Exercice n°1 :

P1	Send(Ma)=1	2	Send(Mb)=3		
P2	2	3	Send(Mc)=4		
P3	1	5	6	7	8
P4	1	Send(Md)=2			

P1	Send(Ma)=1,0,0,0	2,0,0,0	Send(Mb)=3,0,0,0		
P2	0,1,1,0	2,2,1,0	Send(Mc)=2,3,1,0		
P3	0,0,1,0	2,3,2,0	2,3,3,0	3,3,4,0	3,3,5,2
P4	0,0,0,1	Send(Md)=0,0,0,2			

Partie A : (2 +1.5 point)

1. Dater les événements par les horloges scalaires (Voir tableau),
 Etablir le passé de l'évènement Send(Mc) par :
 - a. La causalité : Passé (Send(Mc))={e11,e12,e21,e22,e31},
 - b. Les horloges vectorielles : HV(Send(Mc))= 2,3,1,0 → Passé(Send(Mc))={e11,e12,e21,e22,e31},
2. Vérifier s'il y a un non-respect du protocole de délivrance causale dans les deux cas suivants :
 - a. La causalité : oui il y a un non-respect car Send(Ma)→Send(Mc) mais Receive(Mc)→Receive(Ma),
 - b. Les horloges vectorielles de Mattern: Elles ne peuvent pas détecter s'il y a un non-respect du protocole de délivrance causale.

Partie B : (1.5 +1.5+1.5 point)

1. Les deux formules que vous avez fournies décrivent deux types de protocoles de délivrance des messages : l'ordre de délivrance FIFO et l'ordre de délivrance causale. Comparons ces deux formules pour déduire la différence entre les deux protocoles :

- **Ordre de délivrance FIFO** : $send_i(m_1, j) \rightarrow send_i(m_2, j) \Rightarrow del_j(m_1) \rightarrow del_j(m_2)$
- **Ordre de délivrance causale** : $send_i(m_1, j) \rightarrow send_k(m_2, j) \Rightarrow del_j(m_1) \rightarrow del_j(m_2)$

Différences :

- **Ordre de délivrance FIFO :**
 - La première formule indique que si un site i envoie le message m_1 , puis le message m_2 au même site j, alors la délivrance de m_1 chez j doit précéder la délivrance de m_2 chez j. Cela garantit un ordre de délivrance strictement basé sur l'ordre d'envoi.
- **Ordre de délivrance causale :**
 - La deuxième formule indique que si un site i envoie le message m_1 à un site j, et plus tard, un site k envoie le message m_2 au même site j, alors la délivrance de m_1 chez j doit précéder la délivrance de m_2 chez j. Cela reflète une relation causale entre les messages, où l'envoi du message m_2 dépend de la réception du message m_1 .

La principale différence réside dans la nature de l'ordonnement des messages. L'ordre de délivrance FIFO garantit un ordre strict basé sur l'envoi, tandis que l'ordre de délivrance causale prend en compte la relation causale entre les messages, où la délivrance d'un message peut dépendre de la réception d'un autre message.

2. A) Non. Les événements M_A, M_B, M_C ne respectent pas l'ordre de délivrance causale, car :
 - $send_1(M_A,3) \rightarrow send_1(M_B,3) \Rightarrow M_A \rightarrow M_B$
 - $send_1(M_A,3) \rightarrow send_2(M_C,3) \Rightarrow \text{Non } (M_A \rightarrow M_C)$ Faux
 - $send_1(M_B,3) // send_1(M_C,3) \Rightarrow$ On ne peut rien conclure.
- B) Non. Voir les valeurs des horloges de Matternn correspondants aux événements ci dessus.
 - $(1,0,0,0) < (3,0,0,0) \Rightarrow (1,3,3,0) < (1,2,3,4)$
 - $(1,0,0,0) < (2,3,1,0) \Rightarrow (1,3,3,0)$ n'est pas $< (2,3,2,0)$ Faux
 - $(3,0,0,0)$ non $< (2,3,1,0) \Rightarrow$ On ne peut rien conclure.

C) Oui. Car $EM_k[i,j]$ donne le nombre de messages en provenance du site S_i délivrables sur le site S_j dont le site S_k a connaissance (i.e dont l'envoi est causalement antérieur à tout ce qui arrivera dorénavant sur S_k).

3. Sur le site 3,
 - o $HM_3[3,1]$ correspond au nombre de messages issus du site 3 vers le site 1 (Ici le site 3 a émis 1 message vers le site 1).
 - o Le site 3 sait que :
 - i. $HM_3[1,3]$ correspond au nombre de messages issus du site 1 vers le site 3 (Ici le site 1 a émis 2 messages vers le site 3).
 - ii. $HM_3[2,3]$ correspond au nombre de messages issus du site 2 vers le site 3 (Ici le site 1 a émis 1 messages vers le site 2)
 - o $EM_3[1,2]$ donne le nombre de messages en provenance du site S_1 délivrables sur le site S_2 dont le site S_3 a connaissance (i.e dont l'envoi est causalement antérieur à tout ce qui arrivera dorénavant sur S_3).

Exercice n°2 :

1. Types des messages échangés : (1.5 +1 point)

- a. Message d'élection (Election): Envoyé par un site initié pour annoncer une demande d'élection.
- b. Message d'élu (Elected): Envoyé par un site élu pour informer les autres sites de son élection.
- c. Accusé de réception (ACK) : Envoyé par un site pour confirmer la réception d'un message.

