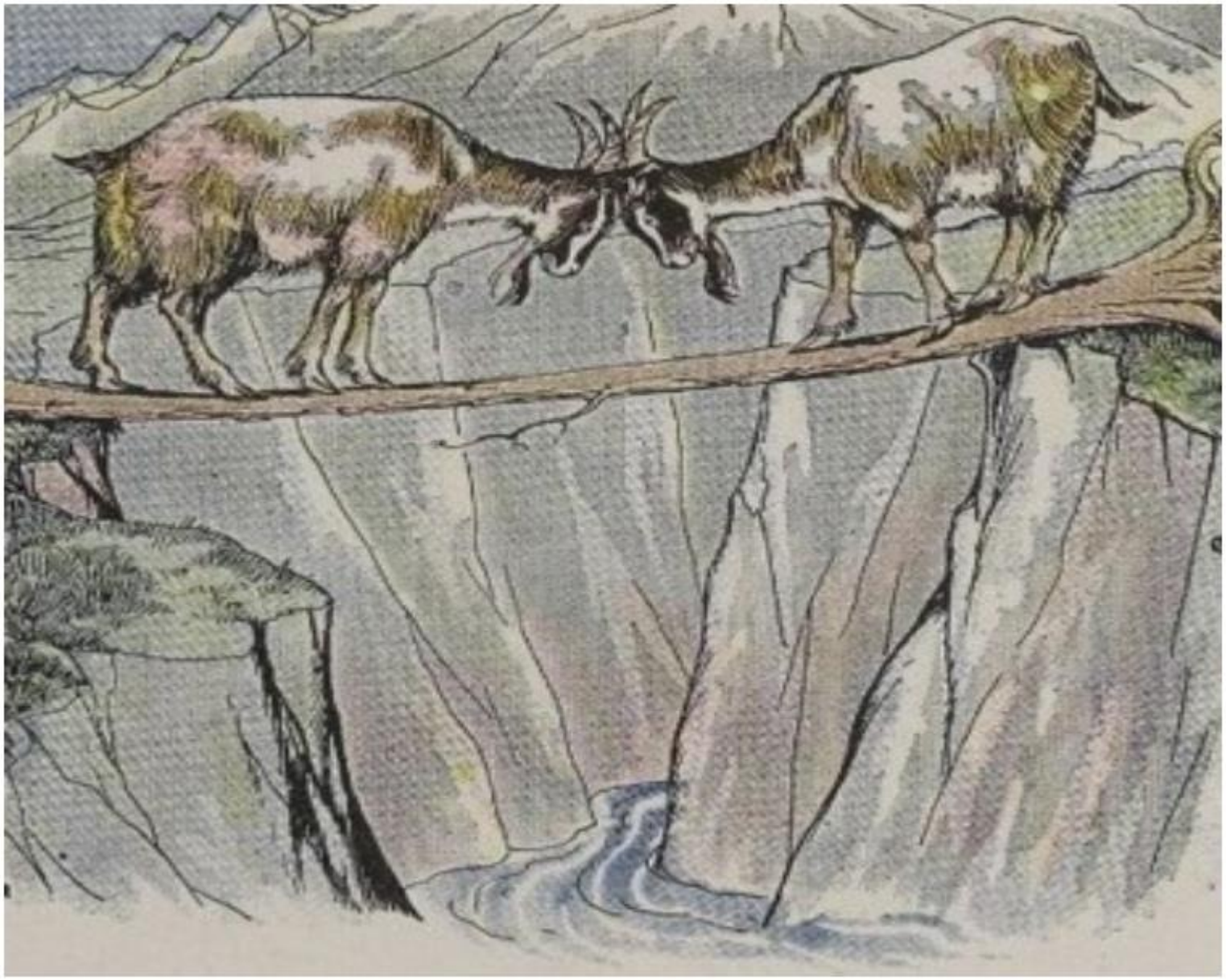
PHẦN 2: QUẢN LÝ TIẾN TRÌNH

1. Tiến Trình
2. Luồng
3. Điều phối CPU
4. Tài nguyên găng và điều độ tiến trình
5. Bế tắc và xử lý bế tắc



(*Nguồn: http://sedition.com/a/393*)

