CHƯƠNG II

CÁC VẤN ĐỀ VÀ GIẢI PHÁP CƠ BẢN TRONG CÁC HỆ PHÂN TÁN

NỘI DUNG

- Truyền thông
- Định danh
- Đồng bộ
- Tiến trình trong các hệ thống phân tán
- Quản trị giao dịch và điều khiển tương tranh
- Phục hồi và chịu lỗi
- Bảo mật
- Tính nhất quán và vấn đề nhân bản

PHỤC HỒI VÀ CHỊU LỖI

NỘI DUNG

- Giới thiệu về tính chịu lỗi (Fault tolerance)
- Các phương pháp che giấu lỗi
  - Phương pháp dư thừa
  - Tiến trình bền bỉ
  - Truyền thông khách – chủ tin cậy
  - Truyền thông theo nhóm tin cậy
- Khẳng định, cam kết (COMMIT) phân tán
- Phục hồi

QUAN ĐIỂM CƠ BẢN VỀ TÍNH CHỊU LỖI

- Tính chịu lỗi liên quan tới hệ thống đáng tin cậy.
- Hệ thống đáng tin cậy thể hiện trong các khía cạnh sau:
  - ***Tính sẵn sàng*** (Availability): luôn sẵn sàng thực thi tốt khi có yêu cầu gửi đến.
  - ***Tính tin cậy*** (Reliability): có khả năng hoạt động trong một thời gian dài mà không bị gián đoạn, không xảy ra lỗi.
  - ***Tính an toàn*** (Safety): khi xảy ra lỗi cũng không dẫn tới sai lệch (thông tin, điều khiển).
  - ***Khả năng bảo trì*** (Maintainability): Dễ dàng sửa chữa khi có lỗi, có khả năng phục hồi lại được sau khi có lỗi. Nếu sự phục hồi này diễn ra tự động thì có thể nói hệ thống này cũng có tính sẵn sàng cao.
- Tính chịu lỗi còn có liên quan tới khái niệm ***điều khiển lỗi (Fault control)***. Điều khiển lỗi bao gồm ngăn ngừa , loại bỏ và dự báo lỗi .

PHÂN LOẠI LỖI

- **Lỗi nhất thời (Transient faults)**: lỗi xuất hiện một lần rồi biến mất. Để khắc phục thì chỉ cần thực hiện lại hoạt động gây nên lỗi này.
- **Lỗi chập chờn (Intermittent faults):** lỗi xuất hiện rồi biến mất, sau đó lại xuất hiện lại và cứ tiếp tục như vậy. Lỗi này thường gây ra các hậu quả nghiêm trọng vì rất khó xác định nguyên nhân gây lỗi. Để khắc phục cần phải dò tìm và phân tích.
- **Lỗi thường xuyên (Permanent faults)**: Là loại lỗi vẫn tồn tại ngay cả khi đã xác định được nguyên nhân và sửa thành phần gây lỗi.

CÁC MÔ HÌNH LỖI

- **Lỗi sụp đổ (crash failure):** máy chủ bất ngờ bị treo mặc dù trước đó vẫn hoạt động bình thường, cách duy nhất là khởi động lại.
- **Lỗi bỏ sót (omission failure):** máy chủ không thể đáp ứng được yêu cầu gửi tới, gồm hai loại:
  - Lỗi nhận thông điệp: máy chủ không nhận được thông điệp từ máy khách. Lỗi này không ảnh hưởng đến trạng thái của máy chủ.
  - Lỗi gửi thông điệp: máy chủ vẫn nhận được yêu cầu và hoàn thành yêu cầu đó nhưng không thể gửi kết quả tới máy khách. Thường máy khách sẽ gửi lại thông điệp, do đó máy chủ cần phải xử lý tránh trùng lặp.
- **Lỗi thời gian (timing failure):** máy chủ phản hồi kết quả chậm hơn thời gian cho phép.

CÁC MÔ HÌNH LỖI

- **Lỗi đáp ứng (Response failure)**: Máy chủ trả về giá trị không chính xác. Đây là kiểu lỗi rất nghiêm trọng, phân thành hai loại:
  - Lỗi giá trị: Giá trị trả về không chính xác.
  - Lỗi chuyển trạng thái: Tiến trình máy chủ hoạt động không đúng với luồng điều khiển, kết quả trả về ngoài ý muốn.
- **Lỗi phức tạp (Arbitrary failure, Byzantine)**: có thể xảy ra ở bất kì thời gian nào. Đây là loại lỗi nguy hiểm nhất, tiến trình máy chủ tạo ra kết quả sai mà không thể phát hiện ra được hoặc kết quả sai khi tiến trình máy chủ liên kết với các tiến trình máy chủ khác. Thường thể hiện ở ba dạng:
  - fail-stop: Máy chủ không đưa ra kết quả và các tiến trình khác có thể phát hiện ra điều này. Trường hợp tốt nhất tiến trình máy chủ sẽ thông báo về sự cố, nếu không sẽ chỉ đơn giản ngừng hoạt động
  - fail-silent: máy chủ đột ngột hoạt động chậm lại các tiến trình không thể biết được máy chủ còn hoạt động hay không, ảnh hưởng đến hiệu năng của hệ thống.
  - fail-safe: Máy chủ trả về kết quả ngẫu nhiên để các tiến trình khác nhận dạng nhưng không có giá trị sử dụng.

