ULEP Ultra Lightweight Embedded Protocol

1. Protocol purpose

Protocol was created to provide transport layer for applications running over limited bandwidth or data transfer cost expensive link, e.g. satellite connection.

2. System architecture

System basis on large, but limited number of active clients, which connect to base server using TCP/IP protocol. Every message have header which contain message data type and some type specific data.

3. Workflow

Client connects to a known server application using CONNECT message:

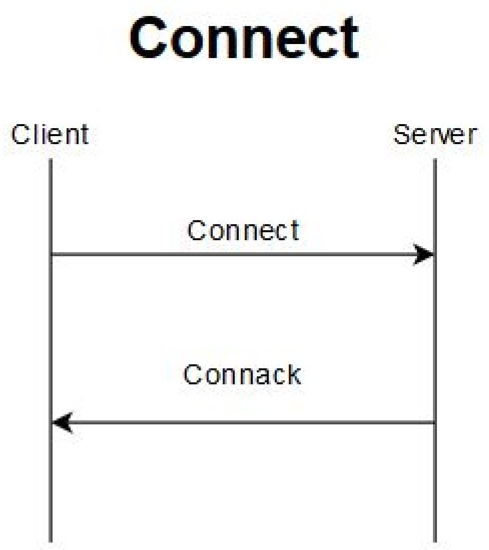

Chart 3.1 Connect workflow

After successful connection data transmission can begin in both directions:

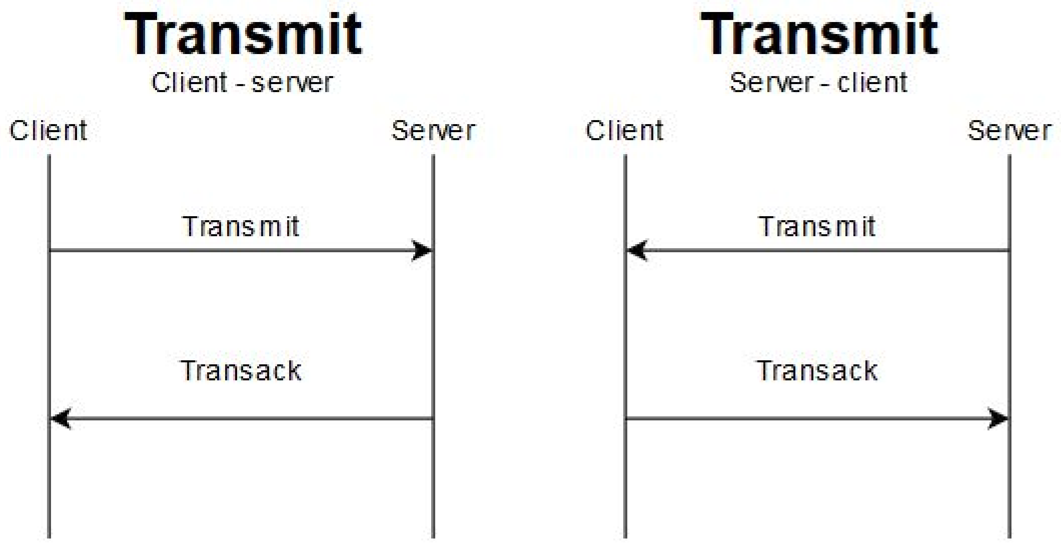

Chart 3.2 Data transmission workflow

When a client decides to close the connection, it has to send a disconnect message. After receiving this message, there should be no further data flow:

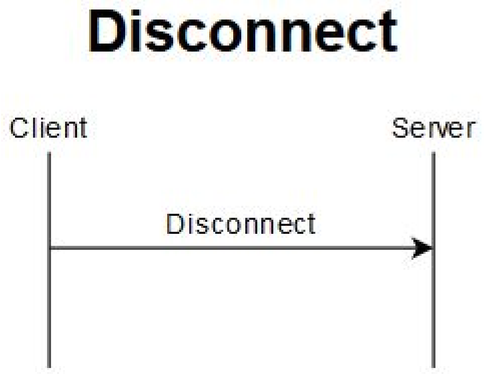

Chart 3.3 Disconnect workflow

4. Messages description

To

4.1. Messages types and format

All messages starts with header byte which has format shown in table 4.1.1:

Bit number	7	6	5	4	3	2	1	0
Description	Type		Type specific usage					

Table 4.1.1Header byte format

All messages have type value present in bits 7 and 6 in byte 1. These types are described in Table 4.1.2.

Type	Value	Description
CONNECT/CONNACK	0x00	Message sent while connecting and acknowledging connection
TRANSMIT	0x40	Message sent from client to server and from server to client
TRANSACK	0x80	Acknowledge message just after receiving message
DISCONNECT	0xC0	Message sent while client disconnects

Table 4.1.2 Message types

4.2. Connect/Connack

Client is the side, which initializes connection. Client sends a fixed length message of connect type and waits for a response from the server with a connack message. Connack has a fixed length of 1 byte. Connect and connack messages are the same type.

