XARXES

Pràctica 1 **Programació d'aplicacions de xarxa**

Curs 2021-2022

Pere Muñoz Figuerol
48252062V

Universitat de Lleida

Índex de continguts

Índex de figures

1. Estructura del client i del servidor

Consideracions a tenir en compte envers els diagrames de blocs:

- Cada bloc està encapçalat per un títol que resumeix la funció d'aquest.
- A més a més, conté diversos noms de funcions, escrits en cursiva, que representen algunes de les funcions encarregades de dur a terme la funció del bloc.
- Els noms de funcions en negreta representen la funció principal de cada bloc, per altra banda, les altres, poden ser considerades funcions auxiliars.

1.1. Estructura del client

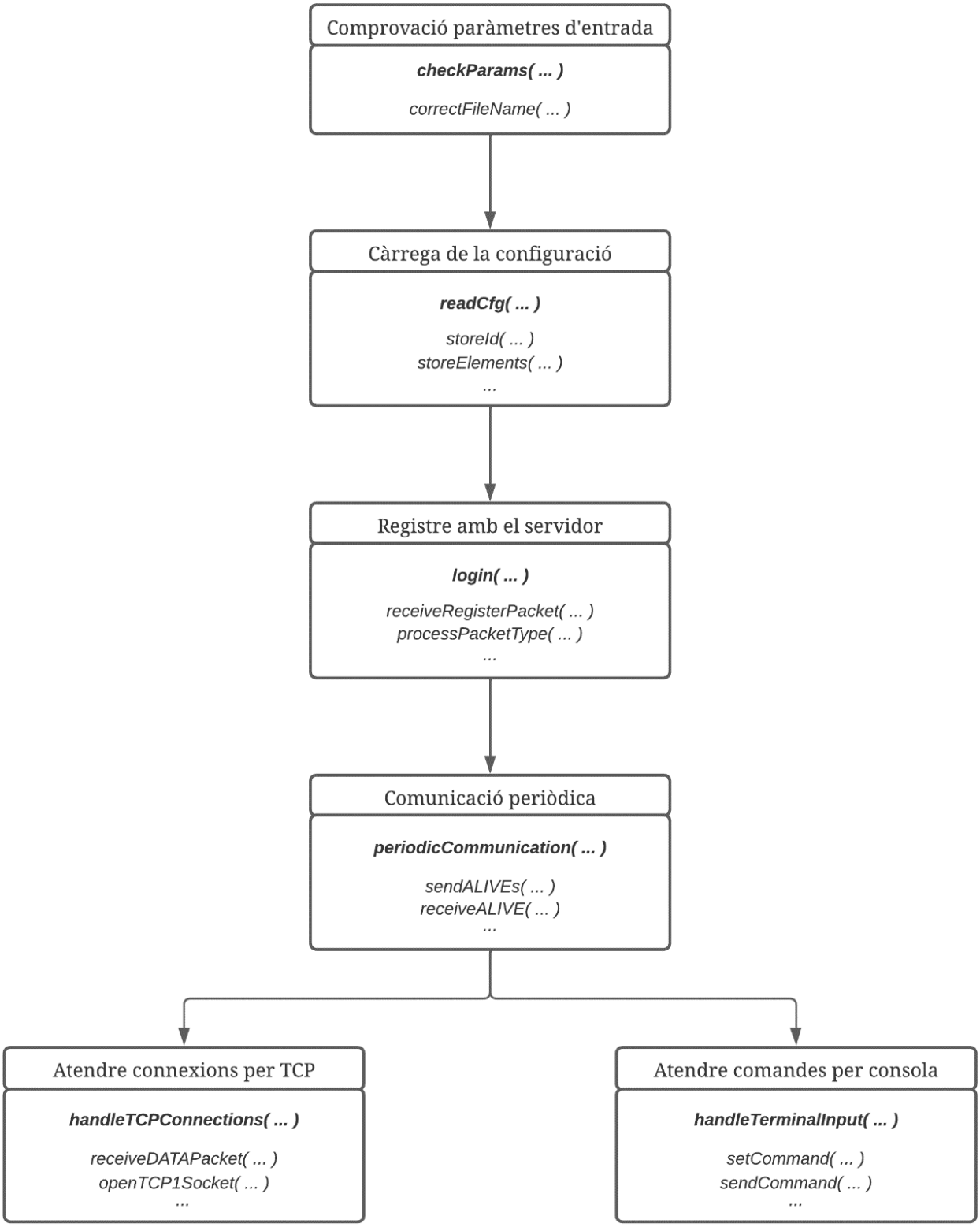

Figura 1. Diagrama de blocs de l'estructura del client

1.2. Estructura del servidor

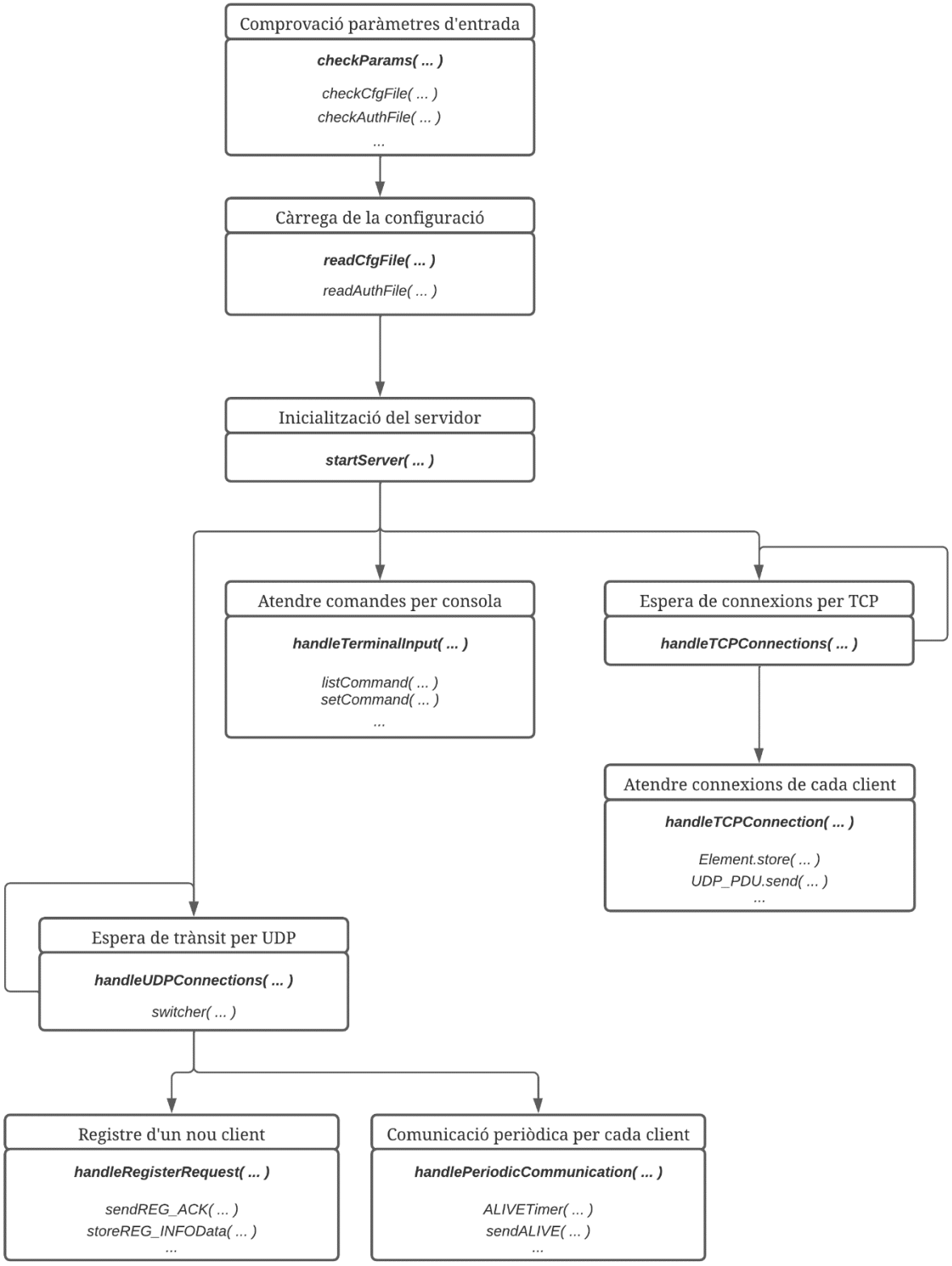

Figura 2. Diagrama de blocs de l'estructura del servidor

2. Estratègia emprada en la comunicació periòdica

2.1. Comunicació periòdica en el client

La comunicació periòdica en el client s'inicia amb la funció *periodicCommunication( )*, el què fa és, una vegada el client s'ha registrat, envia el primer paquet *ALIVE* al servidor i n'espera la resposta.
Si aquesta és rebuda i de manera correcta, el client passa a l'estat *SEND_ALIVE* i crea un fil que executarà la funció *sendALIVEs( )*.

