T. Volker
R. Finkler
C. Couturier, Ed.
A. Blanc, Ed.
IMT Atlantique
March 19, 2018

c2w protocol specification

Abstract

This document contains the protocol specification for the c2w application. The protocol covers the application's requirements specifying client-server interaction to log in, log out, send messages, access different movie rooms and update the movie rooms' states. This protocol is designed to work on an unreliable transport layer such as UDP.

Table of Contents

1. Introduction

The Chat While Watching (abbreviated as c2w) application is a desktop application in which a user, after logging in with a username, can chat with other users in different chat rooms while watching a movie at the same time. When logging in, the user first enters a room called the "Main Room" from which movie rooms are shown and can be accessed.

The c2w protocol specifies the interaction between a client and a server of the c2w application. This includes the login request and response, chat messages, room state updates, room changes and logout, on both success and failure cases. The video streaming for the movies in each room is not addressed in this specification.

This protocol can use an unreliable transport (e.g., UDP), as it uses a send and wait mechanism to reliably exchange messages. When an interacting party sends a packet, it MUST wait for the corresponding acknowledgement packet or resend exactly the same packet after a timeout period. The reliability mechanism is explained in more detail in the Reliability section.

1.1. Requirement Languages

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Definitions

The terminology particular to this document is defined below:

- Main Room: The first chat room a user enters to as soon as it logs in. There is no movie associated with this room and its purpose is to show the available movie rooms.

- Movie Room: A chat room associated with a movie. Users must always go through the main room to access a movie room.

- Session: An instance of a client-server connection which begins with a successful login and ends with a successful logout or when the server considers that the connection is lost. Every message exchanged between a client and a server (except login requests and failed login responses) is associated with a session.

3. General Considerations

The following observations define the general behaviour of the protocol and, unless explicitly stated, they are always valid.

- All integer numbers are encoded as unsigned integers with big endian byte order.
- All character strings are encoded in UTF-8.
- User names MUST not exceed 100 characters.
- Validation of fields sent by the client is the responsibility of the server. The server MUST use the corresponding error packets to communicate an invalid request or input.
- This protocol makes use of unique integer identifiers for Users and Rooms. The server implementation MUST assign and keep track of the identifiers to guarantee the correct behaviour of the protocol.
- Response codes which are not specified SHOULD be interpreted as unknown errors. Particular implementations MAY use response codes that are not specified to communicate custom error.
- Packets with inconsistent field values, that is to say, packets that do not follow this specification MUST be ignored in order for the timeout mechanism to address the issue.
- Client and Server implementations MUST limit the message size in order no respect the maximum packet size allowed by the transport layer. The Server MAY respond a request with an error to limit the size of packets sent by the client, e.g., rejecting a log in with a username too long.

4. Data Structures

The protocol uses the following data structures:

4.1. List

The list data structure is used to serialize an ordered sequence of elements of the same type. Its fields are described below, in their respective order:

1. Length (16 bits): Integer value indicating the number of elements the list contains. It can be zero, indicating that the list is empty.

2. Elements (variable size): The elements in the list in their respective order. All elements MUST be of the same type and they may be of different sizes. The size of each element is deduced from its type and its fields.

For example a list structure composed of integers 1, 2 and 3 is serialized in hexadecimal notation as: 00 03 01 02 03.

4.2. String

All strings are represented by two fields:

1. Length (16 bits): Integer value indicating the length in bytes of the UTF-8 encoded string. The length may be zero, indicating an empty string.

2. Text (variable size): Sequence of UTF-8 encoded characters of the string.

For example a string containing "Hello" is serialized as (in hexadecimal notation): 00 05 48 65 6c 6c 6f

4.3. User Data

The User Data structure is composed of a unique identifier and a username. The identifier is the one used by the server and every client must use the same identifier for each user. It consists on the following fields in this order:

1. User Identifier (16 bits): Unique integer identifier for the user. Each User is identified with the same identifier by the server and by all the clients. This identifier is assigned by the server at the beginning of the connection. It MUST NOT be zero. The server MUST assign User Identifier = 1 to the first client and increment this counter by one for each new client. User Identifiers of clients that have left the system MAY be reassigned by the server to new clients.

