IT4371: Các hệ phân tán

Spring 2015

Bài tập

Hoàng Quốc Hồng Nhật

Bộ môn Hệ thống thông tin
Viện công nghệ thông tin và truyền thông
Đại học Bách khoa Hà Nội

Hôm nay…

- Bài tập:
  - Đặt tên (Naming): Thuật toán Chord
  - Đồng bộ hóa (Synchronization): Đồng bộ thời gian
    - Thuật toán Cristian
    - Đồng hồ Lamport
    - Đồng hồ vector
  - Nhất quán (Consistency): Trình tự (Ordering)
    - Total Ordering
    - Causal Ordering
    - Sequential Ordering

Đặt tên (Naming)

Đặt tên dạng phẳng

Bảng băm phân tán - Chord

Bài 1: Một vòng Chord sử dụng không gian khóa 7-bits. Hiện tại, trên vòng đang có các nút có khóa lần lượt là:

[15; 26; 38; 45; 57; 64; 75; 106; 110; 127]

Câu hỏi:

1a) Cho biết bảng finger-table của các nút [15, 64, 106, 127]

1b) Cho biết một data item có khóa 50 gửi lên từ nút 15 được lưu trữ trên vòng như thế nào?

1c) Cho biết data item có khóa 50 được tìm kiếm từ nút 64 như thế nào?

1d) Nếu thêm nút mới có khóa là 51, finger-table của nó sẽ như thế nào? Sự thay đổi trong finger-table của các nút khác ra sao?

1e) Khi nút 51 tham gia, có cần cập nhật ngay lập tức finger-table của tất cả các nút khác để tìm được khóa 50 hay không?

Đồng bộ hóa

Đồng bộ thời gian

Thuật toán Cristian

Bài 2: Một client cố gắng đồng bộ đồng hồ với server một lần mỗi phút. Kết quả của 3 lần gửi được tổng kết như bảng bên dưới. Thời gian được biểu diễn dưới dạng (hr: min: sec: msec).

Sent at (local time)	Round-trip delay	Response
10:54:00:00	18 ms	10:54:00:10
10:55:00:00	24 ms	10:55:00:12
10:56:00:00	22 ms	10:56:00:10

Câu hỏi: Giả sử thời gian xử lý tại máy chủ bằng không. Client sẽ điều chỉnh đồng hồ của mình như thế nào ở mỗi lần gửi sau khi nhận được phản hồi?

Đồng bộ hóa

Đồng bộ thời gian

Thuật toán Cristian

Bài 2:

Sent at (local time)	Round-trip delay	Response
10:54:00:00	18 ms	10:54:00:10
10:55:00:00	24 ms	10:55:00:12
10:56:00:00	22 ms	10:56:00:10

Lần 1: Client chạy chậm → Chỉnh lên tới 10:54:00:19

Lần 2: Chính xác → Không cần chỉnh

Lần 3: Client chạy nhanh → Chỉnh xuống còn 10:56:00:21

Đồng bộ hóa

Đồng bộ thời gian

Thuật toán Cristian

Bài 3: Xét hai server NTP đồng bộ hóa đồng hồ của chúng. Máy chủ B nhận được thông điệp M của máy chủ A lúc 02:00:00 mang dấu thời gian 02:00:04 và trả lời bằng tin nhắn M'. Máy chủ A nhận được thông báo M' lúc 02:00:08 mang dấu thời gian của B 02:00:03.

Câu hỏi: Ước tính độ lệch thời gian giữa các máy chủ B và A?