4.2.1. Connect

Connect message is 22 byte long message, which should be sent by client during first communication. It contains 4 information in proper bit configuration described in table 4.2.1.1. Those values are:

- Type - 2 bits defining message type
- Keep alive - 6 bits value of levels of keep alive time (predefined in whole system)
- Client ID - client individual identifier defined in 4 byte space (unsigned 32 bit integer)
- Server API key for authorization - value which has to be equal to predefined value stored in server application.

Byte 1								
Bit number	7	6	5	4	3	2	1	0
Description	Type (0x40)		Keep alive					
Values	0	0	X	X	X	X	X	X
Byte 2-6 (default)								
Bit number	7	6	5	4	3	2	1	0
Description	Client ID							
Values	X	X	X	X	X	X	X	X
Byte 7-22 (default)								
Bit number	7	6	5	4	3	2	1	0
Description	API Key							
Values	X	X	X	X	X	X	X	X

Table 4.2.1.1

4.2.2. Connack

Connack message is 1 byte long and contains 2 fields:

- Type - as in connect message (0x00)
- Return code - feedback value from server which determine if connection is successful or there is some issue with connection.

Format of connack message is described in table 4.2.2.1

Byte 1								
Bit number	7	6	5	4	3	2	1	0
Description	Type (0x00)		Return code					
Values	0	0	X	X	X	X	X	X

Table 4.2.2.1

Return codes are described in table 4.2.1.2

Value	Name	Description
0x00	OK	Connection successful
0x01	API_KEY	Authorization failed - wrong API key
0x02	ID_PROHIBITED	Wrong client ID value - server doesn’t allow it to authenticate
0x03	NOK	Unknown (other) issue

Table 4.2.1.2

4.3. Transmit

Transmit message contains 3 bytes header followed by defined data bytes containing message.

Byte 1								
Bit number	7	6	5	4	3	2	1	0
Description	Type (0x40)		Topic identifier					
Values	0	1	X	X	X	X	X	X
Byte 2								
Bit number	7	6	5	4	3	2	1	0
Description	Message ID							
Values	X	X	X	X	X	X	X	X
Byte 3								
Bit number	7	6	5	4	3	2	1	0
Description	Message length							
Values	X	X	X	X	X	X	X	X
Byte 4+ (optional)								
Bit number	7	6	5	4	3	2	1	0
Description	User data							
Values	X	X	X	X	X	X	X	X

4.4. Transack

Transack message has a fixed length of 2 bytes and its content should be equal to the first 2 bytes of the transmit message which it is acknowledging except of the message type. Message should point to received message by its topic identifier and message ID. Description of message is presented in table 4.4.1.

Byte 1								
Bit number	7	6	5	4	3	2	1	0
Description	Type (0x80)		Topic identifier					
Values	1	0	X	X	X	X	X	X
Byte 2								
Bit number	7	6	5	4	3	2	1	0
Description	Message ID							

Table 4.4.1

4.5. Disconnect

Disconnect message is fixed length of 1 byte and can be sent from client and server side. It contains only type value, presented in table 4.5.1. When this message is transmitted or received, both sides should close the connection.

Byte 1								
Bit number	7	6	5	4	3	2	1	0
Description	Type (0xC0)		Reserved					
Values	1	1	0	0	0	0	0	0

Table 4.5.1

4.6. Example of communication process

Server settings:
Api key: 0123456789abcdef

Connect packet:

`0x3C 0x00 0x00 0x00 0x01 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x61 0x62 0x63 0x64 0x65 0x66`

Header - 0x00 type combined with 0x3C (60) of keep alive
Cliend ID - 0x00000001 - 1
Api key - 0123456789abcdef utf-8 encoded*

Connack packet:

`0x00`

Header - 0x00 type combined with 0x00 return code (OK)

Transmit packet (client -> server):

`0x41 0x00 0x04 0x74 0x65 0x73 0x74`

Header - 0x40 type combined with 0x01 (1) topic ID
Message ID - 0x00 (0) as counter starts from 0
Message length - 0x04 (4) bytes long
Message data - `test` utf-8 encoded

Transack packet (server -> client)

`0x81 0x00`

Header - 0x80 combined with 0x01 (1) topic ID
Message ID - 0x00 (0) as message ID of received message

Transmit packet (server -> client):

`0x41 0x00 0x04 0x74 0x65 0x73 0x74`

Header - 0x40 type combined with 0x01 (1) topic ID
Message ID - 0x00 (0) as counter starts from 0
Message length - 0x04 (4) bytes long
Message data - `test` utf-8 encoded

Transack packet (client -> server)

`0x81 0x00`

Header - 0x80 combined with 0x01 (1) topic ID
Message ID - 0x00 (0) as message ID of received message

Disconnect packet (client -> server)

```
0xC0
```

Header - 0xC0 as it is fixed message for disconnect