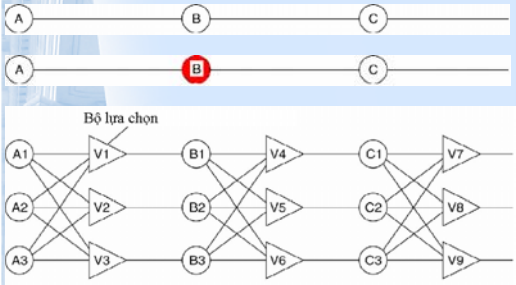
CÁC BIỆN PHÁP CHE GIẤU LỖI

- Phương pháp dư thừa
- Tiến trình bền bỉ
- Truyền thông khách – chủ tin cậy
- Truyền thông theo nhóm tin cậy

CHE GIẤU LỖI BẰNG BIỆN PHÁP DƯ THỪA

- Nếu là hệ thống chịu lỗi thì biện pháp tốt nhất là che giấu lỗi đối với các tiến trình khác
- **Dư thừa thông tin :** bổ sung thêm các bit dư thừa để phát hiện lỗi và phục hồi lỗi. Ví dụ trong việc truyền dữ liệu thường thêm vào các bit kiểm tra chẵn lẻ, mã Haming, CRC… để phát hiện lỗi và phục hồi lỗi.
- **Dư thừa thời gian**: khi một hoạt động đã được thực hiện, nếu dư thừa thời gian nó có thể được thực hiện lại. Kĩ thuật dư thừa thời gian phù hợp khi lỗi là nhất thời và lỗi chập chờn. Có thể sửa lỗi bằng cách thực hiện lại các giao tác bị lỗi. ***Tuy nhiên, cần phải chú ý tính chính xác đối với các thao tác thực hiện theo thời gian thực***.
- **Dư thừa vật lý**: bổ sung thêm tài nguyên vật lý

PHƯƠNG PHÁP DƯ THỪA CHUYỂN MÔ ĐUN (TRM – Triple Modular Redundancy)

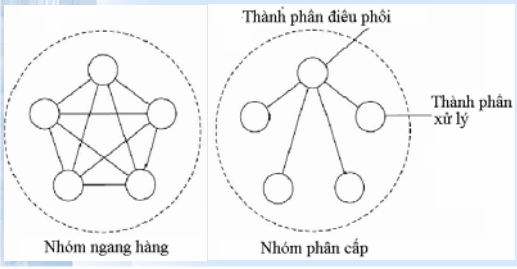


TIẾN TRÌNH BỀN BỈ (Resilent)

- Cách tiếp cận mấu chốt là gộp các tiến trình giống nhau vào cùng một nhóm.
- Thuộc tính mấu chốt của nhóm là khi nhóm nhận được thông báo thì thông báo này sẽ được gửi tới tất cả các thành viên trong nhóm. Nếu có tiến trình nào trong nhóm bị lỗi thì hy vọng sẽ có các tiến trình khác thay thế.
- Nhóm các tiến trình có thể động, tính động thể hiện ở các mặt sau:
  - Có thể tạo thêm hay hủy bỏ một nhóm.
  - Một tiến trình có thể gia nhập hay rời khỏi nhóm.
  - Một tiến trình có thể là thành viên của nhiều nhóm trong cùng thời một điểm.
- Do tính động của nhóm nên cần phải đưa ra các cơ chế quản lý mối quan hệ giữa các nhóm và các thành viên trong nhóm.

PHÂN LOẠI NHÓM TIẾN TRÌNH BỀN BỈ

Dựa trên tổ chức bên trong, nhóm phân làm hai loại: Nhóm ngang hàng (flat) và nhóm phân cấp



NHÓM TIẾN TRÌNH BỀN BỈ NGANG HÀNG

- Tất cả các tiến trình trong nhóm là ngang hàng nhau.
- Khi thực hiện một công việc nào đó sẽ phải có một quá trình bầu cử để xác định xem tiến trình nào phù hợp để thực hiện công việc đó.
- Ưu điểm: khi một tiến trình bị lỗi thì chỉ làm cho kích thước của nhóm giảm đi chứ không ảnh hưởng đến hoạt động của cả nhóm.
- Nhược điểm: do phải có quá trình bầu cử nên tốn thời gian.

NHÓM TIẾN TRÌNH BỀN BỈ PHÂN CẤP

- Trong mỗi nhóm sẽ có một tiến trình giữ vai trò điều phối, còn các tiến trình khác đóng vai trò xử lý. Các tiến trình xử lý chịu sự điều khiển của tiến trình điều phối.
- Khi có yêu cầu gửi đến nhóm, yêu cầu này sẽ được gửi tới tiến trình điều phối. Tiến trình điều phối sẽ quyết định xem tiến trình nào trong nhóm đảm nhiệm công việc đó một cách phù hợp nhất và chuyển yêu cầu nhận được đến tiến trình đó.
- Ưu điểm: không bị trễ như kiến trúc ngang hàng.
- Nhược điểm: khi tiến trình điều phối gặp sự cố thì toàn bộ hoạt động của nhóm sẽ bị dừng lại.

QUẢN LÝ THÀNH VIÊN TRONG NHÓM

- Phương pháp 1(dùng máy chủ nhóm): Máy chủ này chứa tất cả các thông tin về các nhóm và các thành viên của từng nhóm.
  - Ưu điểm: hiệu quả, dễ sử dụng
  - Nhược điểm: nếu máy chủ bị lỗi thì không thể quản lý được toàn bộ hệ thống và các nhóm có thể phải xây dựng lại từ đầu các công việc mình đã thực hiện.
- Phương pháp 2 (phương pháp phân tán): Khi tiến trình muốn gia nhập hay rời khỏi nhóm thì nó phải gửi bản tin thông báo tới tất cả các tiến trình khác.
  - Phương pháp này không phù hợp với dạng Fail-Stop.
  - Vấn đề phức tạp khác là việc gia nhập hoặc rời khỏi nhóm được đồng bộ với thông điệp dữ liệu đang gửi.
    - Khi một tiến trình gia nhập nhóm nó sẽ nhận tất cả các thông điệp từ nhóm đó.
    - Khi một tiến trình rời khỏi nhóm thì nó sẽ không được nhận bất kì thông điệp nào từ nhóm đó nữa và không một thành viên nào của nhóm cũ nhận được các thông điệp từ nó.
    - Để giải quyết vấn đề trên cần phải lập thứ tự các thông điệp

CHE GIẤU LỖI VÀ NHÂN BẢN

- Sử dụng nhóm tiến trình cũng là một trong các biện pháp che giấu lỗi. Nhóm các tiến trình giống nhau sẽ che giấu một tiến trình nào đó bị lỗi.
- Nhân bản một tiến trình sau đó gộp chúng thành một nhóm. Có hai phương pháp nhân bản: giao thức dựa trên thành phần chính (primary-based protocol) và giao thức dựa trên ghi nhân bản (replicated-write protocol)
- **Dựa trên thành phần chính** : Các tiến trình trong nhóm tổ chức theo mô hình phân cấp. Một tiến trình đóng vai trò tiến trình chính có nhiệm vụ điều phối tất cả các thao tác ghi. Nếu tiến trình chính của nhóm dừng hoạt động thì các tiến trình sao lưu sẽ thực hiện giải thuật bầu chọn để lựa chọn tiến trình chính mới.
- **Dựa trên ghi nhân bản**: Các tiến trình trong nhóm tổ chức theo mô hình nhóm ngang hàng.Vấn đề là cần nhân bản với số lượng là bao nhiêu?