Com el seu nom bé indica, aquesta funció trametrà al servidor paquets *ALIVE* correctes cada *V* segons. A més a més, per cada paquet remès, crearà un altre fil que en controli la seva respectiva resposta.

Aquesta resposta és monitorada per la funció *receiveALIVE( )*, que en cas de rebre un *ALIVE* correcte posarà el comptador que emmagatzema el nombre d'*ALIVE* perduts a zero. En cas de no rebre resposta, el comptador augmentarà i si sobrepassa el valor *S*, la funció encarregada de l'enviament periòdic d'*ALIVE*, *sendALIVEs( )*, pararà el seu enviament i matarà tots els fils que s'estiguin executant per poder començar un nou procés de registre. En cas de rebre un *ALIVE* erroni, la mateixa funció *receiveALIVE( )* farà el procés de matar els fils esmentat amb anterioritat.

2.2. Comunicació periòdica en el servidor

La comunicació periòdica en el servidor és controlada principalment per dues funcions, *handlePeriodicCommunication(...)* i *ALIVETimer(...)*.

Aquesta primera el que fa és actualitzar una variable booleana que té cada client, anomenada *ALIVEReceived*.
Si el servidor rep un paquet *ALIVE* correcte d'un client en concret, posarà la variable a *True* i li respondrà.
Si rep un paquet *ALIVE* erroni, es passarà el client a estat *DISCONNECTED*.

La segona funció, *ALIVETimer(...)*, actua com un comptador, que cada *V* segons comprova si d'un client en concret s'ha rebut un paquet *ALIVE*, comprovant la seva variable booleana *ALIVEReceived*.
A continuació, la variable es retornarà al valor *False* per la comprovació del següent. Aquesta funció és executada per un fil, un per cada client, que s'inicia just en el moment que un client es registra al servidor de manera correcta.

Evidentment, es distingeix si un client encara ha de rebre el primer *ALIVE* o no, per així ajustar el temps d'espera corresponent, a *W* i *V* respectivament.

En cas que es pari de rebre més de *S* paquets *ALIVE* consecutius d'un client en concret, aquest es posarà a estat *DISCONNECTED*.

3. Diagrama d’estats del protocol UDP

Consideracions a tenir en compte envers el diagrama d’estats:

- En qualsevol moment es pot finalitzar l’execució, tant del client com del servidor, fet que faria que en tornar-se a executar es reiniciés des de l’estat inicial.

- Només s’han esmentat les accions que són exclusivament del protocol UDP. Altres, com podrien ser errors en l’execució de la comanda *send* o paquets erronis rebuts del servidor a través de TCP, no s’han inclòs en el diagrama.

- Ja que el client i el servidor mantenen un flux d’estats lleugerament diferent, s’ha estructurat el diagrama utilitzant diferents colors i estils de línia per poder representar correctament aquestes variàncies. Tota la informació necessària per poder llegir i interpretar de manera idònia el diagrama està especificada a la llegenda d’aquest.

