CalPolyPomona
College of Science
Computer Science

Communications Standard Proposal

Lit Internet Transmission Protocol (LitP)

Noah King

`nckingcpp.edu`

Abdelfattah Amamra
CS 3800
Spring 2024

Department of Computer Science

March 25, 2024

Abstract

This paper details the specifications pertaining to the proposed “Lit Internet Transmission Protocol”, hereby referred to in short as “LitP”. The Lit Internet Transmission Protocol is a messaging protocol intended to facilitate secure, and reliable simple multimedia communication (supporting text and image transmission). This particular document details the general structure of a LitP packet, encryption has not been discussed.

Contents

1 Message Format

1.1 Packet Structure (Header + Payload)

Figure 1 below details the width of each component of a complete LitP packet (header and payload). A LitP packet has a maximum size of 2^{72} [B], which is around 4.3 [GB].

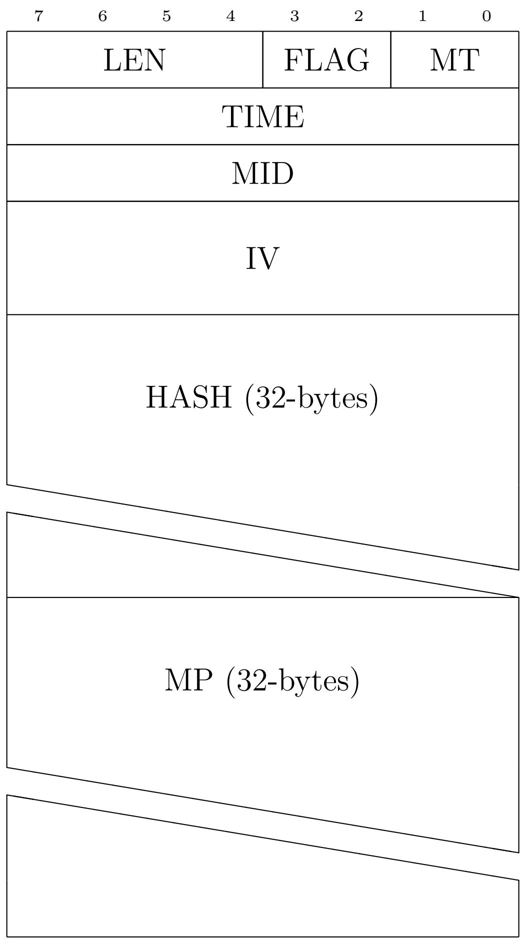


Figure 1: LitP Packet Structure

The initial element of the header, denoted as MT, specifies the category of the message (e.g., text, image, audio) utilizing two bytes. Following this, the second element, labeled FLAG, encompasses binary flags pertaining to different messaging options, options are detailed in the next section. The element TIME captures the timestamp of the message dispatch, represented by an 8-byte Unix timestamp. The message identifier MID, represents a descriptive message ID that can be used by the server. The identifier IV represents an initialization vector that can be used for encryption. Additionally, HASH constitutes a 32-byte hash for message integrity verification. Lastly, MP denotes the message payload, encapsulating the actual 32-byte content of the message. The allocation of two bytes for MT and FLAG is is to accommodate future enhancements and functionalities not presently envisioned within the standard's subsequent iterations, this allows for flexibility in the creation of new message types.

Full Name of Packet Term	Abbreviation	Length in Bytes
Message Type	MT	2
Options Flags	FLAG	2
Packet Length in Bytes	LEN	4
UNIX Style Timestamp	TIME	8
Message ID	MID	8
Initialization Vector	IV	16
Hash	HASH	32

