Phần 1: Lý thuyết chung về mô phỏng mạng và đánh giá hiệu năng

Chương 1- Tổng quan về sự đánh giá và phân tích hiệu năng của hệ thống (system performance evaluation and analysis)

Tác giả: R. Jain
Dịch thuật: Nguyễn Thị Hiền, Nguyễn Mạnh Linh
Biên tập: Hoàng Trọng Minh

1.1 Các lỗi thường gặp

Mục này sẽ trình bày về các lỗi thường gặp trong phân tích và đánh giá hiệu năng hệ thống. Hầu hết các lỗi được liệt kê ở đây là lỗi không cố ý mà do các nhầm lẫn đơn giản, nhận thức sai hoặc thiếu kiến thức về các kỹ thuật đánh giá hiệu năng.

E1- Không có mục đích

Mục đích là phần quan trọng nhất trong bất kỳ phương pháp đánh giá hiệu năng nào. Tuy nhiên, trong nhiều trường hợp, khởi đầu của việc đánh giá hiệu năng chưa xác định được mục đích rõ ràng. Một người làm công việc phân tích hiệu năng thường gắn bó chặt chẽ và lâu dài cùng với bộ phận thiết kế. Sau quá trình thiết kế, người phân tích hiệu năng có thể bắt đầu mô hình hóa hoặc mô phỏng thiết kế đó. Khi được hỏi về mục đích, những câu trả lời tiêu biểu của các nhà phân tích là: mô hình này sẽ giúp chúng ta trả lời các vấn đề phát sinh từ thiết kế. Yêu cầu chung đối với các mô hình kiểu này là đảm bảo tính mềm dẻo và dễ thay đổi để giải quyết những vấn đề phức tạp. Người phân tích có kinh nghiệm đều hiểu rằng: Không có mô hình cụ thể nào được sử dụng cho một mục đích chung chung. Mỗi một mô hình phải được phát triển với mục đích cụ thể xác định trước. Các thông số, khối lượng công việc và phương pháp thực hiện đều phụ thuộc vào mục đích. Các phần của thiết kế hệ thống trong một mô hình được nghiên cứu tùy theo các vấn đề khác nhau. Bởi vậy, trước khi viết dòng mã chương trình mô phỏng đầu tiên hoặc viết phương trình đầu tiên của một mô hình phân tích hoặc trước khi cài đặt một thí nghiệm đo, người phân tích cần hiểu về hệ thống và nhận thức rõ vấn đề để giải quyết. Điều đó sẽ giúp nhận biết chính xác các thông số, khối lượng công việc và phương pháp thực hiện.

Xác lập các mục đích không phải là một bài toán đơn giản. Bởi vì hầu hết các vấn đề về hiệu năng đều mơ hồ khi được trình bày lần đầu. Vì vậy, nhận thức rõ vấn đề viết ra một tập hợp của các mục đích là việc khó. Một khi vấn đề được sáng tỏ và mục đích đã được viết ra, thì việc tìm ra giải pháp thường sẽ dễ dàng hơn.

E2- Các mục đích thiên vị (biased)

Một lỗi thông thường khác là việc đưa ra các mục đích theo hướng thiên vị ngầm hoặc thiên vị rõ rệt. Ví dụ, nếu mục đích là "Chỉ ra rằng hệ thống của chúng ta tốt hơn hệ thống của người khác", vấn đề này trở thành việc tìm kiếm các thông số và tải làm việc sao cho hệ thống của chúng ta trở nên tốt hơn. Đúng ra thì cần phải tìm ra các thông số và tải làm việc đúng đắn để so sánh hai hệ thống. Một nguyên tắc của quy ước chuyên nghiệp của người phân tích là không thiên vị. Vai trò của người phân tích giống như vai trò của Ban giám khảo. Không nên có bất kỳ sự thiên vị nào định trước và mọi kết luận phải dựa vào kết quả phân tích chứ không phải là dựa vào các niềm tin thuần túy.

E3. Phương pháp tiếp cận không có hệ thống

Các nhà phân tích thường tiến hành theo phương pháp tiếp cận không có hệ thống; bởi vậy họ lựa chọn tham số hệ thống, các yếu tố ảnh hưởng, thông số (hiệu năng) và tải làm việc một cách tùy ý. Điều này sẽ dẫn tới các kết luận sai. Phương pháp tiếp cận hệ thống được sử dụng để giải quyết một

vấn đề về hiệu năng là nhận biết một tập hoàn chỉnh của các mục đích, tham số hệ thống, các nhân tố ảnh hưởng, các thông số hiệu năng và tải làm việc.

E4. Phân tích mà không hiểu về vấn đề
Các nhà phân tích thiếu kinh nghiệm cảm thấy rằng không có gì thực sự có được trước khi một mô hình được dựng nên và chỉ có được một số kết quả. Với kinh nghiệm đã có, họ nhận ra rằng, phần lớn của các nỗ lực phân tích là dùng cho việc xác định một vấn đề. Phần này thường chiếm tới 40% tổng số nỗ lực này. Điều này khẳng định một châm ngôn xưa: "Khi một vấn đề được nêu ra rõ ràng thì đã được giải quyết xong một nửa". 60% còn lại liên qua tới vấn đề thiết kế các phương pháp, giải thích kết quả và trình bày kết luận. Việc phát triển của mô hình tự bản thân nó là phần nhỏ của quá trình giải quyết vấn đề. Ví dụ như, xe ô tô và tàu hỏa là phương tiện để đi tới đâu đó chứ không phải là điểm đến cuối cùng. Các mô hình là phương thức để đi đến kết luận chứ không phải là kết quả cuối cùng. Các nhà phân tích đã được đào tạo về các khía cạnh mô hình hóa của vấn đề đánh giá hiệu năng nhưng không được đào tạo về việc xác định vấn đề hoặc trình bày kết quả thì thường thấy rằng mô hình của họ bị bỏ đi bởi người phê duyệt, vì người phê duyệt là người đang tìm kiếm đường hướng chứ không tìm kiếm một mô hình.

E5. Các thông số hiệu năng không đúng
Một thông số hiệu năng (metric) ứng với một tiêu chí được sử dụng để định lượng hiệu năng của hệ thống. Các ví dụ về các thông số hiệu năng hay dùng là thông lượng (throughput) và thời gian đáp ứng (response time). Sự lựa chọn của các thông số hiệu năng đúng đắn phụ thuộc vào các dịch vụ cung cấp bởi hệ thống hoặc bởi hệ thống con đang được mô hình hóa.
Một lỗi chung khi lựa chọn các thông số hiệu năng đó là các nhà phân thích thường chọn các thông số dễ tính toán hoặc dễ đo đạc hơn là chọn thông số thích hợp.

E6 Tải làm việc không có tính đại diện (unrepresentative workload)
Tải làm việc được sử dụng để so sánh hai hệ thống cần đại diện cho khía cạnh sử dụng thực tiễn của các hệ thống này trong lĩnh vực của chúng. Ví dụ, nếu các gói dữ liệu trong mạng thông thường bao gồm hai loại có kích thước ngắn và dài thì tải làm việc dùng để so sánh hai mạng phải bao gồm các gói dữ liệu có kích thước ngắn và dài.
Việc chọn tải làm việc ảnh hưởng rất lớn tới kết quả của việc nghiên cứu hiệu năng. Tải làm việc sai sẽ dẫn tới các kết luận sai.

E7 Phương pháp đánh giá sai
Có ba phương pháp đánh giá: đo lường, mô phỏng và mô hình hóa phân tích. Các nhà phân tích thường có một phương pháp ưa thích và được sử dụng thường xuyên đối với mọi vấn đề về đánh giá hiệu năng. Ví dụ như những ai thành thạo về lý thuyết hàng đợi sẽ có xu hướng quy đổi mọi vấn đề về hiệu năng sang một vấn đề về hàng đợi ngay cả khi hệ thống quá phức tạp và thuận lợi cho việc đo lường. Những ai thành thạo về lập trình sẽ thường có xu hướng giải quyết mọi vấn đề bằng mô phỏng. Việc gắn với một phương pháp đơn lẻ này dẫn tới kết quả một mô hình mà họ có thể giải quyết tốt nhất hơn là một mô hình giải quyết tốt nhất vấn đề này. Vấn đề đối với các quy trình chuyển đổi này là chúng có thể đưa thêm các hiện tượng vào mô hình, trong khi các hiện tượng này không có trong hệ thống nguyên gốc hoặc dẫn đến có thể bỏ qua các hiện tượng quan trọng thuộc về hệ thống nguyên gốc.
Một nhà phân tích cần có hiểu biết cơ bản về cả ba phương pháp. Khi xem xét lựa chọn phương pháp đánh giá hiệu năng, cần chú ý tới nhiều hệ số khác nhau.

E8 Bỏ qua các thông số quan trọng
Nên tạo ra một danh sách hoàn chỉnh về các đặc điểm của hệ thống và của tải làm việc ảnh hưởng tới hiệu năng của hệ thống. Những đặc điểm này được gọi chung là thông số. Các thông số tải làm việc có thể bao gồm số người sử dụng, các loại yêu cầu đến, sự ưu tiên, v…v. Nhà phân tích có thể

chọn một tập hợp các giá trị cho mỗi thông số. Kết quả nghiên cứu cuối cùng phụ thuộc nhiều vào các chọn lựa này. Bỏ sót một hoặc nhiều thông số quan trọng có thể nhận được các kết quả vô ích.

E9. Bỏ qua các hệ số quan trọng
Các thông số biến đổi trong nghiên cứu thì được gọi là các hệ số. Ví dụ như trong số các thông số về tải làm việc liệt kê trên đây, chỉ có số lượng người sử dụng có thể được chọn như là một hệ số, và các thông số khác có thể được giữ nguyên tại các giá trị điển hình. Không phải tất cả các thông số có tác động như nhau đối với hiệu năng. Điều quan trọng là nhận ra những tham số mà nếu chúng thay đổi thì sẽ gây nên ảnh hưởng quan trọng tới hiệu năng. Trừ khi có lý do nào khác, những thông số này nên được sử dụng như là các hệ số trong việc nghiên cứu hiệu năng. Ví dụ như nếu tốc độ (rate) gói đến tác động tới thời gian đáp ứng của một gateway của mạng hơn là ảnh hưởng của kích thước gói tới nó, việc nghiên cứu sẽ tốt hơn nếu như sử dụng một số tốc độ đến khác nhau thay vì quan tâm tới kích thước gói.

E10 Thiết kế thí nghiệm không thích hợp
Sự thiết kế thí nghiệm liên quan tới số lượng các phép đo hoặc các thí nghiệm mô phỏng được thực hiện và các giá trị của các thông số sử dụng trong mỗi thínghiệm. Việc chọn đúng các giá trị này có thể mang tới nhiều thông tin hơn đối với cùng một số lượng các thí nghiệm. Chọn lựa không đúng có thể gây ra lãng phí thời gian của nhà phân tích và tài nguyên.

E11. Mức độ chi tiết không thích đáng
Mức độ chi tiết được sử dụng trong mô hình của hệ thống có ảnh hưởng quan trọng trong việc hệ thống hóa, công thức hóa vấn đề. Một lỗi chung xảy ra là việc sử dụng lối tiếp cận chi tiết đối với mô hình hóa ở mức cao sẽ thực hiện và ngược lại.

E12. Không phân tích
Một vấn đề chung trong dự án đo lường là chúng thường được thực hiện bởi các nhà phân tích hiệu năng, đó thường là những người giỏi về các kỹ thuật đo nhưng thiếu sự thành thạo trong phân tích dữ liệu. Họ thu thập một lượng khổng lồ các dữ liệu nhưng không biết phương pháp phân tích hoặc giải thích nó như thế nào.

E13. Phân tích sai
Các nhà phân tích có thể gây nên hàng loạt các lỗi trong khi đo đạc, mô phỏng và mô hình hóa phân tích vì dụ như lấy giá trị trung bình của các tỷ số và thời gian mô phỏng quá ngắn.

E14. Không phân tích độ nhậy
Các nhà phân tích thường quá nhấn mạnh đến kết quả của sự phân tích của họ, trình bày nó như là một thực tế hơn là một bằng chứng. Thực tế mà trong đó các kết quả nhạy cảm đối với tải làm việc và thông số hệ thống thì thường bị coi nhẹ. Khi không có sự phân tích độ nhậy, không thể chắc chắn rằng liệu các kết luận có thay đổi hay không nếu như phân tích này được thức hiện trong một thiết lập khác biệt đôi chút. Không có phân tích độ nhạy thì sẽ khó khăn cho việc đánh giá sự quan trọng tương đối của các thông số khác nhau.

E15. Bỏ qua các lỗi đầu vào
Thường là các thông số được lựa chọn không thể đo được. Thay vào đó, ta sử dụng các biến có thể đo được khác để ước lượng thông số này. Ví dụ như trong một thiết bị mạng máy tính, các gói dữ liệu được lưu trữ trong danh sách liên kết của bộ đệm. Mỗi một bộ đệm có dung lượng là 512x8bit. Với một số lượng bộ đệm được yêu cầu để lưu trữ gói dữ liệu, không thể dự báo trước một cách chính xác số gói hoặc số bít trong gói dữ liệu. Điều nãy dẫn tới độ bất định được cộng thêm ở dữ liệu đầu vào. Nhà phân tích cần điều chỉnh mức độ tin cậy trong kết quả đầu ra của mô hình thu được từ dữ liệu này

E16. Cách xử lý mẫu ngoại lai không thích hợp
Những giá trị quá cao hoặc quá thấp so với phần lớn giá trị trong một tập hợp được gọi là mẫu ngoại lai. Mẫu ngoại lai trong đầu vào hoặc đầu ra của mô hình là một vấn đề. Nếu mẫu ngoại lai không được tạo ra bởi một hiện tượng trong hệ thống thực, nó có thể được bỏ qua. Mô hình bao trùm mẫu ngoại lai có thể tạo nên một mô hình không hợp lệ. Mặt khác, nếu mẫu ngoại lai là sự xuất hiện có thể xảy ra trong trong hệ thống thực, mẫu ngoại lai này cần hiện diện trong mô hình. Việc bỏ qua mẫu ngoại lai kiểu này có thể tạo nên một mô hình không hợp lệ.

E17. Giả thiết không có thay đổi trong tương lai: Tương lai thường được giả thiết sẽ giống như quá khứ.
Một mô hình dựa trên tải làm việc và hiệu năng quan sát được trong quá khứ được sử dụng để dự báo hiệu năng trong tương lai. Tải làm việc và hoạt động hệ thống trong tương lai được giả thiết là giống như những gì đã đo được. Nhà phân tích và người thực hiện quyết định nên thảo luận về giả thiết này và giới hạn thời lượng tương lai cho các dự đoán.

E18. Bỏ qua tính biến thiên
Thường thì người ta chỉ phân tích hiệu năng trung bình bởi vì việc xác định tính biến thiên thường gặp khó khăn. Nếu độ biến thiên lớn, nếu chỉ sử dụng duy nhất giá trị trung bình có thể dẫn tới quyết định sai. Ví dụ, việc quyết định dựa trên nhu cầu máy tính trung bình hàng ngày có thể không có ích nếu như không tính tới đặc tính tải đạt đỉnh điểm theo giờ, gây tác động bất lợi tới hiệu năng người sử dụng.

E19 Phân tích quá phức tạp
Các nhà phân tích hiệu năng nên đi đến kết luận bằng phương thức đơn giản nhất có thể. Tốt nhất là luôn bắt đầu với một mô hình hoặc thí nghiệm đơn giản nhằm đạt được một số kết quả và sau đó tăng thêm tính phức tạp. Các mô hình công bố trong tài liệu khoa học và các mô hình sử dụng trong thực tế khác nhau rõ rệt. Các mô hình trong các tài liệu khoa học, trong các trường học thường là quá phức tạp. Phần lớn các vấn đề hiệu năng trong thực tế hàng ngày được giải quyết bởi các mô hình đơn giản. Các mô hình phức tạp nếu có thì cũng hiếm khi được sử dụng.

E20. Trình bày kết quả không thích hợp
Đích cuối cùng của mọi nghiên cứu hiệu năng là để hỗ trợ bài toán quyết định. Một phân tích mà không tạo ra bất kỳ kết quả hữu ích nào thì đó là một sự thất bại bởi đó là sự phân tích với kết quả khó hiểu đối với người đưa ra quyết định. Người phân tích phải có trách nhiệm chuyển tải các kết quả phân tích tới người đưa ra quyết định qua việc sử dụng các từ ngữ, hình ảnh, đồ thị để giải thích kết quả phân tích.

E21. Bỏ qua các khía cạnh xã hội
Sự trình bày thành công kết quả phân tích yêu cầu 2 loại kỹ năng: xã hội và chuyên biệt. Kỹ năng viết và nói là kỹ năng xã hội trong khi mô hình hóa và phân tích dữ liệu là các kỹ năng chuyên biệt. Hầu hết các nhà phân tích đều có các kỹ năng chuyên biệt tốt, nhưng chỉ những người có các kỹ năng xã hội tốt thì mới thành công khi bán các kết quả của họ cho những người ra quyết định. Việc chấp nhận kết qủa phân tích yêu cầu hình thành sự tin tưởng giữa người ra quyết định và nhà phân tích, và sự trình bày các kết quả tới người ra quyết định theo cách hiểu chính xác nhất. Nếu những người ra quyết định không tin tưởng hoặc không hiểu sự phân tích, thì nhà phân tích thất bại trong việc tạo nên ấn tượng đối với quyết định cuối cùng. Các kỹ năng xã hội đặc biệt quan trọng khi trình bày các kết quả mà chúng có ảnh hưởng tới niềm tin và giá trị của người ra quyết định hoặc yêu cầu về một thay đổi quan trọng trong thiết kế.

E22. Bỏ sót các giả thiết và các giới hạn
Các giả thiết và các giới hạn của mô hình phân tích thường bị bỏ qua trong báo cáo cuối cùng. Điều này có thể làm cho người sử dụng áp dụng mô hình phân tích này vào một ngữ cảnh khác khi mà

các giả thiết không còn hợp lệ. Đôi khi các nhà phân tích lên danh sách các giả thiết ngay ở phần mở đầu báo cáo nhưng họ quên mất các giới hạn và đưa ra các kết luận về các môi trường mà phân tích này không áp dụng vào.

Bảng 1.1: Danh sách kiểm tra để tránh các lỗi thường gặp khi đánh giá hiệu năng

1. Liệu hệ thống được định nghĩa đúng chưa và mục đích được nêu ra rõ ràng chưa ?
2. Các mục tiêu được nêu ra có đảm bảo tính không thiên vị?
3. Các bước phân tích đi theo hệ thống không?
4. Vấn đề được hiểu rõ ràng trước khi phân tích không?
5. Các tham số hiệu năng có thích hợp cho vấn đề này không ?
6. Tải làm việc có đúng cho vấn đề này không?
7. Kỹ thuật đánh giá có phù hợp không?
8. Danh sách thông số có ảnh hưởng đến hiệu năng đã được hoàn thiện chưa?
9. Tất cả các thông số ảnh hưởng đến hiệu năng mà được coi như các thừa số được thay đổi chưa?
10. Thiết kế thí nghiệm hiệu quả chưa khi xét theo thời gian và kết quả?
11. Mức độ chi tiết đã hợp lý chưa?
12. Dữ liệu đo đạc được phân tích và giải thích chưa?
13. Sự phân tích đã đúng về thống kê chưa?
14. Độ nhạy phân tích được thực hiện chưa?
15. Các lỗi đầu vào có thay đổi kết quả đáng kể không?
16. Các mẫu ngoại lai của đầu vào hoặc đầu ra được xem xét một cách thích đáng chưa ?
17. Các thay đổi trong tương lai của hệ thống và tải làm việc được mô hình hóa chưa?
18. Phương sai của dữ liệu đầu vào được quan tâm không?
19. Phương sai của kết quả được phân tích chưa?
20. Sự phân tích này có dễ giải thích không?
21. Cách thức trình bày có phù hợp với người đọc không?
22. Các kết quả có được trình bày dưới dạng đồ thị nhiều nhất có thể không?
23. Các giả thiết và các giới hạn của sự phân tích được đưa vào tài liệu rõ ràng không?

<u>Một cách tiếp cận có hệ thống cho việc đánh giá hiệu năng</u>

Các thông số, tải làm việc và kỹ thuật đánh giá được sử dụng đối với một vấn đề thì thường không thể được sử dụng cho vấn đề tiếp theo. Tuy nhiên có các bước chung cho tất cả dự án đáng giá hiệu năng mà chúng giúp bạn tránh được các lỗi ghi trong phần 1.1 Các bước này thực hiện như sau.

Bước 1- Xác định mục tiêu và định nghĩa hệ thống

Bước đầu tiên trong vài dự án đánh giá hiệu năng là xác định mục tiêu của việc nghiên cứu và định nghĩa xem cái gì tạo nên hệ thống bằng cách phác họa các giới hạn của hệ thống.

Bước 2: Lập danh sách các dịch vụ và kết quả nhận được

Mỗi một hệ thống cung cấp một tập hợp các dịch vụ. Danh sách của dịch vụ và kết quả khả thi sẽ hữu ích sau này trong việc chọn thông số và tải làm việc đúng.

Bước 3 Lựa chọn các thông số đo

Bước tiếp theo là lựa chọn các tiêu chuẩn để so sánh hiệu năng, chúng được gọi là các thông số đo. Nhìn chung, các thông số này liên hệ với tốc độ, độ chính xác, và ích lợi của dịch vụ.

Bước 4: Lập danh sách các thông số

Bước tiếp theo trong dự án thực hiện là tạo danh sách tất cả các thông số ảnh hưởng tới hiệu năng. Danh sách này có thể được phân chia thành các thông số hệ thống và các thông số tải làm việc

Bước 5: Lựa chọn các thừa số để nghiên cứu

Danh sách các thông số có thể phân chia thành 2 phần: các thông số sẽ được thay đổi trong quá trình đánh giá và các thông số không thay đổi. Những thông số được thay đổi gọi là thừa số và những giá trị của chúng được gọi là mức độ.

1.2 Lựa chọn kỹ thuật và thông số đo

Lựa chọn một kỹ thuật đánh giá và lựa chọn tham số đo là hai bước quan trọng trong tất cả các dự án đánh giá hiệu năng. Có rất nhiều vấn đề cần xem xét để có được lựa chọn chính xác.

1.2.1 Lựa chọn một kỹ thuật đánh giá

Có ba kỹ thuật đánh giá hiệu năng là mô hình hóa phân tích, phương pháp mô phỏng và đo đạc. Có một số khía cạnh cần xem xét để quyết định xem kỹ thuật nào là phù hợp nhất để sử dụng. Những khía cạnh đó được liệt kê trong bảng 3.1 và được sắp xếp theo mức quan trọng giảm dần.

Một vấn đề quan trọng trong việc quyết định kỹ thuật đánh giá đó là chu trình vòng đời trong hệ thống. Các phương pháp đo đạc chỉ khả thi nếu đã tồn tại hệ thống khác tương tự như hệ thống ta đưa ra khảo sát, như là khi thiết kế một phiên bản cải tiến hơn của một sản phẩm. Nếu hệ thống đưa ra là một khái niệm mới thì chỉ có thể chọn mô hình hóa phân tích và phương pháp mô phỏng để thực hiện đánh giá. Mô hình hóa phân tích và mô phỏng có thể sử dụng cho những trường hợp mà đo đạc là không khả thi, nhưng nhìn chung để thuyết phục hơn thì nên kết hợp những phương pháp đó dựa trên các kết quả đo đạc trước đó.

Bảng 1.2 Những tiêu chí để lựa chọn kỹ thuật đánh giá

Tiêu chí	Mô hình phân tích	Mô phỏng	Đo đạc
1. Giai đoạn	Bất cứ giai đoạn nào	Bất cứ giai đoạn nào	Sau thiết kế thử nghiệm (postprototype)
2. Thời gian yêu cầu	Ngắn	Trung bình	Thay đổi
3. Công cụ	Nhà phân tích	Các ngôn ngữ máy tính	Các dụng cụ đo
4. Tính chính xác [a]	Thấp	Vừa phải	Thay đổi
5. Tính đánh đổi	Dễ	Vừa phải	Khó
6. Giá thành	Thấp	Trung bình	Cao
7. Tính dễ bán	Thấp	Trung bình	Cao

a

[a] Trong tất cả các trường hợp, kết quả có thể bị sai lệch.

Vấn đề tiếp theo cần cân nhắc đó là thời gian sử dụng cho công việc đánh giá. Trong hầu hết các trường hợp, các kết quả được yêu cầu từ *ngày hôm trước (thời gian đánh giá ngắn)*. Nếu đúng như vậy thì mô hình hóa phân tích là sự lựa chọn duy nhất. Các phương pháp mô phỏng cần một thời gian dài, các phương pháp đo đạc thường mất nhiều thời gian hơn mô hình hóa phân tích nhưng không lâu như mô phỏng. Theo ý nghĩa của định luật Murphy thì phương pháp đo đạc được sử dụng thường xuyên hơn hai phương pháp còn lại (Nếu một việc có thể diễn ra theo chiều hướng xấu, nó sẽ như vậy – Họa vộ đơn chí). Kết quả là, thời gian cần thiết dành cho đo đạc biến động nhiều nhất trong ba kỹ thuật.

Điểm quan tâm tiếp theo là tính sẵn sàng của công cụ. Các loại công cụ bao gồm: các kỹ năng mô hình hóa, các ngôn ngữ mô phỏng, và các thiết bị đo đạc. Nhiều nhà phân tích hiệu năng rất thành thạo và khéo léo trong mô hình hóa. Họ không cần sử dụng tới bất cứ hệ thống thật đắt tiền nào. Những người khác không thành thạo các lý thuyết hàng đợi thì lại quan tâm hơn đến đo đạc và mô phỏng. Thiếu kiến thức về các ngôn ngữ và kỹ thuật mô phỏng khiến nhiều nhà phân tích xa rời các công cụ mô phỏng hữu dụng.

Mức độ chính xác đòi hỏi cũng là một mối quan tâm khác. Nhìn chung, mô hình hóa phân tích yêu cầu nhiều sự đơn giản hóa và giả thiết đến nỗi nếu các kết quả trở nên chính xác thì ngay cả các nhà

phân tích cũng phải ngạc nhiên. Các phương pháp mô phỏng có thể kết hợp nhiều chi tiết, yếu tố hơn và yêu cầu ít giả thiết hơn là mô hình hóa phân tích và do đó thường gần hơn với thực tế. Các phương pháp đo đạc mặc dù nghe có vẻ gần thực tế nhất nhưng kết quả lại có thể thiếu chính xác, đơn giản vì có nhiều tham số môi trường tác động đến đối với từng thử nghiệm, như là cấu hình hệ thống, loại tải làm việc, và thời gian đo đạc. Các tham số cũng có thể không thể hiện khoảng làm việc thay đổi trong các hệ thống thực tế. Do vậy tính chính xác của kết quả đo đạc thu được có thể thay đổi từ cao đến không có gì.
Cần phải chỉ ra rằng mức độ chính xác và tính đúng đắn của kết luận là không đồng nhất. Một kết quả chính xác đến mười chữ số thập phân cũng có thể bị hiểu sai hay hoặc nhầm lẫn; do đó có thể dẫn tới kết luận sai.
Mục đích của công việc nghiên cứu hiệu năng là vừa để so sánh các phương án khác nhau vừa để tìm ra giá trị tham số tối ưu. Các mô hình phân tích thường cung cấp một cái nhìn tốt nhất về tác dụng của các tham số khác nhau và sự tương tác giữa chúng. Với các phương pháp mô phỏng, có thể tìm được khoảng giá trị tham số cho tổ hợp tối ưu, nhưng thường không thể hiện được rõ ràng sự tương xứng giữa tham số này với tham số khác. Các phương pháp đo đạc là kỹ thuật ít thể hiện được tính tương xứng giữa các tham số nhất. Thật không dễ khi khẳng định rằng hiệu năng được cải thiện là kết quả của một vài sự thay đổi ngẫu nhiên về môi trường hoặc hiệu chỉnh một vài tham số nhất định.
Chi phí cấp cho dự án cũng là một yếu tố quan trọng. Đo đạc đòi hỏi phải có thiết bị thật, dụng cụ đo đạc và thời gian. Đây chính là kỹ thuật tốn kém nhất trong 3 kỹ thuật đã nêu. Chi phí, đi kèm với khả năng dễ dàng thay đổi cấu hình, thường là lí do để phát triển các phương pháp mô phỏng cho các hệ thống đắt tiền. Mô hình hóa phân tích chỉ đòi hỏi giấy và bút chì (cộng với thời gian của nhà phân tích) nên có thể xem như là kỹ thuật rẻ nhất.

Tính bán được của kết quả đánh giá có thể là lý lẽ quan trọng khi chọn xem xét các chi phí, lao động của phương pháp đo đạc. Và kết quả đó dễ được thuyết phục hơn nếu nó được thực hiện với hệ thống thực. Nhiều người hoài nghi về các kết quả phân tích đơn giản vì họ không hiểu công nghệ thực hiện hoặc kết quả cuối cùng. Trong thực tế, người phát triển các kỹ thuật mô hình hóa phân tích mới thường kiểm chứng, xác nhận chúng bằng cách sử dụng các phương pháp mô phỏng hoặc đo lường thực sự.

Đôi khi việc sử dụng hai hay nhiều kỹ thuật đồng thời mang lại nhiều lợi ích. Ví dụ, bạn có thể sử dụng mô phỏng và mô hình hóa phân tích cùng nhau để kiểm tra và xác nhận kết quả riêng của từng phương pháp. *Cho đến khi chưa chứng minh được tội lỗi, mọi người đều được xem là vô tội*, nghĩa là cho đến khi chưa được xác nhận kiểm chứng thì mọi kết quả đánh giá đều đáng nghi ngờ. Điều đó đưa chúng ta đến với 3 quy tắc xác minh sau đây:

- Không tin tưởng vào kết quả của một mô hình mô phỏng cho đến khi chúng đã được xác nhận bởi mô hình hóa phân tích hay đo đạc.

- Không tin tưởng vào kết quả của một mô hình phân tích cho đến khi cũng đã được xác nhận bởi mô hình mô phỏng hay đo đạc.

- Không tin tưởng vào kết quả của phương pháp đo đạc cho đến khi chúng đã được xác nhận bởi mô hình mô phỏng hay mô hình phân tích.

Trong thực tế , sự cần thiết của quy tắc thứ 3 cho việc xác nhận các kết quả đo đạc nên được nhấn mạnh vì đây là quy tắc hay bị bỏ qua nhất. Phương pháp đo đạc dễ bị mắc các lỗi khi thí nghiệm hoặc các sai sót hơn là hai kỹ thuật kia. Yêu cầu duy nhất của phép xác minh là kết quả phải không trái với trực quan mong đợi. Phương pháp xác minh như thế gọi là *trực giác của những chuyên gia*, thường được sử dụng cho các mô hình mô phỏng. Phương pháp này và các phương pháp khác có thể được sử dụng cho các kết quả đo đạc và phân tích.

Hai hay nhiều kỹ thuật cũng có thể được sử dụng tiếp nối nhau. Ví dụ, trong một trường hợp, một mô hình phân tích đơn giản được sử dụng để tìm ra khoảng phù hợp cho các tham số của hệ thống và một mô hình mô phỏng được sử dụng sau đó để nghiên cứu hiệu năng của thệ thống trong khoảng đó. Điều này làm giảm số trường hợp mà phép mô phỏng cần xét đến và dẫn đến việc sử dụng tài nguyên hiệu quả hơn.

1.3- Ý nghĩa của "Confidence interval" trong việc so sánh kết quả

Từ tiếng Anh *sample* và *example* đều bắt nguồn từ một từ Pháp cổ là *essample.* Mặc dù hiện nay đây là hai từ riêng biệt, nhưng việc nhớ đến nguồn gốc chung của chúng cũng khá là quan trọng. Một mẫu *(sample)* chỉ đơn giản là một ví dụ *(example).* Một ví dụ thường không đủ để chứng minh một giả thiết. Tương tự như vậy, một mẫu thường là không đủ để đưa ra một phát biểu rõ ràng về mọi hệ thống. Nhưng sự khác biệt này thường bị bỏ quên. Chúng ta thường đo đạc 2 hệ thống với chỉ 5 hay 10 tải làm việc (workloads) và sau đó kết luận rằng một hệ thống tốt hơn hệ thống kia. Mục đích của phần này là để củng cố sự khác biệt và để thảo luận làm thế nào để sử dụng các mẫu dữ liệu để so sánh hai hệ thống hoặc nhiều hơn.

Ý tưởng cơ bản là một phát biểu chính xác có thể không chính xác với các thuộc tính của tất cả các hệ thống, nhưng một tuyên bố xác suất về khoảng trong đó các thuộc tính của hầu hết các hệ thống tồn tại có thể đúng. Khái niệm về *khoảng tin cậy* (confidence interval) được giới thiệu trong phần này là một khái niệm cơ bản mà bất cứ nhà phân tích hiệu năng hệ thống nào cũng cần biết để hiểu rõ vấn đề.

1.3.1 Mẫu đối ngược với quần thể

Giả sử chúng ta viết một chương trình máy tính để tạo ra vài triệu số ngẫu nhiên với thuộc tính cho trước, ví dụ như có giá trị trung bình ¼ và độ lệch chuẩn Ã. Bây giờ chúng ta đưa các số đó vào một cái bình và rút ra một mẫu của n số .

Giả thiết mẫu $\{x_1, x_2, \ldots, x_n\}$ có giá trị trung bình mẫu là $\bar{x}$. Giá trị trung bình mẫu $\bar{x}$ khác với Ã. Để phân biệt hai giá trị đó, $\bar{x}$ được gọi là giá trị trung bình mẫu và ¼ được gọi là trung bình của quần thể. Từ *quần thể* ám chỉ tất cả các số nằm trong chiếc bình.

Trong hầu hết các vấn đề thực tế, các thuộc tính của quần thể (ví dụ như giá trị trung bình quần thể là không được biết , và mục đích của nhà phân tích là ước lượng các thuộc tính đó. Ví dụ, trong thử nghiệm của chúng ta về đo thời gian xử lý của một chương trình, giá trị trung bình mẫu rút ra từ một mẫu đơn lẻ của n giá trị chỉ là một ước lượng đơn giản của giá trị trung bình. Để xác định chính xác giá trị trung bình, chúng ta cần thực hiện lại thí nghiệm tới vô hạn lần, điều đó gần như là không thể làm được.

Các thuộc tính của quần thể được gọi là các *tham số* trong khi các mẫu thử được gọi là các *thống kê*. Ví dụ, trung bình tập hợp là một tham số trong khi giá trị trung bình mẫu là một thống kê. Ta cần phải phân biệt hai khái niệm này bởi vì các tham số là cố định (*fixed*) trong khi thống kê là một biến ngẫu nhiên. Ví dụ, nếu chúng ta lấy ra hai mẫu n phần tử từ một tập phân phối bình thường với trung bình ¼ và độ lệch chuẩn Ã, kỳ vọng mẫu $\overline{x_1}$ và $\overline{x_2}$ của hai mẫu sẽ khác nhau. Trong thực tế, chúng ta có thể rút ra nhiều mẫu và đưa ra một hàm phân bố cho giá trị trung bình mẫu. Không có phân bố nào như vậy đúng cho giá trị trung bình của cả quần thể. Nó là cố định và chỉ có thể xác định nếu chúng ta xem xét trên toàn bộ quần thể. Thông thường, các ký hiệu Hy Lạp như ¼ hay Ã thường được dùng để chỉ các tham số, trong khi các ký hiệu tiếng Anh như $\overline{x}$ và ***s*** được dùng để chỉ thống kê.

1.3.2 Khoảng tin cậy cho giá trị trung bình

Mỗi giá trị trung bình mẫu là một đánh giá của giá trị trung bình quần thể. Đưa ra ***k*** mẫu, chúng ta có ***k*** đánh giá và những đánh giá đó là khác nhau. Vấn đề tiếp theo là lấy ra một đánh giá duy nhất cho giá trị tủng bình quần thể từ ***k*** đánh giá trên.

Trong thực tế, không thể lấy ra một đánh giá hoàn hảo cho giá trị trung bình quần thể từ một số hữu hạn các mẫu có độ kích thước hữu hạn. Điều tốt nhất chúng ta có thể làm là lấy ra được các biên xác suất. Từ đó, chúng ta có thể lấy ra 2 biên, ví dụ, c_1 và c_2, như thế sẽ có một xác suất cao, 1 - ±, mà kỳ vọng lý thuyết nằm trong khoảng (c_1,c_2):

Probability{c_1d¼dc_2} = 1 - ±

Khoảng (c_1,c_2) được gọi là **khoảng tin cậy** cho giá trị trung bình của quần thể , ± được gọi là **mức ý nghĩa (significant level),** 100(1 - ±) được gọi là **mức tin cậy (confidence level),** và (1 - ±) được gọi là **hệ số tin cậy (confidence coefficient)**. Chú ý rằng mức tin cậy thường được biểu diễn dưới dạng phần trăm và thường gần đến giá trị 100%, ví dụ, 90% hay 95%; trong khi mức ý nghĩa được biểu diễn bởi một phân số và thường có giá trị gần 0, ví dụ 0.05 hay 0.1.

Một cách để xác định khoảng tin cậy 90% là sử dụng 5% và 95% của các giá trị trung bình mẫu làm các biên. Ví dụ, chúng ta có thể lấy k mẫu, tìm các giá trị trung bình mẫu, sắp xếp chúng ra theo một thứ tự tăng dần và lấy ra trong tập sắp xếp đó phần tử thứ [1+0.05(*k*-1)] và [1+0.95(*k*-1)] .