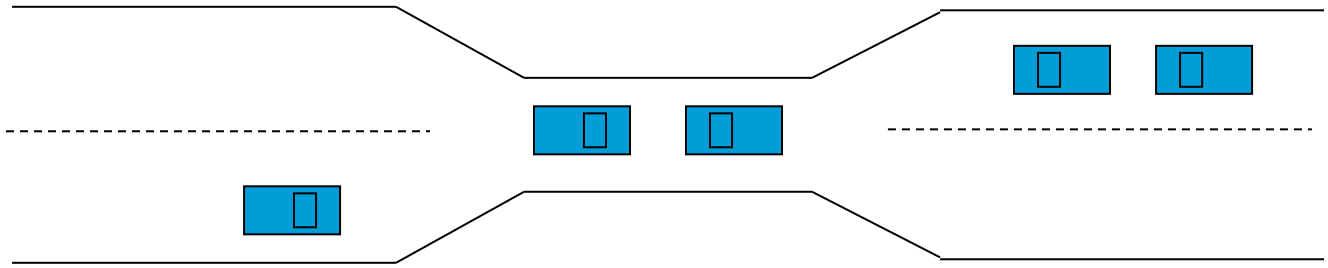
4.1. Khái niệm

4.2. Các phương pháp giải quyết bài toán tới hạn

4.1. Khái niệm

4.2. Các phương pháp giải quyết bài toán tới hạn

Khái niệm



- **Trong hệ đa chương có nhiều process hoạt động song hành, mỗi process có 1 không gian làm việc độc lập, không ai có thể truy xuất trực tiếp không gian làm việc của mỗi process**
- **Khi 2 hay nhiều process cần trao đổi dữ liệu ta cần cung cấp cơ chế cho chúng. Có 2 cơ chế giao tiếp chính giữa những process là :Truy xuất bộ nhớ và gửi/nhận thông điệp.**
- **Truy xuất bộ nhớ dùng chung là một trong nhiều hoạt động gây ra tương tranh. Như vậy ở đây bộ nhớ chung là một dạng tài nguyên "Găng"**

Định nghĩa

- **Tài nguyên : Tất cả những gì cần thiết cho thực hiện tiến trình**
- **Tài nguyên găng:**
  - ***Tài nguyên hạn chế về khả năng sử dụng chung cần đồng thời cho nhiều tiến trình***
  - ***Tài nguyên găng có thể là thiết bị vật lý hay dữ liệu dùng chung***
- **Vấn đề**
  - ***Dùng chung tài nguyên găng có thể dẫn đến không đảm bảo tính toàn vẹn dữ liệu***

    →Đòi hỏi cơ chế đồng bộ hóa các tiến trình

Đoạn găng(Critical section)

- **Đoạn găng (chỗ hẹp) là đoạn chương trình sử dụng tài nguyên găng**
  - ***Đoạn chương trình thực hiện truy nhập và thao tác trên dữ liệu dùng chung***
- **Khi có nhiều tiến trình sử dụng tài nguyên găng thì phải điều độ**
  - ***Mục đích: đảm bảo không có quá một tiến trình nằm trong đoạn găng***

Yêu cầu của chương trình điều độ

- **Loại trừ lẫn nhau (Mutual Exclusion) : Mỗi thời điểm, tài nguyên găng không phải phục vụ một số lượng tiến trình vượt quá khả năng của nó**
  - ***Một tiến trình đang thực hiện trong đoạn găng (sử dụng tài nguyên găng) → Không một tiến trình nào khác được quyền vào đoạn găng***
- **Tiến triển (Progress) Tài nguyên găng còn khả năng phục vụ và tồn tại tiến trình muốn vào đoạn găng thì tiến trình đó phải được sử dụng tài nguyên găng**
- **Chờ đợi hữu hạn (Bounded Waiting) Nếu tài nguyên găng hết khả năng phục vụ và vẫn tồn tại tiến trình muốn vào đoạn găng, thì tiến trình đó phải được xếp hàng chờ đợi và sự chờ đợi là hữu hạn**

