CHƯƠNG 6: Đồng bộ hóa

Họ tên sinh viên: Phan Thành Đạt **MSSV:20173001**

Mã Lớp: 118636 **Mã học phần: IT4611**

Câu hỏi 1:

Giả sử hệ thống phân tán có tài nguyên R dùng chung và trong 1 thời điểm chỉ có một tiến trình duy nhất sử dụng. Nếu ta chia thời gian dùng tài nguyên R trong 1 khoảng thời gian cho từng tiến trình và nếu như mỗi tiến trình không đồng bộ đồng hồ vật lý thì sẽ dẫn đến hiện tượng 1 thời điểm có tiến trình cố gắng cùng truy cập tài nguyên.

Câu hỏi 2:

Vì theo giáo sư Lamport, không nhất thiết phải đồng bộ hóa đồng hồ vật lý về giống nhau rồi các tiến trình dựa vào đó mà đồng bộ hóa, chỉ tập trung vào thứ tự sự kiện, cần đảm bảo thứ tự sự kiện trước sau.Vì vậy, ông đề ra mối quan hệ xảy ra trước - The "happens-before" relatio.

Câu hỏi 3:

Việc triển khai các giải thuật bình thường không phù hợp với mạng không dây wireless network bởi vì các mạng cảm ứng có sự ràng buộc về năng lượng của mỗi nút mạng và việc gửi các thông tin định tuyến broadcast trong mạng là có chi phí cao đối với những mạng kiểu này, vì vậy cần phải đưa ra một giải thuật tối ưu về mặt sử dụng năng lượng của các nút mạng.

Câu hỏi 4:

a) Để thực hiện thành công 1 tiến trình sử dụng SR, hệ thống sẽ cần n-1 thông điệp gửi đi và n-1 thông điệp trả lời REPLY và n-1 thông điệp release => vậy cần 3 * (n-1) thông điệp.
b) Cải thiện trên là đúng, trường hợp tốt nhất là gửi n-1 request và n-1 release, trường hợp tồi nhất là gửi cả n-1 request, n-1 release và n-1 REPLY. 2(n-1) < số thông điệp < 3*(n-1).
   Vậy nên tiết kiệm có thể tiết kiệm được n-1 thông điệp.

Câu hỏi 5:

Ban đầu, flag[i]=1 tức là tiến trình đang ở ngoài phòng chờ, để vào được phòng chờ thì cửa phải được mở, cửa chỉ được mở khi và chỉ khi trong phòng chờ không còn có ai nữa, hay nói cách khác phải đợi các tiến trình trong phòng chờ ra hết phòng chờ để tiến trình cuối cùng mở cửa.

Nên thao tác `await(all flag[1..N]∈{0,1,2})` là đợi các tiến trình ra khỏi phòng chờ để cửa được mở, suy ra cờ 0 và cờ 2 sẽ nằm trong các trạng thái : Rời phòng, mở lại nếu không còn ai trong phòng chờ, đứng ở ngoài cổng xếp hàng. Mà vì dòng code cuối cùng là gán cờ bằng 0 nên 2 sẽ là đứng ngoài cổng chờ xếp hàng hay nói cách khác là chờ tiến trình khác vào phòng chờ, 0 sẽ là rời phòng và mở lại cổng nếu không còn ai trong phòng chờ.

Sau khi hàm await thỏa mãn thì tiến trình sẽ vào phòng chờ, cờ 3 sẽ được gán tức cờ 3 là đứng đợi trong phòng chờ. Khi ở trong phòng chờ thì tiến trình phải chờ sao cho tất cả các tiến trình đã vào phòng chờ và cửa ra phải được mở, tức là không tồn tại tiến trình nào ở ngoài phòng chờ, tức cờ của các tiến trình phải khác 1 : **if**`any flag[1..N]=1:`

```
flag[i] ← 2
await(any flag[1..N]=4)
```

Nếu tồn tại bất cứ 1 tiến trình nào có cờ = 1 thì sẽ toàn bộ các tiến trình ở phòng chờ phải chờ sao cho đủ số tiến trình vào phòng chờ, gán bằng 2. Vậy cờ 2 sẽ là chờ tiến trình khác vào phòng chờ.

Khi các tiến trình đã vào đủ thì sẽ đóng cửa vào lại

➔ Cờ 4 sẽ là cờ đóng cửa vào.

Chờ tiến trình khác vào phòng chờ: 2

Cổng vào đóng: 4

Tiến trình i đang ở ngoài phòng chờ :1

Rời phòng, mở lại nếu không còn ai trong phòng chờ :0

Đứng đợi trong phòng chờ: 3