Có một điều may mắn là chúng ta không cần thiết phải lấy ra quá nhiều mẫu. Có thể xác định được khoảng tin cậy chỉ từ duy nhất một mẫu, bởi vì **định lý giới hạn trung tâm** cho ta xác định được phân phối của giá trị trung bình mẫu. Định lý đó phát biểu rằng nếu các giá trị trong mẫu {x1, x2, . . . , xn} là độc lập và được lấy ra từ cùng một tập có giá trị trung bình ¼ và độ lệch chuẩn Ã thì giá trị trung bình mẫu của mẫu đó có phân phối thường xấp xỉ với giá trị trung bình ¼ và độ lệch chuẩn

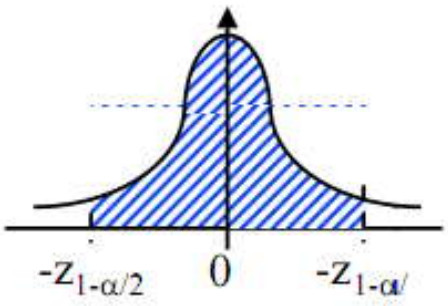
$$\sigma / \sqrt{n} .$$

$$\bar{x} : \ N(\mu, \sigma / \sqrt{n})$$

Độ lệch chuẩn của giá trị trung bình mẫu được gọi là **sai số chuẩn**. Sai số chuẩn khác với độ lệch chuẩn của tập. Nếu độ lệch chuẩn của tập là Ã, thì sai số chuẩn chỉ là $\sigma / \sqrt{n}$ Từ biểu thức trên, dễ thấy rằng khi kích thước mẫu *n* tăng lên thì sai số chuẩn sẽ giảm xuống.

Sử dụng **định lý giới hạn trung tâm,** khoảng tin cậy 100(1 - ±)% cho trung bình quần thể được đưa ra:

$$(\bar{x} - z_{1-\alpha/2} s / \sqrt{n}, \bar{x} + z_{1-\alpha/2} s / \sqrt{n})$$



Ở đây, $\bar{x}$ là giá trị trung bình mẫu, *s* là độ lệch chuẩn của mẫu, *n* là độ lớn mẫu, và $z_{1-\pm/2}$ là điểm phân vị (1 -±/2) của một đại lượng ngẫu nhiên.

Ví dụ 1. Với mẫu ở ví dụ 12.4, kỳ vọng $\bar{x}$ = 3.90, độ lệch chuẩn *s* = 0.95 *và* n = 32:

$$3.90 \text{ m}(1.645)(0.95) / \sqrt{32}$$

Một khoảng tin cậy 90% cho kỳ vọng = (3.62, 4.17)

Chúng ta có thể phát biểu rằng với khoảng tin cậy 90% thì trung bình tập nằm trong khoảng 3.62 và 4.17. Xác suất sai của phát biểu này là 10%. Có nghĩa là, nếu chúng ta lấy 100 mẫu và đưa ra một khoảng tin cậy cho mỗi mẫu như chỉ ra trên hình 13.1, thì trong 90 mẫu sẽ có khoảng tin cậy chứa giá trị trung bình lý thuyết và 10 mẫu thì giá trị trung bình lý thuyết sẽ không nằm trong khoảng tin cậy.

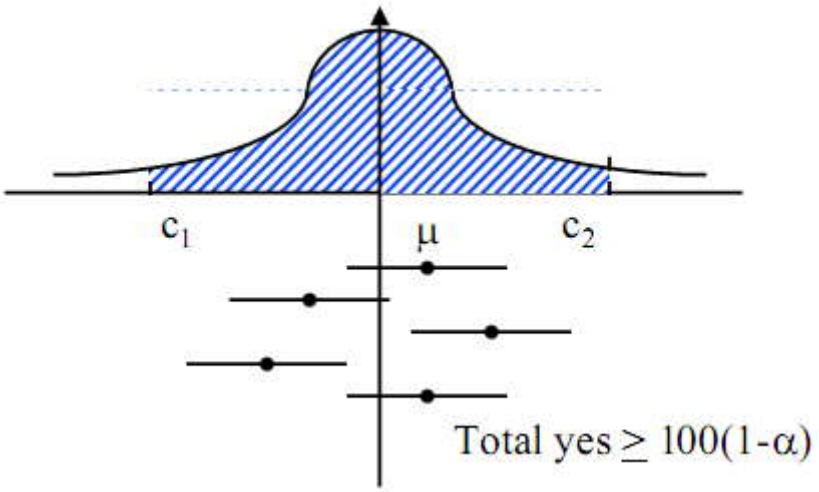
Tương tự như vậy:

Khoảng tin cậy 95% cho kỳ vọng = $3.90 \text{ m}(1.960)(0.95) / \sqrt{32}$

$$= (3.57, 4.23)$$

Khoảng tin cậy 99% cho giá trị trung bình = $3.90 \text{ m}(2.576)(0.95) / \sqrt{32}$

$$= (3.46, 4.33)$$



Hình 1.3.2 Ý nghĩa của khoảng tin cậy

Khoảng tin cậy đề cập ở trên chỉ áp dụng cho các mẫu lớn có độ lớn (kích thước) lớn hơn 30 giá trị. Với các mẫu nhỏ, khoảng tin cậy chỉ có thể được xây dựng nếu các giá trị đến từ một tập phân bố chuẩn. Với những mẫu như vậy khoảng tin cậy 100(1 - ±)% được đưa ra:

$$(\bar{x} - t_{[1-\alpha/2;n-1]} s / \sqrt{n}, \bar{x} + t_{[1-\alpha/2;n-1]} s / \sqrt{n})$$

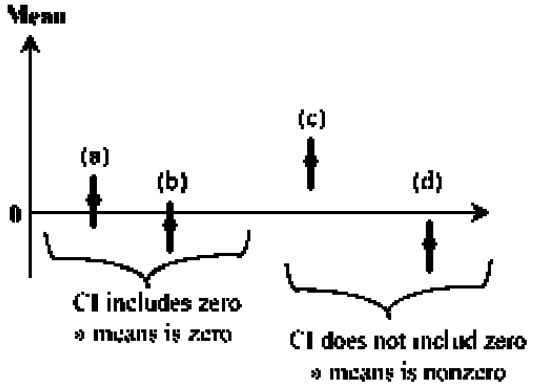
Ở đây $t_{[1-\pm/2;n-1]}$ là điểm phân vị (1 - ±/2) của biến ngẫu nhiên *t* với *n* – 1 bậc tự do. Các điểm phân vị đó được liệt kê trong bảng A.4 của phụ lục. Khoảng này dựa trên một sự thật là với các mẫu từ một tập chuẩn $N(\mu, \sigma^2), (\bar{x} - \mu) / (\sigma / \sqrt{n})$ có một phân bố $N(0,1)$ và $(n-1)s^2 / \sigma^2$ có một phân bố chi bình phương (phân bố x^2) với *n* – 1 bậc tự do, và do đó $(\bar{x} - \mu) / \sqrt{s^2 / n}$ có một phân bố *t* với *n* – 1

bậc tự do . Hình 13.1 chỉ ra một hàm hàm mật độ mẫu *t*, giá trị $t_{[1-\pm/2;n-1]}$ nói lên rằng xác suất của biến ngẫu nhiên nhỏ hơn $t_{[1-\pm/2;n-1]}$ là ±/2. Tương tự với xác suất của biến ngẫu nhiên lớn hơn $t_{[1-\pm/2;n-1]}$. Xác suất biến sẽ nằm trong khoảng $mt_{[1-\alpha/2;n-1]}$ là 1-±.

Ví dụ 13.2: Xét dữ liệu bị lỗi ở ví dụ 12.5 (đã được chỉ ra là có phân bố chuẩn). 8 giá trị lỗi là -0.04, -0.19, 0.14, -0.09, -0.14, 0.19, 0.04, và 0.09.
Trung bình của các giá trị lỗi trên là 0 và độ lệch chuẩn của chúng là 0.138. Giá trị $t_{[0.95;7]}$ từ bảng A.4 là 1.895. Do vậy, khoảng tin cậy cho lỗi trung bình là:
$0\,\mathrm{ml}.895 \times 0.138 = 0\,\mathrm{m}0.262 = (-0.262, 0.262)$

1.3.3 Kiểm tra một giá trị trung bình ZERO

Một ứng dụng thông thường của khoảng tin cậy là để kiểm tra xem một giá trị đo đạc có khác Zero hay không?. Khi so sánh một phép đo đạc ngẫu nhiên với Zero, các phát biểu cần mang tính xác suất, nghĩa là ở một độ mức tin tưởng nhất định nào đó. Nếu giá trị đo đạc thỏa mãn phép kiểm tra sự khác biệt với một xác suất lớn hơn hoặc bằng mức tin cậy 100(1 - ±)% thì giá trị đó là khác 0. Khâu kiểm tra bao gồm xác định một khoảng tin vậy và đơn giản xác định xem khoảng đó có chứa giá trị 0 hay không. Bốn trường hợp được chỉ ra trên hình 13.3, CI viết tắt cho confidence interval (khoảng tin cậy). CI được thể hiện bằng một đoạn thẳng đứng giữa giới hạn tin cậy trên và dưới. Kỳ vọng mẫu được thể hiện bằng một vòng tròn nhỏ. Trong trường hợp (a) và (b), khoảng tin cậy bao gồm giá trị 0, do đó, giá trị của phép đo có thể không khác 0. Trong trường hợp (c) và (d), khoảng tin cậy không chứa giá trị 0, và do đó giá trị đo đạc là khác giá trị 0.



Hình 1.3.3 Kiểm tra một giá trị trung bình Zero

Ví dụ 1.3.3.1 Sự khác nhau giữa thời gian xử lý của hai cách cải thiện khác nhau của cùng một thuật toán được đo trên 7 tải làm việc giống nhau. Các giá trị khác biệt là {1.5, 2.6, -1.8, 1.3, -0.5, 1.7, 2.4}. Chúng ta có thể nói với 99% độ tin cậy rằng một cách cải thiện là tốt hơn cách kia hay không?
Kích thước mẫu = $n = 7$
Giá trị trung bình = 7.20/7 = 1.03
Phương sai mẫu (sample variance) = (22.84 - 7.20*7.20/ 7)/6 = 2.57
Độ lệch chuẩn mẫu = $\sqrt{2.57} = 1.60$
Khoảng tin cậy = $1.03\,\mathrm{m}t * 1.60 / \sqrt{7} = 1.03\,\mathrm{m}0.6t$
$100(1-\alpha) = 99, \alpha = 0.01, 1-\alpha/2 = 0.995$
Tra bảng A.4 trong phụ lục ta có giá trị $t_{[0.995;6]} = 3.707$, và khoảng tin cậy 99% = (-1.21, 3.27).
Khoảng tin cậy chứa giá trị 0. Do đó, chúng ta không thể nói với 99% độ tin tưởng rằng giá trị khác nhau trung bình là khác 0 được.
Thủ tục để kiểm tra giá trị trung bình 0 có thể áp dụng tốt cho bất cứ các giá trị nào khác. Ví dụ, để kiểm tra xem kỳ vọng có bằng một giá trị *a* hay không, một khoảng tin cậy được lập ra và nếu khoảng tin cậy chứa giá trị *a* thì giả thiết cho rằng kỳ vọng bằng *a* không thể bị loại bỏ ở một mức độ tin cậy. Ví dụ sau sẽ phác họa sự mở rộng của phép kiểm tra này.

Ví dụ 1.3.3.2. Xét một lần nữa dữ liệu trong ví dụ 1.3.3.1 Để kiểm tra xem sự khác nhau về giá trị có bằng 1 với mức tin cậy 99% hay không, khoảng bảo vệ được xác định trong ví dụ đó là (-1.21, 3.21). Khoảng tin cậy này chứa 1. Do đó, một giá trị khác nhau bằng 1 được công nhận với mức tin cậy đó.

Chương 2- Tổng quan về kỹ thuật mô phỏng-

Tác giả: R. Jain

Dịch thuật: Tô Thành Công, Hoàng Diệp Anh

Biên tập: Nguyễn Xuân Hoàng

Phần này sẽ đưa ra các vấn đề trọng tâm trong mô hình hóa mô phỏng. Dưới là danh sách các câu hỏi mà độc giả có thể trả lời được sau khi đọc xong phân này.

- Các lỗi phổ biến trong mô phỏng là gì và tại sao khi thực hiện mô phỏng thường gặp các lỗi này ?
- Ngôn ngữ bạn nên sử dụng để phát triển một mô hình mô phỏng là gì ?
- Các loại mô phỏng?
- Bạn sẽ lập lịch (schedule) các sự kiện trong quá trình mô phỏng như thế nào ?
- Bạn sẽ kiểm tra và xác thực một mô hình như thế nào ?
- Làm cách nào bạn xác định được một mô phỏng đã đạt tới một trạng thái ổn định?
- Thời gian mô phỏng sẽ mất bao nhiêu lâu?
- Bạn sẽ tạo ra các số ngẫu nhiên như thế nào ?
- Bạn sẽ kiểm tra lại bộ tạo số ngẫu nhiên có tốt hay không như thế nào ?
- Bạn sẽ lựa chọn số ngẫu nhiên bắt đầu như thế nào từ bộ tạo số ngẫu nhiên như thế nào ?
- Bạn tạo số ngẫu nhiên theo một phân bố cho trước nhu thế nào?
- Bạn sử dụng những phân bố gì và khi nào thì sử dụng chúng?

Kiến thức đưa ra trong phần này sẽ giúp bạn thực hiện việc mô phỏng chính xác và thành công.

Mô phỏng là kỹ thuật có ích cho công việc phân tích hiệu năng của hệ thống máy tính. Nếu hệ thống chưa sẵn sàng để sử dụng, giống như trong các giai đoạn thiết kế thì một mô hình mô phỏng sẽ là một cách cho phép dễ dàng dự đoán hiệu năng của hệ thống hoặc so sánh các hệ thống khác nhau. Hơn nữa, ngay cả khi hệ thống đã có sẵn để đo thì một mô hình mô phỏng thường được ưa thích hơn bởi vì nó cho phép so sánh các cấu hình khác nhau khi thay đổi về môi trường và tải của hệ thống. Các tiêu chí để lựa chọn mô phỏng, mô hình hóa phân tích, và đo lường đã được thảo luận trước.

Tuy nhiên, các mô hình mô phỏng thường gặp thất bại, tức là chúng cung cấp các kết quả không có ích hoặc làm sai kết quả. Sở dĩ gặp phải thất bại này bởi vì những người xây dựng các mô hình là các chuyên gia trong lĩnh vực phát triển phần mềm nhưng họ lại thiếu kiến thức nền tảng về thống kê , hoặc họ là các chuyên gia trong kỹ thuật thống kê nhưng họ lại không có các kỹ thuật tốt về phát triển phần mềm. Hơn nữa việc mô phỏng đã bị dừng trước khi hoàn thành. Đó là vì các mô hình mô phỏng cần một thời gian khá lâu để phát triển, lâu hơn rất nhiều thời gian dự đoán lúc ban đầu.

2.1 Các lỗi thường gặp trong mô phỏng

1. *Mức độ chi tiết không hợp lý:* Mô phỏng cho phép nghiên cứu một hệ thống chi tiết hơn mô hình phân tích.. Việc phân tích đòi hỏi nhiều đơn giản hóa và giả thiết. Hay nói cách khác, mô hình phân tích là kém chi tiết hơn. Trong mô hình mô phỏng, mức độ chi tiết chỉ bị hạn chế bởi thời gian giành cho việc phát triển mô phỏng. Mô phỏng càng chi tiết yêu cầu càng nhiều thời gian để phát triển. Khả năng có lỗi sẽ tăng cao và sẽ khó khăn hơn để phát hiện ra các lỗi này. Thời gian gỡ rối của các lỗi này cũng sẽ tăng. Mô phỏng chi tiết hơn cũng đòi hỏi máy tínhcần nhiều thời gian hơn để thực hiện. Điều này đặc biệt quan trọng đối với các công việc mô phỏng lớn, ở đó thời gian thực hiện có thể mất nhiều giờ hoặc nhiều ngày.

Về tổng quát có thể cho rằng một mô hình càng chi tiết thì càng tốt, vì nó chỉ cần ít giả thiết hơn. Tuy nhiên điều này không phải lúc nào cũng đúng. Một mô hình chi tiết yêu cầu các kiến thức chi tiết của thông số đầu vào, và nếu không có các kiến thức chi tiết đó có thể sẽ làm cho các mô hình không chính xác. Ví dụ, trong quá trình mô phỏng hệ thống phân chia thời gian, giả sử rằng cần phải mô phỏng thời gian đáp ứng yêu cầu của đĩa . Một cách là tạo ra nó bằng cách sử dụng một hàn phân bố hàm mũ. Một cách chi tiết hơn là mô phỏng sự chuyển động của đầu đọc và sự quay của đĩa Nếu lựa chọn thứ 2 được chọn thì kết quả sẽ chỉ chính xác hơn khi các thông tin cần tham chiếu của sector và track là đã biết . Trong thực tế, người phân tích sẽ chọn lựa chọn 2. Tuy nhiên, khi thông tin tham khảo về sector không có sẵn để đưa vào mô hình thì người phân tích sẽ quyết định tạo ra các con số sector ngẫu nhiên theo phân bố hàm mũ. Kết quả là thời gian phục vụ là theo phân bố hàm mũ, một kết quả mà có thể sẽ ít tốn kém hơn nếu lựa chọn theo cách đầu tiên

Một vấn đề khác nữa trong các mô hình chi tiết là chúng tốn nhiều thời gian để phát triển. Tốt hơn hết là hãy bắt đầu với một mô hình ít chi tiết hơn, thu được một số kết quả, xem xét các tác động, sau đó đưa thêm chi tiết trong những yếu tố có ảnh hưởng lớn nhất đến kết quả.

2. *Ngôn ngữ không thích hợp:* Việc lựa chọn ngôn ngữ lập trình có ảnh hưởng khá quan trọng đối với giai đoạn đầu phát triển của mô hình. Các ngôn ngữ mô phỏng với mục đích đặc biệt cho phép rút ngắn thời gian phát triển mô hình và dễ dàng hơn trong các nhiệm vụ thường phải thực hiện như việc xác nhận (sử dụng các vết) và các phân tích thống kê. Mặt khác, các ngôn ngữ có mục đích tổng quan có tính khả chuyển cao hơn và cho phép điều khiển nhiều hơn về mức độ hiệu quả và thời gian thực hiện của mô phỏng.

3. *Các mô hình chưa được kiểm tra* . Các mô hình mô phỏng thường là các chương trình máy tính lớn và trừ khi được đề phòng đặc biệt, nó có khả năng sẽ có nhiều lỗi chương trình hoặc lỗi về lập trình, dẫn đến việc cho ra các kết quả không có ý nghĩa.

4.*Các mô hình không hợp lệ.* Ngay cả khi chương trình mô phỏng không có lỗi,nó vẫn có thể biểu diễn hệ thống không chính xác bởi vì các giả thiết không chính xác về hành vi của hệ thống. Do đó các mô hình cần phải được kiểm tra để đảm bảo rằng kết quả có được là giống với kết quả có được từ hệ thống thật Kết quả của tất cả các mô hình mô phỏng này đều có thể sai, nó chỉ đúng sau khi đã được xác nhận bằng các mô hình phân tích, các phép đó, hoặc bằng trực giác.

5. *Điều kiện xử lý ban đầu không thích hợp.* Phần đầu tiên của mô phỏng thường không miêu tả các hoạt động của hệ thống trong trạng thái ổn định. Bởi vậy phần ban đầu của mô phỏng thường được loại bỏ.

6. *Sự mô phỏng quá ngắn.* Người phân tích thường cố gắng tiết kiệm thời gian của mình và thời gian của máy tính bằng cách thực hiện các chương trình mô phỏng trong thời gian ngắn. Kết quả trong các trường hợp này bị phụ thuộc rất nhiều vào các điều kiện ban đầu và không thể hiện được hệ thống thực. Thời gian giành cho mô phỏng phụ thuộc vào mức độ chính xác mong muốn (độ rộng của khoảng cách tin cậy) và sự thay đổi của các đại lượng quan sát.

Các tính toán đơn giản cho sự thay đổi từcác giá trị của một biến ngẫu nhiên không đem lại một ước lượng chính xác về các thay đổi trong mô hình mô phỏng bởi vì có sự tương quan giữa các giá trị.

7. *Hàm sinh số ngẫu nhiên quá kém.* Mô hình mô phỏng đòi hỏi các số ngẫu nhiên, thủ tục tạo ra các số ngẫu nhiên được gọi là các hàm sinh số ngẫu nhiên. Sẽ an toàn hơn nếu sử dụng các hàm sinh nổi tiếng đã được phân tích kỹ lưỡng hơn là phát triển các hàm sinh riêng của mình Thậm chí các hàm sinh nổi tiếng này cũng đã có lỗi.

8. *Sự lựa chọn khởi đầu không hợp lệ.* Hàm sinh số ngẫu nhiên là các thủ tục máy tính, nó tạo ra một con số ngẫu nhiên từ một con số ngẫu nhiên khác cho trước. Số ngẫu nhiên đầu tiên trong dải số đó được gọi là hạt giống và được đưa ra bởi các nhà phân tích. Con số ngẫu nhiên đầu tiên cho các chuỗi số ngẫu nhiên cần phải được lựa chọn một cách cẩn thận để đảm bảo tính độc lập giữa các chuỗi số ngẫu nhiên. Thông thường, các nhà phân tích dùng chung một chuỗi số cho các quá trình xử lý khác nhau hoặc sử dụng cùng một giá trị ban đầu giống nhau cho tất cả các chuỗi (thường thường người ta khởi tạo bằng 0). Nó tạo ra sự tương quan giữa các quá trình khác nhau trong hệ thống và dẫn đến các kết quả có thể sẽ không thể hiện được hệ thống thực.

2.2 Các lý do khác khiến mô phỏng bị sai

1. Sự ước lượng thời gian không hợp lý: Nguyên nhân chính và trước tiên khiến cho các mô hình mô phỏng thất bại là các nhà phát triển đánh giá không đúng thời gian và nỗ lực cần thiết để pháp triển một mô hình mô phỏng. Thời gian giành cho các dự án mô phỏng thường được bắt đầu là các dự án 1 tuần, 1 tháng và sau đó thì kéo dài đến một vài năm. Nếu một mô phỏng thành công và cung cấp nhiều thông tin hữu ích, các người dùng sẽ muốn thêm vào nhiều tính năng, nhiềuthông số và chi tiết hơn. Mặt khác thì, nếu một chương trình mô phỏng không cung cấp thông tin hữu ích thì nó thường được mong muốn rằng nếu bổ xung thêm nhiều tính năng, nhiều thông số và chi tiết hơn có thể làm chương trình có ích hơn. Trong bất kỳ tình huống nào kể trên thì các dự án thường kéo dài hơn so với kế hoạch ban đầu.

Trong 3 kỹ thuật phân tích hiệu năng : mô hình hóa phân tích, đo đạc và mô phỏng thì việc thực hiện mô phỏng là cần nhiều thời gian nhất, nhất là đối với các mô hình hoàn toàn mới thì quá trình phát triển phải bắt đầu từ con số không. Ngoài ra cũng cần có thời gian để xác thực lại mô hình. Những người phân tích mới đối với lĩnh vực này thường không đánh giá đúng về độ phức tạp khi triển khai mô phỏng. Đối với các dự án mô phỏng trong thời gian dài, cần phải chuẩn bị trước cho

những thay đổi trong hệ thống, điều khó có thể tránh khỏi đối với các dự án mô phỏng trong thời gian dài.

2. Mục tiêu không thể thực hiện được. Mô phỏng là một kế hoạch khá phức tạp, cũng giống như các kế hoạch khác,cần phải xác định rất rõ ràng các mục tiêu với các đặc tính như cụ thể, đo lường được, có thể thực hiện được, có thể lặp lại được và toàn diện. Mục tiêu cần được viết ra rõ ràng và có sự thống nhất giữa người phân tích và người sử dụng về kết quả trước khi phát triển mô hình

Một ví dụ phổ biến của mục tiêu đo lường được là " mô hình X". Có thể mô hình các đặc tính khác nhau của X với các mức độ chi tiết khác nhau.được cấu thành từ một vài mô hình có mức độ chi tiết khác nhau. Nếu không có quy định đúng, không thể nói rằng mục tiêu đã đạt được hay chưa. Kế hoạch mà không có mục tiêu sẽ kéo dài mãi và cuối cùng sẽ kết thúc khi tài trợ đã hết.

3. **Thiếu các kỹ năng cần thiết.** Một dự án mô phỏng cần ít nhất bốn kỹ năng sau

(a) Kỹ năng về lãnh đạo dự án. Có khả năng tạo ra động lực, lãnh đạo và quản lý các thành viên trong nhóm mô phỏng.

(b) Kỹ năng về mô hình hóa và thống kê thông tin. Có khả năng chỉ ra các đặc trưng chủ chốt của hệ thống và mô hìnhchúng ở mức độ chi tiết cần thiết.

(c) Kỹ năng về lập trình. Là khả năng viết ra một chương trình máy tính có thể đọc được, kiểm tra được để thực hiện mô hình chính xác.

(d) Am hiểu về hệ thống được mô hình mô hình hệ thống. Là khả năng hiểu về hệ thống, giải thích được cho nhóm mô phỏng về hệ thống, iễn giải các kết quả mô phỏng dưới dạng ảnh hưởng của chúng lên thiết kế hệ thống

Một nhóm mô phỏng nên bao gồm các thành viên có các kỹ năng trên, và người đứng đầu lý tưởng nhất là người có đủ 4 kỹ năng trên

4. Mức độ tham gia của người dùng chưa hợp lí : Nhóm mô phỏng và các tổ chức của người dùng cần phải được gặp mặt định kỳ và thảo luận về tiến độ, các vấn đề, các sự thay đổi nếu có trong hệ thống. Hầu hết các hệ thống tiến triển hoặc thay đổi theo thời gianvà một mô hình được phát triển mà không có sự tham gia của người sử dụng thì hiếm khi thành công. Các cuộc gặp mặt định kỳ sẽ giúp tìm ra các lỗi trong mô hình từ giai đoạn đầu và cho phép mô hình được thống nhất với các thay đổi của hệ thống.

5. Các tài liệu hướng dẫn đã lỗi thời hoặc không có thực. Hầu hết các mô hình mô phỏng đều tiến triển trong một thời gian rất dài và liên tục được thay đổi khi hệ thống bị thay đổi hoặc đã được tìm hiểu ở mức tốt hơn . Các tài liệu về các mô hình này thường là cũ quá so với sự phát triển, trừ khi nó được quan tâm đặc biệt không thì sẽ bị lỗi thời. Chiến lược tốt nhất bao gồm cả tài liệu bản thân trong chương trình và sử dụng ngôn ngữ máy tính có thể đọc một cách dễ dàng hơn.

6. Không đủ khả năng để quản lý một chương trình máy tính có độ phức tạp lớn. Một số công cụ kỹ thuật phần mềm sẵn có cho quản lí các dự án phần mềm lớn. Các công cụ này giúp ta giữ đúng định hướng của việc thiết kế các đối tượng, các yêu cầu về chức năng, cấu trúc dữ liệu và sự ước lượng về tiến độ công việc. Ngoài ra còn một số nguyên tắc thiết kế như nguyên tắc về thiết kế từ trên xuống và nguyên tắc về lập trình cấu trúc đã được phát triển để giúp đỡ sự phát triển của các

chương trình máy tính lớn theo thứ tự. Nếu không sử dụng các công cụ và kỹ thuật này thì sẽ không thể có sự thành công cho việc phát triển một mô hình mô phỏng lớn.

7. Kết quả khó hiểu: Phần lớn các kết quả mô phỏng khó hiểu là do có lỗi trong chương trình mô phỏng,các giả thiết về mô hình không hợp lý hoặc không có các kiến thức về hệ thống thực tế. Bởi vậy người phát triển mô hình cần phải kiểm tra lại mô hình và nếu vẫn còn các kết quả khó hiểu thì hãy thông báo cho người sử dụng chú ý. Nó có thể cung cấp những cái nhìn thấu đáo hơn các hoạt động của hệ thống hoặc các đặc tính của hệ thống cần được chi tiết hóa hơn nữa.

Danh sách dưới gồm 3 danh sách con tương ứng với các giai đoạn lập kế hoạch, phát triển, sử dụng của một dự án mô phỏng.

Bảng 2.1 Danh sách cần KIỂM TRA về các MÔ PHỎNG

1.Các việc cần kiểm tra trước khi phát triển mô phỏng.

(a) Mục tiêu của mô phỏng trên lý thuyết đã được xác định hợp lý ?

(b) Mức độ chi tiết trong mô hình thích hợpvới mục đích hay không ?

(c) Nhóm mô phỏng có bao gồm đầy đủ các thành viên với các vị trí: lãnh đạo, mô hình hóa, lập trình và am hiểu về hệ thống máy tính hay không ?

(d) Có đủ thời gian cho kế hoạch của dự án hay không ?

2.Kiểm tra trong quá trình phát triển.

(a) Tính độc lập và tính đơn trị của bộ sinh số ngẫu nhiên được sử dụng trong mô phỏng đã được kiểm tra hay chưa ?

(b) Mô hình có được xem xét thường xuyên bởi người sử dụng hay không?

(c) Có tài liệu của mô hình hay không ?

3.Kiểm tra sau khi thực hiện mô phỏng.

(a) Độ dài của mô phỏng có thích hợp hay không ?

(b) Quá trình chuyển tiếp của giai đoạn đầu đã đượclooại bỏ trước khi tính toán hay chưa

(c) Mô hình đã được kiểm tra kỹ lưỡng hay chưa ?

(d) Mô hình đã được xác nhận là hợp lý trước khi sử dụng kết quả của nó hay chưa ?

(e) Nếu có kết quả đáng ngạc nhiên, xem chúng đã được kiểm duyệt hay chưa ?

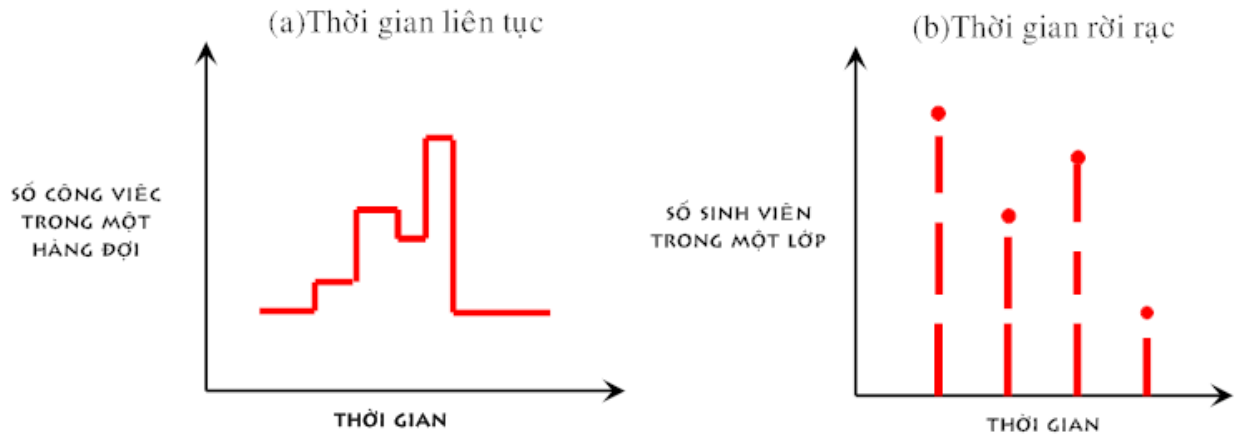
(f) Các bộ số đầu tiên (hạt giống)của các dải số ngẫu nhiên có bị trùng lặp hay không ?

2.3 Các thuật ngữ

Dưới đây là một số thuật ngữ thường được sử dụng trong mô hình hóa. Để định nghĩa chúng, một ví dụ mô phỏng về việc lập lịch của CPU sẽ được sử dụng trong phần này. Vấn đề đặt ra là nghiên cứu các kỹ thuật lập lịch khác nhau cho CPU với những đặc điểm về yêu cầu công việc khác nhau. Các thành phần khác của hệ thống như phương tiện ghi nhớ, thiết bị đầu cuối sẽ bị bỏ qua trong phần này.

Biến trạng thái: Các biến mà giá trị của nó để xác định các trạng thái của hệ thống gọi là biến trạng thái. Nếu một chương trình mô phỏng bị dừng lại ở giữa chừng, nó có thể được khởi

động lại nếu chỉ khi các giá trị của biến trạng thái đã được biết trước Trong một chương trình mô phỏng lập lịch cho CPU biến trạng thái là độ dài hàng đợi các công việc.

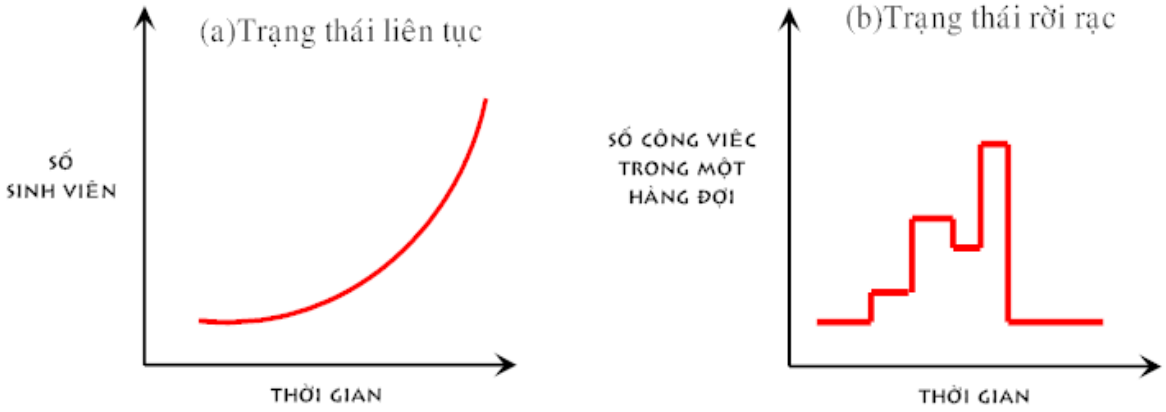


Hình 2.1 Mô hình thời gian rời rạc và thơi gian liên tục

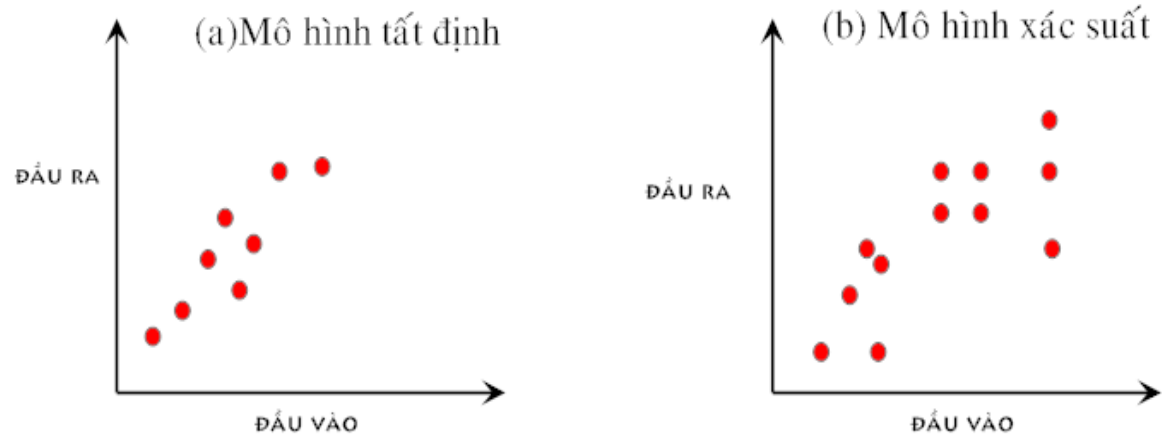
Sự kiện: Một thay đổi trong trạng thái của hệ thống gọi là *sự kiện.* Trong việc mô phỏng lập lịch cho CPU có 3 sự kiện như sau: sự kiện tới của một công việc, bắt đầu thực hiện một chương trình và đích đến của công việc.

Mô hình thời gian liên tục và thời gian rời rạc Một mô hình mà trạng thái hệ thống được xác định trong mọi thời điểm gọi là mô hình thời gian liên tục. Ví dụ mô hình lập lịch của CPU là một mô hình liên tục. Nếu trạng thái một hệ thống được định nghĩa chỉ tại một thời điểm cụ thể thì mô hình đó gọi là mô hình rời rạc. Ví dụ như một lớp học thường gặp mặt vào các chiều thứ 6. Giả sử trong mô hình, trạng thái của lớp được quy định là số sinh viên tham dự lớp. Chú ý rằng sĩ số lớp chỉ được xácc định vào các thứ 6. Trong tất cả các ngày khác, trạng thái của lớp là không xách định. Đây là một ví dụ về mô hình thời gian rời rạc. Hình 2.1 thể hiện 2 kiểu mô hình trên.

