M. ASSAL
F. TAMAGNAN
Telecom Bretagne
October 2017

Group 3 :c2w protocol specification proposal

Abstract

A specification of C2W protocol that allows a C2W application online chatting while watching a video.The system is composed of different rooms, each of them correponds to a film where clients in a same room CAN discuss.

Table of Contents

1. Introduction

The C2W is an application protocol which allows an online chatting with users present in the same room ,either the main room or the movie room .

The client needs to login at the beginning to establish a connection with the server: He enters his USER NAME, SERVER'S IP ADRESS and

SERVER’S PORT NUMBER.The server check if the USER NAME does already exist.If it’s the case it sends to the client an error message ,else he sends to the client a unique ID and directes him to the main room.

In the main room client CAN see the list of available movies and the list of users present in the same main room . He CAN chat with them or join a movie room where he can also have acces to the list of users in the same room while watching a video.

When a client sends a message in a the main room or the movie room, all the users in that same room will receive it.

If a client wants to leave a movie room he will be directed to the main room there he CAN leave the Application bye quitting the main room and turning back to the login window.

C2W protocol SHOULD work on both TCP and UDP transport protocoles. It should ensure the reliability given that the UDP is an unreliable protocol.

To make it simple, we are going to consider a fixed format of a packet for every message. Only the data field changes from one message to another. .

1.1. Requirements Language

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. Server Configuration

An event is defined as every action on the software (join a movie room, add message, etc) The server MUST keep in its database the list of events. When an event happens, the server updates its database. It MUST keep the mapping between every user_name and user_ID. The server has also in his database the list of movies. It gives the smallest user_ID to each new user. It MUST make sure that it can send back every client request.

3. terminology and abbreviaitons

- UDP : User Datagram Protocol
- TCP : Transmission Control Protocol.
- ACK: Acknowledgement.

- IP : Internet Protocol.
- c2w : Chat While Watching.

You can write text here as well.

4. Packet Format

Our protocol is going to consider the following fixed-format of a packet:

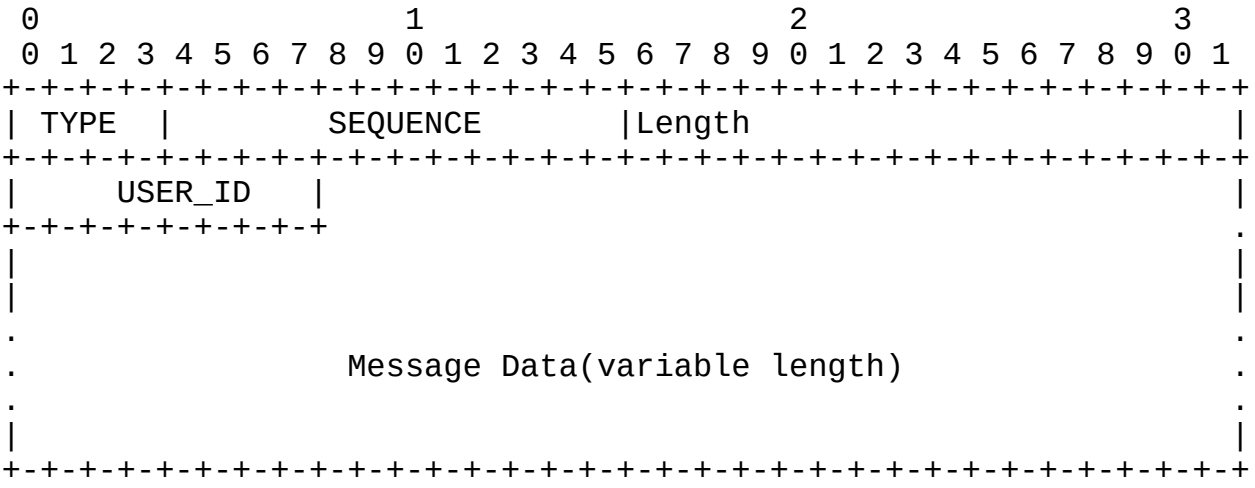


Figure 1

4.1. SEQUENCE (16 bits)

This field is necessary since the c2w protocol must work with both TCP and UDP Transport layer protocol. The unreliability of the UDP protocol imposes to sequence each transmitted packet. The sequence number is an integer between 0 and 2^16. When logging in, the sequence field is initialized with 0 and incremented when an ACK of the message sent is recieved. The server and each client have their own single sequence Each message MUST be acknowleged with the same number of sequence.In the case that a server or the client haven’t receive an ACK in a certain delay,it MUST re-send the message with the same number of sequence. If it’s always the case it MUST stop sending the message to avoid saturating the receiver.