4.1. Khái niệm

4.2. Các phương pháp giải quyết bài toán tới hạn

Quy ước

- **Có 2 tiến trình P1&P2 thực hiện đồng thời**
- **Các tiến trình dùng chung một tài nguyên găng**
- **Mỗi tiến trình đặt đoạn găng ở đầu, tiếp theo là phần còn lại**
  - ***Tiến trình phải xin phép trước khi vào đoạn găng {phần vào}***
  - ***Tiến trình khi thoát khỏi đoạn găng thực hiện {phần ra}***
- **Cấu trúc tổng quát của một tiến trình**

Repeat

Phần vào

{Đoạn găng của tiến trình}

Phần ra

{Phân còn lại của tiến trình}

Until false

Các phương pháp giải quyết bài toán tới hạn

- **1. Phương pháp khóa trong**
  - ***Chỉ 1 tiến trình được vào đoạn tới hạn còn các tiến trình khác phải chờ.***
  - ***Sử dụng 1 byte khóa K để làm khóa. Khi TT vào đoạn tới hạn K=1, khi ra khỏi đoạn tới hạn K=0.***
  - ***Trước khi vào đoạn tới hạn các TT phải kiểm tra byte khóa của TT khác. Nếu tiến trình khác có byte khóa bằng 1 thì TT phải chờ đến khi nó nhận giá trị 0.***

```
Var k1,k2: Interger;
BEGIN k1:=0;k2:=0;
        PARBEGIN
        TT1: Repeat
              While k2=1 do {không làm gi}
              {k1=1}
              {Đoan găng TT1}
              k1:=0;
             {Phần còn lai của TT1}
        Until false;
```

```
TT2: Repeat
              While k1=1 do {không làm gi}
              {k2=1}
              {Đoan găng TT2}
              k2:=0;
              {Phần còn lai của TT2}
              Until false;
PAREND;
END;
```

Các phương pháp giải quyết bài toán tới hạn

- **1. Phương pháp khóa trong**

```
Var k1,k2: Interger;
BEGIN k1:=0;k2:=0;
        PARBEGIN
        TT1: Repeat
                While k2=1 do {không làm gi}
                {k1=1}
                {Đoan găng TT1}
                k1:=0;
               {Phần còn lai của TT1}
        Until false;
```

```
TT2: Repeat
                While k1=1 do {không làm gi}
                {k2=1}
                {Đoan găng TT2}
                k2:=0;
                {Phần còn lai của TT2}
                Until false;
PAREND;
END;
```

Phương pháp khóa trong

- **Về ý tưởng phương pháp khóa trong rất đơn giản và giải quyết tốt vấn đề tranh chấp. Tuy nhiên trong 1 số tình huống phương pháp này sẽ thất bại.**
- **Ví dụ: 2 Tiến trình P1 có tốc độ lớn hơn P2 rất nhiều cùng muốn vào đoạn găng. P2 kiểm tra byte khóa của TT khác không thấy byte nào có giá trị 1. Nó dừng quá trình kiểm tra và chuyển sang quá trình thiết lập biến khóa lên 1. Ngay lúc này TT P1 cũng kiểm tra và thấy không có byte khóa nào bằng 1(Do P2 chưa kịp set K=1), vì vậy P1 cũng set biến trạng thái K1=1 và vào đoạn găng →*tranh chấp***

Thuật toán Dekker

- ***Ý tưởng:*** **Thuật toán Dekker sử dụng thêm 1 biến trạng thái tt để xác định mức độ ưu tiên giữa các tiến trình cùng muốn vào đoạn găng.**

```
Var c1,c2,tt: Integer;
BEGIN c1 := 0; c2 := 0; tt := 1;
 PARBEGIN
  TT1: Repeat
         c1 := 1;
         While c2 = 1 do
             if tt = 2 then
              begin c1 := 0;
                 while tt = 2 do ;
                 c1 := 1
              end;
         { Đoạn găng TT 1}
            c1 := 0; tt := 2;
         { Phần còn lại của TT 1}
        Until false;
```

```
    TT2: Repeat
           c2 := 1;
           While c1 = 1 do
               if tt = 1 then
                begin c2 := 0;
                   while tt = 1 do ;
                   c2 := 1
                end;
           { Đoạn găng TT 2}
              c2 := 0; tt := 1;
           { Phần còn lại của TT 2}
          Until false
     PAREND
END.
```