Mô hình trạng thái liên tục và trạng thái rời rạc: Một mô hình được gọi là mô hình có trạng thái liên tục hoặc trạng thái rời rạc phụ thuộc vào các biến trạng thái là liên tục hay rời rạc. Biến liên tục có thể lấy các giá trị vô hạn không thể đếm được. Ví dụ, trong một mô hình về lớp học hàng tuần, trạng thái được xác định bằng thời gian sinh viên tham dự một môn học,là mô hình trạng thái liên tục. Mặt khác, nếu trạng thái được xác định bởi số sinh viên thì nó là mô hình trạng thái rời rạc. Trong mô hình lập lịch cho CPU, biến trạng thái – chiều dài hàng đợi- có thể chỉ được gán giá trị nguyên. Bởi vậy mô hình trạng thái rời rạc ở hình 2.2 cũng có thể gọi là mô hình sự kiện rời rạc. Tương tự như vậy mô hình trạng thái liên tục cũng có thể gọi là mô hình sự kiện liên tục.



Hình 2.2 Mô hình trạng thái liên tục và trạng thái tiếp diễn



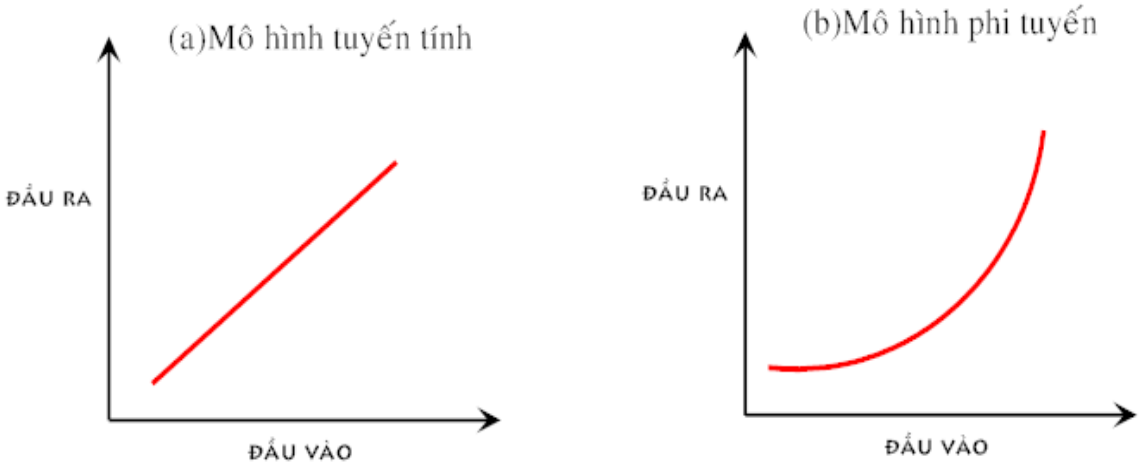
Hình 2.3 Mô hình tất định và mô hình xác suất

Chú ý rằng tính liên tục của thời gian không bao hàm tính liên tục của trạng thái và ngược lại. Vì vậy đối với các ví dụ có thể tìm thấy nhiều ví dụ cho 4 kết hợp có thể xảy ra đó là: trạng thái rời rạc/thời gian rời rạc, trạng thái rời rạc /thời gian liên tục, trạng thái liên tục/ thời gian rời rạc và trạng thái liên tục/ thời gian liên tục.

Mô hình tất định và mô hình xác xuất: Nếu đầu ra (kết quả) của mô hình có thể dự đoán một cách chắc chắn thì đó là một mô hình tất định. Mặt khác, một mô hình là mô hình xác xuất nếu với lặp lại cùng một bộ thông số đầu vào nhưng cho các kết quả khác nhau . Điều này thể hiện trong hình 24.3. Hình 24.3b là kết quả của mô hình xác xuất. Các điểm trên đường thẳng đứng bi ểudiễn các kết quả của đầu ra khác nhau cho một giá trị đầu vào. Trong hình 2.3, với đầu vào giống nhau thì cho đầu ra giống nhau, do vậy trên trục tung chỉ có một điểm.

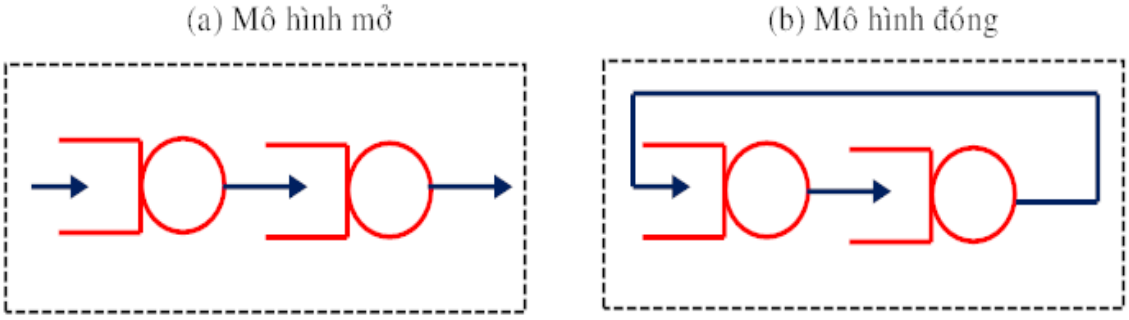
Mô hình tĩnh và mô hình động: Một mô hình mà trong đó thời gian không là biến số thì được gọi là mô hình tĩnh, còn nếu trạng thái của hệ thống thay đổi theo thời gian thì gọi là mô hình động. Ví dụ, mô hình lập lịch cho CPU là mộ mô hình động. Một ví dụ về mô hình tĩnh là mô hình cho thể hiện công thức tính năng lượng: $E=mc^2$.

Mô hình tuyến tính và mô hình phi tuyến: Nếu mô hình có thông số đầu ra là một hàm tuyến tính của các thông số đầu vào thì đó là mô hình tuyến tính, nếu không thì đó là mô hình phi tuyến, hãy xem hình 2.4 dưới đây.



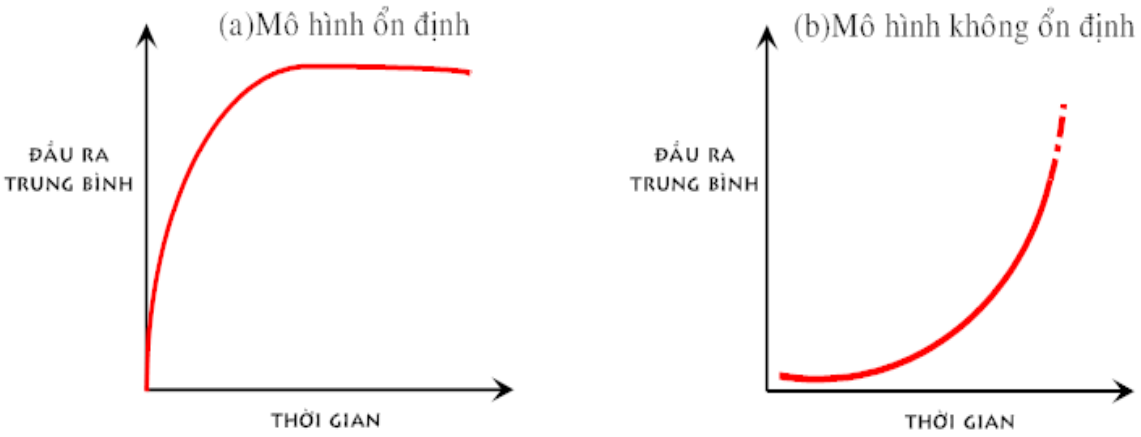
Hình 2.4 Mô hình tuyến tính và mô hình phi tuyến

Mô hình mở và mô hình đóng: Nếu tham số đầu vào là ở bên ngoài so với mô hình và độc lập với mô hình thì đó là mô hình mở. Mô hình đóng là mô hình có đầu vào ở bên trong nó. Hình 2.5 thể hiện hai mô hình hàng đợi của một hệ thống máy tính.Trong Hình 2.5b các công việc giống nhau được luân chuyển trong mô hình. Một công việc đi ra khỏi hàng đợi thứ 2 tiếp tục đi vào hàng đợi thứ nhất.. Đây chính là mô hình đóng. Còn hình 2.5a thể hiện một mô hình mở, trong đó các công việc mới được đưa vào mô hình.



Hình 2.5 Mô hình mở và mô hình đóng

Mô hình ổn định và không ổn định: Nếu các hoạt động động của mô hình ở mức trạng thái ổn định, đó là độc lập về mặt thời gian, thì nó được gọi là ổn định. Còn nếu một mô hình có các hoạt động liên tục thay đổi thì nó gọi là không ổn định. Những điều này được minh họa ở hình 2.6



Hình 2.6 Mô hình ổn định và không ổn định

Các mô hình hệ thống máy tính thường là các mô hình: thời gian liên tục, trạng thái rời rạc, xác suất, động và phi tuyến. Có một vài mô hình mở và một vài mô hình đóng. Ngoài ra các mô hình ổn định và không ổn định cũng được sử dụng.

2.4 Lựa chọn ngôn ngữ mô phỏng

Lựa chọn ngôn ngữ là bước quan trọng nhất trong tiến trình phát triển của một mô hình mô phỏng. Một quyết định sai trong bước này có thể làm cho thời gian thực hiện mô phỏng lâu, không hoàn thành được việc nghiên cứu và không khả thi

Có bốn lựa chọn: một ngôn ngữ mô phỏng, một ngôn ngữ lập trình đa dụng, ngôn ngữ lập trình đa dụng mở rộng, và gói mô phỏng (như là một bộ giải quyết mạng). Mỗi lựa chọn có ưu nhược điểm riêng.

Các ngôn ngữ mô phỏng tiết kiệm được cho người phân tích đáng kể thời gian khi phát triển mô hình. Những ngôn ngữ này có sẵn các chức năng để cải thiện về thời gian, lập lịch sự kiện, biến đổi các thực thể, tạo các biến ngẫu nhiên , thu thập dữ liệu thống kê, và tạo các báo cáo. Chúng cho phép các nhà phân tích danh nhiều thời gian vào các kết quả cụ thể của hệ thông được mô hình mà không cần phải quan tâm nhiều đến các vấn đề chung chung đối với các mô phỏng. Các ngôn ngữ này cũng cho phép một mã module rất dễ đọc, có thể phát hiện lỗi rất tốt.….

Ngôn ngữ lập trình đa dụng như Pascal hoặc FORTRAN được chọn chủ yếu bởi vì các nhà phân tích đã quen với ngôn ngữ này. Hầu hết các nhà thiết kế mạng máy tính và các nhà phân tích mới không quen với những loại ngôn ngữ mô phỏng. Bên cạnh đó các yêu cầu về thời hạn cũng không cho phép họ học một loại ngôn ngữ mô phỏng nào. Hơn nữa, các ngôn ngữ mô phỏng thường không có sẵn trên hệ thống máy tính của họ. Đó là lý do tại sao hầu hết mọi người viết mô phỏng đầu tiên của mình bằng ngôn ngữ lập trình đa dụng

Ngay cả đối với những người mới, thời gian chọn lựa giữa ngôn ngữ mô phỏng và ngôn ngữ lập trình đa dụng cũng không xảy ra. Nếu họ chọn ngôn ngữ mô phỏng, họ phải bỏ thời gian để học ngôn ngữ này. Trong một số trường hợp, họ thậm chí còn phải cài đặt nó vào trong hệ thống máy tính của họ và chú ý để không bỏ sót một file nhỏ nào trong quá trình cài đặt. Nếu họ chọn ngôn ngữ lập trình đa dụng, họ có thể bắt đầu ngay lập tức . Nhưng họ phải mất thời gian để thực hiện các thường trình để xử lý sự kiện, tạo số ngẫu nhiên và những thứ tương tự như thế. Có thể phải mất một khoảng thời gian đáng kể để tìm hiểu các vấn đề này và khám phá lại những vấn đề đã

biết.

Điều đó không có ý là các nhà phân tích lúc nào cũng phải sử dụng các ngôn ngữ mô phỏng. Có những điều phải quan tâm khác, như hiệu xuất, tính linh động và tính cơ động, những yếu tố này có thể làm cho ngôn ngữ lập trình đa dụng trở thành một sự lựa chọn tốt nhất. Một mô hình được phát triển bằng ngôn ngữ lập trình đa dụng thì hiệu xuất hợn và chiếm ít thời gian của CPU. Ngôn ngữ lập trình đa dụng mang đến cho các nhà phân tích khả năng mềm dẻo hơn vì nó cho phép họ sử dụng những đường tắt bị cấm trong ngôn ngữ mô phỏng. Hơn nưa, một mô hình được phát triển bằng ngôn ngữ lập trình đa dụng có thể chuyển đổi để thực hiện ở những hệ thống máy tính khác nhau một cách dễ dàng

Để có được một lựa chọn khách quan giữa ngôn ngữ mô phỏng và ngôn ngữ lập trình đa dụng, các nhà phân tích được khuyên cáo nên học ít nhất một ngôn ngữ mô phỏng để các yếu tố khác cùng với kiến thức sẽ giúp ích trong việc lựa chọn một ngôn ngữ thích hợp

Một mở rộng của ngôn ngữ lập trình đa dụng như GASP (đối với FORTRAN) là một ngôn ngữ thay thế. Các mở rộng này bao gồm một tập các thường trình để xử lý các nhiệm vụ thường được yêu cầu trong các mô phỏng. Mục đích của chúng là đem đến một thỏa hiệp dưới dạng hiệu xuất, tính linh động và tính cơ động.

Các gói mô phỏng như QNET và RESQ cho phép người dùng định nghĩa một mô hình sử dụng đối thoại. Các gói này có một thư viện cac cấu trúc dữ liệu, các thường trình và các giải thuật. Ưu điểm lớn nhất của chúng là tiết kiệm thời gian. Vì dụ, sử dụng gói mô phỏng, một người có thể phát triển mô hình, giải quyết và có được kết quả trong vòng một ngày. Mặt khác, phát triển một mô phỏng sử dụng một ngôn ngữ có thể mất một vài ngày (nếu không muốn nói là vài tháng), tùy thuộc vào độ phức tạp của hệ thống.

Vấn đề chính của các gói mô phỏng là tính cứng nhắc. Chúng chỉ cung cấp các tính chất mềm dẻo đã được thấy trước bởi những người thiết kế ra chúng. Trong hầu hết trường hợp trong thực tế, các nhà phân tích gặp phải một số vấn đề này khác không thể mô hình được bằng gói mô phỏng. Điều này bắt buộc các nhà phân tích phải làm đơn giản hóa các quá trình. Tuy nhiên, đối với các hệ thống không thể mô hình theo phương pháp phân tích, sẽ tiết kiệm thời gian hơn rất nhiều nếu nhà phân tích xem xét khả năng sử dụng một gói mô phỏng trước khi bắt đầu phát triển một mô hình mô phỏng mới

Các ngôn ngữ mô phỏng có thể được phân loại vào 2 danh sách chính, những ngôn ngữ mô phỏng liên tục và những ngôn ngữ mô phỏng sự kiện rời rạc, phụ thuộc vào loại sự kiện mà chúng mô phỏng. Các ngôn ngữ mô phỏng liên tục được thiết kế để xử lý những mô hình sự kiện liên tục thường được diễn tả bằng các phương trình vi phân. Ví dụ về loại này là cáci ngôn ngữ CSMP và DYNAMO. Những ngôn ngữ này thường phổ biến trong các mô hình hệ thống hóa học. Mặt khác, các ngôn ngữ mô phỏng sự kiện rời rạc được thiết kế để xử lý những thay đổi sự kiện dời rạc. Hai ví dụ của loại ngôn ngữ này là SIMULA và GPSS. Một số ngôn ngữ như SIMSCRIPT và GASP cho phép các mô phỏngrời rạc, liên tục, và kết hợp. Bốn ngôn ngữ sau là các loại ngôn ngữ được các nhà phân tích hiệu suất hệ thông máy tính sử dụng.

2.5 Các loại mô phỏng

Trong các loại mô phỏng khác nhau được để cập đến trong các tài liệu, những loại như mô phỏng Monte Carlo, mô phỏng Trace-Driven và mô phỏng sự kiện dời dạc là những loại mô

phỏng thu hút được nhiều hứng thú nhất của các nhà khoa học máy tính

Mô phỏng sử dụng phần cứng hoặc phần sụn gọi là sự giả lập. Ví dụ, một bộ giả lập đầu cuối mô phỏng một loại đầu cuối trên một đầu cuối khác. Giả lập bộ xử lý giả lập một tập lệnh của một bộ xử lý trên một bộ khác. Mặc dù giả lập là một loại của mô phỏng, các vấn đề thiết kế cho giả lập hầu hết là các vấn đề về thiết kế phần cứng. Do đó, giả lập sẽ không được để cập đến trong tài liệu này nữa.

Ba loại mô phỏng khác được miêu tả ở đoạn sau.

2.5.1 Phương pháp mô phỏng Monte Carlo

Phương pháp mô phỏng tĩnh hay một phương pháp mô phỏng nào đó không có trục thời gian thì được gọi là phương pháp mô phỏng Monte Carlo. Những phương pháp thường được dùng để mô hình những hiện tượng xác suất , những hiện tượng không thay đổi đặc tính theo thời gian. Giống như một phương pháp mô phỏng động, các phương pháp mô phỏng tĩnh cũng cần phải có một bộ tạo các số giả ngẫu nhiên. Phương pháp mô phỏng Monte Carlo cũng được sử dụng để tính toán các biểu thức không theo sắc xuất bằng cách sử dụng các phương pháp theo sắc xuất.

Ví dụ. 2.5.1 Tính toán phép tích phân sau

$$I = \int_0^2 e^{-x^2} dx$$

Một cách để tính phép tích phân trên là tạo ra các số ngẫu nhiên được phân bố đều x và đối với từng số tính toán được một hàm y như sau:

$$x \sim \text{Uniform}(0,2)$$
$$\text{Density function } f(x) = \tfrac{1}{2} \quad \text{iff} \quad 0 \le x \le 2$$
$$y = 2e^{-x^2}$$

Giá trị y mong muốn là:

$$\begin{aligned} E(y) &= \int_0^2 2e^{-x^2} f(x)\,dx \\ &= \int_0^2 2e^{-x^2} \tfrac{1}{2}\,dx \\ &= \int_0^2 e^{-x^2}\,dx \\ &= I \end{aligned}$$

Do đó, tích phân trên có thể được tính bằng cách tạo ra các số ngẫu nhiên được phân bố đều x_i, tính toán y_i, và sau đó tính trung bình cộng như sau:

$$x_i \sim \text{Uniform}(0,2)$$
$$y_i = 2e^{-x_i^2}$$
$$I = E(y) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

2.5.2 Phương pháp mô phỏng Trace - Driven

Mô phỏng sử dụng trace làm đầu vào là phương pháp mô phỏng trace-driven. Trace là một bản ghi các sự kiện được sắp xếp theo thời gian của một hệ thống thật. Các phương pháp mô phỏng trace-driven này khá thông dụng trong các phân tích hệ thống máy tính. Chúng thường được sử dụng để phân tích và điều chỉnh các giải thuật quản lý tài nguyên. Giải thuật paging, phân tích bộ nhớ cache, các giải thuật lập lịch CPU, các giải thuật ngăn chặn nghẽn và các giải thuật để phân chia động bộ nhớ là các ví dụ về các trường hợp đã áp dụng thành công phương pháp mô phỏng Trace-driven và được để cập đến trong các tài liệu Trong những nghiên cứu đó, Trace của tài nguyên yêu cầu được dùng làm đầu vào để thực hiện mô phỏng để mô hình những giải thuật khác nhau. Ví dụ, để thực hiện so sánh các lưu đồ quản lý bộ nhớ khác nhau, một trace các mẫu tham chiếu trang nhớ của các chương trình quan trọng sẽ được lấy trên hệ thống. Sau đó có thể sử dụng các trace này để tìm ra tập các tham số tối ưu đối với một giải thuật quản lý bộ nhớ cho trước hoặc để so sánh những giải thuật khác nhau.

Cần phải chú ý rằng các trace phải là độc lập với hệ thống đang nghiên cứu. Ví dụ, một trace của các trang nhớ được lấy ra từ một đĩa phụ thuộc vào qui mô công việc và các chính sách thay thế trang nhớ được sử dụng. Trace này không thể dùng để nghiên cứu các chính sách thay thế trang nhớ khác. Do đó, một nhà phân tích có thể cần một trace của các trang nhớ được tham chiếu. Tương tự như vậy, một trace lệnh được lấy từ một hệ điều hành không thể dùng để phân tích một hệ điều hành khác.

Những ưu điểm của phương pháp mô phỏng trace –driven như sau:

1. *Tính tin cậy*: Rất dễ dàng chuyển kết quả của phương pháp mô phỏng trace-driven cho các thành viên khác trong đội thiết kế. Ví dụ, một trace các tham chiếu trang nhớ có độ tin cậy cao hơn so với các tham chiếu được tạo ngẫu nhiên sử dụng một phân bố giả định.

2. *Dễ dàng kiểm tra:* Bước đầu tiên của phương pháp mô phỏng trace - driven là thực hiện giám sát hệ thông thực để thu các trace. Trong suốt quá trình giám sát này, nhà phân tích cũng có thể đo kiểm các đặc tính hiệu suất của hệ thống. Bằng việc so sánh hiệu suất đã đó được với hiệu suất có được khi mô phỏng, nhà phân tích có thể kiểm tra mô hình trace – driven một cách dễ dàng

3. Khối lượng công việc chính xác: Một trace duy trì các tác dụng tương quan và đan xen trong khối lượng công việc (tải). Việc đơn giản hoá như khi bắt đầu một mô hình phân tích về tải là không cần thiết.

4. *Cân bằng về mức độ chi tiết:* Do mức độ chi tiết trong tải là cao, có thể nghiên cứu ảnh hưởng của từng thay đổi nhỏ trong mô hình hoặc các giải thuật

5. *Ít ngẫu nhiên:* Trace là một đầu vào xác định. Nếu lặp lại mô phỏng, đầu vào trace là không đổi nhưng đầu ra có thể khác nhau do tính ngẫu nhiên ở những phần khác của mô hình. Nói chung, đầu ra của mô hỉnh trace- driven ít thay đổi hơn, điều đó có nghĩa là mô hình không cần phải lập lại nhiều lần một mô phỏng để có được kết quả tin cậy thống kê mong muốn. Nếu các phần khác của hệ thống cũng không ngẫu nhiên, thì có thể thu được kết quả chính xác trong một lần thực hiện mô phỏng của mô hình

6. So sánh công bằng: Trace cho phép những khả năng khác nhau được so sánh với cùng một luồng đầu vào. Đây là một phép so sánh công bằng hơn những mô hình mô phỏng có đầu vào được tạo ra

từ luồng ngẫu nhiên và khó mô phỏng các khả năng khác nhau.

7. Tương tự như triển khai thực sự: Một mô hình trace-driven nhìn chung là rất giống với hệ thống thực tế mà nó mô hình hoá. Do đó, khi thực hiện nó, nhà phân tích có thể cảm thấy rất thoải mái về độ phực tạp trong việc thực hiệngiải thuật đề xuất

Những nhược điểm của phương pháp mô phỏng trace-driven như sau:

1. *Phức tạp:* Mộ mô hình trace – driven yêu cầu mô phỏng chi tiết hơn về hệ thống. Đôi lúc độ phực tạp của mô hình làm lu mờ giải thuật đang được mô hình hoá

2. *Tính điển hình*: Các Traces được thưc hiện trong một hệ thống có thể không đại diện cho tải trong một hệ thông khác. Thậm chí trong một hệ thống tải có thể thay đổi theo thời gian, và vì vậy các trace trở nên vô dụng nhanh hơn những loại mô hình tải có thể hiệu chỉnh theo thời gian

3. *Tính hữu hạn*: Trace là một chuỗi dài. trace chi tiết về hoạt động trên một hệ thống trong vài phút có thể đủ để làm đầy một phần đĩa. Kết quả dựa trên những phút ít ỏi đấy không ứng dụng được cho các hoạt động trong khoảng thời gian còn lại của cả ngày

4. *Tính hợp lý ở từng điểm*: Trong khi sử dụng các trace để kiểm tra tính hợp lý, phải hết sức cẩn thận vì các trace chỉ cung cấp một điểm kiểm tra đơn lẻ. Một giải thuật là tốt nhất cho một trace có thể không hề tốt cho các trace khác. Nhà phân tích cần dùng nhiều trace khác nhau để kiểm tra các kết quả .

5. *Chi tiết*: Vấn đề chính của phương pháp mô phỏng trace-driven là độ chi tiết cao. Nhìn chung các trace là những chuỗi dài cần phải được đọc ra từ đĩa và sau đó cần phải hoàn thành việc tính toán cho từng phần tử của trace

6. *Cân bằng*: Đối với các trace, rất khó thay đổi các đặc tính tải. Bản thân một trace cũng không tự thay đổi nó được. Do đó để kết luận về sự ảnh hưởng của những thay đổi trong tải, thì cần phải có một trace cho tải bị thay đổi. Tương tự như vậy, nếu một trace chứa các đặc tính yêu cầu tài nguyên của một số công việc thì rất khó để nghiên cứu các ảnh hưởng tới từng công việc đơn lẻ.

2.5.3 Mô phỏng sự kiện rời rạc

Một mô phỏng sử dụng mô hình trạng thái rời rạc của hệ thống được gọi là phương pháp mô phỏng sự kiện rời rạc. Phương pháp này ngược với các phương pháp mô phỏng sự kiện liên tục ở chỗ trong mô phỏng sự kiện liên tục trạng thái của hệ thống lấy các giá trị liên tục. Các mô hình trạng thái liên tục được sử dụng trong các mô phỏng hóa học do trạng thái của hệ thống được mô tả bởi sự tập trung của một chất hóa học. Trong các hệ thống máy tính, các mô hình sự kiện rời rạc được sử dụng bởi vì trạng thái của hệ thống được mô tả bởi số lượng công việc ở những thiết bị khác nhau. Lưu ý rằng thuật ngữ "rời rạc" không dùng để chỉ giá chị thời gian được sử dụng trong mô phỏng. Phương pháp mô phỏng sự kiện rời rạc có thể sử dụng các giá trị thời gian liên tục hay rởi rạc.

Tất cả phương pháp mô phỏng sự kiện rời rạc đều có chung một cấu trúc. Bất kể hệ thống được mô hình là gì, thì phương pháp mô phỏng cũng sẽ có một số thành phần như sau: Nếu sử dụng ngôn ngữ lập trình đa dụng, thì các nhà phân tích phải tự phát triển tất cả các thành phần. Nguồn ngữ lập trình mô phỏng thì có thể cung cấp một vài thành phần còn lại các nhà phân tích phải tự phát triển. Các thành phần đó như sau

1. Bộ lập lịch sự kiện (Event Scheduler): Lưu trữ một danh sách liên kết các sự kiện xắp xẩy ra. Bộ lập lịch này cho phép tính toán các sự kiện theo nhiều cách khác nhau. Một số phép tính toàn như

sau:

(a) Lập lịch sự kiện X tại thời điểm T

(b) Giữ sự kiện X trong khoảng thời gian dt

(c) Hủy sự kiện X đã được lập lịch trước đó

(d) Giữ sự kiện X vô hạn định (Cho đến khi nó được một sự kiện khác lập lịch)

(e) Lập lịch cho một sự kiện đang bị giữ vô hạn định

Bộ lập lịch sự kiện là một trong những thành phần hoạt động thường xuyên nhất khi tiến hành mô phỏng.Nó được thực thi trước các sự kiện, và có thể được gọi nhiều lần trong một su kiện để lập lich các sự kiện mớikhác. Do lập lịch sự kiện có ảnh hưởng đáng kế tới hiệu xuất tính toán của một phương pháp mô phỏng, nên chủ đề này sẽ được đề cập nhiều hơn trong phần 24.6

2. Cơ chế *Simulation Clock and a Time-advancing*: Mỗi mô phỏng đều có một biến toàn cục đại diện cho thời gian được mô phỏng. Bộ lập lịch có trách nghiệm thực hiện trước thời gian này. Có 2 cách để thực hiện. Cách đầu tiên là phương pháp **thời gian đơn vị (unit time),** gia tăng thời gian bằng một gia tăng nhỏ và sau đo kiểm ra để tìm xem có sự kiện nào xuất hiện hay không. . Phương pháp thứ 2, được gọi là phương pháp **event-driven,** thời gian tự động tăng đến thời điểm có sự kiện tiếp theo gần nhất xẩy ra. Phương pháp thời gian đơn vị thường không được dùng trong các mô phỏng máy tính.

3. Các biến trạng thái hệ thống (*System State Variables)*: Là các biến toàn cục mô tả trạng thái của hệ thống. Ví dụ trong mô phỏng lập lịch CPU, biến trạng thái hệ thống là số lượng công việc có trong hang đợi. biến toàn cục này khác với các biến cục bộ như thời gian CPU cần để xử lý một công việc, là biến được lưu trong cấu trúc dữ liệu đại diện cho công việc đó.

4. Các thường trình của sự kiện (*Event Routines)*: Từng sự kiện được mô phỏng bằng thường trình của chính nó. Các thường trình này cập nhật các biến trạng thái hệ thống và lập lịch những sự kiện khác. Ví dụ, trong khi mô phỏng cơ chế lập lịch CPU, có thể cần đến các thường trình để xử lý 3 sự kiện là việc mới xuất hiển, lập lịch công việc, hoàn thành công việc

5. Các thường trình đầu vào (*Input Routines)*: Các thường chình này chứa các tham số mô hình, như yêu cầu về CPU trung bình từ người sử dụng trên một công việc. Vì cần nhiều thời gian để hoàn thành mô phỏng, sẽ tốt hơn nếu yêu cầu tất cả đầu vào ngay từ lúc đầu và sau đó thì giải phỏng người sử dụng. Các thường trình đầu vào thông thường cho phép các tham số thay đổi theo một cách cụ thể. Ví dụ mô phỏng có thể được thực hiện với yêu cầu CPU trung bình thay đổi từ 1 đến 9 ms trong các bước 2ms. Mỗi tập các giá trị đầu vào xác định một phép lặp có thể được lặp lại nhiều lần với những khởi đầu khác nhau. Do đó, một lần thực hiện mô phỏng có nhiều vòng lặp và mỗi vòng lặp lại bao gồm nhiều lần lặp lại.

6. *Bộ tạo báo cáo (Report Generator):* Chúng là các thường trình đầu ra được thực hiện vào giai đoạn cuối của mô phỏng. Chúng tính toàn kết quả cuối cùng và in ra theo một định dạng cụ thể.

7. Các thường trình khởi tạo (*Initialization Routines*): Chúng thiết lập trạng thái khởi tạo cho các biến trạng thái hệ thống và khởi tạo những luồng tạo số ngẫu nhiên khác nhau. Các thường trình này nên là các thường trình riêng biệt để khởi tạo trạng thái vào lúc bắt đầu mô phỏng, lúc bắt đầu một vòng lặp lại và lúc bắt đầu lặp.

8. *Các thường trình bám vết (Trace Routines):* Chúng in ra các tham số trung gian trong quá trình

thực hiện mô phỏng bắt đầu. Chúng giúp sửa lỗi chương trình mô phỏng. Các thường trình này nên có tính năng on/ off để có thể tắt chúng đi trong lần các lần chạy sản phẩm cuối cùng của mô hình. Một mô hình thậm chí còn có khả năng ngắt hoạt động từ bàn phím và bật/ tắt các thường trình bám vết.

9. Quản lý bộ nhớ động (*Dynamic Memory Management)*: Số thực thể trong mô phỏng thay đổi liên lục khi các thực thể mới được tạo ra và các thực thể cũ bị phá hủy, do đó cần phải dọn rác thường xuyên. Hầu hết ngôn ngữ mô phỏng và các ngôn ngữ lập trình đa dụng có chức năng dọn rác tự động. Trong các trường hợp khác, người lập trình phải tự viết code để quản lý bộ nhớ động

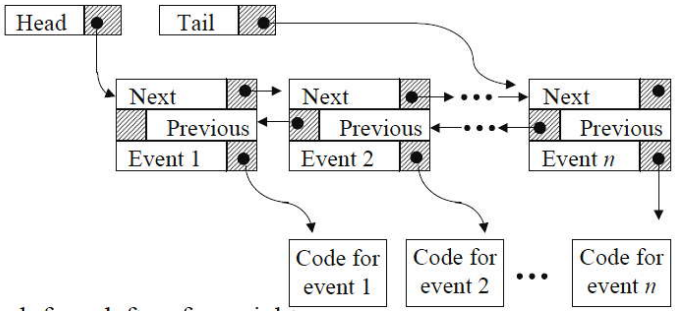
10. *Chương trình chinh (Main Program)*: Tập hợp tất các thường trình lại với nhau. Nó gọi các thường trình đầu vào, khởi tạo mô phỏng, thực hiện những lập lại khác nhau, và cuối cùng gọi các thường trình đầu ra

2.4.6 Các giải thuật thiết lập sự kiện

Trong các mô phỏng sự kiện rời rạc, phải đảm bảo rằng các sự kiện xuất hiện trong một trình tự và thời gian thích hợp. Hầu hết ngôn ngữ lập trình đều có chức năng tự động xắp sếp sự kiện này tuy nhiên đối với nhưng mô phỏng được viết bằng ngôn ngữ đa dụng, người lập trình phải thực hiện chức năng này. Đôi khi, đối với những mô phỏng sử dụng ngôn ngữ mô phỏng, nhà phân tích cũng thích sử dụng giải thuật xắp sếp của riêng họ. Ví dụ, trong một số trường hợp, hiệu quả hoạt động của những giải thuật này tiếp kiệm được khoảng 30% tổng tời gian của bộ xử lý

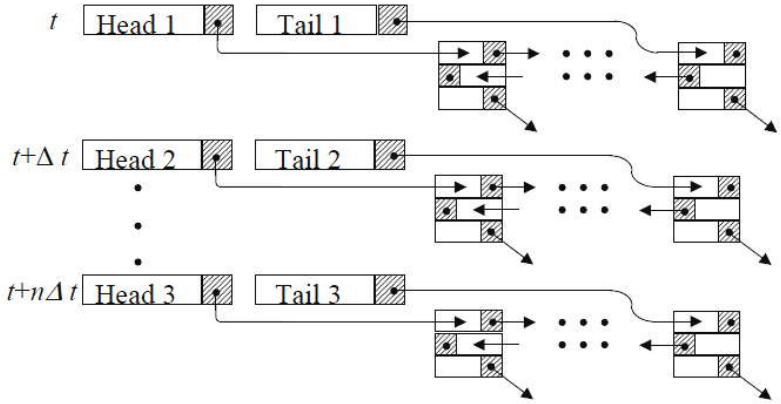
Lập lịch sự kiện thường được thực hiện bằng việc giữ các thông báo sự kiện theo một danh sách liên kết có thứ tự. Mỗi thông bao chứa thời gian xẩy ra sự kiện và một con trỏ tới mã có thể phải thi hành tại thời điểm đó. Có hai thao tác cần phải thực hiện thường xuyên trên tập này: Thứ nhất là chèn các sự kiện mới và thứ 2 là tìm sự kiện tiếp theo (xuất hiện sớm nhất) và xoá nó ra khỏi tập. Lựa chọn câu trúc dữ liệu để duy trì tập này ảnh hưởng đến thời gian xử lý cần thiết cho 2 hoạt động này. Một số cấu trúc dữ liệu cần ít thời gian để chèn nhưng yêu cầu xử lý đáng kể để tìm sự kiện tiếp theo. Các cấu trúc dữ liệu khác thực hiện tất cả những công việc tại thời điểm chèn để việc tìm sự kiện tiếp theo không hề phức tạp. Bởi vậy lựa chọn cấu trúc dữ liệu phụ thuộc vào tần suất chèn vào, xoá đi và vào số lượng trung bình của các sự kiện trong tập sự kiện tương lai. Một số cấu trúc dữ liệu đã được đề xuất như sau :

1. *Danh sách liên kết theo thứ tự (Ordered Linked List) :* Phương pháp phổ biến nhất được sử dụng trong các ngôn ngữ mô phỏng như SIMULA, GPSS, và GASP IV là những phương pháp có danh sách kiên kết theo thứ tự lớn gấp đôi. Đầu vào đầu tiên trong danh sách là sự kiện tiếp theo gần nhất. Đo đó, việc xoá bỏ tương đối dễ dàng Để chèn them sự kiện mới, danh sách được tìm kiếm để tìm ra vị trí thích hợp nhất cho một đầu vào mới. Một số phương pháp tìm kiếm khác đã được đề xuất. Phương pháp phổ biến nhất là tìm kiếm ngược từ giá trị thời gian cao nhất. Lần lượt, danh sách sẽ được tìm kiếm từ đầu vào đầu tiên trở đi. Một số phương pháp khác còn dữ con trỏ ở giữa danh sách, trược hết là xác định nửa danh sách chứa vị trí thích hợp, rồi sau đó tìm quyết định tìm xuôi hay tìm ngược để tìm vị trí đúng



Hình 2.7 Danh sách liên kết theo thứ tự

2. *Danh sách chỉ mục tuyến tính (Indexed Linear List):* Trong phương pháp này, tập các sự kiện tương lai được chia thành những tập nhỏ. Mỗi tập có độ dài cố định trong khoảng thời gian t và được duy trì như là một danh sách con. Một ma trận các chỉ mục được giữ theo qui tắc là đầu vào của chỉ số thứ *i* trỏ tới danh sách con thứ *i* chưa các sự kiện đã được lập lịch trong khoảng [(*i* - 1)”*t*, *i*”*t*),nghĩa là, tại hoặc sau thời điểm (*i* - 1)”*t* nhưng trước thời điểm *I* ”*t*. Ở đây, khoảng thời gian t được người sử dụng đặt. Do đó, Khi có một sự kiện mới cần được chèn vào, danh sách con yêu cầu có thể được xác định mà không cần tìm kiếm. Sau đó một danh sách con thích hợp được tìm ngược lại để tìm ra vị trí thích hợp của đầu vào mới.



