utente che fa login:

user: _____
password: _____

login_result: _____

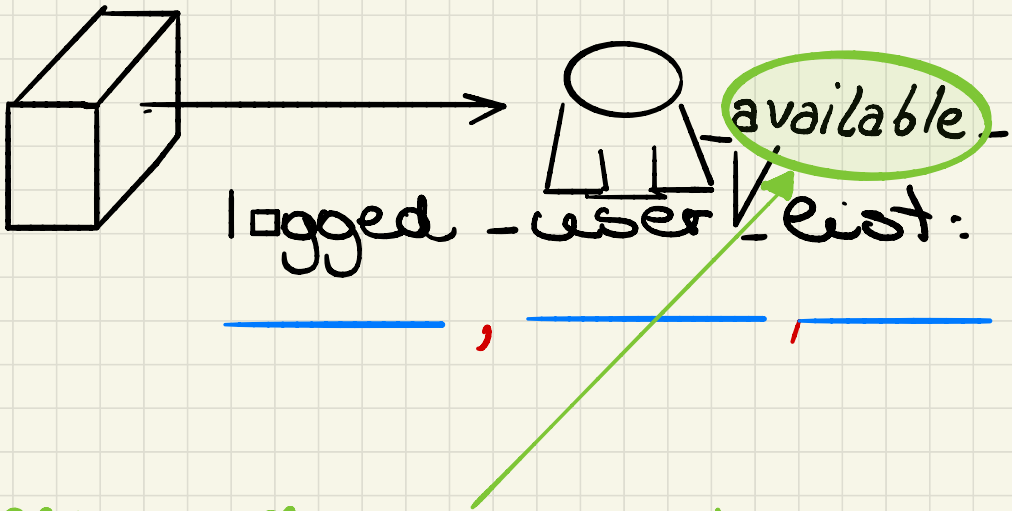
ERROR_MSG:
CODE: _____
DESCRIPTION: ____

Possibilità:
- "OK" (0)
- ~~"invalid user" (1)~~
- ~~"wrong password" (2)~~

Dopo il login il server invia all'utente la lista degli utenti online:

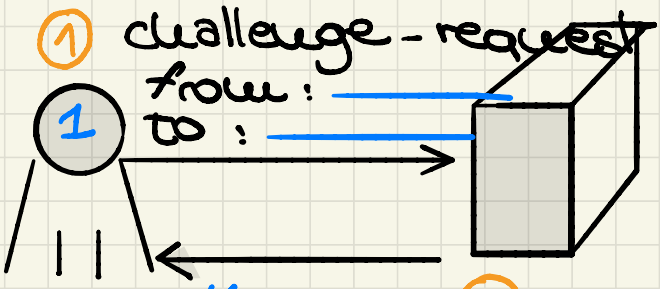
server: If (login_result == "OK"){
1- "sends_login_result_to_user";
2- sends_logged_available_user
3- add_user_to_logged_user(user);

in questo modo non inviamo all'utente se stesso



Assumo che un utente non può giocare con un utente che sta già giocando.

Invio di una sfida e risposta dell'altro utente:

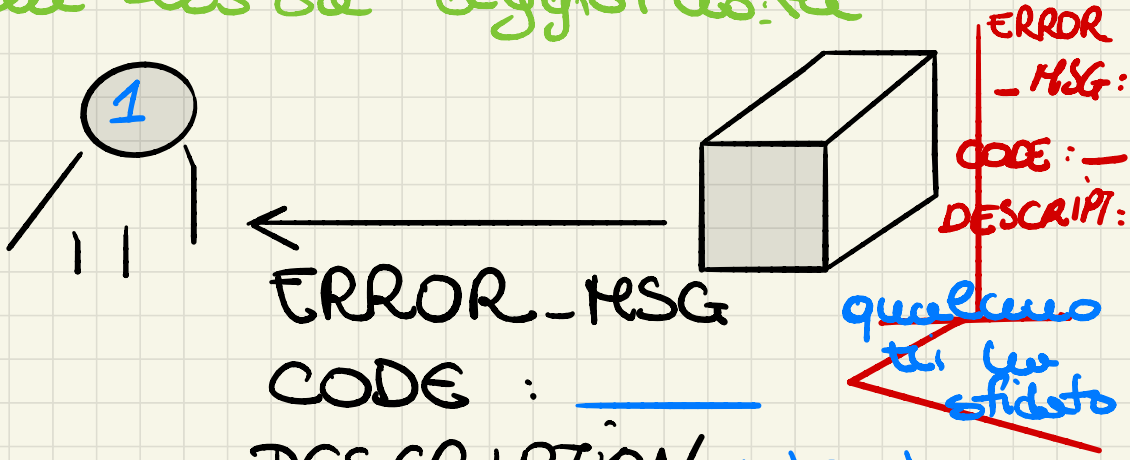


Appena il server riceve una request user 1 e 2 devono essere in stato pending ma anche il mio stato

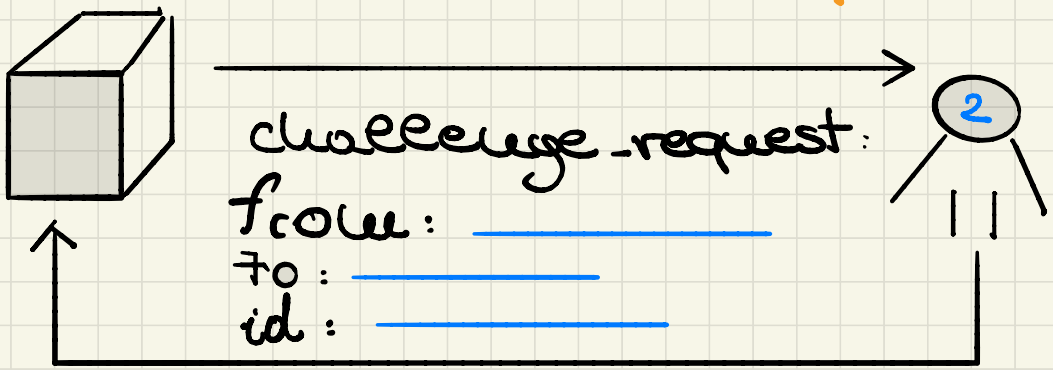
2 controllo che l'avversario non sia già impegnato in un'altra partita o non abbia fatto logout

3a (l'avversario ha inviato una nuova partita / ha fatto logout)

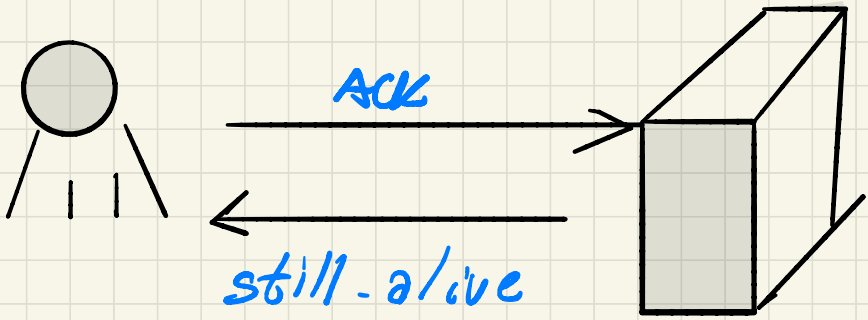
In questo caso secondo me va aggiunta la possibilità per un utente di richiedere la lista aggiornata