2. Username (variable size): String data structure containing the name of the user.

For example a User with id 10 and username "Bob" is serialized as (in hexadecimal notation): 00 0A 00 03 42 6f 62

4.4. Room Data

The Room Data structure is used to serialize the state of both the Main Room and a Movie Room. To use the same data structure for both types of rooms, the data type is defined recursively by containing a Room List. This Room list contains the data structures of all the Movie Rooms in the case of the Main Room and it is empty in the case of a Movie Room. The fields of this structure are described below in their respective order:

1. Room Identifier (16 bits): Unique identifier for the room. The value zero is reserved and it MUST NOT be used to identify a Room. Room Identifier 1 is used to identify the Main Room. Each Room is identified with the same identifier by the server and by all the clients.

2. Room name (variable size): String data structure containing the name of the room.

3. Movie IP (32 bits): IP address identifying the multicast group corresponding to the movie (used to stream the video. A value of zero indicates that no movie is available for the room. For the Main Room (Room Identifier 1) it MUST be set to zero.

4. Movie Port (16 bits): Port number associated with the RTP video flow. A value of zero indicates that no Movie is available for the room. For the Main Room (Room Identifier 1) it MUST be set to zero.

5. User List (variable size): A List containing User Data structures, each one corresponding to one of the users in the given room.

6. Room List (variable size): A List containing Room Data structures, making Room a recursive data structure. For the Main Room (Room Identifier 1), it lists the available Movie Rooms. For a Movie Room, this list MUST be empty.

Below we show an example of a Room data structure containing the information of a Main Room with two users and two available movies.

```
Room
 |-Room Identifier: 00 01 (1)
 |-Room Name: 00 09 4d 61 69 6e 20 52 6f 6f 6d (Main Room)
 |-Movie IP: 00 00 00 00 (0.0.0.0)
 |-Movie Port: 00 00 (0)
 |-User List
 |      |-Length: 00 02 (2)
 |      |-User
 |      |  |-User Identifier: 00 05 (5)
 |      |  |-Username: 00 03 42 6f 62 (Bob)
 |      |-User
 |         |-User Identifier: 00 12 (18)
 |         |-Username: 00 05 41 6c 69 63 65 (Alice)
 |-Room List
        |-Length: 00 02 (2)
        |-Room
        |   |-Room Identifier: 00 08 (8)
        |   |-Room Name: 00 07 54 69 74 61 6e 69 63 (Titanic)
        |   |-Movie IP: 0A 1d ec f2 (10.29.236.142)
        |   |-Movie Port: 27 d8 (10200)
        |   |-User List
        |   |      |-Length: 00 00 (0)
        |   |-Room List:
        |          |-Length: 00 00 (0)
        |-Room
            |-Room Identifier: 00 ae (174)
            |-Room Name: 00 05 41 6c 69 65 6e (Alien)
            |-Movie IP: 0A 1d ec f2 (10.29.236.142)
            |-Movie Port: 27 e2 (10210)
            |-User List
            |      |-Length: 00 01 (1)
            |      |-User
            |         |-User Identifier: 00 03 (3)
            |         |-Username: 00 07 43 68 61 72 6c 69 65 (Charlie)
            |-Room List:
                   |-Length: 00 00 (0)
```

Figure 1: Example of a Room Data Structure

The resulting byte string from serializing this example is:

```
00 01 00 09 4d 61 69 6e 20 52 6f 6f 6d 00 00 00 00 00 00 00 02 00 05
00 03 42 6f 62 00 12 00 05 41 6c 69 63 65 00 02 00 08 00 07 54 69 74
61 6e 69 63 0A 1d ec f2 27 d8 00 00 00 00 00 ae 00 05 41 6c 69 65 6e
0A 1d ec f2 27 e2 00 01 00 03 00 07 43 68 61 72 6c 69 65 00 00
```

5. Packet format

All specified packets in this protocol contain a fixed size header followed by a variable size payload, when needed.

