Exercici 4 de Sistemes Distribuïts: Replicació Epidèmica

Curs Acadèmic 2025-2026

Aniol Vergés Herrera
Aleix Batchellí I Abad
La Salle - Universitat Ramon Llull

Índex

1 Introducció

Aquest informe detalla el disseny i la implementació d'una aplicació de sistemes distribuïts centrada en la **Replicació Epidèmica**. L'objectiu d'aquest exercici és implementar una arquitectura multicapa on la consistència de les dades varia segons les capes, des d'una consistència forta al nucli fins a una consistència causal a les capes exteriors.

El sistema consta de 7 nodes organitzats en 3 capes diferents, dissenyats per simular una topologia de xarxa realista on les estratègies de propagació difereixen (Ansiosa vs. Mandrosa).

2 Arquitectura del Sistema

2.1 Topologia

El sistema consta de tres capes que contenen un total de 7 nodes.

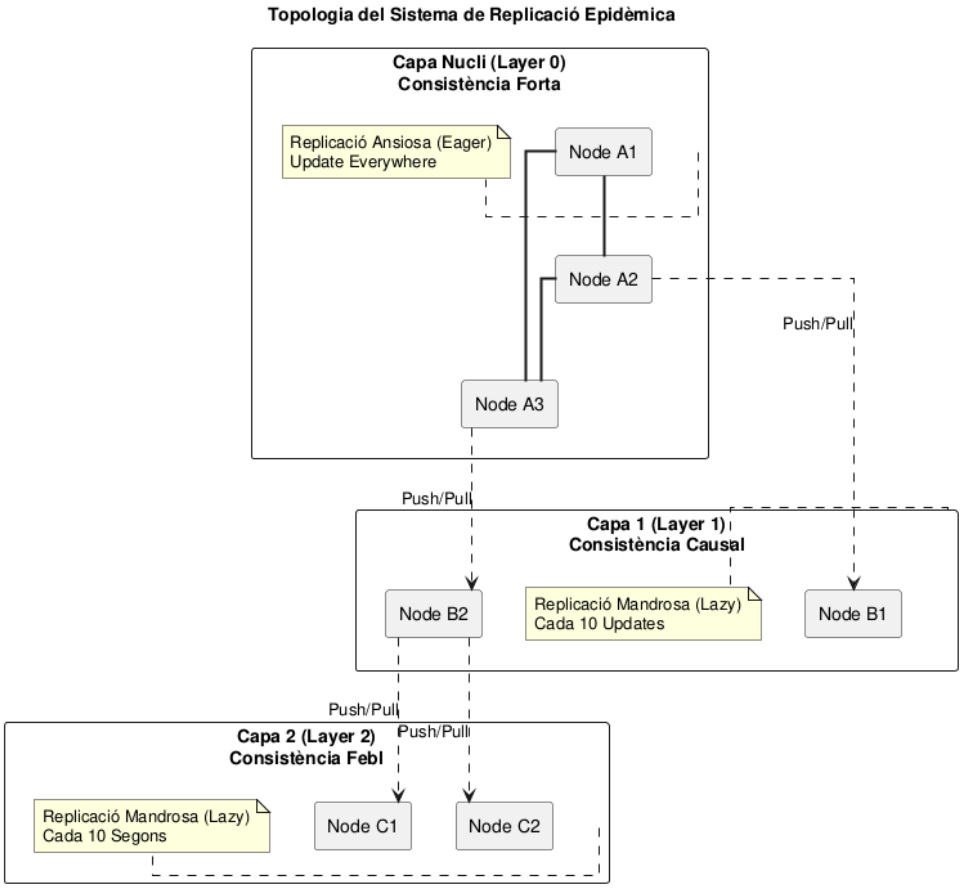

Figura 1: Diagrama de la topologia de la xarxa mostrant les 3 capes i les connexions entre els nodes A, B i C.

- **Capa Nucli (Capa 0):** Nodes A1, A2, A3. Aquests nodes mantenen les versions més actualitzades de les dades ($V = n$) i apliquen una consistència forta.
- **Capa 1:** Nodes B1, B2. Aquests nodes mantenen versions més antigues ($V = n-1$) i utilitzen consistència causal.

- **Capa 2:** Nodes C1, C2. Aquests nodes mantenen les versions més antigues ($V = n - 2$) i representen el nivell de consistència més feble.

2.2 Models de Consistència i Replicació

El sistema implementa una estratègia de replicació híbrida tal com es defineix als requisits:

1. **Capa Nucli (Nodes A):** Utilitza *Actualització a tot arreu, Activa i Replicació Ansiosa (Eager)*. Les actualitzacions es propaguen immediatament entre els nodes del nucli per garantir una consistència forta.
2. **Capa 1 (Nodes B):** Utilitza *Replicació Passiva, Còpia Primària i Mandrosa (Lazy)*. La sincronització es produeix cada 10 actualitzacions.
3. **Capa 2 (Nodes C):** Utilitza *Replicació Passiva, Còpia Primària i Mandrosa (Lazy)*. La sincronització es produeix cada 10 segons.