Hình 2.8 Danh sách chỉ mục tuyến tính

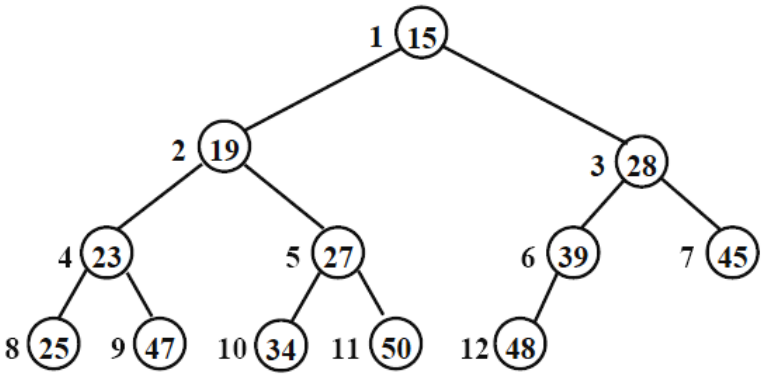
Một số thay đổi đã được đề xuất cho phương pháp này dựa trên lý luận rằng việc khoảng thời gian giữ sự kiện (thời gian từ khi lặp lịch một sự kiện và thời điểm xuất hiện của nó) được phân bố không đồng đều. Trong một thay đổi, đã có một đề xuất là để cho tất cả các danh sách có cùng một chiều dài bằng khoảng cách giữa mỗi đầu vào của danh sách; điều đó nghĩa là, *t* là biến đổi. Phép tìm kiếm nhị phân được sử dụng để tìm đầu vào danh sách thích hợp. Trong một đề xuất thay đổi khác, chỉ có danh sách con đầu tiên mới được sắp xếp; những danh sách còn lại không được sắp xếp. Một danh sách con chỉ được sắp xếp khi nó thành danh sách thứ nhất, do đó giảm được chi phí cho việc sắp xếp

Một đề xuất thay đổi đáng quan tâm của phương pháp này được gọi là **các hàng lịch**. Nó dựa

trên loại lịch để bàn mà con người hay dùng để lập lịch sự kiện. Một cuốn lịch để bàn thông thường có 365 trang – mỗi trang là một ngày trong năm. Tất cả các sự kiện trong một ngày được ghi vào trang ứng với ngày đó. Các sự kiện của cùng một ngày nhưng của năm tiếp theo cũng có thể được ghi vào trang đó Đặc điểm này sẽ không gây nhầm lẫn gì nếu năm xảy ra sự kiện cũng được ghi cùng với sự kiện và các sự kiện được xóa đi sau khi được thực hiện. Ý tưởng này có thể được thực hiện một cách dễ dàng bằng cách sử dụng danh sách tuyến tính được chỉ mục. Khoảng *t* ứng với một ngày, và kích thước của chỉ mục ứng với số ngày trong năm,Cả hai tham số này cần phải được lựa chọn cẩn thận để cho số lượng sự kiện trên một trang là nhỏ (gần với 0 hoặc 1).

3. Cấu trúc cây: Các cấu trúc dữ liệu hình cây cũng được sử dụng trong phương pháp mô phỏng các tập sự kiện. Thường là các cây nhị phân do đó thời gian tìm kiếm *n* sự kiện là $\log_2 n$.

Trường hợp đặc biệt của cây nhị phân là **heap,** trong đó mỗi sự kiện được lưu lại như là một node của cây nhị phân. Mỗi node có thể có tới 2 nhánh con, và thời gian cho mỗi sự kiện ở mỗi node nhỏ hơn thời gian ở mỗi nhánh con của nó. Điều đó nghĩa là sự kiện ở gốc có thời gian sớm nhất. Ưu điểm của các heap là cây có thể được lưu trong một ma trận (ngược với danh sách liên kết) bằng việc đặt gốc tại vị trí 1 của ma trận và các nhánh con tại vị trí 2 và 3. Các node sự kiện tiếp theo nằm ở vị trí 4, 5, 6, 7 của mà trận, và cứ thế, như ở ***hình 2.xyz3***. Điểm giao nhau của heap rất đơn gian vì rất dễ để tìm ra node cha và noed con của một node bất kỳ. Hai nhánh con của node nằm ở vị trí i là 2i và 2i + 1. Node cha của node nằm ở vị trí i là ở vị trí [*i*/2]. Ở đây, [.] nghĩa là bỏ bớt tới số nguyên nhỏ hơn tiếp theo. Ma trận có thể được sắp xếp lại một phần khi có thêm hoặc loại bỏ một phần tử



Hình 2.9 Cây nhị phân.

Việc lựa chọn cấu trúc dữ liệu thích hợp phụ thuộc vào phân bố thời gian giữ sự kiện và số lượng sự kiện trong tập sự kiện tương lai. Nó cũng phụ thuộc vào mức độ rằng buộc mà cấu trúc dữ liệu có thể được thực hiện trong một ngôn ngữ lập trình cho trước. Danh sách được liên kết đơn giản được coi là một thay thế có hiệu quả trong trường hợp số lượng sự kiện ít (ít hơn 20 sự kiện). Đối với các tập có kích cỡ 20 đến 120, danh sách tuyến tính chỉ mục là lựa chọn phù nhất, đối với những tập lớn hơn heap được coi là có hiệu quả nhất.

Chương 3: Sự hoạt động của chương trình mô phỏng sự kiện rời rạc

Tác giả: Thomas J. Schiriber, Daniek T. Brunner

Người dịch: Lê Thị Nga, Trịnh Việt Thi, Trần Diệu Linh, Nguyễn Minh Nguyệt, Hà Tất Thành, Đặng Thanh Chương, Vũ Thúy Vân, Nguyễn Thành Đạt

Biên tập: Nguyễn Nam Hoàng

3.1- Giới thiệu

Cách tiếp cận kiểu "black box" thường được sử dụng trong giảng dạy và học phần mềm mô phỏng các sự kiện rời rạc. Các đặc tính bên ngoài của phần mềm này được nghiên cứu, nhưng cơ sở mà phần mềm dựa trên hầu như bị bỏ qua hoặc nếu có đề cập đến thì cũng chỉ ở mức độ sơ lược. Sự tương ứng giữa cơ sở lí thuyết và quá trình thực hiện của phần mềm có thể không được nghiên cứu hết và liên quan đến việc từng bước thực hiện mô hình. Do đó người thiết kế mô hình có lẽ không thể tìm ra những cách để phát triển hướng tiếp cận đối với các tình huống tạo mô hình phức tạp, không thể sử dụng hiệu quả các công cụ tương hỗ để hiểu rõ những nguyên nhân gây lỗi phát sinh trong quá trình phát triển mô hình, và cũng ko thể sử dụng các công cụ tương hỗ để kiểm tra lại xem logic hệ thống phức tạp có được tuân theo một cách chính xác trong một mô hình hay không.

Mục đích của chương là mang đến cách hiểu tốt nhất những đặc điểm của mô phỏng sự kiện rời rạc và thúc đẩy người sử dụng nghiên cứu cách thực hiện những đặc tính trong phần mềm mô phỏng họ sử dụng. Kết quả là sẽ tăng cường tính hiệu quả mà nhờ nó người sử dụng có thể xây dựng, kiểm tra, và sử dụng các mô hình mô phỏng sự kiện rời rạc.

Cách tiếp cận sử dụng trong chương này nhằm mục đích phát triển một cách đánh giá tổng quát về cơ sở logic của mô phỏng sự kiện rời rạc, giới thiệu từ vựng chung và tăng cường sự ủng hộ đối với cách đánh giá này. Ba ví dụ về phần mềm mô phỏng thương mại (SIMAN với ngôn ngữ mô phỏng được viết trong ARENA; ProModel; và GPSS/H) được thảo luận theo cách đánh giá tổng quan này. Sự khác biệt giữa ba phần mềm này trong trong một vài tình huống được mô tả để nêu bật sự cần thiết phải hiểu các đặc tính của phần mềm mô phỏng.

Mục 3.2 thảo luận về cách đánh giá luồng lưu lượng và bản chất của mô phỏng sự kiện rời rạc, bao gồm đơn vị lưu lượng (unit of traffic), sự kiện (event), và thời điểm sự kiện đồng nhất (identical event times). Thảo luận này tiếp tục ở mục 3.3 với thực thể (entities), tài nguyên (resource), phần tử điều khiển (control elements) , sự hoạt động, và sự tóm tắt về mô hình thực hiện (mục 3.4). Mục 3.5 và mục 3.6 nêu các chủ đề về trạng thái thực thể và cấu trúc quản lý thực thể. Các cơ chế đáp ứng yêu cầu logic của mô phỏng sự kiện rời rạc sử dụng trong SIMAN, ProModel, và GPSS/H được mô tả trong mục 3.7. Mục 3.8 bao gồm các ví dụ về về sự khác biệt trong cách thực hiện của các phần mềm ở mục 3.7 dẫn đến kết quả khác biệt trong một vài tình huống mô phỏng.

Thuật ngữ sử dụng chung trong chương này không được nhấn mạnh nhưng các thuật ngữ được sử dụng bởi SIMAN, ProModel, và GPSS/H được viết bằng chữ in hoa. Bảng thuật ngữ chung liên quan đến các thuật ngữ tương đương của các phần mềm cụ thể được kèm theo để giúp đỡ phân biệt giữa các thuật ngữ có tính chất chung và riêng.

3.2. Các khía cạnh của mô phỏng sự kiện rời rạc

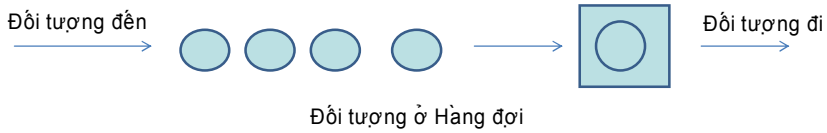
3.2.1. Cách đánh giá theo luồng giao dịch

Khái niệm luồng lưu lượng cung cấp cơ sở cho mô phỏng sự kiện rời rạc. Theo khái niệm này, một hệ thống bao gồm các đơn vị lưu lượng rời rạc cạnh tranh lẫn nhau để sử dụng nguồn tài nguyên hữu hạn trong khi di chuyển ("flowing') từ điểm này đến điểm khác trong hệ thống. Đơn vị lưu lượng có khi được gọi là lưu lượng, dẫn đến cụm từ luồng lưu lượng.

Một ví dụ đơn giản về sự cạnh tranh của đơn vị lưu lượng để sử dụng nguồn tài nguyên hữu hạn đó là hệ thống một server, một hàng như hình 3.1. Trong đó đơn vị lưu lượng được thể hiện bằng các hình tròn, hình vuông là *server* ("nguồn tài nguyên"), và hình tròn nằm trong hình vuông là đơn vị lưu lượng đang được đáp ứng. Hàng lưu lượng đợi để được đáp ứng dịch vụ gọi là một

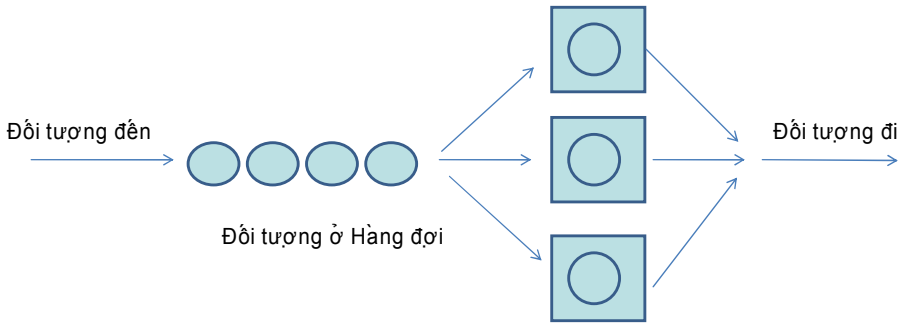
hàng đợi (*queue*). Hệ thống gồm server, đơn vị lưu lượng đang được đáp ứng và đang đợi để được đáp ứng được gọi là hệ thống hàng đợi *(queuing system)*. Trong thực tế, các đơn vị lưu lượng có thể hoạt động trong quy trình vận hành và server chính là máy móc. Hoặc lưu lượng có thể là các công việc in ấn và server chính là máy in. Hoặc lưu lượng là bệnh nhân và server có thể là một bác sỹ, v...v.

Một ví dụ khác về đơn vị lưu lượng sử dụng nguồn tài nguyên hữu hạn là hệ thống một hang chờ và nhiều server của hình 3.2. Lưu lượng đợi được đáp ứng trên một hàng. Đơn vị đầu hàng sẽ đi đến server rỗi kế tiếp. Hệ thống như thế gồm các cuộc gọi đến và điện thoại viên tại một công ty nhận đặt hàng qua điện thoại, hoặc của khách hàng và nhân viên ngân hàng, hoặc của khách du lịch và nhân viên làm thủ tục nhập cảnh tại sân bay.



Hình 3.1 Hệ thống hàng đợi một hàng đợi, một server

Ước chừng 80% đến 90% phần mềm thương mại mô phỏng sự kiện rời rạc hiện tại dựa trên quan điểm, cách đánh giá theo luồng giao dịch.



Hình 3.2 Hệ thống hàng đợi một hàng đợi, nhiều server

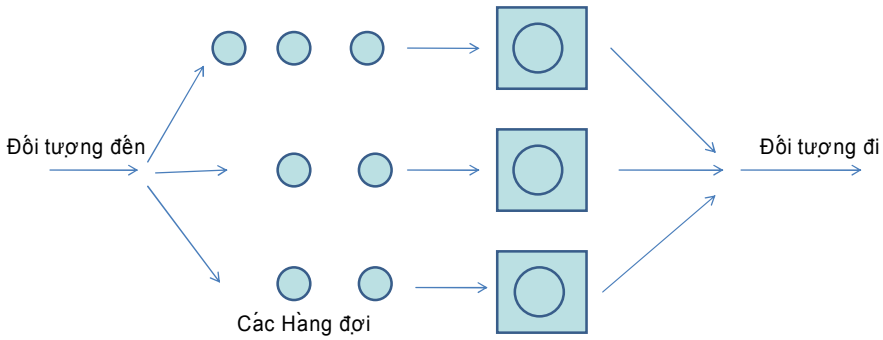
Một dạng khác của hệ thống luồng giao dịch đó là hệ thống nhiều hàng đợi, nhiều server như hình 3.3. Ở đây có nhiều hàng đợi, mỗi hàng một server. Đơn vị lưu lượng đợi được đáp ứng tại đầu mỗi hàng. Các hệ thống có dạng thiết kế này bao gồm có các điểm thu phí trên đường, hệ thống thanh toán tiền trong siêu thị, và các sân bay được xây dựng hai hay nhiều đường băng (nơi mà các đơn vị lưu lượng là máy bay và đường băng là các server)

Ở hình 3.1 đến 3.3, các kiểu hệ thống là các khối kiến trúc tạo nên nhiều hệ thống phức tạp hơn. Ví dụ, xem xét hệ thống cảng đơn giản trong hình 3.4. Các thuyền (đơn vị lưu lượng) đến cảng để trả hoặc bốc hàng hóa. Có hai loại thuyền: loại A và loại B. Có 3 loại server: tàu kéo, chỗ đậu thuyền loại A và chỗ đậu thuyền loại B.

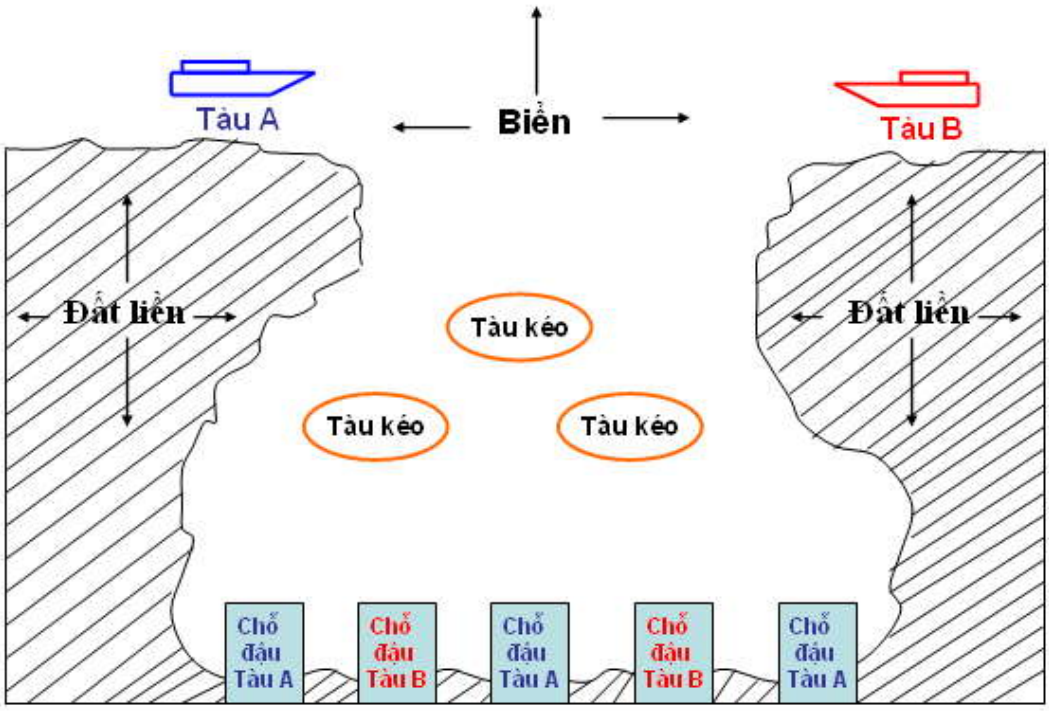
Thuyền loại A và B chỉ có thể sử dụng chỗ đậu tương ứng của từng loại. Các thuyền sử dụng tàu kéo để di chuyển vào trong cảng và vào chỗ đậu, và sau đó lại sử dụng tàu kéo để ra khỏi chỗ đậu và cảng. Số lượng tàu kéo mà thuyền cần dùng phụ thuộc vào loại tàu và nơi thuyền di chuyển đến.

Đặc điểm của hệ thống này là nhiều loại lưu lượng, nhiều loại tài nguyên, lưu lượng phụ thuộc vào kiểu lưu lượng, các dịch vụ tĩnh (chỗ đậu), dịch vụ di động (tàu kéo), và sự cần thiết điều khiển đồng thời nhiều loại tài nguyên bởi các đơn vị lưu lượng (ví như 1 thuyền cần một chỗ đậu và một hay nhiều tàu kéo trước khi có thể di chuyển liên tục vào trong cảng và đến chỗ đậu). Thử hình

dung các hệ thống trong hình 3.1 đến 3.3 có thể được kết hợp như thế nào trong trong hệ thống hình 3.4



Hình 3.3 Hệ thống nhiều hàng đợi, nhiều server



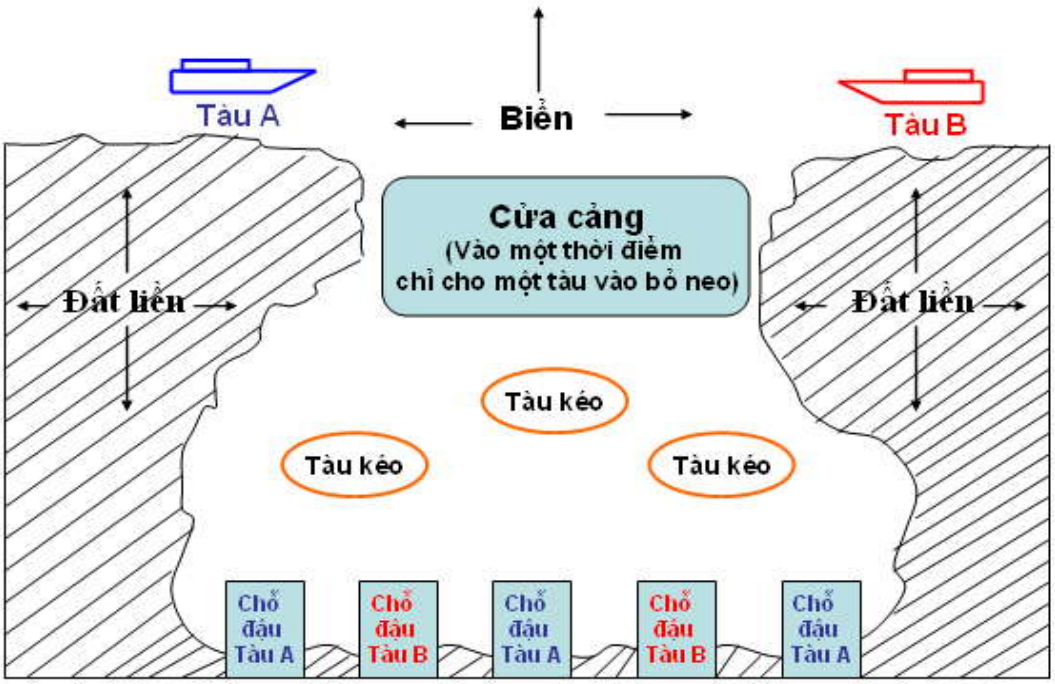
Hình 3.4 Minh họa một hệ thống cảng biển

Không gian thường là nguồn tài nguyên hữu hạn. Diễn giải điều này trong hệ thống nhiều server, nhiều hàng của hình 3.3, ví dụ, các máy bay là lưu lượng và server là đường băng (không gian). Trong hệ thống cảng hình 3.4, không nói gì về không gian nếu cửa cảng đủ rộng, ví dụ đủ để nhiều thuyền di chuyển qua tại cùng thời điểm. Giả thiết rằng, cửa cảng chỉ rộng đủ cho một thuyền qua lại tại một thời điểm. Như vậy không gian là tài nguyên hữu hạn, được thể hiện trong hình 3.5.
Nhiều hệ thống liên hệ tới sự diễn giải luồng giao dịch. Bao gồm các hệ thống sản xuất, chăm sóc sức khỏe, vận chuyển, dân sự, truyền thông, bảo vệ và hệ thống xử lý thông tin, và hệ thống hàng đợi nói chung.

3.2.2. Bản chất của mô phỏng sự kiện rời rạc

Một hệ thống sự kiện rời rạc là một hệ thống mà ở đó trạng thái của hệ thống thay đổi rời rạc nhưng có thể theo cách ngẫu nhiên, tập hợp các thời điểm được coi như thời điểm sự kiện. Mỗi sự kiện là một sự thay đổi trạng thái hệ thống. Ví dụ, giả thiết rằng, thuyền loại A rời khỏi cửa cảng trong hệ thống hình 3.5. Sự rời đi này là một sự kiện. Điều này xảy ra tại một thời điểm và thay đổi trạng thái của hệ thống. (Số lượng thuyền loại A ra khỏi cửa cảng tăng thêm một). Tương tự, giả sử một

thuyền loại B đang sử dụng một tàu kéo. Sử dụng tàu kéo là một sự kiện, xảy ra tại một điểm thời gian và thay đổi trạng thái của tàu kéo từ trạng thái "rỗi" sang trạng thái "đang được sử dụng".
Đồng hồ mô phỏng ghi lại các thời điểm mà tại đó các sự kiện xuất hiện trong mô phỏng sự kiện rời rạc. Đồng hồ như vậy được tạo ra bởi phần mềm mô phỏng sự kiện rời rạc và thông số của nó được quản lý tự động bởi phần mềm. Thông số của đồng hồ thay đổi trong suốt quá trình mô phỏng, chỉ ghi lại các thời điểm rời rạc mà tại đó các sự kiện xuất hiện.



Hình 3.5 Hệ thống cảng biển với tài nguyên hữu hạn

Một vài khía cạnh của của trạng thái hệ thống thay đổi liên tục theo thời gian, thay vì thay đổi tại các thời điểm rời rạc. Ví dụ, giả sử rằng quá trình xử lý bốc hàng hóa lên thuyền bắt đầu lúc 1:30 chiều, tiếp diễn trong 4 giờ, kết thúc lúc 5:30 chiều. Do đó mức độ hoàn thành (trạng thái của hệ thống) thay đổi liên tục trong suốt 4 giờ. Tuy vậy, quá trình bốc hàng này có thể mô hình hóa bởi các điều kiện sự kiện rời rạc bằng cách tập trung vào hai sự kiện rời rạc tương ứng với sự bắt đầu quá trình và sau đó là sự hoàn thành quá trình và them vào đó thời gian trễ mô phỏng (thay thế cho thời gian bốc hàng) giữa hai sự kiện. Sử dụng cách tiếp cận này, ta có thể mô hình hóa sự thay đổi trạng thái liên tục bởi các điều kiện sự kiện rời rạc.
Thảo luận ở đây chỉ giới hạn trong các hệ thống mà mọi thay đổi trạng thái hệ thống có thể được mô hình hóa bởi các kiện rời rạc.

3.2.3. Đơn vị lưu lượng, sự kiện và thời điểm sự kiện đồng nhất.

Đơn vị lưu lượng xảy ra khi hệ thống cho phép hoặc yêu cầu. Hoạt động như vậy dẫn tới một hay nhiều sự thay đổi trong trạng thái hệ thống (các sự kiện). Trong một hệ thống cảng, một *sự kiện đến* xảy ra khi một tàu đến ngoài cửa cảng. *Sự kiện sử dụng* xuất hiện khi có một thuyền sử dụng tàu kéo. *Sự kiện bắt đầu dịch vụ* xuất hiện khi thuyền bắt đầu được kéo vào trong chỗ đậu.

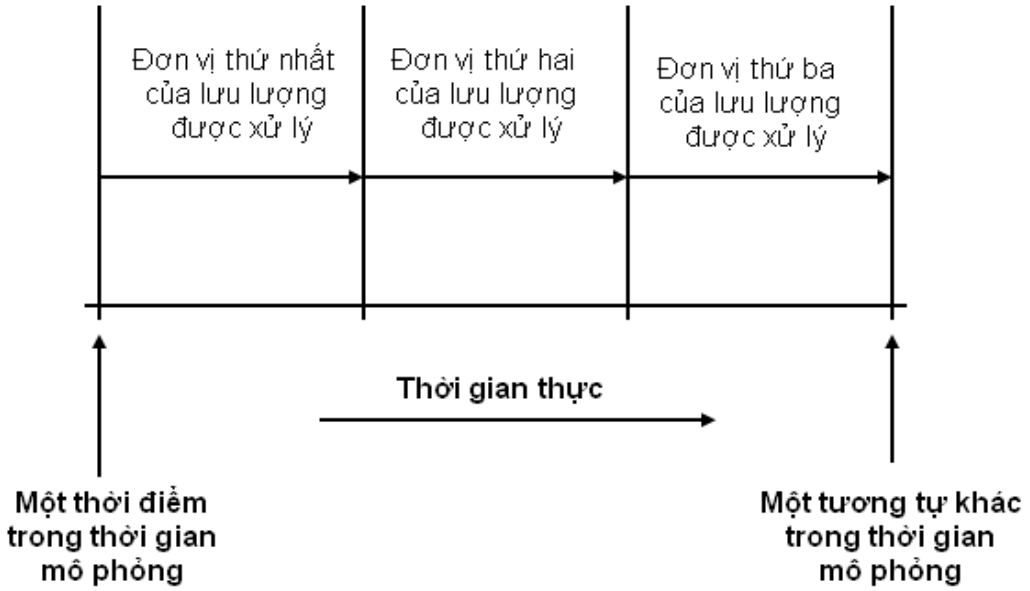
Hai hay nhiều sự kiện thường xảy ra tại cùng một thời điểm; đó là các thời điểm sự kiện đồng nhất. Ví dụ, hai sự kiện xảy ra tại cùng một thời điểm nếu như một thuyền sử dụng một tàu kéo và ngay lập tức được bắt đầu kéo vào một bến. Ở đây, hành động do một đơn vị lưu lượng gây ra dẫn đến một dãy hai sự kiện tại cùng một thời điểm. Đồng hồ mô phỏng vẫn giữ nguyên giá trị, trong khi đó các sự kiện với thời điểm đồng nhất lần lượt xảy ra. Thời gian thực (đồng hồ tường) trôi qua trong khi máy tính cập nhật từng trạng thái của mô hình tại thời điểm mô phỏng yêu cầu.

Các dãy sự kiện khác xảy ra nối tiếp tại các thời điểm sau đó. Ví dụ, giả sử rằng thời gian đến cảng giữa các thuyền liên tiếp nhau biến đổi ngẫu nhiên và luôn lớn hơn 0. Khi một thuyền mới

đến, thuyền kế tiếp nó sẽ không đến bến được cho đến thời điểm mô phỏng sau. Một ví dụ khác, nếu như một thuyền bắt đầu bốc hàng tại một thời điểm sẽ không hoàn thành việc bốc hàng cho đến một thời điểm đã định sau đó.

Hai đơn vị lưu lượng có thể tham gia vào nhiều sự kiện tại cùng thời điểm. Trong hệ thống một hang chờ, một server hình 3.1, giả sử rằng một đơn vị lưu lượng gây ra sự kiện hoàn thành dịch vụ khi hàng đợi không rỗng. Điều này tạo nên trạng thái cho một đơn vị lưu lượng khác (được đáp ứng tiếp theo) để tạo ra sự kiện sử dụng dịch vụ và sự kiện khởi đầu dịch vụ. Ở đây, một sự kiện xảy ra do một đơn vị lưu lượng tạo điều kiện cho sự xuất hiện của 2 sự kiện liên quan khácđối với đơn vị lưu lượng khác tại cùng thời điểm.

Bây giờ ta xét tình huống với ba đơn vị lưu lượng tạo nên nhiều sự kiện với thời điểm sự kiện đồng nhất. Giả sử rằng một thuyền trong hệ thống cảng hình 3.4 đang sử dụng 2 tàu kéo để ra khỏi cảng. Khi thuyền kết thúc quá trình này gây ra sự kiện hoàn thành dịch vụ, thay đổi trạng thái của hai tàu kéo từ "đang sử dụng" thành "rỗi". Nếu có hai thuyền khác đang yêu cầu một tàu kéo, thì ba đơn vị lưu lượng này có thể gây ra nhiều sự kiện thời gian đồng nhất (kết thúc dắt; sau đó một tàu kéo được sử dụng, có thể theo sau bởi các sự kiện khởi tạo dịch vụ; sau đó tàu kéo khác được sử dụng và cũng có thể cả hai tàu kéo được sử dụng đồng thời)



Hình 3.6 Thời gian thực (theo đồng hồ thực- real time) vs thời gian mô phỏng (simulated time)

Đồng hồ mô phỏng vẫn không thay đổi các giá trị trong khi các sự kiện có thời điểm sự kiện đồng nhất được thực hiện lần lượt. Sự kiện đầu tiên xuất hiện, sau đó đến sự kiện thứ hai và cứ thế tiếp tục. Thực tế là thời gian thực trôi qua tại cùng thời gian mô phỏng cố định trong khi máy tính làm việc lần lượt với nhiều đơn vị lưu lượng, và thực hiện lần lượt nhiều sự kiện, được minh họa trong hình 3.6. Hình này tương ứng với mô tả trong đoạn trước, khi ba đơn vị lưu lượng gây ra nhiều sự kiện có các thời điểm sự kiện đồng nhất.

Trật tự thời gian thực trong đó hai hay nhiều sự kiện xuất hiện tại một thời điểm cố định đôi khi phụ thuộc vào logic điều khiển. Ví dụ như đối tượng chờ tiếp theo không thể sử dụng server cho đến khi đối tượng sử dụng trước đặt server vào trạng thái rỗi, vì vậy sự kiện "server rỗi" xảy ra trước sự kiện "sử dụng sever" sau đó. Tương tự như vậy, một thuyền không thể bắt đầu quá trình cập bến cho đến khi nó giành được tàu kéo. Ở đây, logic điều khiển các dãy sự kiện.

. Trật tự thời gian thực của các sự kiện có thời điểm sự kiện đồng nhất không phải lúc nào cũng được điều khiển theo logic. Ví dụ, khi một thuyền đặt hai tàu kéo trong trạng thái "rỗi" và hai thuyền khác đang đợi để giành một tàu kéo, logic không điều khiển trật tự thời gian thực trong đó hai sự kiện giành tầu kéo cùng xuất hiện. Liệu trật tự thời gian có phải là một vấn đề có ý nghĩa quan trọng hay không? Điều này là có thể. Giả sử rằng hai tàu kéo khác loại (ví dụ, một loại mạnh

hơn và nhanh hơn loại kia) và cả hai thuyền đều muốn giành tàu kéo này. Thuyền hoạt động trước sẽ giành được tàu kéo tốt hơn, để tàu kéo còn lại cho chiếc thuyền kia.

Một ví dụ khác bao gồm một sự thay đổi toàn cục trong trạng thái hệ thống trong đó hai hay nhiều sự kiện có thể xuất hiện tại một thời điểm cố định, nhưng trong một trật tự thời gian thực tùy ý. Trong hệ thống cảng ở hình 3.5, giả sử rằng một cơn bão đang xảy ra, nên không thuyền nào có thể ra khỏi cảng. Cuối cùng cơn bão ngớt, các thuyền có thể rời khỏi cảng an toàn. Nếu hai thuyền đang đợi để rời đi nhưng tại một thời điểm chỉ có một thuyền có thể di chuyển qua cửa cảng, trật tự thời gian thực của hai thuyền sẽ quyết định thuyền nào sẽ ra khỏi cảng trước và thuyền nào phải đợi để rời đi sau

Phần thảo luận trước đó bao gồm các tình huống trong đó các dãy sự kiện phụ thuộc xuất hiện tại cùng thời điểm. Cũng có khả năng các sự kiện độc lập xuất hiện tại cùng thời điểm. Trong cảng ở hình 3.4, ví dụ một thuyền loại A đến cảng cùng thời điểm thuyền loại B hoàn thành quá trình bốc hàng. Nếu như thời gian giữa hai chuyến hàng của thuyền loại A và thời gian xử lý bốc hàng thay đổi ngẫu nhiên, xác suất mà sự kiện đến và hoàn tất dịch vụ có cùng thời điểm sẽ nhỏ. Nếu thời điểm sự kiện đồng nhất, trật tự thời gian thực tại đó đơn vị lưu lượng cố gắng thực thi có thể có ý nghĩa.

Trong ví dụ này, giả sử rằng thuyền loại A cần một tàu kéo để vào trong cảng và thuyền loại B cần một tàu kéo để ra khỏi bến đỗ và ra khỏi cảng. Cũng giả thiết chỉ có một tàu kéo trong trạng thái "rỗi". Nếu như thuyền loại A hoạt động trước và giành tàu kéo, thì thuyền loại B phải đợi. Tương tự, nếu như thuyền loại B hoạt động trước và giành tàu kéo thì thuyền loại A phải đợi. Điều đó có nghĩa gì? Trong mô hình này có thể để tùy ý hoạt động xảy ra. Hoặc người thiết kế mô hình có thể xác định rõ cách hoạt động hệ thống thực trong trường hợp này và sau đó xây dựng mô hình mô phỏng theo hệ thống thực.

Thực tế là nhiều sự kiện xuất hiện tại cùng một thời điểm có thể dẫn đến sự phức tạp về logic trong mô phỏng sự kiện rời rạc. Sự phức tạp này có thể được cả người thiết kế mô hình và ở mức độ cao hơn là người thiết kế ngôn ngữ tìm hiểu và xem xét. Người thiết kế mô hình phải xem xét sự phức tạp trong các tình huống tạo mô hình cụ thể, trong khi đó người thiết kế ngôn ngữ phải xem xét bằng một cách tổng quát. Đặc biệt, lựa chọn và cân bằng các yếu tố tồn tại là do người thiết kế ngôn ngữ. Kết quả là, mặc dù ngôn ngữ mô phỏng sự kiện rời rạc nói chung là như nhau, chúng vẫn có một số điểm khác biệt.

3.3 Thực thể, tài nguyên, phần tử điều khiển và các hoạt động

Các hệ thống bao gồm các thực thể, các tài nguyên, các phần tử điều khiển và các hoạt động. Những thành phần này được trình bày sau đây

3.3.1 Các thực thể

Thuật ngữ *entity* được sử dụng ở đây như là tên chung của một đơn vị lưu lượng (một "giao dịch"). Các thực thể mô hình hóa các đối tượng chẳng hạn như những con tàu trong hệ thống bến cảng, việc đang thực hiện hệ thống sản xuất, những người mua sắm ở chợ, những máy bay tại sân bay, những cuộc gọi trong hệ thống thông tin v…v. Như chúng ta đã thấy, các thực thể thực hiện các hành động mà làm thay đổi trạng thái của hệ thống.

Ngôn ngữ mô hình hóa cung cấp những công cụ để tạo ra và hủy bỏ các thực thể trong suốt quá trình mô phỏng. Các thực thể tiến vào một mô hình theo thời gian chẳng hạn như những con tàu cập cảng theo thời gian. Tương tự như vậy, các thực thể rời khỏi mô hình lần lượt theo thời gian chẳng hạn như những con tàu đã được phục vụ nay rời khỏi cảng. Trong suốt quá trình mô phỏng số lượng các thực thể là ngẫu nhiên.

