HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

CÁC HỆ THỐNG PHÂN TÁN VÀ ỨNG DỤNG

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

CHƯƠNG 4: ĐỒNG BỘ HÓA TRONG HPT

Based on the lectures of Assoc. Prof. Hà Quốc Trung

Nội dung

- Đồng bộ hóa đồng hồ vật lý
- Đồng bộ hóa đồng hồ logic
- Các thuật toán loại trừ lẫn nhau
- Các thuật toán bầu chọn

Mở đầu

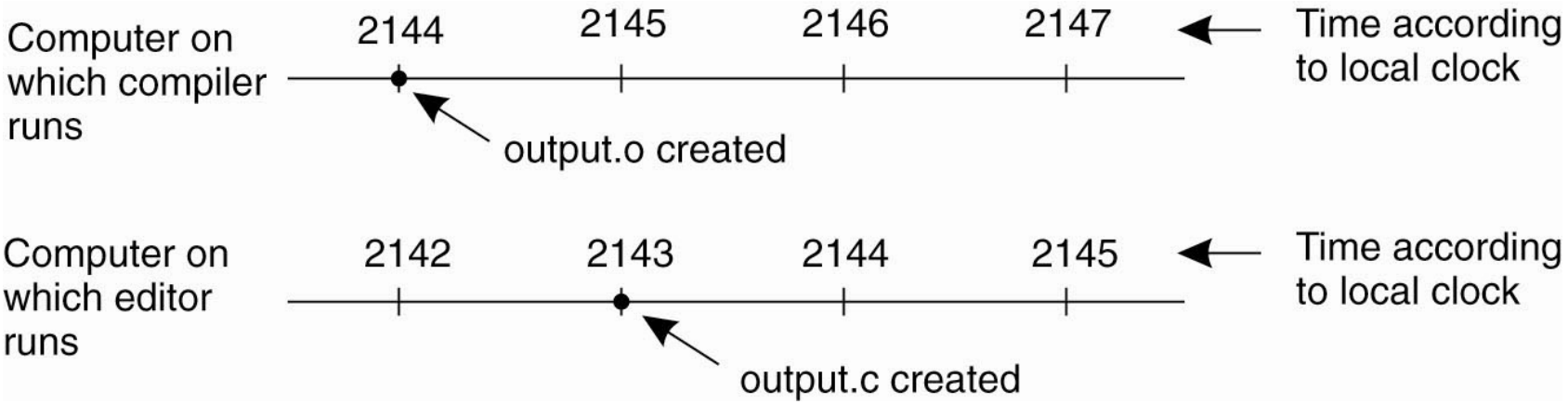
- Các tiến trình thực hiện đồng bộ hóa ntn?
  - Nhiều tiến trình cần lần lượt vào sử dụng tài nguyên chia sẻ dùng chung
  - Nhiều tiến trình cần thống nhất thứ tự các sự kiện
  - Vấn đề đối với ngữ cảnh HPT?
- Đồng bộ hóa dựa trên giá trị thời gian thực
- Đồng bộ hóa dựa trên thứ tự các sự kiện

1. Đồng bộ hóa đồng hồ vật lý

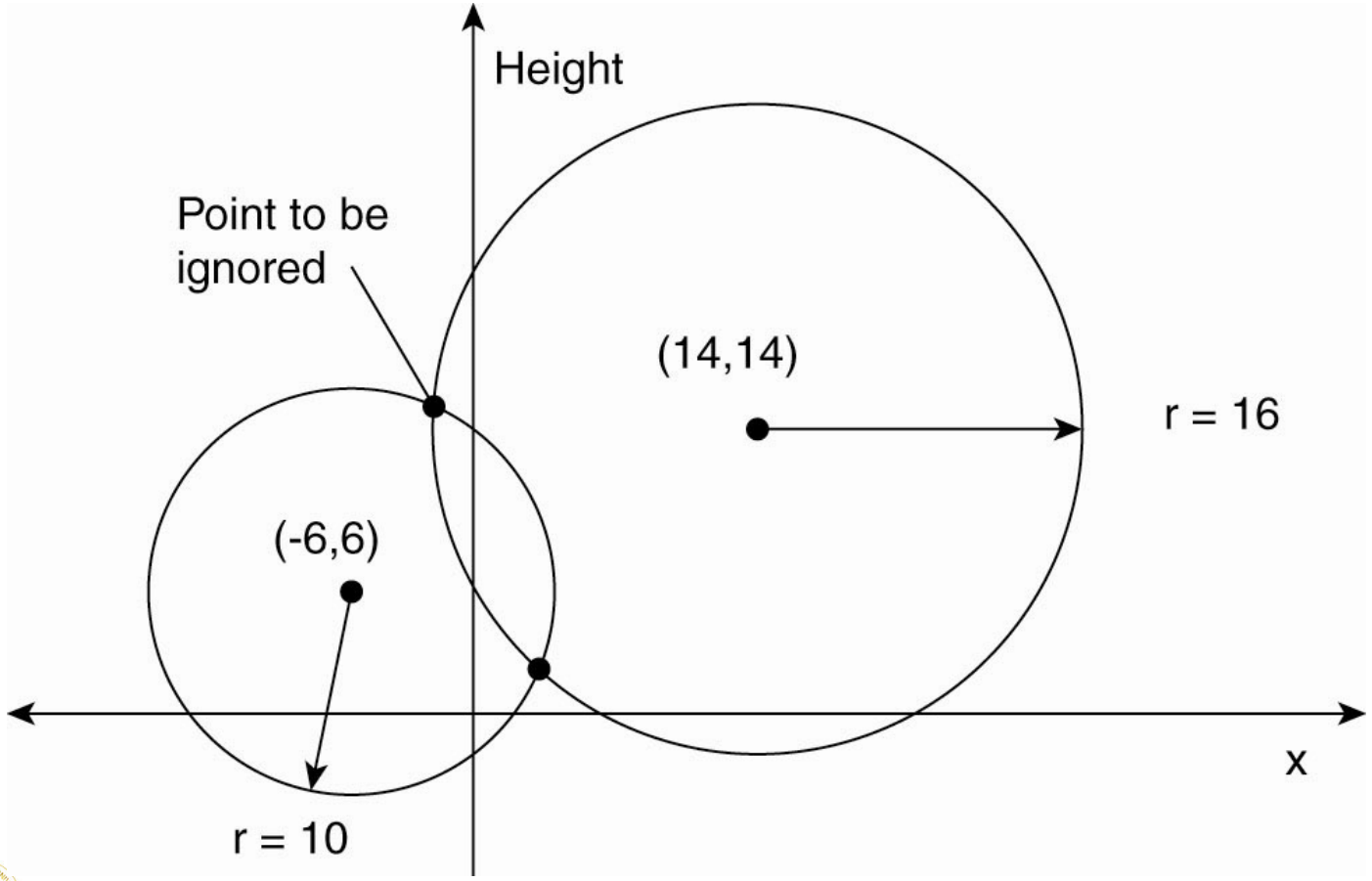
1. Đồng bộ hóa đồng hồ vật lý

- Đồng hồ vật lý
- Các vấn đề khi không đồng bộ hóa đồng hồ vật lý
- Các thuật toán đồng bộ hóa đồng hồ vật lý

Ví dụ 1: Lập trình trong HPT



VD 2: Global Positioning System (1)



Global Positioning System (2)

- Những khó khăn khi triển khai GPS trong thực tế

1. Tín hiệu đi qua các tầng khí quyển trước khi đến với thiết bị thu
2. Đồng hồ vật lý của thiết bị thu và vệ tinh không đồng bộ

Đồng hồ vật lý

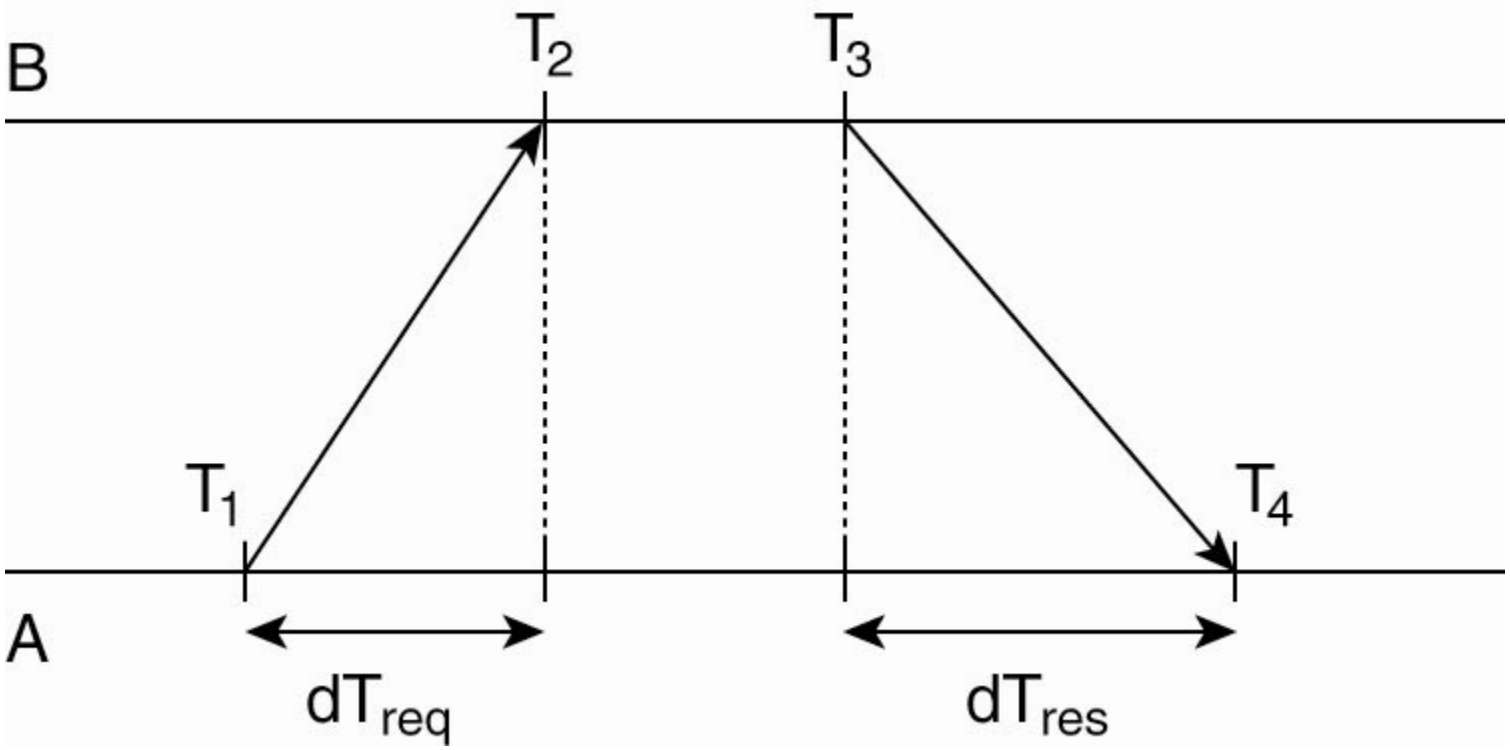
- Timer
- Counter & Holding register
- Clock tick
- Vấn đề trong HPT
  - Đồng bộ với giá trị thời gian vật lý thật
  - Đồng bộ các đồng hồ vật lý với nhau

RTC IC
(Real Time Clock)

Các thuật toán đồng bộ hóa đồng hồ vật lý

- Network Time Protocol
- Thuật toán Berkeley
- Đồng bộ hóa đồng hồ vật lý trong các mạng không day