THỎA THUẬN TRONG CÁC HỆ THỐNG LỖI

- Vấn đề trở nên phức tạp nếu muốn các tiến trình trong nhóm đạt được thỏa thuận trong một số trường hợp: bầu cử, COMMIT, phân chia nhiệm vụ xử lý…
- Mục đích chung của thuật toán thỏa thuận phân tán là để tất cả các tiến trình lỗi đạt được sự đồng thuận trên một số vấn đề và thiết lập sự đồng thuận đó trong một số bước hữu hạn, tuy nhiên thực tế rất phức tạp và cần phải phân biệt các trường hợp sau:
  - Các hệ thống đồng bộ đối với các hệ thống không đồng bộ
  - Có ràng buộc độ trễ truyền thông hay không, chỉ có thể ràng buộc nếu thông điệp được phân phát trong một khoảng thời gian xác định trước.
  - Phân phát thông điệp theo thứ tự hay không? Chỉ có thể coi là tuần tự nếu thứ tự bên gửi và bên nhận giống nhau.
  - Truyền thông điệp được thực hiện qua cơ chế Unicast hay Multicast

CÁC TÌNH HUỐNG CÓ THỂ ĐẠT ĐƯỢC THỎA THUẬN PHÂN TÁN

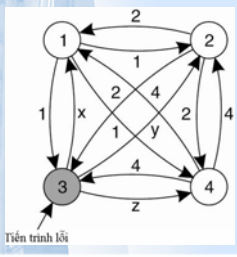
Hành vi tiến trình xử lý		Thứ tự thông điệp: Không – Unicast	Không – Multicast	Có – Unicast	Có – Multicast		Độ trễ truyền thông
	Đồng bộ			X		Ràng buộc	
				X			
	Không đồng bộ	X	X	X	X	Không ràng buộc	
				X	X		
Loại truyền thông điệp							

VẤN ĐỀ THỎA THUẬN Byzantine

- Hệ thống gồm N tiến trình, mỗi tiến trình i cung cấp giá trị Vi cho các tiến trình khác
- Mục tiêu cần đạt: Cho phép mỗi tiến trình xây dựng vector V[N] với V[i]=Vi nếu tiến trình i không lỗi và V[i] không xác định nếu tiến trình i bị lỗi.
- Bước 1: Mỗi tiến trình i không lỗi gửi giá trị Vi cho các tiến trình khác, tiến trình lỗi gửi giá trị bất kỳ
- Bước 2: Kết quả nhận được từ bước 1 sẽ tập hợp lại thành vector
- Bước 3: Mỗi tiến trình gửi Vector bước 2 cho các tiến trình khác
- Bước 4: Mỗi tiến trình kiểm tra phần tử thứ I trong các Vector nhận được ở bước 3, nếu kết quả kiểm tra chiếm đa số thì đặt giá trị vào Vector kết quả thỏa thuận, nếu không thì đặt giá trị UNKNOWN.

VẤN ĐỀ THỎA THUẬN Byzantine

- Giả thiết có 04 tiến trình đồng bộ, truyền thông điệp Unicast theo thứ tự và ràng buộc thời gian trễ
- Tiến trình số 3 bị lỗi



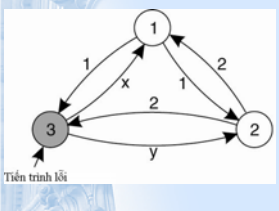
Kết quả bước 2	
Tiến trình	Vector
1	(1,2,x,4)
2	(1,2,y,4)
3	(1,2,3,4)
4	(1,2,z,4)

VẤN ĐỀ THỎA THUẬN Byzantine

Kết quả bước 3		
Tiến trình 1	Tiến trình 2	Tiến trình 4
(1,2,y,4)	(1,2,x,4)	(1,2,x,4)
(a,b,c,d)	(e,f,g,h)	(1,2,y,4)
(1,2,z,4)	(1,2,z,4)	(i,j,k,l)

Kết quả bước 4		
Tiến trình 1	Tiến trình 2	Tiến trình 4
(1,2,Unknow,4)	(1,2,Unknow,4)	(1,2,Unknow,4)

VẤN ĐỀ THỎA THUẬN Byzantine



Tiến trình	Vector
1	(1,2,x)
2	(1,2,y)
3	(1,2,3)

Tiến trình 1	Tiến trình 2
(1,2,y)	(1,2,x)
(a,b,c)	(d,e,f)

Không có tiến trình chiếm đa số, thất bại trong thỏa thuận!

Tiến trình 1	Tiến trình 2
(Unknow,Unknow,Unknow)	(Unknow,Unknow,Unknow)

PHÁT HIỆN LỖI

- Để che giấu lỗi thì bước đầu tiên là phải phát hiện chính xác lỗi. Phát hiện lỗi là một trong những công việc quan trọng của hệ thống phân tán có khả năng chịu lỗi.
- Chỉ cần tối thiểu hai cơ chế để phát hiện lỗi: Chủ động gửi thông điệp hỏi "IS ALIVE" hoặc thụ động nhận thông điệp đó từ tiến tình khác, trong thực tế thường dùng cơ chế thứ nhất (PING). Cần thường xuyên trao đổi thông tin với các nút có liên quan.
- Cần phân biệt giữa lỗi mạng và lỗi trên mỗi nút mạng
- Khi phát hiện lỗi, cần có cơ chế thông báo tới mỗi nút có liên quan