4.2. TYPE (4 bits)

Binary Code	Type
0000	login request
0001	login response
leave_room request	0010
leave_room response	0100
message request	1000
message forward	0011
message ACK	0101
users' list response	1001
users' list ACK	0110
movies' response	1010
movies' list ACK	1100
Select movie request	0111
Select movie response	1011
error message log in pseudo	1101
error message log in, too much users	1110

Table 1: A Very Simple Table

This field specifies the id number of the type of the message according to the following table.

login request -> |0000|

When logging in, the client MUST send a user name to the server.He enters in the login window the socket corresponding to the server and his user name

DATA=user name

login response -> |0001|

When a login request is sent ,the server checks if the USER NAME has already been used then sends a ACK to the client either LOG_IN DONE or LOG_IN FAIL.

If the LOG_IN DONE the server sends a unique USER ID to the client.The it sends the movies' list to the client and Then it updates the user's list in the main room and sends it to each main_room_user.

leave_room request -> |0010|

The client asks the server to leave a room,either from a movie room to the main room or from th main room to the login window(disconnection).

DATA=Empty

leave_room response -> |0100|

It’s an acknowledgement from the server to the client

DATA=Empty

message request -> |1000|

The client in a room sends a message to the server

DATA=MESSAGE

message forward -> 0011

The server forwards the message request to all the users in the same room including the client sender. This message plays the role of an ACK from the server to the client.

Data=Message

message ACK -> 0101

Each client sends an ACK to the server. So that the server make sure that each client has received the message. In case of non receiving one ACK the server in a certain delay, the server re_sends the message to the concerned client. DATA=EMPTY

users’ list response -> |1001|

The server sends the list of users of the same movie room requested by the client to the client himself and the other users in the same room to update the list. DATA= length1 user_name1 length2 user_name2

users’ list ACK -> |0110|

Each client MUST send a ACK to the server, after receiving the user’s list.

DATA=EMPTY

movies’ response -> 1010

The server sends to the client the list of movies available in the main room

DATA= length1 movie1

length2 movie2

movies’ list ACK -> |1100|

After receiving the Movie’s list the client MUST send a ACK to the server DATA=EMPTY

Select movie request -> |0111|

After receiving the Movie’s list the client MUST send a ACK to the server DATA=EMPTY

Select movie response -> |1011|

Every movie corresponds to a socket. The server provides the video flow sens to the client the user’s list in that same movie room. It, then upadates the new user’s list and sends it to each user in that same room.

error message log in pseudo -> |1101|

The server sends an error message when the user enters a name which has already been used by another .

DATA=EMPTY

error message log in, too much users -> |1110|

The server sends an error message when the sever is saturated.

DATA=EMPTY

4.3. Packet Length (16 bits)

It contains the length of the Data. The data on UDP can’t exceed 65527 that’s why the length is above 2 bytes

4.4. User ID (16 bits)

It contains the ID of the sending client which has been choosen by the server in the phase of logging in.

4.5. Message Data (variable length)

This field contains the name that is the object of the request, encoded in ASCII.

Both the client and the server MUST always specify the name in each message.

5. Example scenario

5.1. scenario 1:log in

When a client log in, he enters his user name and the (IP, PORT) of the server. the packet sent by the client to the server has the following format

Example: USER_NAME="MAYA"

TYPE:0000 SEQUENCE=0 LENGTH=4 USER_ID=0 DATA=MAYA

Packet send by the server in case of success

TYPE:0001 SEQUENCE=0 LENGTH=0 USER_ID=00000001 DATA=EMPTY

Then the server sends the list of users in the main room to each user present in the main room

Example of packet sent to user "Maya"

TYPE: 1001 SEQ=1 LENGTH=... ID_user=00000001 DATA=Length1 User_Name1 Length2 USER _NAME2

The server Waits the ACK from each user.

