Tracker BLE Communication Protocol

Author Yangjie Gu

Revision 0.2

Index

1 About This Document

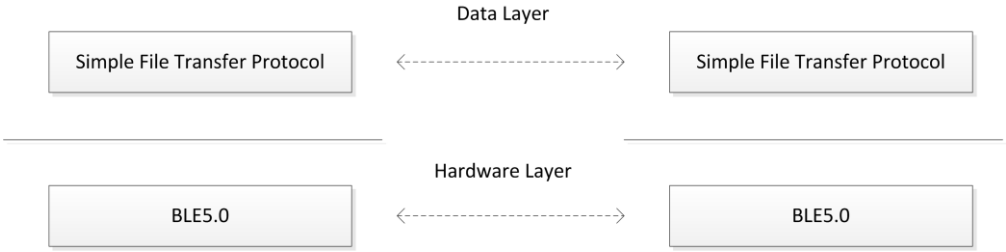
Tracker features a data transfer BLE profile. This document defines the communication protocol between Tracker and BLE_M.
Notation list is as the table below:

Notation	
BLE_S	Device works as a BLE slave role. (BLE peripheral)
BLE_M	Device works as a BLE master role. (BLE central)
Data_S	Data Sender, device send data to Data Receiver.
Data_R	Data Receiver , device receive data from Data sender.
MTU	Maximum Transmission Unit.

Tracker works as a BLE peripheral (BLE Slave).

2 Protocol Stack

Here is the picture of the protocol stack. The same layer in two devices could talk directly and ignore the low layer protocol.



BLE5.0 works in hardware layer which is common protocol, so let's focus on Simple File Transfer Protocol.

2.1 Simple File Transfer Protocol

Simple file transfer protocol is similar with TFTP protocol. All of the data in this protocol is in little endian. Below is the format of data packet.

opcode	data
2 bytes	N bytes

The generic format is very simple: opcode and data. The size of opcode is always 2 bytes and the size of data depends on the opcode. Below is the definition of the opcode.

Value (Hex)	Value (Dec)	Content
0x80	128	Server Data Transmit
0x90	144	Server Data Acknowledgement
0x81	129	Client Data Transmit
0x91	145	Client Data Acknowledgement

For each opcode, the content of data is different. Here is the definition of the data in different opcode.

2.1.1 Server/Client Data Transmit (0x80/0x81)

For BLE has a retransmit mechanism when the connection is not stable and every BLE data packet length is limited, it is necessary to give every BLE data packet ID, Total Number and Sequence Number. With them, Tracker and BLE_M can discard repeated packet and handle the data missing cases.

Format:

Packet ID	Total Number	Sequence Number	Data Type	Payload
1 byte	2 bytes	2 bytes	1 byte	N bytes

Description:
Packet ID: The type is integer, range: 0x00 – 0xFF. It is the ID of BLE data packet. The initial ID number is 0, increase by 1 for each new packet.
Total Number: The type is integer, range 0x0000-0xFFFF. It is the total number of parts of the BLE data packet.
Sequence Number: The type is integer, range 0x0000 - (Total Number - 1). It is the Sequence Number of the part in the whole BLE data packet.
Payload: The type is binary.
Data Type: The type is binary. Indicates the type of the Payload in the data Packet.

Data Type Value	Data Type
0x00	AT Command
0x01	AT Command Response
0x02-0xFF	Reserved

2.1.2 Server/Client Data Acknowledgement (0x90/0x91)

Acknowledgement of server/client data transmit.
Format:

Packet ID	Bitmap
1 byte	(Total Number + 7 – 1)/8 bytes

Description:
Packet ID: The type is integer, range: 0x00 – 0xFF. It is the ID of BLE data packet.
Bitmap: If every bit in bitmap is 0, it means all data received correctly. If bit N is 1, it means sequence number N of data packet is not received.

