**Tuần 6:**

Câu 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.  
  
Các tiến trình nằm trên các máy khác nhau, do đó các máy có độ lệch về thời gian vật lý.  
  
Ví dụ:  
  
Lập trình và dịch compile từ file .c sang file .o  
  
Lập trình viên sửa file .c thì compile không dịch toàn bộ mà chỉ dịch những file .c có thời gian sau thời gian của file .o đã dịch tương ứng, hay là chỉ dịch những file mà lập trình viên sửa.  
  
Giả sử: nằm trên máy khác nhau: 1 máy lập trình, 1 máy dịch:  
  
ở thời điểm người lập trình viên chỉnh sửa file .c có thể xảy ra trước thời điểm file .o tương ứng do độ lêch thời gian giữa máy lập trình và máy dịch do đó dù file .c đã chỉnh sửa nhưng sẽ không được compile lại nữa   
  
Do đó sẽ gây ảnh hưởng tới chương trình, làm chó kết quả chạy không như mong muốn.  
  
Câu 2: Tại sao Lamport lại đề xuất sử dụng đồng hồ logic thay cho đồng hồ vật lý trong hệ phân tán?  
  
Vì: Lamport chỉ quan tâm đến thứ tự các sự kiện xảy ra trong hệ phân tán,  
  
mà không cần hiệu chỉnh tất cả các máy cùng thời gian vật lý.  
  
Chỉ hiệu chỉnh những clock mà thời gian gưir lớn hơn thời gian nhận bằng cách gán cho thời gian nhận bằng Max của( thời gian gửi +1, thời gian nhận)  
  
Câu 3: Đặc điểm gì của mạng không dây (wireless network) khiến cho thiết kế các giải thuật đồng bộ khác các kiểu mạng khác?  
  
Mỗi nút trong mạng hoạt động dựa trên năng lượng của nó.  
  
Do đó cần tìm cơ chế để tiết kiệm năng lượng, tối ưu hóa gửi và nhận  
  
chỉ những nút nhận thông điệp mới tham gia đồng bộ hóa.  
  
Mỗi nút biết được độ lệch của mình so với các nút khác.  
  
Câu 4:  
  
Nếu có n tiến trình P1,P2, .. Pn;  
  
a, Trong hệ thống sẽ cẫn tất cả:  
  
Mỗi tiến trình sẽ gửi n-1 thông điệp REQUEST tới n-1 tiến trình còn lại do đó có: n(n-1) thông điệp REQUEST  
  
Mỗi tiến trình sẽ gửi n-1 thông điệp REPLY trả lại n-1 tiến trình còn lại do đó có: n(n-1) thông điệp REPLY  
  
khi 1 tiến trình sử dụng xong SR thì sẽ gửi n-1 thông điệp RELEASE tới n-1 tiến trình còn lại;  
  
Do đó sẽ có 2n(n-1) + n-1 thông điệp.  
  
b, <Nếu timestamp nhỏ hơn sẽ không gửi trả REPLY??>  
  
Đúng vì 1 tiến trình muốn dùng SR sẽ cần nhận tất cả các REPLY từ n-1 tiến trình còn lại:  
  
Mỗi tiến trình sẽ gửi n-1 thông điệp REQUEST tới n-1 tiến trình còn lại do đó có: n(n-1) thông điệp REQUEST  
  
tiến trình i sẽ gửi i-1 thông điệp REPLY do cơ chế trên do đó có: n(n-1)/2 thông điệp REPLY  
  
tiến trình khi sử dụng xong SR sẽ gửi lại n-1 thông điệp REPLY tới n-1 thông điệp còn lại  
  
Do đó sẽ có tất cả n(n-1) +n(n-1)/2 + n-1 thông điệp  
  
Câu 5:  
  
2:Chờ tiến trình khác vào phòng chờ  
  
Chờ các tiến trình khác vào phòng chờ và chwof cổng đóng để sử dụng SR  
  
4:Cổng vào được đóng  
  
Các tiến trình ở trong phòng chờ sẽ chờ cho đến khi cổng được đóng để bắt đầu sử dụng SR  
  
1:Tiến trình i đang ở ngoài phòng chờ:  
  
tiến trình i ở ngoài phòng chờ khi: sử dụng xong SR, đang chờ vào phòng chờ . hoặc do cổng đã bị đóng.  
  
0: Rời phòng, mở lại cổng vào nếu không còn ai trong phòng chờ  
  
3: Đứng đợi trong phòng chờ