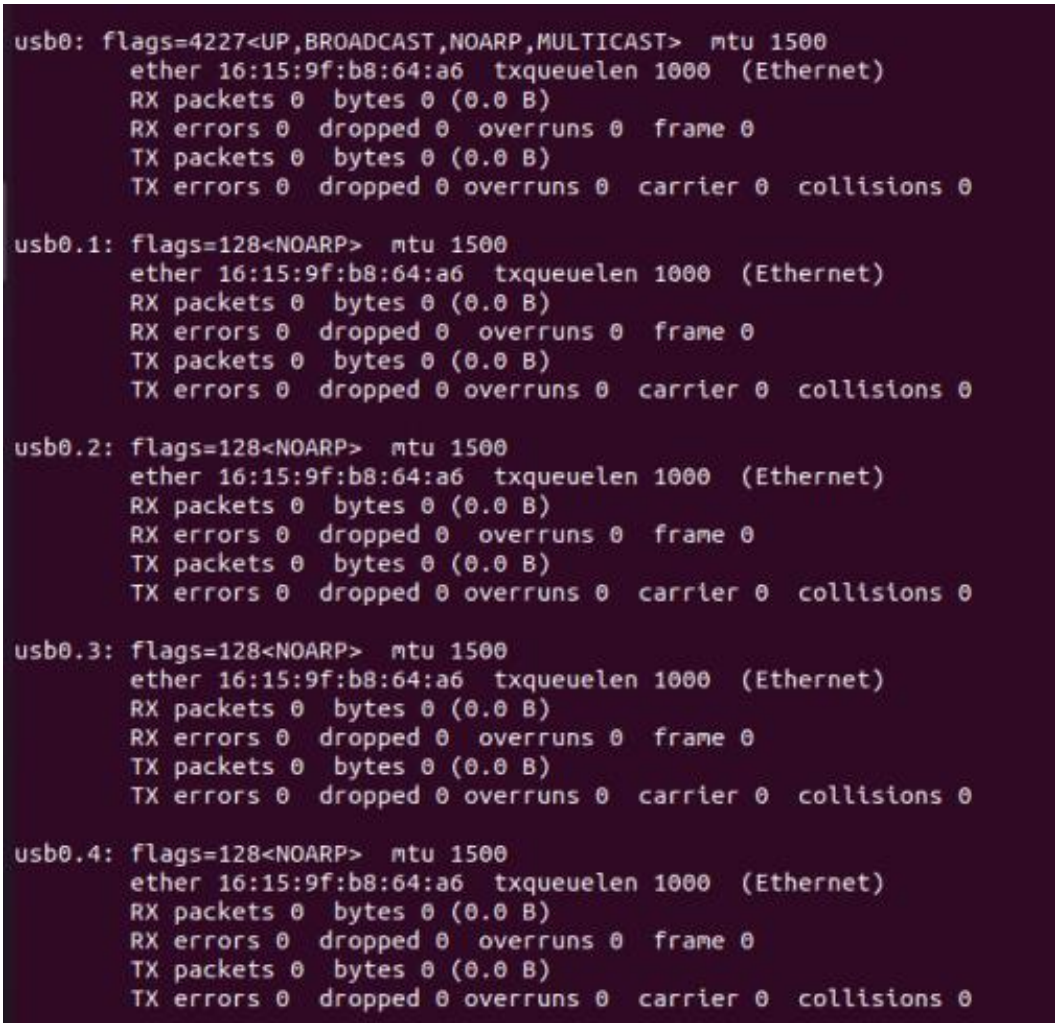

Figure 36. Setting the number of PDNs to 4

2. Run **./fibocom-dial -n 2 -s ct 123 234 0 –m 2 -4 -6** to set channel 2 to implement dual-stack dial-up.

Where:

- **-n 2** indicates that the channel ID is 2. That is, network interface usb0.2 is used as the data channel for dial-up.
- **-m 2** indicates to set the PDP profile subscript, corresponding to the CID in **AT+CGDCONT**.
- **-4** indicates to enable IPv4 dial-up.
- **-6** indicates to enable IPv6 dial-up.
- **-s ct 123 234 0** indicates to set the APN to **ct**, username to **123**, password to **234**, and authentication type to **0**.

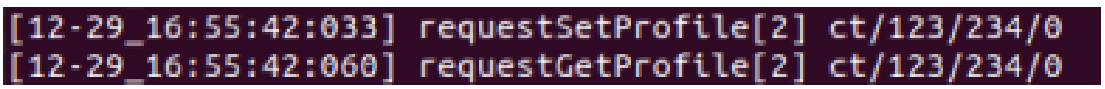

Figure 37. APN settings

If the **-s** parameter is configured, requestSetProfile[2] will be output in the application, as shown in the preceding figure. 2 indicates the PDP profile subscript set using the **-m** parameter.

If the -s parameter is not set, the application takes the preset dial-up parameters of channel 2 from the module software.

The authentication type is defined as follows: 0 – none; 1 – Pap; 2 – Chap; 3 – PAP&CHAP.

```
root@liucc-virtual-machine:/home/liucc/fibocom_dial# ./fibocom-dial -n 2 -s ct 123 234 0 -m 2 -4 -6
[12-29_17:21:36:649] Fibocom-dial_Linux_Tool_V2.0.2
[12-29_17:21:36:649] ./fibocom-dial profile[2] = ct/123/234/0, pincode = (null)
[12-29_17:21:36:649] socket[3] successfuly!
[12-29_17:21:36:649] Waiting client to connect.....
[12-29_17:21:36:650] Find /sys/bus/usb/devices/4-1 idVendor=2cb7 idProduct=0104
[12-29_17:21:36:650] Find /sys/bus/usb/devices/4-1:1.4/net/usb0
[12-29_17:21:36:650] Find usbnet_adapter = usb0
[12-29_17:21:36:650] Find /sys/bus/usb/devices/4-1:1.4/GobiQMI/qcqmi0
[12-29_17:21:36:650] Find qmichannel = /dev/qcqmi0
[12-29_17:21:36:650] qmichannel(/dev/qcqmi0) usbnet_adapter(usb0)
[12-29_17:21:36:650] usb rmnet mode
[12-29_17:21:36:651] qmap_mode = 4, muxid = 0x82, qmap_netcard = usb0.2
[12-29_17:21:36:651] qmap_mode = 4, muxid = 0x82, qmap_netcard = usb0.2
[12-29_17:21:36:659] GobiNetGetClientID: QMIType = 1 clientid 9
[12-29_17:21:36:659] Get clientWDS = 9
[12-29_17:21:36:668] GobiNetGetClientID: QMIType = 1 clientid 10
[12-29_17:21:36:669] Get clientWDS = 10
[12-29_17:21:36:674] GobiNetGetClientID: QMIType = 2 clientid 11
[12-29_17:21:36:674] Get clientDMS = 11
[12-29_17:21:36:679] GobiNetGetClientID: QMIType = 3 clientid 12
[12-29_17:21:36:679] Get clientNAS = 12
[12-29_17:21:36:683] GobiNetGetClientID: QMIType = 11 clientid 13
[12-29_17:21:36:684] Get clientUIM = 13
[12-29_17:21:36:689] requestBaseBandVersion 89100.1000.00.04.05.07
[12-29_17:21:36:734] requestGetSIMStatus SIMStatus: SIM_READY
[12-29_17:21:36:734] requestSetProfile[2] ct/123/234/0
[12-29_17:21:36:763] requestGetProfile[2] ct/123/234/0
[12-29_17:21:36:772] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[12-29_17:21:36:775] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[12-29_17:21:36:780] requestQueryDataCall IPv6ConnectionStatus: DISCONNECTED
[12-29_17:21:36:781] if_link_down usb0.2
[12-29_17:21:36:787] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[12-29_17:21:38:951] requestSetupDataCall WdsConnectionIPv4Handle: 0x79f63b10
[12-29_17:21:39:095] requestSetupDataCall WdsConnectionIPv6Handle: 0x79e2d150
[12-29_17:21:39:134] if_link_up usb0.2
[12-29_17:21:39:134] IPv4 MTU: 1500
[12-29_17:21:39:134] IPv4 Address: 10.212.234.161
[12-29_17:21:39:134] IPv4 Netmask: 30
[12-29_17:21:39:134] IPv4 Gateway: 10.212.234.162
[12-29_17:21:39:134] IPv4 DNS1: 61.134.1.6
[12-29_17:21:39:134] IPv4 DNS2: 218.30.19.40
[12-29_17:21:39:134] if_link_up usb0.2
[12-29_17:21:41:135] IPv6 MTU: 1500
[12-29_17:21:41:135] IPv6 Address: 240e:454:2bd:9f8a:50ed:1e9f:5462:88b8
[12-29_17:21:41:135] IPv6 Netmask: 64
[12-29_17:21:41:135] IPv6 Gateway: 240e:454:2bd:9f8a:b51d:297f:1d53:e4ef
[12-29_17:21:41:135] IPv6 DNS1: 240e:f:a::8
[12-29_17:21:41:135] IPv6 DNS2: ::
[12-29_17:21:41:135] if_link_up usb0.2
[12-29_17:21:56:871] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[12-29_17:21:56:875] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
```

Figure 38. Dial-up log of channel 2

3. Run ./fibocom-dial -n 3 -m 3 -4 to set channel 3 to implement IPv4 dial-up.

Where:

- -n 3 indicates that network interface usb0.3 is used as the data channel for dial-up.
- -m 3 indicates to set the PDP profile corresponding to the CID in AT+CGDCONT.
- -4 indicates to enable IPv4 dial-up.

If the -s parameter is not set, the application takes the preset dial-up parameters of channel 3 from the module.