Thuật toán Dekker

- **Ưu điểm: Thuật toán Dekker giải quyết được trường hợp tốc độ thực hiện của các tiến trình là khác nhau.**
- **Nhược điểm: Thuật toán sẽ trở nên vô cùng phức tạp khi số lượng tiến trình tăng.**

Test and Set

- **Phân tích vấn đề của phương pháp biến khóa ta nhận thấy nếu 2 quá trình kiểm tra và xác lập biến khóa đồng nhất không tách rời thì lỗi của chương trình trước sẽ không thể xảy ra.**
- **Để thực hiện điều này HĐH cần sự trợ giúp của phần cứng. Hãng chế tạo CPU cung cấp 1 lệnh máy đặc biệt gọi là TSL (test and set), TSL làm việc với 2 biến: Biến chung G và biến riêng L.**
- **TS(L) -> L:=G;**

  G:=1;

Test and Set

- **Sơ đồ điều độ**

```
Var l1, l2, g: Integer;
BEGIN
   g := 0;
 PARBEGIN
  TT1: Repeat
         l1 := 1;
        While l1 = 1 do TS(l1);
        {Đoạn găng TT1}
        g := 0;
        {Phần còn lại của TT1}
       Until false;
  TT2: Repeat
         l2 := 1;
        While l2 = 1 do TS(l2);
        {Đoạn găng TT2}
        g := 0;
        {Phần còn lại của TT2}
       Until false
 PAREND
END.
```

g	l1	l2
0	1	
1	0	1
1		1
1		1
1		1
0		0
1	1	
1	1	
1	1	
	1	

Test and Set

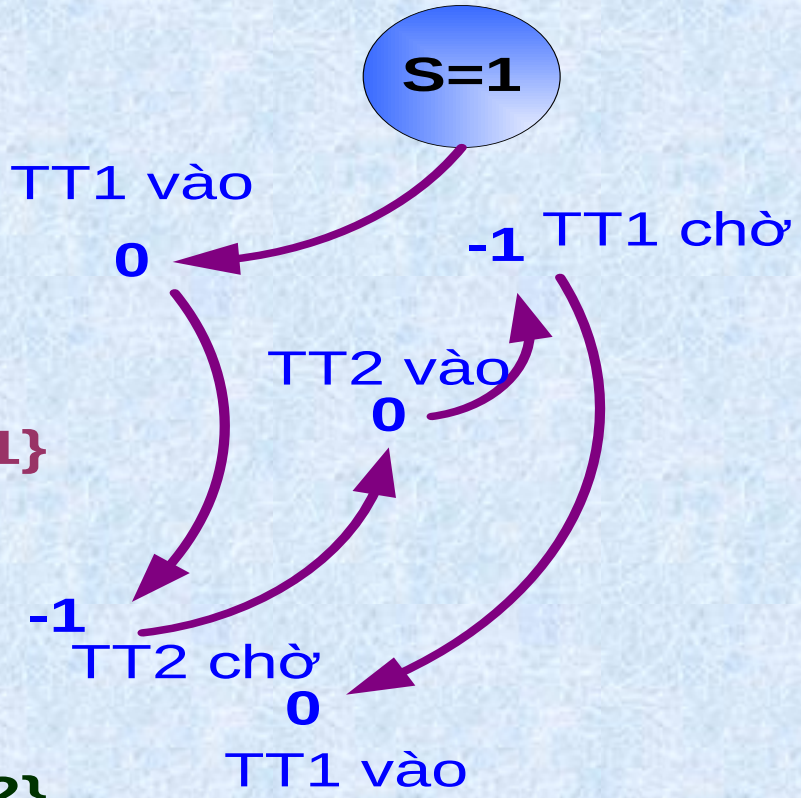
- **Ưu điểm: Đơn giản, độ phức tạp không tăng khi số tiến trình tăng.**
- **Nhược điểm: Trong trường hợp có nhiều tiến trình chờ khó xác định tiến trình nào sẽ vào đoạn găng sớm nhất phụ thuộc vào thời điểm giải phóng tài nguyên găng của tiến trình đang sử dụng.**