5.1. Header

Every packet begins with a 8-byte header, containing the protocol version, the packet type, a session token, the sequence number and the payload size. Below we describe in detail each field:

1. Version (4 bits): Integer indicating the version of the protocol. It MUST be set to 1 to respect the protocol specified in this document. It can be updated in future versions of the protocol to enable new features.

2. Type (4 bits): Integer indicating the packet type according to Table 1.

3. Session Token (24 bits): Integer identifying the session in which the packet was sent. The zero value is reserved for the login request, therefore it MUST NOT be chosen by the server as a valid session token. All active sessions in a server at a given time MUST have different Session Tokens. Two sessions can have the same session token, even if they are from different users, but only if they don’t exist at the same time.

4. Sequence Number (16 bits): Identifies uniquely a packet and its corresponding acknowledgement in a certain direction of the session (client-server or server-client). Sequence number MUST be 0 for the first packet in a connection and then MUST be incremented by one for each new packet.

5. Payload Size (16 bits): Indicates the size of the payload in bytes. It may be zero indicating no payload.

The payload format depends in the packet type and are described later in this document. A packet with a packet type which is not specified in this document MUST be ignored.

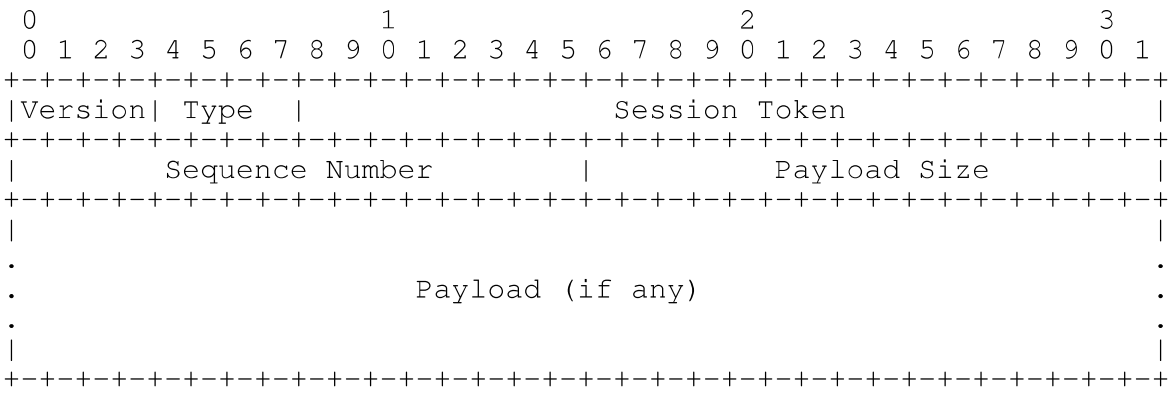


Figure 2: Header format

Short Name	Full Name	Value
ACK	Acknowledgement	0
LRQ	Login Request	1
LRP	Login Response	2
RRS	Request Room State	3
RST	Room State	4
GTR	Go To Room	5
MSG	Message	6
LOR	Logout Request	7
HEL	Hello	8

Table 1: Packet Type header field values

5.2. Packet Types

5.2.1. ACK: Acknowledgement

Sender: Client or Server

The Acknowledgement packet is part of the reliability mechanism and it is used only to confirm the reception of sent packets, both by the server and the client. The header fields are set as follows:

- Version: 1.
- Type: 0.

- Session Token: The Token contained in the acknowledged packet (i.e. 0 in the case of the ACK of the first LRQ message or the tokenID of the current session otherwise).

- Sequence Number: The sequence number of the acknowledged packet. When received, the next packet sent MUST have the next sequence number.

- Payload Size: 0 (no payload).

The ACK packet MUST NOT be acknowledged.