3 Working procedure

3.1 Server/Client Data Transmit/Acknowledgement

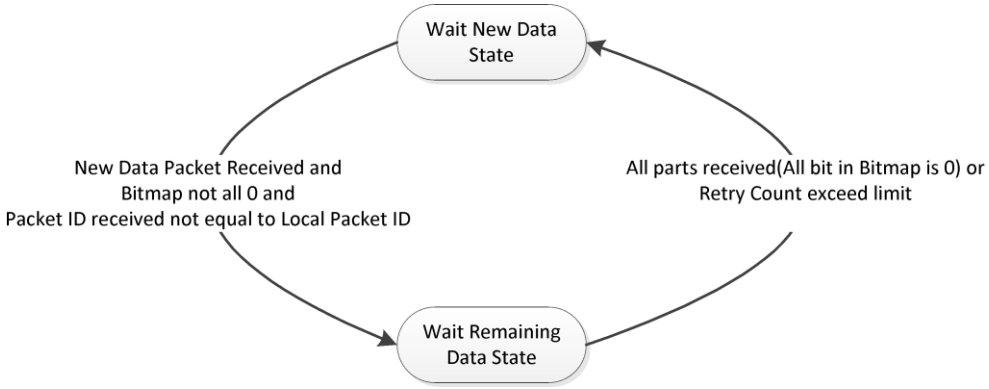
In field, both Tracker and BLE_M may have data to send to each other, from Tracker to BLE_M or from BLE_M to Tracker. To distinguish the data direction, Tracker works as Data Server and BLE_M works as Data Client. Both Server and Client can work as Data Sender or Data Receiver. Data Sender uses opcode Data Transmit, Data Receiver uses opcode Data Acknowledgement.
The procedure of Data Sender side is as below steps:
1, Check the maximum transmission unit (MTU) that can send in one data packet.
2, Wait data to be send and any Data Acknowledge received. If there is any data to send, jump to step 3. If there is Data Acknowledgement, discard it.
3, Increase Packet ID by one and set Retry Count to 0. (Packet ID range is 0x00 – 0xFF, if bigger than 0xFF, roll back to 0x00)
4, Divide the data into Total Number parts and set all bits in Local Bitmap to 1.
Total Number = (Data Length + MTU - Opcode Size – Packet ID Size – Total Number Size – Sequence Number Size – 1) / (MTU - Opcode Size – Packet ID Size – Total Number Size – Sequence Number Size)
5, Fill Packet ID field, Total Number field and Sequence Number field in Total Number data packets.
Sequence Number initial value is 0x0000, increase by one for every sequential data part.
6, Send all data parts with corresponding bit value 1 in Local Bitmap.

7, Wait for Data Acknowledge or Timeout. if Data Acknowledgement received jump to step 8. If Timeout, jump to step 9.
8, Set Retry Count to 0. Extract Packet ID and Bitmap in the Data Acknowledge and set the Local Bitmap to Bitmap in the Data Acknowledge. If all bits in Local Bitmap is 0, jump to step 1, otherwise jump to step 6.
9, Increase Retry Count by one. If Retry Count exceed the Maximum Retry Count jump to step 1, otherwise jump to step 6.

Data Receiver has two states:

a. Wait New Data State: Wait for data packet with new Packet ID
b. Wait Remaining Data State: Wait for remaining parts of the same Packet ID

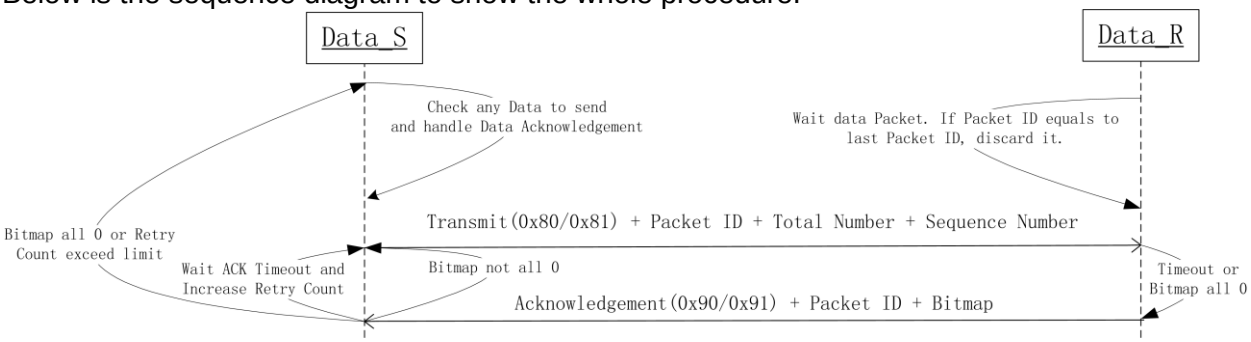
State switch is as below:



