Connecting to Featherweight TCP Sockets

Eric Metzler, 10/2/18

# Message Structure

Messages are structured using the following table. This structure is generic so that each application can decide what header strings to use and how the body is formatted. All other fields are controlled by FTW.

| Part | Length (bytes) | Type | Description | Example | Example (hex) |
| --- | --- | --- | --- | --- | --- |
| Message Length | 8 | U64 | Length of message minus these 8 bytes  Can be used to retrieve the rest of the message | 87 | 0000 0000 0000 0057 |
| Header Length | 4 | U32 | Length of header string | 9 | 0000 0009 |
| Header String | - | String | Message header | my header | 6D79 2068 6561 6465 72 |
| Body Length | 4 | U32 | Length of body string | 11 | 0000 000B |
| Body String | - | String | Message body | hello world | 6865 6C6C 6F20 776F 726C 64 |
| Sender Identity Length | 4 | U32 | Length of sender identity string | 9 | 0000 0009 |
| Sender Identity String | - | String | Name of the sending process | Viewpoint | 5669 6577 706F 696E 74 |
| Reply Address Length | 4 | U32 | Length of reply address string | 19 | 0000 0013 |
| Reply Address String | - | String | Address for replies to be sent (only used by a request socket) | tcp://1.2.3.4:56789 | 7463 703A 2F2F 312E 322E 332E 343A 3536 3738 39 |
| Error Status | 1 | Boolean | 0 = no error or warning  1 = error occurred | True | 01 |
| Error Code | 4 | I32 | Unique code for error/warning  0 = no error | -1 | FFFF FFFF |
| Error Source Length | 4 | U32 | Length of error source string | 14 | 0000 000E |
| Error Source String | - | String | Describes the error and/or source of the error | could not send | 636F 756C 6420 6E6F 7420 7365 6E64 |

Table 1 – Message Structure

Notes:

* All words are big endian
* Error information is generated by the connected application only and will not contain FTW errors

# Request Reply (ReqRep)

Request reply is a N:1 scalability protocol that can be used to send lossless messages from a request socket to a reply socket with an expected one-time reply from the reply socket.

Each request sent expects one reply. Each reply is specific to each request.

## Connecting/Receiving

1. Open a connection to the TCP listener at the desired IP and port
2. Repeat as long as messages need to be sent
   1. Encode a message using Table 1
   2. Send the message to the TCP endpoint
   3. Wait for 8 bytes to be received
   4. Read the rest of the message from the TCP buffer based on the received message length
   5. Decode the reply based on Table 1 and handle the message
3. Close the connection

# Subscriptions (PubSub)

Subscribers can connect to a publisher socket to listen for one-way, lossy messages coming from a publisher process. PubSub is a 1:N scalability protocol.

## Connecting/Receiving

1. Open a connection to the TCP listener at the desired IP and port
2. Loop until done listening
   1. Wait for 8 bytes to be received
   2. Read the rest of the message from the TCP buffer based on the received message length
   3. Decode the message based on Table 1 and handle the message
3. Close the connection