Thực thể có các thuộc tính. Thuộc tính của con tàu ở bến cảng bao gồm thời gian chuyến hàng đến, loại tàu, số lượng tàu kéo cần đến khi cập bến, thời gian sắp xếp hàng lên tàu, số tàu kéo cần đến khi tàu rời bến v…v

Có hai loại thực thể đặc trưng là thực thể ngoại và thực thể nội. Các thực thể ngoại là các thực thể mà sự tạo ra và hoạt động của chúng được hình dung rõ ràng và sắp xếp bởi người thực

hiện mô hình hóa. Các thực thể được đề cập trên đây (ví dụ như: những con tàu, quá trình làm việc, cuộc gọi) là ví dụ về các thực thể ngoại.

Trong một mô hình thường có trên hai lớp thực thể ngoại trong đó mỗi lớp có đặc trưng và những thuộc tính riêng. Ví dụ như trong dây chuyền sản xuất, có thể có một lớp thực thể cho một số loại công việc theo đơn hàng nào đó, có một lớp thực thể cho các công nhân nào đó v…v (Một đơn hàng có thể có thuộc tính là kỳ hạn thực hiện nhưng công nhân thì không có. Một công nhân chắc chắn có thuộc tính là các kỹ năng nhưng đơn hàng thì không). Về mặt mô hình hoá, sự khác biệt giữa các lớp thực thể là do chính người thiết kế mô hình, là người xây dựng mô hình thật hợp lý dựa trên những sự khác biệt. Ngược lại, định nghĩa chính thức về các lớp thực thể ngoại có thể được yêu cầu trong một mô hình. Định nghĩa như vậy phụ thuộc vào phần mềm đang được sử dụng.

Đối lập với các thực thể ngoại là những đối tượng dựa trên các khái niệm riêng biệt rõ ràng, các thực thể nội là các thực thể ẩn được sử dụng bởi một số ngôn ngữ mô hình hoá để hỗ trợ các nhu cầu khác trong mô hình hóa sự kiện rời rạc. Thực thể nội được tạo ra và thực hiện hoàn toàn bởi chính phần mềm mô phỏng và về khía cạnh này, người tạo mô hình không thể nhìn thấy được chúng. Người thiết kế mô hình thậm chí có thể không nhận ra được rằng các thực thể nội đang hoạt động trong một mô hình.

Ví dụ thực thể nội được sử dụng trong một số ngôn ngữ mô hình hóa để tạo ra việc máy móc bị hỏng và sau đó đưa máy móc này quay về trạng thái làm việc theo yêu cầu (điều hiển nhiên là thông số kỹ thuật của việc dừng hoạt động và sửa chữa phải được cung cấp bởi người thiết kế mô hình, nhưng người thiết kế mô hình không phải tạo ra các logic cần thiết cho việc máy móc bị hỏng này nếu như các thực thể nội được sử dụng với mục đích này). Ngược lại, một số ngôn ngữ mô hình hoá không sử dụng thực thể nội để mô hình hóa việc máy móc bị hỏng. Trong những ngôn ngữ như vậy, người thiết kế mô hình làm việc với thực thể ngoại và cung cấp rõ ràng những logic cần thiết một cách hợp lý để thực hiện các sự hỏng hóc này.

Ví dụ khác, một thực thể nội được sử dụng trong vài ngôn ngữ để dừng một mô phỏng. Ví dụ như người thiết kế mô hình có thể đặt trạng thái rằng một chương trình mô phỏng sẽ dừng sau tám tiếng thời gian mô phỏng và phần mềm mô hình hoá tạo ra một thực thể thực hiện điều này (Nếu các thực thể nội không được tạo ra cho mục đích này thì người thiết kế mô hình sẽ dùng thực thể ngoại để đạt được hiệu quả như mong muốn).

3.3.2 Tài nguyên

Tài nguyên là một phần tử của hệ thống để cung cấp dịch vụ. Các tài nguyên trên một bến cảng bao gồm các tàu kéo và những bến tàu. Tài nguyên trong dây chuyền sản xuất bao gồm máy móc, công nhân vận hành, thiết bị vận chuyển (chẳng hạn như là phương tiện truyền dẫn tự động và băng truyền), không gian để chứa hàng tạm thời chờ đóng gói và hàng hoá đã được đóng gói. Các tài nguyên ở sân bay bao gồm khu vực đỗ xe ôtô, xe buýt trung chuyển, người khuân vác, quầy vé, thiết bị an ninh, lối dành cho đi bộ, quầy làm thủ tục đi máy bay, đường ống đi ra máy bay, máy bay và đường băng.

Một vài tài nguyên chỉ có thể phục vụ một người dùng tại một thời điểm. Ví dụ một khu vực đỗ xe chỉ có thể giữ mỗi lần một xe và một cầu thang máy bay chỉ có thể tiếp cận mỗi lần một máy bay. Tuy nhiên trong vài trường hợp, một tài nguyên có thể phục vụ hai hoặc nhiều người sử dụng tại một thời điểm. Một phương tiện truyền dẫn tự động có thể truyền cùng lúc 3 sản phẩm từ điểm A đến điểm B và xe buýt trung chuyển có thể chở nhiều người từ bãi đỗ đến cổng vào sân bay.

Tài nguyên có giới hạn về số lượng. Ở bến cảng có thể có 3 tàu kéo, 2 bến tàu loại A và 3 bến tàu loại B. Một sân bay có thể có 3 đường băng, có thể có 250 chỗ trong bãi đỗ xe. Trong dây chuyền sản xuất có thể có 4 thiết bị truyền dẫn tự động (AGV).

Người sử dụng các nguồn tài nguyên thường là các thực thể. Một thực thể con tàu cập một bến tàu và rồi có thể một cần 1 tàu kéo đưa vào cập bến. Một thực thể đang được thực hiện chiếm chỗ ở bộ đệm đầu vào của chiếc máy tiếp theo, và rồi sử dụng thiết bị truyền dẫn tự động để đưa từ vị trí hiện tại đến bộ đệm đầu vào. Một hành khách đi máy bay có thể liên tục sử dụng một xe buýt trung chuyển, một người khuân vác hành lý, thiết bị an ninh, một lối dành cho đi bộ, một quầy làm thủ tục đi máy bay, một chỗ ngồi và một đướng ống đi ra máy bay.

Trong thực tế, tài nguyên bị giới hạn về sử dụng vì thế các thực thể đôi khi phải chờ đợi đến lượnt sử dụng tài nguyên. Khi một sản phẩm đang trong quá trình sản xuất yêu cầu một thiết bị truyền dẫn tự động, nó có thể phải chờ tới khi yêu cầu đó được đáp ứng. Khi sản phẩm này thậm chí đã đudợc đưa vào bộ đệm đầu vào của máy móc tiếp theo, nó có thể phải chờ để sử dụng máy móc này.

Các ngôn ngữ mô hình hóa có các cơ chế (constructs) được dùng để điều khiển các thực thể truy cập trực tiếp tới các tài nguyên. Những cơ chế này sẵn sàng ghi lại trạng thái của tài nguyên ("nhàn rỗi" hoặc "đang được sử dụng") và điều kiện hoạt động ("đang hoạt động" hoặc "hỏng hóc"). Khi một thực thể cố gắng giành lấy một tài nguyên, trạng thái giành lấy và điều kiện hoạt động có thể được kiểm tra bằng phần mềm để xác định xem liệu việc giành lấy tài nguyên này có thể thực hiện hay không.

3.3.3 Phần tử điều khiển

Ngoài các cơ chế tài nguyên, các ngôn ngữ mô hình hoá cung cấp các cơ chế khác để hỗ trợ các khía cạnh liên quan tới sự điều khiển của trạng thái hệ thống. Thuật ngữ *"Control element"*(phần tử điều khiển) được sử dụng cho những cơ chế như vậy. Một chuyển mạch là ví dụ về phần tử điều khiển. Một chuyển mạch là biến có hai trạng thái (Bật hoặc Tắt). Một chuyển mạch có thể được sử dụng trong mô hình hệ thống cảng biển để báo hiệu xem liệu đang có bão hay không (Nếu trời đang bão, tàu bè buộc phải neo trong bến cảng chờ cho bão tan). Trong nghiệp vụ ngân hàng, một chuyển mạch có thể được sử dụng để xem liệu các cửa ra vào ngân hàng khoá lại hay chưa.

Bộ đếm là một dạng phần tử điều khiển khác. Một bộ đếm có thể sử dụng để đếm số lượng các khối động cơ đã có các lỗ khoan kể từ khi máy khoan thay mũi khoan. Mũi khoan có thể được thay thế sau 100 lần sử dụng. Thực hiện việc thay thế này yêu cầu lưu lại số lần khoan. Bộ đếm được sử dụng để giúp điều khiển hoạt động của hệ thống trong việc này.

Các biểu thức số học có thể là phần cơ bản của các phần tử điều khiển. Hãy xem xét một siêu thị là một hệ thống gồm nhiều hàng, nhiều server tương ứng với các đường ra để thanh toán. Có thể có 12 đường ra để thanh toán nhưng chỉ có vài ba đường được mở tại một thời điểm. Giả thiết rằng nếu số lượng khách hàng đang đợi ở lối ra để thanh toán là 5 hoặc nhiều hơn thì lối ra thanh toán khác sẽ được mở. Người thiết kế mô hình có thể giới thiệu một biểu thức số học để tính số lượng trung bình của các khách hang đang chờ ở lối ra thanh toán. Một thực thể dùng để mô phỏng "bộ phận quản lý đường ra" có thể theo dõi giá trị của biểu thức này để xác định xem liệu các điều kiện hiện tại có yêu cầu mở lối ra thanh toán khác. Biểu thức số học được sử dụng để điều khiển hoạt động của bộ phận quản lý đường ra.

Biểu thức Boolean (các biểu thức gồm giá trị *True* hoặc *False* với những toán tử *and*, *or* và *not*) cũng có thể được sử dụng như là phần tử điều khiển. Ví dụ như một con tàu không thể rời cảng cho đến khi nó có được một tàu kéo công suất lớn hoặc (*or)* hai tàu kéo công suất nhỏ hơn và (*and)* không có cơn bão nào đổ bộ.

Giống như tài nguyên, phần tử điều khiển có thể bắt buộc các thực thể chờ đợi, trì hoãn sự di chuyển xuyên suốt một hệ thống. Các phương thức trong đó phần mềm mô hình hóa quản lý các thực thể trễ được thảo luận ở mục 3.6.

3.3.4 Các hoạt động

Một *hoạt động* là một bước di chuyển hoặc một hành động được thực hiện bởi hoặc trên một thực thể trong suốt sự di chuyển của nó trên hệ thống. Các ví dụ của sự hoạt động bao gồm đơn hàng mới đến trong một hệ thống xử lý đơn hàng, việc giành lấy một AGV bởi một sản phẩm đang trong quá trình sản xuất, việc chụp cánh tay bệnh nhân bằng X-quang và sự vận chuyển một đơn vị sản phẩm đã hoàn thiện tới nơi lưu kho sản phẩm.

Một tập hợp có thứ tự của các hoạt động là một chuỗi của các bước di chuyển hoặc của các hành động liên tiếp được thực hiện hoặc đã trải qua bởi một thực thể nó di chuyển từ điểm tới điểm trong một hệ thống. Một chuỗi kết hợp của các hoạt động đôi khi được gọi là *logic hoạt động*. Ví dụ như, logic hoạt động cho việc tàu di chuyển vào bến cảng: đến bên ngoài bến cảng; giành lấy bến tàu; giành lấy 2 tàu kéo; sử dụng tàu kéo để tiến tới không ngừng vào bến cảng và cập bến; giải

phóng tàu kéo; sử dụng bến tàu để nhập hoặc xuất hàng; giành lấy 1 tàu kéo; dùng tàu kéo đẩy không ngừng ra khỏi bến tàu và ra khỏi bến cảng; khởi hành.

3.4 Tổng quan về việc thực hiện mô hình

3.4.1 Các dự án, thử nghiệm, và sự lặp lại

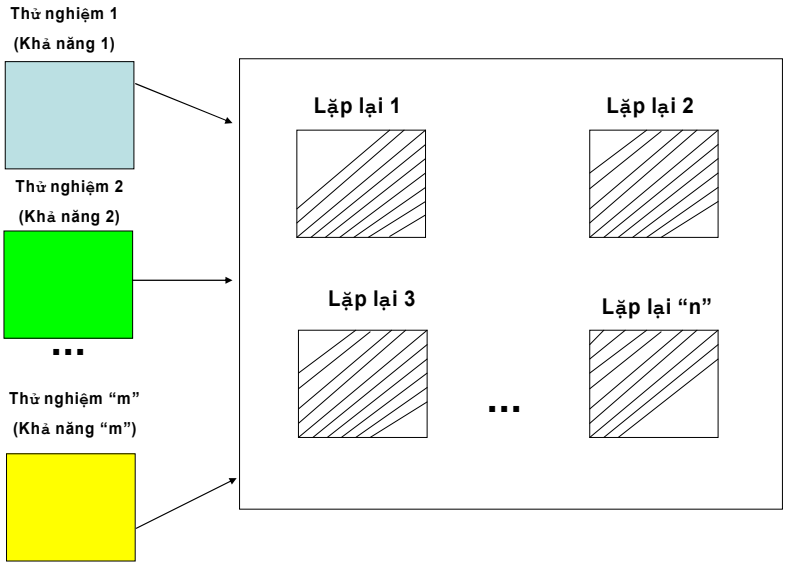
Thực hiện một dự án mô phỏng bao gồm việc thực hiện một hoặc thử nghiệm và với mỗi thử nghiệm, thực hiện lặp lại một hoặc nhiều lần. Minh hoạ như trong hình 3.7, có *m* thử nghiệm, mỗi thử nghiệm có *n* lần lặp lại. Những thử nghiệm được phân biệt bởi các cách thức trong một logic của mô hình và/hoặc dữ liệu. Sự lặp lại thường được phân biệt bởi việc sử dụng các tập hợp khác nhau của các số ngẫu nhiên từ sự lặp lại này tới sự lặp lại khác và qua các thử nghiệm khau.

Giả sử rằng một dự án mô phỏng được thực hiện cho một hệ thống cảng biển như trong hình 3.5. Mục tiêu của dự án giả thiết là để nghiên cứu các cách thức nhằm giảm sự chậm trễ của con tàu khi rời bến cảng. Giả thiết rằng hình 3.5 mô tả bến cảng (3 tàu kéo, 3 bến tàu loại A, 2 bến tàu loại B và cửa bến cảng chỉ đủ rộng để tiếp nhận mỗi lần 1 con tàu) và dữ liệu hoạt động sẵn sàng (phân bố của thời gian tàu đến, thời gian nhập cảng và xuất cảng, trọng tải nhập và xuất của loại tàu A và B). Giả sử rằng trật tự phục vụ để sử dụng tàu kéo và bến tàu là tàu nào đến trước được phục vụ trước. Thử nghiệm 1 như hình 3.7 dùng cho mô hình bến cảng này (mục đích của thử nghiệm 1 là phê chuẩn mô hình này bằng việc so sánh các đặc tính của nó với các đặc tính quan sát trên hệ thống thực. Ví dụ như, phân bố của thời gian ở lại cảng của tàu loại A và loại B có thể được so sánh với giá trị tương ứng trên hệ thống thực).

Thử nghiệm 2 trong hình 3.7 nghiên cứu sự tác động (dựa trên phân bố của thời gian cập cảng) của việc nới rộng cửa bến cảng để tiếp nhận 2 tàu một lần. Thử nghiệm 3 nghiên cứu sự tác động vủa việc thêm một tàu kéo. Thử nghiệm 4 nghiên cứu tác động của việc kết hợp sự mở rộng cổng bến cảng và thêm một tàu kéo. Thử nghiệm 5 nghiên cứu tác động của việc dành cho tàu loại A ưu tiên cao hơn loại B khi dùng tàu kéo. Thử nghiệm 6 nghiên cứu tác động của việc thay thế ba tàu kéo này với 3 tàu kéo tốc độ cao hơn v…v.

Như đề nghị trong hình 3.7, mỗi thử nghiệm kèm theo một hoặc nhiều lần lặp lại (lần chạy mô phỏng). Mỗi lần lặp lại là một lần thực hiện của một mô hình mô phỏng trong đó kết hợp logic và dữ liệu của mô hình cho thử nghiệm này nhưng sử dụng một tập hợp các số ngẫu nhiên duy nhất cho lần lặp lại này. Mỗi lần lặp lại thống kê được các kết quả khác nhau. Kết quả thống kê có thể được phân tích sau một loạt sự lặp lại.

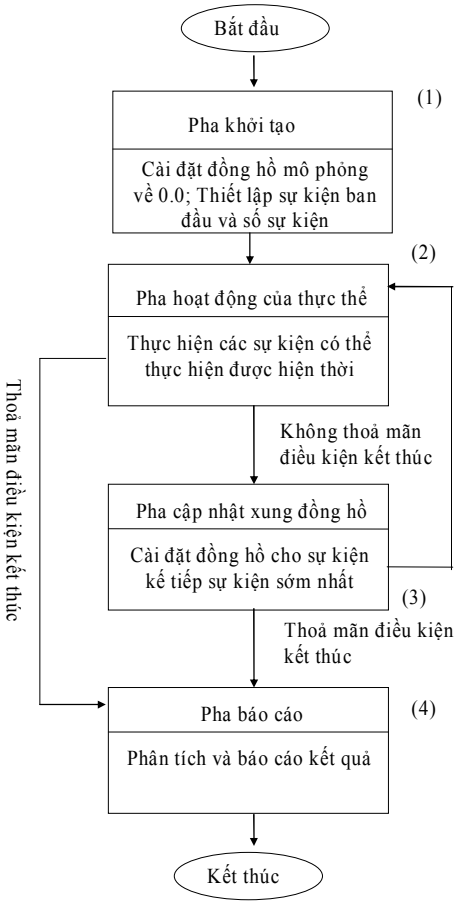
Trong một vài thiết kế thử nghiệm có sử dụng kỹ thuật "*variance reduction*", tập hợp các số ngẫu nhiên sử dụng trong một lần lặp lại có thể tương quan với những giá trị tương ứng ở một lần lặp lại cùng lúc khác. Hơn nữa trong một vài thiết kế, chỉ một lần lặp lại tương đối dài có thể được thực hiện.



Hình 3.7 Thử nghiệm và sự lặp lại trong thiết kế mô phỏng

3.4.2 Cơ cấu hoạt động của một lần lặp lại

Mục này thảo luận các giai đoạn của một lần lặp lại. Để cụ thế hóa, các vấn đề sẽ được giải thích cho trường hợp bến cảng ở hình 3.4.



Hình 3.8 Cơ cấu hoạt động của một lần lặp lại

Giai đoạn khởi tạo: Lưu đồ hình 3.8 trình bày các giai đoạn của một lần lặp lại. Lần lặp lại này bắt đầu với 1 giai đoạn khởi tạo với đồng hồ mô phỏng luôn bắt đầu giá trị 0.0 Thời gian mô phỏng sẽ được tính tương ứng với giá trị khởi đầu này Ví dụ như, thời gian 0.0 có thể tương ứng với thời gian thực là 9h sáng trong ngày đầu tiên của một chuỗi ngày mô phỏng của một sự lặp lại.

Khi bắt đầu giai đoạn khởi tạo, không có thực thể nào tồn tại. Trong suốt quá trình khởi tạo, một hay nhiều thực thể ngoại được tạo ra và sự kiện tiến đến (arrival event) của chúng ngay lập tức được lên kế hoạch, như trong hình vẽ 3.8 (khối 1). Ví dụ trong hệ thống bến cảng như hình 3.4, tàu loại A đầu tiên đi vào bến cảng được tạo ra trong giai đoạn khởi tạo và thời điểm sự kiện của việc đi vào này, chẳng hạn như thời gian mô phỏng là 23.5 phút, sẽ được xác lập. Sự xác lạp này thực hiện bởi việc lấy một mẫu giá trị từ phân bố thời gian giữa các lần đến của các tàu loại A. Đây chính là thời gian mô phỏng để tàu loại A cập bến tương ứng với thời gian bắt đầu là 0.0 (để thuận tiện chúng ta mặc định đơn vị thời gian là phút nhưng cũng có thể là đơn vị khác tuỳ thuộc vào người thiết kế mô hình). Trong giai đoạn khởi tạo, tàu loại B đầu tiên đi vào bến cảng cũng được tạo ra và thời gian đến của nó, giả thiết tại thời giam mô phỏng là 18.2 phút, được xác lập.

Thảo luận trên đối với việc tạo ra các thực thể ngoại trong giai đoạn khởi tạo. Nếu mô hình này cũng có các thực thể nội thì chúng cũng cần được khởi tạo. Ví dụ như, giả sử rằng mô hình bến cảng trong hình 3.4 được xây dựng với phần mềm sử dụng một thực thể nội để điều khiển khoảng thời gian mô phỏng và sự lặp lại này sẽ thực hiện tới 43,200 (số phút của 30 ngày với 1 ngày có 3 giờ). Một thực thể nội được tạo ra tương ứng trong pha khởi tạo.

Giai đoạn thực thể hoạt động**:** Như trong hình 3.8, giai đoạn thực thể hoạt động tiếp theo sau pha khởi tạo và trước pha cập nhật xung đồng hồ. Mục đích của giai đoạn thực thể hoạt động là các thực thể đủ điều kiện thực hiện các hoạt động của chúng tại thời điểm mô phỏng. Một thực thể đủ điều kiện để hoạt động nếu thời điểm hoạt động dự kiến của nó bằng với thời gian mô phỏng hiện tại.

Lưu ý: Các nhóm từ như "thời điểm hoạt động", "thời điểm di chuyển" và "thời điểm sự kiện" thường được dùng thay thế cho nhau.

Giai đoạn thực thể **hoạt động đầu tiên: Giai đoạn thực thể** hoạt động này được bắt đầu từ giai đoạn khởi tạo tại thời điểm mô phỏng 0.0. Nếu không có thực thể khởi tạo nào (thực thể nội hoặc thực thể ngoại) dự định bắt đầu thực hiện lúc 0.0 thì sẽ không có giai đoạn thực thể hoạt động tại thời điểm 0.0. Trong mô hình bến cảng chúng ta cho thời gian tới của tàu loại A và B lần lượt là 23.5 và 18.2 phút. Trong ví dụ này không có hoạt động nào của thực thể ngoại trong giai đoạn thực thể hoạt động đầu tiên này.

Thông thường, cần có hoạt động nào đó tại giai đoạn thực thể hoạt động đầu tiên. Ví dụ, nếu nhà băng mở cửa hoạt động lúc 9 giờ sáng (thời điểm mô phỏng là 0.0), có thể có 3 khách hàng- là các thực thể đang đợi nhà băng mở cửa. Ba thực thể này sẽ lần lượt hoạt động trong giai đoạn thực thể hoạt động đầu tiên.

Các giai đoạn thực thể hoạt động sau đó: Hoạt động luôn diễn ra trong các giai đoạn thực thể hoạt động sau khi thực hiện các giai đoạn cập nhật đồng hồ. Đó là do một giai đoạn cập nhật đồng hồ luôn đưa trạng thái của mô hình tới thời điểm mô phỏng sớm nhất sau đó mà tại đó ít nhất một thực thể được lập lịch thực hiện hoạt động.

Trong mô hình bến cảng này, chũng ta giả thiết thời gian cập bến của tàu loại B là 18.2 phút. Thời gian bắt đầu giai đoạn thực thể hoạt động là 0.0 (tại đó không có hoạt dộng nào diễn ra), giai đoạn cập nhật xung đồng hồ thiết lập thời gian là 18.2 (thời gian sớm nhất để thực hiện hành động theo lịch trình: thời gian để 1 tàu loại A theo như lịch trình thực hiện cập bến sau đó và lúc 23.5 phút. Vậy rồi giai đoạn thực thể hoạt động tiếp theo xảy ra và tàu loại B đầu tiên giành lấy một bến đỗ trống, một tàu kéo rỗi và khởi tạo quá trình đi vào bến cảng bằng tàu kéo.

Các giai đoạn **cập nhật xung đồng hồ (CUP):** Sau khi các hoạt động có thể có được thực hiện ở một giai đoạn thực thể hoạt động, một CUP sẽ diễn ra. Mục đích của CUP thiết lập đồng hồ tới thời điểm mô phỏng tiếp theo sớm nhất mà tại thời điểm đó đã có một hoặc vài hoạt động được lập lịch trước. Thời điểm mô phỏng tiếp theo sớm nhất này có thể là thời điểm mô phỏng hiện thời, tùy thuộc vào liệu hai hoặc nhiều thực thể có được lập lịch tài cùng một thời điểm mô phỏng này hay không và cũng phụ thuộc vào các lựa chọn bởi người thiết kế ngôn ngữ mô phỏng. Hoặc thời gian mô phỏng tiếp theo sớm nhất có thể là thời điểm sau đó. (Đồng hồ mô phỏng không bao giờ giảm giá trị. Mô phỏng các sự kiện rời rạc không được thiết kế để có thể quay ngược thời điểm mô phỏng).

Sau khi kết thúc một giai đoạn cập nhật xung đồng hồ, giai đoạn thực thể hoạt độngđược thực hiện trở lại để các thực thể đủ điều kiện có thể thực hiện các hoạt động tại thời điểm mô phỏng mới được thiết lập. Sau đó giai đoạn cập nhật xung đồng hồ tiếp tục thực hiện trở lại, rồi đến giai đoạn thực thể hoạt động tiếp theo và cứ thế tiếp tục.

Điểm chính của một sự lặp lại việc hoạt động lien tiếp nhau của giai đoạn thực thể hoạt động và giai đoạn cập nhật xung đồng hồ. Trong suốt giai đoạn thực thể hoạt động, thời gian mô phỏng giữ nguyên cố định và thời gian thực trôi qua trong khi trạng thái của mô hình được cập nhật (hình 3.6).

Thu thập các kết quả thống kê**: Một mục tiêu** của mô phỏng các sự kiện rời rạc là để thu thập các kết quả thống kê về hoạt động của hệ thống được mô phỏng. Phần mềm mô phỏng các sự kiện rời rạc được thiết kế để thu thạp tự động nhiều loại kết quả thống kê khác nhau trong một sự lặp lại. Đối với các tài nguyên, các giá trị thống kê chẳng hạn như số lần giành được, thời gian chiếm giữ trung bình và mức độ khai thác có thể được đo đạc tự động. ĐỐi với các hang chờ, giá trị trung bình, giá trị thống kê là thời gian chờ trung bình và chiều dài hang chờ lớn nhất có thể được đo tự động. . Ngoài ra việc thu thập các giá trị thống kê một cách tự động, phần mềm mô phỏng cũng cung cấp công cụ để người làm mô phỏng có thể thu thập chúng một cách tùy ý.

Sự theo dõi và ghi lại các giá trị thống kê thường được thực hiện trong giai đoạn thực thể hoạt động (theo cách tự động bằng phần mềm hoặc và tùy ý người dung). Kết quả được sử dụng trong báo cáo cuối cùng của sự lặp lại.

Kết thúc một lần lặp lại: Khi một lần lặp lại được thực hiện, một điều kiện kết thúc thương xảy ra trong giai đoạn thực thể hoạt động hoặc trong giai đoạn cập nhật xung đồng hồ. Điều kiện kết thúc có thể dựa vào thời gian (ví dụ, bến cảng hoạt động 3 giờ) hoặc theo số lượng đếm được (ví dụ, lô hàng sản xuất gồm 500 sản phẩm) hoặc có thể nhiều yếu tố khác (ví dụ, ngân hàng mở cửa vào buổi sáng, đóng cửa vào buổi chiều và tất cả các khách hàng có mặt trong ngân hang khi cửa đóng đã được phục vụ xong). Điều kiện kết thúc có thể xãy ra hoặc ở giai đoạn thực thể hoạt động hoặc ở giai đoạn cập nhật xung đồng hồ.

Sau đó giai đoạn báo cáo được thực hiện và kết thúc lần lặp lại này. Trạng thái của mô hình được mô tả khi kết thúc lần lặp lại này. Chẳng hạn như mô tả giá trị của đồng hồ mô phỏng, số lượng các thực thể khác nhau dung trong mô hình, số thực thể hiện tại trong mô hình. Các kết quả thống kê của lần lặp lại này thường dưới dạng các báo cáo, bao gồm kết quả thống kê về tài nguyên và hang chờ.

3.5. Các trạng thái của thực thể

Một thực thể được tạo ra thời điểm được mô phỏng, hoạt động ở trong một mô hình trong khi thời gian mô phỏng tăng lên và sau đó được hủy đi. Trong chu trình tồn tại của nó, thực thể thay đổi từ trạng thái này sang trạng thái khác, thường nhiều lần đi qua các trạng thái khác nhau trước khi bị hủy. (Đó không phải là yêu cầu bắt buộc, tuy nhiên, thực thể cuối cùng sẽ bị hủy. Một vài thực thể có thể lặp lại sự quay vòng qua các phần của mô hình với các giai đoạn ngừng tùy theo thiết kế mô hình).

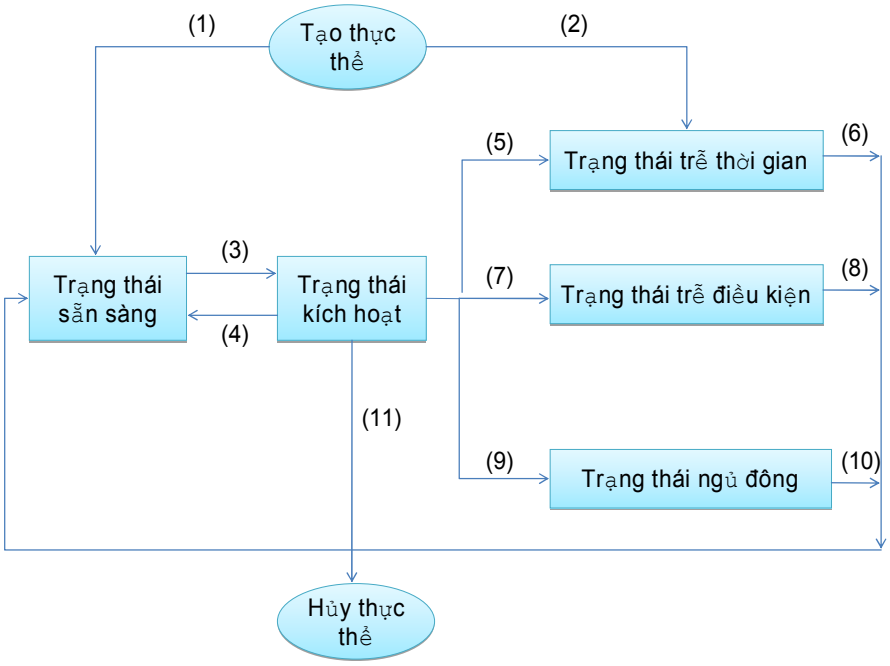
Có năm trạng thái của thực thể luân phiên nhau: *trạng thái sẵn sàng*, *trạng thái hoạt đọng*, *trạng thái trễ thời gian, trạng thái trễ có điều kiện và trạng thái không hoạt động*, được chỉ ra trong hình 3.9. Hình này cũng chỉ ra các thời điểm tạo ra thực thể và hủy thực thể, cũng như là đường dẫn mà các thực thể di chuyển từ trạng thái này sang trạng thái khác. Các đường dẫn này được đánh số để hỗ trợ cho các phần sau. Sau đây trình bày nội dung về *trạng thái sẵn sàng*, *trạng thái hoạt đọng*, *trạng thái trễ thời gian, trạng thái trễ có điều kiện và trạng thái không hoạt động cũng như về việc tạo và hủy thực thể.*

3.5.1. Trạng thái hoạt động

Có thể chỉ có một thực thể di chuyển tại một thời điểm bất kỳ tính theo đồng hồ thời gian thực. Trạng thái hoạt động là trạng thái của thực thể hiện tại đang di chuyển. Thực thể ở trạng thái hoạt động (thực thể hoạt động) di chuyển không ngừng đến khi nó bị trễ, hoặc bị hủy, hoặc tạo trạng thái kích hoạt cho một vài thực thể khác trước khi trở lại trạng thái kích hoạt của nó tại cùng thời gian mô phỏng. Thời gian mô phỏng giữ nguyên trong khi một thực thể ở trạng thái hoạt động.

Nếu thực thể hoạt động này bị trễ, nó chuyển từ trạng thái hoạt động đến một trong ba trạng thái trễ thay thế khác: trạng thái trễ thời gian (đường số 5 trong hình 3.9), trạng thái trễ điều kiện (đường số 7), hoặc trạng thái không hoạt động (đường số 9). Nếu thực thể hoạt động này tạo ra một sự di chuyển mà gây ra việc hủy bỏ chính nó (đường số 11 trong hình 3.9), nó sẽ được xóa ngay lập tức trong mô hình và không còn tồn tại.

Trong một vài mô hình, thực thể hoạt động này có thể tạm thời rời trạng thái hoạt động và sau đó nó sẽ trở lại trạng thái hoạt động sau tại cùng thời điểm mô phỏng. Mục đích của trường hợp này là để cho một hoặc nhiều thực thể khác trải qua trạng thái hoạt động này để hoàn thành một hoặc nhiều nhiệm vụ trước khi thực thể khởi tạo này một lần nữa trở lại trạng thái hoạt động của chính bản thân nó tại cùng thời điểm mô phỏng. Thực thể khởi tạo này thực hiện việc tạm ngừng trạng thái hoạt động này bằng việc di chuyển từ trạng thái hoạt động về trạng thái sẵn sàng (đường số 4 hình 3.9).



Hình 3.9 Các trạng thái của thực thể và các đường di chuyển

3.5.2. Trạng thái sẵn sàng

Tại thời điểm mô phỏng nào đó, có thể có nhiều hơn một thực thể sẵn sàng di chuyển, nhưng chỉ có một thực thể tại một thời điểm có thể ở trong trạng thái hoạt động. Nếu có hai hoặc nhiều thực thể muốn di chuyển tại thời điểm mô phỏng đã được thiết lập, chỉ có một thực thể phải chờ đến lượt để vào trạng thái hoạt động tại thời điểm mô phỏng hiện tại. Thời gian mô phỏng giữ nguyên khi một thực thể đang ở trong trạng thái sẵn sàng.

Có một số đường trong hình 3.9 mà các thực thể có thể di chuyển sang trạng thái sẵn sàng. Ví dụ: việc di chuyển từ các trạng thái trễ thời gian, trễ có điều kiện, và trạng thái không hoạt động luôn tới trạng thái sẵn sàng (các đường theo thứ tự 6, 8, 10). Trong số các khả năng đó, việc di chuyển từ trạng thái trễ thời gian và trễ có điều kiện sang trạng thái sẵn sàng xảy ra thường xuyên nhất.

Di chuyển từ trạng thái không hoạt động đến trạng thái sẵn sàng từ (đường số 10) ít xảy ra hơn, vì các mô hình không thường xuyên sử dụng trạng thái không hoạt động. Đường dẫn từ việc tạo ra thực thể đến trạng thái sẵn sàng (Đường số 1) chỉ được thực hiện bởi các thực thể đã sẵn sàng để di chuyển tại thời điểm mô phỏng của việc tạo thực thể. Đường từ trạng thái hoạt động đến trạng thái sẵn sàng (đường số 4) là ít xảy ra nhất. Hầu hết các mô hình không yêu cầu sử dụng đường di chuyển này. (Một vài phần mềm mô phỏng sự kiện rời rạc không cung cấp đường này).

3.5.3. Trạng thái trễ thời gian

Trạng thái trễ thời gian là trạng thái các thực thể đang chờ đến thời điểm mô phỏng nào đó đã xác định trong tương lai mà chúng có thể trở lại trạng thái sẵn sàng mào đầu cho việc di chuyển trở lại. Các thực thể chuyển sang trạng thái trễ thời gian từ trạng thái hoạt động (Đường số 5, hình 3.9) và từ điểm tạo thực thể (Đường số 2). Sau đó chúng chuyển từ trạng thái trễ thời gian sang trạng thái sẵn sàng (Đường số 6) khi thời gian mô phỏng biết trước của chúng xảy ra.

Hãy xem xét một ví dụ về sự ảnh hưởng lẫn nhau giữa các trạng thái trễ thời gian, trạng thái hoạt động và trạng thái sẵn sàng. Giả sử rằng, một thực thể đang mô phỏng sản phẩm đang trong quá trình sản xuất. Sản phẩm này cần được khoan một lỗ và giả sử rằng khi thực thể sẵn sang cho việc này thì máy khoan đang rỗi. Tại thời điểm mô phỏng đó, thực thể này chuyển từ trạng thái sẵn sàng sang trạng thái hoạt động (đường 3 hình 3.9), giành lấy máy khoan và sau đó chuyển sang trạng thái trễ thời gian (đường 5), và giữ nguyên trạng thái đó rồi chờ đến khi thời gian mô phỏng xác định trong tương lai khi mà hoạt động khoan này kết thúc. (Thực thể "biết" thời gian mô phỏng trong tương lai vì nó lấy mẫu thời gian của việc khoan lỗ từ sự phân bố thời gian khoan trong lúc nó chuyển sang trạng thái trễ thời gian).

Mở rộng ví dụ này, chúng ta giả sử rằng, đã đến thời gian mô phỏng khi các việc khoan lỗ kết thúc. Sau đó thực thể này chuyển từ trạng thái thời gian trễ sang trạng thái sẵn sàng (đường số 6 hình 3.9) như là sự mào đầu cho việc di chuyển. Sau đó, thực thể chuyển từ trạng thái sẵn sàng sang trạng thái hoạt động và thay đổi trạng thái của máy khoan "đang bị chiếm dụng" sang "rỗi". Giả sử thực thể này sau đó cần được chuyển tới một đích khác bằng AGV và cũng giả sử rằng AGV đang rỗi. Thực thể này tiếp tục trạng thái hoạt động: thực thể giành lấy một AGV và chuyển về trạng thái trễ thời gian (đường số 5) trong khi chờ cho thời gian mô phỏng được xác định trước trong tương lai khi hoạt động truyền tải này sẽ kết thúc.