Semaphore

- **Ý tưởng:**
  - ***Mỗi tài nguyên Găng được biểu diễn bởi 1 biến nguyên dương s. Đa số các tài nguyên găng đều có khả năng phục vụ bằng 1 (s=1)***
  - ***Tồn tại 2 lệnh máy làm thay đổi giá trị của s.***
    - P(s)→ s-1. If s<0 then đưa TT vào hàng đợi
    - V(s)→ s+1. If s=<0 then kích hoạt tiến trình đang xếp hàng.

Semaphore

```
Var s:Integerl
BEGIN
  s := 1;
 PARBEGIN
  TT1:Repeat
         P(s);
        {Đoạn găng TT1}
        V(s);
        {Phần còn lại TT1}
       Until false;
TT2:Repeat
         P(s);
        {Đoạn găng TT2}
        V(s);
        {Phần còn lại TT2}
       Until false
 PAREND
END.
```

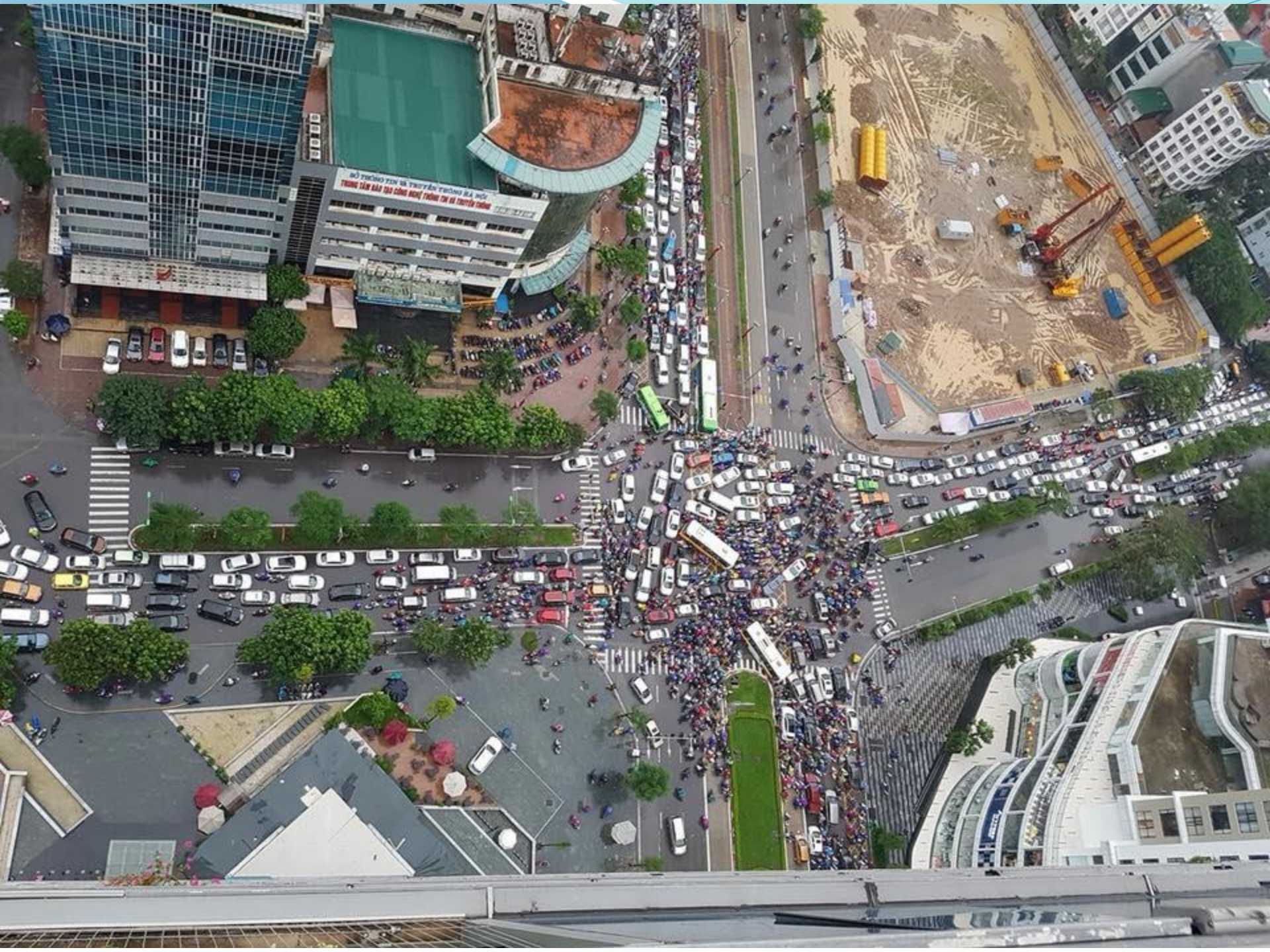


Semaphore

- **Đặc điểm: Mỗi tiến trình chỉ phải kiểm tra quyền vào đoạn găng 1 lần. Sau đó nó vào đoạn găng hoặc chờ tại hàng đợi. Trường hợp phải chờ ở hàng đợi nó sẽ không phải làm gì mà sẽ chờ tiến trình trong đoạn găng đánh thức khi nó ra khỏi hàng đợi.**

PHẦN 2: QUẢN LÝ TIẾN TRÌNH

1. Tiến Trình
2. Luồng
3. Điều phối CPU
4. Tài nguyên găng và điều độ tiến trình
5. Bế tắc và xử lý bế tắc


BỘ THÔNG TIN VÀ TRUYỀN THÔNG
TRUNG TÂM ĐÀO TẠO CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Bế tắc (Deadlock)