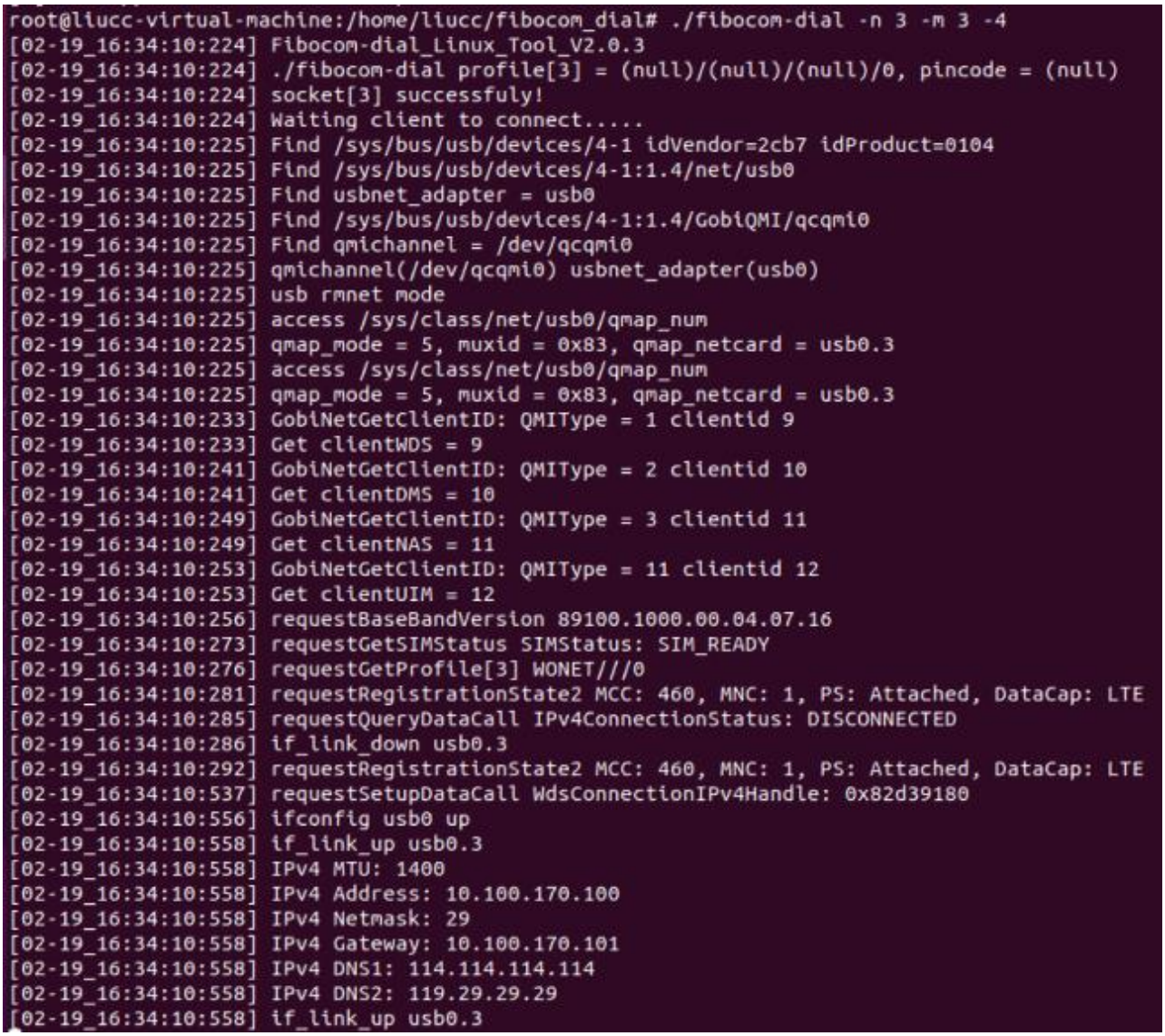

Figure 39. Dial-up log of channel 3

4. The application calls the DHCP to automatically obtain IP address and DNS, and configure route information. Run ifconfig to view the IP addresses that are allocated to the two channels.

```
usb0.2: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.142.7.108  netmask 255.255.255.248  broadcast 0.0.0.0
        inet6 240e:454:bc:fe84:6c21:323d:c20e:2252  prefixlen 64  scopeid 0x0<global>
        ether 00:30:30:30:30:30  txqueuelen 1000  (Ethernet)
        RX packets 11  bytes 1234 (1.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 195  bytes 27480 (27.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

usb0.3: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.139.223.171  netmask 255.255.255.248  broadcast 0.0.0.0
        ether 00:30:30:30:30:30  txqueuelen 1000  (Ethernet)
        RX packets 135  bytes 17242 (17.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 320  bytes 36417 (36.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 40. IP addresses successfully allocated to channel 2 and channel 3

5. Use the two channels of NICs to perform the ping test respectively.

```
root@liucc-virtual-machine:/home/liucc/fibocom_dial# ping 114.114.114.114 -I usb0.3
PING 114.114.114.114 (114.114.114.114) from 10.139.223.171 usb0.3: 56(84) bytes of data.
64 bytes from 114.114.114.114: icmp_seq=1 ttl=63 time=350 ms
64 bytes from 114.114.114.114: icmp_seq=2 ttl=90 time=47.6 ms
^C
--- 114.114.114.114 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 47.619/199.013/350.408/151.394 ms
root@liucc-virtual-machine:/home/liucc/fibocom_dial# ping 114.114.114.114 -I usb0.2
PING 114.114.114.114 (114.114.114.114) from 10.142.7.108 usb0.2: 56(84) bytes of data.
64 bytes from 114.114.114.114: icmp_seq=1 ttl=67 time=56.4 ms
64 bytes from 114.114.114.114: icmp_seq=2 ttl=80 time=193 ms
^C
--- 114.114.114.114 ping statistics ---
```

Figure 41. Ping test for two channels

6. Run **at+cgdcont?** to query the PDP profile of the module. The profile is modified and saved by Fibocom-dial.

```
at+cgdcont?
+CGDCONT: 1,"IP","1234567890","10.137.102.45",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 2,"IPV4V6","ct","10.142.142.155,36.14.4.84.0.188.254.132.0.1.0.2.93.90
+CGDCONT: 3,"IP","ctnet","10.139.223.171",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 4,"IP","ctnet","0.0.0.0",0,0,0,1,,,,,,,,,"",,,,0
```

Figure 42. Successful modification of PDP profile of channel 2

6.4.1.3 Example of Fibocom-dial Obtaining IPv6 Private Network Gateway

Run **./fibocom-dial -6 –g** to implement IPv6 dial-up and obtain the private network gateway.

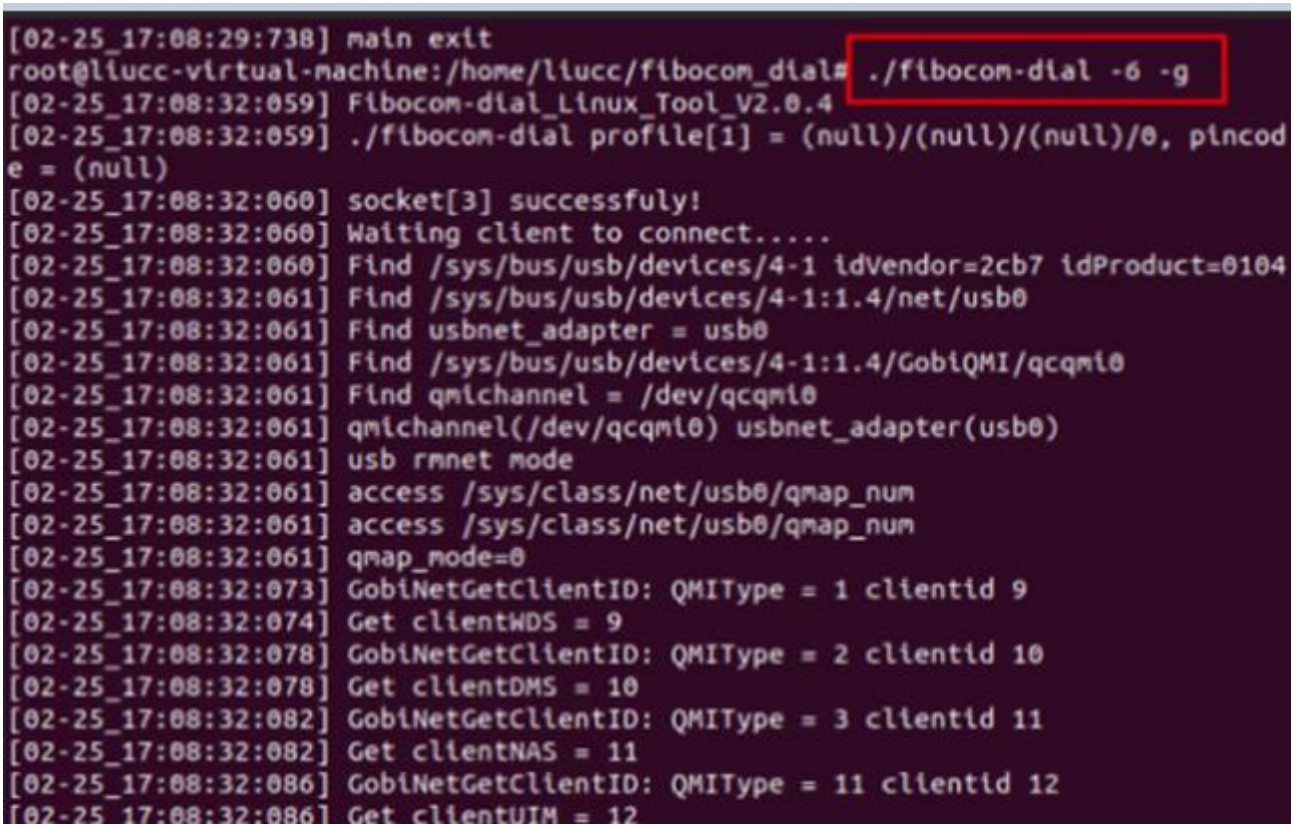

Figure 43. Dial-up by Fibocom-dial

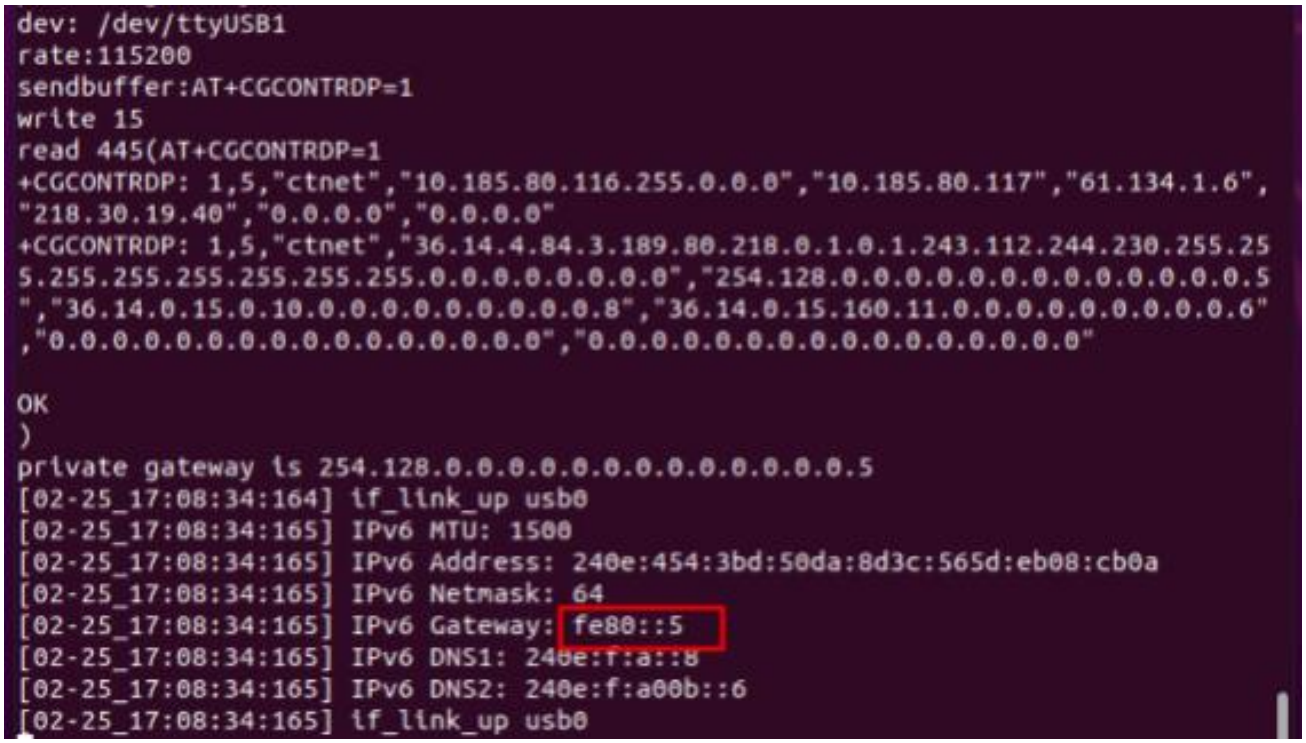

Figure 44. Obtaining the IPv6 private network gateway

As shown in the figure above, after IPv6 dial-up, the private network gateway obtained from the return value of the AT command is "254.128.0.0.0.0.0.0.0.0.0.0.0.0.0.5.", which is "fe80::5" in hexadecimal.

6.4.2 multi-pdn-manager Dial-up Management

6.4.2.1 multi-pdn-manager Application

Fibocom provides the multi-pdn-manager application for managing multiple dial-up connections at the same time and facilitating link management.

multi-pdn-manager implements dial-up control and management through Fibocom-dial. Run the **make** command in the **fibocom-dial/code/src** directory to generate the multi-pdn-manager binary file. Run the **./multi-pdn-manager** command and press **Enter** to open the application, as shown in the following figure.

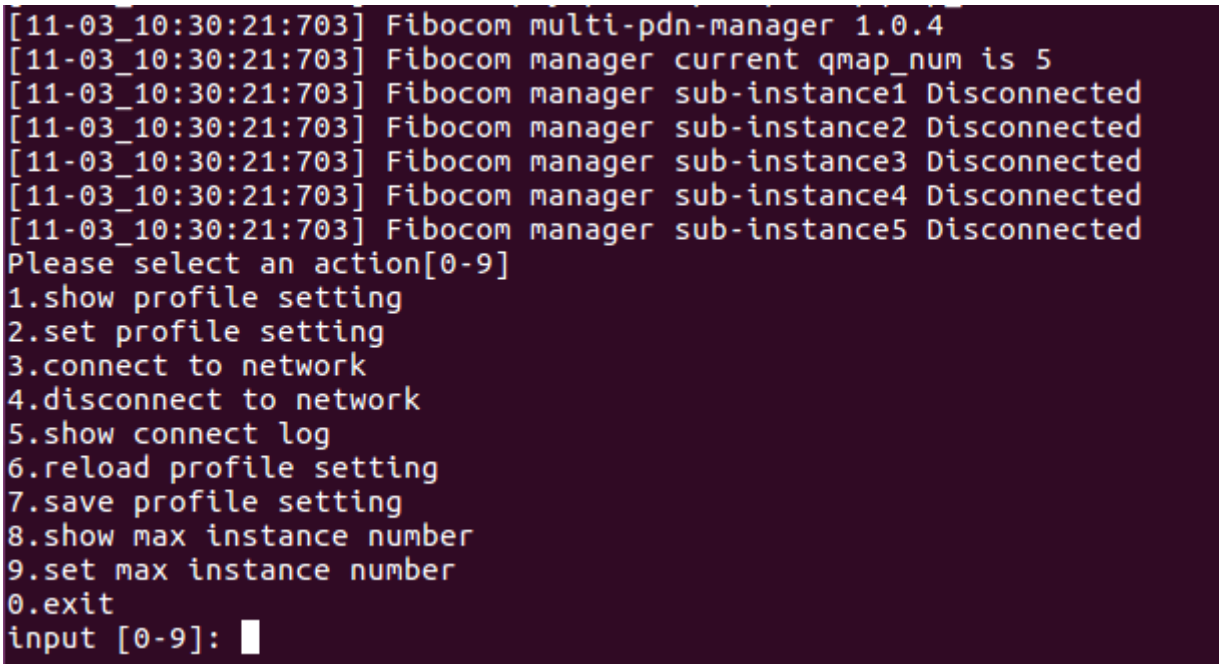

Figure 45. multi-pdn-manager application

To reduce overhead generated by interaction with the module, multi-pdn-manager uses the **multi-pdn.ini** profile to manage the number of PDNs and PDP profile. During dial-up, the corresponding PDP profile will be saved to the module through the QMI.

The application provides the following functions:

```
1. show profile setting //Display all current PDP profiles.
2. set profile setting //Set the specified PDP profile.
3. connect to network //Use the specified interface to connect to a network.
4. disconnect to network //Disconnect the specified network.
5. show connect log //Display the dial-up log.
6. reload profile setting //Reload the PDP profile settings from the profile.
7. save profile setting //Save the PDP profile settings to the file.
8. show max instance number //Display the maximum number of PDNs.
9.  set  max  instance  number  (This  function  is  invalid  in  versions  later  than
```

```
GobiNet2.0) //Set the maximum number of PDNs.
0. exit //Exit the application.
```

6.4.2.2 multi-pdn-manager Dial-up Example

1. Input option 1 displays the current PDP profile.

Figure 46. Displaying the current PDP profile in option 1

2. Input option 2 modifies the specified PDP profile.
   a. During setting, you will be prompted to set the PDP profile ID: set profile index.
   b. Enter the APN, username, password, auth, and IP family (specify IPv4 and IPv6) in sequence as prompted.

   Value options of IP family are as follows: 1: IPv4; 2: IPv6; 3: IPv4&IPv6.

   c. After the setting, the set PDP profile list will be displayed again.

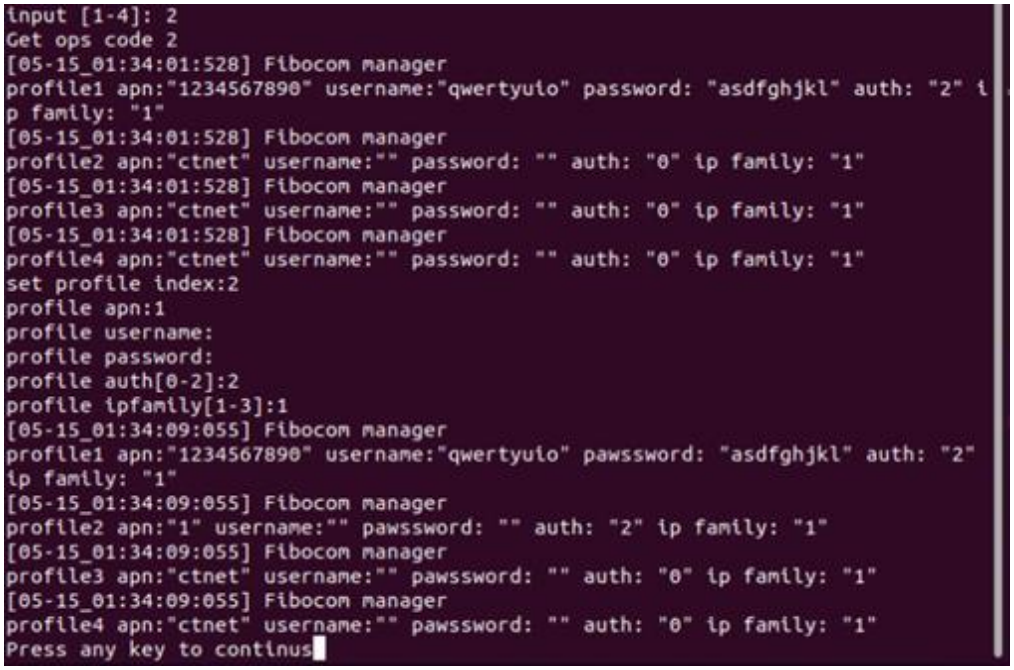

Figure 47. Modifying the specified PDP profile in option 2

3. Input option 3 to initiate the dial-up.
   a. When you initiate dial-up, you will be prompted to enter the interface to initiate the dial-up: connect instance use.
   b. During the dial-up, use the PDP profile: connect profile use.
   c. The application will call the Fibocom-dial to initiate the dial-up, and display the dial-up log on the screen. The following figure shows part of the log.

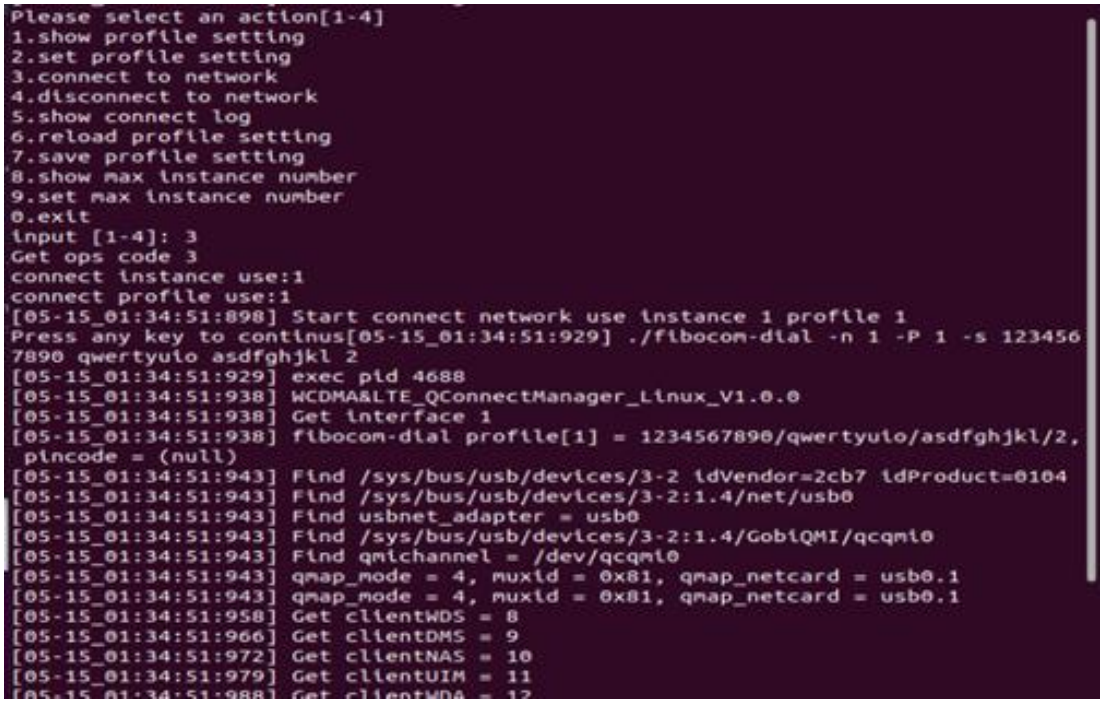

Figure 48. Initiating the dial-up in option 3

4. Input option 4 to disconnect the connection.
   a. Set the instance to disconnect: disconnect instance use.
   b. The application will terminate the Fibocom-dial, stop dial-up, and disable the corresponding usb0.x to release the IP address.

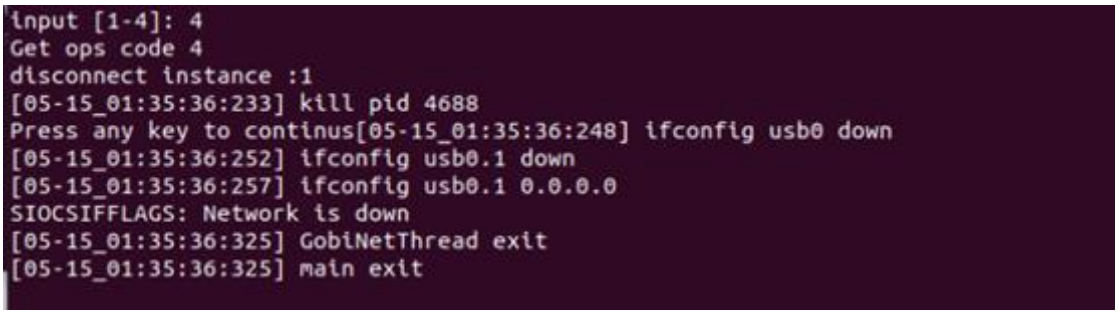

Figure 49. Disconnecting a connection in option 4

5. Input option 5 displays the dial-up log.

You will be prompted to select the instance. The application will output the complete dial-up log onto the screen.

