LẬP TRÌNH ĐỒNG THỜI & PHÂN TÁN

BÀI 8:
BÀI TOÁN BẦU CỬ

Giảng viên: Lê Nguyễn Tuấn Thành
Email: thanhlnt@tlu.edu.vn

NỘI DUNG

- Bài toán bầu cử
- Thuật toán dựa trên vòng tròn
  - Thuật toán Chang-Roberts
  - Thuật toán Hirschberg-Sinclair

Bài giảng có sử dụng hình vẽ trong cuốn sách "***Concurrent and Distributed Computing in Java***, *Vijay K. Garg, University of Texas, John Wiley & Sons, 2005*"

Bài toán bầu cử

- Một bài toán quan trọng trong hệ thống phân tán là *Bài toán bầu cử tiến trình lãnh đạo*
- Tiến trình lãnh đạo có thể được sử dụng như người điều phối trong những thuật toán tập trung cho bài toán mutex

Giao diện Election

```
public interface Election extends MsgHandler {
    void startElection();
    int getLeader(); //blocks till the leader is known
}
```

- Phương thức **getLeader()** trả về định danh của tiến trình được chọn là lãnh đạo
- Nếu định danh của tiến trình lãnh đạo chưa được biết, thì phương thức này sẽ khóa cho đến khi người lãnh đạo được chọn.

Bài toán bầu cử: Giải pháp (1)

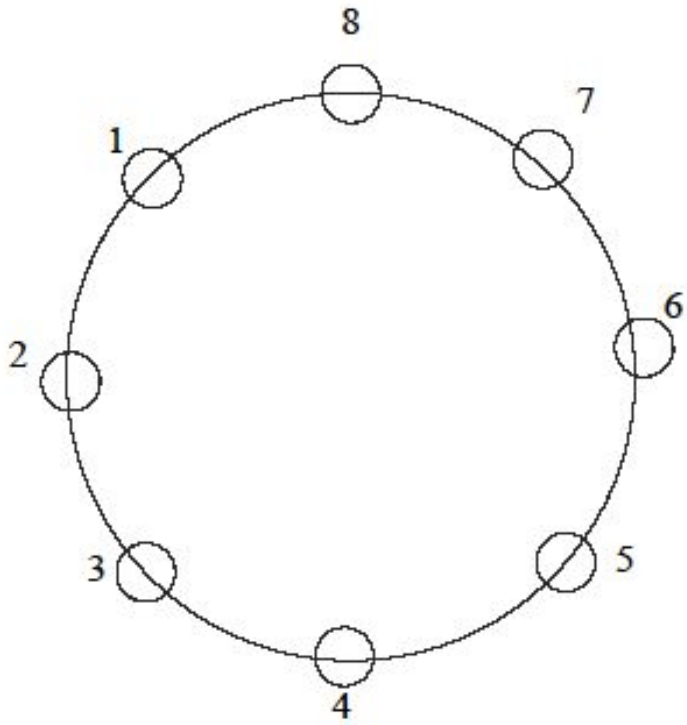
- Bài toán bầu cử người lãnh đạo tương tự như bài toán loại trừ lẫn nhau
  - Trong cả 2 bài toán, chúng ta đều quan tâm đến việc chọn ra một trong số các tiến trình, được gọi là tiến trình đặc quyền
- Các giải pháp dựa trên người điều phối cho bài toán mutex không thể áp dụng cho bài toán bầu cử người lãnh đạo
  - Lý do: việc quyết định tiến trình nào đóng vai trò người điều phối hoặc giữ token là tương đương với bài toán bầu cử người lãnh đạo

Bài toán bầu cử: Giải pháp (2)

- Thuật toán mutex của Lamport có thể áp dụng cho bài toán bầu cử
  - Tiến trình đầu tiên đi vào CS được xem là người lãnh đạo
- Tuy nhiên thuật toán này không tổng quát, do:
  - Nền tảng giao tiếp mạng phía dưới phải kết nối hoàn toàn
  - Các thông điệp phải truyền đi theo thứ tự FIFO
  - Mỗi tiến trình phải giao tiếp với mọi tiến trình khác

Bài toán bầu cử: Giải pháp (3)

Trong trường hợp các tiến trình trong hệ thống phân tán được sắp xếp theo hình tròn, tồn tại các thuật toán hiệu quả hơn của bài toán bầu cử