Network Time Protocol



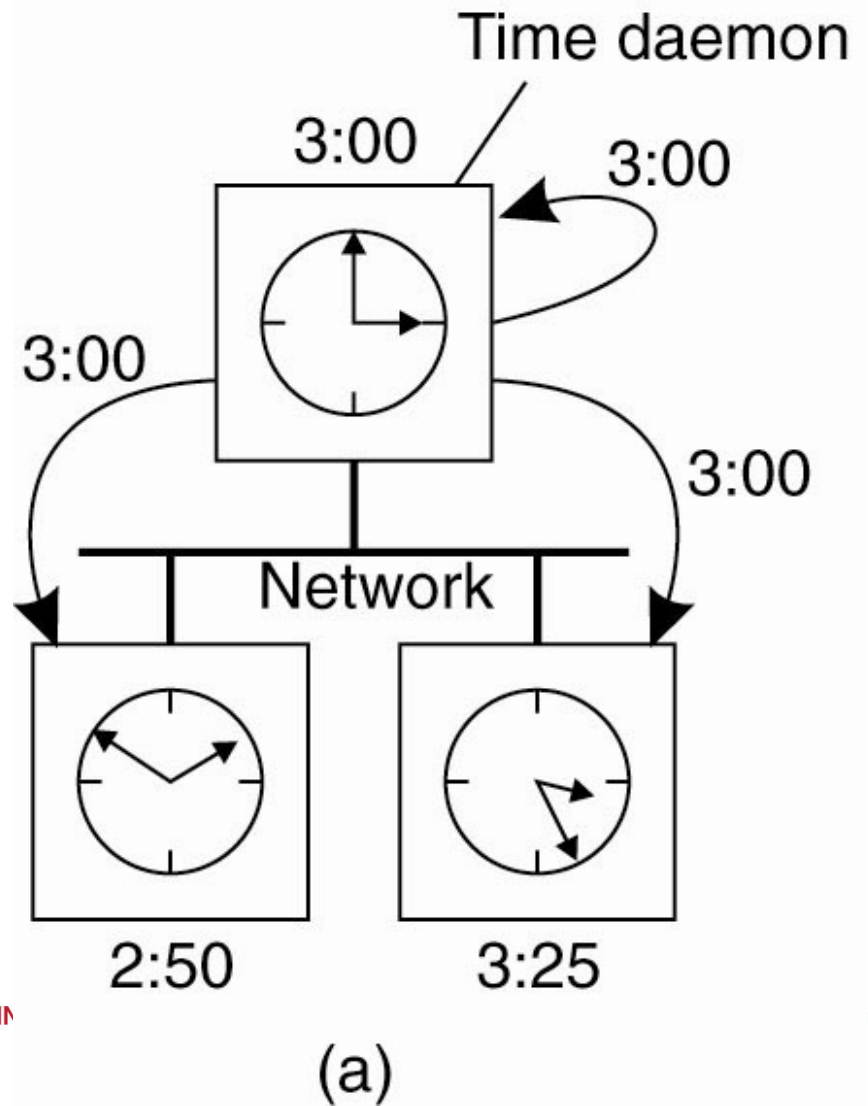
$$\theta = T_3 + \frac{(T_2 - T_1) + (T_4 - T_3)}{2} - T_4 = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

Getting the current time from a time server

Cristian's Algorithm

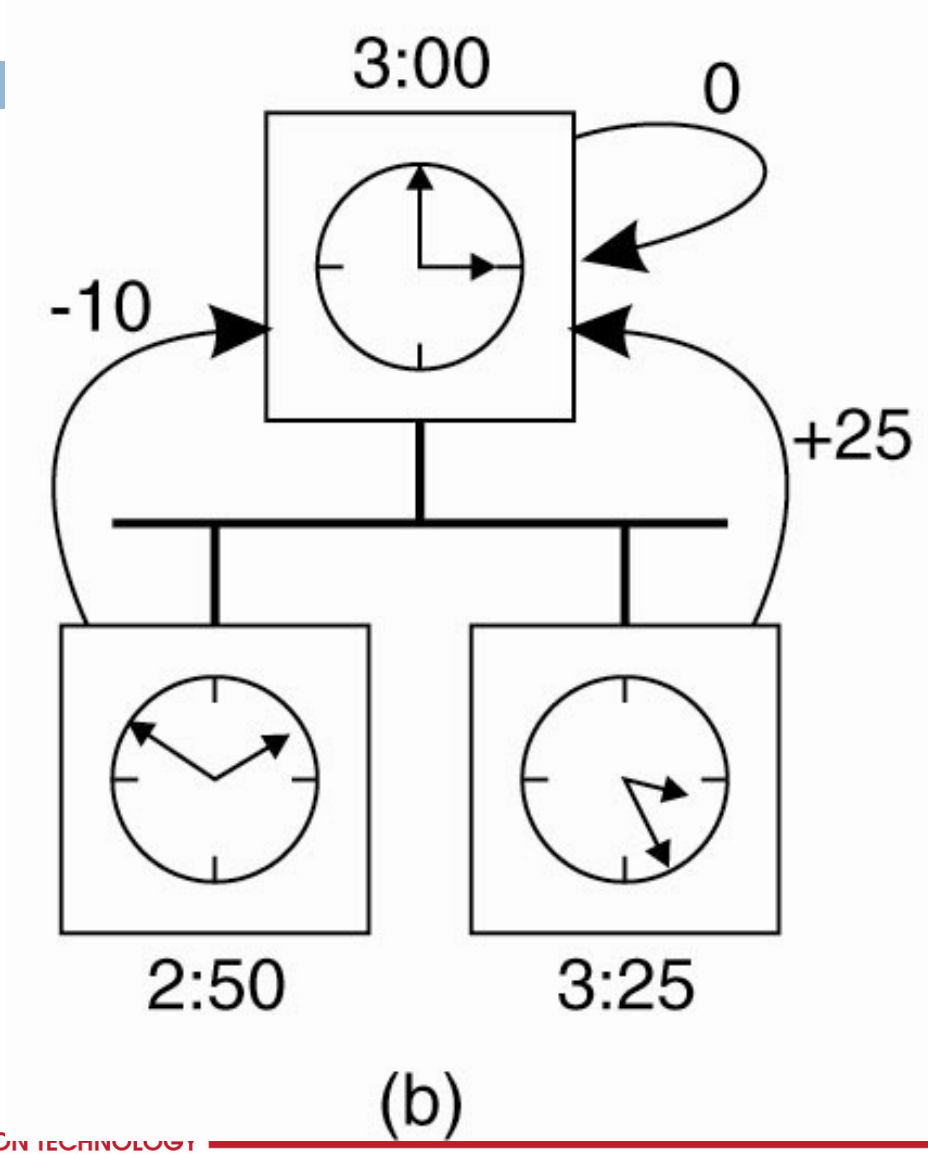
The Berkeley Algorithm (1)

- Time deamon quảng bá giá trị đồng hồ vật lý của mình



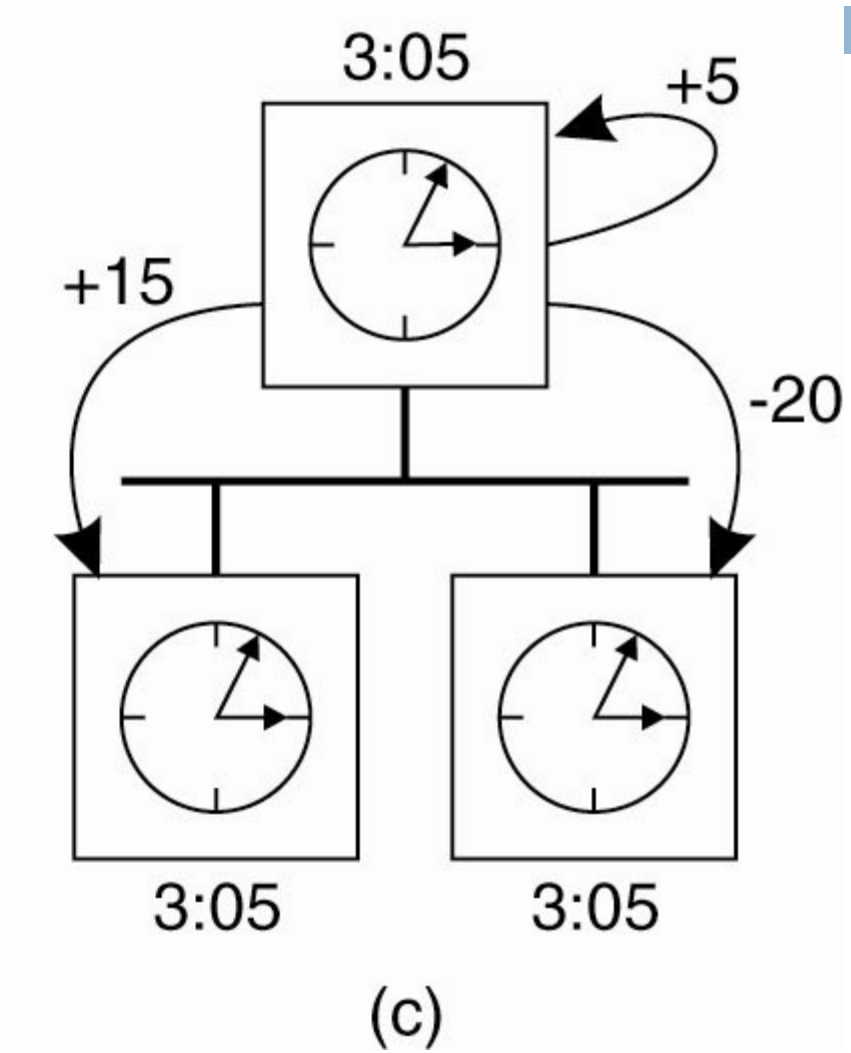
The Berkeley Algorithm (2)

- các máy khác trả lời

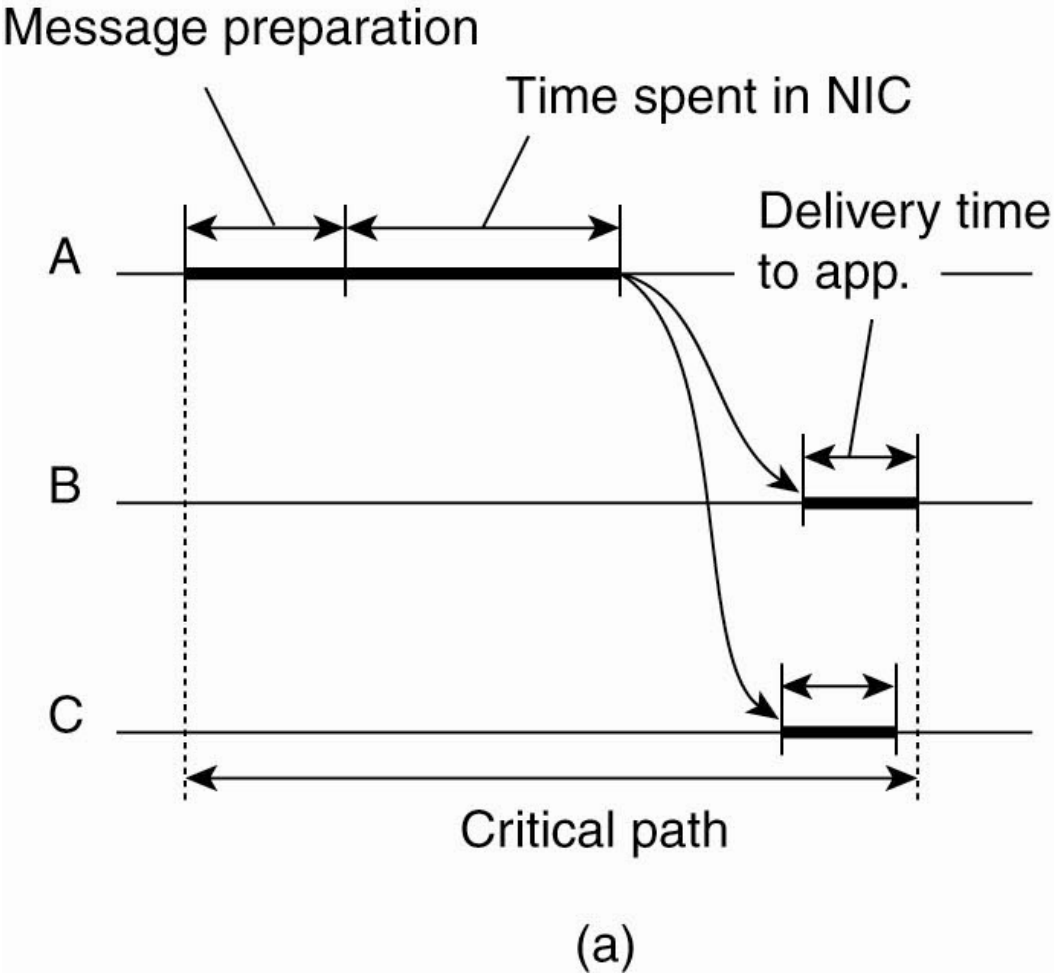


The Berkeley Algorithm (3)