```
input [1-4]: 5
Get ops code 5
show instance :1
[05-15_01:34:51:938] fibocom-dial profile[1] = 1234567890/qwertyuio/asdfghjkl/2,
 pincode = (null)
[05-15_01:34:51:943] Find /sys/bus/usb/devices/3-2 idVendor=2cb7 idProduct=0104
[05-15_01:34:51:943] Find /sys/bus/usb/devices/3-2:1.4/net/usb0
[05-15_01:34:51:943] Find usbnet_adapter = usb0
[05-15_01:34:51:943] Find /sys/bus/usb/devices/3-2:1.4/GobiQMI/qcqmi0
[05-15_01:34:51:943] Find qmichannel = /dev/qcqmi0
[05-15_01:34:51:943] qmap_mode = 4, muxid = 0x81, qmap_netcard = usb0.1
[05-15_01:34:51:943] qmap_mode = 4, muxid = 0x81, qmap_netcard = usb0.1
[05-15_01:34:51:958] Get clientWDS = 8
[05-15_01:34:51:966] Get clientDMS = 9
[05-15_01:34:51:972] Get clientNAS = 10
[05-15_01:34:51:979] Get clientUIM = 11
[05-15_01:34:51:988] Get clientWDA = 12
[05-15_01:34:51:993] requestBaseBandVersion 89100.1000.00.02.09.21
[05-15_01:34:51:998] qmap_settings.rx_urb_size = 16384
[05-15_01:34:52:027] requestGetSIMStatus SIMStatus: SIM_READY
[05-15_01:34:52:027] requestSetProfile[1] 1234567890/qwertyuio/asdfghjkl/2
[05-15_01:34:52:046] requestGetProfile[1] 1234567890/qwertyuio/asdfghjkl/2
[05-15_01:34:52:053] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached,
DataCap: LTE
[05-15_01:34:52:058] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[05-15_01:34:52:058] ifconfig usb0 down
[05-15_01:34:52:070] ifconfig usb0.1 down
[05-15_01:34:52:074] ifconfig usb0.1 0.0.0.0
[05-15_01:34:52:085] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached,
DataCap: LTE
[05-15_01:34:52:113] requestSetupDataCall WdsConnectionIPv4Handle: 0x68af8aa0
[05-15_01:34:52:125] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[05-15_01:34:52:130] 1 /dev/qcqmi0
[05-15_01:34:52:130] ifconfig usb0 up
[05-15_01:34:52:132] ifconfig usb0.1 up
[05-15_01:34:52:162] Fail to access /usr/share/udhcpc/default.script, errno: 2 (
No such file or directory)
[05-15_01:34:52:163] busybox udhcpc -f -n -q -t 5 -i usb0.1
[05-15_01:35:36:248] ifconfig usb0 down
[05-15_01:35:36:252] ifconfig usb0.1 down
[05-15_01:35:36:257] ifconfig usb0.1 0.0.0.0
[05-15_01:35:36:325] GobiNetThread exit
[05-15_01:35:36:325] main_exit
```

Figure 50. Displaying the dial-up log in option 5

6. Input option 6 to reload the PDP profile configuration from the multi-pdn.ini file.

```
input [1-4]: 6
Get ops code 6
Press any key to continus
```

Figure 51. Reloading the PDP profile configuration in option 6

7. Input option 7 to save the current configuration to the multi-pdn.ini file.

```
input [1-4]: 7
Get ops code 7
Press any key to continus
```

Figure 52. Saving the current configuration to the multi-pdn.ini file in option 7

8. Input option 8 to display the maximum number of PDNs.

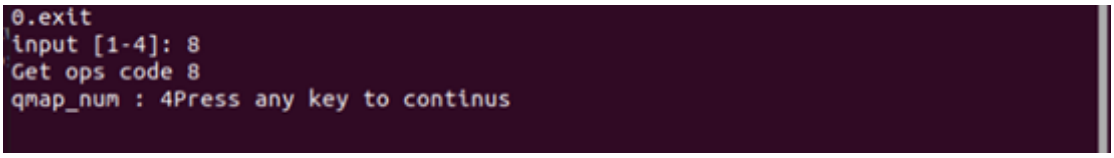

Figure 53. Displaying the maximum number of PDNs in option 8

9. Input option 9 to set the maximum number of PDNs (This function is invalid in versions later than GobiNet2.0).

Figure 54. Setting the maximum number of PDNs in option 9

10. Input option 0 to exit the application. To avoid misoperation, confirm the setting again.

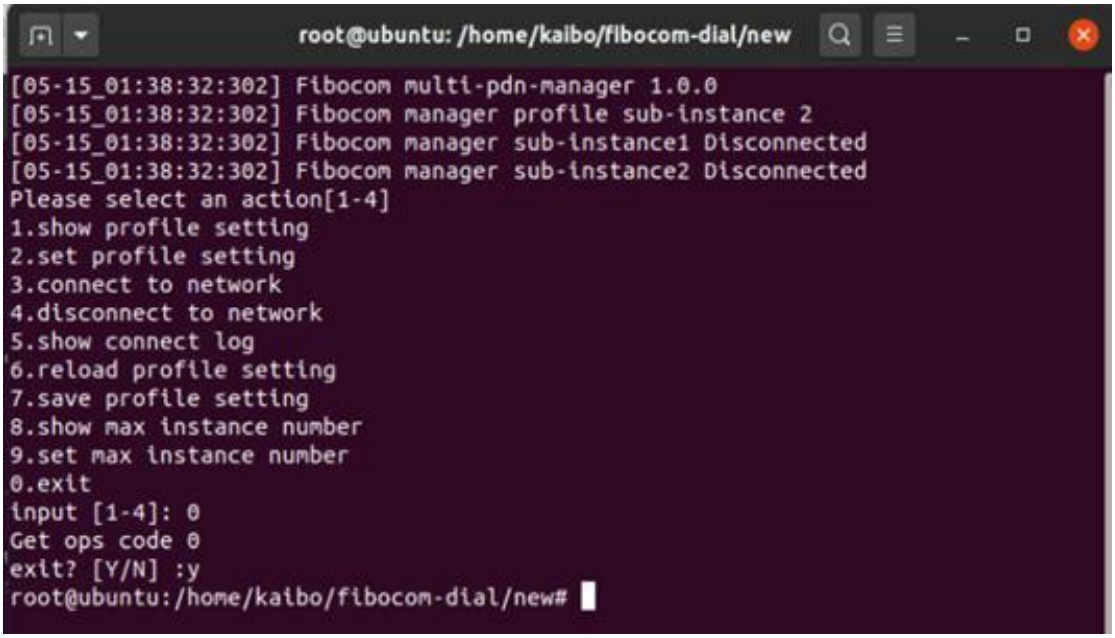

Figure 55. Exiting the application in option 0

6.5 FAQs

6.5.1 Failed to Obtain IP Address

If you cannot obtain an IP address by manually disabling and re-enabling the NIC after successful dial-up, send the following command through the AT interface to restart the module:

```
AT+CFUN=15
```

Then, manually disable the network service of the system:

```
service network-manager stop
```

Use the AT command to perform RmNet dial-up. After successful dial-up, manually apply for an IP address:

```
udhcpc -i NIC name
```

Perform the Ping test:

```
ping www.baidu.com
ping 8.8.8.8
```

Manually disable and then re-enable the NIC:

```
ifconfig NIC name down
ifconfig NIC name up
```

At this time, you can successfully apply for the IP address of the NIC and conduct the Ping test.

6.5.2 Failed to Ping the Domain Name

If the IP address is obtained successfully, address 8.8.8.8 can be successfully pinged. However, it is unable to ping www.baidu.com nor access to the Internet. In this situation, add the DNS to /etc/resolv.conf.

```
nameserver 8.8.8.8
nameserver 114.114.114.114
```

6.5.3 Failed to Refresh the IP Address

If the IP address is not refreshed in a timely manner when the network connection is disconnected, you need to use one of the following commands to refresh it:

```
ip -4 addr flush dev xxx (xxx is NIC name)
ip -6 addr flush dev xxx (xxx is NIC name)
```

6.5.4 NIC Name Modified

In the new Linux kernel, the host will automatically modify the network port name, and the NIC name is not constant. This may cause NIC failure after the dial-up is successful, and the IP address and other information cannot be allocated, which results in an uncontrollable state. The following method can prevent the NIC from being modified automatically.

1. Run sudo vim /etc/default/grub.
2. Find GRUB_CMDLINE_LINUX, and add the following two parameters: net.ifnames=0 and biosdevname=0.

```
GRUB_CMDLINE_LINUX="net.ifnames=0 biosdevname=0"
```

3. Run sudo grub-mkconfig -o /boot/grub/grub.cfg. The NIC name will not be automatically changed

after reboot.

7 QMI_WWAN Dial-up

7.1 QMI_WWAN Overview

QMI_WWAN and GobiNet are drivers of the Qualcomm module in Linux system, and both of them use the QMI interface. QMI_WWAN is not added to Linux kernel in versions earlier than Linux 3.4. Therefore, Qualcomm develops GobiNet, including its 4G driver and 4G dial-up application. QMI_WWAN has been added to Linux kernel in versions later than Linux 3.4.

Based on the native QMI_WWAN driver, Fibocom provides Fibocom_QMI_WWAN_Driver for customers, which currently supports QMI single-channel and multi-channel dial-up.

In versions later than Linux 4.5, the QMI_WWAN driver already supports the RAW IP mode. If the kernel version is later than Linux 4.5, it is recommended to use the kernel's native qmi_wwan driver.

You can choose either Fibocom_QMI_WWAN_Driver or qmi_wwan driver. There is no need to integrate both drivers at the same time.

7.2 QMI_WWAN Driver Integration

7.2.1 Integrating Fibocom_QMI_WWAN_Driver

7.2.1.1 Fibocom_QMI_WWAN_Driver Code Structure

As shown below, the driver is provided in the form of source code, and it is independently compiled in the customer's system.

```
qmi_wwan_f/
├── qmi_wwan_f.c
├── Makefile
├── README
```

7.2.1.2 Fibocom_QMI_WWAN_Driver Configuration

Copy the Fibocom_QMI_WWAN_Driver source code file to the **kernel/drivers/net/usb** (except the QMI_WWAN driver **makefile**) directory of the image to be compiled.

7.2.1.3 Fibocom_QMI_WWAN_Driver Compilation

7.2.1.3.1 Compilation in builtin mode

Add the following content to **kernel/drivers/net/usb/Makefile**. After that, the QMI_WWAN driver will be automatically compiled each time the kernel is compiled.

```
obj-y += qmi_wwan_f.o
qmi_wwan_f -objs: = qmi_wwan_f.o
```

7.2.1.3.2 Compilation in .ko mode

Add the following content to **kernel/drivers/net/usb/Makefile**. After that, the QMI_WWAN driver will be automatically compiled each time the kernel is compiled.

```
obj-m := qmi_wwan_f.o
qmi_wwan_f-objs := qmi_wwan_f.o
```

7.2.1.4 Fibocom_QMI_WWAN_Driver Loading

1. If the Fibocom_QMI_WWAN_Driver is compiled in .ko format, qmi_wwan_f.ko will be added to the system as a module and will be loaded automatically.

Figure 56. Loading the qmi_wwan_f driver

2. If it is not loaded, copy **qmi_wwan_f.ko** to the system. You can load it manually in two ways:
   a. Use the **insmod** command to load the QMI_WWAN driver:

```
$ sudo modprobe usbnet
$ sudo modprobe cdc-wdm
```

```
$ sudo insmod qmi_wwan_f.ko
```

b. Use the **modprobe** command to load the QMI_WWAN driver:

```
$ cp -f qmi_wwan_f.ko /lib/modules/'uname -r'/kernel/drivers/net/usb/

$ sudo modprobe usbnet

$ sudo modprobe cdc-wdm

$ modprobe qmi_wwan_f
```

c. Use the **usb-devices** command to check the driver loading information:

```
I:  If#=0x0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=30 Driver=option
I:  If#=0x1 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
I:  If#=0x2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
I:  If#=0x3 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=00 Prot=00 Driver=option
I:  If#=0x4 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff Driver=qmi_wwan_f
```

Figure 57. Successfully loaded the qmi_wwan_f driver

d. Use the **ifconfig -a** command to check the NIC information. If wwan0 is displayed, the driver is loaded successfully. As shown in the following figure, wwan0 indicates the added interface:

```
wwan0: flags=193<UP,RUNNING,NOARP>  mtu 1500
        ether 0a:44:72:90:32:b4  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 217 (217.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3  bytes 196 (196.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The NIC name may vary with the specific Linux version.