Đồng bộ hóa

Đồng bộ thời gian

Thuật toán Cristian

Bài 3:

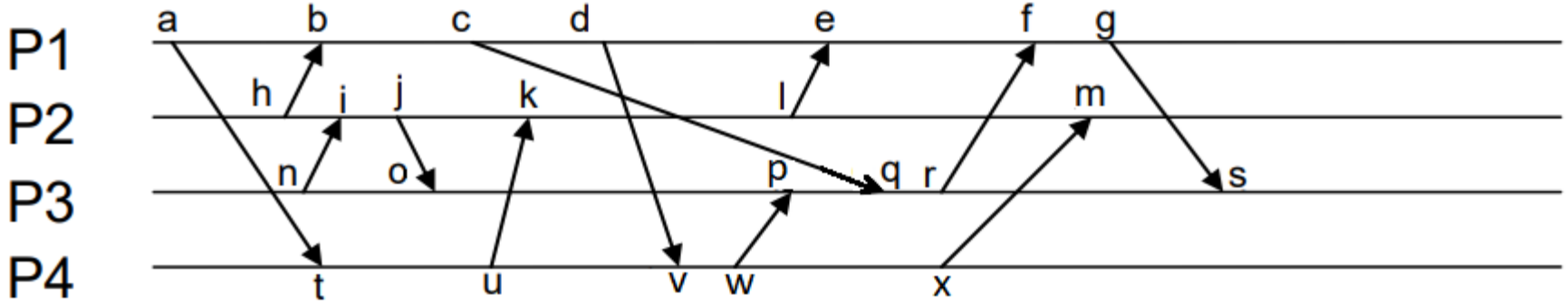
(02:00:00 – 02:00:04 + 02:00:03 – 02:00:08)/2 = -4.5 s

Đồng bộ hóa

Đồng bộ thời gian

Đồng hồ Lamport và đồng hồ vector

Bài 4: Trong hình dưới có bốn quy trình (P1, P2, P3, P4) với các sự kiện a, b, c, …, u và các tin nhắn giao tiếp giữa chúng.
Giả sử các giá trị đồng hồ logic ban đầu được khởi tạo là 0.



Câu hỏi:

4a) Đánh dấu thời gian cho mỗi sự kiện theo đồng hồ Lamport.

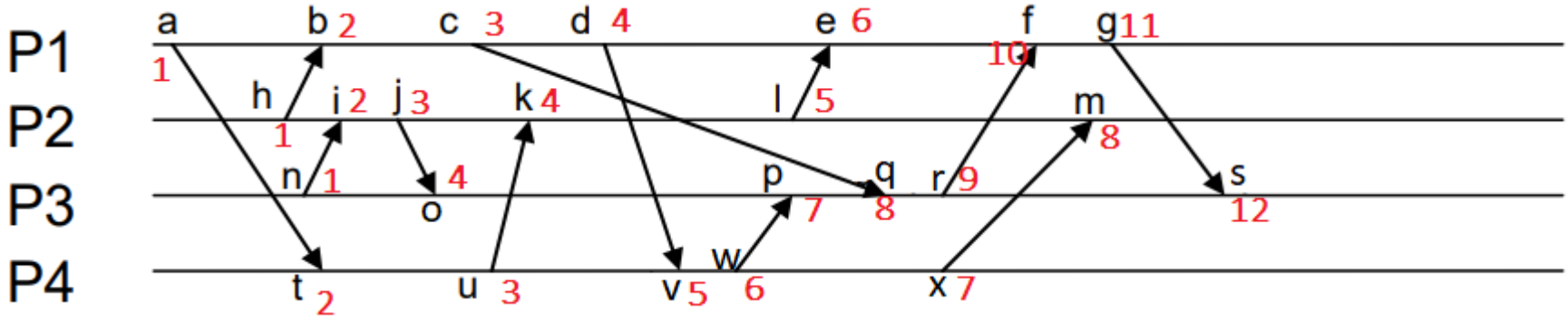
4b) Đánh dấu thời gian cho mỗi sự kiện theo đồng hồ vector.

Đồng bộ hóa

Đồng bộ thời gian

Đồng hồ Lamport và đồng hồ vector

4a) Đánh dấu thời gian cho mỗi sự kiện theo đồng hồ Lamport.