3 Detalls d'Implementació

3.1 Pila Tecnològica

La solució s'ha implementat en Python, utilitzant biblioteques estàndard per minimitzar les dependències:

- **Multiprocessing:** Utilitzat per simular nodes diferents com a processos separats dins d'una sola màquina.
- **Sockets:** Utilitzats per a la comunicació TCP entre nodes i clients.
- **WebSockets:** Utilitzats per al monitoratge en temps real de les rèpliques.

3.2 Lògica i Orquestració dels Nodes

El punt d'entrada del sistema, `main.py`, gestiona la configuració de la topologia i la creació de processos. Un registre mapeja els identificadors dels nodes a ports específics (començant des del 5001 per a A1), i un diccionari de topologia defineix les relacions entre veïns.

Figura 2: Estructura de classes dels nodes mostrant l'herència i els mètodes específics per a cada estratègia de replicació.

S'instancien classes diferents basant-se en el prefix de l'ID del node per gestionar la lògica de replicació diferenciada:

```
if node_id.startswith("A"):
    node_instance = CoreNode(node_id, my_port, neighbors)
elif node_id.startswith("B"):
    node_instance = Layer1Node(node_id, my_port, neighbors)
elif node_id.startswith("C"):
    node_instance = Layer2Node(node_id, my_port, neighbors)
```

Listing 1: Lògica d'Instanciació de Nodes (main.py)

Aquesta estructura assegura que `CoreNode` gestioni la replicació ansiosa, mentre que `Layer1Node` i `Layer2Node` implementen les seves respectives estratègies mandroses (basades en comptador vs. basades en temps).

3.3 Implementació del Client

L'aplicació client (`client.py`) és responsable d'analitzar les transaccions i encaminar-les a la capa adequada.

3.3.1 Anàlisi de Transaccions

Les transaccions es llegeixen des d'un fitxer o mitjançant entrada manual. El client distingeix entre tipus de transaccions basant-se en el format del valor:

- **Escriptura (Actualització):** Identificada per valors de coma flotant (ex., `49.53`). Aquestes s'encaminen **sempre** a la Capa Nucli (Capa 0), independentment de la capa sol·licitada a la cadena de transacció.
- **Només Lectura:** Identificada per valors enters (ex., `49`). Aquestes s'encaminen a la capa específica sol·licitada ($< f >$).

```
# Heur stica: Els floats s n actualitzacions (49.53)
is_write = '.' in val_str

# Les escriptures SEMPRE van al Nucli (Capa 0)
# Les lectures van a la capa especificada
target_layer = 0 if is_write else layer_id_requested
```

Listing 2: Heurística d'Encaminament del Client (client.py)

