cdc_mbim - Driver for CDC MBIM Mobile Broadband modems

The cdc_mbim driver supports USB devices conforming to the "Universal Serial Bus Communications Class Subclass Specification for Mobile Broadband Interface Model" [1], which is a further development of "Universal Serial Bus Communications Class Subclass Specifications for Network Control Model Devices" [2] optimized for Mobile Broadband devices, aka "3G/LTE modems".

Command Line Parameters

The cdc_mbim driver has no parameters of its own. But the probing behaviour for NCM 1.0 backwards compatible MBIM functions (an "NCM/MBIM function" as defined in section 3.2 of [1]) is affected by a cdc_ncm driver parameter:

prefer_mbim

Type: Boolean
Valid Range: N/Y (0-1)
Default Value: Y (MBIM is preferred)

This parameter sets the system policy for NCM/MBIM functions. Such functions will be handled by either the cdc_ncm driver or the cdc_mbim driver depending on the prefer_mbim setting. Setting prefer_mbim=N makes the cdc_mbim driver ignore these functions and lets the cdc_ncm driver handle them instead.

The parameter is writable, and can be changed at any time. A manual unbind/bind is required to make the change effective for NCM/MBIM functions bound to the "wrong" driver

Basic usage

MBIM functions are inactive when unmanaged. The cdc_mbim driver only provides a userspace interface to the MBIM control channel, and will not participate in the management of the function. This implies that a userspace MBIM management application always is required to enable a MBIM function.

Such userspace applications includes, but are not limited to:

- mbimcli (included with the libmbim [3] library), and
- ModemManager [4]

Establishing a MBIM IP session reequires at least these actions by the management application:

- open the control channel
- configure network connection settings
- connect to network
- configure IP interface

Management application development

The driver <-> userspace interfaces are described below. The MBIM control channel protocol is described in [1].

MBIM control channel userspace ABI

/dev/cdc-wdmX character device

The driver creates a two-way pipe to the MBIM function control channel using the cdc-wdm driver as a subdriver. The userspace end of the control channel pipe is a /dev/cdc-wdmX character device.

The cdc_mbim driver does not process or police messages on the control channel. The channel is fully delegated to the userspace management application. It is therefore up to this application to ensure that it complies with all the control channel requirements in [1].

The cdc-wdmX device is created as a child of the MBIM control interface USB device. The character device associated with a specific MBIM function can be looked up using sysfs. For example:

```
bjorn@nemi:~$ ls /sys/bus/usb/drivers/cdc_mbim/2-4:2.12/usbmisc
cdc-wdm0

bjorn@nemi:~$ grep . /sys/bus/usb/drivers/cdc_mbim/2-4:2.12/usbmisc/cdc-wdm0/dev
180:0
```

USB configuration descriptors

The wMaxControlMessage field of the CDC MBIM functional descriptor limits the maximum control message size. The managament application is responsible for negotiating a control message size complying with the requirements in section 9.3.1 of [1], taking this descriptor field into consideration.

The userspace application can access the CDC MBIM functional descriptor of a MBIM function using either of the two USB configuration descriptor kernel interfaces described in [6] or [7].

See also the ioctl documentation below.

Fragmentation

The userspace application is responsible for all control message fragmentation and defragmentaion, as described in section 9.5 of [1].

/dev/cdc-wdmX write()

The MBIM control messages from the management application *must not* exceed the negotiated control message size.

/dev/cdc-wdmX read()

The management application *must* accept control messages of up the negotiated control message size.

/dev/cdc-wdmX ioctl()

IOCTL_WDM_MAX_COMMAND: Get Maximum Command Size This ioctl returns the wMaxControlMessage field of the CDC MBIM functional descriptor for MBIM devices. This is intended as a convenience, eliminating the need to parse the USB descriptors from userspace.