5.2.2. LRQ: Login Request

Sender: Client

The Login Request packet is the first packet sent in a session by the client. It is used to request the start of a session. Its header is filled as follows:

- Version: 1.

- Type: 1.

- Session Token: 0.

- Sequence Number: 0, since it is the first packet of the connection.

- Payload Size: The payload size.

The payload of Login Request is a User data structure with identifier zero and with the username of the user attempting to login.

After receiving a LRQ packet, the Server MUST respond with a Login Response packet indicating whether the login was successful or not.

5.2.3. LRP: Login Response

Sender: Server

The Login Response packet is the first non acknowledgement packet sent in a session by the server. It responds to a Login Request packet in both success and failure cases. The header must be filled as follows:

- Version: 1.

- Type: 2.
- Session Token: If the login request has been accepted, this field contains the session token assigned by the server to this client. It is set to 0 otherwise
- Sequence Number: 0, since it is the first packet sent by the server to the client for this connection.
- Payload Size: The payload size, in bytes.

The client and the server MUST send the session token assigned by the server on the subsequent packets in the session. The payload of this packet contains the following fields:

- Response Code (8 bits): The response code to the Login attempt. Zero indicates success, a positive value indicates an error. The error values are defined in Table 2.
- User: A User data structure with the information of the user. The identifier is set to the one assigned by the server if the login was successful or zero if not. The username must be the same as the one the user attempted to use to login, regardless if the login was successful or not.

As soon as the server receives the acknowledgement of a successful login LRP message, it MUST consider that the corresponding client has joined the main room. Thus, the server MUST notify all the users (including the recently joining one) using appropriate RST messages.

Name	Value
Log in Successful	0
Invalid User	1
Username too long	2
Username not available	3
Service not available	4
Unknown Error	255

Table 2: Login Response codes

5.2.4. RRS: Request Room State

Sender: Client

The Request Room State packet is used to request a Room State packet from the server. The header fields are as follows:

- Version: 1.
- Type: 3.
- Session Token: The session token assigned by the server at the beginning of the session.
- Sequence Number: The corresponding sequence number.
- Payload Size: 0 (no payload).

The RRS packet has no payload. The server MUST respond with a Room State packet.

5.2.5. RST: Room State

Sender: Server

The Room State packet is used to communicate the room in which the user is located and to inform the user of any updates on it. If there is any change in the room name, movie ip, movie port or user list of the current room this packet MUST be sent from the server to the clients currently in that room to inform about the change. In the case of the main room, this packet MUST be sent to the clients in the main room if there is any of the mentioned changes in any room, sending the full room list. The header fields are as follows:

- Version: 1.
- Type: 4.
- Session Token: The session token assigned by the server at the beginning of the session.
- Sequence Number: The corresponding sequence number.
- Payload Size: The payload size.

The payload of this packet is a Room Information data structure containing the information of the current Room.

5.2.6. GTR: Go to Room

Sender: Client

The Go to Room packet indicates the client's intention to change Room. The header fields are filled in the following way:

- Version: 1.
- Type: 5.
- Session Token: The session token assigned by the server at the beginning of the session.
- Sequence Number: The corresponding sequence number.
- Payload Size: 2.

The payload of this packet is a 16 bit integer containing the ID of the Room that the user wants to join.

5.2.7. MSG: Message

Sender: Client or Server

When the client sends a Message packet, it indicates its intention of sending a new chat message to the current room. When the server sends a Message packet, it communicates that a new message in the chat Room has been sent. The messages of the user to the server MUST NOT be sent back to the user who sent it. The header fields are filled as follows:

- Version: 1.
- Type: 6.
- Session Token: The session token assigned by the server at the beginning of the session.
- Sequence Number: The corresponding sequence number.
- Payload Size: The payload size.

The payload of this packet contains the following fields:

- User Id (16 bits): User identifier of the sender of the message. If the client is sending the message, it MUST fill this field with its own user identifier.

- Message: A String data structure with the content of the message.