7.2.2 QMI_WWAN Driver Integration

If the kernel version is later than version 4.5, it is recommended to use **drivers/net/usb/qmi_wwan.c** in the kernel and modify the code as follows:

```
static const struct usb_device_id products[] = {
    ….
```

```
    /*start add by fibocom for qmi_wwan*/
    {QMI_QUIRK_SET_DTR(0x2cb7, 0x0104, 4)},
    {QMI_QUIRK_SET_DTR(0x2cb7, 0x0109, 2)},
    {QMI_QUIRK_SET_DTR(0x05c6, 0x90db, 3)},
    {QMI_QUIRK_SET_DTR(0x05c6, 0x9025, 4)},
    {QMI_QUIRK_SET_DTR(0x2cb7, 0x0112, 0)},
    {QMI_QUIRK_SET_DTR(0x2cb7, 0x0117, 0)},
    {QMI_QUIRK_SET_DTR(0x1508, 0x1001, 4)},
    {QMI_QUIRK_SET_DTR(0x1508, 0x1000, 3)},
    /*end add by fibocom for qmi_wwan*/
    …
}
```

Select the following items in config for kernel compilation:

```
CONFIG_USB_USBNET=y
CONFIG_USB_NET_DRIVERS=y
CONFIG_USB_NET_QMI_WWAN=y
```

Alternatively, select the following items in the make menuconfig menu:

```
-> Device Drivers
   -> Network device support
       -> USB Network Adapters
           -><*> QMI WWAN driver for Qualcomm MSM based 3G and LTE modems
```

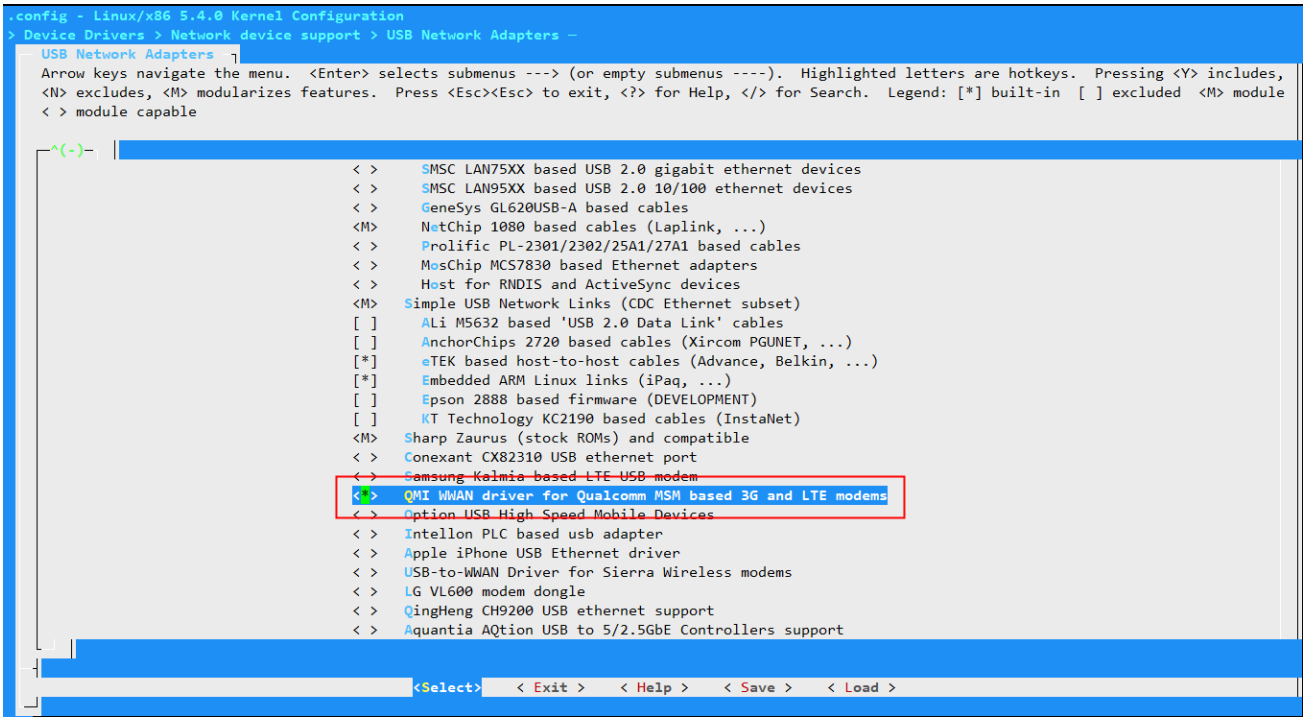

Figure 58. Detecting the qmi_wwan driver configuration

7.3 QMI_WWAN Dial-up Process

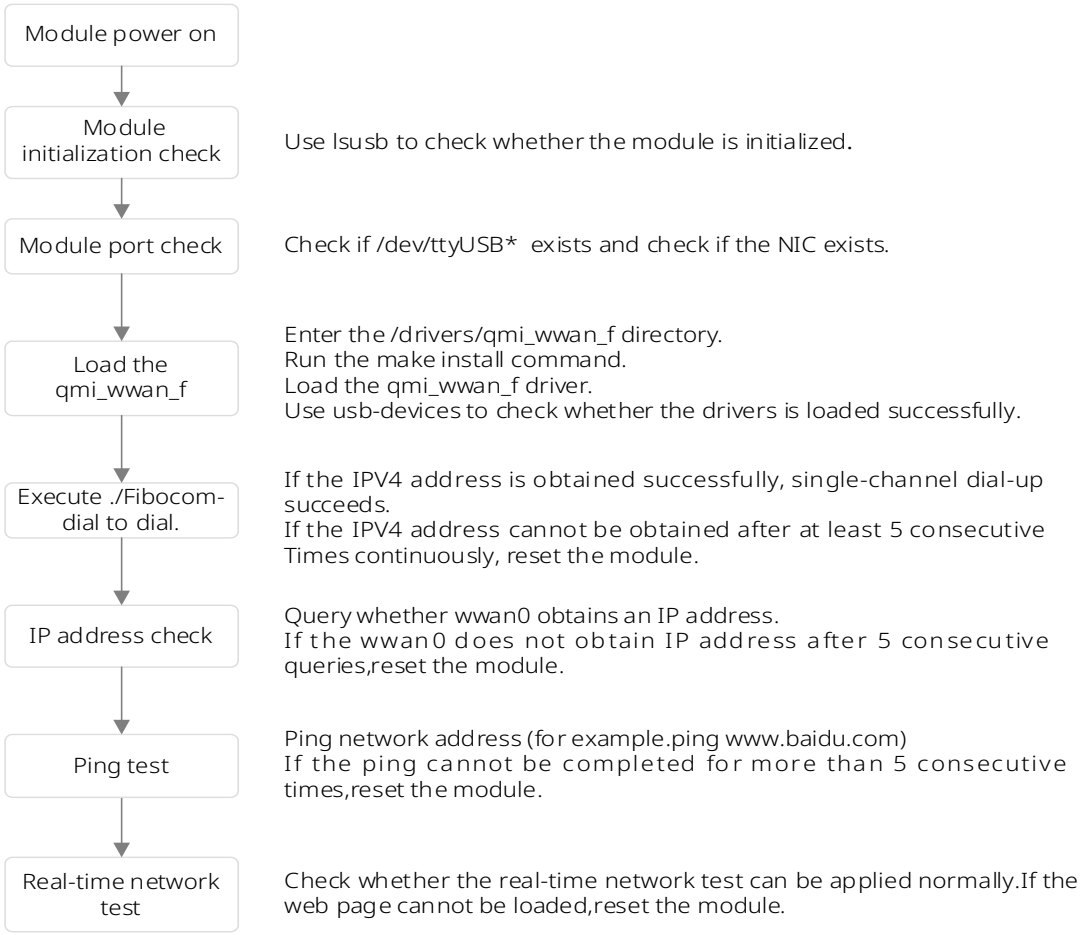

Figure 59. QMI_WWAN dial-up process

7.4 QMI_WWAN Single-channel Dial-up

After the QMI_WWAN driver is loaded, you can enter the **/fibocom-dial** directory (subject to actual directory). The Fibocom-dial tool transmits the QMI messages. For parameter types and functions, refer to Fibocom-dial parameter and application table in section 6.4.1.1. Run the **make** command to compile and generate the **fibocom-dial** binary file. Run the **./fibocom-dial** command to implement data dial-up and obtain the IPv4 address, as shown in the following figure.

```
[07-19_14:28:57:848] Fibocom-dial_Linux_Tool_V2.0.8
[07-19_14:28:57:848] ./fibocom-dial profile[1] = (null)/(null)/(null)/0, pincode = (null)
[07-19_14:28:57:849] socket[3] successfuly!
[07-19_14:28:57:849] Waiting client to connect...
[07-19_14:28:57:851] Find /sys/bus/usb/devices/1-1 idVendor=2cb7 idProduct=0104
[07-19_14:28:57:851] Find /sys/bus/usb/devices/1-1:1.4/net/wwan0
```

```
[07-19_14:28:57:852] Find usbnet_adapter = wwan0
[07-19_14:28:57:852] Find /sys/bus/usb/devices/1-1:1.4/usbmisc/cdc-wdm0
[07-19_14:28:57:852] Find qmichannel = /dev/cdc-wdm0
[07-19_14:28:57:852] qmichannel(/dev/cdc-wdm0) usbnet_adapter(wwan0)
[07-19_14:28:57:852] pcie mode
[07-19_14:28:57:852] ioctl(0x89f3, qmap_settings) failed: Operation not supported, rc=-1
[07-19_14:28:57:852] access /sys/class/net/wwan0/qmap_mode
[07-19_14:28:57:853] qmap_mode = 1, muxid = 0x81, qmap_netcard = wwan0
[07-19_14:28:57:853] ioctl(0x89f3, qmap_settings) failed: Operation not supported, rc=-1
[07-19_14:28:57:853] access /sys/class/net/wwan0/qmap_mode
[07-19_14:28:57:853] qmap_mode = 1, muxid = 0x81, qmap_netcard = wwan0
[07-19_14:28:57:853] qmap_mode=1
[07-19_14:28:57:865] cdc_wdm_fd = 9
[07-19_14:28:57:865] write triger_event: 4098 to qmidevice_control_fd
[07-19_14:28:57:955] QmiWwanGetClientID: QMIType = 1 clientid 14
[07-19_14:28:57:955] Get clientWDS = 14
[07-19_14:28:57:987] QmiWwanGetClientID: QMIType = 2 clientid 4
[07-19_14:28:57:987] Get clientDMS = 4
[07-19_14:28:58:019] QmiWwanGetClientID: QMIType = 3 clientid 2
[07-19_14:28:58:019] Get clientNAS = 2
[07-19_14:28:58:051] QmiWwanGetClientID: QMIType = 11 clientid 2
[07-19_14:28:58:051] Get clientUIM = 2
[07-19_14:28:58:083] QmiWwanGetClientID: QMIType = 26 clientid 1
[07-19_14:28:58:083] Get clientWDA = 1
[07-19_14:28:58:115] requestBaseBandVersion 89602.1000.00.04.08.21
[07-19_14:28:58:147] qmap_settings.rx_urb_size = 4096
[07-19_14:28:58:211] sim_select = 0
[07-19_14:28:58:243] curr_ints_sim1
[07-19_14:28:58:243] curr_ints->CardState is 1
[07-19_14:28:58:243] curr_ints->NumApp is 1
[07-19_14:28:58:243] AppType = 2
[07-19_14:28:58:243] requestGetSIMStatus SIMStatus: SIM_READY
[07-19_14:28:58:275] requestGetICCID DeviceICCID: 898600F0261831632928
[07-19_14:28:58:307] requestGetIMSI DeviceIMSI: 460026092319783
[07-19_14:28:58:339] requestGetProfile[1] ///0
[07-19_14:28:58:371] requestRegistrationState2 MCC: 460, MNC: 0, PS: Detached, DataCap: UNKNOW
```

```
[07-19_14:28:58:371] write signo: 12 to signal_control_fd
[07-19_14:28:58:371] epoll fd = 6, events = 0x0001
[07-19_14:28:58:371] get signo: 12
[07-19_14:28:58:403] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[07-19_14:28:58:403] usbnet_link_change :link:0
[07-19_14:28:58:403] enter udhcpc_stop
[07-19_14:28:58:403] enter fibo_set_driver_link_state
[07-19_14:28:58:404] if_link_down wwan0
[07-19_14:28:58:404] write signo: 10 to signal_control_fd
[07-19_14:28:58:404] epoll fd = 7, events = 0x0000
[07-19_14:28:58:404] epoll fd = 6, events = 0x0001
[07-19_14:28:58:404] get signo: 10
[07-19_14:28:58:404] usbnet_link_change :link:0
 [07-19_14:28:59:011] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached,
DataCap: LTE
[07-19_14:28:59:011] epoll fd = 6, events = 0x0000
[07-19_14:28:59:011] epoll fd = 7, events = 0x0001
[07-19_14:28:59:011] write signo: 12 to signal_control_fd
[07-19_14:28:59:011] epoll fd = 6, events = 0x0001
[07-19_14:28:59:011] get signo: 12
[07-19_14:28:59:075] usbnet_link_change :link:1
[07-19_14:28:59:107] enter fibo_set_driver_link_state
[07-19_14:28:59:107] if_link_up wwan0
[07-19_14:28:59:108] IPv4 MTU: 1500
[07-19_14:28:59:108] IPv4 Address: 10.9.199.226
[07-19_14:28:59:108] IPv4 Netmask: 30
[07-19_14:28:59:108] IPv4 Gateway: 10.9.199.225
[07-19_14:28:59:108] IPv4 DNS1: 211.137.130.2
[07-19_14:28:59:108] IPv4 DNS2: 211.137.130.4
```