Example of ACK paquet sent by "MAYA" to the server:

TYPE 0110 SEQ=1 LENGTH=0 ID_user=00000001 DATA: EMPTY
The server sends the list of movies to each user in the main room

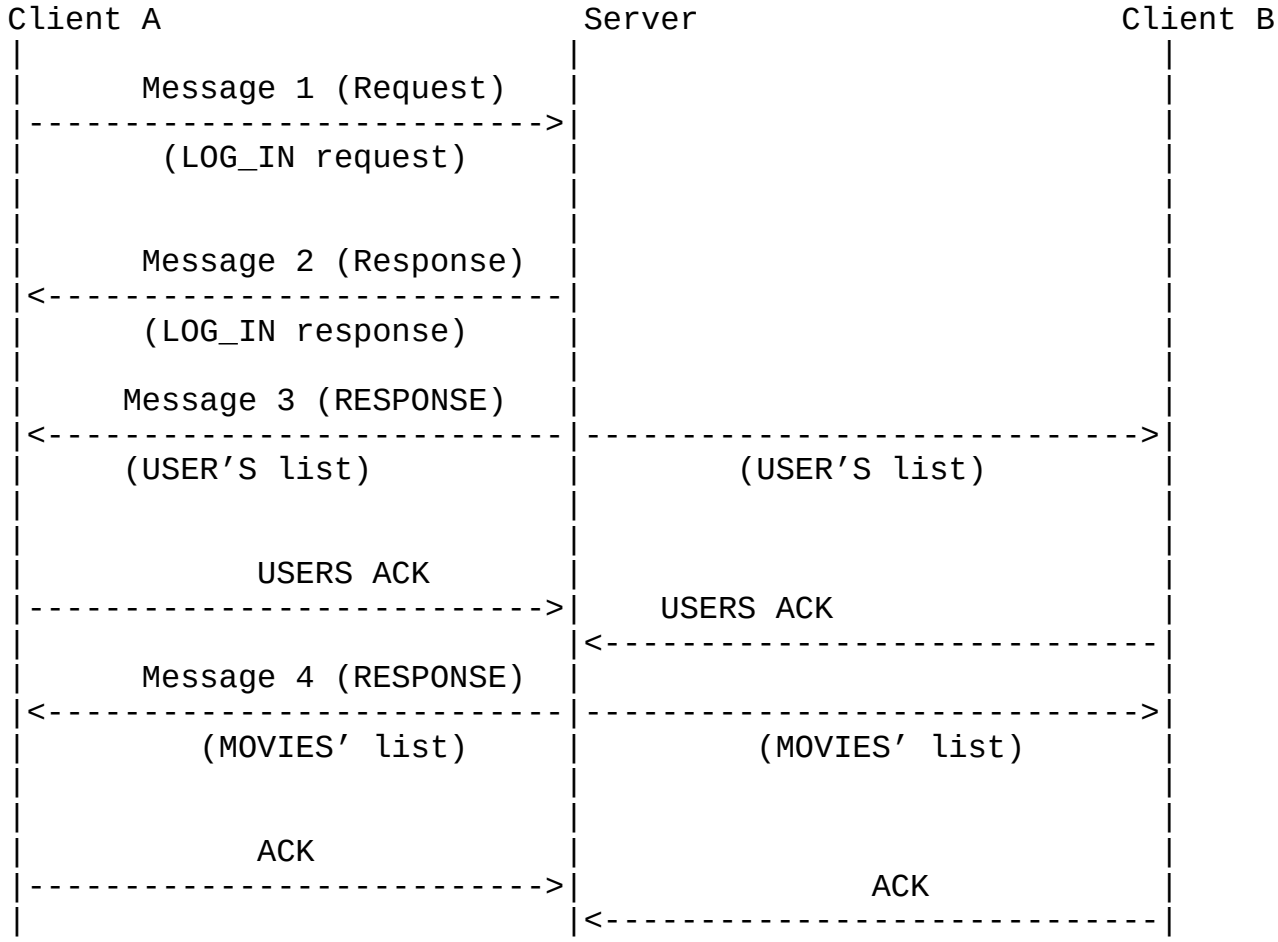
Example of packet sent to user Maya:

TYPE:1010 SEQ=2 LENGTH=... ID_user=00000000 DATA=Length1 MOVIE1 Length2 MOVIE2
The server then waits for the ACK from each client:

Example of ACK sent by "MAYA"

TYPE: 1100 SEQ=2 LENGTH=... ID_user=00000001 DATA=EMPTY

We can also use a figure like this:



5.2. Scenario 2: Enter Movie Room

When a client select a movie, he enters in the correspondant movie room. The server sends to each user in that movie room (including the client himself) the new list of users.

It then updates the users' list of th main room and send it to users in the main room.

SELECT MOVIE REQUEST:

Example: USER_NAME="MAYA"

SELECT MOVIE1="SCARRY"

TYPE:0111 SEQUENCE=3 LENGTH=6 USER_ID=00000001 DATA="SCARRY"

Then the server sends the list of users in the same movie room to each user present in the same room

Exemple of packet sent to user "MAYA"

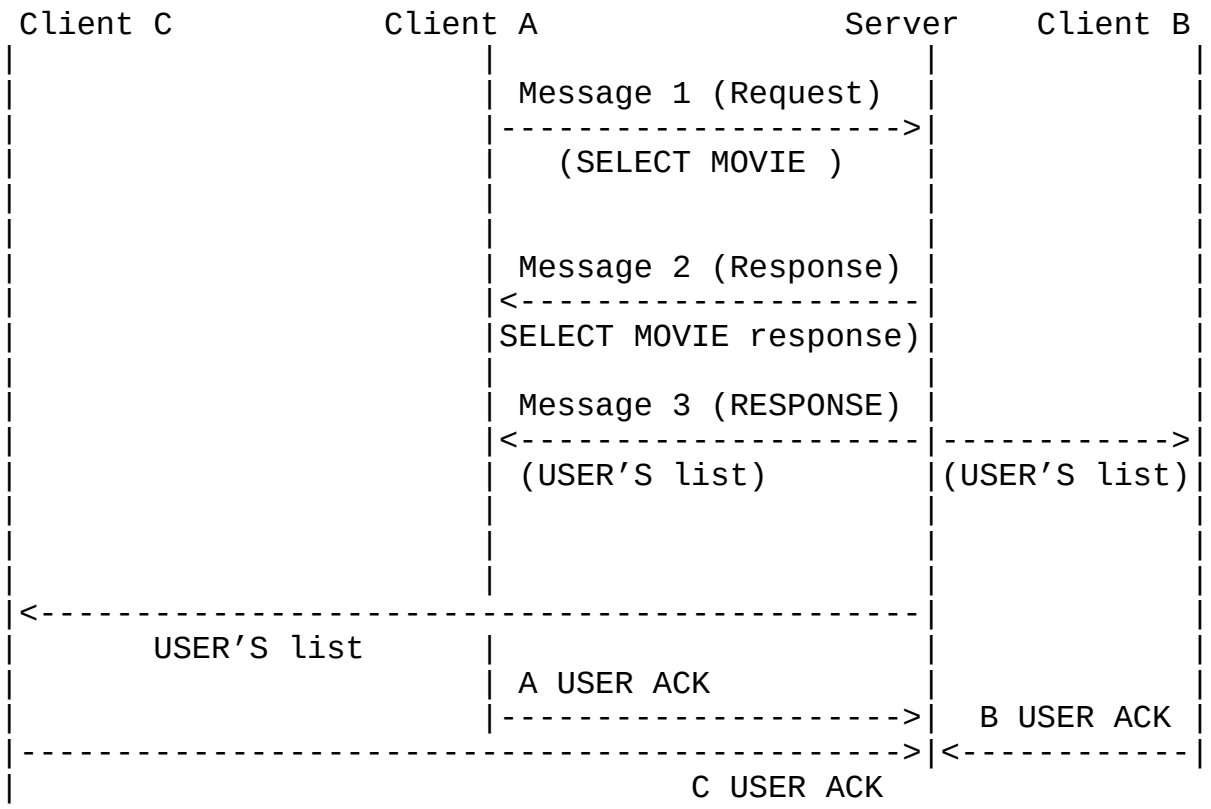
TYPE: 1001 SEQ=3 LENGTH=... ID_user=00000001 DATA=LENGTH1 User_Name1 LENGTH2 USER _NAME2

The server Waits the ACK from each user.