- **Ví dụ 2 tiến trình P1 và P2 cùng sử dụng các tài nguyên r1 và r2 được điều khiển bởi 2 đèn hiệu S1 và S2. Tại mỗi thời điểm, mỗi tài nguyên chỉ phục vụ hoạt động của 1 tiến trình.**

P1	Thời điểm	P2
Wait(S1)	T1	Wait(S2)
…	T2	…
Wait(S2)	T3	…
	T4	Wait(S1)

- **T1: P1 yêu cầu được cấp r1, P2 yc cấp r2**
- **T2: 2 tiến trình được cấp tài nguyên và đi vào hoạt động**
- **T3: P1 yêu cầu hệ thống cấp tài nguyên r2, tuy nhiên r2 đang được P2 sử dụng nên P1 phải chờ r2**
- **T4: P2 yêu cầu hệ thống cấp tài nguyên r1, tuy nhiên r1 đang được P1 sử dụng nên P2 phải chờ r2**
- **Như vậy sau T4 cả 2 tiến trình rơi vào trạng thái chờ vô hạn.**

Bế tắc

- **Khái niệm: Bế tắc là trạng thái khi hai hay nhiều tiến trình cùng chờ đợi một số sự kiện nào đó và nếu không có tác động từ bên ngoài thì sự chờ đợi đó là vô hạn.**
- **Điều kiện xảy ra bế tắc:**
  - ***Có tài nguyên Găng***
  - ***Có hiện tượng giữ và đợi: Có một tiến trình đang giữ một số tài nguyên và đợi tài nguyên bổ sung đang được giữ bởi các tiến trình khác.***
  - ***Không có hệ thống phân phối lại tài nguyên: Việc sử dụng tài nguyên Găng không bị ngắt***
  - ***Có hiện tượng chờ đợi vòng tròn***

