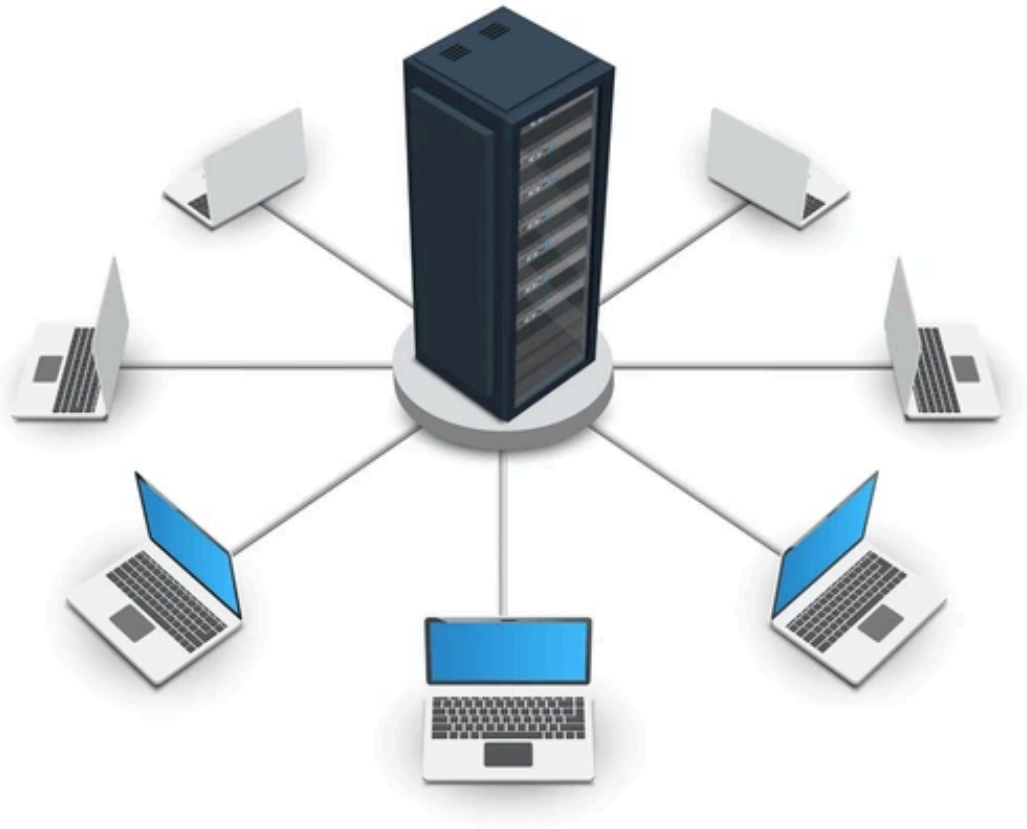
Exercici 1 – Arquitectura Distribuïda: Sistema Master–Client



Assignatura: Arquitectura Distribuïda
Grau: Enginyeria Informàtica (4t curs)
Autors: Aniol Vergés, Aleix Batchelli
Curs acadèmic: 2025–2026

Índex

1. Introducció

Aquest exercici té com a objectiu implementar un sistema distribuït senzill que permeti compartir una variable entre múltiples processos, complint les restriccions establertes a l'enunciat. El sistema es desenvolupa utilitzant comunicació mitjançant sockets TCP, sense fer ús de cap mecanisme d'IPC addicional, i serveix com una primera aproximació pràctica als principals reptes dels sistemes distribuïts.

La pràctica permet treballar conceptes fonamentals com la comunicació entre processos, la coordinació de múltiples nodes, l'exclusió mútua en l'accés a recursos compartits i la consistència de dades en presència de concurrència.

2. Objectius de la pràctica

Els objectius principals d'aquest exercici són:

- Implementar un sistema distribuït basat en sockets seguint una arquitectura Master–Client.
- Permetre operacions de lectura i actualització sobre una variable compartida.
- Garantir l'exclusió mútua quan diversos processos intenten modificar el recurs compartit.
- Analitzar el comportament del sistema en termes de consistència, rendiment i escalabilitat.
- Reflexionar sobre les limitacions de la solució implementada i possibles millores.