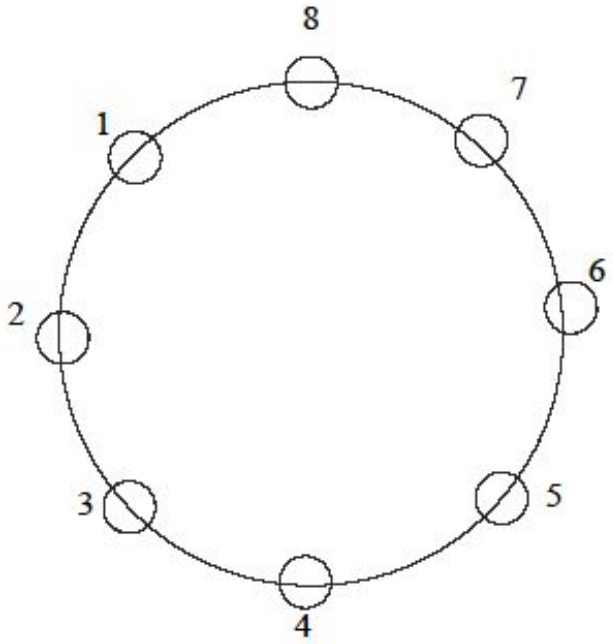
Thuật toán dựa trên vòng tròn

Ring-based algorithms

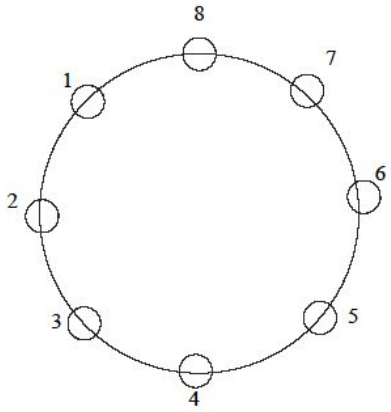
i < j : xoá
i > j : chuyển tiếp
i = j : được chọn

Yêu cầu

- Các thuật toán cho bài toán bầu cử người lãnh đạo giả định rằng các tiến trình có định danh duy nhất



Thuật toán của Chang-Roberts



- Đảm bảo rằng tiến trình với ***định danh lớn nhất*** được lựa chọn là tiến trình lãnh đạo
- Mỗi tiến trình:
  - Chỉ **gửi** thông điệp cho hàng xóm **bên trái** của nó và
  - Chỉ **nhận** thông điệp từ hàng xóm **bên phải**
- Các tiến trình không biết được tổng số tiến trình trong hệ thống, cũng như không biết định danh của các tiến trình khác

Các bước thực hiện (1)

1. Mỗi tiến trình P_i gửi thông điệp *tự ứng cử (election)* cùng với định danh của nó cho tiến trình bên trái
   - Nếu nó chưa nhận được thông điệp nào từ một tiến trình với định danh cao hơn
2. Khi P_i nhận được thông điệp *election* từ tiến trình bên phải:
   - Tiến trình chuyển tiếp các thông điệp với id lớn hơn id của nó sang bên trái
   - Ngược lại (e.g id của thông điệp nhỏ hơn id của nó), tiến trình nuốt thông điệp đó

Các bước thực hiện (2)

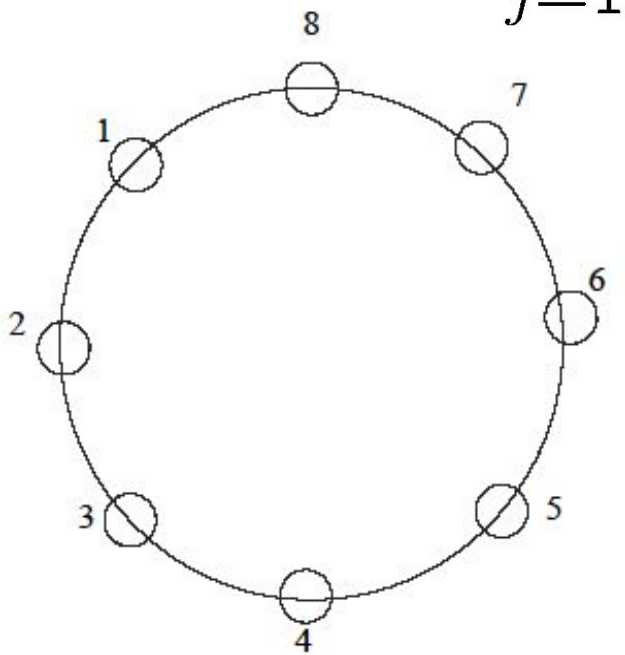
3. Nếu tiến trình P_k nhận lại được thông điệp *tự ứng cử* (*election*) của mình
   - Nó sẽ biết bản thân mình là tiến trình lãnh đạo và thông báo bằng cách gửi thông điệp *leader* tới tất cả tiến trình khác
4. Khi tiến trình P_k nhận lại thông điệp *leader* của mình
   - Nó hiểu rằng tất cả tiến trình khác đều đã biết nó là tiến trình lãnh đạo