Khi một thực thể được tạo ra, nó bắt đầu sự tồn tại của nó trong trạng thái sẵn sàng hoặc trong trạng thái trễ thời gian, phụ thuộc vào việc liệu thời gian mô phỏng khi nó sẽ di chuyển lần đầu tiên bằng hay vượt qua thời gian tạo ra nó. Nếu nó di chuyển lần đầu tiên tại thời điểm tạo nó, thực thể bắt đầu trạng thái sẵn sàng (đường số 1 hình 3.9), nếu không, nó bắt đầu bằng trạng thái trễ thời gian (đường 2). Các thực thể thường xuyên bắt đầu sự tồn tại của mình trong trạng thái trễ thời gian.

Một ví dụ về việc tạo thực thể. Giả sử rằng, thời gian giữa các sự kiện đến của các cuộc gọi thoại đến một nhà khai thác mạng là hàm phân phối mũ, và một thực thể dung để mô phỏng một cuộc gọi. Khi một cuộc gọi đến, thời gian trong tương lai mà một cuộc gọi tiếp theo sẽ đến được quyết đinh bởi việc lấy mẫu từ hàm phân phối mũ của khoảng thời gian giữa hai cuộc gọi. Thực thể gọi tương ứng được tạo và đặt vào trong trạng thái trễ thời gian (đường số 2 hình 3.9), và chờ thời gian mô phỏng trong tương lai đã được xác định tại lần di chuyển đầu tiên ("đến", làm cho điện thoại đổ chuông).

Ví dụ thứ hai về tạo thực thể, giả sử rằng, khi ngân hàng mở cửa vào 9h sáng (thời gian mô phỏng là 0.0), ba khách hàng đang chờ mở cửa. Mỗi một khách hàng sẽ vào trong ngân hàng khi cửa mở. (Trong thực tế, một vài giây sẽ trôi qua trong khi những khách hàng này vào ngân hàng từng người một, nhưng chúng ta hãy giả sử rằng cả ba khách hàng đều vào đồng thời). Tại thời điểm mô phỏng 0.0, khách hàng (thực thể) đầu tiên được tạo và cho vào trạng thái sẵn sàng (đường 1 hình 3.9). Thực thể khách hàng thứ 2 cũng sẽ được tạo tại thời điểm 0.0 và đặt vào trạng thái sẵn sàng. Cuối cùng, thực thể khách hàng thứ 3 cũng sẽ được tạo tại thời điểm 0.0 và ở trong trạng thái sẵn sàng. Trong suốt khoảng thời gian thực thể di chuyển tại thời điểm 0.0, mỗi một khách hàng sẽ chuyển một cách tuần tự từ trạng thái sẵn sàng sang trạng thái hoạt động và bắt đầu hoạt động.

3.5.4. Trạng thái trễ có điều kiện

Trạng thái trễ có điều kiện là trạng thái của các thực thể mà sự di chuyển của chúng gặp trở ngại vì một vài điều kiện. Các thực thể đợi trong trạng thái trễ có điều kiện cho đến khi điều kiện trễ của chúng được giải quyết. Trong khi trễ có điều kiện, các thực thể không biết khi nào điều kiện trễ sẽ được giải quyết, do đó chúng chờ trong một khoảng thời gian mô phỏng không xác định trước. (trái ngược với trạng thái trễ có điều kiện là trạng thái trễ thời gian, trong đó các thực thể chờ trong khoảng thời gian mô phỏng đã biết trước).

Một ví dụ của trạng thái trễ có điều kiện, xem xét một sản phẩm đang trong quá trình sản xuất (WIP), cần khoan một lỗ. Giả sử rằng khi sản phẩm này sẵn sàng cho hoạt động này, máy khoan không rỗi (Máy khoan đang được sử dụng bởi một sản phẩm khác). Thực thể hoạt động WIP này cố gắng chiếm lấy máy khoan nhưng gặp trở ngại. Giả sử rằng, nó không có lựa chọn nào khác, nó sẽ chuyển sang trạng thái trễ có điều kiện (đường số 7 hình 3.9) và bắt đầu chờ đến lượt để sử dụng máy khoan. (Có thể có một vài thực thể WIP khác cũng đang chờ sử dụng máy khoan). Thực thể WIP ở trong trạng thái trễ có điều kiện cho đến khi máy khoan trở nên rỗi và đến lượt nó sử dụng máy khoan. Tại thời điểm đó thực thể chuyển sang trạng thái sẵn sàng (đường số 8), và sau đó chuyển sang trạng thái hoạt động (đường số 3) và sau đó sẽ chiếm lấy máy khoan và chuyển sang trạng thái trễ thời gian (Đường số 5).

Như đã giới thiệu trong các ví dụ trước, trong một số trường hợp, một sự kiện đơn có thể là đủ cho việc thực hiện sự trễ. Đây là những trường hợp của sự trễ đơn giản. Nếu một hoặc nhiều WIP bị trễ và đang chờ đến lượt sử dụng máy khoan, thì khi người thao tác máy khoan hiện tại đặt máy khoan vào trạng thái rỗi, sự kiện đơn này thực hiện việc trễ cho sản phẩm tiếp theo đang chờ.

Việc chuyển từ trạng thái trễ có điều kiện sang trạng thái sẵn sàng dễ dàng liên quan đến sự kiện thực hiện trễ trong các trường hợp này.
Một ví dụ về trạng thái trễ có điều kiện phức hợp hơn: hãy xem xét một thực thể tàu đang chuyển hàng hóa tại bến cảng rồi sẵn sàng rời khỏi cảng. Giả sử rằng, cần thỏa mãn hai điều kiện đối với con tàu để rời khỏi cảng: (1) cần tàu kéo và (2) thời tiết tốt. Vẫn giữ nguyên trạng thái hoạt động sau khi hoàn tất việc chuyển hàng hóa, thực thể tàu này kiểm tra nếu liệu có tàu kéo nào đang rỗi hay không và thời tiết có tốt không. (Một chuyển mạch on/off có thể được sử dụng như phần tử điều khiển báo hiệu trạng thái của thời tiết). Nếu điều kiện kết hợp này được thỏa mãn, thực thể tàu chuyển từ trạng thái hoạt động sang trạng thái trễ thời gian (đường số 5) với một khoảng thời gian đã biết được yêu cầu cho việc rời khỏi. Nếu không thì nó chuyển từ trạng thái hoạt động sang trạng thái trễ có điều kiện (đường số 7) và bắt đầu chờ một khoảng thời gian không xác định hai điều kiện trên được thỏa mãn đồng thời.

Trong trường hợp này chúng ta có một ví dụ về sự trễ phức hợp. Độ chính xác của trễ phức tạp không thể liên quan đến trường hợp của một sự kiện đơn lẻ. Khi tàu kéo rỗi, không có nghĩa là thời tiết sẽ tốt. Khi thời tiết tốt không có nghĩa là tàu kéo rỗi. Việc giải quyết trễ phức hợp khó hơn nhiều so với trễ đơn.

3.5.5. Trạng thái không hoạt động (ngủ đông)

Giống như trạng thái trễ thời gian và trễ có điều kiện, trạng thái không hoạt động là một trạng thái mà các thực thể được đặt vào trạng thái dừng trong một khoảng thời gian mô phỏng nào đó. Các thực thể ở trong trạng thái không hoạt động người thiết kế mô hình thay vì quản lý tự động bởi chương trình phần mềm mô phỏng. Cụ thể như sau.

Các trạng thái trễ có điều kiện, trễ thời gian, trạng thái hoạt động và trạng thái sẵn sàng được quản lý theo các quy tắc được xây dựng trong phần mềm mô hình hóa. Ví dụ như các thực thể đủ tiêu chuẩn tự động được truyền tại thời điểm mô phỏng tới trong trạng thái trễ thời gian và sau đó vào trạng thái sẵn sàng.

Tương tự, các thực thể đủ tiêu chuẩn được tự động truyền vào trạng thái trễ có điều kiện và cuối cùng từ đó chuyển vào trạng thái sẵn sàng khi các điều kiện được thỏa mãn. Người thực hiện mô hình hóa có thể thực hiện tùy ý nhưng bị giới hạn nhất định Ví dụ, trong năm thực thể đang ở trong trạng thái trễ có điều kiện, đang chờ một thiết bị. Khi thiết bị rỗi, thực thể nào trong năm thực thể là người tiếp theo được sử dụng thiết bị? Vậy thì trình tự phục vụ sẽ ra sao? Người thực hiện mô hình hóa có thể dùng một trong số các trình tự phục vụ nào đó: ví dụ: ai đến trước thì được phục vụ trước, hoặc dựa trên độ ưu tiên, hoặc ai đến sau thì phục vụ trước.

Ví dụ về việc sử dụng trạng thái không hoạt động, giả sử một hệ thống của nhà sản xuất sử dụng trình tự phục vụ dạng "thời gian đáo hạn ngắn nhất" để quyết định công việc đang đợi nào sẽ được sử dụng máy tiếp theo. Thời gian đáo hạn là đơn vị đo dựa trên mối quan hệ giữa ngày tháng đến hạn của công việc, thời điểm hiện tại (ví dụ: thời điểm tại đó khoảng thời gian đáo hạn đang được tính toán) và khoảng thời gian hoạt động còn lại của công việc. (khoảng thời gian hoạt động còn lại của công việc được tính bởi tổng thời gian xử lý còn lại được yêu cầu bởi công việc trước khi nó rời khỏi hệ thống khi công việc được hoàn thành). Khoảng thời gian đáo hạn được tính như sau:
Khoảng thời gian đáo hạn = (ngày đến hạn) – (thời điểm hiện tại) – (thời gian hoạt động còn lại)

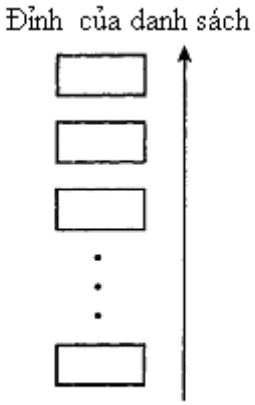
Khoảng thời gian đáo hạn của một công việc càng nhỏ thì càng phải nhanh chóng hoàn thành công việc đúng với ngày đến hạn. Chú ý rằng, thời gian đáo hạn giảm đi trong khi thời gian mô phỏng (thời điểm hiện tại) trôi qua. Do đó thời gian đáo hạn của một công việc không được tính toán khi một công việc "đến" máy. Thay vào đó, thời gian còn lại được tính toán "sau đó", tại thời điểm máy bắt đầu một công việc khác. Trong khi chờ thời điểm mới này, các công việc trễ có thể giữ ở trạng thái không hoạt động. Khi đến thời điểm để máy bắt đầu công việc khác, thời gian đáo hạn của mỗi công việc đang chờ ở trạng thái không hoạt động được tính toán và công việc với thời gian đáo hạn ngắn nhất được truyền từ trạng thái không hoạt động sang trạng thái sẵn sàng, để giành sử dụng máy.

Bảng 3.1 Các trạng thái của thực thể

Trạng thái	Mô tả
Hoạt Động	Trạng thái hoạt độnglà trạng thái của sự di chuyển hiện tại của thực thể.
Sẵn Sàng	Trạng thái sẵn sàng là trạng thái của các thực thể chờ chuyển sang trạng thái hoạt động tại thời điểm mô phỏng hiện tại.
Trễ Thời Gian	Trạng thái trễ thời gian là trạng thái của các thực thể đang chờ trong một khoảng thời gian mô phỏng biết trước, sau khoảng thời gian đó chúng sẽ chuyển sang trạng thái sẵn sàng.
Trễ có Điều Kiện	Trạng thái trễ có điều kiện là trạng thái của các thực thể đang chờ trong một khoảng thời gian mô phỏng không biết trước khi điều kiện gây nên trễ được giải quyết. Khi các điều kiện được thỏa mãn, các thực thể đó sẽ được truyền tự động từ trạng thái trễ có điều kiện sang trạng thái sẵn sàng theo các thiết kế được xây dựng trong phần mềm.
Không hoạt động	Trạng thái không hoạt động là trạng thái của các thực thể đang chờ trong một khoảng thời gian mô phỏng chưa biết trước khi điều kiện gây ra trễ được giải quyết. Khi các điều kiện được thỏa mãn, các thực thể đó sẽ được truyền từ trạng thái không hoạt động sang trạng thái sẵn sàng theo logic do người thiết kế mô hình đặt ra

3.6 Quản lý thực thể

Phần mềm mô phỏng quản lý các thực thể bằng cách tổ chức chúng thành các danh sách tuyến tính. Mỗi danh sách tương ứng với một trang thái thực thể và được sắp xếp theo thứ tự, với một thực thể nào đó ở trên đỉnh của danh sách và các thực thể khác ở phía sau nó, tới tận cuối danh sách. Hình 3.10 mô tả một danh sách thực thể khái quát, trong đó mỗi một ô hình chữ nhật đại diện cho một thực thể.



Hình 3.10 Danh sách thực thể

Sự sắp xếp thứ tự của danh sách thực thể nêu lên những câu hỏi về các quy luật được sử dụng để xác định thứ tự và vai trò của việc sắp xếp thứ tự trong quản lý danh sách. Khi một thực thể được thêm vào trong một danh sách, làm thế nào để xác định được điểm chèn vào của nó? Khi một thực thể được lấy ra khỏi danh sách, thực thể đó được lấy ra từ vị trí nào? Các thực thể có thể thay đổi vị trí tương quan của chúng trong một danh sách hay không ? Chi tiết về năm loại danh sách thực thể sẽ được miêu tả trong những phần sau đây.

3.6.1 Danh sách thực thể hoạt động (Active – Entity List)

Ở đó chỉ có một thực thể ở trạng thái hoạt động, như vậy danh sách này chỉ có một phần tử. Do đó từ "danh sách" không được dung ở đây. Thực thể hoạt động này di chuyển không ngừng tại

thời điểm mô phỏng cho tới khi nó chuyển sang trạng thái khác (chuyển sang sang danh sách khác) hoặc bị huỷ bỏ.

3.6.2 Danh sách sự kiện hiện thời (Current Events List)

Những thực thể ở trạng thái sẵn sàng được tạo thành một danh sách đơn gọi là danh sách sự kiện hiện tại (CEL). Cái tên này phản ảnh rằng những thực thể có trạng thái sẵn sàng đuộc chuẩn bị sẵn để di chuyển tại thời điểm mô phỏng hiện tại và như thế sự di chuyển của chúng sẽ tạo ra các sự kiện (ví dụ như: sự kiện đến, sự chiếm tài nguyên, sự rời bỏ) xuất hiện trong mô hình.

Những quy luật khác nhau có thể được sử dụng để xác định điểm chèn các thực thể vào trong danh sách sự kiện hiện tại. Trong một vài trường hợp, các thực thể di chuyển tới CEL, có thể được đặt vào cuối danh sách. Trong những trường hợp khác, các thực thể có thể được sắp xếp vào CEL theo thứ tự giảm dần khi mà thứ tự ưu tiên là một thuộc tính của thực thể. Thứ tự ưu tiên có thể được giải quyết theo cách chèn thực thể mới vào trong danh sách dưới những thực thể với việc so khớp thứ tự ưu tiên. Trong những trường hợp còn khác, các thực thể có thể được đặt vào đỉnh của CEL. Khi tới thời điểm chuyển một thực thể từ trạng thái sẵn sàng sang trạng thái hoạt động, thực thể thường được chuyển đi khỏi đỉnh của danh sách sự kiện hiện tại.

3.6.3 Danh sách sự kiện tương lai

Những thực thể trễ thời gian tạo ra một danh danh sách đơn gọi là danh sách sự kiện tương lai (FEL). Cái tên này phản ánh rằng những thực thể này sẽ không cố gắng di chuyển nữa cho tới khi đến thời điểm mô phỏng nào đó trong tương lai đã được biết trước. Danh sách những biến cố tương lai là một kiểu sắp xếp theo kiểu từ trên xuống (top - down) theo sự tăng dần của thời gian di chuyển của thực thể. (Thời gian di chuyển của thực thể là thời gian được mô phỏng tại đó thực thể sẽ thử tạo ra sự di chuyển tiếp của mình hoặc tạo ra một tập những di chuyển trong một mô hình). Khi một thực thể được chèn vào trong FEL, thời gian di chuyển của nó được tính toán bằng cách cộng thời gian của đồng hồ mô phỏng với khoảng thời gian đã biết (đã lấy mẫu) của sự trễ thời gian này. Trật tự về thời gian di chuyển thường được thực hiện theo phương thức "vào trước, ra trước – FIFO".

Khi một giai đoạn cập nhật đồng hồ xảy ra, thời gian mô phỏng tiếp theo sớm nhất (là khi mà một di chuyển mới xảy ra trong mô hình) bằng thời điểm di chuyển của thực thể nằm tại đỉnh của danh sách sự kiện tương lai. Giai đoạn cập nhật đồng hồ này thiết lập thời gian mô phỏng bằng giá trị thời điểm di chuyển của thực thể này và sau đó chuyển thực thể ra khỏi danh sách FEL và rồi chèn nó vào danh sách các sự kiện hiện thời. Khi một giai đoạn cập nhật đồng hồ bắt đầu, hai hoặc nhiều thực thể tại đỉnh của danh sách sự kiện hiện thời có thể có cùng thời gian di chuyển. Trật tự thời gian di chuyển trong trường hợp này sẽ tùy thuộc vào phần mềm mô hình hóa. Một phương pháp áp dụng là mỗi thực thể có cùng trật tự thời gian này từ FEL sang CEL trong một giai đoạn cập nhật đồng hồ.

Một phương pháp khác là trong trường hợp của trật tự thời gian giống nhau, liên tiếp hai hoặc nhiều hơn giai đoạn cập nhật đồng hồ và các giai đoạn di chuyển thực thể sẽ được thực hiện tại cùng thời điểm mô phỏng.

Những ngôn ngữ mô phỏng có dụng các thực thể nội thương sử dụng FEL để hỗ trợ những thực thể như vậy. Trong đó, FEL thường bao gồm cả thực thể ngoại và thực thể nội. Trong suốt một mô phỏng sự kiện rời rạc, phần mềm mô phỏng như vậy thường chèn các thực thể vào FEL. Nếu danh sách này trở nên dài thì sự tìm ra điểm chèn có thể làm mất khá nhiều thời gian tính toán. Điều này thúc đẩy người thiết kế ngôn ngữ pháp triển những thuật toán có hiệu năng cao trong việc tìm kiếm những điểm chèn trong các danh sách. Và kết quả là, một vài phần mềm mô hình hóa dùng FEL không tuyến tính **(linear), mà** thay vào đó dung các loại cấu trúc dữ liệu khác. Tuy nhiên, thực tế hóa thì chúng ta có thể hình dung FEL như là tuyến tính.

3.6.4 Các danh sách trễ

Các danh sách trễ bao gồm những thực thể trong trạng thái trễ có điều kiện. Những thực thể này đang chờ các điều kiện gây trễ được giải quyết để cho chúng có thể được tự động truyền sang trạng thái sẵn sang trong danh sách sự kiện hiện thời. Có thể có nhiều danh sách trễ trong một mô

hình, một (hoặc nhiều) danh sách cho mỗi điều kiện gây ra sự trễ. Điều này trái ngược với sự tồn tại duy nhất một "danh sách" trạng thái hoạt động, danh sách sự kiện hiện thời, và danh sách sự kiện tương lai. Ví dụ như trong hệ thống cảng ở hình 3.4, ở đó có thể là một danh sách trễ bao gồm những chiếc thuyền kiểu A chờ chỗ cập bến kiểu A; danh sách khác bao gồm các kiểu B chờ chỗ cập bến kiểu B; và danh sách khác bao gồm những tàu kiểu A và B đang chờ một hay nhiều tàu kéo.

Khi một thực thể được chèn vào một danh sách trễ, vị trí của nó bằng được xác định bằng việc sử dụng một tiêu chuẩn xếp hạng tạo bởi người xây dựng mô hình. Có thể có các lựa chọn khác nhau. Danh sách trễ đăc trưng bằng danh sách được sắp xếp FIFO (vào trước, ra trước); hoặc LIFO (vào sau, ra trước); hoặc được sắp xếp theo kiểu tăng dần hoặc giảm dần dựa trên một thuộc tính của thực thể do người dùng thiết lập hoặc một biểu thức số học. Tất nhiên là mỗi một phương thức sắp xếp đều có các tác động đối với thứ tự lấy các thực thể đang chờ ra khỏi danh sách trễ và được đặt vào trong trạng thái sẵn sàng vào danh sách sự kiện hiện tại.

Khi một điều kiện gây ra trễ được giải quyết, thực thể được xếp hạng cao nhất trong danh sách trễ tương ứng sẽ được phần mềm tự động chuyển thành trạng thái sẵn sàng thuộc danh sách sự kiện hiện thời.

3.6.5 Thời gian đợi liên kết và thời gian thăm dò

Cả hai phương pháp có thể được phần mềm mô phỏng sử dụng để di chuyển tự động một thực thể ra khỏi các danh sách trễ. Nếu sự trễ có thể dễ dàng được gắn với một sự kiện đơn mà có thể lọa bỏ sự trễ này thì phương pháp *thời gian đợi liên* kết có thể được sử dụng để quản lý danh sách trễ. Ví dụ, giả sử rằng trạng thái của máy chiếm giữ thay đổi từ trạng thái "bị chiếm giữ" sang trạng thái "rỗi". Để phản ứng trực tiếp tới sự thay đổi này của trạng thái máy, phần mềm có thể di chuyển thực thể đang chờ xếp hạng cao nhất ra khỏi danh sách trễ tương ứng và đặt nó vào trạng thái sẵn sàng trong danh sách sự kiện hiện thời. (thực thể này sẽ là thực thể tiếp theo sử dụng máy.)

Trong một vài trường hợp, giải pháp cho sự trễ có thể thể yêu cầu hai hoặc nhiều hơn sự kiện đặc biệt xuất hiện trong một mô hình. Việc xảy ra một trong só những sự kiện đó không nhất thiết có nghiã rằng vấn đề trễ đã được giải quyết. Các điều kiện cần được đảm bảo để chuyển một thực thể từ danh sách trễ sang trạng thái sẵn sàng, do đó không thể chỉ đơn giản là có liên quan tới việc xảy ra một sự kiện đơn. Trong những trường hợp này phương pháp thời gian thăm dò có thể được sử dụng để quản lý các thực thể trễ. Trong phương pháp thời gian thăm dò, một thực thể không được chuyển từ một danh sách trễ ngay khi xuất hiện một sự kiện đơn. Thay vào đó, phần mềm mô phỏng kiểm tra lần cuối cùng (với thời gian thực sau đó, nhưng cùng thời gian mô phỏng) để xem liệu sự kết hợp các điều kiện đã xảy đến chưa nhằm bảo đảm việc chuyển một hay nhiều thực thể trễ từ trạng thái trễ sang trạng thái sẵn sàng trên danh sách sự kiện hiện thời.

Một ví dụ cho phương pháp thời gian thăm dò: xem xét một cảng biển trong đó một tàu không thể di chuyển vào trong một chỗ cập bến tới khi có một bến rỗi và một tàu kéo cũng ở trạng thái rảnh rỗi. (giả thiết rằng bến cảng này được quản lý theo cách: trước tiên tàu không đề nghị một bến trống và sau đó sẽ hỏi xem có tàu kéo nào không, mà tàu chờ đợi cho tới khi cả bến và tàu kéo đều đồng thời rảnh rỗi trước khi đề nghị một trong số chúng. Chú ý rằng điều kiện logic ở đây là: "chờ cho tới khi một bến và một tàu kéo cùng rảnh rỗi." Sự thay đổi trạng thái một bến sang rảnh rỗi không nhất thiết triệt tiêu sự trễ. Sự thay đổi trạng thái của tàu kéo sang rảnh rỗi cũng không nhất thiết triệt tiêu sự trễ. Giải pháp của việc trễ không thể liên hệ với một trong số sự thay đổi trạng thái đơn. Vì vậy phương pháp thời gian thăm dò được sử dụng để xác định xem sự trễ có được giải quyết hay không. Thông thường việc kiểm tra thăm dò được thực hiện thường xuyên và tự động bởi phần mềm mô phỏng tại một điểm nào đó trong giai đoạn di chuyển thực thể (chẳng hạn như khi một giai đoạn di chuyển thực thể đang hoạt động).

3.6.6 Danh sách quản lý người sử dụng

Những danh sách quản lý người sử dụng quản lý bao gồm những thực thể trong trạng thái không hoạt động. Giống như danh sách trễ, có thể có nhiều danh sách được người dùng quản lý trong một mô hình. Tuy nhiên trái ngược với danh sách trễ, danh sách sự kiện hiện tại và tương lai mà được tạo ra và quản lý tự động bởi bằng phần mềm mô phỏng, người thiết kế mô hình phải tự

tạo ra các danh sách quản lý người sử dụng và cung cấp logic cần thiết đối với việc chèn thêm vào hay lấy các thực thể ra khỏi những danh sách đó.

Các thực thể tự quyết định xem có tự đưa chúng vào một danh sách quản lý người sử dụng hay không. Khi đang ở trong trạng thái hoạt động, chúng thực hiện các đánh giá do người sử dụng thiết kế ra để quyết định. Ví dụ như trong phạm vị trật tự dịch vụ kiểu "thời gian đáo hạn sớm nhất" được đưa ra ở phần 3.5.5, khi thao tác tiếp theo của thực thể là để sử dụng máy móc, nó có thể đánh giá xem máy này đang rảnh rỗi khong. Thực thể này có thể chiếm được máy nếu như máy đó đang rảnh rỗi hoặc có thể chuyển từ trạng thái hoạt động sang trạng thái không hoạt động trong danh sách quản lý người sử dụng nếu như máy này đang trong trạng thái bị chiếm.

Người thiết kế mô hình thường có nhiều lựa chọn khi chỉ rõ điểm chèn vào danh sách quản lý người sử dụng. Những sự lựa chọn này là tùy thuộc vào sự thực hiện. Phương thức thực hiện có thể bao gồm việc chèn một thực thể vào đáy của một danh sách, vào đỉnh của một danh sách, hoặc vào trong danh sách được xếp hạng theo xu hướng tăng dần hoặc giảm dần dựa vào thuộc tính của thực thể theo đặc trưng người sử dụng.

Những thực thể trong danh sách quản lý người sử dụng không thể tự di chuyển ra khỏi danh sách, và phần mềm mô phỏng cũng sẽ không tự động di chuyển các thực thể ra khỏi danh sách. Việc di chuyển thực thể từ một danh sách quản lý người sử dụng xảy ra khi thực thể khác nào đó làm nó xảy ra. Dựa trên một phép thử cung cấp bởi người thực hiện mô hình hóa, thực thể khác này quyết định xem liệu có nên chuyển một hoặc nhiều thực thế từ một danh sách quản lý người sử dụng nào đó tới trạng trái sẵn sang trong danh sách sự kiện hiện thời hay không.

Người thực hiện mô hình hóa linh hoạt trong việc lựa chọn một hay nhiều thực thể nào để di chuyển khỏi danh sách quản lý người sử dụng. Những lựa chọn đơn giản nhất là di chuyển các thực thể ra khỏi đỉnh hoặc đáy danh sách. Sự lựa chọn phức tạp hơn là quét một lượt danh sách trên xuống dưới, lặp lại việc đánh giá một biểu thức logic được người dùng định nghĩa, đánh giá từng thực thể một, sự di chuyển một hay nhiều thực thể ứng với giá trị "true" của biểu thức logic. Giá trị của biểu thức logic sẽ phụ thuộc vào một hay nhiều thuộc tính của thực thể hiện đang được đánh giá. Những sự lựa chọn mà của người thực hiện mô hình hóa dung để có các quyết định di chuyển thực thể đang trong trạng thái không hoạt động, phụ thuộc vào sự thực hiện phần mềm mô phỏng.

3.6.7 Tổng kết về các danh sách được sử dụng để quản lý thực thể

Những trạng thái thực thể khác nhau và những tên chung được đưa ra các danh sách được sử dụng để quản lý các thực thể trong những trạng thái đó được tổng kết trong bảng 3.2

Bảng 3.2 Các trạng thái thực thể và danh sách thực thể đi kèm

Trạng thái thực thể	Tên chung của danh sách thực thể	Chú thích
Hoạt động	None	Tối đa là một thực thể hoạt động
Sẵn sàng	Danh sách sự kiện hiện thời	Chỉ có một danh sách
Trễ thời gian	Danh sách sự kiện tương lai	Chỉ có một danh sách
Trễ có điều kiện	Danh sách trễ	Có nhiều danh sách
Không hoạt động	Danh sách quản lý người dùng	Có nhiều danh sách

3.7 Triển khai với 3 trường hợp phần mềm mô phỏng.

Sau đây, chúng tôi sẽ giới thiệu ba phần mềm mô phỏng sự kiện rời rạc gồm có SIMAN [Bank95b], ProModel [Benson] và GPSS/H [Bank95a]. Trong đó SIMAN và GPSS/H được dùng chung cho nhiều trường hợp còn ProModel là ứng dụng cho sản xuất.

Table 3.3 Thuật ngữ dùng trong SIMAN

Các cụm từ hay thuật ngữ chung	Thuật ngữ của SIMAN
Thực thể ngoại (External entity) Thực thể nội (Internal entity) Tài nguyên (Resource)	Thực thể (Entity) Ko có thuật ngữ tương ứng Tài nguyên, khối vận chuyển(Resource,

	Conveyor, Transporter)
Thành phần điều khiển	Khối bao vây (blockage)
Hoạt động	Khối hay các khối (Block or Blocks)
Danh sách các sự kiện hiện thời (Curent Events Chain)	Chuỗi các sự kiện hiện thời (Curent Events Chain)
Danh sách các sự kiện tương lai(Future Events list)	Cây sự kiện tương lai (Future Events Heap)
Danh sách trễ (Delay list)	Hàng đợi đính, Hàng đợi nội
Danh sách do người dùng được quản lý	Hàng đợi tách
Giai đoạn thực thể di chuyển (entity movement giai đoạnse)	Không có từ/cụm từ tương ứng
Giai đoạn cập nhật xung đồng hồ (clock update phase)	Không có từ/cụm từ tương ứng

3.7.1 SIMAN

Các phần sau đây bao gồm (1) việc tóm tắt thuật ngữ của phần mềm SIMAN, (2) các chi tiết của giai đoạn thực thể di chuyển và giai đoạn cập nhật đồng hồ của SIMAN và sự tương tác của chúng với sự hoạt động tương ứng của các danh sách các sự kiện hiện thời và tương lai, và (3) danh sách trễ và danh sách do người dùng quản lý của SIMAN

Thuật ngữ dùng trong SIMAN.

Các thuật ngữ SIMAN tương đương với các thuật ngữ chung được đưa ra ở bảng 3.3.

Các thực thể, tài nguyên, các thành phần điều khiển và các hoạt động.

Trong SIMAN, các thực thể ngoại được gọi là các thực thể. Cũng giống như sự trình bày tổng quát về các thực thể, các thực thể SIMAN được dùng để mô hình hóa đối tượng ở thể động hay tĩnh dang di chuyển trên hệ thống và gây ra sự thay đổi trạng thái của hệ thống. Các thực thể có thể có các thuộc tính (ví dụ như một sản phẩm đang trong quá trình sản xuất thời gian tới, thời gian cần hoàn thành, số nhận dạng khách hàng v...v). Nhiều loại thực thể ngoại không được định nghĩa chính thức trong SIMAN.

SIMAN dùng các thực thể nội để quản lý sự bắt đầu và sự kết thúc của thời gian tài nguyên không sẵn sàng cho sử dụng (máy móc hỏng, công nhân nghỉ việc ---downtime) và cho sự chấm dứt việc chạy mô phỏng khi thời gian mô phỏng ấn định đã hết. Các thực thể nội này không có tên gọi cụ thể trong SIMAN. (Các thực thể nội có được từ các thành phần được xác định trong tập tin thực hành SIMAN bởi người thực hiện mô hình hóa). Các tài nguyên của SIMAN mô hình hóa các đối tượng thực hiện cung cấp dịch vụ với tư cách các thực thể. Các băng tải (Conveyor) và Máy chuyển hàng (Transporter) là các tài nguyên dùng cho các mục đích đặc biệt tương ứng với việc mô hình hóa sự di chuyển của các đối tượng trên các đường dẫn cố định và đường dẫn tự do trong hệ thống.

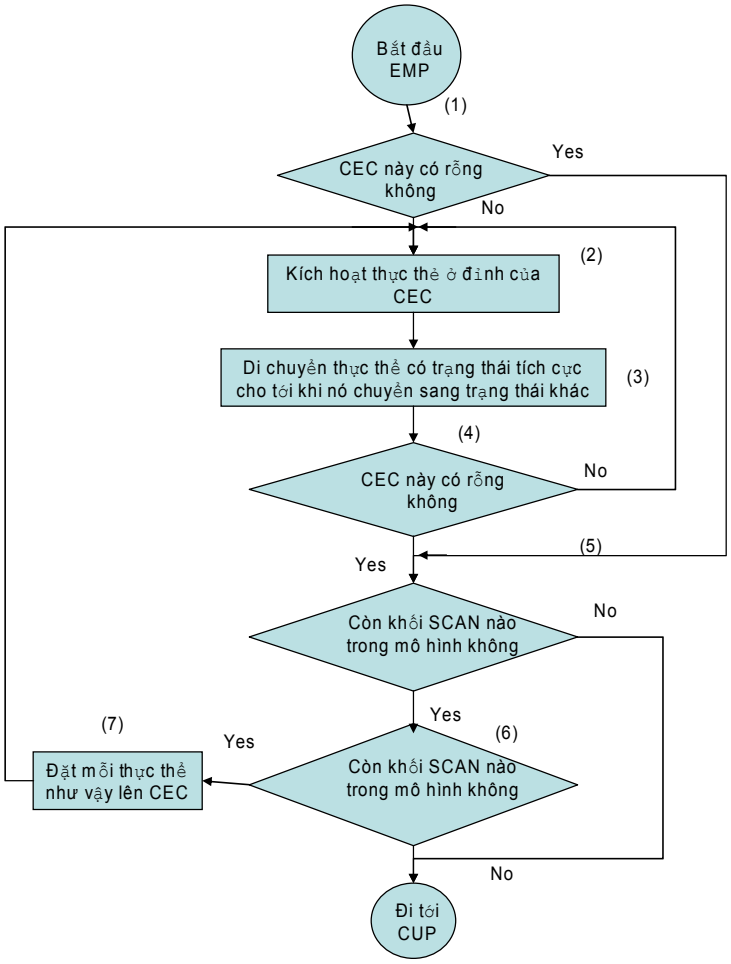
Trong SIMAN, các khối (Blocks) được dùng để mô tả hoạt động được thực hiện bởi hoặc trên các thực thể. Các khối được sắp thành chuỗi theo thứ tự các hoạt đông và được kết nối bằng các đường dẫn. Các thực thể di chuyển dọc theo những đường dẫn này từ khối này sang khối khác, kích hoạt sự hoạt động của khối khi chung di chuyển qua. Mỗi loại khối có một từ khóa và các phép tính (toán hạng --- operand) mà các giá trị của chúng cụ thể hóa từng trường hợp các khối.

Ví dụ khối CREATE được dùng để tạo các thực thể trong đó phân bố thời gian giữa hai lần tạo ra thực thể (cùng với biến ngẫu nhiên tương ứng) được quy đinh ở một trong số các phép tính của nó. Khối WAIT được dùng để đặt một thực thể vào trạng thái trễ; nó có một toán hạng để mô tả phân bố của thời gian trễ. Khối SIEZE được dùng bởi thực thể để yêu cầu 1 hay nhiều đơn vị tài nguyên; nó có các toán hạng để nhận dạng từng tài nguyên cụ thể và số các đơn vị tài nguyên yêu cầu v…v. SIMAN cung cấp hơn 40 loại khối.

Chuỗi sự kiện hiện thời (CEC)
Chuỗi sự kiện hiện thời của SIMAN có các đặc điểm của danh sách sự kiện hiện thời được nêu tổng quát trước kia. Các thực thể ở trạng thái sẵn sàng được đưa vào chuỗi này tại giai đoạn cập nhật xung đồng hồ, bởi việc sao chép trong giai đoạn thực thể hoạt động và bởi sự thực hiện thời gian đợi liên kết (related waiting) và thời gian đợi thăm dò (polled waiting) trong giai đoạn thực thể hoạt động. Khi một giai đoạn thực thể hoạt động kết thúc, chuỗi này rỗng.