TRUYỀN THÔNG TIN CẬY GIỮA MÁY KHÁCH VÀ MÁY CHỦ

- Tính chịu lỗi trong hệ thống phân tán không chỉ tập trung vào các lỗi ở các tiến trình mà còn phải chú ý đến các lỗi truyền thông
- Lỗi truyền thông có thể là: Mất kênh truyền, thất lạc thông tin, quá thời gian, lặp thông điệp…
- Để khắc phục lỗi truyền thông, người ta thường sử dụng phương pháp truyền tin tin cậy giữa điểm với điểm (unicast) hoặc giữa điểm với nhiều điểm (Multicast)
- Truyền tin tin cậy giữa điểm với điểm được vận dụng dựa trên các giao thức truyền tin cậy (ví dụ TCP)

CÁC TÌNH HUỐNG LỖI KHI GỌI THỦ TỤC TỪ XA

1. Tiến trình trên máy khách không thể định vị được tiến trình trên máy chủ
2. Mất thông điệp yêu cầu từ tiến trình trên máy khách đến tiến trình trên máy chủ
3. Sau khi nhận được yêu cầu, tiến trình trên máy chủ bị lỗi
4. Mất thông điệp trả về kết quả từ tiến trình trên máy chủ đến tiến trình trên máy khách
5. Tiến trình trên máy khách bị lỗi sau khi gửi yêu cầu

TIẾN TRÌNH TRÊN MÁY KHÁCH KHÔNG ĐỊNH VỊ ĐƯỢC TIẾN TRÌNH TRÊN MÁY CHỦ

- Tất cả các tiến trình cần thiết trên các máy chủ đã bị tắt
- Sai phiên bản phần mềm (do nâng cấp máy chủ)
- Cách phát hiện: Hầu hết các ngôn ngữ lập trình đều hỗ trợ cơ chế

  try {… }

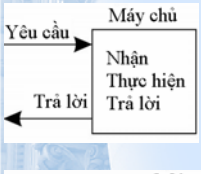
  catch (Exception){….}

  Tuy nhiên điều này vi phạm tính trong suốt của hệ thống, do đó lập trình viên cần phải tự phát triển mô đun dựa trên các thông điệp mô tả trong Exception
- Mô đun phát hiện lỗi này trước hết phải kiểm tra xem có lỗi trên mạng hoặc máy chủ còn cung cấp dịch vụ hay không

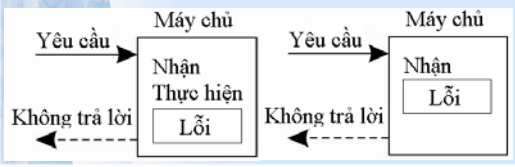
MẤT THÔNG ĐIỆP GỬI YÊU CẦU

- Thiết lập đồng hồ thời gian khi bắt đầu gửi thông điệp, nếu quá thời gian cho phép mà chưa nhận được phản hồi từ phía máy chủ thì gửi lại.
- Máy khách gửi lại yêu cầu:
  - Nếu vẫn gặp lỗi thì trở lại trường hợp thứ nhất.
  - Nếu gửi lại thành công thì cần phân biệt trường hợp yêu cầu được gửi lại

LỖI TRÊN MÁY CHỦ



- Trường hợp thứ nhất chỉ cần thông báo lỗi
- Trường hợp thứ hai yêu cầu máy khách gửi lại



CHIẾN LƯỢC XỬ LÝ TRÊN MÁY CHỦ

- Đợi đến khi nào máy chủ hoạt động trở lại, nó sẽ cố thực hiện yêu cầu đã nhận được trước khi xảy ra lỗi. Như vậy thủ tục gọi từ xa được thực hiện ít nhất một lần.
- Máy chủ sau khi được khôi phục sẽ không thực hiện yêu cầu nhận được trước khi bị lỗi mà sẽ gửi lại thông báo hỏng cho máy khách biết để máy khách gửi lại yêu cầu. Như vậy thủ tục từ xa được thực hiện nhiều nhất một lần và cũng có thể không được thực hiện lần nào.
- Máy chủ không đảm bảo gì hết. Khi máy chủ bị lỗi, máy khách không hề hay biết gì cả. Gọi thủ tục từ xa có thể được thực hiện nhiều lần cũng có thể không thực hiện lần nào.

CHIẾN LƯỢC XỬ LÝ TRÊN MÁY KHÁCH

- Máy khách không gửi lại yêu cầu. Vì thế không biết bao giờ yêu cầu đó mới thực hiện được hoặc có thể không bao giờ được thực hiện.
- Máy khách liên tục gửi lại yêu cầu, có thể dẫn tới trường hợp một yêu cầu được thực hiện nhiều lần.
- Máy khách chỉ gửi lại yêu cầu nào đó khi không nhận được bản tin xác nhận từ máy chủ thông báo đã nhận thành công. Máy khách dùng bộ đếm thời gian, sau một khoảng thời gian xác định trước mà không nhận được xác nhận từ phía máy chủ thì sẽ gửi lại yêu cầu.
- Máy khách chỉ gửi lại yêu cầu khi nhận được thông báo lỗi từ phía máy chủ.

TỔ HỢP CÁC TÌNH HUỐNG LỖI

- Với ba chiến lược trên máy chủ và bốn chiến lược trên máy khách sẽ cho 12 tổ hợp, tuy nhiên không có tổ hợp nào trọn vẹn.
- Để giải thích tính không trọn vẹn của mỗi tổ hợp, có thể lấy ví dụ về việc thực hiện trên máy chủ. Ba sự kiện có thể xảy ra trên máy chủ:
  - Hoàn thành gửi thông điệp (M)
  - In văn bản (P)
  - Lỗi (C)

TỔ HỢP CÁC TÌNH HUỐNG LỖI TRÊN MÁY CHỦ

1. M →P →C: Lỗi xảy ra sau khi đã hoàn thành việc gửi thông điệp và in văn bản.
2. M →C (→P): Lỗi xảy ra khi đã hoàn thành gửi thông điệp nhưng chưa kịp in văn bản.
3. P →M →C: Lỗi xảy ra sau khi đã hoàn thành việc in văn bản và gửi thông điệp.
4. P→C(→M): In xong văn bản thì bị lỗi trước khi gửi thông điệp
5. C (→P →M): Lỗi xảy ra khi máy chủ chưa kịp làm gì.
6. C (→M →P): Lỗi xảy ra khi máy chủ chưa kịp làm gì.