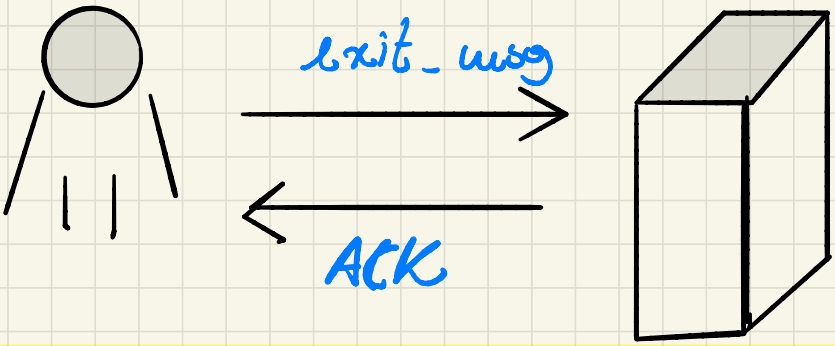


3b - l'utente è disponibile

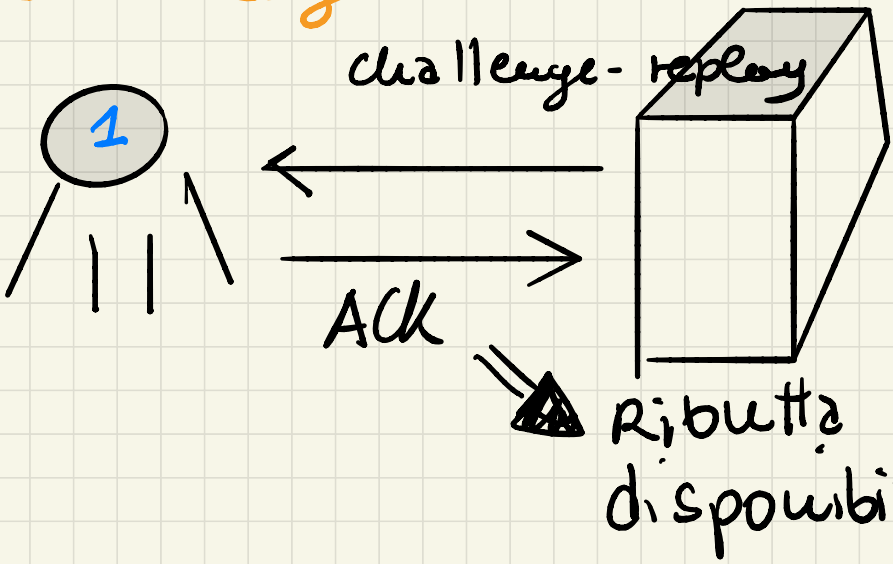


challenge_replay:
from: _____
to: _____
status: _____ id: _____

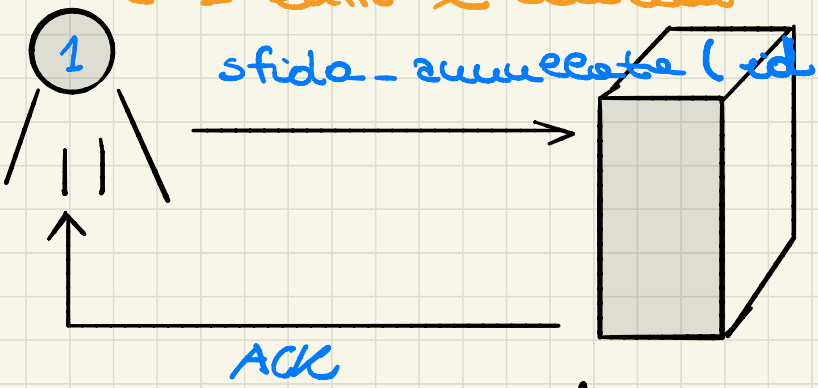




4a - L'utente sfidato rifiuta la sua challenge



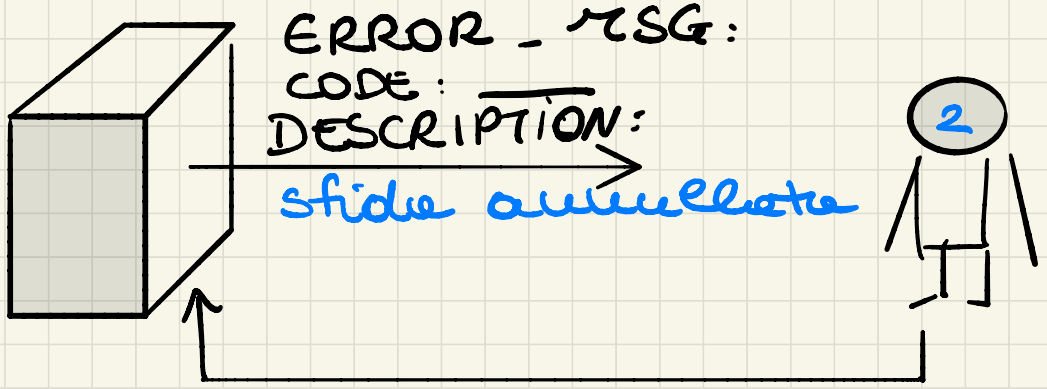
- Caso in cui utente 2 non risponde a 1 entro x secondi