Exemple of ACK paquet sent by "MAYA" to the server:

TYPE 0110 SEQ=3 LENGTH=0 ID_user=00000001 DATA: EMPTY

Suppose that the client A is in the main room B in the movie room selected by A and C still in the main room:



5.3. Scenario 3:Message Request

When a client sends a message in a room, the server forwards it to each user present in that room (including the client sender).

the packet sent by the client to the server has the following format

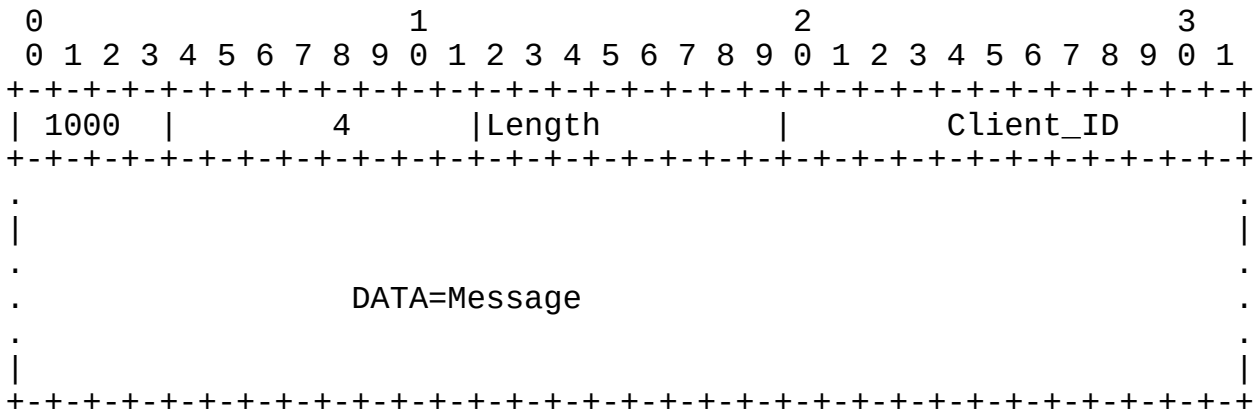


Figure 2

the Server sends to each client present in that same room the message(including the client sender):