TỔ HỢP CÁC CHIẾN LƯỢC XỬ LÝ KHI LỖI TRÊN MÁY CHỦ

Chiến lược gửi lại của máy khách	Máy chủ					
	Chiến lược M →P			Chiến lược P →M		
	MPC	MC(P)	C(MP)	PMC	PC(M)	C(PM)
Luôn luôn	Lặp	Tốt	Tốt	Lặp	Lặp	Tốt
Không	Tốt	Không	Không	Tốt	Tốt	Không
Khi xác nhận	Lặp	Tốt	Không	Lặp	Tốt	Không
Khi xác nhận có lỗi	Tốt	Không	Tốt	Lặp	Lặp	Tốt

MẤT THÔNG ĐIỆP TRẢ VỀ KẾT QUẢ

- Máy trạm sử dụng cơ chế đồng hồ thời gian, tuy nhiên chưa xác định được nguyên nhân mất thông diệp gửi yêu cầu, lỗi trên máy chủ hay mất thông điệp trả về kết quả.
- Có thể xảy ra tình huống gửi lặp, cần phân biệt hai trường hợp:
  - Lặp không nguy hại cho hệ thống (không thay đổi giá trị)
  - Lặp có nguy hại cho hệ thống (Có thay đổi giá trị)
- Giải pháp xử lý:
  - Máy khách gán định danh cho mỗi yêu cầu
  - Máy chủ lưu vết các yêu cầu xử lý của máy khách
  - Mỗi yêu cầu của máy khách chỉ được máy chủ xử lý một lần
  - Gán thời gian quá hạn thích hợp

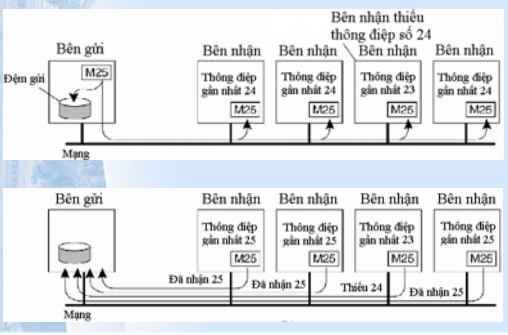
MÁY KHÁCH BỊ LỖI

- Máy khách có thể bị treo (khóa tập tin, tiến trình…) , tiến trình trên các máy khác có liên quan có thể chờ vô thời hạn.
- Giải pháp xử lý:
  - Kiểm tra ngoài: Lưu vết thông điệp gửi đi, sau khi khởi động lại phải đọc tập tin lưu vết trước khi thực hiện trao đổi thông tin với máy chủ. Nhược điểm: Phải ghi lượng lớn dữ liệu vào ổ đĩa.
  - Tái sinh: Máy khách không lưu vết mà chia khoảng thời gian đánh số tuần tự, khi khởi động lại thì loan tin cho các máy khác để loại bỏ tất cả các giao tác có liên quan đến giai đoạn trước của máy khách
  - Tái sinh lịch sự: Tương tự như trên, nhưng chỉ loại bỏ giao tác khi không tìm thấy chủ của chúng

TRUYỀN THÔNG NHÓM TIN CẬY

- Sau khi phân nhóm tiến trình, một tiến trình khác muốn thực hiện gửi thông điệp tới tất cả các tiến trình trong nhóm đó. Truyền thông nhóm tin cậy là phải có cơ chế để đảm bảo thông điệp đến được tất cả các thành viên trong nhóm.
- Khi xảy ra lỗi thì che giấu bằng phương pháp đánh số tuần tự các thông điệp cần gửi.
  - Các thông điệp được lưu tại một vùng đệm của bên gửi cho đến khi nhận được bản tin xác nhận từ tất cả các thành viên trong nhóm.
  - Nếu bên nhận xác định là bị mất một bản tin nào đó thì nó sẽ gửi về một thông điệp yêu cầu gửi lại.
  - Thông thường, bên gửi sẽ tự động gửi lại thông điệp sau trong khoảng thời gian xác định trước nếu không nhận được thông điệp xác nhận.

SƠ ĐỒ NHÓM TIN CẬY CƠ BẢN



TRUYỀN NHÓM TIN CẬY TRONG HỆ THỐNG QUI MÔ LỚN

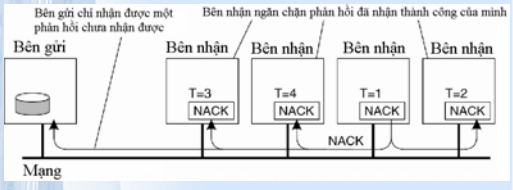
- Truyền thông nhóm tin cậy cơ bản không hỗ trợ cho các hệ thống lớn. Gửi tin cho N nút thì sẽ phải nhận lại ít nhất N phản hồi.
- Giải pháp đề xuất: Bên nhận không cần phải gửi xác nhận cho tất cả các thông điệp mà chỉ thông báo những thông điệp còn thiếu, do đó bên gửi phải lưu toàn bộ các thông điệp đã gửi.
- Hai giải pháp xử lý:
  - Điều khiển phản hồi không phân cấp
  - Điều khiển phản hồi có phân cấp

ĐIỀU KHIỂN PHẢN HỒI KHÔNG PHÂN CẤP

- Giảm số lượng phản hồi đến bên gửi.
- Thực hiện bằng giao thức SRM (Scalable Reliable Multicasting) do Floyd đề xuất năm 1997
- Các trạm nhận không trả về thông điệp ACK (nhận thành công). Khi một trạm phát hiện thiếu thông điệp sẽ gửi cho trạm gửi và toàn bộ thành viên trong nhóm thông điệp NACK (chưa nhận được thông điệp).
- Cho phép các thành viên khác trong nhóm tự triệt tiêu phản hồi NACK của mình:
  - Một trạm cũng có NACK đối với một thông điệp sẽ chờ một thời gian nhất định.
  - Sau thời gian chờ, nếu không nhận được NACK cho thông điệp đó thì mới gửi thông điệp NACK cho các thành viên khác trong nhóm và trạm gửi.