Table 1: LitP Packet Header Terms

1.2 Contents of “FLAG” Field

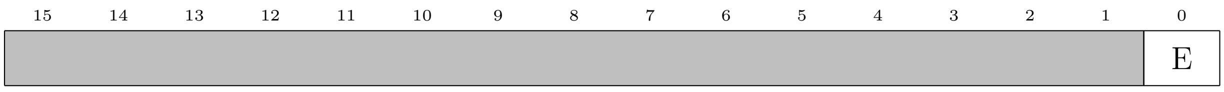
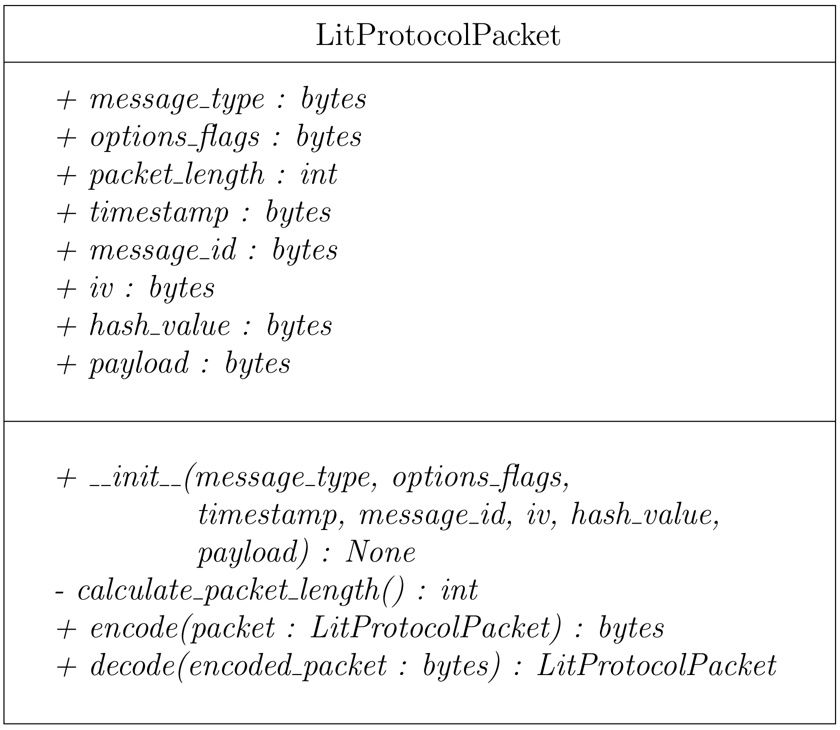


Figure 2: Contents of “FLAG” Field in LitP Packet Header

Figure 2 above depicts the contents of the 16-bit “FLAG” field shown in the previous section. As of now, most features are not agreed upon, however the remaining 15 bits allows for flexibility in the protocol for the addition of different features. The first bit (LSB) “E”, when set to 1 indicates an encrypted message, likewise, when it is 0, the message is not encrypted. This scheme allows us to maintain end-to-end encryption while still allowing users to send commands to the server, enabling features such as moderation.

1.3 LitP Packet Example Class

An LitP packet can be represented as the class shown by the UML Class diagram shown below in Figure 3. This class is created with the intention of streamlining the organization of data prior to serialization for TCP transmission. No encryption is handled in this class, only the encoding of a LitP packet into data that can be transmitted over TCP, or data that can be decoded.



```
LitProtocolPacket

+ message_type : bytes
+ options_flags : bytes
+ packet_length : int
+ timestamp : bytes
+ message_id : bytes
+ iv : bytes
+ hash_value : bytes
+ payload : bytes

+ __init__(message_type, options_flags,
           timestamp, message_id, iv, hash_value,
           payload) : None
- calculate_packet_length() : int
+ encode(packet : LitProtocolPacket) : bytes
+ decode(encoded_packet : bytes) : LitProtocolPacket
```

Figure 3: LitP Packet UML Diagram (Python)

2 Data Processing Control Flow

2.1 Sending a Message to Connected Clients

Figure 4 depicts the order in which an LitP packet should be processed when the header field FLAG = 0x00_01. The message signal $m[n]$ (a LitP packet) will first be encoded into a binary string prior to encryption. Following encryption, the serialized binary data will then be sent to the server via a transportation layer protocol such as TCP. The server will re-transmit the encrypted message to all connected clients, the message will then be decrypted by the clients recieving the message. Depending on the number encoded in the MT field in the packet header, the data could be processed by the client as a image, text, audio file, etc. after decryption.

$$\begin{tikzcd}
\boxed{m[n]} \arrow[r] & \boxed{\text{Encoding}} \arrow[r] & \boxed{\text{Encryption}} \arrow[r] & \boxed{\text{Server}} \arrow[r] & \boxed{\text{Decryption}} \arrow[r] & \boxed{\text{Decoding}} \arrow[r] & \boxed{m[n]}
\end{tikzcd}$$

Figure 4: Data Processing Stages, Encypted Case (FLAG = 0x00_01)

2.2 Issuing Command to Server

Figure 5 depicts the order in which an LitP packet should be processed when the header field FLAG = 0x00_00. The unencrypted message will be sent directly to the server, so that the server can process a command. This scheme of using an encryption flag to determine the decoding of a packet enables end to end encryption while also allowing a user to send an unencrypted command to the server. If encryption was always enabled, the server would decode the packet, and be unable to process the message due to the encryption. While encryption can be added server side, the encryption scheme would no longer be end to end.

$$\begin{tikzcd}
\boxed{m[n]} \arrow[r] & \boxed{\text{Encoding}} \arrow[r] & \boxed{\text{Server (Decoding is Server-Side)}}
\end{tikzcd}$$

Figure 5: Data Processing Stages, Unencypted Case (FLAG = 0x00_00)

3 Encryption

3.1 End-to-End Encryption Scheme

TBD