Example: paquet forwarded to client B with USER_ID=00010001

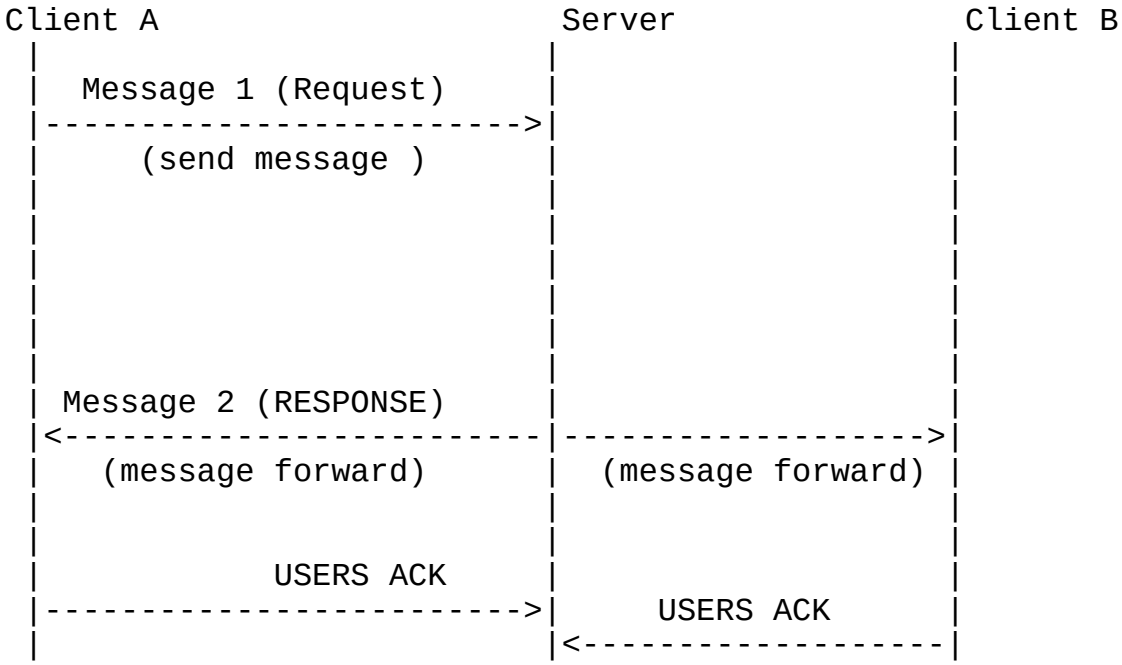
TYPE=0011 SEQUENCE=4 LENGTH=5 ID_USER=00010001 DATA="HELLO"

Then the server waits an ack from other users sends

Exemple of packet sent by the user "Maya" to the server

TYPE: 0101 SEQ=4 LENGTH=0 ID_user=00000001 DATA=EMPTY

We can also use a figure like this:



5.4. scenario 4:leave a movie room

When a client wants to leave a movie room, he sends a leave_room request to go back to the main room.

The packet sent by the client to the server has the following format

Example: USER_NAME="MAYA"

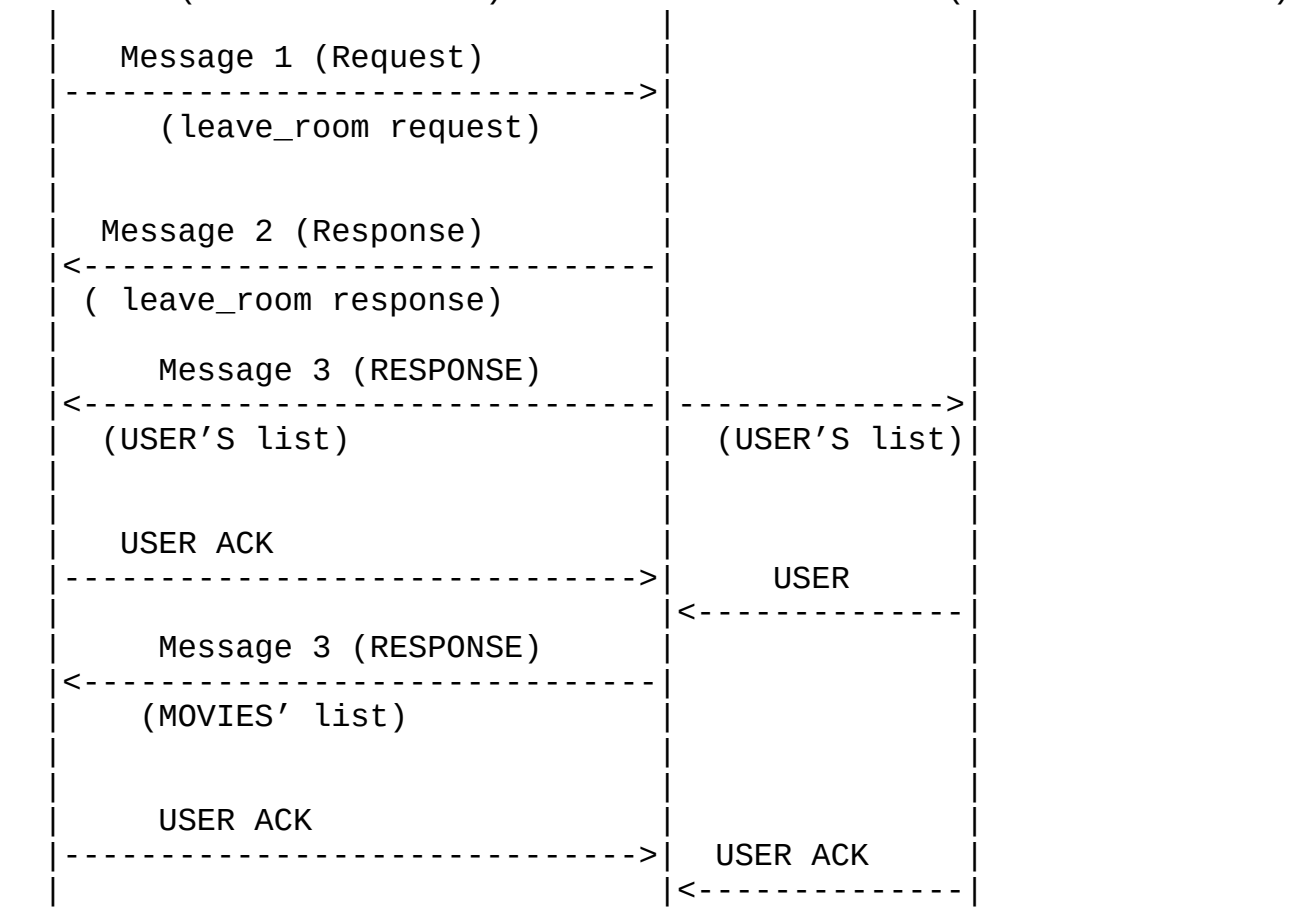
TYPE:0010 SEQUENCE=5 LENGTH=0 USER_ID=00000001 DATA=empty

