

CSE423 (Lập trình phân tán): Đề thi cuối kỳ (GD2, 2021-2022)

Ngày thi: 19/01/2022

LƯU Ý:

Sinh viên ĐƯỢC tham khảo tài liệu, nhưng **KHÔNG ĐƯỢC** trao đổi bài, gửi bài cho sinh viên khác hoặc làm hộ sinh viên khác trong khi làm bài thi. Nếu bị phát hiện vi phạm (trong hoặc sau ca thi), sinh viên sẽ bị xử lý theo quy định thi trực tuyến của Trường !

Một câu hỏi có thể có nhiều đáp án đúng. Sinh viên phải chọn tất cả các đáp án đúng mới được điểm của câu đó.

Điểm của bài làm sẽ được tính theo kết quả của lần gửi đầu tiên.

Điểm thi sẽ được công bố sau khi hết ca thi. Các thắc mắc sẽ được giải đáp sau khi hết ca thi.

1

Họ và tên *

Lưu Đức Khánh

2

Mã sinh viên *

1851061574

3

Lớp *

60TH5

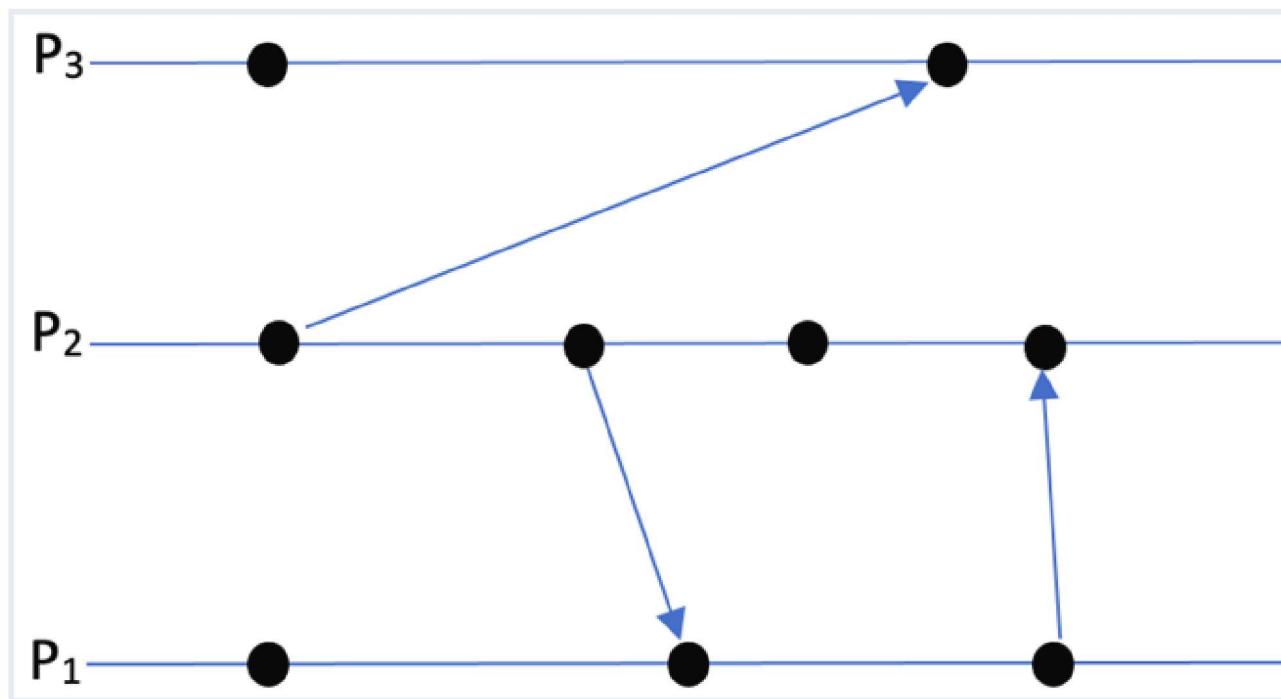
4

Cho sơ đồ tiến trình - thời gian, hoặc sơ đồ đã-xảy-ra-trước, như hình vẽ dưới đây.

Sử dụng **cơ chế đồng hồ phụ-thuộc-trực-tiếp** để đánh dấu thời gian của các trạng thái cho các tiến trình.

Dấu thời gian của trạng thái cuối cùng, sau khi sự kiện cuối cùng xảy ra, của **tiến trình P3** là gì? *

(2 Điểm)



(3,6,0)

(3,5,0)

(4,2,0) (0,5,6)

5

Sự khác nhau giữa Monitor kiểu Hoare và Monitor kiểu Mesa là gì ?

Monitor kiểu Hoare còn được gọi là "*Signal-and-Urgent-Wait Monitor*"

Monitor kiểu Mesa còn được gọi là "*Signal-and-Continue Monitor*"

*

(1 Điểm)

- Trong Monitor kiểu Hoare, khi một luồng T_i đang trong monitor và gọi **notify()** để đánh thức luồng T_j khác, luồng T_i sẽ KHÔNG bị mất quyền chiếm giữ monitor ngay lập tức, mà nó vẫn tiếp tục công việc cho đến khi hoàn thành và ra khỏi monitor thì luồng T_j lúc này mới có thể chiếm giữ monitor

- Trong Monitor kiểu Mesa, không cần thiết phải có **hàng đợi riêng chỉ để chứa các luồng được đánh thức.** Thay vào đó, các luồng được đánh thức sẽ được chuyển sang hàng đợi dành cho các luồng mới đi vào monitor

- Trong Monitor kiểu Mesa, khi một luồng T_i đang trong monitor và gọi **notify()** để đánh thức luồng T_j khác, luồng T_i sẽ **ngay lập tức** bị tạm dừng và mất quyền chiếm giữ monitor

