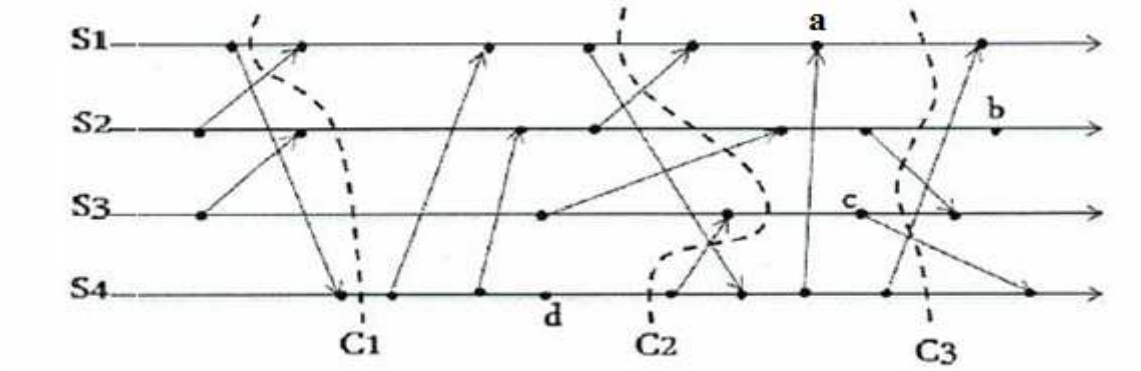
Université de Msila **2025 -2026**
Faculté des mathématiques et de l'informatique **1ère Année**
Département d'Informatique
Master RTIC

Epreuve de Moyenne Durée
Module : Systèmes Distribués

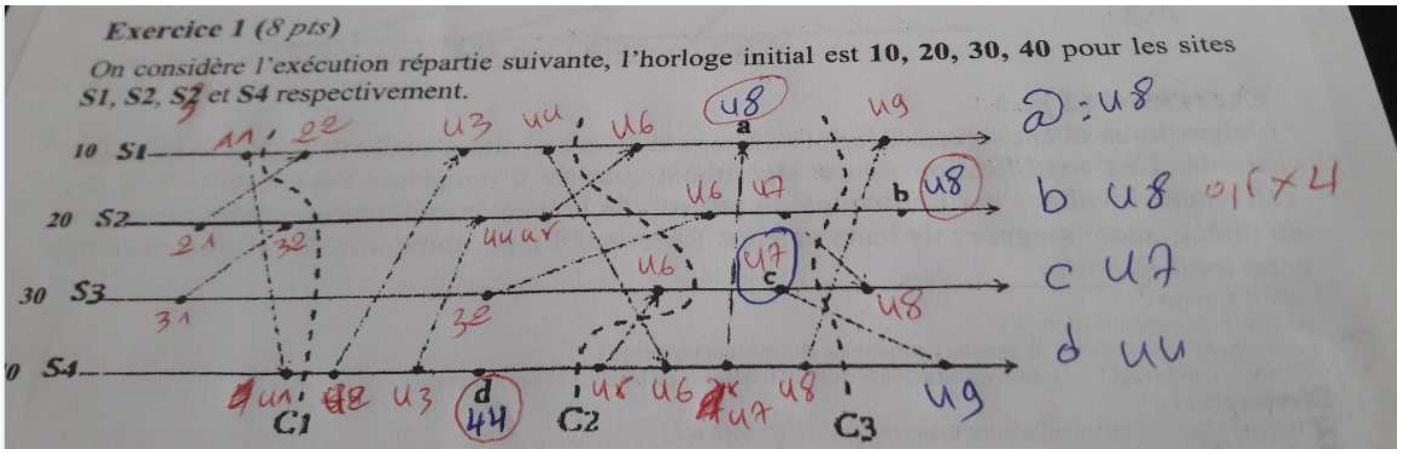
19-01-2024 Durée : 2H

Exercice 1 (*8 pts*)

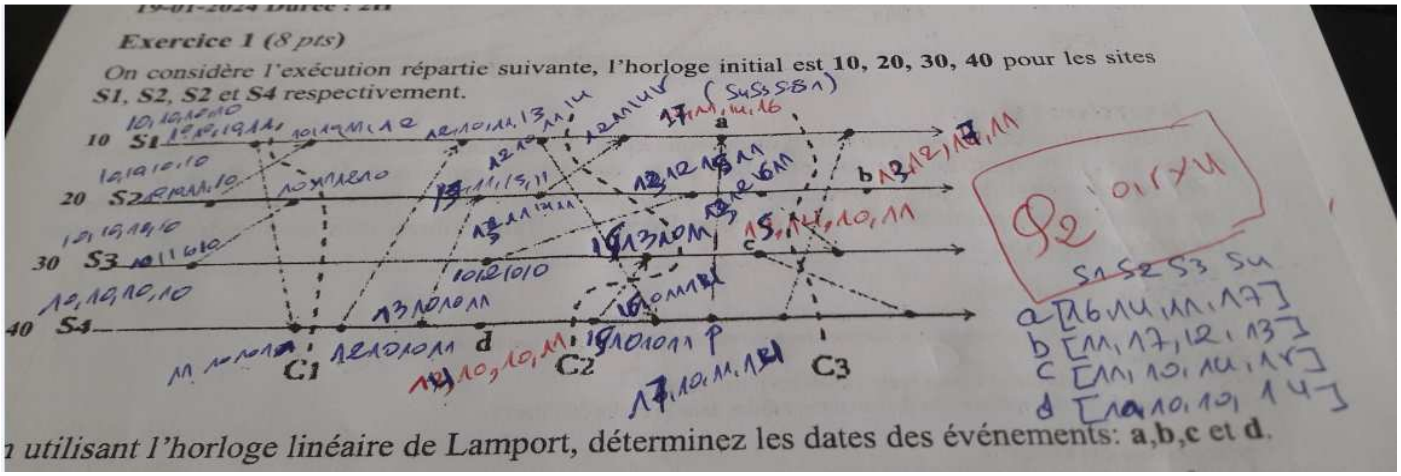
On considère l'exécution répartie suivante, l'horloge initial de chaque est **10, 20, 30, 40** pour les sites **S1**, **S2**, **S3** et **S4** respectivement.