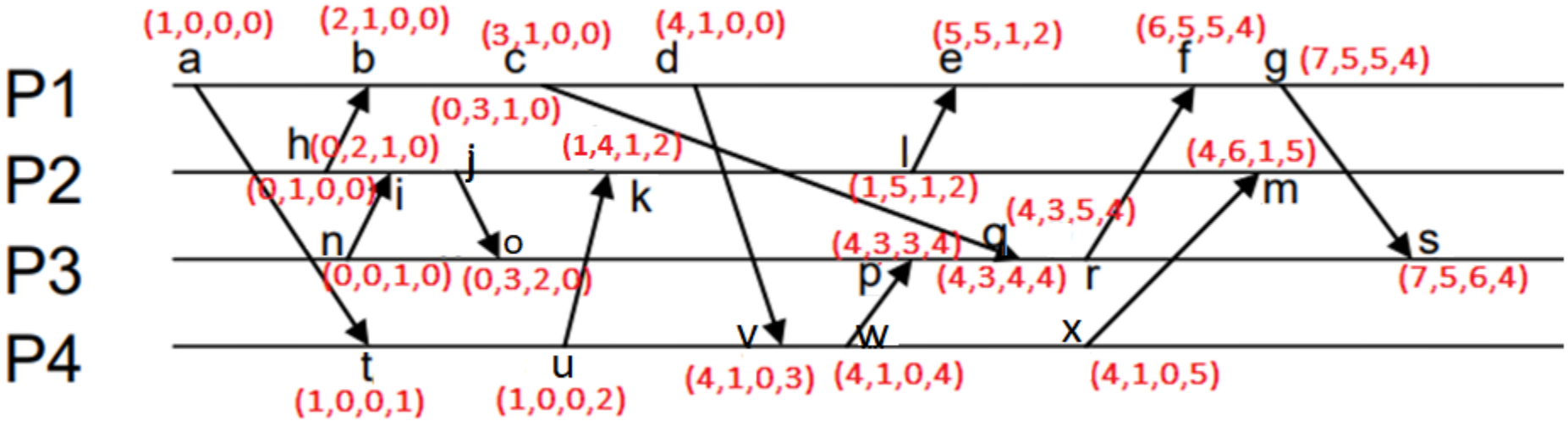


Đồng bộ hóa

Đồng bộ thời gian

Đồng hồ Lamport và đồng hồ vector

4b) Đánh dấu thời gian cho mỗi sự kiện theo đồng hồ vector.

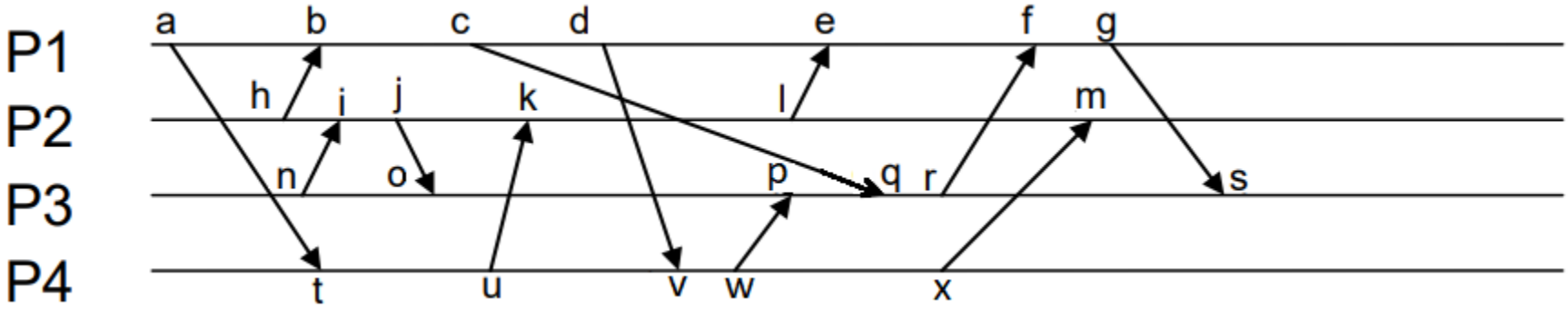


Nhất quán

Ordering

Total Ordering và Causal Ordering

Bài 5: Có các tiến trình và sự kiện như bài 4.