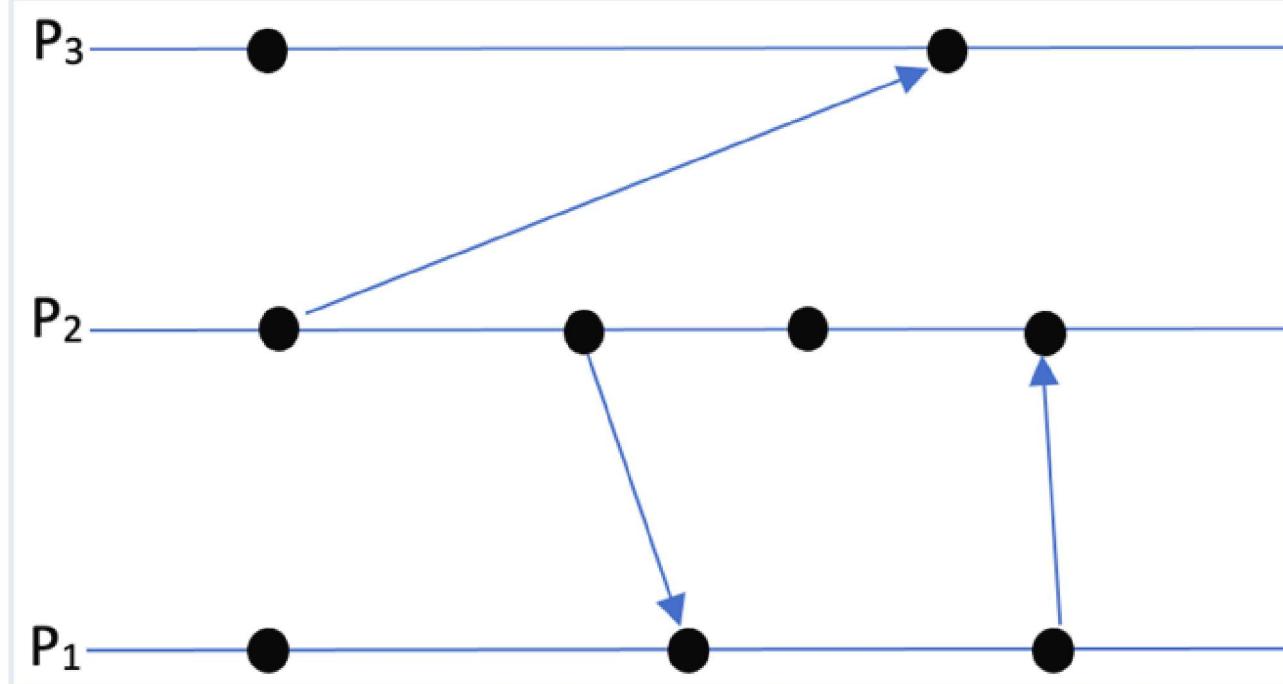
6

Cho sơ đồ tiến trình - thời gian, hoặc sơ đồ đã-xảy-ra-trước, như hình dưới đây.

Sử dụng **cơ chế đồng hồ logic** để đánh dấu thời gian của các trạng thái cho các tiến trình.

Dấu thời gian của trạng thái cuối cùng, sau khi sự kiện cuối cùng xảy ra, của **tiến trình P3** là gì? *

(2 Điểm)



5

6

7

8

7

Nhược điểm chung của các thuật toán Peterson, Dekker, Bakery khi giải quyết bài toán loại trừ lẫn nhau trong các chương trình đồng thời là gì? *(1 Điểm)

Không có nhược điểm gì

Sử dụng các biến chia sẻ, dẫn đến có thể mất mát dữ liệu

Các luồng phải liên tục kiểm tra xem điều kiện đi vào khu vực quan trọng đã được thoả mãn hay chưa, thông qua vòng lặp. Điều này dẫn đến gây lãng phí chu trình CPU

8

Những phát biểu nào sau đây là đúng về khái niệm MOM (Message-Oriented Middleware) ? *

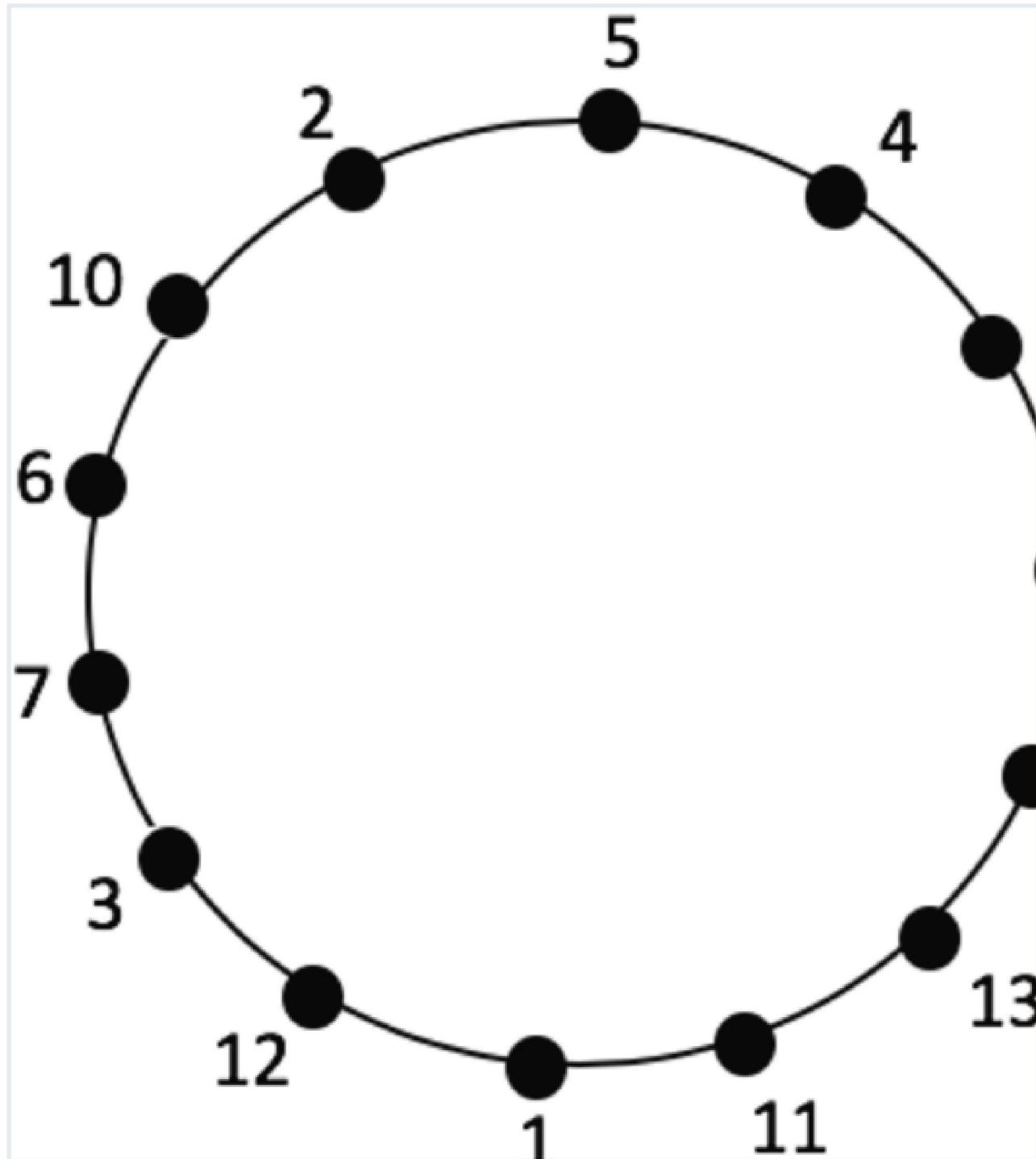
(1 Điểm)