- Time deamon tính toán ra giá trị trung bình và gửi cho các máy giá trị cần hiệu chỉnh

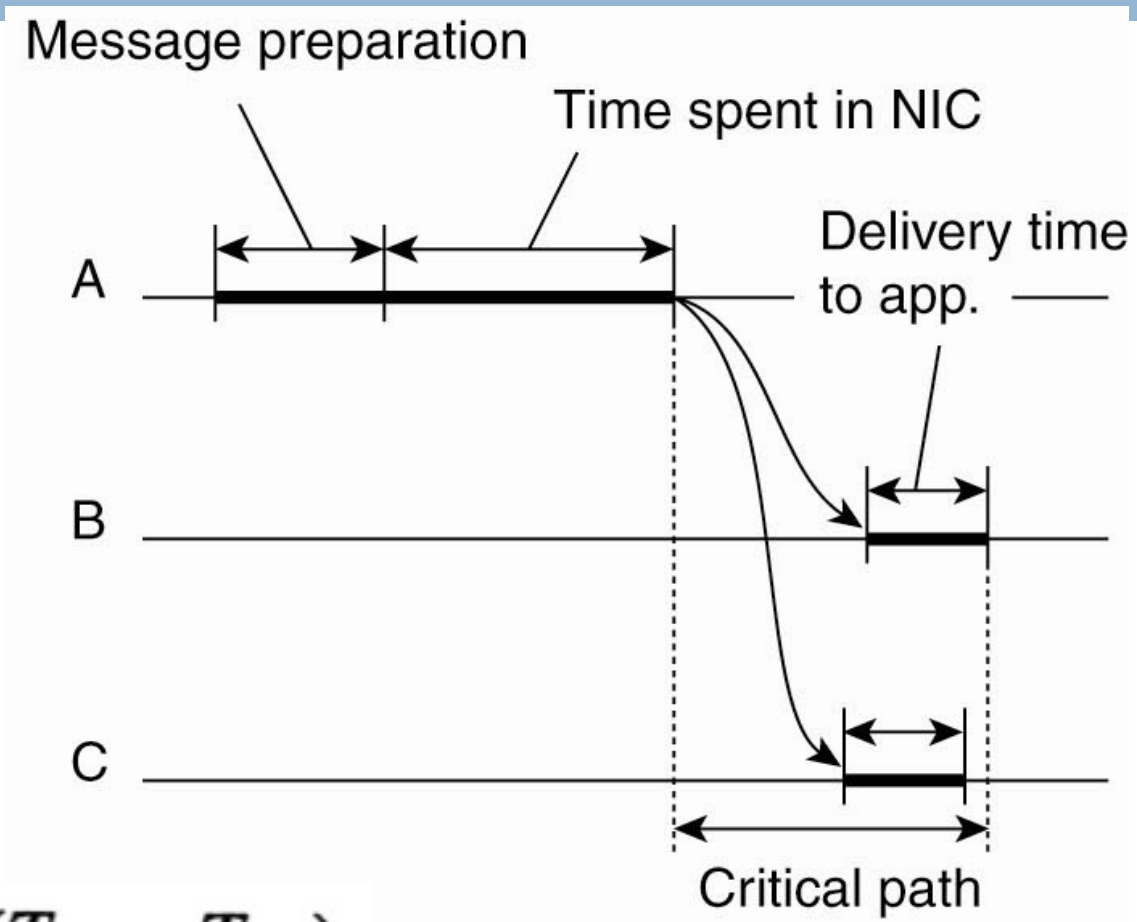


Đồng bộ hóa đhvl trong các mạng không dây (1)



Đồng bộ hóa đhvl trong các mạng không dây (2)

- **RBS** (Reference Broadcast Synchronization)



(b)

$$Offset[p,q] = \frac{\sum_{k=1}^{M}(T_{p,k} - T_{q,k})}{M}$$

2. Đồng bộ hóa đồng hồ logic

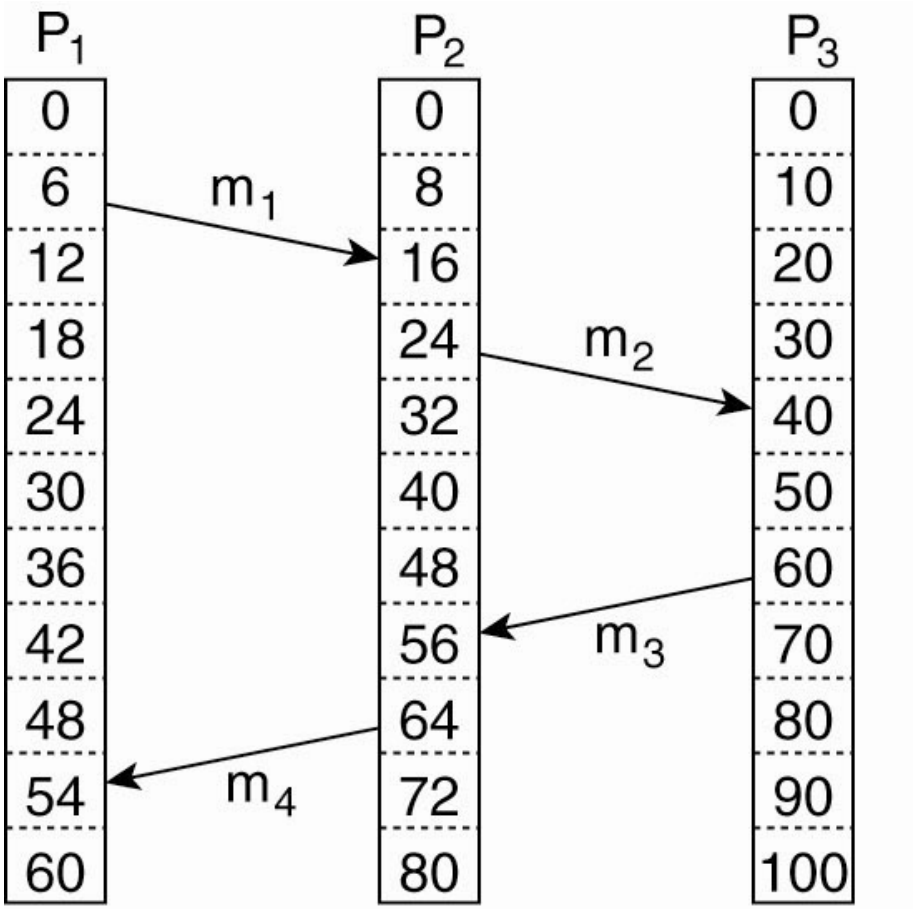
2. Đồng bộ hóa đồng hồ logic

- Cơ chế đồng bộ hóa đồng hồ logic của Lamport
- Vector clocks

2.1. Cơ chế đbh đồng hồ logic của Lamport (1)

- Mối quan hệ "xảy ra trước" (happens-before) → diễn ra với 2 ngữ cảnh:
  1. Nếu a và b là các sự kiện của cùng 1 tiến trình, và a xảy ra trước b, thì $a \rightarrow b$ là đúng.
  2. 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 $a \rightarrow b$
- Mối qh bắc cầu: $a \rightarrow b$ và $b \rightarrow c$, thì $a \rightarrow c$
- Với các sự kiện tương tranh a và b thì không có $a \rightarrow b$ và cũng không có $b \rightarrow a$

Cơ chế đbh đồng hồ logic của Lamport (2)



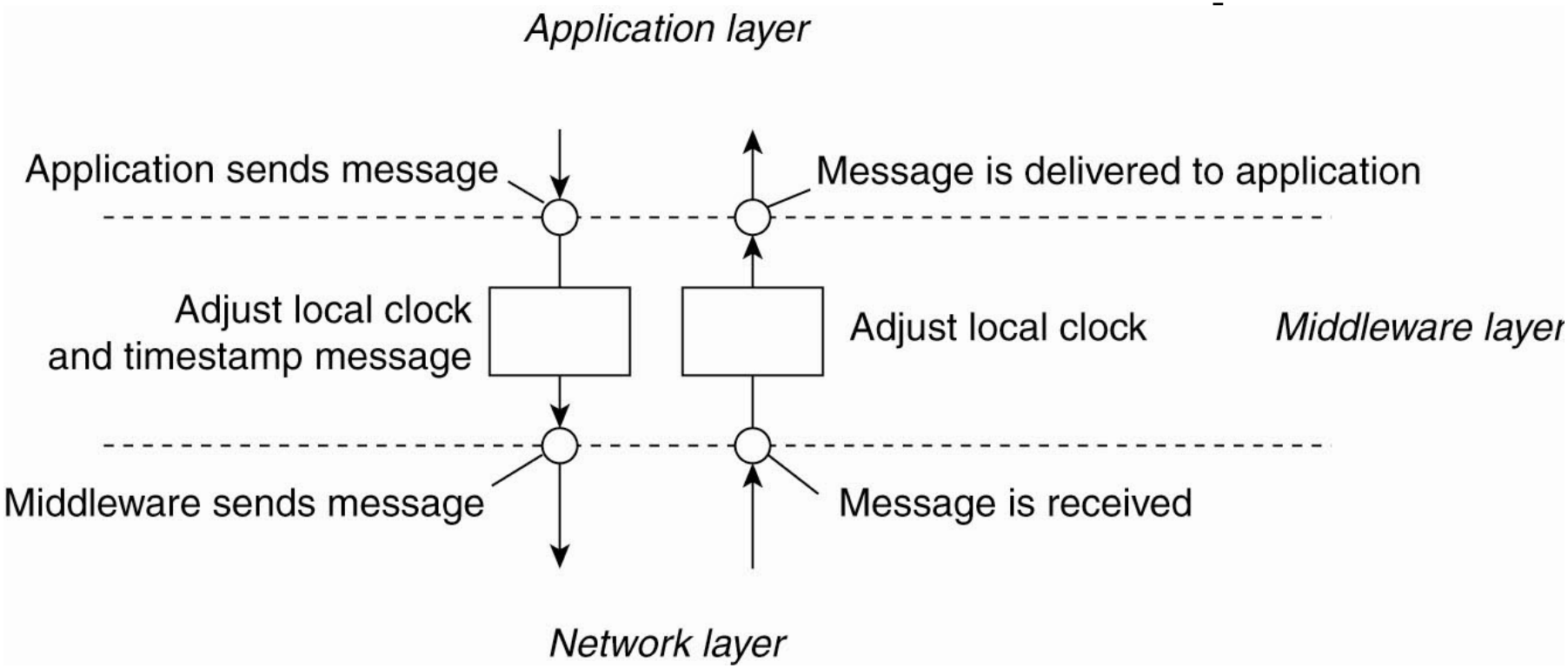
(a)