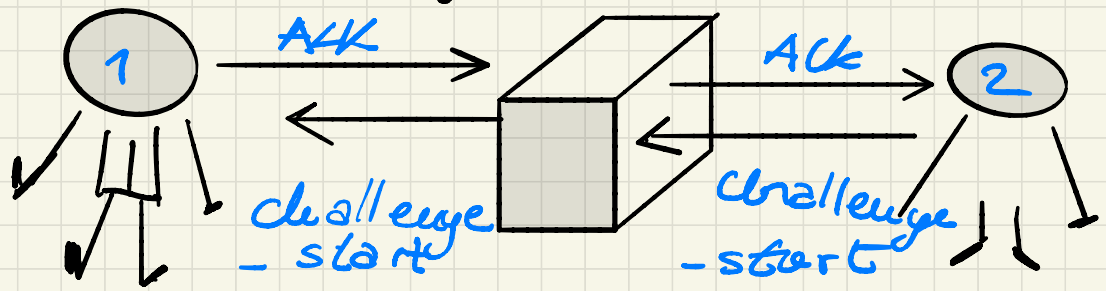
utente 1 di nuovo online

Una struttura x ogni challenge

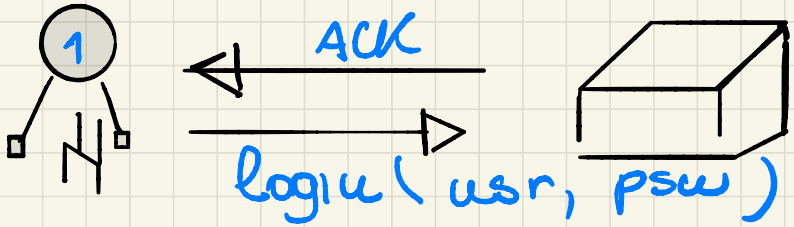
FROM	TO	STATUS
USR 1	USR 2	ANNULLATA



Il server sorteggia e l'utente che inizia



terminare della partita o crash

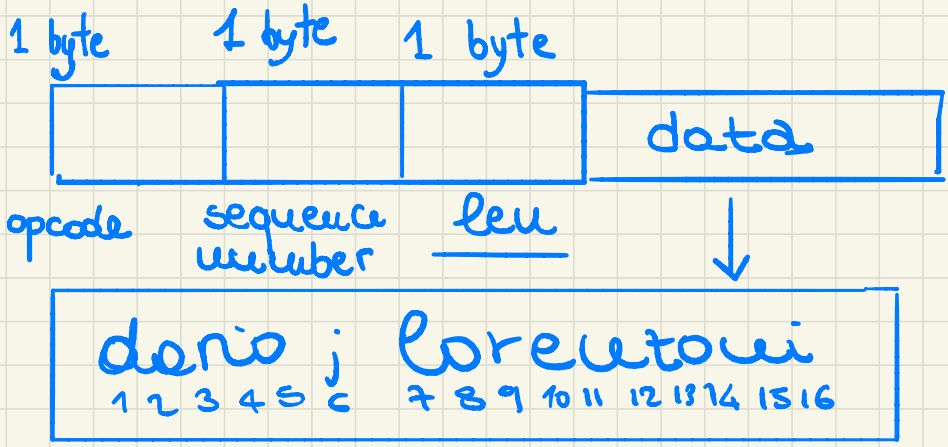


→ se un utente fa exit

→ se un utente gioca

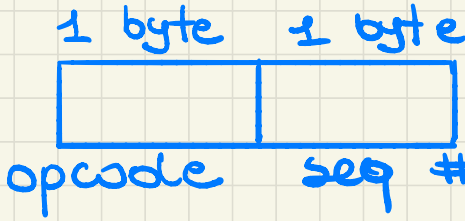
} OFFLINE

- logiu 255



memcpy (dest, src+3, len)