Q1.



Q2.



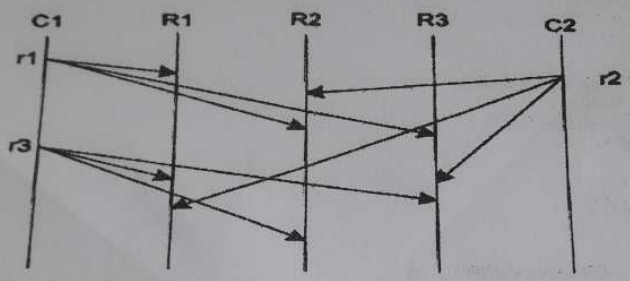
Q3+Q4.

1. En utilisant l'horloge linéaire de Lamport, déterminez les dates des événements: **a**,**b**,**c** et **d**.
2. En utilisant les horloges vectorielles déterminez les dates des événements **a**,**b**,**c** et **d**, avec l'horloge initiale de chaque site égale à : [10,10,10,10].
3. Donner la relation de précédence (||,→) entre les événements **a**,**b**,**c** et **d**.
4. Donner le type des coupures **C1**, **C2**, **C3**. Justifier votre réponse.

***Exercice 2** (5 pts)*

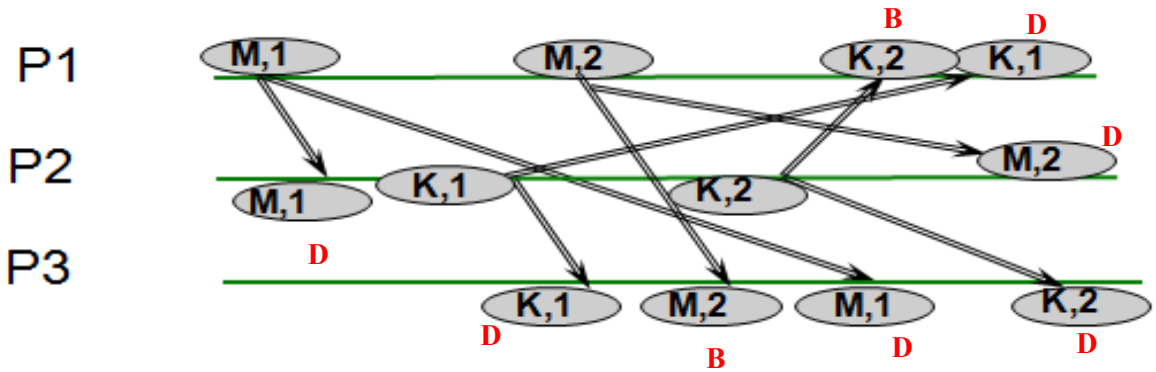
Exercice 2 (*5 pts*)

On considère un système distribué constitué de 3 *Réplicas* **R1**, **R2**, **R3** et deux clients **c1** et **c2**. **r1**, **r2** et **r3** représentent des requêtes de mise à jour.



1. *Donner l'ordre d'exécution des requêtes r1, r2 et r3 dans chaque réplica.*
2. *Expliquer le problème soulevez dans la question 1.*
3. *Décrire une solution au problème de la question 2.*

Exercice 3 (8*0,5 *pts*)



Légende : D : délivré. B : Hold-Back Queue

Exercice 4 (3 *pts*) (0,25+0,25+0,5+0,5+0,5+0,5+0,5)

Sur le serveur :
MG[ref] : la mémoire globale ;
version[ref] : les numéros de version courante de chaque mot référencé ;
version_cache[c,ref] : les numéros de version des références en cache chez chaque client.
Pour un client c :
cache**c**[ref] : cache du client c pour toute variable référencée ref ;
valide**c** : l'ensemble des référencées de variables valides dans le cache du client c.
valide [ref,val] ou tout simplement valide : l'ensemble des référencées de variables valides avec leurs valeurs.

Lecture sur un site Client

```
value read(ref x) {
if (x ∉ validec)
valide,v = Serveur.read(x,c) ;
validec = (validec ∪ valide[ref])+{x} ; cachec[x] = v ; for each ref { cachec[ref]= valide[val]}
}
return cachec [x]
}
```

Ecriture sur un site Client c

```
write(ref x, value v) {
valide = Serveur .write(x,v,c) ;
validec = (validec ∪ valide)+{x} ; cachec[x] = v ; for each ref { cachec[ref]= valide[val]}
}
```

Lecture sur le serveur : appel d'un client c

```
<set<ref,val>, v>read(ref x, client c) {
cache_version[c,x]=version[x];
return <valider(c,x),MG[x] >
}
```

Ecriture sur le site Serveur : appel d'un client c

```
set<ref,val> write(ref x, value v, client c) {
MG[x]=v; version[x]++;
cache_version[c,x]=version[x];
return valider(c,x)
}
```

Algorithme d'invalidation

```
set<ref,val> valider(client c, ref x) {
set<ref> invalide = {} ;
for (ref y ∈ MG && y ≠ x) {
if (cache_version[c,y] ≠ 0
&& cache_version[c,y]<version[y]){
   valide = valide + {y, MG [y]} ;
 }
}
return valide ;
}
```