Câu hỏi:

5a) Cho biết các sự kiện xảy ra trước **w**, xảy ra sau **w** và đồng thời với **w?**

5b) Đưa ra 1 cách sắp xếp các sự kiện theo causal ordering ?

Nhất quán

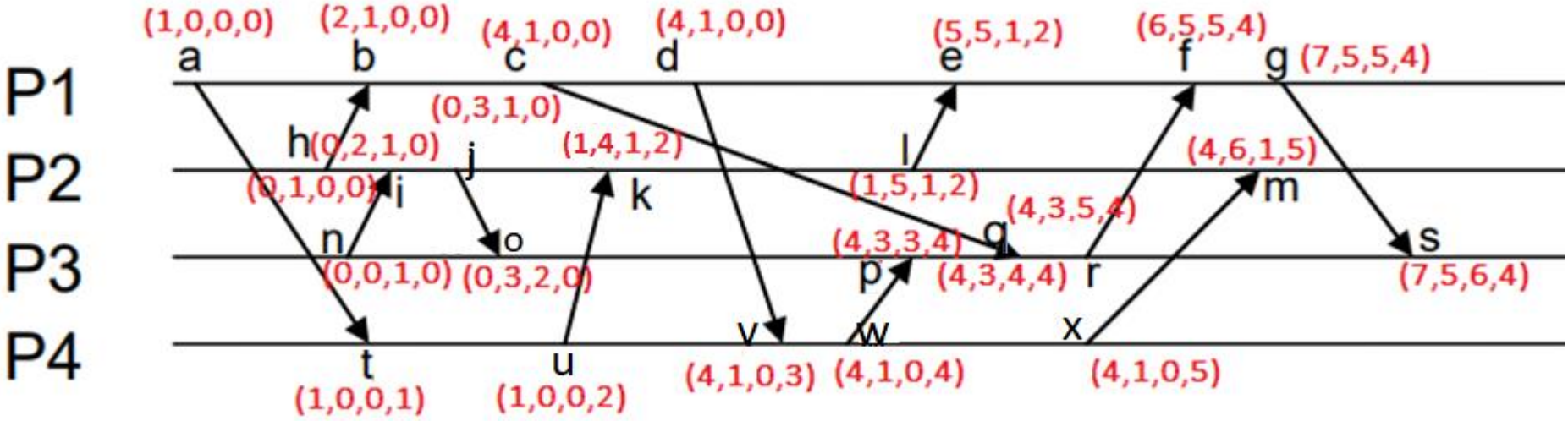
Ordering

Total Ordering và Causal Ordering

5a) Xảy ra trước **w**: **a, t, h, b, c, d, v, u**

Xảy ra sau **w**: **p, x, m, r, f, g, s, q**

Xảy ra đồng **w**: **i, j, o, k, l, e**



Nhất quán

Ordering

Total Ordering và Causal Ordering

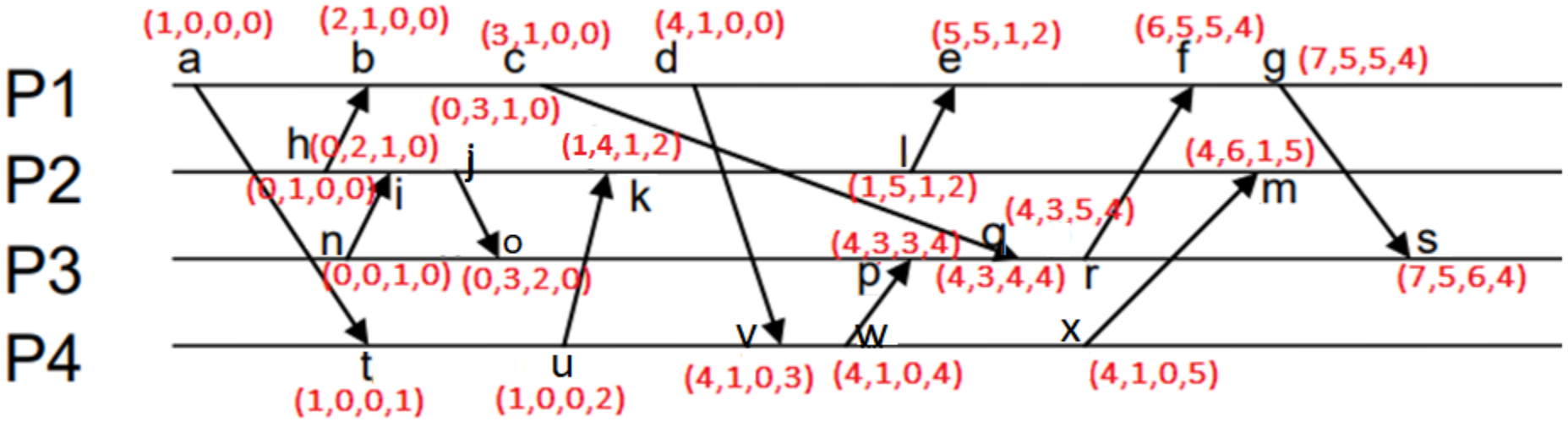
5b) Ví dụ:

a → t → u → k → l → e → g → s

a → t → u → v → w → p → r → f → g → s

a → b → c → q → r → s

…



Nhất quán

Ordering

Sequential Ordering

Bài 6: Cho biết thứ tự thực thi các thao tác đọc (R) và ghi (W) của các tiến trình sau có thỏa mãn sequential ordering hay không?

Câu hỏi:

6a) P1: W(x,1);
P2: R(x,0); R(x,1)

6b) P1: W(x,1);
P2: R(x,1); R(x,0);

6c) P1: W(x,1);
P2: W(x,2);
P3: R(x,1); R(x,2);

6d) P1: W(x,1); R(x,1); R(y,0);
P2: W(y,1); R(y,1); R(x,1);
P3: R(y,1); R(x,0);

6e) P1: W(x,1);
P2: W(x,2);
P3: R(x,2); R(x,1);
P4: R(x,1); R(x,2);

6f) P1: W(x,1); R(x,1); R(y,0);