After the dial-up is successful, run the ifconfig –a command to check whether the wwan0 interface has obtained the IP address. As shown in the following figure, the IP address is the same as that obtained after successful dial-up:

```
wwan0: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet 10.9.199.226  netmask 255.255.255.252
        inet6 fe80::844:72ff:fe90:32b4  prefixlen 64  scopeid 0x20<link>
        ether 0a:44:72:90:32:b4  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 217 (217.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3  bytes 196 (196.0 B)
```

```
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ping packet sending test:

```
ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=52 time=105 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=75.9 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
```

7.5 QMI_WWAN Multi-channel Dial-up

Multi-channel dial-up requires that multiple PDNs be configured before the driver is loaded. Open the source code qmi_wwan_f.c file and change qmap_mode to multi-channel, for example, changing it to 4 as shown below, which means that the system supports 4-channel dial-up (currently, the system supports a maximum of 4 channels for dial-up).

```
static uint __read_mostly qmap_mode = 4;
```

After the modification is completed, recompile and install the qmi_wwan_f driver.

After the installation is completed, you can use the ifconfig -a command to view the four virtual NICs wwan0.1, wwan0.2, wwan0.3, and wwan0.4 currently mapped to the physical NIC wwan0.

Enter the directory where the dial-up tool is located, run the make command to compile the driver and generate the fibo_qmimsg_server and multi-pdn-manager binary files. Run "./fibo_qmimsg_server &" to start the QMI message service process.

```
dial_Linux_Tool_V2.0.8/src# ./fibo_qmimsg_server  &
Find /sys/bus/usb/devices/1-1 idVendor=2cb7 idProduct=0104
Find /sys/bus/usb/devices/1-1:1.4/usbmisc/cdc-wdm0
Will use cdc-wdm /dev/cdc-wdm0
qmi_proxy_init enter
qmi_proxy_loop enter thread_id 140040038012672
send_qmi_timeout ret=110
link_prot 2
ul_data_aggregation_protocol 5
dl_data_aggregation_protocol 5
dl_data_aggregation_max_datagrams 63
dl_data_aggregation_max_size 16384
ul_data_aggregation_max_datagrams 1
ul_data_aggregation_max_size 4096
qmi_proxy_init finished, rx_urb_size is 16384
local server: fibo_qmimsg_server sockfd = 4
qmi_start_server: qmi_proxy_server_fd = 4
```

Run **./multi-pdn-manager** to perform data dial-up. For details, see section 6.4.2. Use NIC wwan0.1 of channel 1 and profile1 of channel 1 to dial up, as shown below:

```
./multi-pdn-manager
 [07-19_11:29:01:466] Start Fibocom multi-pdn-manager!
dev: /dev/ttyUSB1
rate:115200
sendbuffer:at+gtpcie=3
at+gtpcie=3
+GTPCIE: RC
OK
[07-19_11:29:01:509] access /sys/class/net/wwan0/qmap_mode
[07-19_11:29:01:509] access /sys/class/net/wwan0/qmap_mode
[07-19_11:29:01:511] Fibocom multi-pdn-manager 1.0.4
[07-19_11:29:01:511] Fibocom manager current qmap_num is 4
[07-19_11:29:01:511] Fibocom manager sub-instance1 Disconnected
[07-19_11:29:01:511] Fibocom manager sub-instance2 Disconnected
[07-19_11:29:01:511] Fibocom manager sub-instance3 Disconnected
[07-19_11:29:01:511] Fibocom manager sub-instance4 Disconnected
Please select an action[0-9]
1.show profile setting
2.set profile setting
3.connect to network
4.disconnect to network
5.show connect log
6.reload profile setting
7.save profile setting
8.show max instance number
9.set max instance number
0.exit
input [0-9]: 3
connect visual net interface use:1
connect profile use:1
[07-19_11:29:04:555] Start connect network use instance 1 profile 1
fibocom-dial -N 4 -n 1 -m 1 -s 1234567890 qwertyuio asdfghjkl 2 -4 -f instance1.txt
Press any key to continus[07-19_11:29:04:561] exec pid 11717
[07-19_11:29:04:562] Fibocom-dial_Linux_Tool_V2.0.8
[07-19_11:29:04:562]  fibocom-dial  profile[1]  =  1234567890/qwertyuio/asdfghjkl/2,
pincode = (null)
[07-19_11:29:04:562] socket[5] successfuly!
```



```
[07-19_11:29:04:562] Waiting client to connect...
[07-19_11:29:04:563] Find /sys/bus/usb/devices/1-1 idVendor=2cb7 idProduct=0104
[07-19_11:29:04:563] Find /sys/bus/usb/devices/1-1:1.4/net/wwan0
[07-19_11:29:04:563] Find usbnet_adapter = wwan0
[07-19_11:29:04:563] Find /sys/bus/usb/devices/1-1:1.4/usbmisc/cdc-wdm0
[07-19_11:29:04:563] Find qmichannel = /dev/cdc-wdm0
[07-19_11:29:04:563] qmichannel(/dev/cdc-wdm0) usbnet_adapter(wwan0)
[07-19_11:29:04:563] pcie mode
[07-19_11:29:04:563] ioctl(0x89f3, qmap_settings) failed: Operation not supported,
rc=-1
[07-19_11:29:04:563] access /sys/class/net/wwan0/qmap_mode
[07-19_11:29:04:563] qmap_mode = 4, muxid = 0x81, qmap_netcard = wwan0.1
[07-19_11:29:04:563] ioctl(0x89f3, qmap_settings) failed: Operation not supported,
rc=-1
[07-19_11:29:04:563] access /sys/class/net/wwan0/qmap_mode
[07-19_11:29:04:563] qmap_mode = 4, muxid = 0x81, qmap_netcard = wwan0.1
[07-19_11:29:04:563] connect to fibo_qmimsg_server sockfd = 11
[07-19_11:29:04:563] cdc_wdm_fd = 11
[07-19_11:29:04:563] write triger_event: 4098 to qmidevice_control_fd
[07-19_11:29:04:621] QmiWwanGetClientID: QMIType = 1 clientid 14
[07-19_11:29:04:621] Get clientWDS = 14
[07-19_11:29:04:652] QmiWwanGetClientID: QMIType = 2 clientid 1
[07-19_11:29:04:652] Get clientDMS = 1
[07-19_11:29:04:684] QmiWwanGetClientID: QMIType = 3 clientid 2
[07-19_11:29:04:684] Get clientNAS = 2
[07-19_11:29:04:716] QmiWwanGetClientID: QMIType = 11 clientid 2
[07-19_11:29:04:716] Get clientUIM = 2
[07-19_11:29:04:749] requestBaseBandVersion 89602.1000.00.04.08.21
[07-19_11:29:04:812] sim_select = 0
[07-19_11:29:04:844] curr_ints_sim1
[07-19_11:29:04:844] curr_ints->CardState is 1
[07-19_11:29:04:845] curr_ints->NumApp is 1
[07-19_11:29:04:845] AppType = 2
[07-19_11:29:04:845] requestGetSIMStatus SIMStatus: SIM_READY
[07-19_11:29:04:875] requestGetICCID DeviceICCID: 898600F0261831632928
[07-19_11:29:04:908] requestGetIMSI DeviceIMSI: 460026092319783
[07-19_11:29:04:908] requestSetProfile[1] 1234567890/qwertyuio/asdfghjkl/2
[07-19_11:29:04:972] requestGetProfile[1] 1234567890/qwertyuio/asdfghjkl/2
[07-19_11:29:05:004] requestRegistrationState2 MCC: 460, MNC: 0, PS: Attached,
```

```
DataCap: LTE
[07-19_11:29:05:004] write signo: 12 to signal_control_fd
[07-19_11:29:05:004] epoll fd = 8, events = 0x0001
[07-19_11:29:05:004] get signo: 12
[07-19_11:29:05:036] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[07-19_11:29:05:036] usbnet_link_change :link:0
[07-19_11:29:05:036] enter udhcpc_stop
[07-19_11:29:05:036] enter fibo_set_driver_link_state
[07-19_11:29:05:037] if_link_down wwan0.1
[07-19_11:29:05:037] write signo: 10 to signal_control_fd
[07-19_11:29:05:037] epoll fd = 9, events = 0x0000
[07-19_11:29:05:037] epoll fd = 8, events = 0x0001
[07-19_11:29:05:037] get signo: 10
[07-19_11:29:05:037] usbnet_link_change :link:0
[07-19_11:29:05:069]  requestRegistrationState2  MCC:  460,  MNC:  0,  PS:  Attached,
DataCap: LTE
[07-19_11:29:05:100] requestSetupDataCall WdsConnectionIPv4Handle: 0x360397e0
[07-19_11:29:05:132] write triger_event: 4101 to qmidevice_control_fd
[07-19_11:29:05:164] epoll fd = 9, events = 0x0000
[07-19_11:29:05:165] epoll fd = 8, events = 0x0000
[07-19_11:29:05:165] epoll fd = 9, events = 0x0001
[07-19_11:29:05:165] write signo: 12 to signal_control_fd
[07-19_11:29:05:165] epoll fd = 8, events = 0x0001
[07-19_11:29:05:165] get signo: 12
[07-19_11:29:05:229] usbnet_link_change :link:1
[07-19_11:29:05:261] enter fibo_set_driver_link_state
[07-19_11:29:05:261] ifconfig wwan0 up
[07-19_11:29:05:269] if_link_up wwan0.1
[07-19_11:29:05:270] IPv4 MTU: 1500
[07-19_11:29:05:270] IPv4 Address: 10.84.69.148
[07-19_11:29:05:270] IPv4 Netmask: 29
[07-19_11:29:05:270] IPv4 Gateway: 10.84.69.149
[07-19_11:29:05:271] IPv4 DNS1: 211.137.130.2
[07-19_11:29:05:271] IPv4 DNS2: 211.137.130.4
[07-19_11:29:05:271] if_link_up wwan0.1
```

The above dial-up log shows that the dial-up is successfully implemented using virtual NIC wwan0.1 of channel 1, and the IP address is successfully obtained. Then, use virtual NIC wwan0.2 of channel 2 and profile2 of channel 2 for dial-up. The dial-up process is as follows:

```
input [0-9]: 3
```

```
connect visual net interface use:2
connect profile use:2
[07-19_11:49:09:141] Start connect network use instance 2 profile 2
fibocom-dial -N 4 -n 2 -m 2 -s ctnet -4 -f instance2.txt
Press any key to continus[07-19_11:49:09:147] exec pid 12115
[07-19_11:49:09:149] Fibocom-dial_Linux_Tool_V2.0.8
[07-19_11:49:09:149] fibocom-dial profile[2] = ctnet///0, pincode = (null)
[07-19_11:49:09:150] socket[5] successfuly!
[07-19_11:49:09:150] Waiting client to connect...
[07-19_11:49:09:152] Find /sys/bus/usb/devices/1-1 idVendor=2cb7 idProduct=0104
[07-19_11:49:09:152] Find /sys/bus/usb/devices/1-1:1.4/net/wwan0
[07-19_11:49:09:152] Find usbnet_adapter = wwan0
[07-19_11:49:09:152] Find /sys/bus/usb/devices/1-1:1.4/usbmisc/cdc-wdm0
[07-19_11:49:09:152] Find qmichannel = /dev/cdc-wdm0
[07-19_11:49:09:152] qmichannel(/dev/cdc-wdm0) usbnet_adapter(wwan0)
[07-19_11:49:09:152] pcie mode
[07-19_11:49:09:153] ioctl(0x89f3, qmap_settings) failed: Operation not supported,
rc=-1
[07-19_11:49:09:153] access /sys/class/net/wwan0/qmap_mode
[07-19_11:49:09:153] qmap_mode = 4, muxid = 0x82, qmap_netcard = wwan0.2
[07-19_11:49:09:153] ioctl(0x89f3, qmap_settings) failed: Operation not supported,
rc=-1
[07-19_11:49:09:153] access /sys/class/net/wwan0/qmap_mode
[07-19_11:49:09:153] qmap_mode = 4, muxid = 0x82, qmap_netcard = wwan0.2
[07-19_11:49:09:154] connect to fibo_qmimsg_server sockfd = 11
[07-19_11:49:09:154] cdc_wdm_fd = 11
[07-19_11:49:09:154] write triger_event: 4098 to qmidevice_control_fd
[07-19_11:49:09:205] QmiWwanGetClientID: QMIType = 1 clientid 15
[07-19_11:49:09:205] Get clientWDS = 15
[07-19_11:49:09:237] QmiWwanGetClientID: QMIType = 2 clientid 2
[07-19_11:49:09:237] Get clientDMS = 2
[07-19_11:49:09:268] QmiWwanGetClientID: QMIType = 3 clientid 3
[07-19_11:49:09:268] Get clientNAS = 3
[07-19_11:49:09:301] QmiWwanGetClientID: QMIType = 11 clientid 3
[07-19_11:49:09:301] Get clientUIM = 3
[07-19_11:49:09:332] requestBaseBandVersion 89602.1000.00.04.08.21
[07-19_11:49:09:397] sim_select = 0
[07-19_11:49:09:429] curr_ints_sim1
[07-19_11:49:09:429] curr_ints->CardState is 1
```

```
[07-19_11:49:09:429] curr_ints->NumApp is 1
[07-19_11:49:09:429] AppType = 2
[07-19_11:49:09:429] requestGetSIMStatus SIMStatus: SIM_READY
[07-19_11:49:09:461] requestGetICCID DeviceICCID: 898600F0261831632928
[07-19_11:49:09:493] requestGetIMSI DeviceIMSI: 460026092319783
[07-19_11:49:09:493] requestSetProfile[2] ctnet///0
[07-19_11:49:09:557] requestGetProfile[2] ctnet///0
[07-19_11:49:09:589]  requestRegistrationState2  MCC:  460,  MNC:  0,  PS:  Attached,
DataCap: LTE
[07-19_11:49:09:589] write signo: 12 to signal_control_fd
[07-19_11:49:09:589] epoll fd = 8, events = 0x0001
[07-19_11:49:09:589] get signo: 12
[07-19_11:49:09:621] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[07-19_11:49:09:621] usbnet_link_change :link:0
[07-19_11:49:09:621] enter udhcpc_stop
[07-19_11:49:09:621] enter fibo_set_driver_link_state
[07-19_11:49:09:622] if_link_down wwan0.2
[07-19_11:49:09:622] write signo: 10 to signal_control_fd
[07-19_11:49:09:622] epoll fd = 9, events = 0x0000
[07-19_11:49:09:622] epoll fd = 8, events = 0x0001
[07-19_11:49:09:622] get signo: 10
[07-19_11:49:09:622] usbnet_link_change :link:0
[07-19_11:49:09:653]  requestRegistrationState2  MCC:  460,  MNC:  0,  PS:  Attached,
DataCap: LTE
[07-19_11:49:18:839] write signo: 12 to signal_control_fd
[07-19_11:49:18:840] epoll fd = 8, events = 0x0001
[07-19_11:49:18:840] get signo: 12
[07-19_11:49:18:874] usbnet_link_change :link:1
[07-19_11:49:18:874] epoll fd = 9, events = 0x0000
[07-19_11:49:19:131] requestSetupDataCall WdsConnectionIPv4Handle: 0x361b1650
[07-19_11:49:19:163] write triger_event: 4101 to qmidevice_control_fd
[07-19_11:49:19:195] epoll fd = 9, events = 0x0000
[07-19_11:49:19:195] epoll fd = 8, events = 0x0000
[07-19_11:49:19:195] epoll fd = 9, events = 0x0001
[07-19_11:49:19:195] write signo: 12 to signal_control_fd
[07-19_11:49:19:195] epoll fd = 8, events = 0x0001
[07-19_11:49:19:195] get signo: 12
[07-19_11:49:19:259] usbnet_link_change :link:1
[07-19_11:49:19:291] enter fibo_set_driver_link_state
```

```
[07-19_11:49:19:291] ifconfig wwan0 up
[07-19_11:49:19:298] if_link_up wwan0.2
[07-19_11:49:19:298] IPv4 MTU: 1500
[07-19_11:49:19:299] IPv4 Address: 10.82.57.76
[07-19_11:49:19:299] IPv4 Netmask: 29
[07-19_11:49:19:299] IPv4 Gateway: 10.82.57.77
[07-19_11:49:19:299] IPv4 DNS1: 211.137.130.2
[07-19_11:49:19:299] IPv4 DNS2: 211.137.130.4
[07-19_11:49:19:299] if_link_up wwan0.2
[07-19_11:49:21:301] epoll fd = 9, events = 0x0000
```

The above dial-up log shows that the dial-up is successfully implemented using virtual NIC wwan0.2 of channel 2, and the IP address is successfully obtained. After the dial-up is successful, open another window and run **ifconfig -a** to check the current NIC state. You can see that both wwan0.1 and wwan0.2 have obtained the corresponding IP addresses, as shown below:

```
wwan0.1: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet 10.84.69.148  netmask 255.255.255.248
        ether 0a:44:72:90:32:b4  txqueuelen 1000  (Ethernet)
        RX packets 7  bytes 917 (917.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7  bytes 488 (488.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
wwan0.2: flags=193<UP,RUNNING,NOARP>  mtu 1500
        inet 10.82.57.76  netmask 255.255.255.248
        ether 0a:44:72:90:32:b4  txqueuelen 1000  (Ethernet)
        RX packets 18  bytes 2474 (2.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1272 (1.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

After the NICs obtain the IP addresses, perform a ping test, as shown below. All connections can be successfully pinged:

```
root@ght-Lenovo-V130-14IKB:~/lcc/qmi_wwan_f/Fibocom_QMI_WWAN_Driver_V1.0.3# ping 8.8.8.8 -I wwan0.1
PING 8.8.8.8 (8.8.8.8) from 10.84.69.148 wwan0.1: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=111 time=140 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=111 time=78.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=111 time=86.1 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=111 time=93.4 ms
root@ght-Lenovo-V130-14IKB:~/lcc/qmi_wwan_f/Fibocom_QMI_WWAN_Driver_V1.0.3# ping 8.8.8.8 -I wwan0.2
```

```
PING 8.8.8.8 (8.8.8.8) from 10.82.57.76 wwan0.2: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=2 ttl=52 time=232 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=52 time=85.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=52 time=106 ms
```

8 PPP Dial-up

8.1 PPP Overview

Point-to-Point Protocol (PPP) is a link-layer protocol designed for simple links that transmit data packets between peer units. This kind of links provides full-duplex operations and transfers packets in order. It is designed to establish point-to-point connections for data transmission through dial-up or a dedicated line. It is a common solution for simple connections between hosts, bridges, and routers.

8.2 Confirming USB Enumeration

Because PPP dial-up requires the AT ports or Modem ports, PPP dial-up can be performed as long as AT ports or Modem ports exist in the enumeration.

8.3 Integrating USB Driver

Refer to section 2.2 to modify drivers/usb/serial/option.c (no modification is required for Eigencomm platform). The zero-length packet mechanism must be modified. For details, see section 2. 4.

8.4 Confirming Linux Environment

8.4.1 Checking if pppd Is Installed

Generally, the Linux system already supports pppd, which can be queried through the following method. If it is not installed, download the latest version from http://ppp.samba.org/.

```
[root@fedoral whl]# pppd --help
pppd version 2.4.2
Usage: pppd [ options ], where options are:
        <device>        Communicate over the named device
        <speed>         Set the baud rate to <speed>
        <loc>:<rem>     Set the local and/or remote interface IP
                        addresses.  Either one may be omitted.
        asyncmap <n>    Set the desired async map to hex <n>
        auth            Require authentication from peer
        connect <p>     Invoke shell command <p> to set up the serial line
        crtscts         Use hardware RTS/CTS flow control
        defaultroute    Add default route through interface
        file <f>        Take options from file <f>
```

```
        modem           Use modem control lines
        mru <n>         Set MRU value to <n> for negotiation
See pppd(8) for more options.
```

8.4.2 PPP Dial-up Script

Take wcdma as an example. The specific operation is as follows.