Cơ chế đbh đồng hồ logic của Lamport (3)

- Cập nhật bộ đếm C_i cho tiến trình P_i

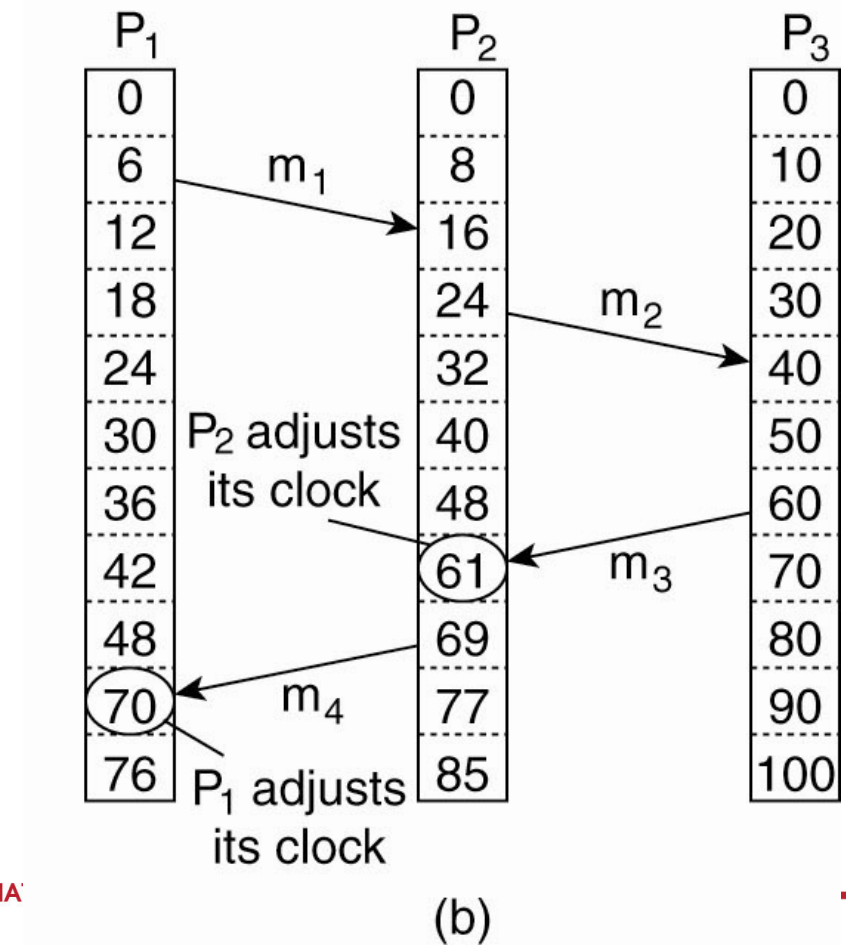
1. Trước mỗi sự kiện, P_i thực thi:
   $C_i \leftarrow C_i + 1$.
2. Khi tiến trình P_i gửi thông điệp m tới P_j, nó sẽ đặt timestamp của *m* là *ts (m)* bằng với giá trị C_i (sau khi thực hiện bước 1).
3. Khi nhận được thông điệp *m*, tiến trình P_j cập nhật lại giá trị bộ đếm cục bộ:
   $C_j \leftarrow \max\{C_j, ts\ (m)\}$, sau đó sẽ chuyển thông điệp lên tầng ứng dụng.

Cơ chế đbh đồng hồ logic của Lamport (4)

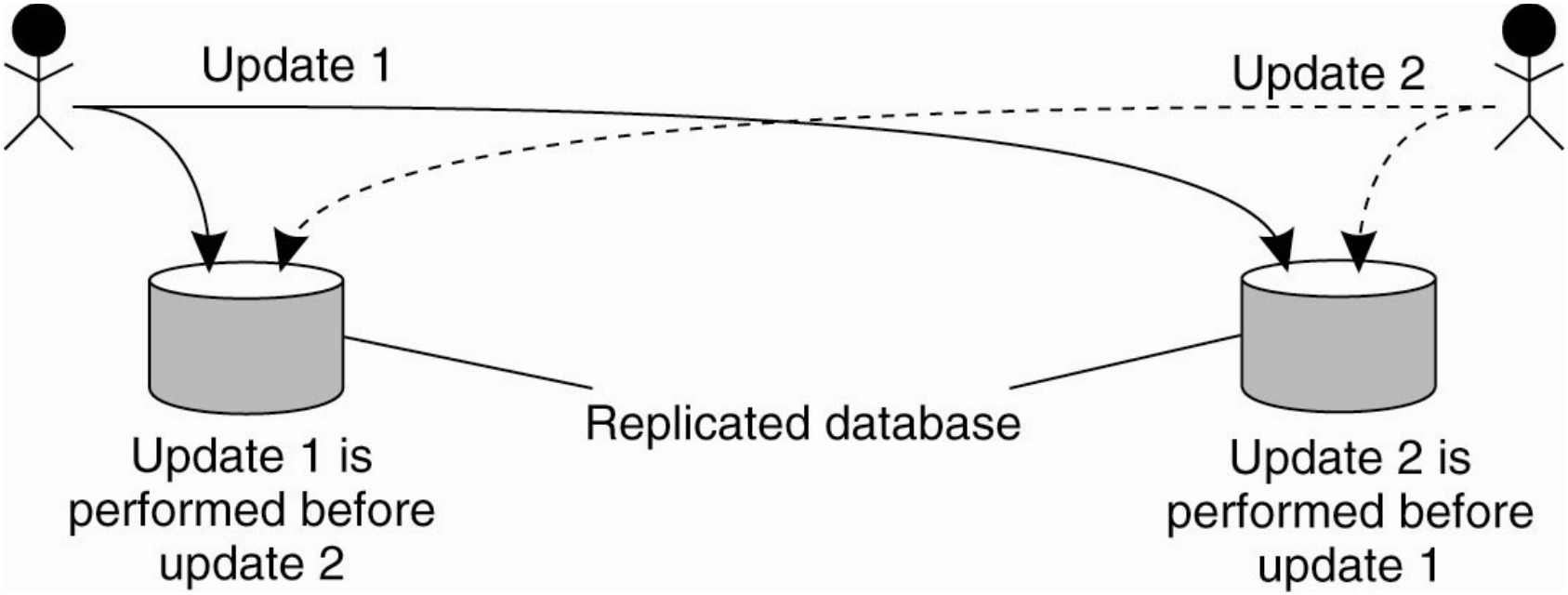


Cơ chế đbh đồng hồ logic của Lamport (5)

(b) Giải thuật Lamport hiệu chỉnh lại các giá trị clock

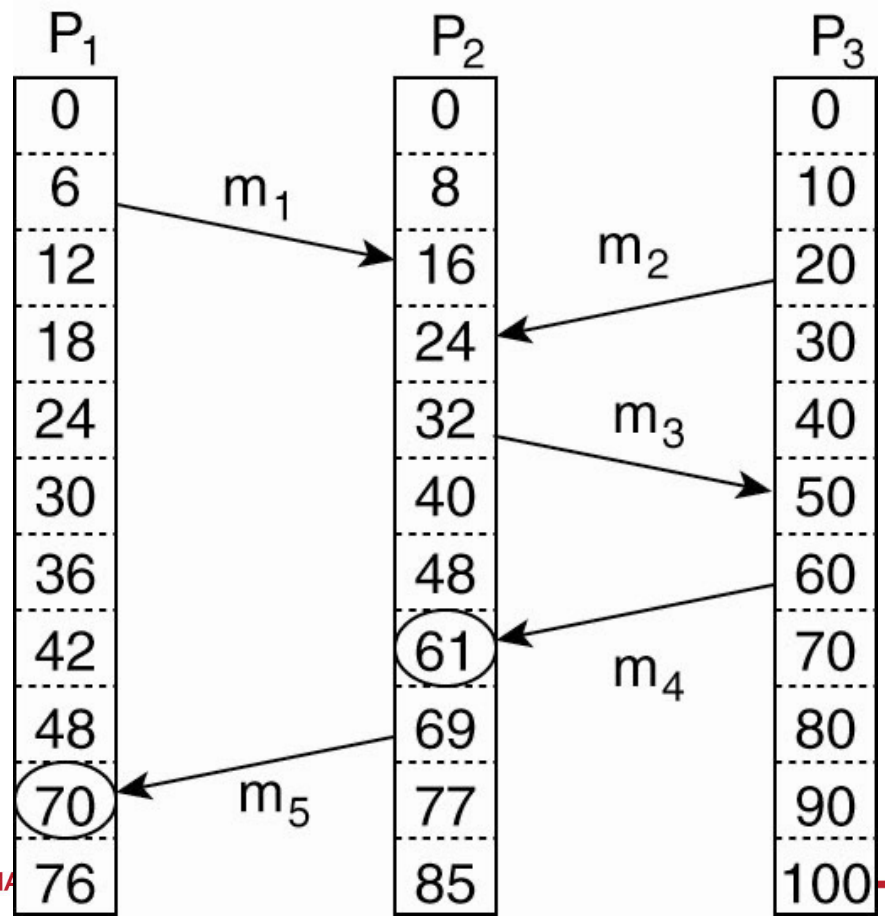


Ứng dụng của đbh đh logic của Lamport: Đảm bảo thứ tự toàn cục của gửi thông điệp theo nhóm (Totally Ordered Multicasting)



2.2. Vector Clocks (1)

- Việc truyền thông điệp có tính tương tranh sử dụng đồng hồ logic



Vector Clocks (2)

- Véc-tơ clock được xây dựng bằng cách để mỗi tiến trình P_i quản lý một véc-tơ VC_i với 2 tính chất sau:

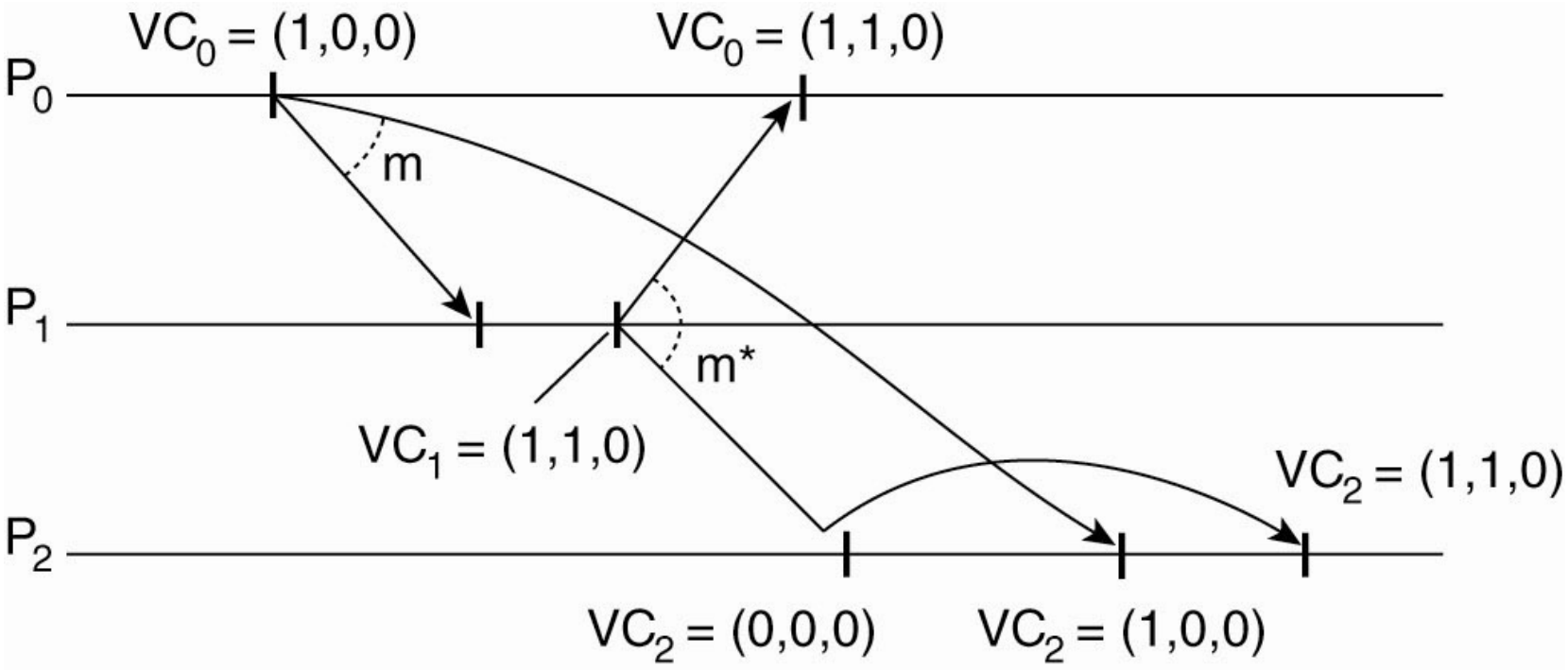
1. VC_i [i] là số các sự kiện đã xảy ra ở P_i. nói cách khác, VC_i [i] là đồng hồ logic cục bộ của P_i .
2. Nếu VC_i [j] = k thì P_i biết rằng đã có k sự kiện xảy ra ở P_j. Là tri thức của P_i về thời gian cục bộ ở P_j .

Vector Clocks (3)

- Những bước cập nhật để đảm bảo tính chất 2:

1. Trước mỗi sự kiện, P_i thực thi:
   $VC_i[i] \leftarrow VC_i[i] + 1$.
2. Khi P_i gửi thông điệp m tới P_j, nó đặt véc-tơ timestamp $ts(m)$ vào m bằng với VC_i sau khi thực hiện bước 1.
3. Khi nhận được 1 thông điệp m, tiến trình P_j cập nhật lại véc-tơ của mình:
   $VC_j[k] \leftarrow \max\{VC_j[k], ts(m)[k]\}$ với mọi k, sau đó mới gửi thông điệp lên tầng ứng dụng.

Thực thi trao đổi thông tin có tính nhân quả



2 điều kiện:

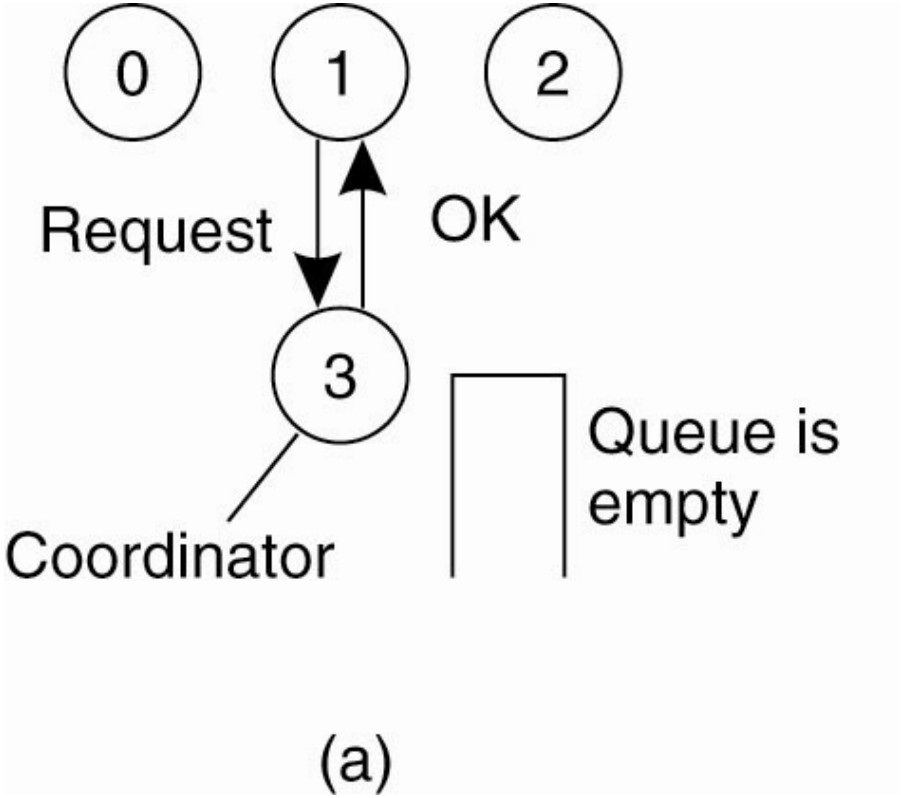
1. $ts(m)[i] = VC_j[i]+1$
2. $ts(m)[k] \leq VC_j[k]$ for all $k \neq i$

3. Các thuật toán loại trừ lẫn nhau

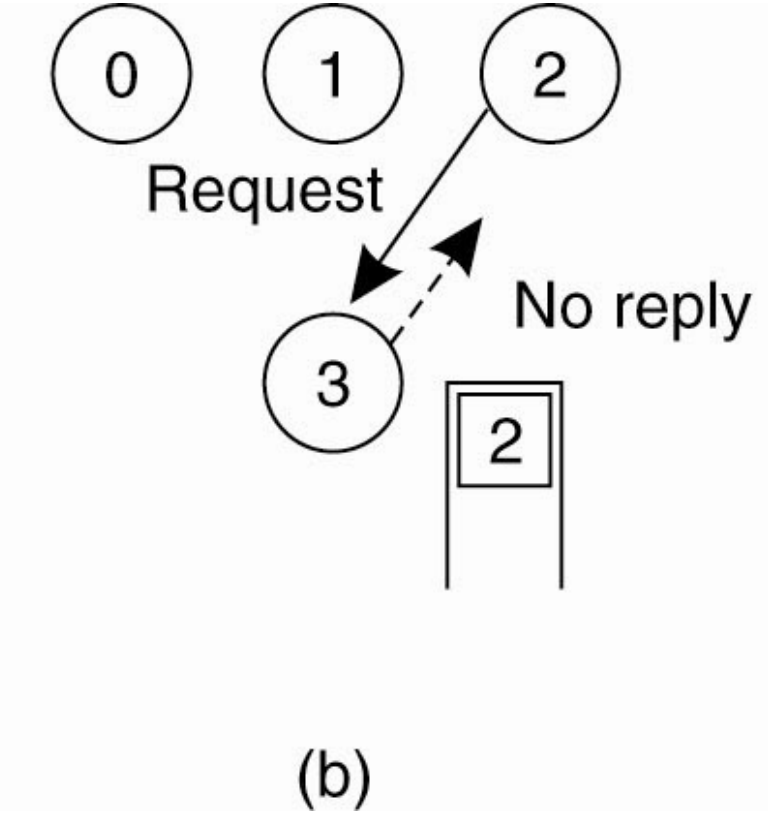
3. Các thuật toán loại trừ lẫn nhau

- Giải thuật tập trung
- Giải thuật phân tán
- Giải thuật Token Ring

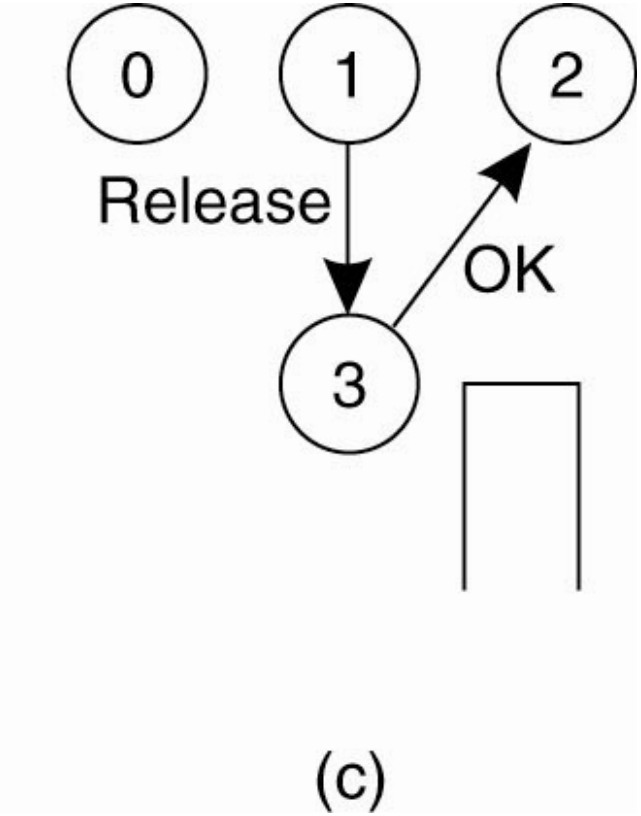
3.1. Giải thuật tập trung (1)



Giải thuật tập trung (2)



Giải thuật tập trung (3)

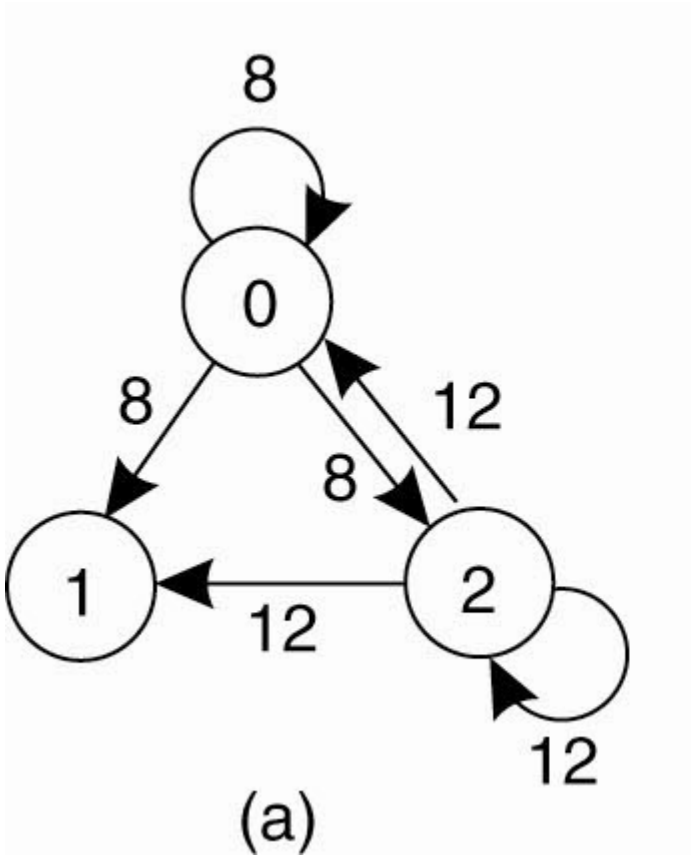


(c)