The server directs him to the main room and updates the list of users in the main room to each user present in the main room and does the same with users in the left movie room.

The server Waits the ACK from each user.If the server doesn’t receive a ACK from a user in a 3 seconds, it re-sends it. In case of failure after 3 times in total it stops sending.

It sends then the list of movies available in the main room to the client.

You can also use a figure like this:



5.5. scenario 4:leave main room (disconnect)

When a client wants to leave a main, he sends a leave_room request to disconnect. The packet sent by the client to the server has the following format

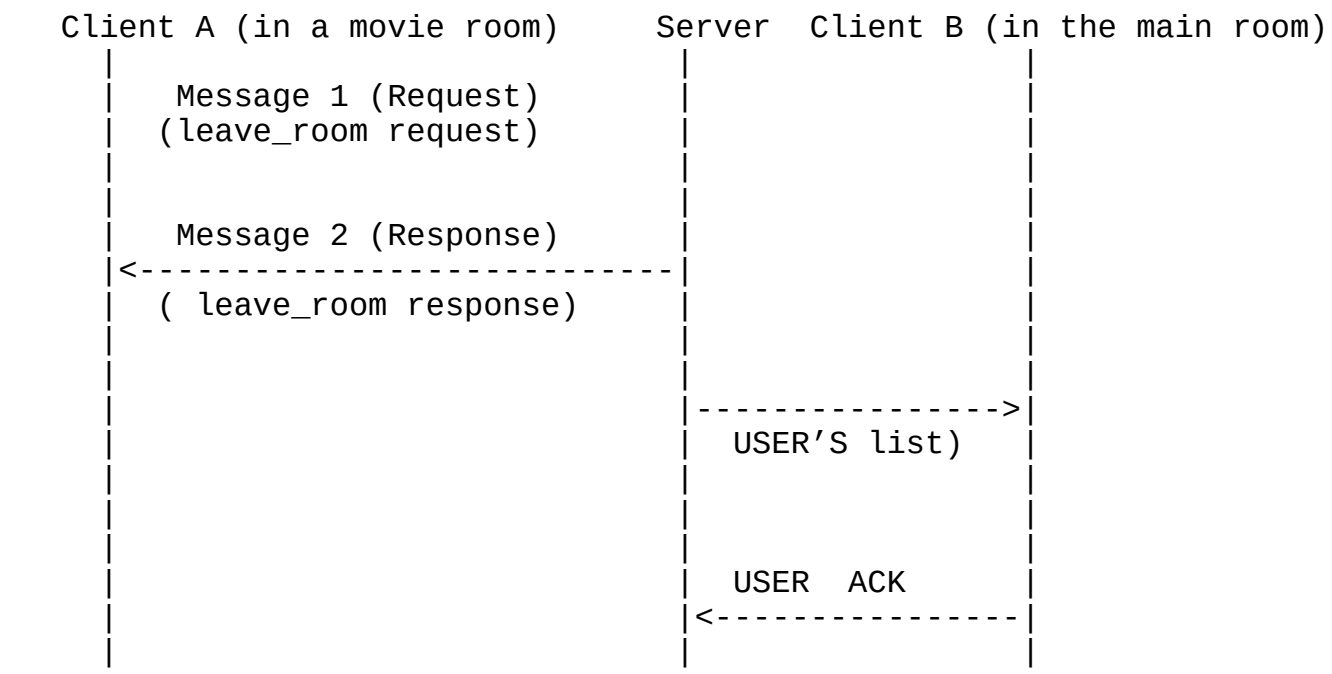
Example: USER_NAME="MAYA"