3. Descripció general de la solució

La solució implementada segueix una arquitectura **Master–Client** centralitzada. Existeix un únic procés Master que manté el valor correcte de la variable compartida i coordina totes les peticions dels clients. Els clients es connecten al Master mitjançant sockets TCP i poden tenir dos comportaments diferents:

- **Client de només lectura (r):** sol·licita periòdicament el valor de la variable compartida.
- **Client de lectura i escriptura (rw):** sol·licita accés exclusiu, actualitza el valor i allibera el bloqueig.

Aquesta arquitectura no pretén ser òptima en termes d'escalabilitat, sinó que prioritza la simplicitat i la claredat per entendre els mecanismes bàsics de coordinació en sistemes distribuïts.

4. Arquitectura del sistema

El sistema està format pels següents components principals:

- **Master:** procés servidor que escolta en un port TCP (127.0.0.1:12345), accepta connexions entrants i gestiona totes les operacions sobre la variable compartida.
- **Clients:** processos que es connecten al Master i executen operacions de lectura o escriptura segons el seu rol.
- **Protocol de missatges:** conjunt de missatges definits per permetre la comunicació entre clients i servidor.

Tota la comunicació passa exclusivament a través del Master, que actua com a punt únic de coordinació. Aquesta decisió simplifica el disseny i garanteix un control centralitzat sobre l'estat del sistema.

A la Figura 1 es mostra el diagrama d'arquitectura del sistema implementat, on es pot observar la relació entre el Master i els diferents clients, així com la gestió centralitzada de la variable compartida.

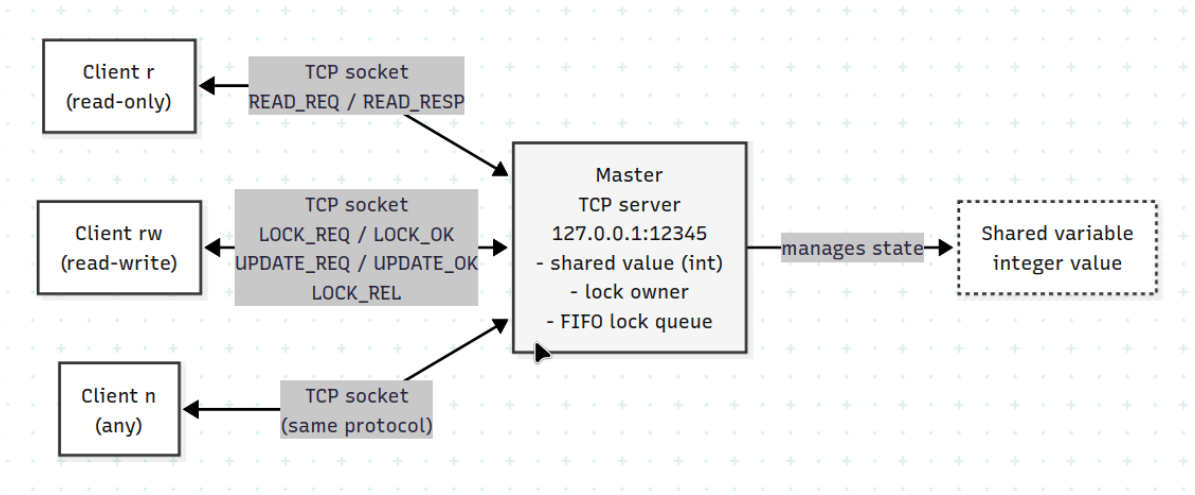


Figura 1: Arquitectura general del sistema Master–Client.

5. Protocol de comunicació i funcionament de l'algoritme

La comunicació entre clients i servidor es basa en un protocol definit mitjançant un conjunt de missatges identificats per un enumerat. Els principals missatges utilitzats són:

- READ_REQ / READ_RESP: permeten als clients obtenir el valor actual de la variable compartida.
- LOCK_REQ / LOCK_OK / LOCK_REL: gestionen la sol·licitud, concessió i alliberament del bloqueig.
- UPDATE_REQ / UPDATE_OK: permeten actualitzar el valor de la variable un cop obtingut el bloqueig.
- ERR: indica errors de comunicació o d'estat.