P2: W(y,1); R(y,1); R(x,1);
P3: R(x,1); R(y,0);
P4: R(y,0); R(x,0);

Nhất quán

Sequential Ordering

6a) P1: W(x,1);
P2: R(x,0); R(x,1)

P2: R(x,0); **P1:** W(x,1); **P2:** R(x,1)

6b) P1: W(x,1);
P2: R(x,1); R(x,0);

Không thỏa mãn vì không thể đọc được x = 0 khi đã ghi x = 1

6c) P1: W(x,1);
P2: W(x,2);
P3: R(x,1); R(x,2);

P1: W(x,1); **P3:** R(x,1); **P2:** W(x,2); **P3:** R(x,2);

Nhất quán

Sequential Ordering

6d) P1: W(x,1); R(x,1); R(y,0);
P2: W(y,1); R(y,1); R(x,1);
P3: R(y,1); R(x,0);

Không thỏa mãn, vì R(y,0) trên P1 xảy ra trước W(y,1) trên P2, và R(x,0) trên P3 phải xảy ra trước W(x,1) trên P1, nên R(y,1) trên P3 xảy ra trước W(y,1) trên P2 → vô lý.

6e) P1: W(x,1);
P2: W(x,2);
P3: R(x,2); R(x,1);
P4: R(x,1); R(x,2);

Không thỏa mãn, vì P3 và P4 thấy hành động ghi thực hiện bởi P1 và P2 theo các thức tự khác nhau.

Nhất quán

Sequential Ordering

6f) P1: W(x,1); R(x,1); R(y,0);
P2: W(y,1); R(y,1); R(x,1);
P3: R(x,1); R(y,0);
P4: R(y,0); R(x,0);

P4: R(y,0); R(x,0);
P1: W(x,1); R(x,1); R(y,0);
P3: R(x,1); R(y,0);
P2: W(y,1); R(y,1); R(x,1)