Giai đoạn thực thể hoạt động. (EMP)
SIMAN không có các tên đặc biệt cho giai đoạn hoạt động thực thể. Hoạt động của một EMP được diễn tả trong sơ đồ ở hình 3.11.
Khi bắt đầu EMP trong SIMAN, chương trình kiểm tra xem liệu chuỗi sự kiện hiện thời có bao gồm các thực thể ở trạng thái sẵn sàng hay là một chuỗi rỗng (ô số 1 trong hình 3.11). Thường thì sẽ có một hay nhiều thực thể ở trạng thái sẵn sàng nằm trong CEC khi một EMP bắt đầu. Nhưng CEC có thể rỗng bởi vì có các thực thể đang trong trạng thái trễ sử dụng phương thức thời gian chờ thăm dò (polling wating).
Giả thiết một CEC ban đầu không rỗng. Sau khi thực hiện bước kiểm tra (ô số 1), SIMAN đặt thực thể ở vị trí đầu tiên của CEC vào trạng thái hoạt động (ô số 2) và sau đó đưa thực thể này qua nhiều khối/hoạt động (blocks/operations) nhất có thể được cho tới khi thực thể chuyển từ trạng thái tích cực sang trạng thái trễ thời gian (được chuyển tới FEC) hay trạng thái trễ có điều kiện (được chuyển tới một *Hàng đợi nội* – Internal Queue hay H*àng đợi đính* Attached Queue) hoặc thanh trạng thái không hoạt động (được chuyển tới *Hàng đợi tách* – Detached Queue).
Sau khi một thực thể đã chuyển khỏi trạng thái hoạt động, tích cực, CEC được kiểm tra (ô số 4, hình 3.11) để xem liệu nó đã rỗng chưa. Nếu CEC vẫn chưa rỗng, và giai đoạn thực thể hoạt động thực hiện vòng lặp quay về ô số 2. Quá trình lặp sẽ tiếp tục đến khi CEC rỗng.
Thực thể hoạt động này có thể thực hiện một khối để tạo ra một hoặc vài bản sao của chính nó. Trong trường hợp này, các bản sao được đặt ngay vào trạng thái sẵn sàng ở vị trí đầu tiên trên CEC (trên bất kì trạng thái CEC nào khác) với thứ tự FIFO. Ví dụ như, nếu thực thể hoạt động tạo ra một bản sao thì bản sao này sẽ được đặt vào vị trí đầu trên CEC và sẽ là thực thể sẵn sàng tiếp theo để đưa vào trạng thái hoạt động.

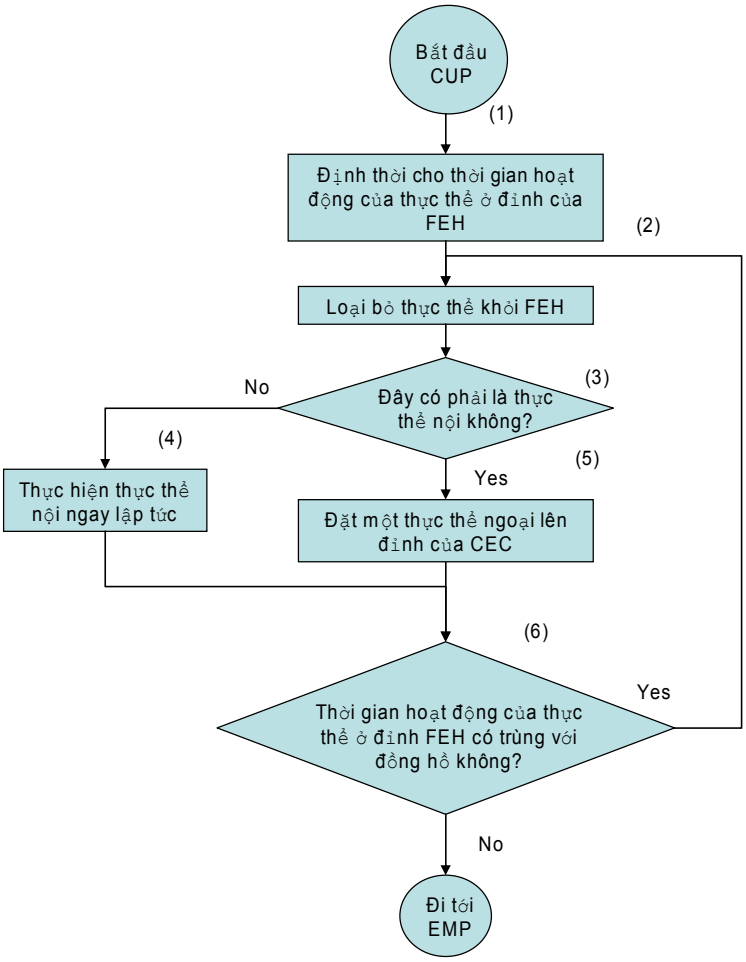
Sau khi CEC rỗng (bởi vòng lặp các ô số 2-3-4), EMP kiểm tra xem có khối SCAN nào trong mô hình này hay không (ô số 5). Các khối SCAN được dùng để thực hiện hiện cơ chế thời gian chờ thăm dò (polled waiting). Khi một khối SCAN được thực hiện, một danh sách trễ tương ứng kiểm tra để xem điều kiện gây ra trễ đã được giải quyết cho thực thể đứng ở vị trí đầu danh sách trễ này hay chưa (ô số 6). Nếu điều kiện trễ đã giải quyết xong, thực thể này chuyển sang trạng thái sẵn sàng ở CEC(ô số 7). Sau khi mỗi khối SCAN được thực hiện, EMP tiếp tục xử lý các thực thể ở trạng thái sẵn sàng xử lí (hộp 2,3,4).
Giai đoạn thực thể hoạt động tiếp tục hoạt động như vậy cho tới khi CEC rỗng và không còn thực thể nào được chuyển sáng trạng thái sẵn sàng bằng cơ chế thời gian chờ thăm dò nữa. Mô hình SIMAN sau đó được cập nhật hoàn toàn tại thời điểm mô phỏng hiện tại. Bước tiếp theo là thực hiện giai đoạn cập nhật đồng hồ tiếp theo.



Hình 3.11 Giai đoạn thực thể hoạt động trong SIMAN

Cây sự kiện tương lai (FEH).
Trong SIMAN, các thực thể trễ thời gian (bao gồm cả thực thể nội và thực thể ngoại) được lưu giữ trong Cây sự kiện tương lai (FEH). Cấu trúc này hoạt động như một danh sách được sắp xếp theo thời gian di chuyển tăng dần. Thực thể có thời gian di chuyển nhỏ nhất là thực thể tiếp theo được đưa ra khỏi FEH khi giai đoạn cập nhật đồng hồ xảy ra. Nếu như có sự trùng hợp thời gian di chuyển, thì thứ tự các thực thể được đưa ra khỏi FEH không nhất thiết phải trùng với thứ tự khi đưa đặt chúng vào cây sự kiện này.

Giai đoạn cập nhật đồng hồ (CUP)
Giai đoạn cập nhật đồng hồ tăng đồng hồ mô phỏng tới giá trị thời gian di chuyển của thực thể hiện đang ở vị trí đỉnh của FEH. Điều gì sẽ diễn a tiếp theo tùy thuộc vào thực thể đó là thực thể nội hay thực thể ngoại. Hình 3.12 trình bày sơ đồ hoạt động của giai đoạn cập nhật đồng hồ dung trong SIMAN.



Hình 3.12 . Giai đoạn cập nhật đồng hồ

Sau khi đông hồ mô phỏng được thiết lập bằng với thời gian di chuyển của thực thể hiện đang ở vị trí đỉnh của FEH (ô số 1, hình 3.12), thực thể được đưa ra khỏi FEH (ô số 2) và được kiểm tra xem liệu nó là thực thể ngoại hay thực thể nội. Nếu là thực thể nội, nó sẽ được xử lý ngay lập tức (ô số 4). Nếu là thực thể ngoại, nó được chèn vào vị trí đầu (vào sau ra trước) của chuỗi sự kiện hiện thời (ô số 5).

Việc đưa thực thể ra khỏi FEH có nghĩa là một thực thể nào đó hiện giờ ở vị trí đỉnh của FEH. Để kiểm tra sự trùng lặp thời gian có thể xảy ra, bước tiếp theo (ô số 6) kiểm tra xem liệu thời gian di chuyển của thực thể đứng đầu FEH có trùng với thời gian mô phỏng hiện tại không. Nếu có, thực thể đó được giải quyết theo cách tương ứng (các hộp 2 và 3, 4, 5). Quá trình đó lặp lại tới khi tất cả các thực thể có cùng thời gian di chuyển được đưa ra khỏi FEH. Sau đó SIMAN thực hiên giai đoạn thực thể di chuyển để cập nhật trạng thái của mô hình tại thời điểm mô phỏng mới được thiết lập này.

Hàng Đợi Đính và Hàng Đợi Nội

Đây là hai loại danh sách của SIMAN tạo bởi các thực thể ở trạng thái trễ có điều kiện. Các danh sách này có khả năng được định dạng với các khối như các khối Hold (Hodl blocks). Ví dụ như khối SIEZE được dùng bởi một thực thể để chiếm giữ tài nguyên, là một khối Hold. Nếu một thực thể hoạt động cố thực hiện một khối SEIZE và nhận ra rằng nó phải đợi tới lượt mình để dùng tài nguyên được yêu cầu, nó sẽ được chuyển từ CEC sang trạng thái trễ có điều kiện nằm ở hàng đợi nội hoặc hàng đợi đính và nó sẽ chờ điều kiện Giữ được giải quyết.

Nếu một khối QUEUE có trước một khối Hold, hàng đợi đính được sử dụng. Hàng đợi đính là danh sách của các thực thể đang đợi để thực hiện khối Hold tương ứng. Các thực thể được dưa vào hàng đợi đính theo FIFO hoặc LIFO, hoặc sắp xếp theo giá trị của biểu thức số học mà người thiết kế mô

hình tạo ra. Nếu không có khối QUEUE nào đến trước khối Hold, SIMAN dùng một hàng đợi nội cho khối Hold đó. Một hàng đợi nội là hàng đợi chưa được đặt tên được thực hiện theo kiểu FIFO.
Đôi khi, sẽ thuận lợi khi sử dụng các khối Hold cùng loại tại nhiều vị trí trong một mô hình (VD: dùng khối "SEIZE DRILL" ở hai hoặc nhiều nơi trong mô hình). Người thiết kế mô hình có thể gắn kết một hàng đợi đính riêng rẽ với mỗi khối Hold như thế. Đây là kiểu hàng đợi đính không chia sẻ, do các thực thể ở hai hay nhiều hàng đợi đang chờ ở các danh sách trễ riêng rẽ đối với cùngmột Tài nguyên. (Mức ưu tiên của khối Hold được dùng để quyết định hàng đợi đính nào sẽ hỗ trợ thực thể thể tiếp theo chiếm lấy Tài nguyên). Người thiết kế mô hình cũng có thể đặt các thực thể đã trễ tại hai hoặc nhiều khối Hold giống nhau ở cùng một hàng đợi đính, được gọi là hàng đợi đính dùng chung. Thực thể ở vị trí cao nhất trong hàng đợi này sẽ chiếm lấy Tài nguyên trước.

Hàng Đợi Tách.

Là các danh sách thực thểđược SIMAN dùng để thực hiện trạng thái không hoạt động. Các thực thể được đặt trong hàng đợi tách khi chúng thực hiện các khối QUEUE với chức năng DETACHED đi kèm. Các thực thể như thế sau đó được chuyển từ trạng thái không hoạt động sang trạng thái sẵn sang bởi tác động của các thực thể khác. (Các thực thể khác dùng các khối như SEARCH và REMOVE, hoặc QPICK và MATCH để chuyển các thực thể từ hàng đợi tách sang CEC.).

3.7.2 ProModel

Trong phần tiếp theo này, thuật ngữ ProModel sẽ được tổng hợp cùng với sự mô tả các chi tiết của giai đoạn thực thể di chuyển và giai đoạn cập nhật đồng hồ và thảo luận về cácht hức xử lý các thực thể trễ trong ProModel. Thuật ngữ của ProModel: Bảng 3.4 so sánh các thuật ngữ của ProModel với các thuật ngữ chung. Nội dung chi tiết về các thuật ngữ được cung cấp sau.
Các Thực Thể: Thực thể ngoại trong ProModel được gọi là Thực Thể. Các lớp thực thể khác nhau phải được định nghĩa khi mô hình hóa trong ProModel. Các lớp thực thể này được gọi là Loại Thực Thể. Các loại thức thể được đựt tên riêng bởi người thiết kế mô hình. ProModel sử dụng các thực thể nội gọi là Các Sự Kiện Nội và hoạt động của chúng được thảo luận dưới đây.
Bảng 3.4. Thuật ngữ ProModel

Cú pháp và thuật ngữ chung	ProModel
Thực thể ngoại	Thực thể
Thực thể nội	Sự kiện nội
Tài nguyên	Tài nguyên, vùng, node
Thành phần điều khiển	Biến
Hoạt động điều hành	Bước xử lý
Danh sách sự kiện hiện thời	Danh sách hoạt động
Danh sách sự kiện tương lai	Danh sách sự kiện tương lai
Danh sách trễ	Danh sách đợi
Danh sách quản lý người dùng	Không có thuật ngữ tương đương
Giai đoạn chuyển thực thể	Không có cú pháp
Giai đoạn cập nhật đồng hồ	Không có cú pháp

Tài Nguyên. Trong ProModel có 3 khái niệm tương ứng với khái niệm chung Tài Nguyên: Vị Trí (Location), Tài Nguyên (Resource) và Nút (Node). Như tên được gọi, Vị Trí (Location) tương ứng với nơi chốn cố định. Chúng được sử dụng để mô hình hóa các thức như các khu vực chờ, các trạm làm việc (workstation), các máy móc cố định. Locations cũng được sử dụng để mô hình hóa băng tải tích lũy (accumulating) và băng tải không tích lũy (nonaccumulating). Tài Nguyên (Resource) có thể được sử dụng để mô hình các vấn đề về con người và các thiết bị. Tài Nguyên có thể là tĩnh, ví dụ như dùng mô hình hóa một công nhân đứng yên điều khiển một máy ở vị trí cố định. Tài Nguyên

có thể là động, ví dụ như dùng để mô hình hóa các xe nâng hàng hoặc các công nhân là những người có thể di chuyển từ vị trí này tới vị trí khác trong một hệ thống.
Tài nguyên động (Dynamic Resource) di chuyển qua lại trong mạng lưới được tạo bởi các đường dẫn (mạng lưới đường dẫn - path network). Có một vài kiểu mạng lưới đường dẫn. Ví dụ như có một mạng lưới đường dẫn trong đó các tài nguyên động phải di chuyển theo kiểu hang đơn. Và cũng có mạng lưới đường dẫn khác trong đó các tài nguyên động có thể vượt qua nhau. Các nút (node) là khái niệm tài nguyên khác trong ProModel. Các nút là các điểm bắt đầu và điểm kết thúc của các đường dẫn tạo nên các mạng lưới mà các tài nguyên động di chuyển bên trong đó. Các tài nguyên cạnh tranh lẫn nhau để sử dụng các nút và di chuyển trong mạng trong khi tìm kiếm một cái gì đó ví dụ như để thu hồi lấy hoặc một vị trí trống.
Bởi vì các tài nguyên động di chuyển, thời điểm chúng di chuyển có ý nghĩa quyết định. Bởi vì chúng cạnh tranh để dành lấy các nút nên sự trễ có thể có cũng có ý nghĩa quyết định đối với chúng. Kết quả là, các tài nguyên động có vẻ giống với các thực thể (Trong khi các thực thể là các đơn vị lưu lượng di chuyển xuyên qua hệ thống và là các người dung dịch vụ, thì các tài nguyên động là các đơn vị của lưu lượng di chuyển bên trong hệ thống và là các nhà cung cấp dịch vụ). Giống như các thực thể, các tài nguyên động có thể được đặt vào trạng thái trễ thời gian trong khi chúng đợi một khoảng thời gian tương lai đã biết trước khi mà chúng sẽ đến được đích của chúng. Chúng cũng có thể chuyển từ trạng thái trễ thời gian sang trạng thái sẵn sàng, và sau đó vào trạng thái hoạt động. Do có sự cạnh tranh dành lấy các nút, chúng cũng có thể được coi là thích hợp đối với trạng thái trễ có điều kiện. ProMode sử dụng danh sách sự kiện tương lai (Future Events List), danh sách hoạt động (Action List), và danh sách chờ (Waiting Lists) không chỉ để quản lý các thực thể mà còn quản lý các tài nguyên động. Các danh sách sự kiện tương lai và danh sách hoạt động cũng được sử dụng để quản lý các sự kiện nội (Internal Events).

Các phần tử điều khiển (Control Elements). Trong ProModel, một Biến (Variable) là một phần tử dữ liệu dùng cho nhiều mục đích mà giá trị của nó có thể là một đối tượng của một ProModel WAIT UNTIL. Giống như tất cả các phần tử điều khiển khác, một Biến (sử dụng trong việc kết hợp với một WAIT UNTIL) có khả năng gây ra trễ cho các thực thể.
Các hoạt động (Operations) Phần luồng di chuyển (transaction-flow) của ProModel là được tạo ra bởi người thiết kế mô hình, như là một tập hợp có thứ tự của các Bước Chu Trình (Process Steps) nằm trong một Bảng Chu Trình (Process Table). Mỗi một bước chu trình định ra tên của của một hoặc tất cả các loại thực thể (Entity Type) và tên của một hoặc tất cả vị trí (Location). Một thực thể di chuyển từ mọt bước chu trình tới bước khác trong một bảng chu trình bằng cách nhảy đến bước chu trình tiếp theo phù hợp với loại (Type) và vị trí (Location) của nó (lặp lại việc quay lại đỉnh của bảng chu trình nếu cần thiết). Các bước chu trình này xác định rõ cái mà loại thực thể (Entity Type) này phải làm tại vị trí (Location) này.
Logic hoạt động (Operation Logic) và logic định tuyến (Routing Logic) là các thành phần tạo nên các bước chu trình (Process Step). Một bước chu trình có thể bao gồm không (zero) hoặc nhiều hơn các thành phần đó. Sự cạnh tranh giữa các thực thể đối với các tài nguyên không có khả năng chuyển tải thực hiện bởi logic hoạt động. Sự cạnh tranh giữa các thực thể đối với các vị trí (Location) và các tài nguyên có khả năng chuyển tải được giải thích rõ ràng trong logic định tuyến . Logic định tuyến được ứng dụng sau khi logic hoạt động đã được thực hiện.
Nhiều khái niệm định nghĩa mô hình trong ProModel có các tùy chọn theo bởi người sử dụng ví dụ như logic thời gian dừng hoạt động (Downtime Logic) và logic thoát kỏi vị trí (Location Exit Logic). Logic là tập hợp của các câu lênh để thực hiện một cách tự động và sự thực hiện của chúng có khả năng đặt các sự kiện nội vào danh sách sự kiện tương lai. Khi được xử lý, các sự kiện nội này có thể quay lại danh sách sự kiện tương lai hoặc quay lại danh sách trễ. Chúng cũng có thể làm cho các thực thể hoặc các tài nguyên động được đưa vào danh sách sự kiện hiện thời.

Danh sách hoạt động (Action List). Danh sách hoạt động là tên gọi dùng trong ProModel đối với danh sách các sự kiện hiện thời. Danh sách hoạt động có thể bao gồm các thực thể, tài nguyên động và các sự kiện nội (Thuật ngữ phần tử - *element* sẽ được sử dụng kể từ đây dung để gọi một thực thể,

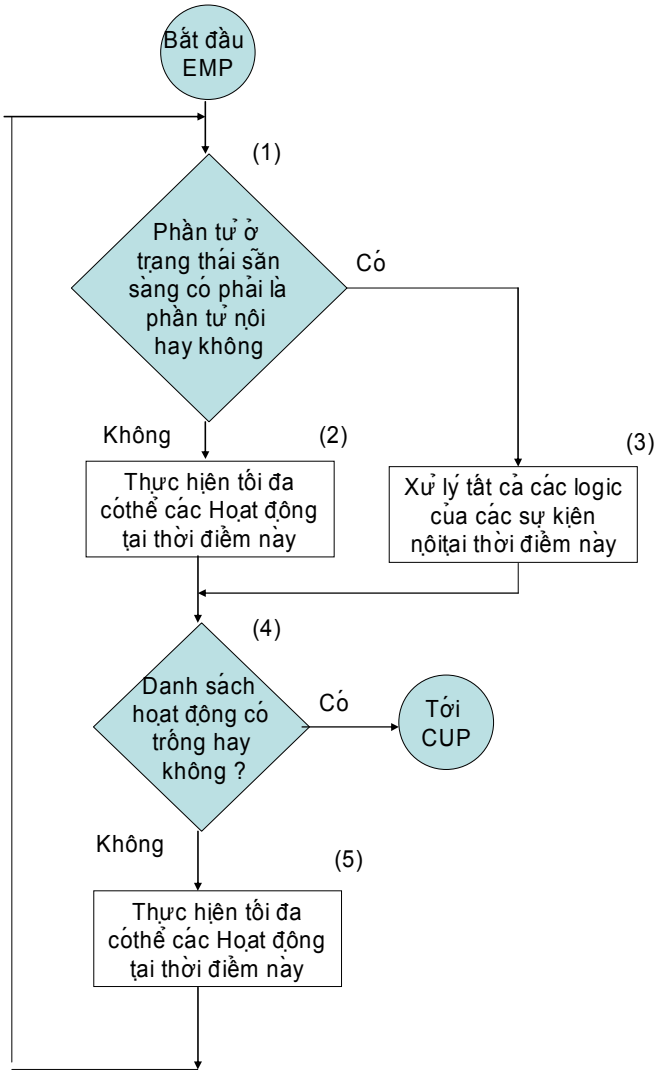
một tài nguyên động hoặc một thực thể nội bất kỳ). Danh sách hoạt động trống rỗng khi một giai đoạn thực thể hoạt động bắt đầu. Nghĩa là không có phần tử nào ở trong trạng thái sẵn sàng. Tuy nhiên, một thực thể, một tài nguyên động, hoặc một sự kiên nội đã ở trong trạng thái hoạt động tại thời điểm này. Các thực thể, các tài nguyên động, và các thực thể nội có thể được đặt vào danh sách hoạt động khi khi phần tử ở trạng thái hoạt động này bắt đầu được di chuyển. Bất kỳ một phần tử mới nào của danh sách này đều được đặt vào đầu danh sách. Khi phần tử hoạt động chuyển từ trạng thái hoạt động, phần tử tại đầu danh sách hoạt động sẽ chuyển sang trạng thái hoạt động. Đây là kết quả trong cách quản lý danh sách hoạt động theo kiểu LIFO (Last-in, First Out).

Giai đoạn thực thể hoạt động (Entity Movenment Phase). Giai đoạn thực thể hoạt động của ProModel được chỉ ra trong hình 3.13. Một phần tử ở trong trạng thái hoạt động khi giai đoạn thực thể hoạt động bắt đầu (được giải thích trong phần giai đoạn cập nhật đồng hồ sau đây), và danh sách hoạt động hiện đang trống. Nếu phần tử hoạt động là một thực thể hoặc một tài nguyên động (ô 1, hình 3.13), nó thực hiện các hoạt động cho đến khi nó bị ép chuyển ra khỏi trạng thái hoạt động (ô số 2). rường hợp khác, nó là một sự kiện nội và tất cả logic tương ứng được thực hiên trên nó (ô số 3)
Theo cách khác, một hoặc hơn một phần từ mới ở trạng thái sẵn sàng có thể được tạo ra và như vậy sẽ được đưa vào danh sách hoạt động theo kiểu LIFO (ví dụ như một thực thể có thể tạo ra một hoặc nhiều hơn bản sao của chính nó). Trong trương hợp này (lựa chọn "no" từ ô số 4, hình 3.13), phần tử hiện tại ở đỉnh của danh sách hoạt động được biến thành phần tử hoạt động (ô số 5) và giai đoạn thực thể hoạt động được tiếp tục. Nếu không thì danh sách hoạt động này trở nên trống và một giai đoạn cập nhật đồng hồ thực hiện tiếp theo (lựa chọn "yes", ô số 4).

Danh sách sự kiên tương lai (Future Events List). Danh sách sự kiện tương lai của ProMode tạo bở các phần tử như thực thể, tài nguyên động, sự kiện nội hiện đang trong trạng thái trễ và chờ. Các phần tử này được sắp xếp trong danh sách từ trên xuống dưới theo thứ tự tăng dần của thời gian di chuyển. ProModel luôn luôn di chuyển duy nhất một phần tử từ đỉnh của danh sách trong mỗi giai đoạn cập nhật đồng hồ (điều này có nghĩa rằng trong trường hợp các phần tử có trùng thời gian di chuyển, sẽ có hai hoặc nhiều hơn giai đoạn cập nhật đồng hộ và giai đoạn thực thể hoạt động tại cùng thời điểm mô phỏng).

Giai đoạn cập nhật đồng hồ (Clock Update Phase). Giai đoạn cập nhật đồng hồ của ProModel được tổng hợp trong hình 3.14. Giai đoạn này bắt đầu bằng cách thiết lập đồng hồ mô phỏng bằng với thời gian di chuyển của phần tử tại đỉnh của danh sách sự kiện tương lai (ô số 1). Phần tử này được đặt ngay lập tức vào trạng thái hoạt động (ô số 2) và chấm dứt giai đoạn cập nhật đồng hồ và bắt đầu giai đoạn di chuyển thực thể. Trong ProModel, không có các thực thể ở trạng thái sẵn sàng khi một giai đoạn thực thể hoạt động bắt đầu. (danh sách hoạt động hiện đang trống). Hơn nữa, nếu có 2 hoặc nhiều hơn các phần tử tại đỉnh của danh sách sự kiện tương lai có cùng thời gian di chuyển, 2 hoặc nhiều hơn giai đoạn cập nhật đồng hồ và giai đoạn thực thể hoạt động liên tiếp sẽ được thực hiện tại cùng thời điểm mô phỏng.

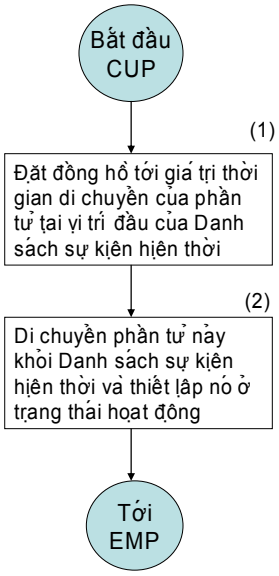
Danh sách chờ (Waiting Lists). Danh sách chờ là các danh sách trễ của ProModel dành cho các thực thể và các tài nguyên động. Một danh sách chờ cho các thực thể được gắn với mỗi vị trí, mỗi tài nguyên tĩnh, mỗi tài nguyên động, và mỗi Biến tùy theo một WAIT UNTIL. Một dạm sách chờ dành cho các tài nguyên động được gắn với mỗi Nút.
Một thực thể đơn (hoặc tài nguyên động) có thể xuất hiện đồng thời trong nhiều danh sách trễ. Hệ quả là ProModel không sử dụng phương pháp thời gian chờ thăm dò để giải quyết các trễ phức hợp.



Hình 3.13 Giai đoạn thực thể hoạt động

Thay vào đó, phương pháp thời gian chờ liên kết được sử dụng để quản lý các danh sách chờ cùng với việc ProModel di chuyển thực thể này hoặc tài nguyên động này khỏi tất các các danh sách trễ lien quan sớm ngay khi co chế thời gian chờ liên kết chuyển giao thực thể này hoặc tài nguyên động này từ một trong các danh sách đó tới danh sách hoạt động.

ProModel không có các danh sách tương đương với danh sách do người dùng quản lý. Tuy nhiên , JOIN, LOAD và SEND là các logic định tuyến tùy chọn sẽ đặt các thực thể vào các danh sách vị trí riêng biệt. Tại đó chúng chờ một câu lệnh hoạt động JOIN, LOAD hoặc SEND tương ứng được thực hiện bởi một thực thể khác tại vị trí đích này. Bởi vì các danh sách này là đặc trưng cho vị trí nên chúng được coi như thuộc về các danh sách trễ.



Hình 3.14 Giai đoạn cập nhật xung đồng hồ ProModel

3.7.3 GPSS/H

Trong những mục tiếp theo, chúng ta sẽ tóm lược về thuật ngữ GPSS/H, mô tả các chi tiết hoạt động của giai đoạn thực thể hoạt động và giai đoạn cập nhật đồng hồ trong GPSS/H và thảo luận về các danh sách do người dùng quản lý.

Thuật ngữ GPSS/H: Bảng 3.5 bao gồm các thuật ngữ GPSS/H tương đương với nhiều thuật ngữ chung khác mà đã được trình bày ở phần trước.

Bảng 3.5 – Thuật ngữ GPSS/H

Các cụm từ và thuật ngữ chung	**Thuật ngữ tương đương trong GPSS/H**
Thực thể ngoại	Lưu lượng (transaction)
Thực thể nội	Lưu lượng hệ thống
Nguồn tài nguyên	Phương tiện, Nguồn lưu trữ
Phần tử điều khiển	Chuyển mạch Logic, Biểu thức số học, Biểu thức Boolean
Hoạt động	Khối
Danh sách sự kiện hiện thời	Chuỗi sự kiện hiện thời
Danh sách sự kiện tương lai	Chuỗi sự kiện tương lai
Danh sách trễ	Chuỗi sự kiện hiện thời
Danh sách người dùng được quản lí	Chuỗi người sử dụng
Gian đoạn thực thể hoạt động	Giai đoạn quét
Gian đoạn cập nhật đồng hồ	Gian đoạn cập nhật đồng hồ

Các thực thể, Tài nguyên, Phần tử điều khiển và Hoạt động:

Trong GPSS/H, các thực thể ngoại được gọi là Lưu Lượng (gọi tắt là Xacts). Không có định nghĩa chính thức về các lớp lưu lượng. Các lưu lượng sở hữu những thuộc tính có giá trị là các số. Chức năng của các thuộc tính này do người thiết kế mô hình quyết định. Người ta sử dụng một thuộc tính lưu lượng đặc biệt gọi là Mức Ưu Tiên (Priority Level) để quyết định thứ tự của một lưu lượng khi đưa nó vào danh sách các sự kiện hiện thời.

GPSS/H sử dụng một số các thực thể nội gọi là Lưu lượng hệ thống (System Transactions). Các Lưu lượng hệ thống có nhiệm vụ thiết lập các quá trình đến (arrival processes), hỗ trợ một số loại quan sát thống kê cũng như các thuật toán xử lý danh sách được dùng để quản lý danh sách sự kiện hiện thời và danh sách sự kiện tương lai. Trong một số trường hợp, GPSS/H sử dụng danh sách sự kiện tương lai để quản lý việc xác định thời gian của các lưu lượng hệ thống. GPSS/H không sử dụng các thực thể nội để mô hình hóa thời gian ngừng hoạt động (downtime) của tài nguyên hay

thời gian dừng mô phỏng. Thay vì đó, các thực thể ngoại (Transactions) sẽ làm điều này. Các thực thể ngoại này sẽ thực thi các khối (Blocks) (được thảo luận dưới đây) để thực hiện logic của thời gian ngừng hoạt động và thời gian dừng mô phỏng.

Các phương tiện và nguồn lưu trữ dùng để mô hình hóa các tài nguyên. Các phương tiện (Facility) mô hình hóa các tài nguyên có dung lượng là một đơn vị. Nghĩa là một phương tiện là một tài nguyên mà chỉ có thể phục vụ một lưu lượng tại một thời điểm.
Trong khi đó, **nguồn lưu trữ (Storages**) mô hình hóa các tài nguyên mà có thể có dung lượng bất kỳ tùy ý người dùng. Nghĩa là nguồn lưu trữ (Storages) có khả năng phục vụ đồng thời nhiều lưu lượng.

Trong GPSS/H, chuyển mạch **Logic** là các biến on/off được dùng để điều khiển. Các lưu lượng có thể bị buộc phải chờ cho đến khi một chuyển mạch Logic định trước nằm trạng thái được chỉ định. Các lưu lượng có thể di chuyển tuần tự hoặc không tuần tự. Điều này phụ thuộc vào việc chuyển mạch Logic định trước có ở trạng thái được chỉ định hay không.

Biểu thức số học và biểu thức Boolean cũng có thể được sử dụng như là các phần tử điều khiển trong GPSS/H. Các lưu lượng có thể bị buộc phải chờ cho đến khi một biểu thức số học định trước thỏa mãn các điều kiện có liên quan định trước (hoặc cho đến khi biểu thức Boolean có giá trị thực định trước). Hoặc các lưu lượng có thể đi theo đường dẫn không tuần tự tùy thuộc vào giá trị của do một biểu thức số học hoặc một biểu thức Boolean.

Trong GPSS/H, **khối (blocks)** được sử dụng để mô tả các hoạt động do các lưu lượng tiến hành hoặc thực thi trên chúng. Các Block được sắp xếp tuần tự theo trật tự các hoạt động và liên kết với nhau bằng đường dẫn. Các lưu lượng di chuyển dọc theo đường dẫn từ khối này đến khối và kích hoạt sự hoạt động của khối khi chúng di chuyển. Mỗi loại khối có một từ khóa và toán hạng trong đó giá trị của chúng nêu rõ các trường hợp của từng loại khối. Ví dụ như, khối GENERATE được sử dụng để tạo ra lưu lượng trong đó sự phân bổ cùng vơi một biến ngẫu nhiên của thời gian giữa hai lần tạo lưu lượng được định trước trong một toán hạng. Khối ADVANCE được dùng để đưa 1 lưu lượng vào trạng thái trễ thời gian và khối này một toán hạng để mô tả sự phân bố của thời gian trễ. Trong khi đó, một lưu lượng sử dụng khối SEIZE để yêu cầu 1 phương tiện và SEIZE có 1 toán hạng để xác định phương tiện cần thiết. GPSS/H cung cấp khoảng 65 loại khối.

Chuỗi sự kiện hiện thời (CEC):
Chuỗi sự kiện hiện thời của GPSS/H gồm các lưu lượng ở trạng thái sẵn sàng. Nó cũng có các lưu lượng ở trạng thái hoạt động. Ngoài ra, chuỗi các sự kiện hiện thời có chức năng giống như một danh sách trễ GPSS/HH toàn cục, gồm các lưu lượng trễ có điều kiện trong mô hình. Các lưu lượng này được trộn lẫn với lưu lượng ở trạng thái hoạt động và các lưu lượng ở trạng thái sẵn sàng trên CEC.
Các lưu lượng có 1 thuộc tính mang trị số riêng biệt – được gọi là Mức Ưu Tiên. Mức ưu tiên của một lưu lượng được gán cho lưu lượng khi tạo ra nó và có thể được thay đổi liên tục trong suốt thời gian hoạt động của lưu lượng đó. Các lưu lượng ở CEC được sắp xếp theo kiểu FIFO theo mức độ ưu tiên. Nghĩa là lưu lượng nào đến trước sẽ được đáp ứng trước theo mức độ ưu tiên. (Chuỗi người sử dụng (được thảo luận dưới đây) được sử dụng để mô hình hóa những thứ tự dịch vụ khác)

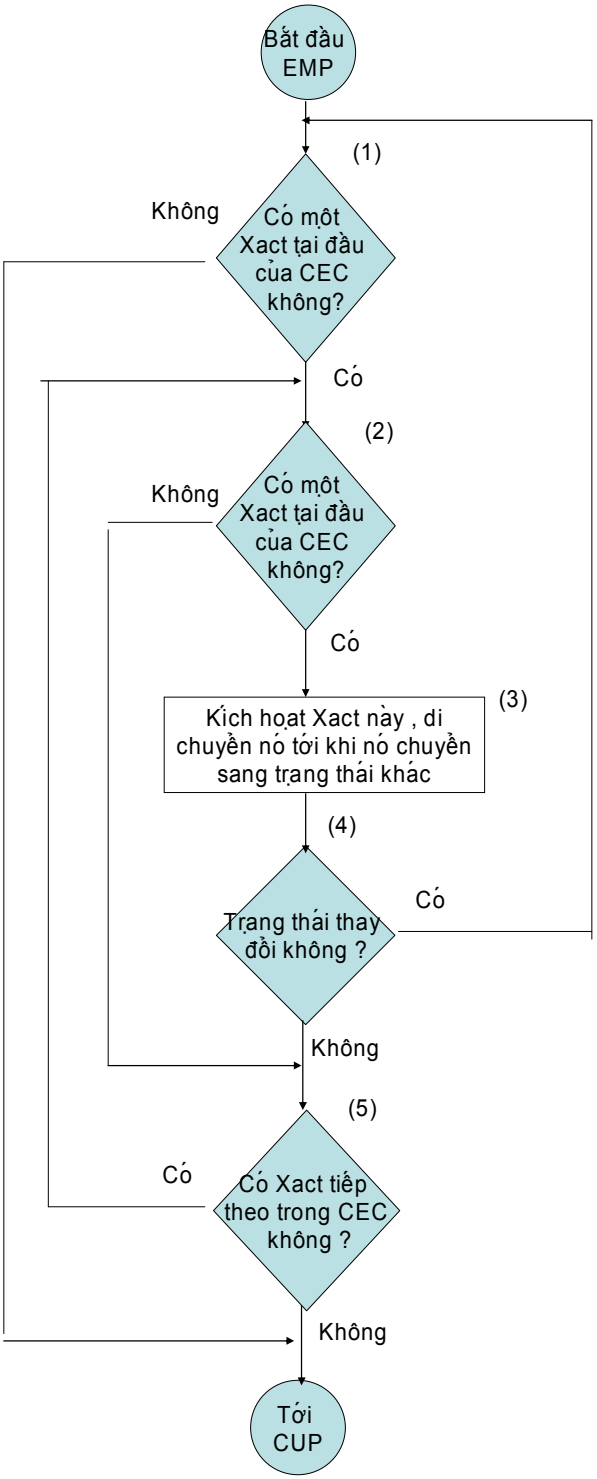
GPSS/H cũng có một cơ cấu hàng đợi (queue) và một khối hàng đợi (khối QUEUE), nhưng ko giống như khối QUEUE của SIMAN, khối QUEUE của GPSS/H không thực hiện chức năng quản lý danh sách. QUEUE của GPSS/H được sử dụng với mục đích duy nhất là thu thập các giá trị thống kê. Phù hợp với thực tế là, ngoại trừ CEC (và một số danh sách trễ bên trong hỗ trợ CEC được thảo luận dưới đây), không có thêm danh sách trễ nào trong GPSS/H. Chính nhờ có chức năng của danh sách trễ mà GPSS/H CEC thường không ở tình trạng trống rỗng khi một giai đoạn thực thể hoạt động kết thúc.