- Các hệ thống phân tán được xây dựng dựa trên MOM cho phép việc truyền thông được thực hiện thông qua trao đổi **bất đồng bộ** các thông điệp, tức là tiến trình gửi yêu cầu sẽ không bị chặn mà có thể tiếp tục công việc của nó sau khi gửi thông điệp
- Trong các hệ thống phân tán được xây dựng dựa trên MOM, luôn có một thành phần trung gian, được gọi là Messaging Server, để điều phối quá trình gửi và nhận các thông điệp
- Để thực hiện được quá trình truyền thông điệp, hai tiến trình gửi và nhận bắt buộc phải cùng thực thi tại thời điểm gửi và nhận

9

Áp dụng thuật toán bầu cử của Hirschberg-Sinclair cho sơ đồ dưới đây, tại vòng $r = 1$ những tiến trình nào được bầu là ứng viên cho vị trí lãnh đạo ? *

(1 Điểm)

 11 13 14 10 13 14 10 12 14 12 13 14

10

Cho đoạn mã giả của một chương trình đồng thời với hai luồng t , u như Hình dưới đây.

Giả sử các câu lệnh được thực thi một cách nguyên tử.

Sau khi hai luồng t , u thực thi xong các câu lệnh của mình, biến chia sẻ **counter** có thể nhận những giá trị nào sau đây ?

*

(1 Điểm)

int counter = 0;	
Luồng t	Luồng u
int cnt; cnt = counter; counter = cnt + 1;	int cnt; cnt = counter; counter = cnt +

2

0

1

3

11

Những phát biểu nào sau đây là đúng về kỹ thuật RMI (Remote Method Invocations) trong Java? *

(1 Điểm)

- Trong kiến trúc của RMI, ngoài hai thành phần tiến trình khách (client) và tiến trình chủ (server)
- chứa đối tượng từ xa, còn có thêm thành phần thứ ba: RMIRegistry

- Mỗi lời gọi từ xa luôn được thực hiện thông qua hai đối tượng đại diện: *stub* ở phía client và *skeleton* ở phía server

- Trong RMI, tiến trình khách không cần biết đến vị trí thực sự của đối tượng từ xa, i.e, không cần
biết địa chỉ IP và cổng của tiến trình cung cấp dịch vụ

- Hai đối tượng *stub* và *skeleton* KHÔNG bắt buộc phải được tạo ra khi thực hiện một lời gọi từ
xa

12

Semaphore đếm khác Semaphore nhị phân như thế nào ? *

(1 Điểm)

- Không có hàng đợi các luồng bị khóa trong Semaphore đếm

- Không có hai thao tác P(), V() trong Semaphore đếm

- Biến *value* của Semaphore đếm có thể nhận nhiều giá trị lớn hơn 0, trong khi biến *value* của
Semaphore nhị phân chỉ có thể nhận 2 giá trị

- Semaphore đếm có thể được dùng để cho phép nhiều luồng đồng thời cùng ở trong khu vực
quan trọng (CS), trong khi Semaphore nhị phân chỉ cho phép tối đa 1 luồng ở trong CS tạ
một thời điểm

13

Những phát biểu nào sau đây là đúng về khái niệm Socket dùng để phát triển các ứng dụng phân tán ? *

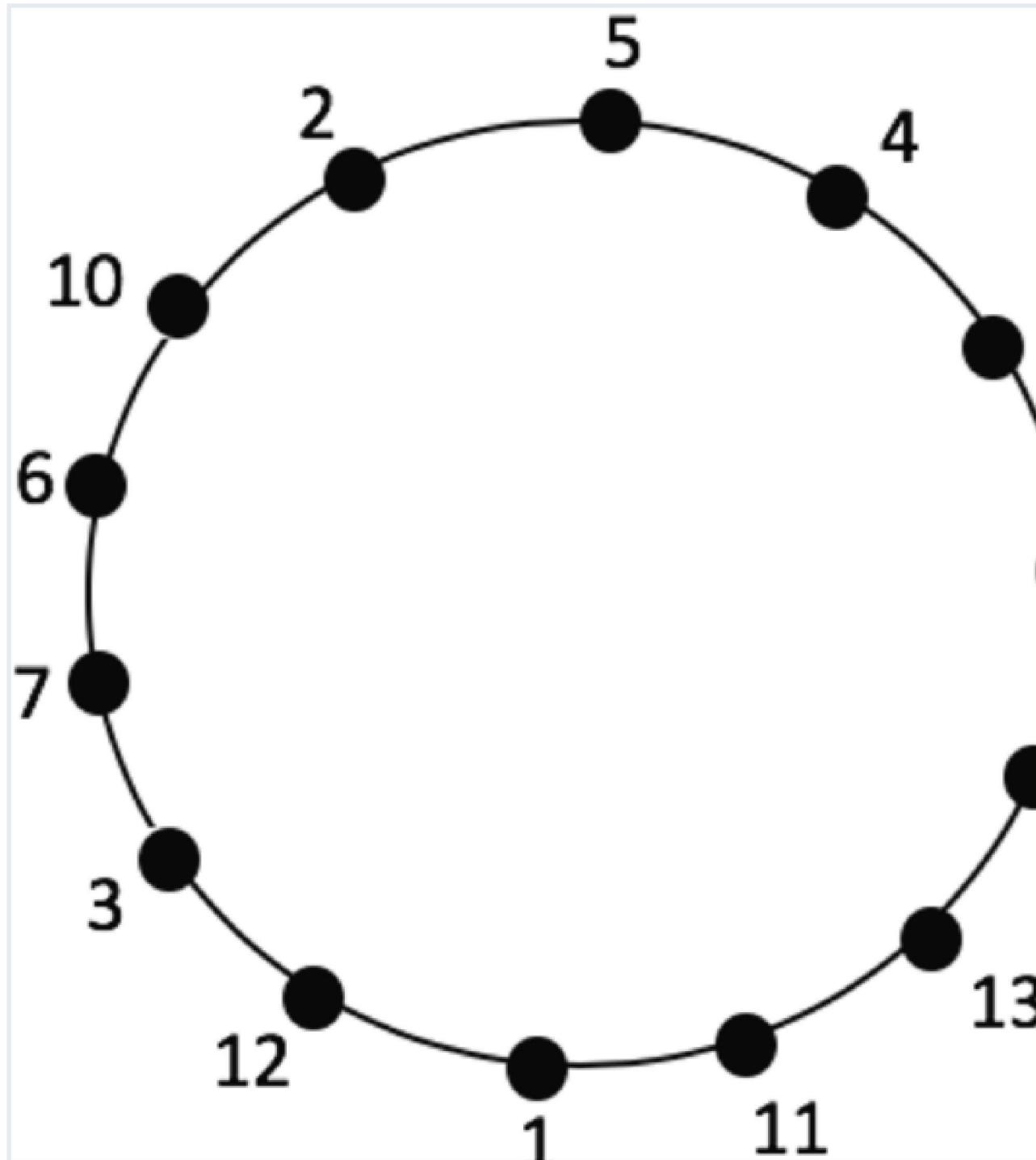
(1 Điểm)