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/usb/cdc-wdm.h>
int main()
{
        __u16 max;
        int fd = open("/dev/cdc-wdm0", O_RDWR);
        if (!ioctl(fd, IOCTL_WDM_MAX_COMMAND, &max))
                printf("wMaxControlMessage is %d\n", max);
}
```

Custom device services

The MBIM specification allows vendors to freely define additional services. This is fully supported by the cdc_mbim driver.

Support for new MBIM services, including vendor specified services, is implemented entirely in userspace, like the rest of the MBIM control protocol

New services should be registered in the MBIM Registry [5].

MBIM data channel userspace ABI

wwanY network device

The cdc_mbim driver represents the MBIM data channel as a single network device of the "wwan" type. This network device is initially mapped to MBIM IP session 0.

Multiplexed IP sessions (IPS)

MBIM allows multiplexing up to 256 IP sessions over a single USB data channel. The cdc_mbim driver models such IP sessions as 802.1q VLAN subdevices of the master wwanY device, mapping MBIM IP session Z to VLAN ID Z for all values of Z greater than 0.

The device maximum Z is given in the MBIM_DEVICE_CAPS_INFO structure described in section 10.5.1 of [1].

The userspace management application is responsible for adding new VLAN links prior to establishing MBIM IP sessions where the SessionId is greater than 0. These links can be added by using the normal VLAN kernel interfaces, either ioctl or netlink.

For example, adding a link for a MBIM IP session with SessionId 3:

```
ip link add link wwan0 name wwan0.3 type vlan id 3
```

The driver will automatically map the "wwan0.3" network device to MBIM IP session 3.

Device Service Streams (DSS)

MBIM also allows up to 256 non-IP data streams to be multiplexed over the same shared USB data channel. The cdc_mbim driver

models these sessions as another set of 802.1q VLAN subdevices of the master wwanY device, mapping MBIM DSS session A to VLAN ID (256 + A) for all values of A.

The device maximum A is given in the MBIM_DEVICE_SERVICES_INFO structure described in section 10.5.29 of [1].

The DSS VLAN subdevices are used as a practical interface between the shared MBIM data channel and a MBIM DSS aware userspace application. It is not intended to be presented as-is to an end user. The assumption is that a userspace application initiating a DSS session also takes care of the necessary framing of the DSS data, presenting the stream to the end user in an appropriate way for the stream type.

The network device ABI requires a dummy ethernet header for every DSS data frame being transported. The contents of this header is arbitrary, with the following exceptions:

- TX frames using an IP protocol (0x0800 or 0x86dd) will be dropped
- RX frames will have the protocol field set to ETH_P_802_3 (but will not be properly formatted 802.3 frames)
- RX frames will have the destination address set to the hardware address of the master device

The DSS supporting userspace management application is responsible for adding the dummy ethernet header on TX and stripping it on RX.

This is a simple example using tools commonly available, exporting DssSessionId 5 as a pty character device pointed to by a /dev/nmea symlink:

```
ip link add link wwan0 name wwan0.dss5 type vlan id 261
ip link set dev wwan0.dss5 up
socat INTERFACE:wwan0.dss5,type=2 PTY:,echo=0,link=/dev/nmea
```

This is only an example, most suitable for testing out a DSS service. Userspace applications supporting specific MBIM DSS services are expected to use the tools and programming interfaces required by that service.

Note that adding VLAN links for DSS sessions is entirely optional. A management application may instead choose to bind a packet socket directly to the master network device, using the received VLAN tags to map frames to the correct DSS session and adding 18 byte VLAN ethernet headers with the appropriate tag on TX. In this case using a socket filter is recommended, matching only the DSS VLAN subset. This avoid unnecessary copying of unrelated IP session data to userspace. For example:

```
static struct sock_filter dssfilter[] = {
      /* use special negative offsets to get VLAN tag */
      BPF_STMT(BPF_LD|BPF_B|BPF_ABS, SKF_AD_OFF + SKF_AD_VLAN_TAG_PRESENT),
      BPF_JUMP(BPF_JMP|BPF_JEQ|BPF_K, 1, 0, 6), /* true */