- logiu - OK / - NO

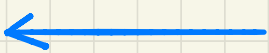


client

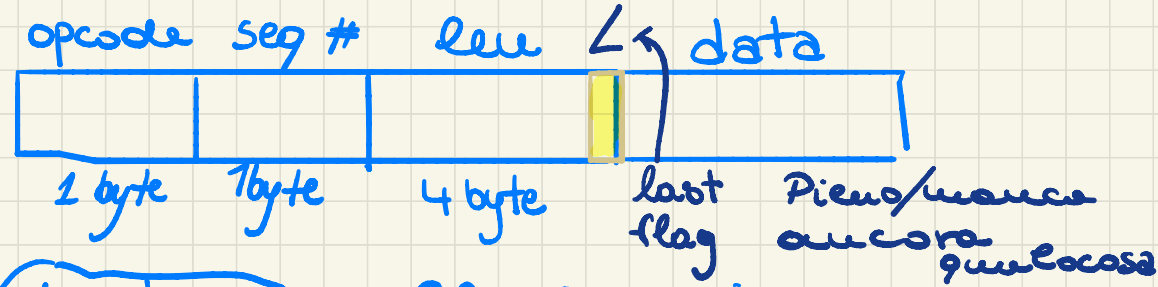
seq #

r = (seq # + 1) % 256

seq # = round()

