**BÀI TẬP LÝ THUYẾT**

**MÔN : HỆ PHÂN TÁN**

**CHƯƠNG 4: ĐỒNG BỘ HÓA**

**Họ và tên: Lê Thị Yên**

**MSSV: 20183861**

**Mã lớp: 128745**

# **Câu hỏi 1:** Trình bày 1 ví dụ để mô phỏng vấn đề gặp phải khi các máy tính/tiến trình hoạt động trong hệ thống phân tán mà không có đồng hộ vật lý dùng chung.

* Ví dụ: một mạng nhận được gói tin có thời gian gửi và sau thời gian chờ trên máy đó.
* Ví dụ trên nếu xảy ra trong trường việc chuyển tiền và gửi tiền vào tài khoản ngân hàng sẽ gây ảnh hưởng nghiêm trọng. Tiền được gửi đến trước nhưng theo hệ thống là đến sau khi người dùng có yêu cầu rút tiền thì người dùng sẽ không thể rút được.

# **Câu hỏi 2:** Tại sao Lamport lại đề xuất sử dụng đồng hồ logic thay cho đồng hồ vật lý trong hệ phân tán?

Theo Lamport thì ko nhất thiết cần sử dụng đồng hồ vật lý vì ông quan trọng thứ tự gói tin chứ không phải thời gian cụ thể:

* Mối quan hệ “xảy ra trước” (happens-before) -> diễn ra với 2 ngữ cảnh:

+ Nếu a, b là các sự kiện của cùng 1 tiến trình, và a xảy ra trước b, thì a → b là đúng.

+ Nếu a là sự kiện gửi một thông điệp từ một tiến trình, b là sự kiện nhận của thông điệp đó ở 1 tiến trình khác, thì a → b

* Mối qh bắc cầu: a → b và b → c, thì a → c
* Với các sự kiện tương tranh a và b thì không có a → b và cũng không có b → a

# **Câu hỏi 3:** Đặc điểm gì của mạng không dây (wireless network) khiến cho thiết kế các giải thuật đồng bộ khác các kiểu mạng khác?

# Mạng không dây có tốc độ không ổn định nên cần tối ưu việc nhận và gửi thông điệp

# **Câu hỏi 4:** Giải thuật Lamport được đưa ra để thực hiện loại trừ lẫn nhau (mutual exclusion). Giải thuật được mô tả như sau: Hệ thống có n tiến trình: P1, P2, ... Pn. Có 1 tài nguyên chia sẻ dùng chung gọi là SR (Shared Resource). Mỗi tiến trình sẽ lưu trữ một hàng đợi queuei để lưu các yêu cầu của các tiến trình khác khi chưa được thực hiện. Khi tiến trình Pi muốn truy cập vào SR, nó sẽ quảng bá 1 thông điệp REQUEST(tsi,i) cho tất cả các tiến trình khác, đồng thời lưu trữ thông điệp đó vào hàng đợi của mình (queuei) trong đó tsi là timestamp của yêu cầu. Khi 1 tiến trình Pj nhận được yêu cầu REQUEST(tsi,i) từ tiến trình Pi thì nó đưa yêu cầu đó vào hàng đợi của mình (queuej) và gửi trả lại cho Pi thông điệp REPLY. Tiến trình Pi sẽ tự cho phép mình sử dụng SR khi nó kiểm tra thấy yêu cầu của nó nằm ở đầu hàng đợi queuei và các yêu cầu khác đều có timestamp lớn hơn yêu cầu của chính nó. Tiến trình Pi, khi không dùng SR nữa sẽ xóa yêu cầu của nó khỏi hàng đợi và quảng bá thông điệp RELEASE cho tất cả các tiến trình khác. Khi tiến trình Pj nhận được thông điệp RELEASE từ Pi thì nó sẽ xóa yêu cầu của Pi trong hàng đợi của nó

Trả lời:

(n-1) thông điệp request

(n-1) thông điệp reply

(n-1) thông điệp release

=> Tổng = 3(n-1)



2(n-1) < ... < 3(n-1)

Tốt nhất> ... >Xấu nhất

=> Tiết kiệm (n-1) thông điệp reply

# **Câu hỏi 5:** Giải thuật Szymanski được thiết kế để thực hiện loại trừ lẫn nhau. Ý tưởng của giải thuật đó là xây dựng một phòng chờ (waiting room) và có đường ra và đường vào, tương ứng với cổng ra và cổng vào. Ban đầu cổng vào sẽ được mở, cổng ra sẽ đóng. Nếu có một nhóm các tiến trình cùng yêu cầu muốn được sử dụng tài nguyên chung SR (shared resource) thì các tiến trình đó sẽ được xếp hàng ở cổng vào và lần lượt vào phòng chờ. Khi tất cả đã vào phòng chờ rồi thì tiến trình cuối cùng vào phòng sẽ đóng cổng vào và mở cổng ra. Sau đó các tiến trình sẽ lần lượt được sử dụng tài nguyên chung. Tiến trình cuối cùng sử dụng tài nguyên sẽ đóng cổng ra và mở lại cổng vào.

Trả lời:

- Chờ tiến trình khác vào phòng chờ: 3

- Cổng vào được đóng: 4

- Tiến trình i đang ở ngoài phòng chờ: 1

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

- Đứng đợi trong phòng chờ: 3

Giải thích:

* Trước khi sử dụng tài nguyên sẽ là trạng thái “Cổng vào đóng” -> 4
* Sau khi sử dụng tài nguyên sẽ là trạng thái “Rời phòng, mở lại cổng vào nếu không còn ai trong phòng chờ” -> 0
* Từ đó, “Đứng đợi trong phòng chờ” -> 2, trước phòng chờ là “Chờ tiến trình khác vào phòng chờ” -> 3
* Còn lại 1 -> “Tiến trình i đang ở ngoài phòng chờ”
* Trước khi sử dụng tài nguyên sẽ là trạng thái “Cổng vào đóng” → 4
* •Sau khi sử dụng tài nguyễn sẽ là trạng thái “Rời phòng, mở lại cổng vào nếu không còn ai trong phòng chờ” → 0
* •Từ đó, “Đứng đợi trong phòng chờ” → 2, trước phòng c