      /* verify DSS VLAN range */
      BPF_STMT(BPF_LD|BPF_H|BPF_ABS, SKF_AD_OFF + SKF_AD_VLAN_TAG),
      BPF_JUMP(BPF_JMP|BPF_JGE|BPF_K, 256, 0, 4),     /* 256 is first DSS VLAN */
      BPF_JUMP(BPF_JMP|BPF_JGE|BPF_K, 512, 3, 0),     /* 511 is last DSS VLAN */

      /* verify ethertype */
      BPF_STMT(BPF_LD|BPF_H|BPF_ABS, 2 * ETH_ALEN),
      BPF_JUMP(BPF_JMP|BPF_JEQ|BPF_K, ETH_P_802_3, 0, 1),

      BPF_STMT(BPF_RET|BPF_K, (u_int)-1),     /* accept */
      BPF_STMT(BPF_RET|BPF_K, 0),             /* ignore */
};
```

Tagged IP session 0 VLAN

As described above, MBIM IP session 0 is treated as special by the driver. It is initially mapped to untagged frames on the wwanY network device.

This mapping implies a few restrictions on multiplexed IPS and DSS sessions, which may not always be practical:

- no IPS or DSS session can use a frame size greater than the MTU on IP session 0
- no IPS or DSS session can be in the up state unless the network device representing IP session 0 also is up

These problems can be avoided by optionally making the driver map IP session 0 to a VLAN subdevice, similar to all other IP sessions. This behaviour is triggered by adding a VLAN link for the magic VLAN ID 4094. The driver will then immediately start mapping MBIM IP session 0 to this VLAN, and will drop untagged frames on the master wwanY device.

Tip: It might be less confusing to the end user to name this VLAN subdevice after the MBIM SessionID instead of the VLAN ID. For example:

```
ip link add link wwan0 name wwan0.0 type vlan id 4094
```

VLAN mapping

Summarizing the cdc_mbim driver mapping described above, we have this relationship between VLAN tags on the wwanY network device and MBIM sessions on the shared USB data channel:

```
VLAN ID       MBIM type   MBIM SessionID           Notes
-------------------------------------------------------
untagged      IPS         0                        a)
1 - 255       IPS         1 - 255 <VLANID>
256 - 511     DSS         0 - 255 <VLANID - 256>
512 - 4093                                         b)
4094          IPS         0                        c)

  a) if no VLAN ID 4094 link exists, else dropped
  b) unsupported VLAN range, unconditionally dropped
  c) if a VLAN ID 4094 link exists, else dropped
```

References

1. USB Implementers Forum, Inc. - "Universal Serial Bus Communications Class Subclass Specification for Mobile Broadband Interface Model", Revision 1.0 (Errata 1), May 1, 2013
    - http://www.usb.org/developers/docs/devclass_docs/
2. USB Implementers Forum, Inc. - "Universal Serial Bus Communications Class Subclass Specifications for Network Control Model Devices", Revision 1.0 (Errata 1), November 24, 2010
    - http://www.usb.org/developers/docs/devclass_docs/
3. libmbim - "a glib-based library for talking to WWAN modems and devices which speak the Mobile Interface Broadband Model (MBIM) protocol"
    - http://www.freedesktop.org/wiki/Software/libmbim/
4. ModemManager - "a DBus-activated daemon which controls mobile broadband (2G/3G/4G) devices and connections"
    - http://www.freedesktop.org/wiki/Software/ModemManager/
5. "MBIM (Mobile Broadband Interface Model) Registry"
    - http://compliance.usb.org/mbim/
6. "/sys/kernel/debug/usb/devices output format"
    - Documentation/driver-api/usb/usb.rst
7. "/sys/bus/usb/devices/.../descriptors"
    - Documentation/ABI/stable/sysfs-bus-usb