4 Protocols de Replicació

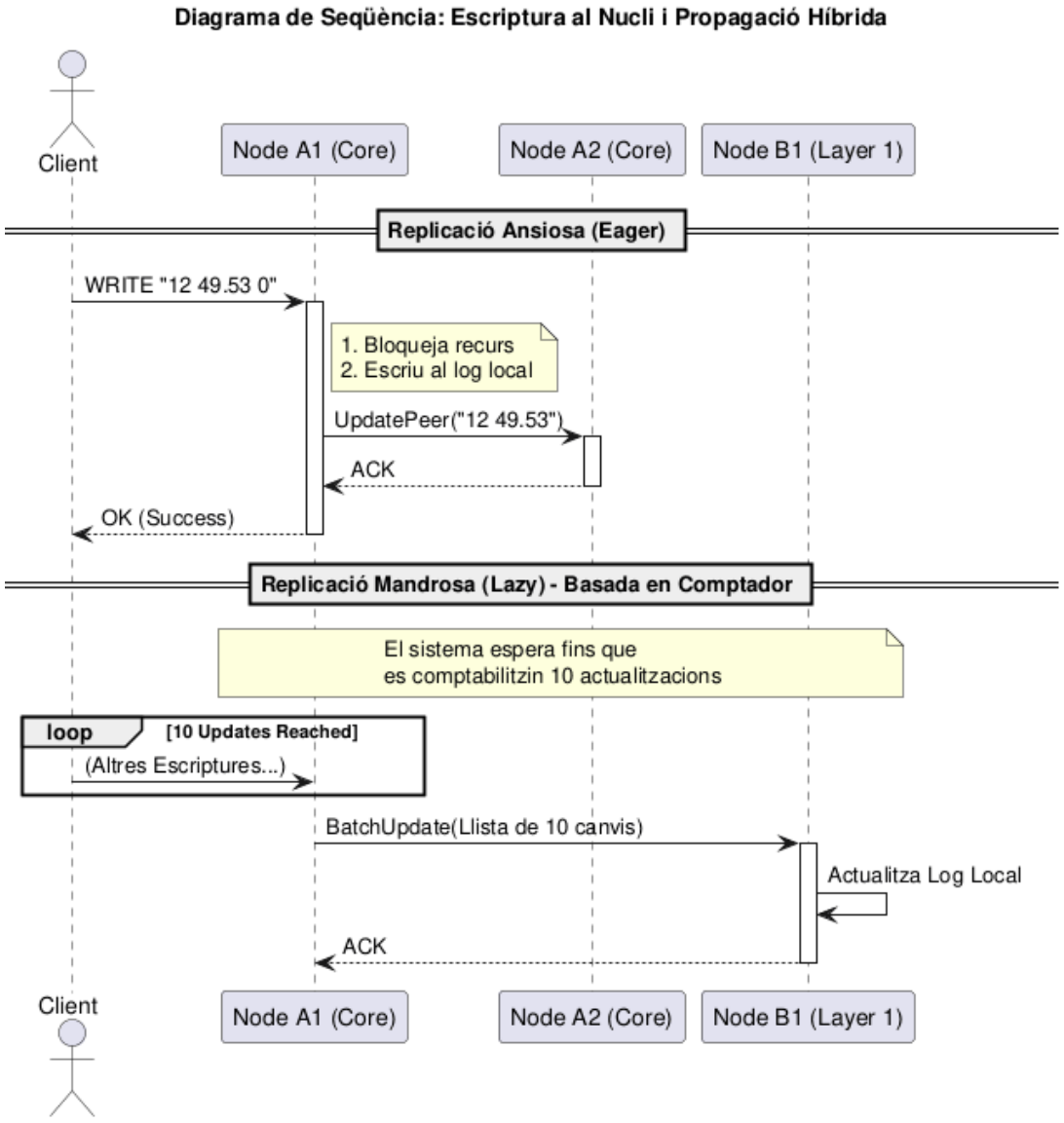

Figura 3: Diagrama de flux que il·lustra el flux d'una escriptura al nucli i la posterior propagació mandrosa a les capes exteriors.

4.1 Capa Nucli: Replicació Ansiosa

Els nodes A1, A2 i A3 formen una malla totalment connectada (conceptualment). Quan un node del nucli rep una sol·licitud d'escriptura, bloqueja el recurs i fa difusió (multicast) de l'actualització als seus parells immediatament. Això assegura que la capa nucli convergeixi ràpidament, mantenint el requisit de tenir la "versió més moderna".

4.2 Capa 1: Replicació Mandrosa Basada en Comptador

Els nodes B1 i B2 actuen com a còpies de seguretat (backups). No sol·liciten actualitzacions contínuament. En canvi, mantenen un comptador d'operacions. Un cop el sistema

registra 10 actualitzacions al Nucli, les dades s'envien a (o són recuperades per) la Capa 1. Això redueix el trànsit de xarxa a costa d'una obsolescència temporal ($V = n - 1$).

4.3 Capa 2: Replicació Mandrosa Basada en Temps

Els nodes C1 i C2 representen la frontera de la consistència eventual. Operen en un bucle de temporitzador. Cada 10 segons, inicien un procés de sincronització amb el seu node pare (B2). Això assegura que, fins i tot si el trànsit d'actualitzacions és baix, la capa exterior acabi convergint.

5 Monitoratge i Validació

5.1 Execució

El sistema es llança mitjançant la comanda `python main.py --all`, que genera 7 processos independents. El client s'executa per separat per emetre comandes de manera interactiva o mitjançant fitxers per lots.

5.2 Verificació de Convergència

Per verificar la naturalesa "Epidèmica"del sistema:

1. S'envia una ràfega d'escriptures al Nucli.
2. Observem actualitzacions immediates als nodes A.
3. Els nodes B s'actualitzen només després de la 10a escriptura.
4. Els nodes C s'actualitzen de manera asíncrona cada 10 segons.
5. Quan les actualitzacions cessen, tots els nodes acaben informant del mateix hash de versió als seus registres locals.

6 Conclusió

Aquest projecte demostra amb èxit els compromisos entre consistència i rendiment en sistemes distribuïts. Mitjançant la implementació d'una arquitectura multicapa amb estratègies de replicació híbrides, hem observat com la consistència forta (Nucli) requereix una elevada sobrecàrrega de comunicació, mentre que la consistència causal (Capes 1 i 2) permet un trànsit reduït a costa de la frescor de les dades.