- Socket gồm có 2 thành phần: *địa chỉ IP* và *cổng* mà tiến trình đang thực thi
- Socket cung cấp một giao diện ở mức cao, hướng đối tượng cho việc xây dựng các chương trình phân tán
- Lập trình Socket CHỈ có thể được thực hiện dựa trên giao thức UDP (Universal Datagram Protocol)

14

Áp dụng thuật toán bầu cử của Hirschberg-Sinclair cho sơ đồ dưới đây, tại vòng $r = 0$ những tiến trình nào được bầu là ứng viên cho vị trí lãnh đạo ? *

(1 Điểm)



9 10 11 12 13 14

5 10 11 12 13 14

5 7 10 12 13 14

8 9 11 12 13 14

15

Nhược điểm của việc sử dụng các kỹ thuật Socket hay RMI để phát triển các ứng dụng phân tán là gì? *

(1 Điểm)

Thông thường khi sử dụng các kỹ thuật này, tiến trình gửi yêu cầu sẽ bị chặn thực thi cho đến khi nhận được kết quả trả về

Để thực hiện được quá trình truyền thông điệp với các kỹ thuật này, hai tiến trình gửi và nhận KHÔNG bắt buộc phải cùng thực thi tại thời điểm gửi và nhận đó

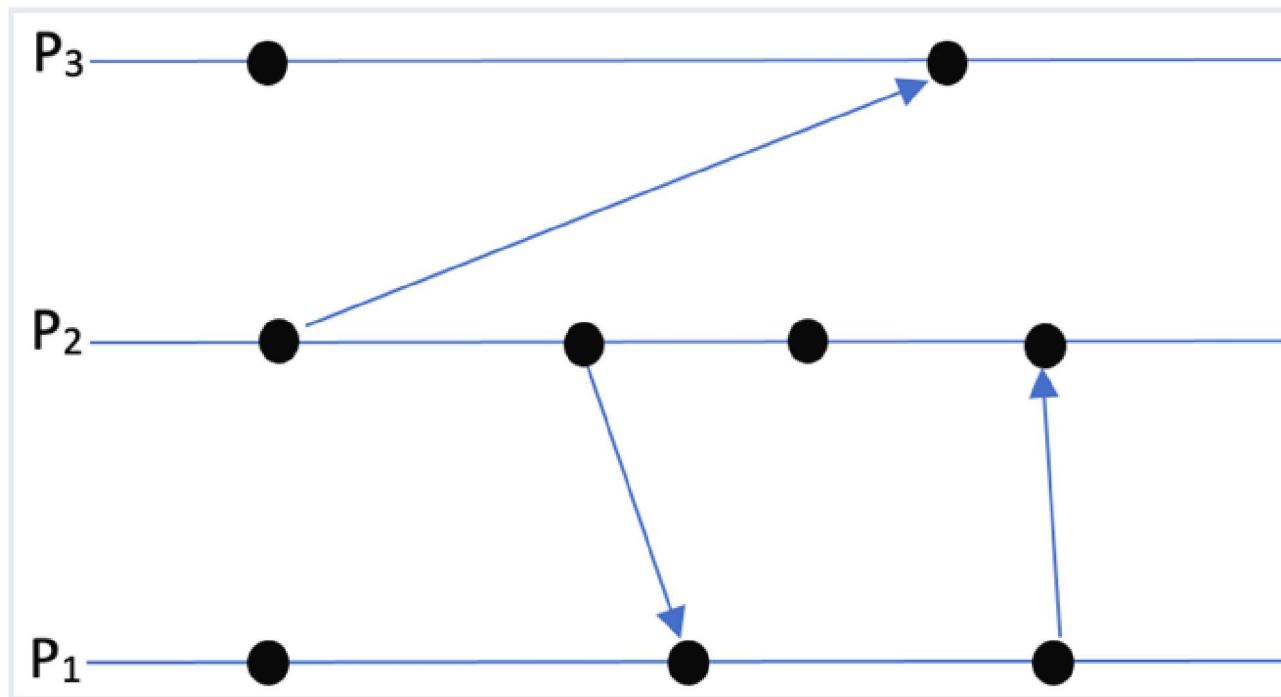
16

Cho sơ đồ tiến trình - thời gian, hoặc sơ đồ đã-xảy-ra-trước, như hình vẽ dưới đây.

Sử dụng **cơ chế đồng hồ vector** để đánh dấu thời gian của các trạng thái cho các tiến trình.