ĐIỀU KHIỂN PHẢN HỒI KHÔNG PHÂN CẤP

- Bên gửi sẽ chỉ nhận được một phản hồi NACK, nhận được phản hồi NACK thì gửi lại thông điệp cho toàn bộ nhóm.



ĐIỀU KHIỂN PHẢN HỒI KHÔNG PHÂN CẤP

- Đảm bảo chỉ phải gửi lại 01 thông điệp bị mất phụ thuộc vào việc lập lịch thông điệp phản hồi tại mỗi trạm nhận, nếu không cùng một thời điểm sẽ có nhiều trạm nhận gửi thông điệp NACK. Việc thiết lập thời gian trên toàn nhóm là điều không dễ dàng.
- Bắt buộc mọi thành viên trong nhóm đều phải nhận thông điệp NACK ngay cả khi không cần thiết
- Biện pháp khắc phục:
  - Tạo thêm một nhóm mới phục vụ cho thông điệp NACK, điều này rất khó quản lý trong mạng qui mô lớn.
  - Cải tiến SRM: Các thành viên trong nhóm hỗ trợ nhau phục hồi những thông điệp bị mất trước khi chuyển thông điệp NACK cho bên gửi

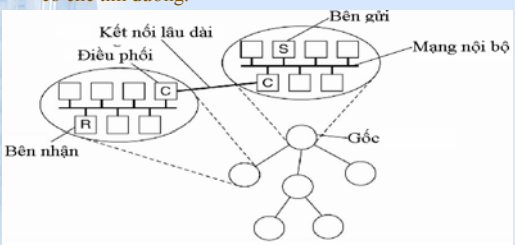
Nguyễn Xuân Anh, anhcz@yahoo.com

ĐIỀU KHIỂN PHẢN HỒI PHÂN CẤP

- Để có thể thực hiện truyền tin cậy cho một nhóm lớn các tiến trình, tổ chức các nhóm theo cấu trúc hình cây:
  - Gốc là nhóm chứa tiến trình gửi.
  - Các nút là các nhóm có chứa tiến trình nhận.
- Mỗi nhóm bầu chọn tiến trình điều phối có nhiệm vụ xử lý các yêu cầu truyền lại. Tiến trình điều phối của mỗi nhóm sẽ có bộ đệm riêng:
  - Nếu không nhận được thông điệp M thì sẽ gửi yêu cầu truyền lại tới tiến trình điều phối của nút cha.
  - Đối với các thành viên trong nhóm có thể sử dụng kịch bản truyền nhóm cơ bản hoặc phản hồi không phân cấp

ĐIỀU KHIỂN PHẢN HỒI PHÂN CẤP

- Việc xây dựng cấu trúc cây khá phức tạp. Nhiều trường hợp yêu cầu cây phải có cấu trúc động nên phải có một cơ chế tìm đường.

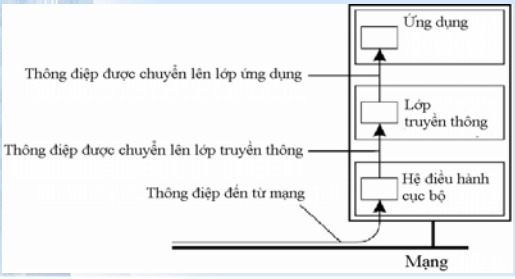


TRUYỀN THEO NHÓM NGUYÊN TỬ

- Tình huống: Hoặc tất cả các tiến trình trong nhóm nhận thông điệp hoặc không có tiến trình nào được nhận (ví dụ khi nhân bản CSDL).
- Ý tưởng thực hiện: khi một tiến trình muốn gửi thông điệp cho một nhóm các tiến trình khác, nó sẽ không gửi bản tin tới tất cả các tiến trình của nhóm chứa các tiến trình nhận mà chỉ gửi đến một nhóm nhỏ các tiến trình cần nhận bản tin đó (G).
- Yêu cầu: phải đảm bảo gửi được bản tin tới tất cả các tiến trình trong nhóm hoặc không được gửi tới bất kì tiến trình nào nếu một tiến trình trong nhóm bị lỗi sụp đổ.
- Thay đổi G: khi đang gửi thông điệp tới G mà có một tiến trình xin gia nhập nhóm hay xin ra khỏi nhóm thì sự thay đổi này sẽ được gửi tới tất cả các thành viên còn lại trong nhóm.

ĐỒNG BỘ ẢO

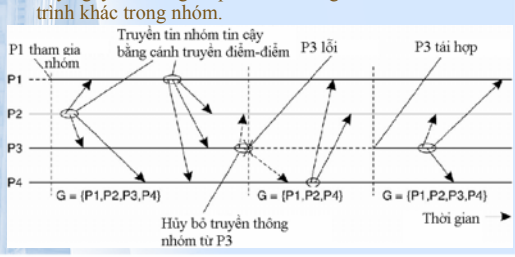
Thông điệp được tạm thời lưu lại trong vùng đệm của lớp truyền thông cho đến khi được phép chuyển lên lớp ứng dụng



1. Nhận được thông điệp từ mạng, chuyển đến vùng đệm tại lớp truyền thông, đánh dấu trạng thái chưa ổn định
2. Kiểm tra xem tất cả các tiến trình trong nhóm đã nhận được thông điệp hay chưa? Nếu đúng thì chuyển trạng thái của thông điệp sang ổn định
3. Chuyển toàn bộ các thông điệp có trạng thái ổn định sang lớp ứng dụng

NGUYÊN LÝ TRUYỀN NHÓM ĐỒNG BỘ ẢO

Thông điệp chỉ được chuyển đến tất cả các tiến trình không có lỗi. Nếu tiến trình gửi bị sụp đổ thì quá trình này bị hủy ngay dù thông điệp đó đã được gửi tới một vài tiến trình khác trong nhóm.