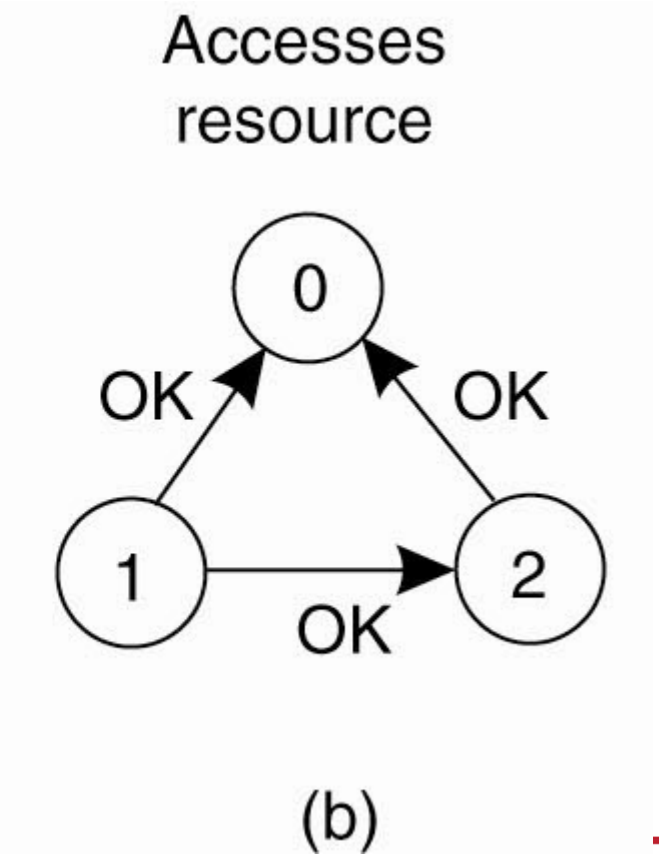
3.2. Giải thuật phân tán (1)

- Muốn vào dùng tài nguyên → quảng bá thông điệp request
- Với mỗi tiến trình nhận được request:
  1. Nếu đang không dùng và không muốn dùng TN: Trả lời OK
  2. Nếu đang dùng: không trả lời và lưu vào hàng đợi
  3. Nếu đang không dùng nhưng cũng muốn dùng: so sánh timestamp
- Chỉ vào dùng TN khi nhận đủ các OK của các tiến trình khác

Giải thuật phân tán (2)

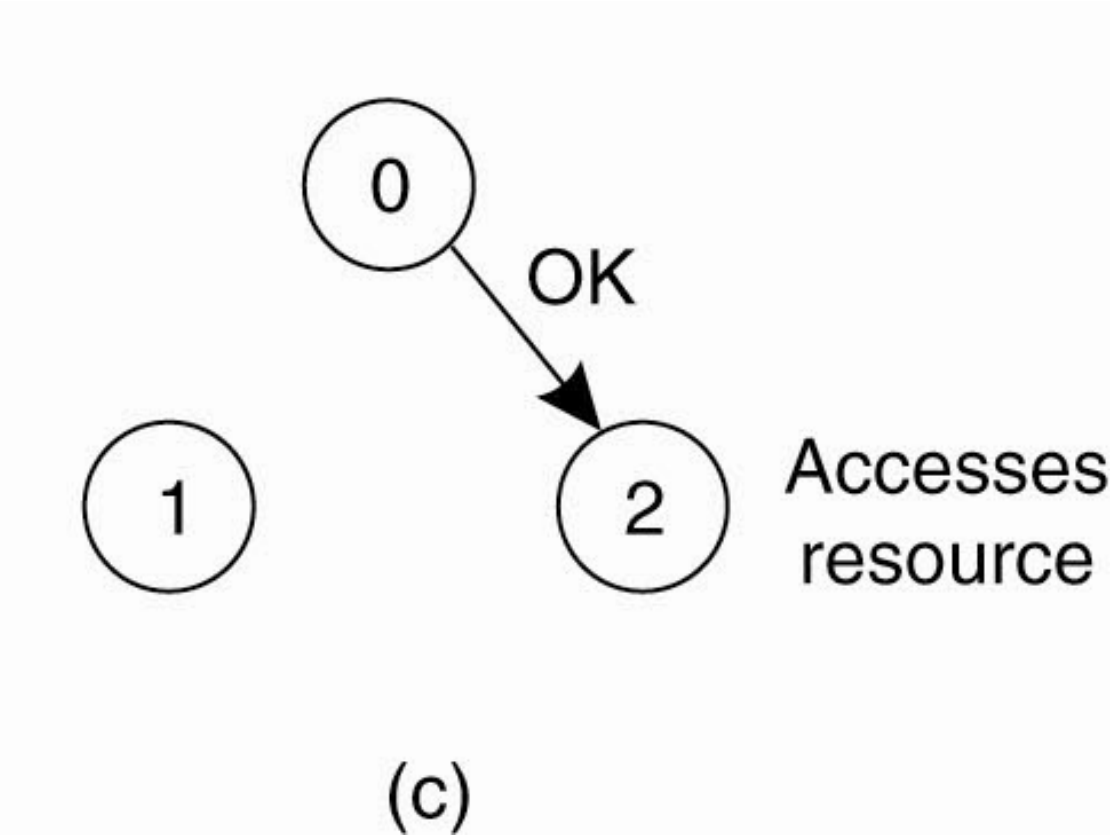


Giải thuật phân tán (3)



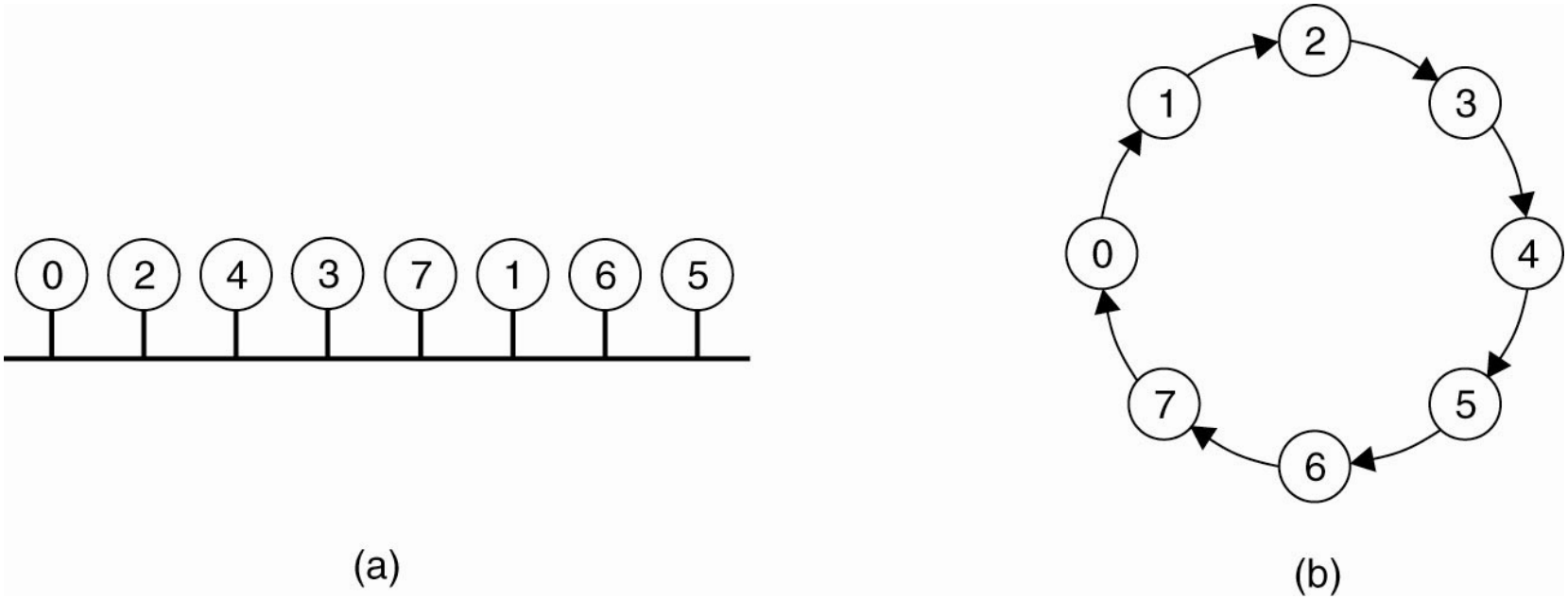
(b)

Giải thuật phân tán (4)



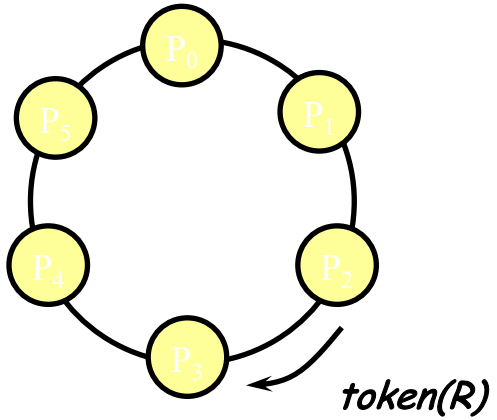
(c)

3.3. Giải thuật Token Ring



Token Ring algorithm

- Khởi đầu
  - Tiến trình P0 có token để vào sử dụng TN
- Token được truyền đi trong vòng topo
  - Từ P_i truyền đến $P_{(i+1)}$mod N
- Khi một tiến trình nhận được token:
  - Tự kiểm tra xem có muốn vào dùng TN không
  - Nếu không, truyền token cho nút kế tiếp
  - Nếu có, giữ lại token. Dùng xong thì truyền tiếp đi



4. Các giải thuật bầu chọn

4. Các giải thuật bầu chọn

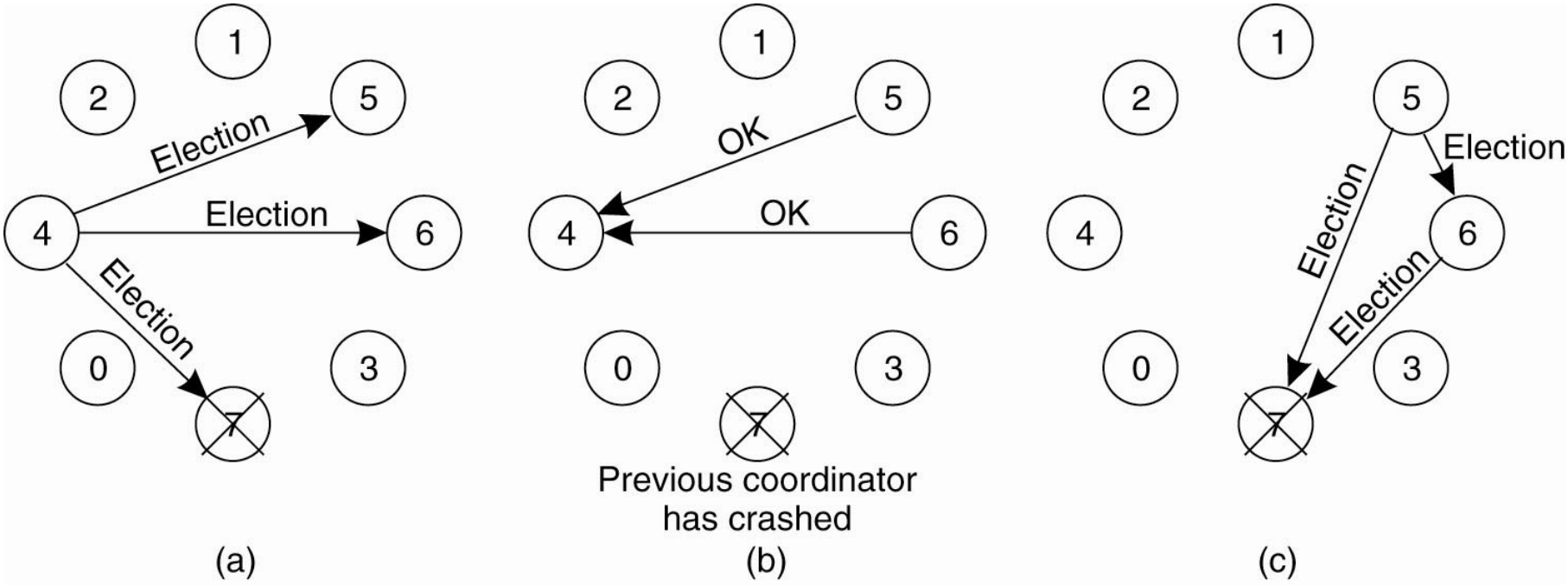
- Các giải thuật truyền thống
  - Giải thuật Bully
  - Giải thuật Ring
- Các giải thuật bầu chọn cho mạng không dây
- Các giải thuật bầu chọn cho mạng diện rộng