3.1. Diagrama d'estats

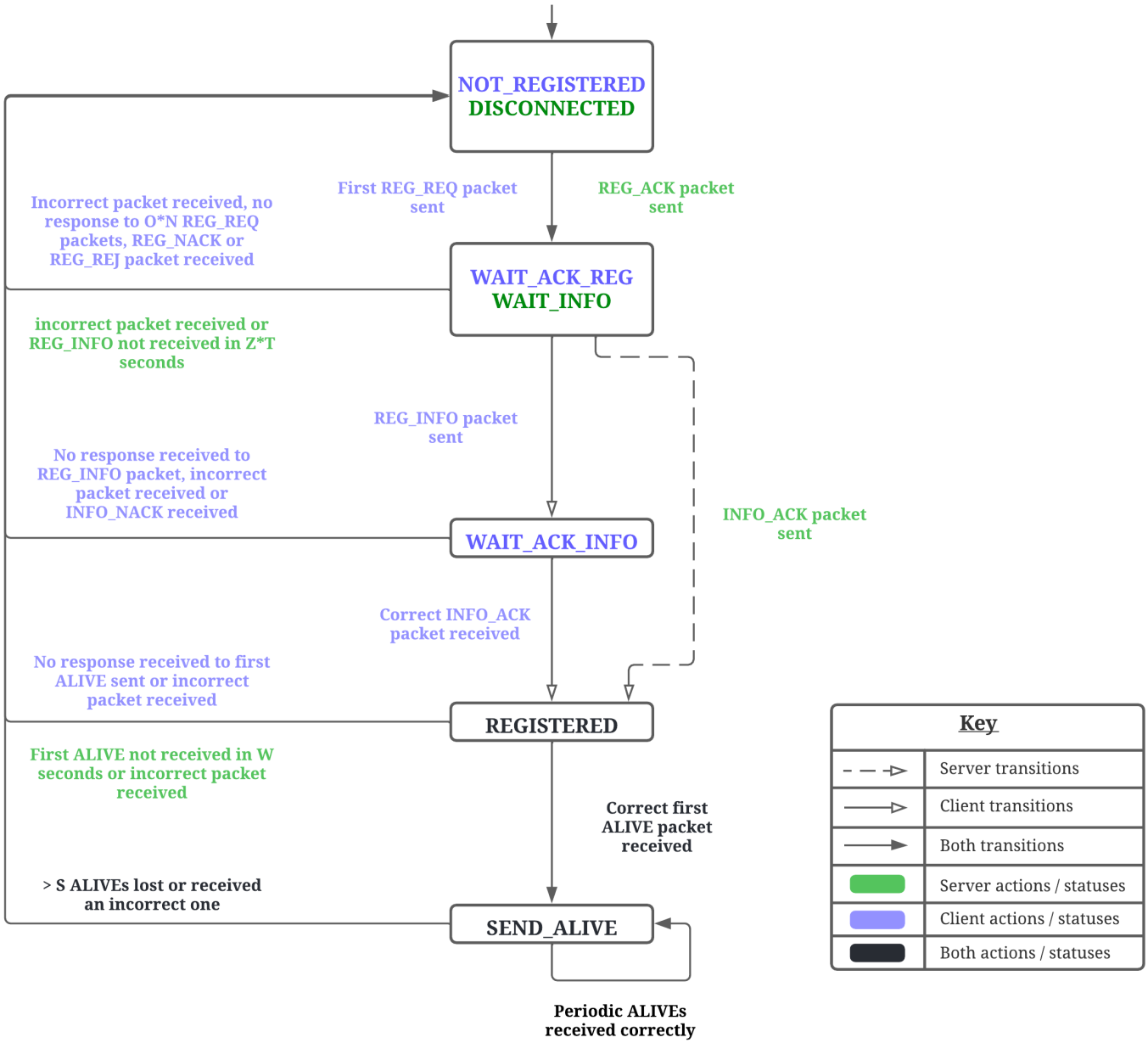

Figura 3. Diagrama d'estats del protocol UDP

4. Algunes consideracions

- Tant en el client com en el servidor no importa l'ordre en els paràmetres d'entrada, sempre que s'introdueixin utilitzant la sintaxi correcta especificada en l'enunciat.

- S'han definit quatre tipus de sortides de text per consola:

    - *DEBUG*: Informa de diversos esdeveniments interns de funcionament, però que no són rellevants per a l'usuari habitual. Està diferenciat en color blau al client i groc al servidor.

    - *INFO*: Informa dels canvis d'estat del client. Està diferenciat en color gris al client i lila subratllat al servidor.

    - *ERROR*: Informa d'alguns errors de funcionament (errors en l'obertura de *sockets*, de connexió per TCP...) i també d'alguns esdeveniments útils per ajudar la interacció amb l'usuari (introduir comandes errònies o mal formades per consola...). Està diferenciat en color vermell en ambdós dispositius.

    - *OK*: Informa a l'usuari del resultat correcte d'alguns esdeveniments, com per exemple, l'ús de la comanda *send* en el client. Està diferenciat en color verd en ambdós dispositius.

- Si s'introdueix una comanda errònia, tant en el servidor com en el client, es notificarà per consola i es mostraran les comandes disponibles amb una petita descripció amb l'objectiu de facilitar la interacció.