5.2.8. LOR: Logout Request

Sender: Client

The Logout Request is sent by the client to close the session. The header fields are filled as below:

- Version: 1.
- Type: 7.
- Session Token: The session token assigned by the server at the beginning of the session.
- Sequence Number: The corresponding sequence number.
- Payload Size: 0 (no payload).

This packet contains no payload.

5.2.9. HEL: Hello

Sender: Server

The Hello Message is sent by the server to check that client connection is still alive (see Connection supervision section). When receiving an HEL message a node has nothing to do (apart from sending the corresponding ACK like for any other message).

- Version: 1.
- Type: 8.
- Session Token: The session token assigned by the server at the beginning of the session.
- Sequence Number: The corresponding sequence number.
- Payload Size: 0 (no payload).

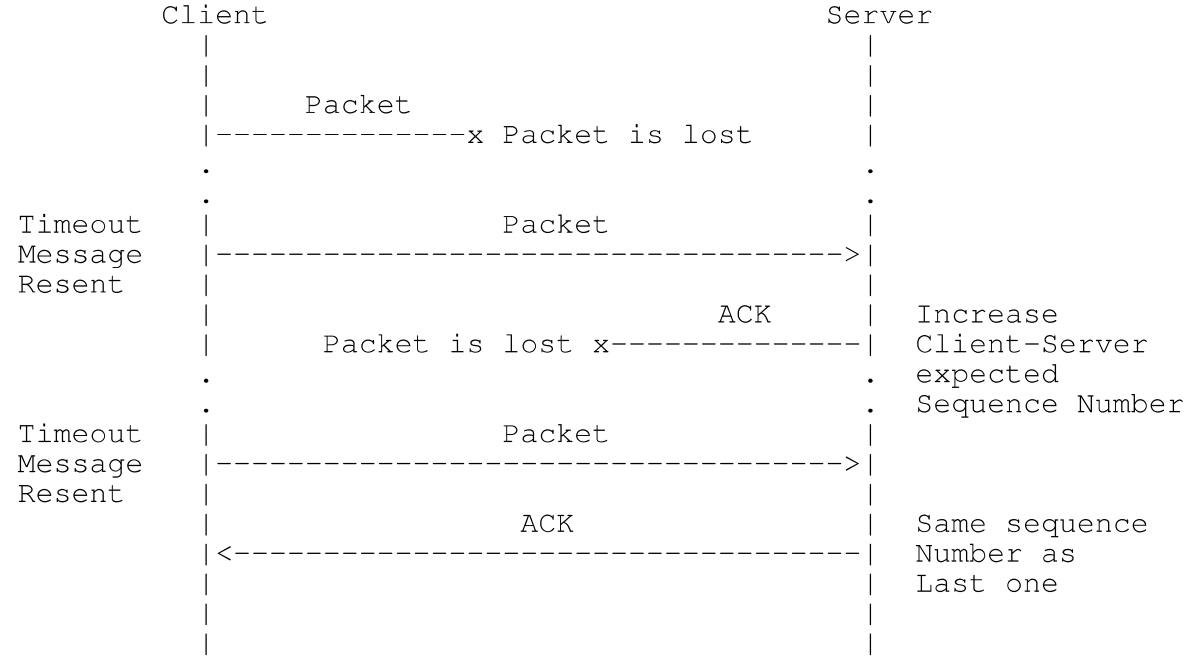
This packet contains no payload.

6. Reliability

A "send and wait" strategy is defined to ensure that exchanged packets are interpreted in the right order. A 16 bit sequence number is defined for each direction of the communication (client-server and server-client). Both client and server implementations MUST keep track of both sequence numbers. Sequence numbers are considered circular, this means that the sequence number after 65535 is 0.