El funcionament general de l'algoritme és el següent:

1. Un client rw sol·licita el bloqueig al Master.
2. El Master concedeix el bloqueig si està disponible o posa la petició en espera.
3. Un cop obtingut el bloqueig, el client actualitza el valor de la variable.
4. El client allibera el bloqueig i el Master el concedeix al següent client en espera.

Aquest flux garanteix que les operacions d'escriptura es realitzin de manera controlada.

Tal com es mostra a la Figura 2, els clients de tipus rw han de sol·licitar un bloqueig abans de modificar la variable compartida. El Master concedeix el bloqueig de forma exclusiva i gestiona una cua de peticions per evitar condicions de cursa. Les operacions de lectura es realitzen sempre a través del Master, garantint la consistència de dades.

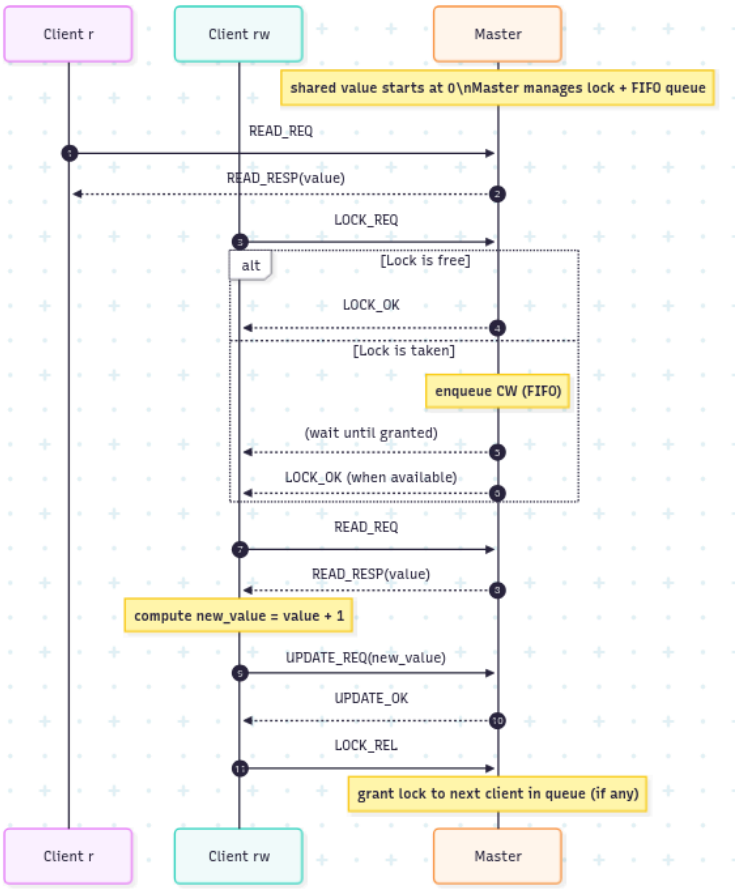


Figura 2: Diagrama de seqüència del funcionament del protocol de comunicació entre clients i Master.

6. Exclusió mútua i gestió de la concurrència

L'exclusió mútua és un dels aspectes centrals d'aquest exercici. El Master implementa un mecanisme de bloqueig que assegura que només un client amb permisos d'escriptura pot modificar la variable compartida al mateix temps.

Quan un client envia un LOCK_REQ, el Master comprova si el bloqueig està lliure. En cas afirmatiu, el concedeix immediatament. Si el bloqueig ja està ocupat, la petició s'afegeix a una cua FIFO. Quan el client que posseeix el bloqueig l'allibera mitjançant LOCK_REL, el Master concedeix el bloqueig al següent client de la cua.

Aquest mecanisme evita condicions de cursa i garanteix que no es produeixin actualitzacions simultànies sobre la variable compartida.