THỨ TỰ PHÂN PHÁT THÔNG ĐIỆP TRONG TRUYỀN THÔNG ĐIỆP THEO NHÓM

- Ba dạng phân phát thông điệp cơ bản:
  - Không theo thứ tự: Lớp truyền thông không đảm bảo phân phát thông điệp theo thứ tự của bên gửi
  - Thứ tự dạng FIFO: Lớp truyền thông phải đảm bảo phân phát thông điệp nhận được theo thứ tự đã gửi
  - Thứ tự theo nhân quả (Causally-ordered): Lớp truyền thông phân phát thông điệp theo thứ tự trước sau (ví dụ khi thông điệp có gán nhãn thời gian)
- Dạng mở rộng (Toàn bộ theo thứ tự): Dù thuộc loại nào trong ba loại cơ bản trên, chỉ cần đảm bảo thứ tự thông điệp đến tất cả các thành viên trong nhóm phải như nhau

THỨ TỰ PHÂN PHÁT THÔNG ĐIỆP TRONG TRUYỀN THÔNG ĐIỆP THEO NHÓM

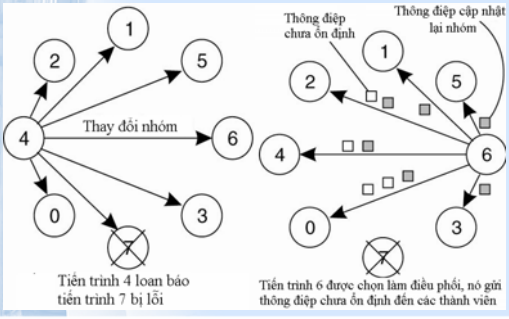
Không theo thứ tự		
Tiến trình P1	**Tiến trình P2**	**Tiến trình P3**
Gửi thông điệp M1	Nhận thông điệp M1	Nhận thông điệp M2
Gửi thông điệp M2	Nhận thông điệp M2	Nhận thông điệp M1

Dạng FIFO			
Tiến trình P1	**Tiến trình P2**	**Tiến trình P3**	**Tiến trình P4**
Gửi M1	Nhận M1	Nhận M3	Gửi M3
Gửi M2	Nhận M3	Nhận M1	Gửi M4
	Nhận M2	Nhận M2	
	Nhận M4	Nhận M4	

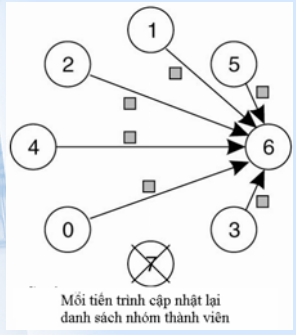
CÀI ĐẶT ĐỒNG BỘ ẢO

Truyền thông theo nhóm	Đảm bảo phân phát thông điệp	
	Dạng cơ bản	Toàn bộ theo thứ tự
Tin cậy	Không có	Không
FIFO	FIFO	Không
Nhân quả	Nhân quả	Không
Nguyên tử	Không có	Có
FIFO nguyên tử	FIFO	Có
Nhân quả nguyên tử	Nhân quả	Có

CÀI ĐẶT ĐỒNG BỘ ẢO

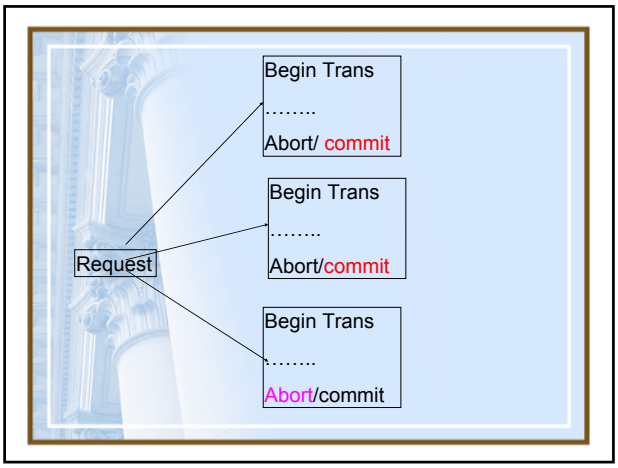


CÀI ĐẶT ĐỒNG BỘ ẢO



COMMIT PHÂN TÁN

- Mô hình thiết lập cam kết phải là mô hình phân cấp và tiến trình điều phối lãnh trách nhiệm thiết lập cam kết phân tán.
- Trong cam kết một pha đơn giản, tiến trình điều phối thông báo với tất cả các thành viên còn lại hoặc là thực hiện hoặc là không thực hiện một thao tác nào đó.
- Nếu thành viên nào đó không thực hiện được cũng không thể báo lại cho tiến trình điều phối biết. Do đó cần đưa ra mô hình mới:
  - COMMIT hai pha
  - COMMIT ba pha



Giả sử A: Tài khoản ngân hàng Techcombak
B: Tài khoản ngân hàng VpBank

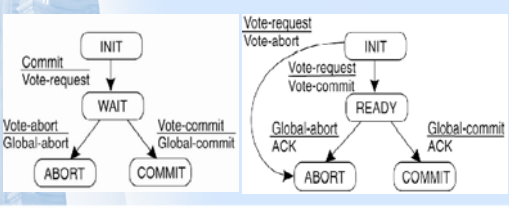
```
Open trans
    Bal=b.getbalence();
    b.deposite(Bal/10); //VpBank
    a.withdraw(bal/10); //Techcombak
COMMIT
```

COMMIT HAI PHA

- Cam kết hai pha gồm hai: Pha bầu cử và pha quyết định
- Pha bầu cử:
  - Tiến trình điều phối gửi một yêu cầu bầu cử VOTE_REQUEST tới tất cả các thành viên trong nhóm.
  - Sau khi nhận được VOTE_REQUEST, từng thành viên quyết định xem có thể thực hiện hay không và gửi lại VOTE_COMMIT hoặc VOTE_ABORT.
- Pha quyết định:
  - Tiến trình điều phối tập hợp các phiếu bầu. Nếu tất cả đều đồng ý thì gửi GLOBAL_COMMIT. Chỉ cần một thành viên từ chối thì gửi GLOBAL_ABORT cho tất cả các thành viên trong nhóm.
  - Các thành viên thực hiện GLOBAL_COMMIT hoặc GLOBAL_ABORT

TRẠNG THÁI CỦA COMMIT HAI PHA

- Các trạng thái của một tiến trình điều phối bao gồm: INIT, WAIT, ABORT, COMMIT.
- Các trạng thái của một thành viên bất kì bao gồm : INIT, READY, ABORT, COMMIT.