TYPE:0010 SEQUENCE=6 LENGTH=0 USER_ID=00000001 DATA=empty

The server diconnect the client and then updates the list of users in the main room to each user present in the main room. The server Waits the ACK from each user.If the server doesn't receive a ACK from a user in a 3 seconds, it re-sends it.

In case of failure after 3 times in total it stops sending.

Suppose that A wants to disconnect, B stays in the main room

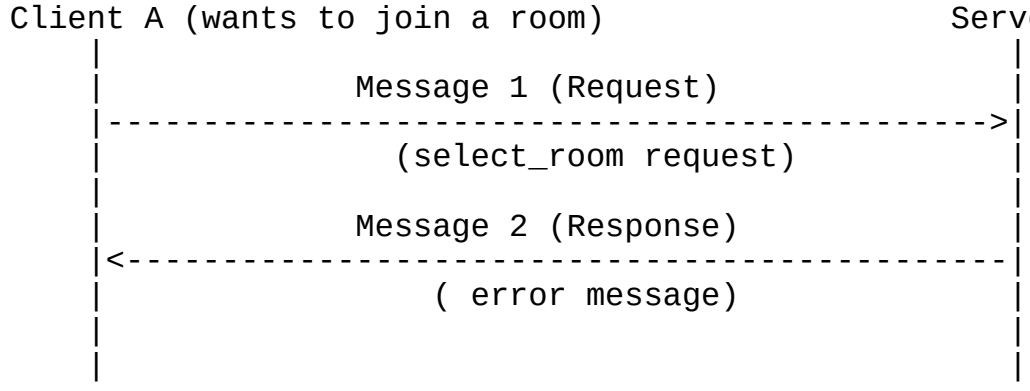


5.6. scenario 5: error message log in, too much users

When a client wants to join a room which is full of users, the server answers him an error message indicating that the room is saturated. Let's assume that Maya is the client.

FORMAT OF ERROR MESSAGE (response) :

TYPE:1110 SEQUENCE=10 LENGTH=0 USER_ID=00000001 DATA=EMPTY



5.7. scenario 6: error message log in, user name already taken

When a client wants to join the main room and the user_name wanted is already taken. Let’s assume that Maya is the client.

FORMAT OF ERROR MESSAGE (response) :

TYPE:1101 SEQUENCE=10 LENGTH=0 USER_ID=00000001 DATA=EMPTY

6. References

6.1. Normative References

[min_ref] Blanc, A., "c2w protocol specification proposal", 2006.

6.2. Informative References

[FindIPAdress]
Blanc, A., "Sample Protocol Specification: FindIpAddress", June 2006, <https://formations.telecom-bretagne.eu/fad/pluginfile.php/69283/mod_resource/content/1/FindIpAddress-bin.xml>.

Authors’ Addresses

Maya Assal
Telecom Bretagne
Brest
France

Email: maya.assal@telecom-bretagne.eu

Frederic Tamagnan
Telecom Bretagne
Brest
France

Email: frederic.tamagnan@telecom-bretagne.eu