Rmnet Driver

1. Introduction

rmnet driver is used for supporting the Multiplexing and aggregation Protocol (MAP). This protocol is used by all recent chipsets using Qualcomm Technologies, Inc. modems.

This driver can be used to register onto any physical network device in IP mode. Physical transports include USB, HSIC, PCIe and IP accelerator.

Multiplexing allows for creation of logical netdevices (rmnet devices) to handle multiple private data networks (PDN) like a default internet, tethering, multimedia messaging service (MMS) or IP media subsystem (IMS). Hardware sends packets with MAP headers to rmnet. Based on the multiplexer id, rmnet routes to the appropriate PDN after removing the MAP header.

Aggregation is required to achieve high data rates. This involves hardware sending aggregated bunch of MAP frames. rmnet driver will de-aggregate these MAP frames and send them to appropriate PDN's.

2. Packet format

a. MAP packet v1 (data / control)

MAP header fields are in big endian format.

Packet format:

```
Bit             0             1           2-7      8-15           16-31
Function   Command / Data   Reserved     Pad   Multiplexer ID    Payload length

Bit            32-x
Function      Raw bytes
```

Command (1)/ Data (0) bit value is to indicate if the packet is a MAP command or data packet. Command packet is used for transport level flow control. Data packets are standard IP packets.

Reserved bits must be zero when sent and ignored when received.

Padding is the number of bytes to be appended to the payload to ensure 4 byte alignment.

Multiplexer ID is to indicate the PDN on which data has to be sent.

Payload length includes the padding length but does not include MAP header length.

b. Map packet v4 (data / control)

MAP header fields are in big endian format.

Packet format:

```
Bit             0             1           2-7      8-15           16-31
Function   Command / Data   Reserved     Pad   Multiplexer ID    Payload length

Bit            32-(x-33)       (x-32)-x
Function      Raw bytes      Checksum offload header
```

Command (1)/ Data (0) bit value is to indicate if the packet is a MAP command or data packet. Command packet is used for transport level flow control. Data packets are standard IP packets.

Reserved bits must be zero when sent and ignored when received.

Padding is the number of bytes to be appended to the payload to ensure 4 byte alignment.

Multiplexer ID is to indicate the PDN on which data has to be sent.

Payload length includes the padding length but does not include MAP header length.

Checksum offload header, has the information about the checksum processing done by the hardware.Checksum offload header fields are in big endian format.

Packet format:

```
Bit             0-14        15              16-31
Function      Reserved   Valid     Checksum start offset

Bit               31-47                    48-64
Function      Checksum length           Checksum value
```

Reserved bits must be zero when sent and ignored when received.

Valid bit indicates whether the partial checksum is calculated and is valid. Set to 1, if its is valid. Set to 0 otherwise.

Padding is the number of bytes to be appended to the payload to ensure 4 byte alignment.

Checksum start offset, Indicates the offset in bytes from the beginning of the IP header, from which modem computed checksum.

Checksum length is the Length in bytes starting from CKSUM_START_OFFSET, over which checksum is computed.

Checksum value, indicates the checksum computed.

c. MAP packet v5 (data / control)

MAP header fields are in big endian format.

Packet format:

```
Bit             0             1         2-7      8-15           16-31
Function   Command / Data  Next header  Pad   Multiplexer ID   Payload length

Bit            32-x
Function      Raw bytes
```

Command (1)/ Data (0) bit value is to indicate if the packet is a MAP command or data packet. Command packet is used for transport level flow control. Data packets are standard IP packets.

Next header is used to indicate the presence of another header, currently is limited to checksum header.

Padding is the number of bytes to be appended to the payload to ensure 4 byte alignment.

Multiplexer ID is to indicate the PDN on which data has to be sent.

Payload length includes the padding length but does not include MAP header length.

d. Checksum offload header v5

Checksum offload header fields are in big endian format.

Bit 0 - 6 7 8-15 16-31 Function Header Type Next Header Checksum Valid Reserved

Header Type is to indicate the type of header, this usually is set to CHECKSUM

Header types = ═══════════════════════════════════════ 0 Reserved 1 Reserved 2 checksum header

Checksum Valid is to indicate whether the header checksum is valid. Value of 1 implies that checksum is calculated on this packet and is valid, value of 0 indicates that the calculated packet checksum is invalid.

Reserved bits must be zero when sent and ignored when received.

e. MAP packet v1/v5 (command specific):

```
Bit             0             1         2-7      8 - 15          16 - 31
Function   Command         Reserved     Pad   Multiplexer ID    Payload length
Bit          32 - 39       40 - 45    46 - 47       48 - 63
Function   Command name    Reserved   Command Type   Reserved
Bit          64 - 95
Function   Transaction ID
Bit          96 - 127
Function   Command data
```

Command 1 indicates disabling flow while 2 is enabling flow

Command types

0	for MAP command request
1	is to acknowledge the receipt of a command
2	is for unsupported commands
3	is for error during processing of commands

f. Aggregation

Aggregation is multiple MAP packets (can be data or command) delivered to rmnet in a single linear skb. rmnet will process the individual packets and either ACK the MAP command or deliver the IP packet to the network stack as needed

MAP header|IP Packet|Optional padding|MAP header|IP Packet|Optional padding....

MAP header|IP Packet|Optional padding|MAP header|Command Packet|Optional pad...

3. Userspace configuration

rmnet userspace configuration is done through netlink library librmnetctl and command line utility rmnetcli. Utility is hosted in codeaurora forum git. The driver uses rtnl_link_ops for communication.

https://source.codeaurora.org/quic/la/platform/vendor/qcom-opensource/dataservices/tree/rmnetctl