Bế tắc

- **Các mức phòng tránh bế tắc**
  - ***Ngăn ngừa: Áp dụng các biện pháp để hệ thống không rơi vào trạng thái bế tắc***
  - ***Dự báo và tránh bế tắc: Áp dụng các biện pháp để kiểm tra các tiến trình xem có rơi vào trạng thái bế tắc hay không. Nếu có thì thông báo trước khi xảy ra***
  - ***Nhận biết và khắc phục: Tìm cách phát hiện và giải quyết***

Bế tắc

- **Các biện pháp phòng chống bế tắc: Để phòng chống bế tắc, cần đảm bảo sao cho 4 điều kiện gây bế tắc không đồng thời xảy ra:**
  - ***Loại bỏ tài nguyên Găng: mô phỏng tài nguyên Găng bằng các tài nguyên có thể dùng chung được (áp dụng kỹ thuật SPOOL)***
  - ***Loại bỏ yếu tố giữ và đợi:***
    - Tiến trình yêu cầu tất cả tài nguyên 1 lần
    - Chỉ xử lý khi đủ tài nguyên cần thiết
  - ***Xây dựng hệ thống ngắt tài nguyên: Nếu tiến trình yêu cầu tài nguyên bổ sung mà hệ thống không đáp ứng được thì mọi tài nguyên của nó sẽ bị giải phóng để phục vụ tiến trình khác.***
  - ***Loại bỏ yếu tố giữ và đợi***

Bế tắc

- **Dãy tiến trình an toàn: Cho dãy tiến trình P1, P2, P3, … Pn song hành. Dãy tiến trình được gọi là an toàn (Safe Process) nếu với mọi tiến trình Pi, tài nguyên mà Pi cần có thể được thỏa mãn bởi các tài nguyên khả dụng của hệ thống và tài nguyên do các tiến trình Pj đang giữ với điều kiện j<i**
- **Hệ thống ở trạng thái an toàn tại một thời điểm nếu dãy tiến trình song hành tại thời điểm đó có thể được sắp xếp thành 1 dãy an toàn.**

Bế tắc

Xét một hệ thống có 12 tài nguyên là 12 băng từ và 3 tiến trình P0, P1, P2 với các yêu cầu cấp phát:

- ***P0 yêu cầu nhiều nhất 10 băng từ***
- ***P1 yêu cầu nhiều nhát 4 băng từ***
- ***P2 yêu cầu nhiều nhất 9 băng từ***

Giả sử tại một thời điểm t_0, P0 đang chiếm 5 băng từ, *P1* và P2 mỗi tiến trình chiếm 2 băng từ. Như vậy có 3 băng từ rỗi

Bế tắc

	Yêu cầu nhiều nhất	Yêu cầu hiện tại
P_0	10	5
P_1	4	2
P_2	9	2

- **Tại thời điêm t_0 hệ thống ở trạng thái an toàn**
- **Thứ tự <*P1*, P0, P2> thỏa mãn điều kiện an toàn**
- **Giả sử ở thời điểm t_1 P2 có yêu cầu và được cấp phát 1 băng từ: Hệ thống không ở trạng thái an toàn nữa... → quyết định cấp tài nguyên cho P2 là sai.**

Bế tắc

	Allocation			*Max*			*Available*			*Need*		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2	7	4	3
P1	2	0	0	3	2	2				1	2	2
P2	3	0	2	9	0	2				6	0	0
P3	2	1	1	2	2	2				0	1	1
P4	0	0	2	4	3	3				4	3	1

Bế tắc

- **Ví dụ về dãy tiến trình an toàn: Có 5 tiến trình P0-P4 và 3 kiểu tài nguyên A, B, C ở trạng thái như sau:**

P	Đang có			Cần tối đa			Cần thêm			Hệ thống còn		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	3	0	2	7	0	3	4	0	1	1	2	2
P1	2	1	0	9	2	2	7	1	2			
P2	0	1	1	2	5	3	2	4	2			
P3	1	2	4	4	3	2	3	1	0			
P4	2	0	0	3	2	2	1	2	2			

- **A. Tìm dãy an toàn của các tiến trình trên**
- **B. Giả sử P4 yêu cầu tài nguyên (1, 2, 2). Hỏi có thể phân phối cho P4 được không?**
- **C. Tương tự câu B với P3?**