The procedure of Data Receiver side in the two states are as below.
Wait New Data State:
1, Wait data. If received any, extract the Packet ID, if the Packet ID equals to the Local Packet ID, discard it and jump to step 1, otherwise set it as Local Packet ID. Extract Total Number and Sequence Number, set all bits in Local Bitmap to 1, set Retry Count to 0, jump to step 2.
2, Record data and set bit Sequence Number in Local Bitmap to 0. If all bits in Local Bitmap are 0, send Server/Client Data Acknowledgement and jump to step 1, otherwise jump to Wait Remaining Data State.
Wait Remaining Data State:
1, Wait data or Timeout. If data come, set Retry Count to 0 and jump to step 2. If Timeout jump to step 5.
2, Extract Packet ID, if Packet ID equals to Local Packet ID jump to step 3, otherwise discard it and jump to Wait New Data State.
3. Extract Sequence Number of the received data, if bit Sequence Number in Local Bitmap is 1, record data and set bit Sequence Number in Local Bitmap to 0 and jump to step 4. If bit Sequence Number in Local Bitmap is 0, discard it and jump to step 1.
4, If all bits in Local Bitmap are 0, send Server/Client Data Acknowledgement and jump to Wait New Data State, otherwise jump to step 1.
5, Increase Retry Count by one, if Retry Count exceed maximum Retry Count, jump to Wait New Data State, otherwise send Server/Client Data Acknowledgement and jump to step 1

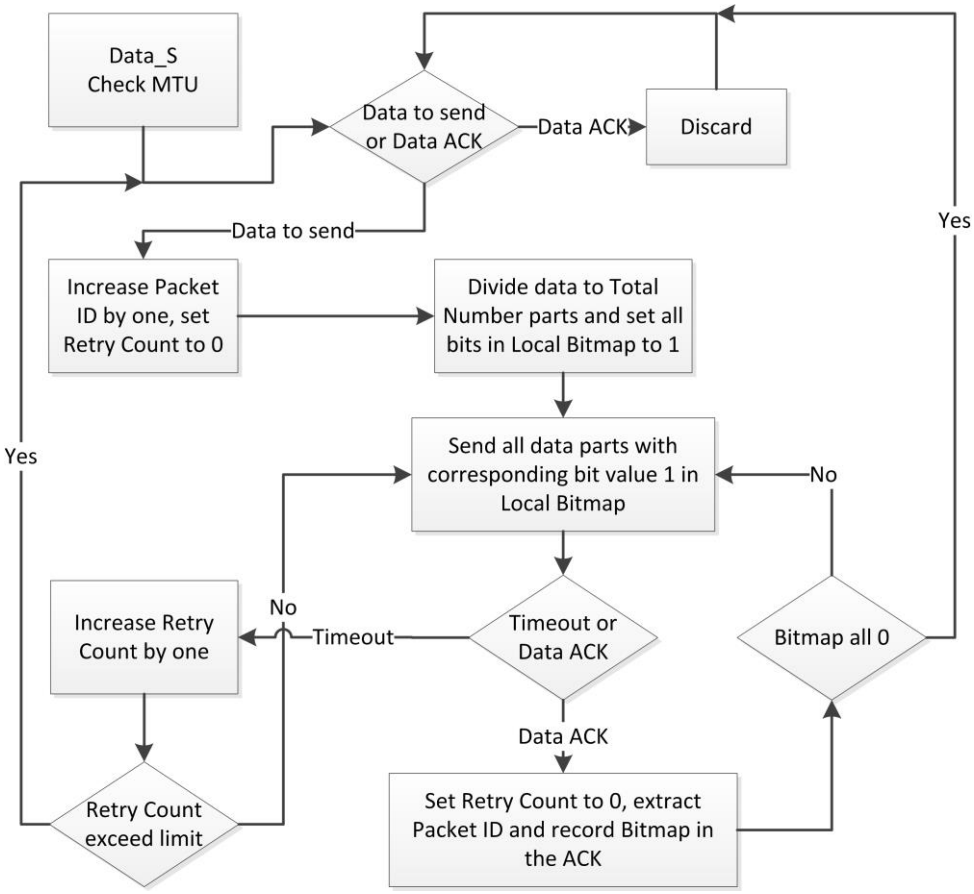
3.1.1 Sequence diagram

Below is the sequence diagram to show the whole procedure.