Dấu thời gian của trạng thái cuối cùng, sau khi sự kiện cuối cùng xảy ra, của **tiến trình P3** là gì? *

(2 Điểm)

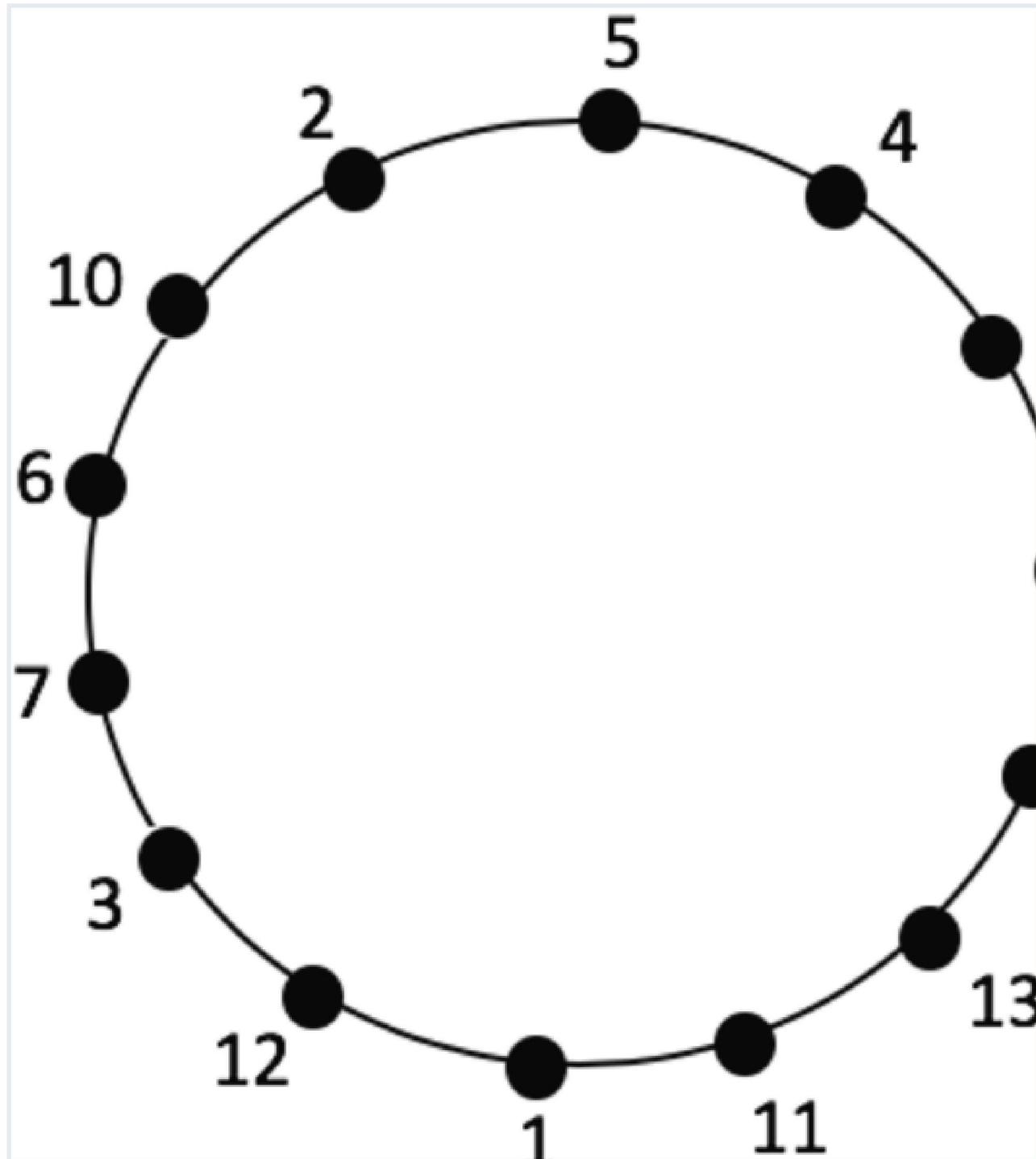


(3,5,0) (0,1,3) (4,2,0) (3,5,4)

17

Áp dụng thuật toán bầu cử người lãnh đạo của Hirschberg-Sinclair cho sơ đồ dưới đây, tại vòng $r = 2$ những tiến trình nào được bầu là ứng viên cho vị trí lãnh đạo ? *

(1 Điểm)

 14 10 14 12 14 13 14

18

Khi chạy chương trình sau có thể sinh ra những kết quả nào ? *
(2 Điểm)

```
public class Lab1 extends Thread{
    private int id;
    public Lab1(int _id) {
        id = _id;
    }
    public void run() {
        System.out.print("T-" + id + " ");
    }

    public static void main(String[] args) throws InterruptedException {
        Lab1 t1 = new Lab1(1);
        Lab1 t2 = new Lab1(2);

        t1.start();
        t2.start();

        t2.join();

        System.out.print("T-m ");
    }
}
```

- T-1 T-m T-2
- T-m T-2 T-1
- T-m T-1 T-2
- T-2 T-1 T-m
- T-2 T-m T-1
- T-1 T-2 T-m

19

Đâu là những đặc điểm chung của các thuật toán sử dụng dấu thời gian (như **Thuật toán của Lamport, Thuật toán của Ricart & Agrawala**) cho bài toán truy cập tài nguyên chia sẻ (i.e., khu vực quan trọng) trong một hệ thống phân tán ?

*

(1 Điểm)

- Tất cả các thuật toán này đều sử dụng 3 kiểu thông điệp cho một lần đi vào khu vực quan trọng: thông điệp *request*, thông điệp *ack*, thông điệp *release*
- Các thuật toán này đảm bảo được yêu cầu về *tính công bằng* do sử dụng dấu thời gian của các thông điệp để quyết định thứ tự đi vào khu vực quan trọng
- Khi muốn đi vào khu vực quan trọng, một tiến trình đầu tiên phải gửi thông điệp yêu cầu (thông điệp *request*), có gắn dấu thời gian, tới **tất cả tiến trình khác**

20

Khi chạy chương trình sau có thể sinh ra những kết quả nào ? *

(2 Điểm)

```

public class Lab1 extends Thread{
    private int id;
    public Lab1(int _id) {
        id = _id;
    }
    public void run() {
        System.out.print("T-" + id + " ");
    }

    public static void main(String[] args) throws InterruptedException {
        Lab1 t1 = new Lab1(1);
        Lab1 t2 = new Lab1(2);

        t1.start();
        t2.start();

        t1.join();
        t2.join();

        System.out.print("T-m ");
    }
}

```

- T-m T-1 T-2
- T-2 T-1 T-m
- T-1 T-2 T-m
- T-m T-2 T-1
- T-1 T-m T-2
- T-2 T-m T-1

21

Ưu điểm của việc sử dụng các cấu trúc đồng bộ hoá (như Semaphore, Monitor) cho bài toán loại trừ lẫn nhau trong các chương trình đồng thời, so với các thuật toán sử dụng vòng lặp để kiểm tra điều kiện đi vào khu vực quan trọng như Peterson, Dekker, Bakery, là gì ? *

(1 Điểm)

- Giải quyết được vấn đề bận chờ (busy-waiting), không gây lãng phí chu trình CPU

Không có ưu điểm gì hơn so với các thuật toán đó

Thời gian chạy chương trình nhanh hơn

22

Xét thuật toán trong Hình dưới đây để giải quyết bài toán loại trừ lẫn nhau trong một chương trình đồng thời có 2 luồng cùng thực thi.