1. Obtain the /etc/ppp/peers/wcdma option file required for pppd connections.

Please note the differences in punctuations such as double quotation marks between Windows and Linux systems.

```
/dev/ttyUSB1
115200
noauth
nocrtscts
nocdtrcts
local
debug
nobsdcomp
nodeflate
nodetach
novj
defaultroute
noipdefault
usepeerdns
ipcp-accept-local
ipcp-accept-remote
mru 1280
mtu 1500
#lock
connect '/usr/sbin/chat -s -v -f /etc/ppp/chat/wcdma-connect-chat'
```

2. Obtain the connection establishment rule file /etc/ppp/chat/cmnet-connect-chat.

```
TIMEOUT 15
ABORT "DELAYED"
ABORT "BUSY"
ABORT "ERROR"
ABORT "NO DIALTONE"
ABORT "NO CARRIER"
```

```
TIMEOUT 40
''  \rAT
OK ATS0=0
OK ATE0V1
OK AT+CGDCONT=1,"IP","3gnet"
OK ATDT*99***1#
CONNECT ''
```

The dial-up command is used as follows:

```
[root@fedoral whl]# pppd call wcdma
```

In the wcdma-connect-chat script, OK AT+CGDCONT=1,"IP","3gnet" is briefly described as follows:

To set the PDP type to IPv4&IPv6 or IPv6, replace the IP address with IPv4&IPv6 or IPv6 address.

To modify the APN name, replace 3gnet with the corresponding carrier's APN (consulting local carrier or Fibocom FAE). To change the carrier to China Mobile, replace 3gnet with cmnet.

8.5 PPP Authentication Mode

Generally, PPP connections require identity authentication. There are two authentication methods, namely, PAP authentication and CHAP authentication. The username and password required for authentication are stored in the pap-secrets and chap-secrets scripts. When authentication is required, by specifying the authentication method as PAP or CHAP in the options script, the PPP module will read the username and password from the pap-secrets and chap-secrets scripts, attach them to the PPP authentication packets, and send them to the server for identity authentication.

8.5.1 PAP Authentication

Modify the authentication file pap-secrets according to the requirements. The file is located in the /etc/ppp/pap-secrets directory.

```
/etc/ppp # cat pap-secrets
# Secrets for authentication using PAP
# client server secret IP addresses
myclient  ISP-server  mypassword  *
```

The parameters are as follows:

- Myclient --- Caller's PAP user name.
- ISP-server --- Remote computer name.
- mypassword --- Caller's PAP password.

- * --- IP address associated with the caller. An asterisk (*) indicates any IP address.

For example: cmnet * cmnet * (Configure according to the actual user name and password of the network.)

8.5.2 CHAP Authentication

Modify the authentication file **chap-secrets** according to the requirements. The file is located in the **/etc/ppp/chap-secrets** directory.

```
/etc/ppp # cat chap-secrets
# Secrets for authentication using CHAP
# client server secret IP addresses
myclient ISP-server secret5748 *
```

The parameters are as follows:

- Myclient --- Caller's CHAP user.
- ISP-server --- Remote computer name.
- secret5748 --- Caller's CHAP secret.
- * --- IP address associated with the caller. An asterisk (*) indicates any IP address.

If you can perform dial-up successfully before configuring the authentication method, but cannot perform dial-up after configuring the authentication method, do as follows: When setting the PAP authentication method, add the corresponding username and password to pap-secrets, and add noauth and user "username" to the **/etc/ppp/peers/wcdma** file of pppd. When setting the CHAP authentication method, add the corresponding username and password to chap-secrets, and add noauth and user "username" to the **/etc/ppp/peers/wcdma** file of pppd. The corresponding user names must be consistent. The corresponding chap-secrets and pap-secrets scripts should be placed in the directories at the same level. For the corresponding parameters in the **wcdma** script, refer to the explanation in the **options** script (saved in the same directory). The document only provides some scripts for reference. For specific configuration, refer to the parameter description in the **options** script.

8.6 PPP Dial-up Process

Enter **pppd call wcdma** in the terminal.

Run the **su** command to switch to the **root** account for dial-up.

After the dial-up is successful, run the **cat /var/log/message** command to query logs. The normal situation is as follows:

```
root@ght-ThinkPad-E470:/etc/ppp/peers# pppd call wcdma
timeout set to 5 seconds
abort on (NO CARRIER)
abort on (ERROR)
abort on (+CMS ERROR:)
```

```
abort on (+CME ERROR:)
abort on (NO ANSWER)
abort on (NO DIALTONE)
abort on (COMMAND NOT SUPPORT)
send (AT+CGDCONT=1,"ip","3gnet"^M)
expect (OK)
AT+CGDCONT=1,"ip","3gnet"^M^M
OK
 -- got it

send (ATD*99***1#^M)
expect (CONNECT)
^M
ATD*99***1#^M^M
CONNECT
 -- got it

Script chat -s -v -f /etc/ppp/peers/chat_wcdma finished (pid 15709), status = 0x0
Serial connection established.
using channel 8
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB0
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x56ac1f37> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <auth chap MD5> <magic 0x633068e> <pcomp>
<accomp>]
sent [LCP ConfNak id=0x1 <auth pap>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x56ac1f37> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x2 <asyncmap 0x0> <auth pap> <magic 0x633068e> <pcomp>
<accomp>]
sent [LCP ConfAck id=0x2 <asyncmap 0x0> <auth pap> <magic 0x633068e> <pcomp>
<accomp>]
sent [LCP EchoReq id=0x0 magic=0x56ac1f37]
sent [PAP AuthReq id=0x1 user="ght-ThinkPad-E470" password=<hidden>]
rcvd [LCP EchoRep id=0x0 magic=0x633068e]
rcvd [PAP AuthAck id=0x1 "Login ok"] 00
Remote message: Login ok
PAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
sent [IPV6CP ConfReq id=0x1 <addr fe80::9020:ec48:82ce:a810>]
```

```
rcvd [IPCP ConfReq id=0x1 <addr 192.168.0.1>] 00
sent [IPCP ConfAck id=0x1 <addr 192.168.0.1>]
rcvd [IPV6CP ConfReq id=0x1 <addr fe80::1068:fa37:05a9:2d45>] 00
sent [IPV6CP ConfAck id=0x1 <addr fe80::1068:fa37:05a9:2d45>]
rcvd [IPCP ConfNak id=0x1 <addr 10.46.153.76> <ms-dns1 61.134.1.6> <ms-dns2 218.20.19.40>] 00
sent [IPCP ConfReq id=0x2 <addr 10.46.153.76> <ms-dns1 61.134.1.6> <ms-dns2 218.20.19.40>]
rcvd [IPV6CP ConfAck id=0x1 <addr fe80::9020:ec48:82ce:a810>] 01
local  LL address fe80::9020:ec48:82ce:a810
remote LL address fe80::1068:fa37:05a9:2d45
Script /etc/ppp/ipv6-up started (pid 15720)
rcvd [IPCP ConfAck id=0x2 <addr 10.46.153.76> <ms-dns1 61.134.1.6> <ms-dns2 218.20.19.40>] 00
local  IP address 10.46.153.76
remote IP address 192.168.0.1
primary   DNS address 61.134.1.6
secondary DNS address 218.20.19.40
Script /etc/ppp/ip-up started (pid 15722)
Script /etc/ppp/ipv6-up finished (pid 15720), status = 0x0
Script /etc/ppp/ip-up finished (pid 15722), status = 0x0
```

You can also run the ifconfig command to query the network interface. The normal situation is as follows, indicating successful dial-up.

```
[root@fedoral whl]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:94:ef:0a:2d:f0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:48351160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48351160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:45562895173 (45.5 GB)  TX bytes:45562895173 (45.5 GB)
ppp0      Link encap:  Point-to-Point Protocol
```

```
inet addr:172.16.178.254  P-t-P:172.16.178.254  Mask:255.255.255.255
UP BROADCAST MULTICAST  MTU:1280  Metric:1
RX packets:3 errors:0 dropped:0 overruns:0 frame:0
TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:3
RX bytes:54 (54.0 B)  TX bytes:76 (76.0 B)
```

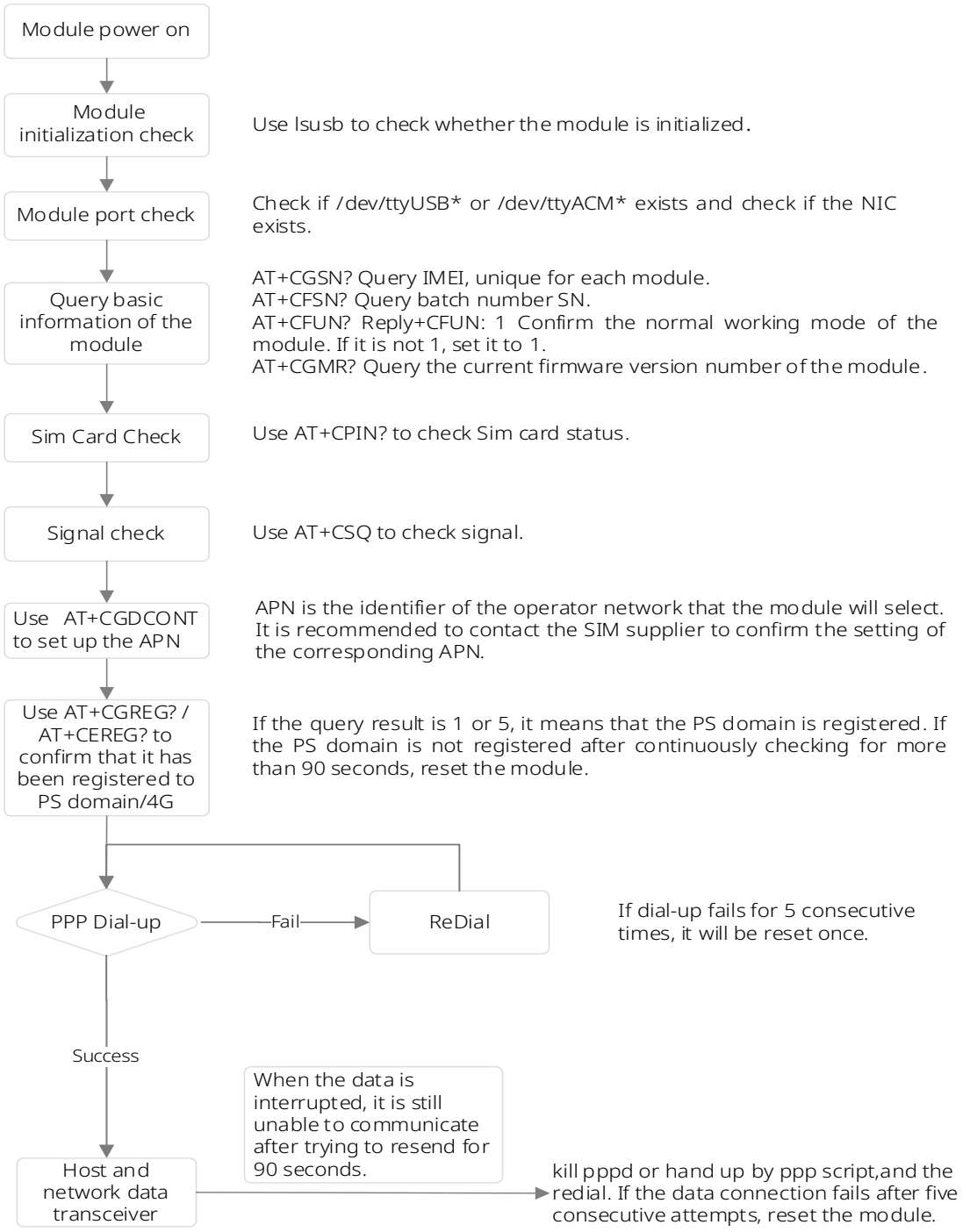

Figure 60. PPP dial-up flowchart

If ttyUSB1 is used for PPP dial-up, the port does not respond to common AT commands after PPP dial-up. The above AT commands are sent through other AT ports.