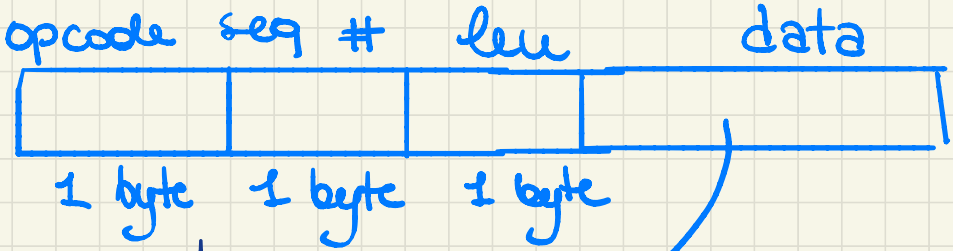


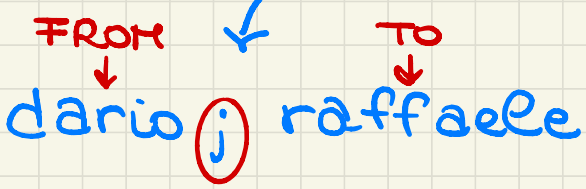
- available_usr_list



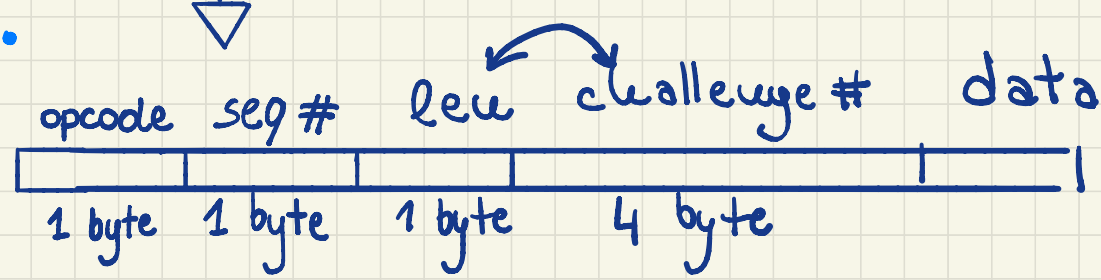
1. dario (j) raffaele ; riccardo

- challenge_req

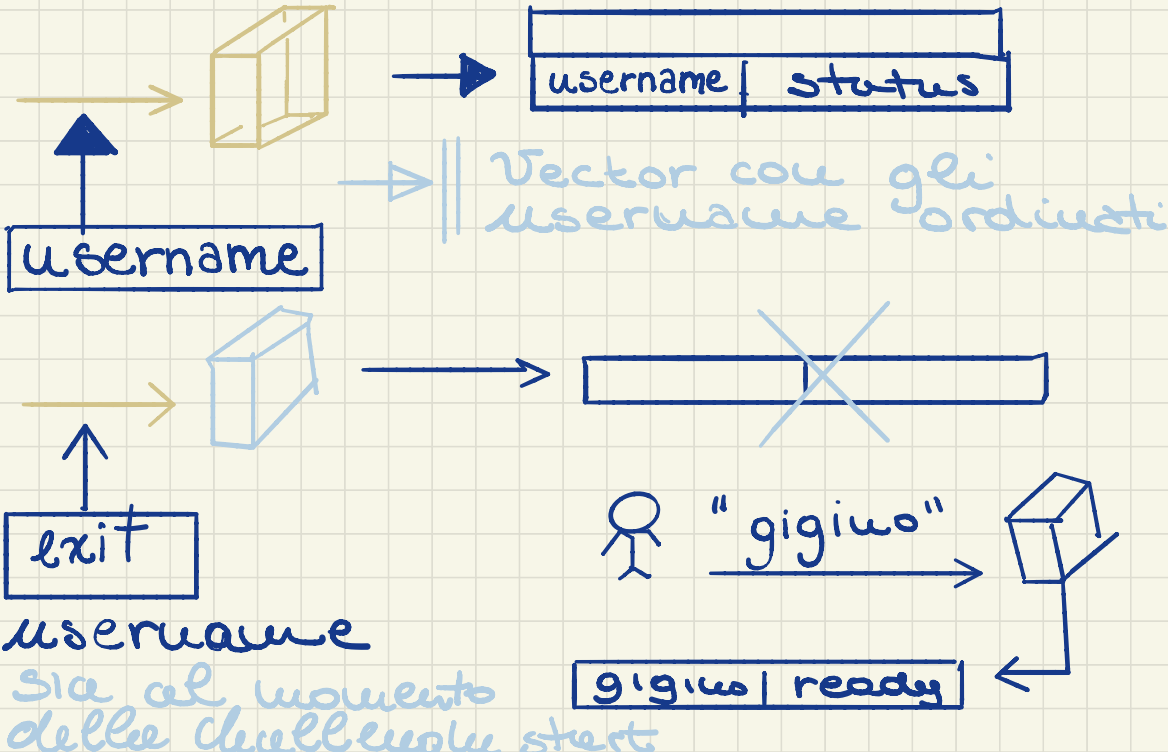




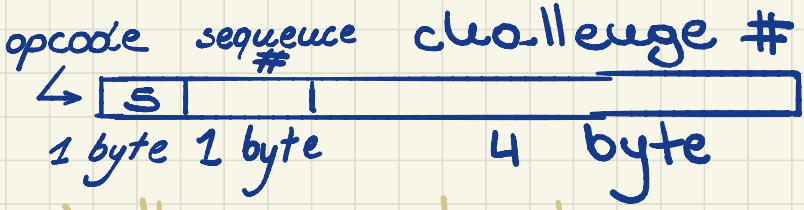
-



STRUTTURA DATI X UTENTI

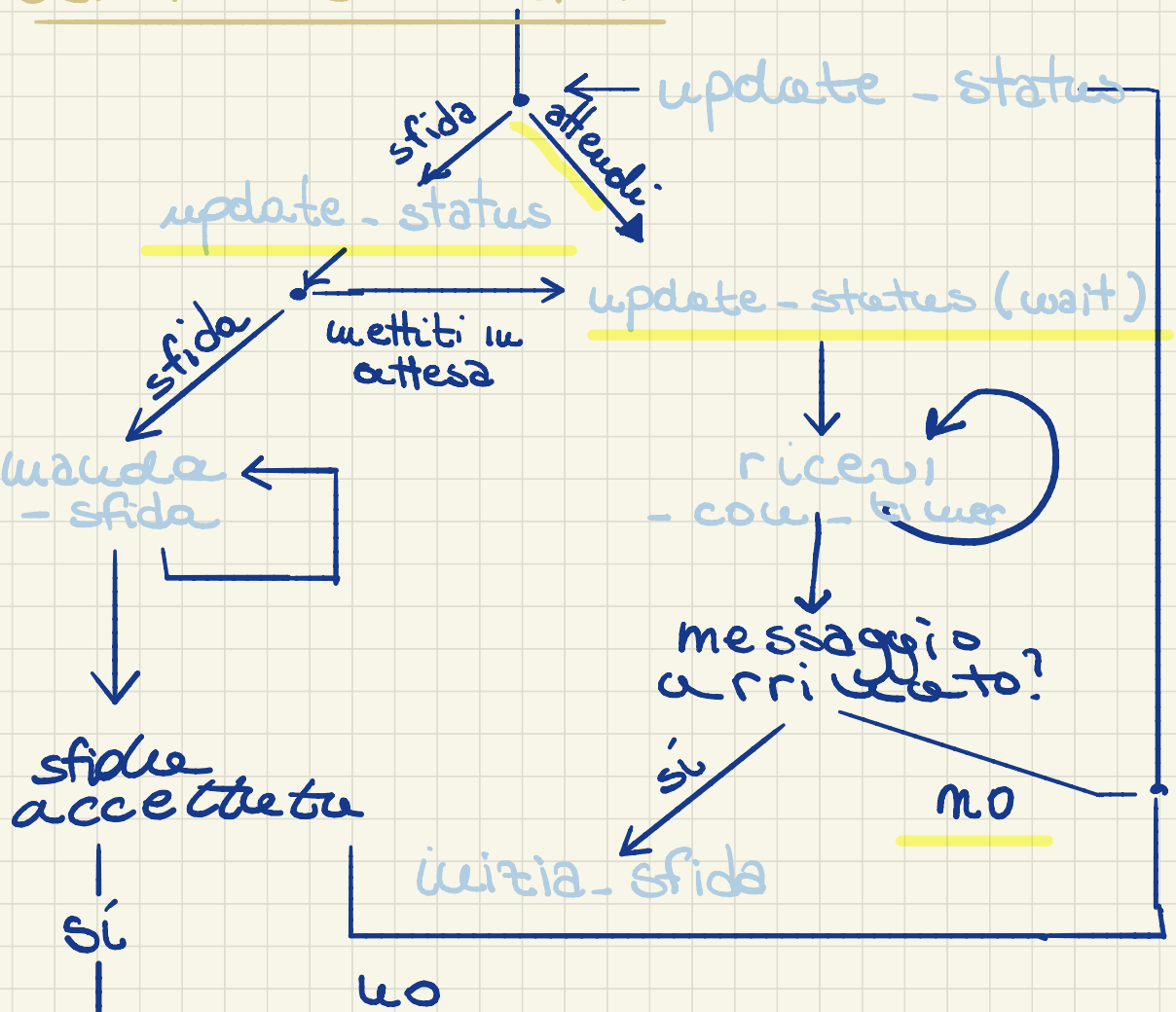


o challenge accepted:



- challenge start:

SCHEMA DEL CLIENT:



- update status

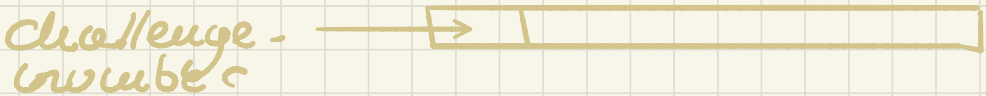
opcode	dlg#	user	status-code

status code:

- 0 → idle
- 1 → challenging
- 2 → waiting

STRUTTURA DATI x CHALLENGE:

challenge:



entry viene eliminata:

- dopo challenge_start
- dopo challenge_response x GiGino (quindi utente a rifiuta)
- quando arriva il mex timer expired.