Những kịch bản thực thi nào sau đây của 4 câu lệnh (được đánh số màu đỏ), trong 2 phương thức **requestCS()**, sẽ khiến cho thuật toán này vi phạm điều kiện loại trừ lẫn nhau, tức là cho phép cả 2 luồng T0 và T1 đều cùng đi vào khu vực quan trọng? *

(1 Điểm)

T₀

T₁

```
class Attempt1 implements Lock{    class Attempt1 impleme
    boolean openDoor = true;        boolean openDoor = t
    public void requestCS(int i) {    public void requestC.
        1      while (!openDoor);        while (!openDoor);
        2      openDoor = false;        openDoor = false;
    }                                }
    public void releaseCS(int i) {    public void releaseCS
        openDoor = true;            openDoor = true;
    }                                }
}
```

3->1->4->2

1->3->4->2

3->4->1->2

1->3->2->4

1->2->3->4

3->1->2->4

23

Các tiến trình (Process) trong một hệ thống phân tán, gồm nhiều máy tính đặt tại nhiều địa điểm khác nhau, sử dụng mô hình nào để giao tiếp với nhau? *(1 Điểm)

Mô hình bộ nhớ chia sẻ

Mô hình truyền thông điệp qua mạng truyền thông

KHÔNG sử dụng Mô hình bộ nhớ chia sẻ, cũng như Mô hình truyền thông điệp qua mạng truyền thông

24

Cho sơ đồ tiến trình - thời gian như hình dưới đây, trong đó chúng ta sử dụng đồng hồ logic để đánh dấu thời gian của các trạng thái.

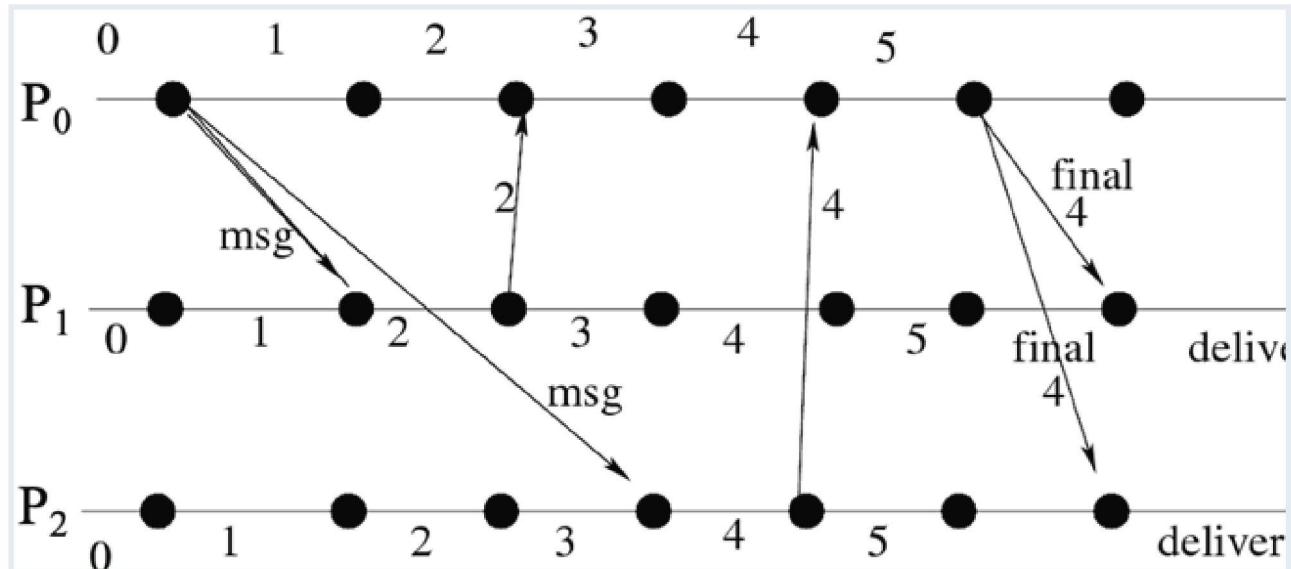
Giả sử tại một thời điểm nào đó, tiến trình P0 gửi một thông điệp đa hướng **msg** tới 2 tiến trình P1 và P2.

Sử dụng thuật toán Skeen để sắp thứ tự toàn bộ cho các thông điệp đa hướng trong hệ thống phân tán trên.

Dấu thời gian của thông điệp **msg** sau khi hoàn tất các bước của thuật toán Skeen là bao nhiêu?

*

(1 Điểm)



5

2

6

4

25

Những phát biểu nào sau đây là đúng về **mô hình đã-xảy-ra-trước** (happened-before model), được kí hiệu là \rightarrow , trong một hệ thống phân tán ? *(1 Điểm)

Trong mô hình đã-xảy-ra-trước, nếu tồn tại một sự kiện g sao cho $e \rightarrow g$ và $g \rightarrow f$, thì $e \rightarrow f$

Trong mô hình đã-xảy-ra-trước, chúng ta LUÔN có được **thứ tự tổng thể** trên tập các sự kiện đã xảy ra, tức là với hai sự kiện bất kỳ, chúng ta luôn biết chắc chắn sự kiện nào đã xảy ra trước

Trong mô hình đã-xảy-ra-trước, chúng ta CHỈ có được một **thứ tự bộ phận** trên tập các sự kiện đã xảy ra

Trong mô hình đã-xảy-ra-trước, nếu e là sự kiện gửi của một thông điệp và f là sự kiện nhận của cùng thông điệp đó, thì $e \rightarrow f$

26

Những phát biểu nào sau đây là đúng khi khái quát về các hệ thống phân tán? ***(1 Điểm)**

- Mỗi tiến trình trong hệ thống phân tán luôn CHỈ gồm có một luồng
- Các tiến trình trong một hệ thống phân tán giao tiếp với nhau bằng cách gửi và nhận thông điệp qua mạng truyền thông
- Các tiến trình trong một hệ thống phân tán có thể KHÔNG biết được trạng thái toàn cục của hệ thống tại bất kỳ thời điểm nào
- LUÔN tồn tại các biến chia sẻ giữa các tiến trình trong một hệ thống phân tán

