Assignment: Communication application using the UDP protocol

Milan Šeliga

ID: 127279

STU FIIT

Table of Contents

[Header Structure 3](#_Toc180420959)

[Estabilishing connection using 3-way handshake 3](#_Toc180420960)

[Stop-and-Wait ARQ and Fragmentation 4](#_Toc180420961)

[CRC for data integrity 7](#_Toc180420962)

[Packet structure and assembly 7](#_Toc180420963)

[Sources 8](#_Toc180420964)

## Header Structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TYPE | Fragment Number | Total Fragments | CRC | Flags | Data |
| 1B | 2B | 2B | 4B | 1B |  |

1. **TYPE (1 Byte):**
2. **Frag. Nr (2 Bytes):** This field holds the fragment number. It indicates the sequence in which the fragment should be reassembled on the receiving end. For example, fragment 1 of 5 would have this field set to 1.
3. **Total Frags (2 Bytes):** This field indicates the total number of fragments the original message is broken into.
4. **CRC (4 Bytes):** This field stores the CRC-32 checksum for integrity verification of the data in the fragment. Each fragment will have *its own CRC* to ensure it arrives without errors.
5. **Flags (1 Byte):** You can use this field for various flags related to fragmentation, such as:
   * *Fragmented flag:* A bit indicating whether the packet is fragmented or not.
   * *Final fragment flag:* A bit indicating whether this is the last fragment in the sequence.
   * *Acknowledgment flag:* A bit for acknowledging receipt of a fragment in case you're using a Stop-and-Wait ARQ mechanism.

## Estabilishing connection using 3-way handshake

A diagram of a diagram

Description automatically generated

#### Example Flow:

Peer A sends a SYN → Server

Peer B responds with SYN-ACK → Client

Peer A responds with ACK → Server

* Once this exchange is complete, the connection is established.

1 generall 3-way handshake

## Stop-and-Wait ARQ and Fragmentation

1. Determine Fragment Size: Based on the network’s MTU, calculate the maximum size for each fragment, subtracting the size of the headers.
2. Split Data: Divide the message into chunks that fit within the calculated fragment size. Each fragment is assigned a fragment number and contains part of the original message along with a header.
3. Send Fragments: Transmit each fragment individually over the network. After each fragment is sent, the sender waits for an acknowledgment (ACK) or a negative acknowledgment (NACK) indicating that a fragment was not received correctly.

### Message Receiving Steps with NACK:

1. Receive Fragments: The receiver collects incoming fragments and checks each one’s fragment number and total number of fragments to track which fragments are missing.
2. Check Integrity: The receiver verifies the integrity of each fragment using methods like CRC. If a fragment is missing or corrupted, the receiver sends a NACK to request retransmission of that specific fragment.
3. Reassemble Message: Once all fragments are received and verified, the receiver reassembles the message in the correct order. If the message isn’t fragmented, the receiver processes it immediately upon receipt and sends an ACK or NACK depending on integrity verification.

A screenshot of a computer

Description automatically generated

2 general Stop-and-Wait ARQ

## CRC for data integrity

We are using CRC32, which divides the message into 32 bites (thus 4B header), used for calculating whether our packets were damaged somehow or they came in pristine condition. We will be using CRC32 from the standard zlib library.

Here are the steps of CRC verification

1. When sending a message, a CRC value is calculated based on the message data.
2. This CRC is appended to the message header in the designated CRC field.
3. Upon receiving the message, the recipient recalculates the CRC using the same algorithm and compares it with the CRC value in the header.
   * If the two values match, the message is considered intact.
   * If they differ, the message is flagged as corrupted and will be retransmitted using the ARQ stop-and-wait mechanism.

## Packet structure and assembly

### Sources

<https://www.geeksforgeeks.org/>

<https://datatracker.ietf.org/doc/html/rfc768>