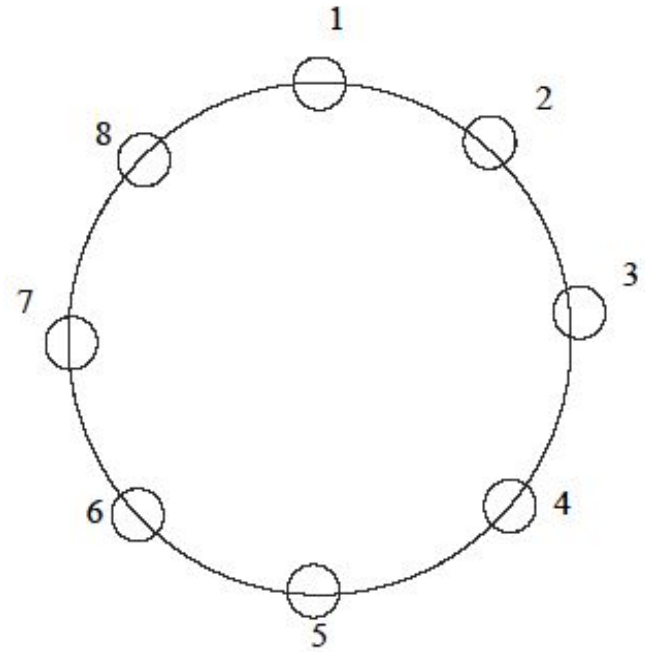
```
public class RingLeader extends Process implements Election {
    int number; int leaderId = -1; int next;
    boolean awake = false;
    …
    public synchronized int getLeader(){
        while (leaderId == -1) myWait();
        return leaderId;
    }
    public synchronized void handleMsg(Msg m, int src, String tag) {
        int j = m.getMessageInt(); // get the number
        if (tag.equals("election")) {
            if (j > number)
                sendMsg(next, "election", j); // forward the message
            else if (j == number) // I won!
                sendMsg(next, "leader", myId);
            else if ((j < number) && !awake) startElection();
        } else if (tag.equals("leader")) {
            leaderId = j;
            notify();
            if (j != myId) sendMsg(next, "leader", j);
        }
    }
    public synchronized void startElection() {
        awake = true;
        sendMsg(next, "election", number);
    }
}
```

Đánh giá: trường hợp tồi nhất & tốt nhất

Độ phức tạp thông điệp trong trường hợp xấu nhất

$$\sum_{j=1}^{j=N} j = O(N^2)$$





Độ phức tạp trung bình **O(N log**

Thuật toán của Hirschberg-Sinclair (1)

2 chiều

- Giả sử việc truyền thông trên vòng tròn theo 2 chiều
  - Có thể gửi thông điệp sang bên trái hoặc bên phải
- **Ý tưởng**: thực hiện việc bầu cử trên một tập con tiến trình với số lượng tăng dần sau các vòng bầu cử
  - Một tiến trình P_i cố gắng bầu cho nó ở vòng *r*
  - Chỉ những tiến trình thắng ở vòng *r* mới có thể tiến vào vòng *r+1*