27

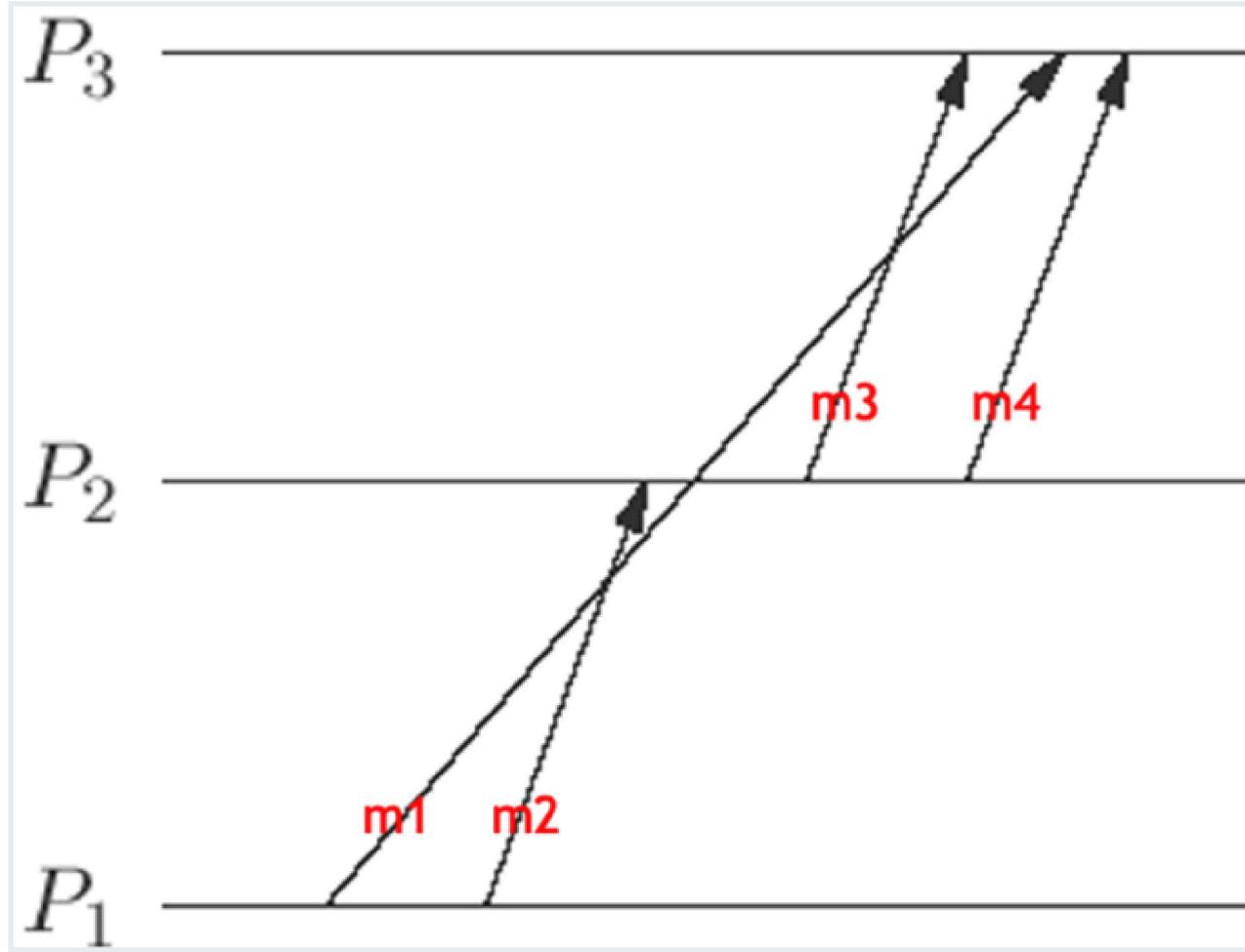
Cho sơ đồ tiến trình - thời gian như hình dưới.

Giả thiết P_i là một tiến trình trong một hệ thống phân tán và có 4 thông điệp (m1, m2, m3, m4) được truyền đi giữa 3 tiến trình.

Bốn thông điệp này thoả mãn kiểu thứ tự thông điệp nào sau đây?

*

(1 Điểm)



Thứ tự FIFO

Thứ tự đồng bộ

Thứ tự nhân quả

28

Các luồng (Thread) trong một chương trình đồng thời dựa trên cơ chế khoá, chạy trên một máy tính gồm nhiều bộ vi xử lý, sử dụng mô hình nào để giao tiếp với nhau? *

(1 Điểm)

Mô hình truyền thông điệp qua mạng truyền thông

Cả hai mô hình: Mô hình bộ nhớ chia sẻ và Mô hình truyền thông điệp qua mạng truyền thông

Mô hình bộ nhớ chia sẻ

29

Cho mã giả của chương trình đa luồng để giải quyết bài toán Sản xuất và Tiêu thụ, sử dụng monitor trong Java, như Hình dưới.

Câu lệnh gọi phương thức **notify()** của đối tượng **sharedBuffer** trong phương thức **fetch()**, được dùng với mục đích gì? *

(1 Điểm)

object sharedBuffer ;	
Producer	Consumer
<pre>void deposit() { synchronized (sharedBuffer) { while (bộ đệm đầy) sharedBuffer.wait(); <Thêm 1 phần tử vào bộ đệm> if (bộ đệm không rỗng) sharedBuffer.notify(); } }</pre>	<pre>void fetch() { synchronized (sharedBuff while (bộ đệm rỗng) sharedBuffer.wait(); <Lấy 1 phần tử khỏi b if (bộ đệm không đầy) sharedBuffer.notify() } }</pre>

- Đánh thức luồng Tiêu thụ (Consumer) để thực hiện tiếp công việc của nó
- Đánh thức cả 2 luồng Sản xuất (Producer) và Tiêu thụ (Consumer)
- Đánh thức luồng Sản xuất (Producer) để thực hiện tiếp công việc của nó

30

Xét thuật toán trong Hình dưới để giải quyết bài toán loại trừ lẫn nhau trong một chương trình đồng thời có 2 luồng cùng thực thi.