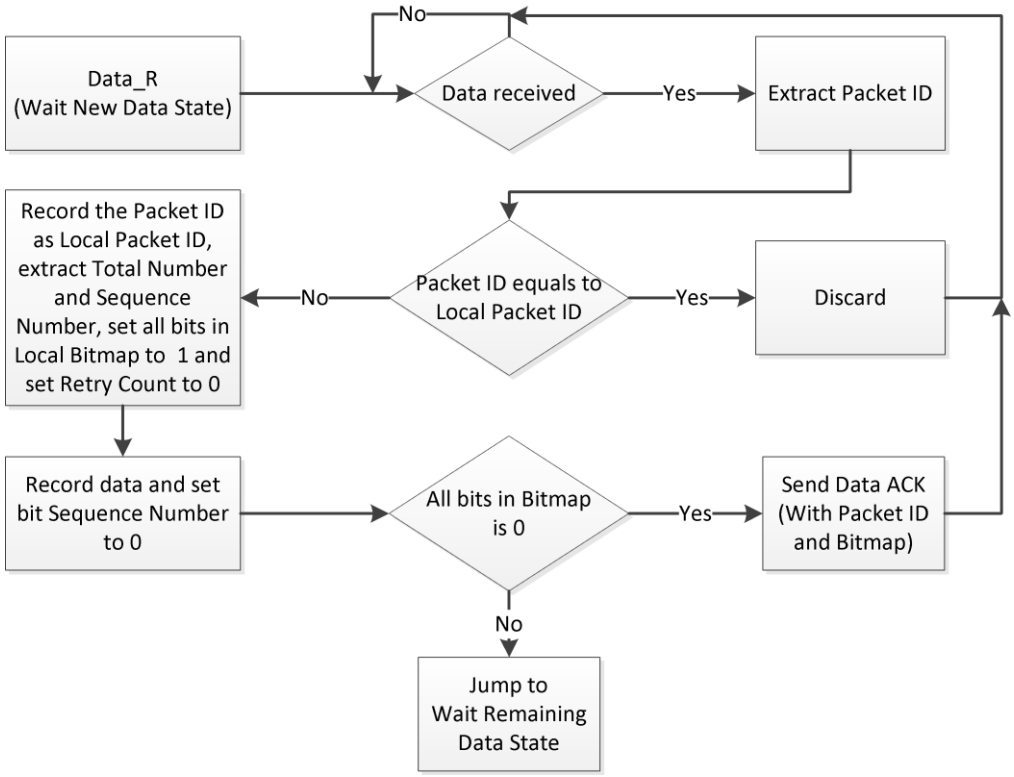


3.1.2 Work flow diagram

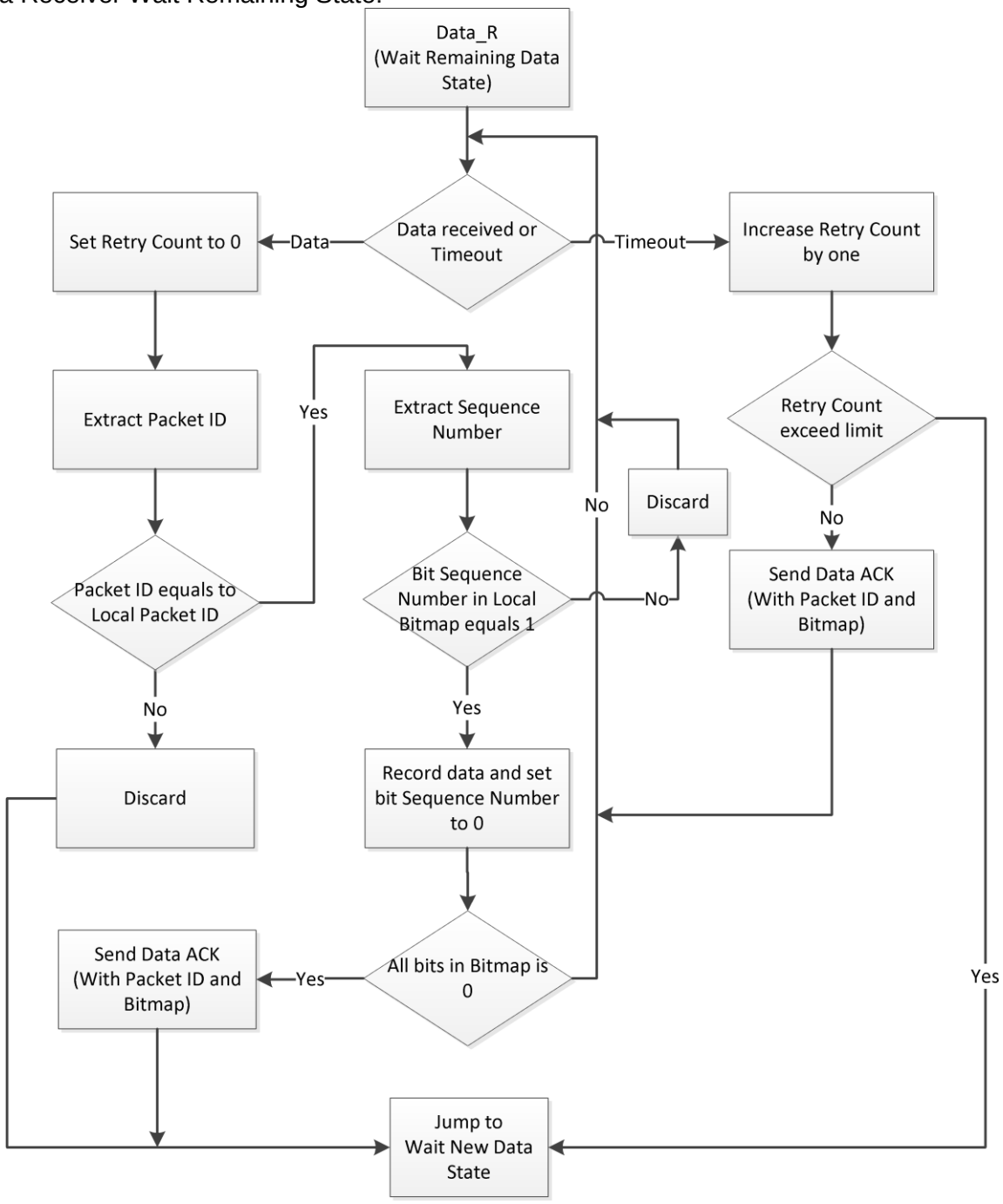
As aforementioned, here are the work flow diagrams for Data Sender and Data Receiver.
Data Sender:



Data Receiver Wait New Sate:

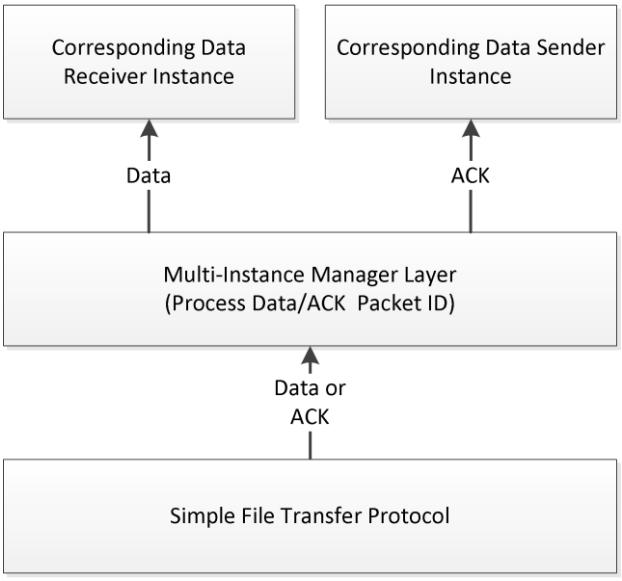


Data Receiver Wait Remaining State:



3.1.3 Data Sender/Receiver Multi-Instance

Every data Packet has its own Packet ID, so we can take every Data Sender and every Data Receiver as an instance. And we can have many different instances at the same time (With the Packet ID, we can process data with different Packet ID at the same time).



4 BLE AT Command

Server/Client Data Transmit/Acknowledgement is for general data, with these opcodes Tracker/BLE_M can send or receive any kind of data, AT Commands is one kind of these data. BLE_M sends AT Command to Tracker with opcode Client Data Transmit. When Tracker receives AT Command, it will parse the AT Command and send AT Command Response to BLE_M with opcode Server Data Transmit.

5 Change History

Revision	Author	Date	Change Notes
0.1	Yangjie	2021/04/22	Draft
0.2	Yangjie	2021/05/13	Add Data Type field in Server/Client Data Transmit.