7. Consistència de dades

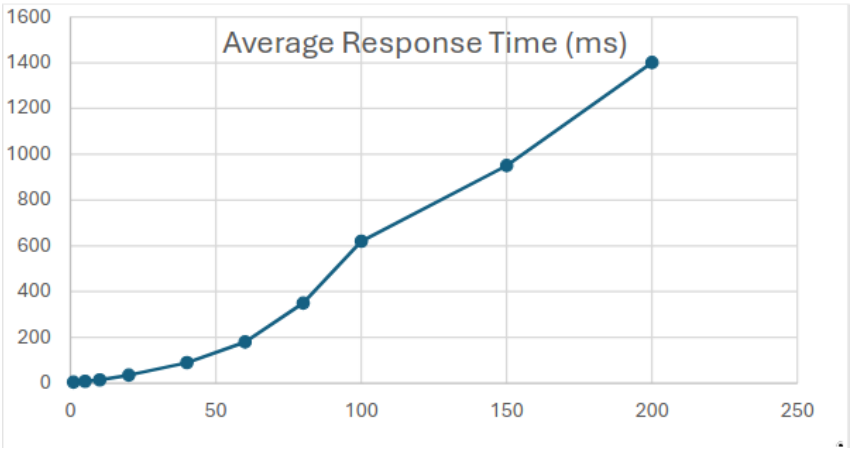
El sistema garanteix una **consistència forta**, ja que el Master és l'únic procés que manté i modifica el valor de la variable compartida. Tots els clients han de sol·licitar el valor actual al Master abans d'utilitzar-lo, complint així les restriccions de l'enunciat.

Aquesta solució assegura que tots els clients observen el mateix estat del sistema en tot moment, a costa de limitar la possibilitat de paral·lelisme en les operacions d'escriptura.

8. Rendiment i escalabilitat

El rendiment del sistema està directament relacionat amb el nombre de clients connectats. A mesura que augmenta el nombre de processos, el Master ha de gestionar més peticions, fet que incrementa el temps de resposta de manera aproximadament lineal.

El Master es converteix en un coll d'ampolla, especialment quan hi ha molts clients de tipus rw, ja que les operacions d'escriptura són necessàriament exclusives. Tot i això, aquest comportament és coherent amb els objectius de l'exercici i amb el disseny centralitzat adoptat.



9. Limitacions i possibles millores

La principal limitació del sistema és la presència d'un punt únic de fallada: si el Master deixa de funcionar, tot el sistema queda inoperatiu. A més, l'arquitectura centralitzada limita l'escalabilitat i el rendiment en entorns amb molts clients.

Durant la fase d'ideació es va considerar la possibilitat d'introduir múltiples Masters per distribuir la càrrega. No obstant això, aquesta solució requeriria mecanismes addicionals per garantir l'exclusió mútua i la consistència de dades entre servidors, com ara protocols de consens o elecció de líder, fet que s'ha considerat fora de l'abast d'aquest exercici.

10. Testing i validació

El sistema s'ha validat mitjançant proves manuals executant múltiples clients de manera simultània. S'ha comprovat que:

- Només un client pot modificar la variable compartida al mateix temps.
- Els clients de només lectura reben sempre el valor correcte.
- El sistema continua funcionant correctament quan un client es desconnecta mentre espera el bloqueig.

Aquestes proves permeten verificar el correcte funcionament del mecanisme d'exclusió mútua i la consistència del sistema.

11. Conclusions

En aquest exercici s'ha implementat un sistema distribuït senzill basat en una arquitectura Master–Client per compartir una variable entre múltiples processos. La solució desenvolupada permet aplicar de manera pràctica la comunicació mitjançant sockets i la coordinació entre processos.

L'ús d'un Master centralitzat permet gestionar correctament l'exclusió mútua, garantint que només un client de tipus rw pot modificar la variable compartida al mateix temps. Aquest mecanisme evita condicions de cursa i assegura un comportament correcte del sistema sota concurrència.

Pel que fa a la consistència de dades, el sistema garanteix una consistència forta, ja que tots els accessos a la variable passen pel Master, que manté sempre el valor actualitzat. Aquesta decisió simplifica el disseny, tot i limitar l'escalabilitat.

En conclusió, el sistema compleix els objectius plantejats a l'enunciat i constitueix una bona base per entendre els principals reptes dels sistemes distribuïts, així com els compromisos entre simplicitat, rendiment i escalabilitat.