Những kịch bản thực thi nào của 4 câu lệnh (được đánh số màu đỏ), trong 2 phương thức **requestCS()**, sẽ khiến cho chương trình rơi vào **trạng thái khoá chết** (deadlock), tức là cả hai luồng đều bị tắc lại ở vòng lặp while, không luồng nào đi vào được khu vực quan trọng ? *

(1 Điểm)

T_0	T_1
<pre> class Attempt2 implements Lock { boolean wantCS[] = {false, false}; public void requestCS(int i) { 1 wantCS[i] = true; 2 while (wantCS[1 - i]) ; } public void releaseCS(int i) { wantCS[i] = false; } } </pre>	<pre> class Attempt2 implemer boolean wantCS[] = {fa public void requestCS(wantCS[i] = true; while (wantCS[1 - i] } public void releaseCS(wantCS[i] = false; } } </pre>

3->1->4->2

3->4->2->1

1->2->3->4

3->1->2->4

1->3->2->4

1->3->4->2

31

Khu vực quan trọng CS (Critical Region hay Critical Section) của một luồng, trong các chương trình đa luồng, là gì ? *(1 Điểm)

Phần dữ liệu chia sẻ giữa các luồng

Phần mã nguồn của luồng cần được thực thi một cách nguyên tử

Phần mã nguồn của luồng chia sẻ với một luồng khác

- Phần mã nguồn thực hiện một tính toán quan trọng của luồng

32

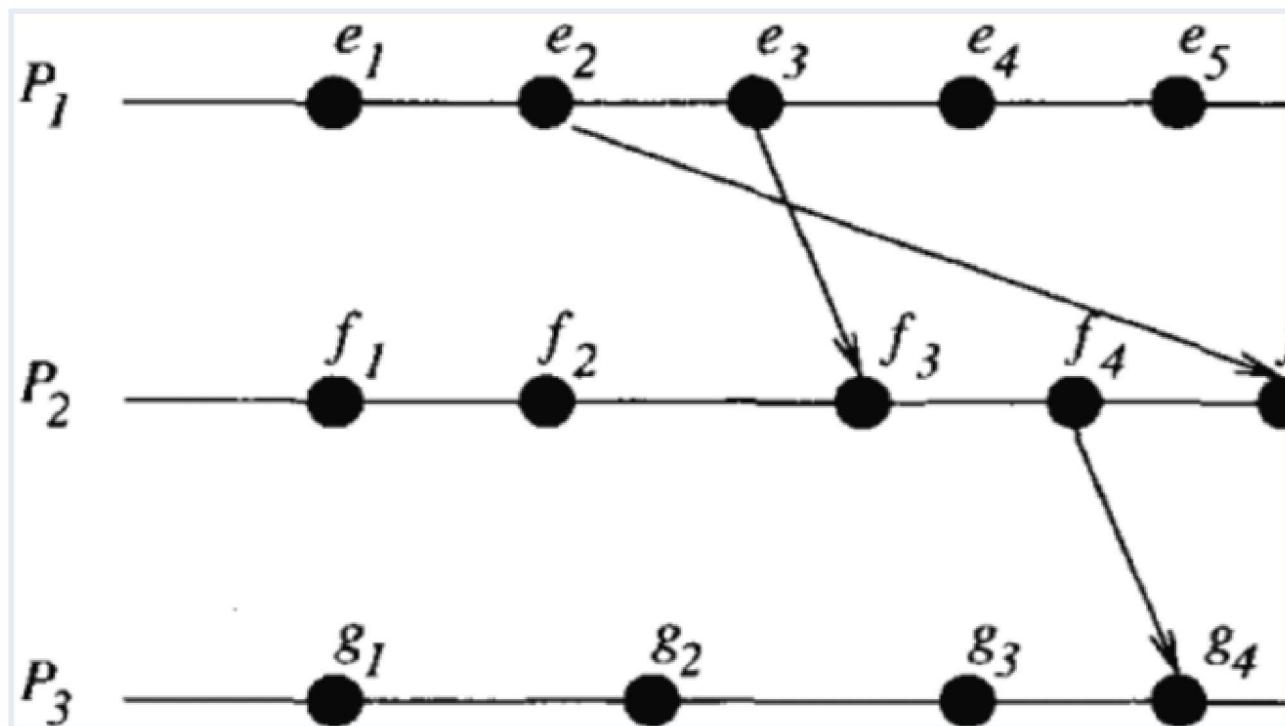
Giả sử:

- P_i là một tiến trình trong hệ thống phân tán
- e_i, f_i, g_i biểu thị các sự kiện xảy ra trên các tiến trình

Ký hiệu $e \rightarrow f$ biểu thị sự kiện e xảy ra trước sự kiện f trong **mô hình đã-xảy-ra-trước**.

Ký hiệu $e \parallel f$ biểu thị sự kiện e xảy ra "đồng thời" với sự kiện f , tức là không có đủ thông tin để kết luận sự kiện nào xảy ra trước

Những phát biểu nào là đúng với sơ đồ tiến trình - thời gian dưới đây? *
 (1 Điểm)



- $f_2 \rightarrow g_4$

- $e_2 \rightarrow f_2$

- $e_2 \rightarrow g_3$

- $f_3 \parallel g_3$

33

Đâu là những đặc điểm chung của các thuật toán sử dụng tài nguyên phụ token (như **Thuật toán Tập trung**, **Thuật toán Vòng tròn Token**) cho bài toán truy cập tài nguyên chia sẻ (i.e., khu vực quan trọng) trong một hệ thống phân tán ?

*

(1 Điểm)

- Tiến trình nào giữ token sẽ được quyền đi vào khu vực quan trọng, và tại một thời điểm chỉ có một tiến trình giữ token
- Các thuật toán này LUÔN đảm bảo được yêu cầu về *tính công bằng*, tức là tiến trình nào yêu cầu đi vào khu vực quan trọng trước thì sẽ được ưu tiên đi vào khu vực quan trọng trước
- Tất cả các thuật toán này đều sử dụng 1 kiểu thông điệp cho một lần đi vào khu vực quan trọng: thông điệp *token*

Nội dung này được tạo bởi chủ sở hữu của biểu mẫu. Dữ liệu bạn gửi sẽ được gửi đến chủ sở hữu biểu mẫu. Microsoft không chịu trách nhiệm về quyền riêng tư hoặc thực tiễn bảo mật của khách hàng, bao gồm cả các biện pháp bảo mật của chủ sở hữu biểu mẫu này. Không bao giờ đưa ra mật khẩu của bạn.

Hoạt động trên nền tảng Microsoft Forms | [Quyền riêng tư và cookie](#) | [Điều khoản sử dụng](#).