L'instruction (*) dans la procédure Lancement_élection sert à armer un délai de garde T. Cela signifie qu'après avoir lancé une demande d'élection, le site S_i attend pendant un certain temps (délai T) avant de prendre des décisions basées sur les réponses reçues.

2. Comportement de S_i lorsqu'il envoie un message (S_i , Election) : (1 point)

Lorsqu'un site S_i envoie un message d'élection à un site S_j :

- a. S_i inclut une attente pour un ACK après avoir envoyé le message.
- b. Le délai de garde T est utilisé après l'envoi des messages d'élection pour attendre les réponses des autres sites.

Le délai de garde T peut influencer le comportement dans le sens où S_i attend pendant ce délai pour recevoir les ACK des autres sites.

3. Procédures : (1 +1.5 point)

a. Réception_msg(S_j , Election) par S_i :

Réception_msg(S_j , Election) par S_i

si (non Demande_Elect_i) alors Demande_Elect_i ← vrai;

envoyer(S_i , ack);

// S_i envoie ack à S_j car $i > j$ //

Pour tout $k > i$ faire envoyer(S_i , Election);

// S_i lance l'élection

Armer délai de garde T

b. Déclenchement_délai_T(S_i) :

Déclenchement_délai_T(S_i)

S_i (aucun msg ack reçu) alors Elu ← vrai;

Pour tout j faire (elected, S_j)

// p_i est leader

sinon armer délai garde T1 //attendre la fin de l'élection et T1 estime le temps max de l'élection

Déclenchement_délai_T1(S_i)

S_i (aucun msg elected reçu) alors Lancement_élection par S_i ;

4. Complexité de l'algorithme de BULLY au pire des cas : (1 point)

$S_1 \rightarrow n-1$ msg

$S_2 \rightarrow n-2$ msg

$S_3 \rightarrow n-3$ msg ...

$S_{n-1} \rightarrow 1$ msg

$\Rightarrow n(n-1)/2$ msg

De plus, le leader doit envoyer un msg de notification à tous les sites → (n-1) messages supplémentaires. D'où une complexité de $O(n^2)$

5. **Comparaison avec l'algorithme de Chang et Robert : (1 point)**

- a. Les deux algorithmes sont utilisés pour élire un coordinateur dans un système distribué, mais leurs approches diffèrent. L'algorithme de BULLY utilise une approche hiérarchique basée sur les identificateurs, tandis que l'algorithme de Chang et Robert utilise une approche basée sur un arbre.
- b. La complexité au pire des cas dépend du nombre de sites dans les deux cas, mais les détails spécifiques peuvent varier en fonction de la topologie du réseau et de l'algorithme particulier. Au pire des cas, les deux algorithmes ont la même complexité $O(n^2)$.

Questions : (2 points +2 points)

1. On distingue les types d'interblocage en fonction de l'allocation de ressources et des échanges de messages.

a. Interblocage basé sur l'allocation de ressources :

Interblocage mutuel : Un groupe de processus est bloqué parce que chaque processus attend une ressource détenue par un autre processus du même groupe. Exemple : Processus A détient R1 et attend R2, Processus B détient R2 et attend R1.

Interblocage cyclique : Un cycle existe dans le graphe des ressources attendues par les processus, chaque processus attendant une ressource détenue par le processus suivant dans le cycle. Processus A attend R1 de B, B attend R2 de C, et C attend R1 de A.

Interblocage à plusieurs niveaux (ou hiérarchique) : Exemple : Processus A détient R1 et attend R2, tandis que Processus B détient R2 et attend R3.

b. Interblocage basé sur les échanges de messages :

Interblocage de communication : Des processus sont bloqués parce qu'ils attendent des messages qui ne seront jamais envoyés. Exemple : Processus A attend un message de B, B attend un message de C, et C attend un message de A.

Interblocage de synchronisation : Des processus attendent des événements de synchronisation qui ne se produiront jamais. Exemple : Processus A attend un signal de terminaison de B, et B attend un signal de terminaison de A.

Interblocage de coordination : Des processus sont bloqués car ils attendent des actions coordonnées qui ne se produiront jamais. Exemple : Processus A attend que B émette un signal, et B attend que A émette un signal.

Ces catégories d'interblocage peuvent se produire simultanément ou de manière isolée dans un système réparti, et les mécanismes de détection et de prévention varient en fonction du type d'interblocage considéré.

2. L'algorithme de détection de faux interblocages est utilisé pour éviter de prendre des mesures incorrectes lors de la détection d'un interblocage. Le phénomène de détection de faux interblocages peut se produire lorsque le système signale à tort la présence d'un interblocage.

Exemple d'explication :

Considérons un scénario où un algorithme centralisé de détection d'interblocage collecte des informations sur l'utilisation des ressources auprès de tous les nœuds du système réparti. Si un nœud X signale qu'il est en attente d'une ressource détenue par le nœud Y, et en même temps, le nœud Y signale qu'il a déjà attribué cette ressource à X, alors il y a une incohérence dans les informations collectées.

Étape 1 : Le nœud X signale au gestionnaire central qu'il attend R1 de Y.

Étape 2 : Le nœud Y signale au gestionnaire central qu'il a attribué R1 à X.

Si le gestionnaire central ne parvient pas à mettre à jour rapidement ses informations, il peut signaler un interblocage alors qu'en réalité, il n'y en a pas. C'est un faux positif.