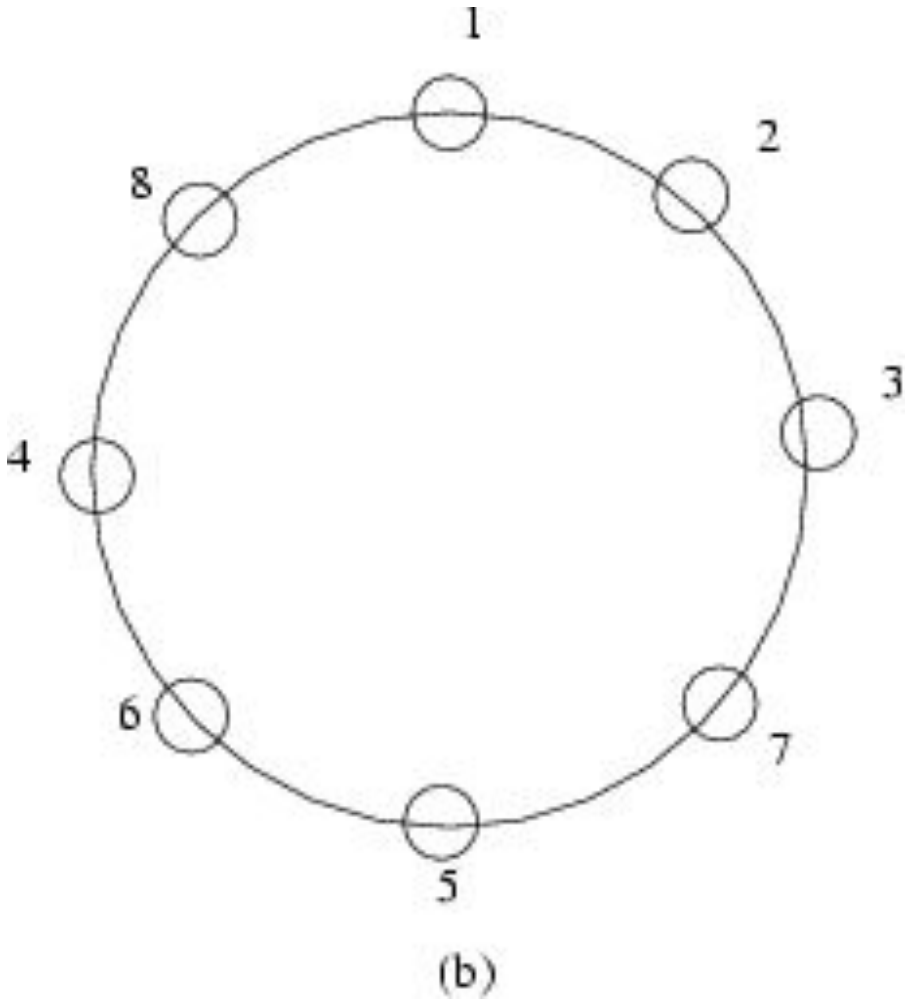
Thuật toán của Hirschberg-Sinclair (2)

- Tiến trình P_i là 1 tiến trình lãnh đạo ở vòng *r* khi và chỉ khi P_i có định danh lớn nhất trong tất cả các tiến trình với khoảng cách nhỏ hơn hoặc bằng 2^r tính từ P_i
- Hai tiến trình được bầu là lãnh đạo sau vòng r phải có khoảng cách ít nhất là 2^r
- Sau vòng r, có nhiều nhất $N/(2^{r-1} + 1)$ tiến trình lãnh đạo
- Sau mỗi vòng, số lượng tiến trình lãnh đạo giảm đi, và sau O(log N) vòng chỉ còn chính xác 1 tiến trình lãnh đạo
- Ở mỗi vòng có nhiều nhất O(N) thông điệp, do đó tổng số thông điệp là O(N log N)

thuật toán này chạy nhanh hơn thuật toán trên

Ví dụ

- Khởi đầu:
  - Tất cả tiến trình đều là người lãnh đạo
- Vòng 0:
  - 6, 7, và 8 là người lãnh đạo
- Vòng 1:
  - 7, 8 là người lãnh đạo
- Vòng 2:
  - 8 là người lãnh đạo duy nhất
- Nhiều nhất **log(N)** vòng



(b)

Thuật toán dừng

Tài liệu tham khảo

- *Concurrent and Distributed Computing in Java*, Vijay K. Garg, University of Texas, John Wiley & Sons, 2005
- Tham khảo:
  - *Principles of Concurrent and Distributed Programming*, M. Ben-Ari, Second edition, 2006
  - *Foundations of Multithreaded, Parallel, and Distributed Programming*, Gregory R. Andrews, University of Arizona, Addison-Wesley, 2000
  - *The SR Programming Language: Concurrency in Practice*, Benjamin/Cummings, 1993
  - *Xử lý song song và phân tán*, Đoàn văn Ban, Nguyễn Mậu Hân, Nhà xuất bản Khoa học và Kỹ thuật, 2009