Các giải thuật bầu chọn

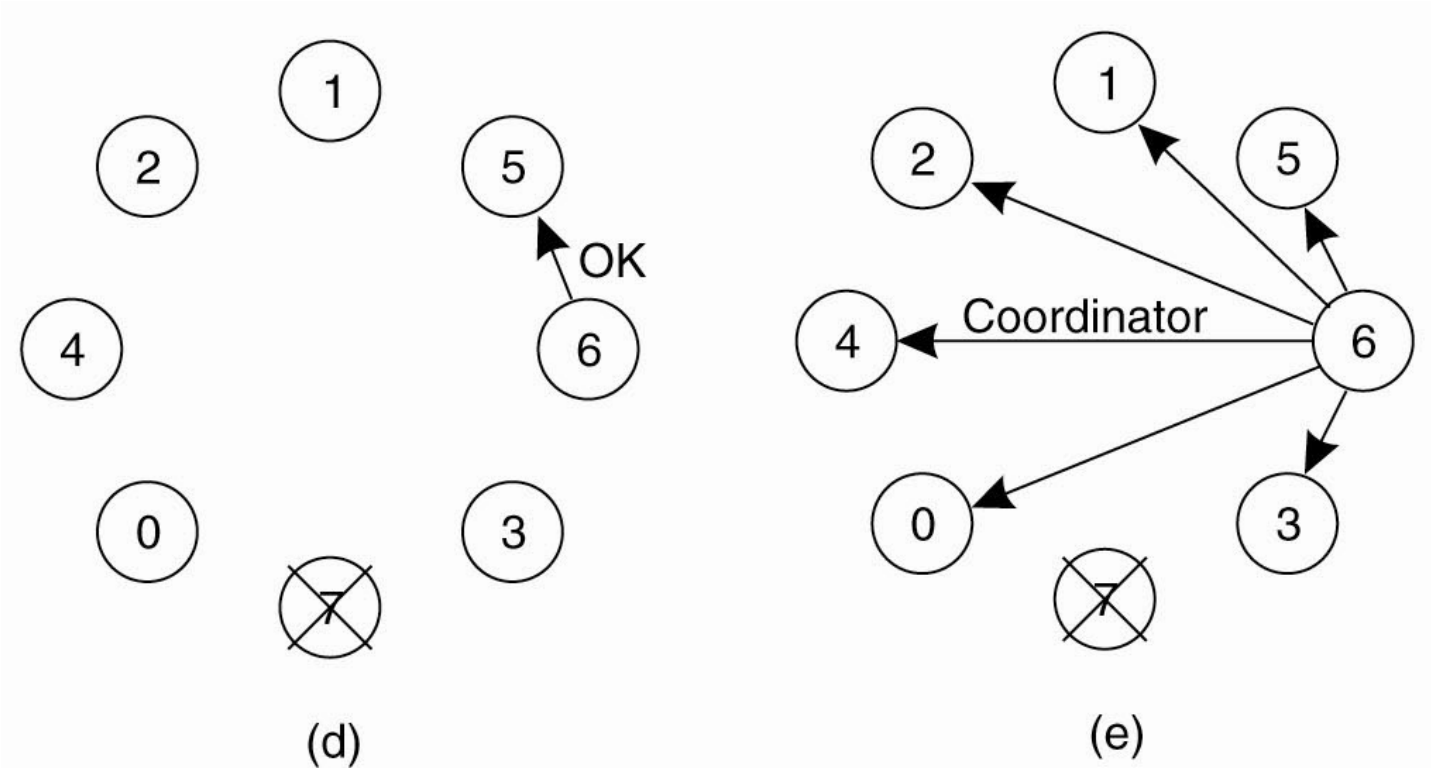
- Giải thuật Bully

1. *P* gửi thông điệp *ELECTION* cho tất cả các tiến trình có ID lớn hơn mình
    1. Nếu không có tiến trình nào trả lời (sau timeout), P biết mình được chọn
    2. Chỉ cần 1 tiến trình trả lời, P biết mình không được chọn và dừng lại.
2. Quá trình trên lặp lại với các tiến trình nhận được ELECTION

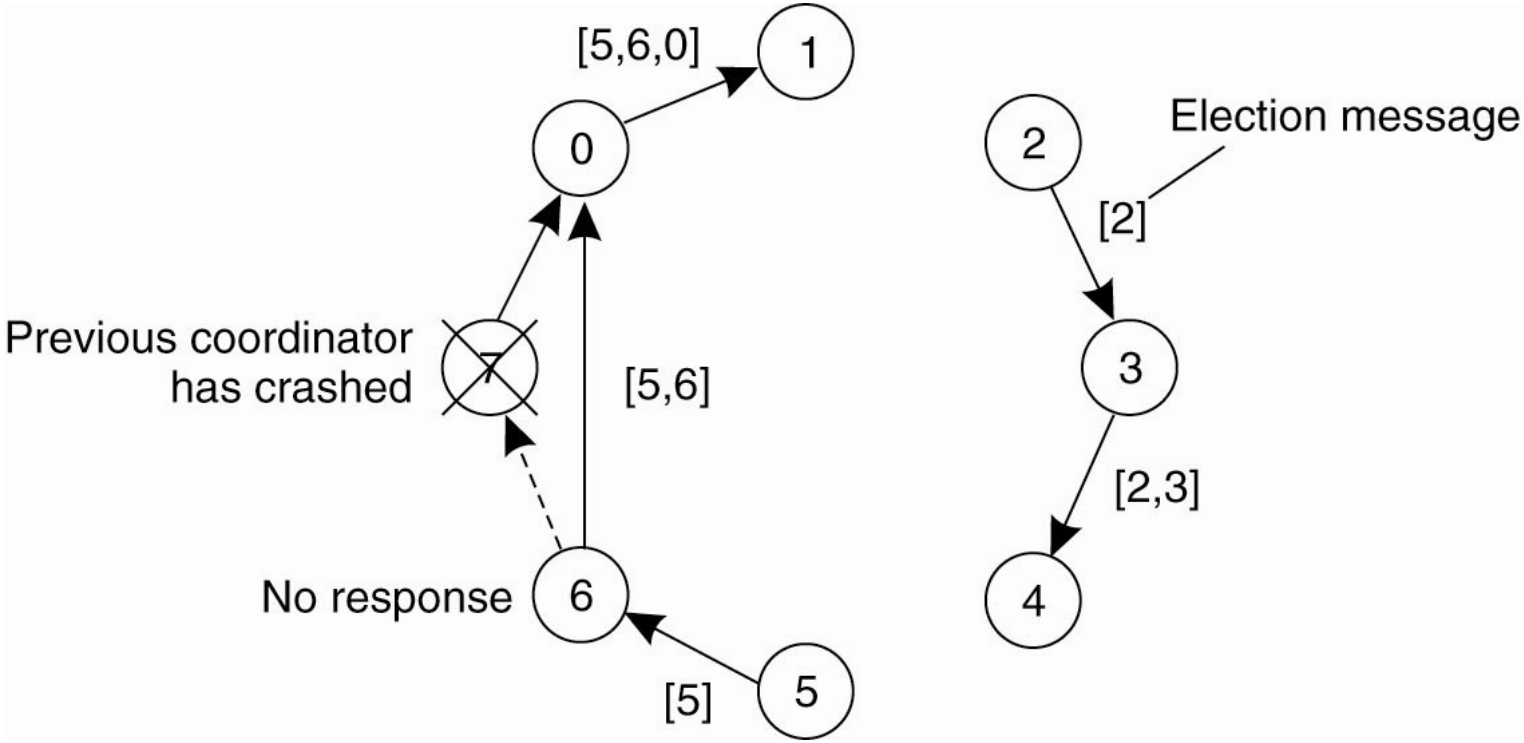
Giải thuật BULLY (1)



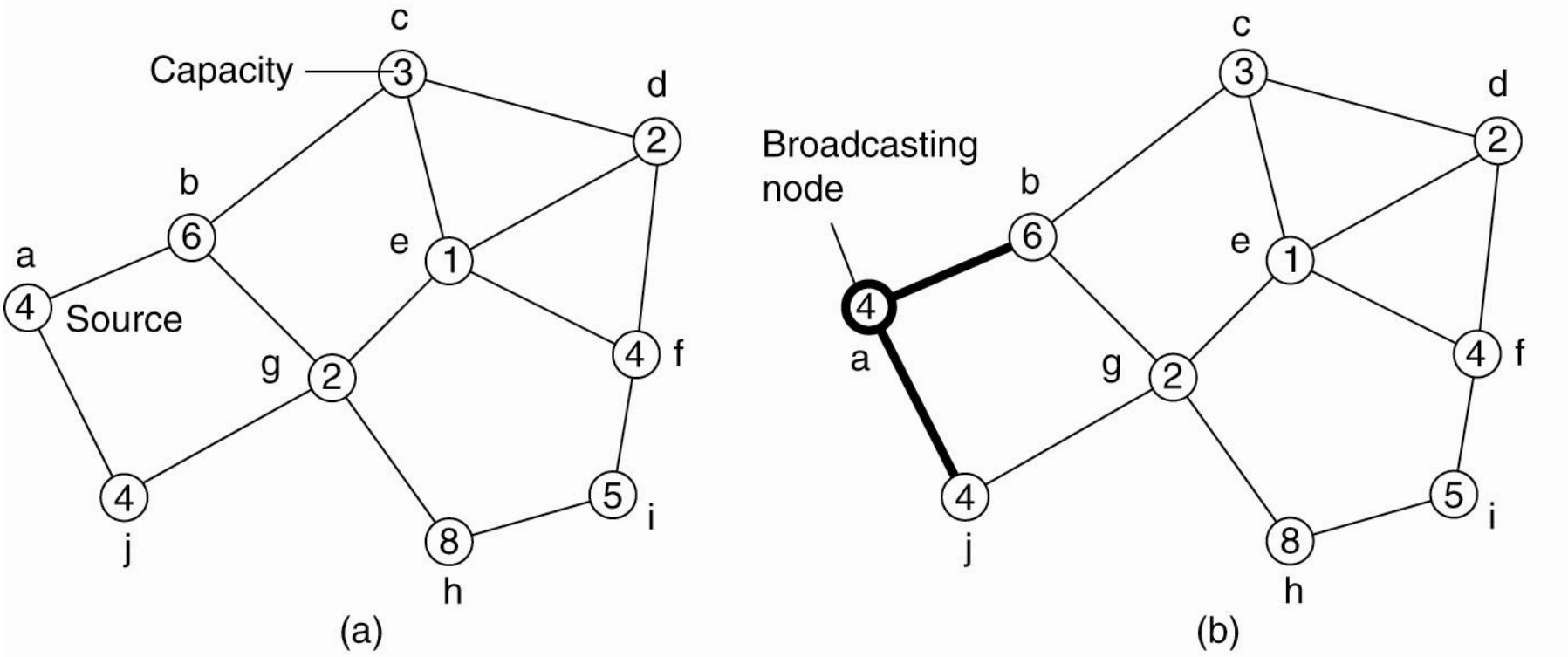
Giải thuật BULLY (2)