All packets except ACK packets MUST be acknowledged by sending an ACK packet with the sequence number of the packet being acknowledged. If a certain response to a packet is specified, it MUST be sent after the ACK. If a packet is received with a sequence number that is not the next expected value, it MUST be ignored to allow the timeout mechanism to address the issue. After sending any packet (except ACK packets) the sender MUST wait for the ACK packet with the same TokenID and sequence number before sending the next one. If the expected ACK has not arrived within 1 second, the sender MUST retry sending exactly the same packet. After three consecutive attempts without success, the connection is considered lost and the session closed.

Example of a Connection with lost packets



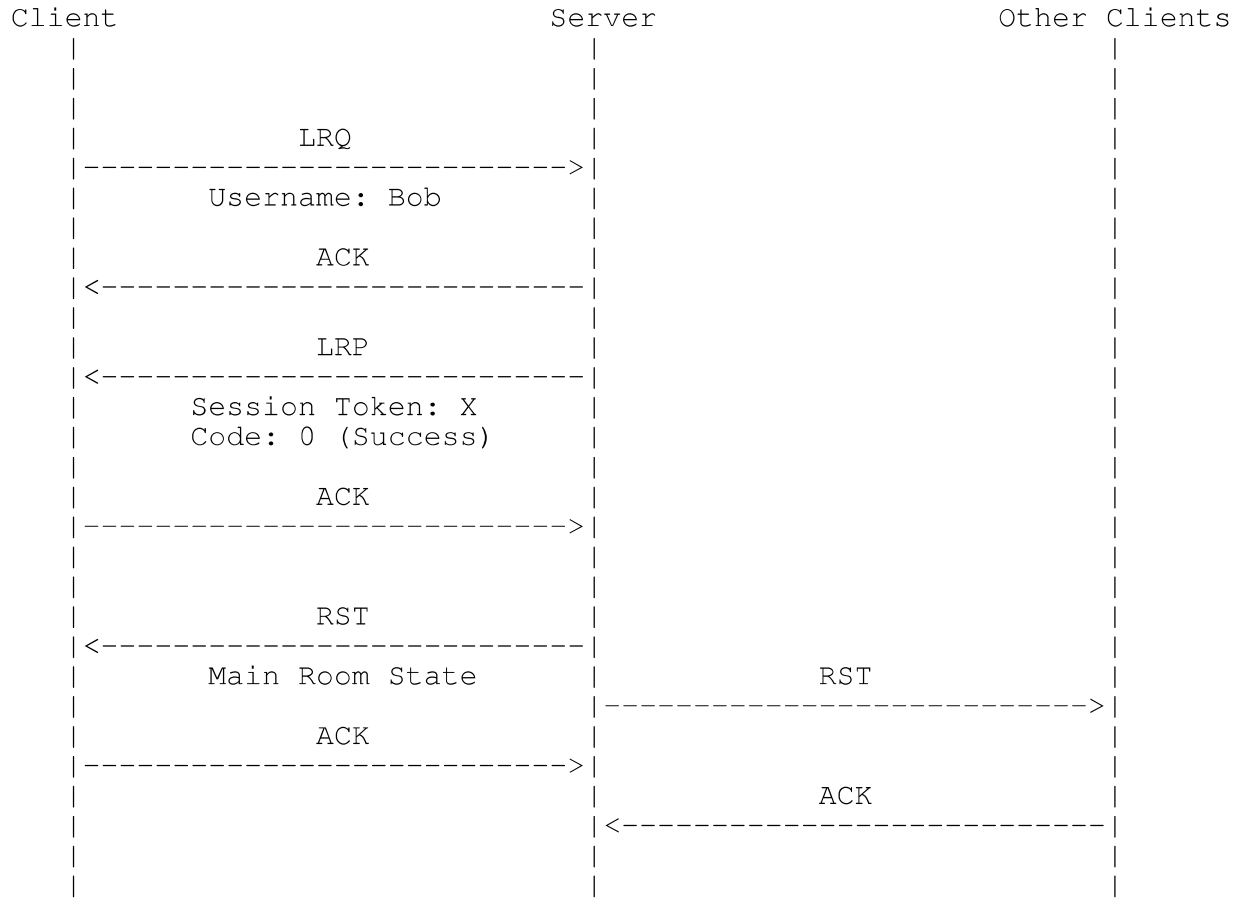
7. Connection supervision (KeepAlive)

In order to verify that connected clients are still alive, the server MUST send a HEL message to clients from which no message has been received during the last 10 seconds. This message has no effect on the client other than acknowledging the packet, by sending back an ACK. Should this ACK not be received after three consecutive HEL messages, the client will be considered disconnected as described in the Reliability section.