NHẬN XÉT COMMIT HAI PHA

- Nhược điểm chính của COMMIT hai pha là tốn nhiều thời gian chờ đợi. Cả tiến trình điều phối và các thành viên đều phải chờ một thông điệp nào đó được gửi đến cho mình.
- Nếu tiến trình bị lỗi thì hoạt động của cả hệ thống sẽ bị ảnh hưởng.
- Để khắc phục nhược điểm của COMMIT hai pha trong trường hợp tiến trình điều phối bị lỗi, người ta đưa ra mô hình cam kết ba pha.

COMMIT HAI PHA TẠI TIÊN TRÌNH ĐIỀU PHỐI

```
write START_2PC to local log;
multicast VOTE_REQUEST to all participants;
while not all votes have been collected {
    wait for any incoming vote;
    if timeout {
        write GLOBAL_ABORT to local log;
        multicast GLOBAL_ABORT to all participants;
        exit;
    }
    record vote;
}
if all participants sent VOTE_COMMIT and coordinator votes COMMIT {
    write GLOBAL_COMMIT to local log;
    multicast GLOBAL_COMMIT to all participants;
} else {
    write GLOBAL_ABORT to local log;
    multicast GLOBAL_ABORT to all participants;
}
```

PHA BẦU CHỌN TẠI TIẾN TRÌNH THÀNH VIÊN

```
write INIT to local log;
wait for VOTE_REQUEST from coordinator;
if timeout {
      write VOTE_ABORT to local log;
      exit;
}
if participant votes COMMIT {
      write VOTE_COMMIT to local log;
      send VOTE_COMMIT to coordinator;
      wait for DECISION from coordinator;
      if timeout {
            multicast DECISION_REQUEST to other participants;
            wait until DECISION is received; /* remain blocked */
            write DECISION to local log;
      }
      if DECISION == GLOBAL_COMMIT
            write GLOBAL_COMMIT to local log;
      else if DECISION == GLOBAL_ABORT
            write GLOBAL_ABORT to local log;
} else {
      write VOTE_ABORT to local log;
      send VOTE_ABORT to coordinator;
}
```

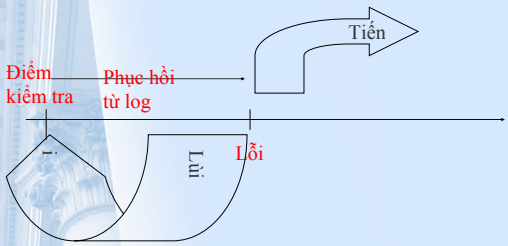
PHA QUYẾT ĐỊNH TẠI TIẾN TRÌNH THÀNH VIÊN

```
while true {
    wait until any incoming DECISION_REQUEST is received; /* remain blocked */
    read most recently recorded STATE from the local log;
    if STATE == GLOBAL_COMMIT
        send GLOBAL_COMMIT to requesting participant;
    else if STATE == INIT or STATE == GLOBAL_ABORT
        send GLOBAL_ABORT to requesting participant;
    else
        skip; /* participant remains blocked */
}
```

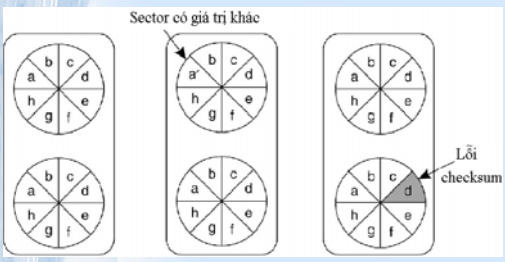
COMMIT BA PHA

PHỤC HỒI

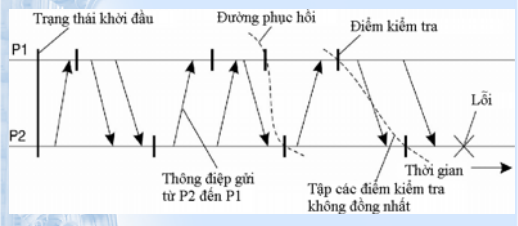
- Phục hồi là các phương pháp đưa trạng thái bị lỗi sang trạng thái không lỗi (fault free).
- Có hai cách tiếp cận cho phục hồi lỗi: phục hồi lùi (back forward) và phục hồi tiến (forward recovery).
  - Phục hồi lùi: khi thực hiện phục hồi lùi sẽ thực hiện phục hồi trạng thái không lỗi của hệ thống trước khi có lỗi và cho hệ thống chạy lại từ điểm đó. Để có thể thực hiện được điều này phải sử dụng các điểm kiểm tra (checkpoint). Tại các điểm này sẽ sao lưu trạng thái hiện hành của hệ thống để khi khôi phục sẽ cho chạy ở điểm kiểm tra gần nhất.
  - Phục hồi tiến: chuyển hệ thống từ trạng thái lỗi sang trạng thái mới để có thể tiếp tục thực hiện



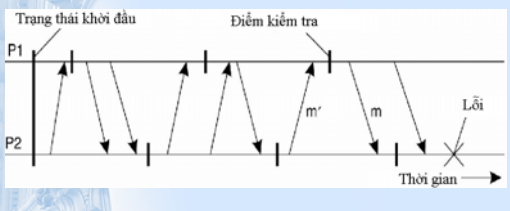
PHỤC HỒI – THIẾT BỊ LƯU TRỮ ỔN ĐỊNH



ĐÁNH DẤU ĐIỂM KIỂM TRA



ĐÁNH DẤU ĐIỂM KIỂM TRA ĐỘC LẬP



GHI NHỚ THÔNG ĐIỆP