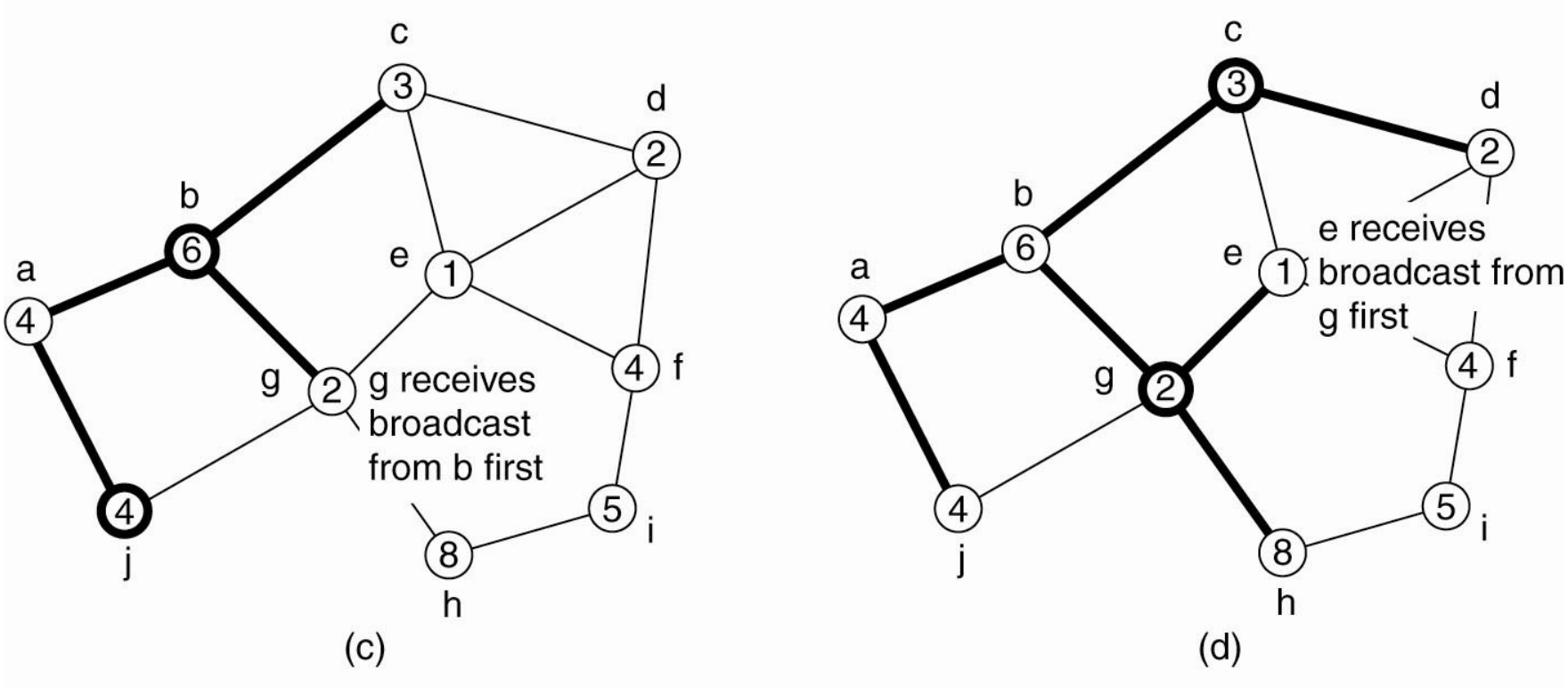
Giải thuật RING



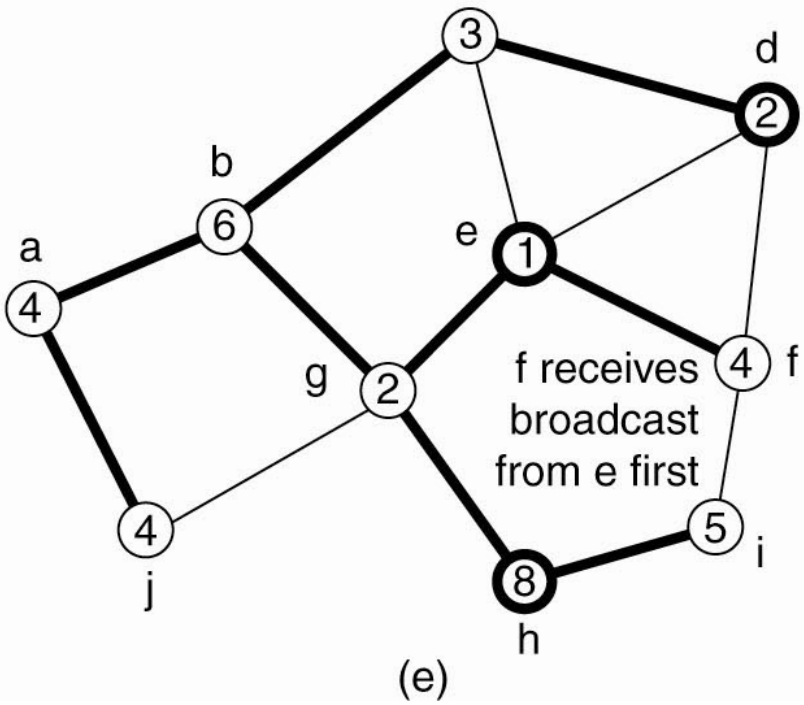
Giải thuật bầu chọn cho mạng không dây (1)

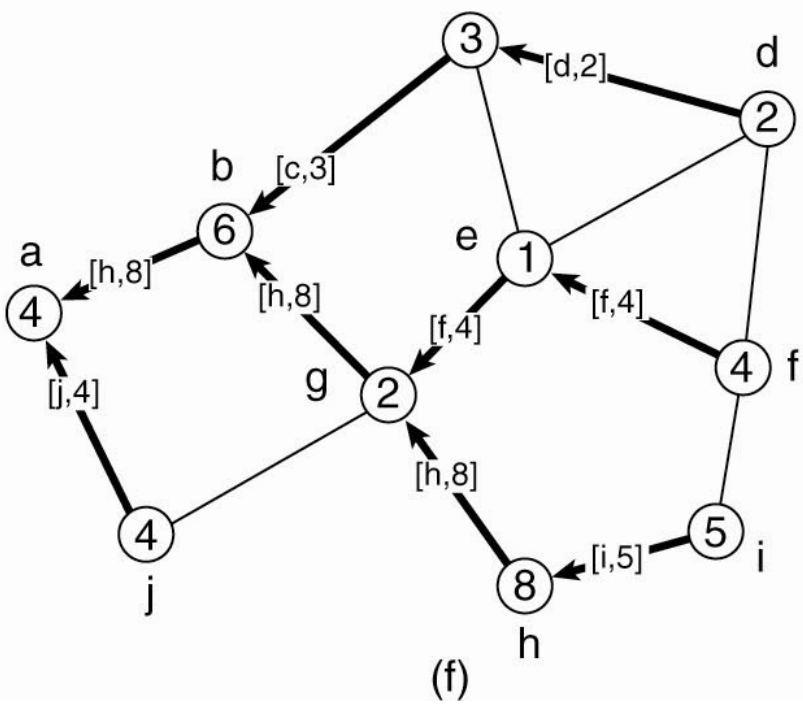


Giải thuật bầu chọn cho mạng không dây (2)



Giải thuật bầu chọn cho mạng không dây (3)



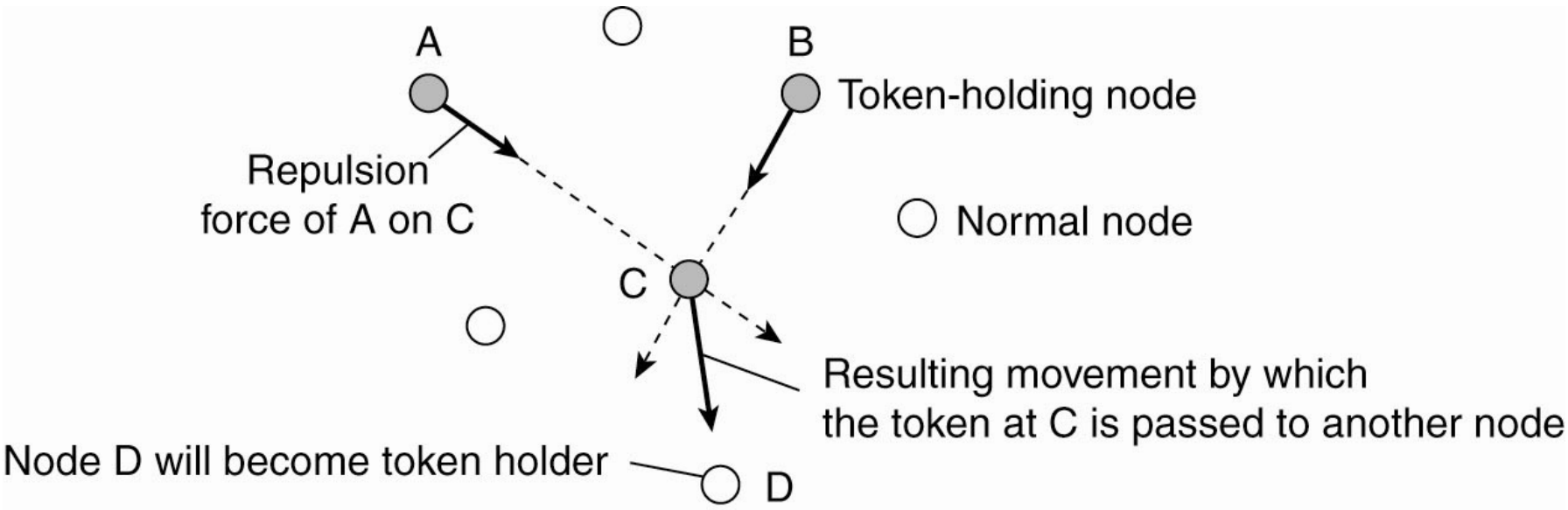


Bầu chọn cho mạng cỡ lớn (1)

- Yêu cầu cho các giải thuật chọn *superpeers*:

1. Các nút thường khi truy cập đến các superpeers phải có độ trễ thấp
2. Các superpeers phải được phân tán đều trong toàn bộ mạng overlay
3. Phải định trước tỷ lệ các nút superpeers trong tổng số các nút trong mạng
4. Mỗi nút superpeers không được phục vụ quá nhiều nút thường

Bầu chọn cho mạng cỡ lớn (2)



VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Câu hỏi?