8. Examples

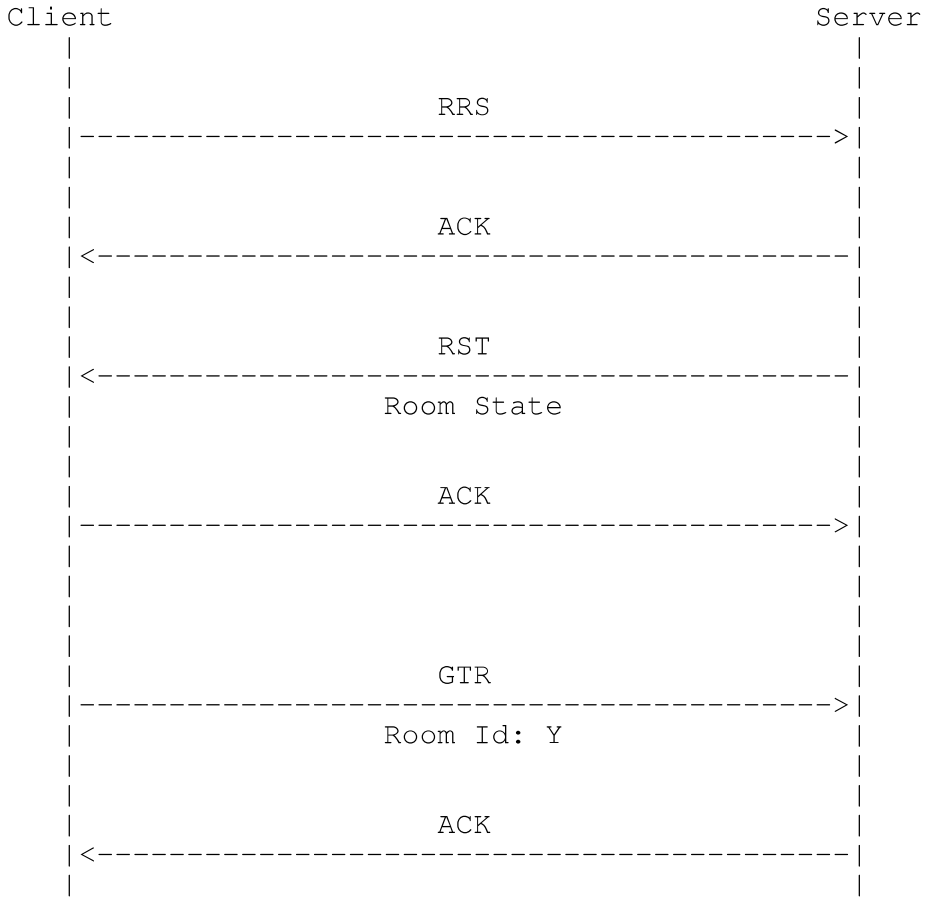
8.1. Successful Login

Example of a Successful Login



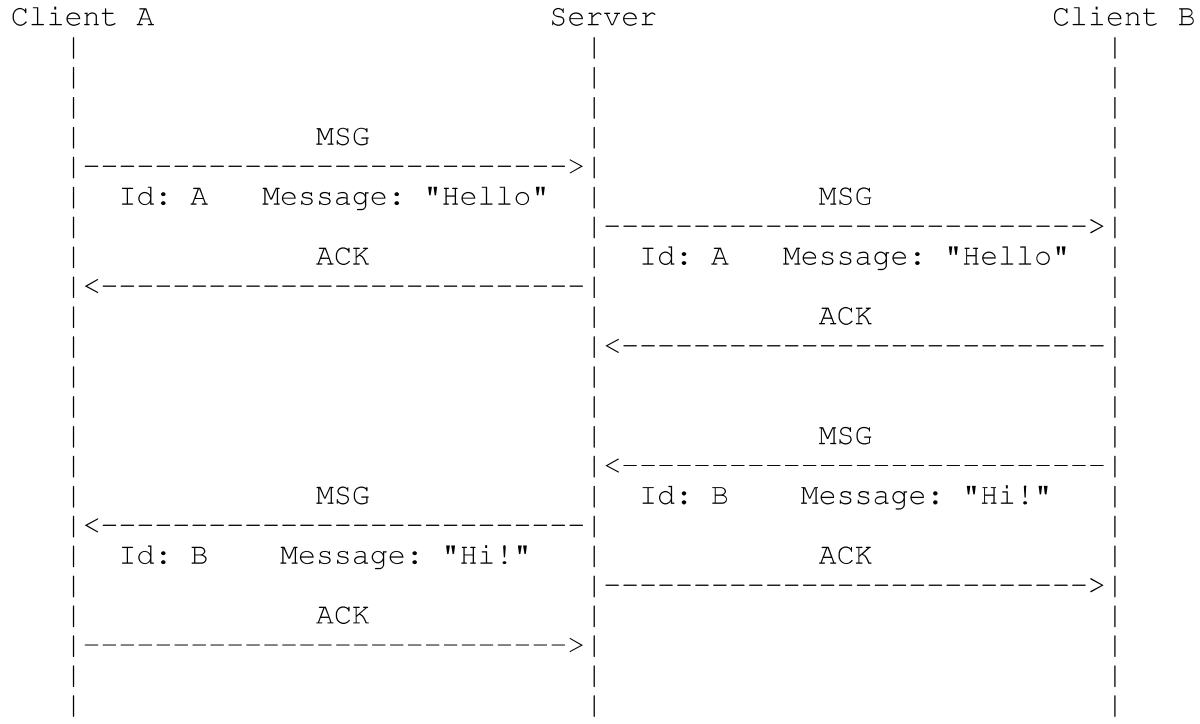
8.2. Successful Room Requests

Example of a Successful Request for Room State and Change Room



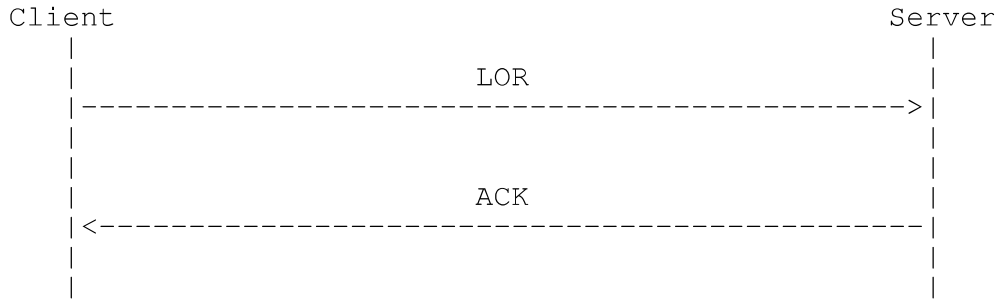
8.3. Successful Message Sending

Example of a Successful Message sending.



8.4. Successful Logout

Example of a Successful Logout



9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997, <https://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Tomas Volker
IMT Atlantique
Brest
France

Email: tomas.volker@imt-atlantique.net

Rodrigo Finkler
IMT Atlantique
Brest
France

Email: rodrigo.finkler@imt-atlantique.net

Christophe Couturier (editor)
IMT Atlantique
Rennes
France

Email: christophe.couturier@imt-atlantique.fr

Alberto Blanc (editor)
IMT Atlantique
Rennes
France

Email: alberto.blanc@imt-atlantique.fr