Giai đoạn quét (Scan phase)

Trong GPSS/H, giai đoạn thực thể hoạt động được gọi là giai đoạn quét (Scan Phase). Giai đoạn này chịu trách nhiệm đưa các lưu lượng trễ có điều kiện và các lưu lượng ở trạng thái sẵn sàng vào chuỗi danh sách hiện thời. Trong một giai đoạn quét, từ đầu tới cuối của 1 CEC, các lưu lượng có thể được kiểm tra tra nhiều lần. Đó là vì khi một lưu lượng ở trạng thái hoạt động thực hiện một khối mà khối này giải quyết vấn đề trễ, các lưu lượng trễ có điều kiện này có thể ở đâu đó phía trên lưu lượng đang hoạt động này trong CEC.

Các lưu lượng trễ có điều kiện chuyển đổi từ trạng thái trễ có điều kiện sang trạng thái sẵn sàng và sau đó phải được chuyển sang trạng thái hoạt động rồi được di chuyển trước khi đồng hồ mô phỏng thay đổi giá trị.

Sau khi lưu lượng hoạt động này ra khỏi trạng thái hoạt động, giai đoạn quét quay lại lưu lượng nằm tại vị trí đầu của CEC và từ đó bắt đầu kiểm tra lại từ trên xuống dưới để đảm bảo là nó sẽ phát hiện được các lưu lượng bất kì nào đó có thể vừa chuyển từ trạng thái trễ có điều kiện sang trạng thái sẵn sàng.



Hình 3.15: Giai đoạn thực thể hoạt độngGPSS/H

Logic của giai đoạn thực thể hoạt động trong GPSS/H (giai đoạn quét) được thể hiện dưới dạng biểu đồ ở hình 3.15. Trong biểu đồ này, các đối tượng khác nhau được đánh số để thuận tiện cho quá trình thảo luận. Theo đó, GPSS/H bắt đầu từ một giai đoạn quét (Scan Phase) kèm 1 sự kiểm tra (ô 1) để xem liệu có lưu lượng nào đang ở trong CEC hay không. (CEC có thể trống nếu giai đoạn quét xảy ra tại thời điểm mô phỏng 0.0 hoặc nếu giai đoạn quét khởi động lại). Nếu CEC trống, giai đoạn cập nhật đồng hồ tiếp tục(đường dẫn "no" từ ô 1)

Giả sử có một lưu lượng ở đỉnh CEC. Nếu lưu lượng này ở trạng thái sẵn sàng (ô số 2), lưu lượng này được kích hoạt và di chuyển (ô 3) cho đến khi nó buộc phải chuyển sang trạng thái hoạt động. Sau đó kiểm tra (ô 4) để quyết định xem liệu những việc làm này có thể giải quyết được 1 hay

nhiều điều kiện gây trễ hay không. Nếu trễ có thể được giải quyết, quá trình quét của CEC bắt đầu từ đỉnh của CEC (đường dẫn "yes" từ ô 5). Nếu không có lưu lượng nào tiếp theo, lưu lượng cuối cùng trên CEC được thực hiện và giai đoạn cập nhật đồng hồ thực hiện tiếp theo (đường dẫn "no" từ ô số 5)

Nếu lưu lượng được xử lí ở ô 2 trong hình 3.15 không ở trạng thái sẵn sàng, nó là một lưu lượng trễ có điều kiện. Trong trường hợp như vậy sẽ có một đường dẫn "no" từ ô số 2 đến ô số 5. Mục đích là nhằm bỏ qua các lưu lượng trễ có điều kiện trong suốt giai đoạn quét. Phương thức thực hiện của GPSS/H, trong đó thực hiện việc giữ lại các lưu lượng trễ có điều kiện trong CEC và kiểm tra chúng 1 hoặc nhiều lần trong suốt giai đoạn quét để xem liệu chúng có được đưa vào trạng thái sẵn sàng chưa, có nghĩa là tất cả các lưu lượng này đang ở trong cơ chế thời gian chờ thăm dò.

Chuỗi sự kiện tương lai:

Trong GPSS/H, danh sách các sự kiện tương lai còn được gọi là Chuỗi sự kiện tương lai (FEC) . Các lưu lượng được sắp xếp trên FEC theo trật tự thời gian di chuyển tăng dần. Với các thời gian di chuyển trùng nhau (move-time ties) thì giải quyết bởi cách thức vào trước cũng sẽ ra trước (FIFO). Giai đoạn cập nhật đồng hồ của GPSS/H lấy ra nhiều lưu lượng từ chuỗi sự kiện tương lai theo trật tự từ trên xuống dưới nếu như chúng có cùng thời gian di chuyển sớm nhất, đưa chúng lần lượt vào chuỗi sự kiện hiện thời (sắp xếp theo mức độ ưu tiên)

Giai đoạn cập nhật xung đồng hồ

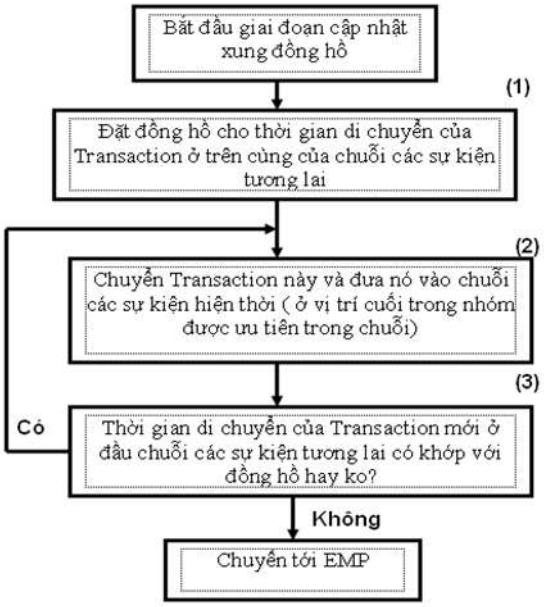
Nguyên lý của giai đoạn cập nhật đồng hồ được thể hiện ở hình 3.16. Thời gian mô phỏng được thiết lập bằng với thời gian di chuyển của lưu lượng từ đầu trong chuỗi các sự kiện tương lai (ô số1); sau đó các lưu lượng được chuyển từ chuỗi sự kiện tương lai sang chuỗi các sự kiện hiện thời (ô số 2). Thực hiện kiểm tra (ô số 3) để xem liệu thời gian di chuyển của lưu lượng ở đầu của chuỗi sự kiện tương lai có khớp với thời gian mô phỏng không. Nếu khớp, lưu lượng này sẽ chuyển từ đầu chuỗi FEC sang chuỗi CEC (xem đường dẫn "yes" từ ô số 3 lên ô số 2). Qúa trình quay vòng này lặp đi lặp lại cho đến khi lưu lượng ở vị trí đầu trong chuỗi sự kiện tương lai có thời gian di chuyển vượt thời gian mô phỏng hiện thời. Pha cập nhật đồng hồ hoàn tất và giai đoạn quét tiếp tục (đường dẫn "no" từ ô số 3)

Chuỗi người sử dụng

GPSS/H thực hiện trạng thái không hoạt động với chuỗi người sử dụng – hay chính là danh sách do người dùng quản lí của các lưu lượng. Các lưu lượng có thể được đưa vào chuỗi người dùng ở đầu, ở cuối hoặc sắp xếp theo thứ tự tăng dần dựa theo thuộc tính lưu lượng được chỉ định bởi người thiết kế mô hình.

Sau khi một Transaction tự đưa nó vào chuỗi người sử dụng, nó chỉ có thể bị 1 lưu lượngở trạng thái hoạt động loại bỏ. Thứ tự chuyểnTransaction ra khỏi chuỗi người dùng có thể từ trên xuống dưới, từ dưới lên trên hoặc có thể dựa vào giá trị của của biểu thức Boolean xác định bởi người sử dụng.

Các lưu lượng bị di chuyển được đưa vào trạng thái sẵn sàng trong chuỗi sự kiện hiện thời và cờ báo hiệu thay đổi trạng thái bật. Kết quả là, quá trình quét CEC sẽ bắt đầu trở lại (đường dẫn "yes" từ ô số 4 quay lại ô số 1 trên hình 3.15) sau khi lưu lượng hoạt động chuyển từ trạng thái hoạt động. Điều này đảm bảo rằng các lưu lượng mới tới ở trạng thái sẵn sàng sẽ được đưa vào trạng thái hoạt động trong chuỗi sự kiện hiện thời trước khi giai đoạn cập nhật đồng hồ diễn ra.



Hình 3.16 Giai đoạn cập nhật xung đồng hồ trong GPSS/H

Tài liệu tham khảo

[Bank95a] J. Banks et al., "Getting started with GPSS/H", 2nd ed., Wolverine Software Corporation, Annandale, VA
[Bank95b] J. Banks et al., "Introduction to SIMAN V and Cinema V", Wiley, New York
[Benson] D. Benson, "Simulation modeling and optimization using ProModel", in the Proc. Of Winter Simulation Conference 1997, La Jolla, Clifornia, pp: 587-593

Chương 4: Mô phỏng sự kiện rời rạc trong các hệ thống máy tính và truyền thông

Tác giả: Alfred Hartmann và Herb Schwetman
Dịch thuật: Nguyễn Quang Huy, Nguyễn Hoàng Minh Phúc, Nguyễn Thanh Hải
Biên tập: Nguyễn Anh Tôn

4.1 Giới thiệu

Trong chương này chúng ta sẽ thảo luận về những kỹ thuật được sử dụng để xây dựng các chương trình mô phỏng sự kiện rời rạc cho các hệ thống máy tính và truyền thông. Trong khi các phương pháp phân tích như lý thuyết hàng đợi đôi khi được dùng (Lazowska , 1984) và có thể cho ta những sự hiểu biết có giá trị, phạm vi của chúng bị giới hạn bởi những vấn đề, mà ở đó chỉ có các phương pháp dùng phép phân tích giải pháp mới khả thi. Mô phỏng dựa trên máy tính được ứng dụng khá rộng rãi, với tiện ích của các gói phần mềm mô phỏng cùng sự tăng trưởng nhanh chóng của khả năng tính toán phí tổn thấp đến mỗi máy tính, việc phân tích dựa trên mô phỏng đang ngày càng có vị trí quan trọng. Ngày nay, hầu hết những nhà thiết kế hệ thống máy tính và truyền thông đều thân thuộc với ngôn ngữ lập trình C/C++ và khái niệm về các quá trình tính toán. Điều này khiến việc mô phỏng trên C/C++ dựa vào các quá trình trở thành một lựa chọn tiện ích và tự nhiên cho rất nhiều những dự án kiểu mẫu.

4.2 Các khái niệm cơ bản

Một mạng máy tính hay truyền thông bao gồm một tập hợp các nguồn tài nguyên. Những thực thể (ví dụ như những công việc, các chương trình, các nhiệm vụ, giao dịch, thông điệp, v.v.) cạnh tranh về quyền sử dụng các nguồn tài nguyên này. Các hình thái của những hệ thống này sỡ hữu, ít nhất là ở mức độ nào đó, những cấu trúc đại diện cho cả những nguồn tài nguyên và những thực thể. Hầu hết bất kỳ kiểu mô phỏng của bất kỳ loại hệ thống nào cũng có cấu trúc tương đồng. Trong chương này chúng ta sẽ khảo sát xem những sản phẩm tạo nên các mạng máy tính và truyền thông điển hình khác biệt như thế nào so với những kiểu mô phỏng của các hệ thống khác loại.

4.2.1 Những mục tiêu

Mục tiêu chính của tất cả các dự án hệ thống điển hình là cho ra những sự đánh giá về hiệu suất của hệ thống. Đối với các hệ thống máy tính, thông số đo lường quan trọng nhất cần đạt được là thời gian đáp ứng nhiệm vụ. Còn với những mạng truyền thông, nó lại là thời gian tiềm tàng của thông điệp. Vì thế mục đích của nhiều dự án là đưa ra những sự đánh giá chính xác về các thông số đo lường. Những mục tiêu khác bao gồm việc cung cấp cái nhìn rõ ràng về hoạt động của hệ thống và việc hướng dẫn để giảm thiểu sự tác động của những đường giới hạn đáp ứng.

4.2.2 Các nguồn tài nguyên và những thực thể

Như đã đề cập ở trên, các mô hình mô phỏng chủ yếu bao gồm những thực thể (các quá trình, những khách hàng, các thông điệp, những giao dịch, v.v.) và những nguồn tài nguyên (bộ vi xử lý, bộ nhớ, bus đường truyền, các liên kết giao tiếp, v.v).

Trong các mô hình này, có một vài kiểu thực thể khác nhau mà cũng có thế có các mẫu đa dạng của từng kiểu thực thể, chúng hoạt động ở những điểm trùng nhau và khác nhau trong thời điểm được mô phỏng.Trong nhiều trường hợp, những thực thể đó đang cạnh tranh để giành được một số nguồn tài nguyên của mô hình hệ thống. Ví dụ, trong mô hình hệ thống máy tính, một vài nhiệm vụ mô phỏng có thể cạnh tranh nguồn tài nguyên của CPU.
Tương tự như vậy, có rất nhiều kiểu tài nguyên trong một mô hình hệ thống. Chúng có thể được chia ra làm 2 loại chính : (1) các nguồn chủ động và (2) các nguồn bị động. Nét đặc thù của chúng phụ thuộc vào việc những kiểu thực thể làm gì sau khi nắm giữ việc truy cập vào một thành phần của nguồn tài nguyên. Một thành phần của một nguồn tài nguyên chủ động thường được dùng hoặc bị nắm giữ trong một khoảng thời gian nhất định và sau đó được giải phóng. Còn một nguồn tài

nguyên bị động thì bị nắm giữ bởi một thực thể; lúc này thực thể xử lý để đáp ứng các hoạt động khác, bao gồm cả việc giành lấy truy cập đến các nguồn tài nguyên khác.

Những ví dụ về các nguồn tài nguyên chủ động trong một hệ thống máy tính bao gồm các bộ vi xử lý trung tâm, ổ đĩa cứng, và các bộ điều khiển. Những ví dụ về nguồn tài nguyên thụ động bao gồm bộ nhớ chính và các bus đường truyền. Trong một mạng truyền thông, nếu các bộ vi xử lý thông điệp và các đường truyền liên nút mạng có thể được xem như là các nguồn tài nguyên chủ động, các bộ đệm sẽ là những ví dụ về nguồn tài nguyên bị động.

Các thực thể của các mô hình mạng này phụ thuộc vào các cấp độ của mô hình. Với mô hình mạng mức cao của một hệ thống máy tính, các thực thể có thể là các chương trình hoặc là các giao dịch. Trong mô hình mạng truyền thông, các thực thể đó có thể là những thông điệp. Đối với mô hình hệ thống máy tính ở mức thấp, những thành phần có thể là những chỉ dẫn tính toán độc lập, các yêu cầu xuất nhập dữ liệu, và sự truyền dữ liệu đến và đi từ bộ nhớ chính. Ở một mô hình mức thấp hơn của mạng truyền thông, các thực thể có thể là các gói hoặc tế bào dữ liệu.

4.2.3 Tải làm việc

Mỗi mô hình hệ thống được đặc trưng bởi một mức tải làm việc của nó. Ở đây *tải làm việc* đề cập đến trình tự các yêu cầu của các thực thể đối với các nguồn tài nguyên của hệ thống. Việc đạt được một đặc tính chính xác về tải làm việc của hệ thống là một trong những bước quan trọng nhất để hướng đến việc xây dựng một mô hình mạng hữu ích và chính xác. Với các hệ thống máy tính, nhiệm vụ này, trong một vài trường hợp nào đó, trở nên đơn giản hơn bởi sự có mặt của những công cụ hệ thống có khả năng tự động thu thập các dữ liệu cần thiết. Ví dụ, nếu các thực thể của mô hình đang giao dịch trực tuyến, thì bộ giám sát giao dịch có thể giữ các biên bản giao dịch nhằm sử dụng để xây dựng một đặc tính dòng giao dịch đầu vào. Một cách tương tự, trong một mạng máy tính, đó là các biên bản thông điệp của các nút mạng để xây dựng nên các bản trích ngang của những luồng thông điệp trong mạng.

Như với hầu hết các mô hình mạng, việc có được một đặc tính chính xác về tải làm việc ứng với các nhu cầu cho các nguồn tài nguyên của hệ thống là cần thiết. Với các hệ thống máy tính, điều này có thể gặp phải một số những khó khăn. Ví dụ, một đầu vào phân tích cho một mô hình có thể là sự phân bổ thời gian dịch vụ tại một ổ đĩa, nhưng gói tính toán hệ thống có thể báo cáo các thời điểm đáp ứng tại ổ này (và thời điểm đáp ứng hữu ích tại ổ đĩa sẽ có thể là một đầu ra của mô hình). Tương tự, gói tính toán có thể báo cáo các ký tự trên hướng truyền đến một ổ đĩa, nhưng cũng khó để tìm được sự tương quan giữa số lượng các ký tự đã truyền với thời gian đáp ứng.

Một cách điển hình, những gì được yêu cầu đối với một mô hình của hệ thống máy tính là một bản mô tả sơ lược các kiểu công việc hay các nhiệm vụ được mô hình hoá. Bản mô tả này sẽ bao gồm một sự phân tầng của các nhiệm vụ theo các lớp khác nhau, và sau đó ứng với từng lớp là một bản tóm tắt các yêu cầu dịch vụ đối với các nguồn tài nguyên khác nhau của hệ thống. Ví dụ, một tải làm việc của một hệ thống xử lý giao dịch trực tuyến có thể được phân loại theo các kiểu giao dịch khác nhau đã và sẽ được đệ trình, và sau đó ứng với mỗi kiểu giao dịch, sẽ lựa chọn các thông tin sau:

- Số lượng các giao dịch của mỗi kiểu
- Lượng thời gian xử lý trên một giao dịch
- Số lượng các truy cập ổ đĩa trên một giao dịch
- Lượng dữ liệu đã được gởi lại cho người yêu cầu trên một giao dịch

Với một mạng truyền thông, đặc tính tải làm việc có thể bao gồm các kiểu thông tin sau:

- Số lượng các thông điệp đã tạo ra tại mỗi nút mạng
- Sự phân bổ chiều dài thông điệp
- Thông tin về điểm đến của mỗi thông điệp (ví dụ, khoảng cách đến đích)

Tỷ lệ các thông điệp yêu cầu một sự hồi âm Kết quả cuối cùng của đặc tính tải làm việc này là một tập các giá trị tham số mà có thể được yêu cầu để mô hình hoá một cách chính xác tải làm việc trong mô hình.

4.2.4 Đầu ra: Sự đo lường khả năng đáp ứng

Như đã đề cập ở trên, những mô hình hệ thống thường được sử dụng để đạt được các đánh giá về thời gian đáp ứng trung bình của hệ thống ứng với các thành phần của tải làm việc. Một cách tương tự, trong các mô hình mạng, những đánh giá về độ trễ (latency) trung bình của thông điệp và/ hoặc băng thông đã được phân phối đều được quan tâm. Ở đây, *thời gian đáp ứng* là thời gian khi một yêu cầu dịch vụ được tạo ra cho đến khi yêu cầu được thực hiện . Độ trễ(***latency)*** ở đây được dùng cho thời gian đáp ứng trong một hệ thống phụ xuất nhập, một hệ thống bộ nhớ phụ, hoặc trong một mạng truyền thông. Những thời gian đáp ứng ***(latencies)*** có đơn vị là thời gian trên yêu cầu. Nghịch đảo của thời gian đáp ứng là tỷ lệ dung lượng (ví dụ như các giao dịch trên một giây), nghịch đảo của độ trễ(latency) là băng thông (ví dụ Mbytes/s). Tất cả các tham số trên thường là những đầu ra của các mô hình mô phỏng.

Nếu một mô hình đang được dùng để có được cái nhìn tường tận về những vấn đề hiệu suất (nơi mà những vấn đề hiệu suất thường được thể hiện bởi những khoảng thời gian đáp ứng hoặc độ trễ dài không thể chấp nhận được), càng nhiều dữ liệu đầu ra được yêu cầu để xác định chính xác nguyên nhân của vấn đề. Những dữ liệu thêm vào này thường chứa các bảng thống kê về thời gian được dùng tại các nguồn tài nguyên độc lập của mô hình hệ thống hoặc là thời gian được dùng trong các hoạt động độc lập để hoàn thành một thành phần tải làm việc. Trong một số trường hợp, việc chèn thêm các thành phần đặc biệt là cần thiết để thu được các dữ liệu cần thiết để xác định nguyên nhân của các vấn đề hiệu suất.

4.2.5 Tính 2 mặt của các mô hình

Các mô hình được định hướng theo tiến trình của các hệ thống truyền thông và máy tính có sự phân công các tiến trình riêng biệt cho các thực thể mô phỏng. Đối với việc mô phỏng hệ thống máy tính, các quá trình được giao việc cho các tải làm việc (ví dụ để đặc trưng cho các quá trình của người dùng và các ứng dụng), và các quá trình này yêu cầu và giải phóng tài nguyên hệ thống khi nó được hoàn thành. Trong mô phỏng hệ thống truyền thông, các quá trình lại được giao việc cho các nguồn tài nguyên (ví dụ như các chuyển mạch hoặc là các kênh), và các quá trình này nắm giữ các thông điệp tải làm việc trong thời gian còn hoạt động. Những sự phân công quá trình này nhằm giữ cho số lượng quá trình trong một phép mô phỏng ở mức hợp lý, kể từ khi hệ thống mô phỏng gặp phải bối cảnh chuyển mạch quá tải với quá nhiều quá trình. Một lẽ thông thường, nó sẽ giới hạn mô hình về mỗi thông điệp như là một quá trình cạnh tranh về nguồn tài nguyên hệ thống truyền thông, do số lượng quá lớn của chúng.

Những khía cạnh khác nhau của các mô hình nhận thức làm nên tính 2 mặt của mô hình giữa các mô hình hệ thống máy tính và các mô hình hệ thống truyền thông.

Thực thể mô phỏng	Hệ thống máy tính	Hệ thống truyền thông
Mục tải làm việc	Quá trình	Token (cấu trúc)
Mục nguồn tài nguyên	Cấu trúc	Quá trình

Tính hai mặt này có thể dẫn đến một sự khác nhau trong tiến độ giữa mô phỏng hệ thống máy tính và hệ thống truyền thông, và có lẽ khác nhau về cả các công cụ mô phỏng.

Các dịch vụ hàng đợi tương thích với sự giành lấy các nguồn tài nguyên cạnh tranh trong một mô phỏng hệ thống máy tính có thể hoặc không thoả mãn những yêu cầu của thông điệp tải làm việc thực hiện trong một mô phỏng hệ thống truyền thông. FCFS (First come, first served) là một quy luật dịch vụ chung cho cả hệ thống máy tính và truyền thông, trong khi quy luật lên lịch của bộ xử lý Round-Robin sẽ gần như chắc chắn không ứng dụng được cho các mô phỏng hệ thống truyền thông. Việc mô phỏng hệ thống truyền thông có thể cần đến một dải rộng các trình tự quản lý bộ

đệm khác nhau mà có thể không được phản ảnh trong việc lựa chọn những trình tự cấp phát nguồn tài nguyên hiện hữu trong các tiện ích mô phỏng của hệ thống máy tính. Ví dụ, trình tự quản lý bộ đệm "leaky bucket" được dùng cho bộ điều khiển nguồn tài nguyên mạng ATM là duy nhất trong truyền thông, trong khi những phương pháp kết hợp thời hạn và sắp đặt các thông điệp tế bào theo trình tự ưu tiên dùng cho việc giảm thiểu tắc nghẽn trong mạng ATM.

4.2.6 Những mô hình hướng đối tượng của các hệ thống

Chúng ta có thể cho rằng việc thiết kế phần mềm hướng đối tượng dựa trên các quy tắc sau:

- Sự tóm lược(Encapsulation): khả năng kết nối dữ liệu và các hàm để tối ưu dữ liệu về những dạng tóm lược
- Sự kế thừa(Inheritance): khả năng rút ra các kiểu phụ từ các kiểu tóm lược đang tồn tại
- Sự đa dạng(Polymorphism): khả năng cho phép các hàm kết hợp khéo léo các đối tượng từ các kiểu khác nhau mà chắc chắn có chia xẻ mối quan hệ tương quan cùng kiểu.

Những ưu điểm của phương pháp này cung cấp tất cả các kiểu ứng dụng phát triển phần mềm cũng như phát triển việc mô phỏng. Các thực thể mô phỏng có thể được gói gọn trong các đối tượng, các lớp của chúng có thể được nhóm thành các lớp đối tượng cơ bản và được phân biệt việc sử dụng tính kế thừa vào bên trong các lớp con tuỳ biến, và các hàm hình thái có thể được định nghĩa một cách hợp lý như các phương pháp của các lớp đối tượng. Tất cả những kỹ thuật hướng đối tượng đều nhằm phục vụ cho việc tổ chức và đơn giản hoá công tác phát triển mô hình mô phỏng.

Các hệ thống phát triển mô phỏng và các ngôn ngữ lập trình cung cấp các lớp cơ bản được định nghĩa trước cho các kiểu chung của các quy trình và nguồn tài nguyên mô phỏng. Chúng có thể được kế thừa bởi các lớp con được định nghĩa mới trong một sự nỗ lực mô phỏng đặc biệt. Các hàm đa dạng có thể được định nghĩa trước tạo nên sự kết hợp khéo léo đúng chuẩn trên các đối tượng người dùng, mà cụ thể là các ví dụ về các lớp xuất phát từ nguồn lớp cơ bản.

4.2.7 Trạng thái giả song song trong các mô hình hệ thống

Mô phỏng hướng theo quá trình sử dụng đa quá trình để mô phỏng các hoạt động song song trong hệ thống được mô phỏng. Tuy nhiên, việc thực thi phép mô phỏng này có thể được chèn một cách tuần tự vào một bộ xử lý đơn lẻ. Việc chèn liên kết nhìn chung được thực hiện *một cách tiền định* để mà các kết quả được lặp lại từ phép mô phỏng này sang phép mô phỏng khác. Ví như các dạng trạng thái song song đã được đề cập đến như *trạng thái giả song song*, để phân biệt chúng với *trạng thái song song thật,* là không tuần tự và thường không tiền định. Trạng thái song song thật có thể được khai thác *phía dưới* môi trường mô phỏng giả song song, để tăng tốc mô phỏng, nhưng điều này rõ ràng trong việc thiết kế mô phỏng. Trạng thái giả song song không rõ ràng trong việc thiết kế mô phỏng nhưng cũng là một thành phần không thể thiếu. Lưu ý, một vài ứng dụng thành công của các kỹ thuật mô phỏng song song đã bao gồm các mô hình của các hệ thống truyền thông lớn.

Trạng thái giả song song phải cung cấp các tiện ích cho các quá trình mô phỏng ở dạng khối (tức là việc chờ đợi các phép mô phỏng xong, các nguồn tài nguyên sẵn sàng trở lại, hoặc một sự kiện mô phỏng xuất hiện). Tại một thời điểm hợp lý trong thời gian mô phỏng rời rạc, các quá trình phải được *tháo rời* và tái lập lại việc mô phỏng. Các đa quá trình cần một vài dạng truyền thông liên quá trình được đồng bộ, mà có thể được cung cấp bởi các sự kiện bổ sung các lớp đồng bộ, các tín hiệu, các hộp thư, các bản mã morse.v.v.

4.3 Các hệ thống máy tính

Một mô hình mô phỏng của hệ thống máy tính (MacDougall, 1987) phải có được các thành phần cần thiết của hệ thống thực. Chúng bao gồm (1) các thành phần tải làm việc, (2) các nguồn tài nguyên hệ thống và (3) các luật hệ thống (có thể chi phối việc cung cấp các nguồn tài nguyên cho các thành phần của tải làm việc). Trong phần này chúng ta sẽ thảo luận về các mục liên quan đến việc mô hình hoá các thành phần trên (Jain, 1991).

4.3.1 Tải làm việc là gì?

Các hệ thống máy tính tồn tại các thành phần xử lý của tải làm việc, nơi mà phụ thuộc cách hệ thống được mô hình hoá, một thành phần tải làm việc có thể là: (1) công việc, chương trình, hay nhiệm vụ; (2) giao dịch hoặc nghi vấn; hay (3) các yêu cầu vào/ra hoặc các yêu cầu về một thứ gì đó trong bộ nhớ chính. Mỗi một loại là một chuỗi các yêu cầu về các dịch vụ sử dụng tài nguyên của hệ thống. Ví dụ, cho rằng một công việc hay nhiệm vụ là thành phần của tải làm việc. Một công việc (một chương trình) là một chuỗi lần lượt các yêu cầu về thời gian của một CPU (sự bùng lên của CPU) và các tác vụ vào/ra (sự chuyển giao các khối dữ liệu vào và ra từ các thiết bị I/O như các ổ đĩa). Thêm vào đó, công việc sẽ yêu cầu việc sử dụng các khối của bộ nhớ chính. Trong một số hệ thống hiện đại, nhiều công việc sẽ thực hiện cùng một lúc, vì thế có sự tranh chấp về nguồn tài nguyên. Đáp ứng hệ thống (các thời gian đáp ứng công việc) phản ánh khả năng của hệ thống có thể thoả mãn các yêu cầu đối lập cho việc sử dụng nguồn tài nguyên.

4.3.2 Mô hình hoá các thành phần hệ thống

Các hệ thống máy tính bao gồm phần cứng và phần mềm. Ngoài ra còn có hệ điều hành quản lý việc truy cập vào các thành phần này; nó đại diện cho cơ chế điều khiển nhằm đảm bảo sự chính xác và tính hiệu quả trong hoạt động của hệ thống. Trong một mô hình mô phỏng hệ thống như vậy, tất cả các thành phần quan trọng, cũng như các quy luật điều khiển truy cập vào chúng, phải được đặc tả chính xác nếu muốn đạt được sự đánh giá đúng về đáp ứng của hệ thống.

Các thành phần phần cứng. Phần cứng quan trọng nhất của một hệ thống máy tính là CPU và các thiết bị ngoại vi. Bộ nhớ chính là điểm then chốt cho hoạt động của hệ thống, nhưng trong hầu hết các hệ thống hiện đại, bộ nhớ chính không lớn lắm và thường không phải là thành phần chính để xác định khả năng đáp ứng của hệ thống. Tuy nhiên, việc kết hợp hoạt động của bộ nhớ chính ra vào một mô hình thường không phải là một nhiệm vụ khó. Và một phần quan trọng nữa thường hay bị bỏ qua đó là việc liên nối giữa các phần cứng (gọi là *bus*), thường được dùng để kiên kết CPU và các thiết bị vào/ra với bộ nhớ chính. Việc mô hình hoá các thành phần phần cứng này là hiển nhiên vì chúng đại diện cho các nguồn tài nguyên tĩnh. Vấn đề ở đây là phải mô hình hoá một cách chính xác các loại truy nhập song song và nối tiếp có thể có đối với tất cả các thành phần trên.

Các thành phần phần mềm. Các thành phần phần mềm có thể khó đặc tả hơn. Ví như, một thành phần phần mềm có thể là một chuỗi dài các hành động và các quy luật hoạt động phức tạp. Thêm vào đó, một số thành phần được cấp bởi một nhà cung cấp thứ 3, và hoạt động nội tại của chúng sẽ không được phơi bày trong các bộ mô hình hoá hệ thống.

Có một lớp của thành phần phần mềm được tìm thấy trong các hệ thống bao gồm tất cả các dịch vụ cung cấp cho các chương trình người dùng. Một ví dụ là hệ thống quản lý cơ sở dữ liệu (DBMS-database management system). Rất nhiều các ứng dụng trong một hệ thống yêu cầu dịch vụ từ DBMS. Tuy nhiên, cũng có những giới hạn về số lượng những yêu cầu mà DBMS có thể nắm giữ một cách đồng thời. Cũng chính vì thế một server có thể bị giới hạn về khả năng đáp ứng các công việc mà phải đòi hỏi truy cập dữ liệu thông qua DBMS. Các thành phần phần mềm khác có thể tác động đến khả năng đáp ứng trong một hệ thống máy tính bao gồm các bộ giám sát quá trình giao dịch, các hệ thống truy cập mạng và các server tập tin từ xa.

Nếu một thành phần mạng được xác định có thể là một nút thắt cổ chai về đáp ứng trong một hệ thống sẽ được mô hình hoá, thì một nổ lực đặc biệt có thể được yêu cầu để đảm bảo rằng một biểu diễn đặc trưng của các thành phần là có giá trị. Ví dụ, nếu một DBMS thương mại là một thành phần quan trọng trong một hệ thống, việc mô hình hệ thống có thể đòi hỏi một mô hình mô phỏng của DBMS. Để kết hợp thành phần này vào hệ thống, bộ mô hình hoá phải thực hiện các bước sau:

- Thu thập thông tin thêm từ các nhà cung cấp hoặc từ các nguồn khác

- Thực hiện một vài nghiên cứu đo lường hộp đen, để suy ra các đặc tính của DBMS
- Đạt được một mô hình mô phỏng tiền tiền cấu trúc của DBMS.

Một khía cạnh khác của các thành phần phần mềm trong một mô hình là sự quan trọng của các tập quy luật về quản lý và điều khiển được cung cấp bởi các thành phần khác. Các quy luật và quy tắc có trong hệ điều hành (có nhiệm vụ điều khiển hoạt động của toàn bộ hệ thống) cũng như có trong các thành phần đã đề cập trên. Chúng đặc tả sự ưu tiên, các quy tắc lên lịch, quản lý các nguồn tài nguyên, và các quy tắc thoả thuận với sự xung đột và quá tải (ví dụ, chiều dài hàng đợi tối đa tại các thiết bị vào/ra). Phụ thuộc vào mức độ chi tiết trong mô hình, các quy luật này có thể là một thành phần quan trọng trong sự phát triển của một mô hình hệ thống chính xác.

4.3.3 Hoạt động không đồng bộ

Nhằm cải tiến việc sử dụng các nguồn tài nguyên của hệ thống, các hệ thống hiện đại thường khuyến khích kiểu không đồng bộ. Nguyên tắc cơ bản là một chương trình có thể nhập vào các hoạt động vào/ra một cách song song với các ứng dụng liên tục của CPU. Một mô hình hệ thống phải đạt được kiểu hoạt động song song không đồng bộ này. Một bộ mô phỏng hướng quá trình cung cấp một nền tảng cho các mô hình thực thi mà trong đó các thành phần chính có thể hoạt động một cách không đồng bộ.

4.3.4 Những sự cân bằng: Chi tiết so với giá cả

Mục đích của một mô hình hệ thống, nói một cách đơn giản nhất, là để tạo ra các khoảng thời gian đáp ứng cho các thành phần của tải làm việc. Những khoảng thời gian này có thể sau đó được sử dụng để cung cấp sự ước tính cần thiết để tạo ra những sự phán đoán về đáp ứng của hệ thống.

Một khoảng thời gian đáp ứng tiêu biểu bao gồm một hay nhiều khoảng con, mà trong đó mỗi khoảng con đều có một khoảng trễ và một khoảng phục vụ tại một vài nguồn tài nguyên. Một hệ thống ở mức cao chỉ có một vài nguồn tài nguyên, và kết quả là khoảng thời gian đáp ứng chỉ có một vài khoảng con. Ví như một mô hình có thể thực thi rất nhanh , nhưng kết quả ước lượng của các khoảng thời gian đáp ứng lại không chính xác.

Nhằm nâng cao tính chính xác, các nguồn tài nguyên ở mức cao có thể được định nghĩa lại cho phù hợp với những sự lựa chọn các nguồn tài nguyên mức thấp hơn. Khi làm được điều này, những gì trước đây là một khoảng thời gian phục vụ đơn lẻ tại một nguồn tài nguyên nay là một chuỗi các khoảng thời gian phục vụ và trễ cho những nguồn tài nguyên thấp hơn. Điều thu được là một đặc trưng chính xác hơn về tài nguyên; giá trị ở đây là sự gia tăng về số lượng các sự kiện được mô phỏng, mà sẽ chạy lâu hơn với mô hình cũ.

Một ví dụ, trong một mô hình cấp cao, một hoạt động vào_ra có thể được mô hình hoá như là một khoảng thời gian phục vụ đơn lẻ. Trong mô hình này nó có thể là một tài nguyên vào/ra đơn lẻ, và là các chương trình lấy chúng, một cách liên tiếp, phục vụ tại tài nguyên. Thiết bị đầu vào/đầu ra thực tế và các thành phần kết nối hỗ trợ thì phức tạp hơn; việc nắm được rõ hơn sự phức tạp này trong một mô hình có thể cho ta một sự ước lượng tốt hơn về thời gian được yêu cầu bởi một chương trình để hoàn thành các yêu cầu vào/ra của nó. Một mô hình phức tạp hơn nữa có thể có 1 bus, một bộ điều khiển, và rất nhiều ổ đĩa. Mô hình này có thể xử lý nhiều yêu cầu song song và có thể cho những ước lượng chính xác về thời gian đáp ứng tại nguồn tài nguyên vào/ra.

Tất cả các mô hình hệ thống đặc trưng cho sự cân bằng giữa sự gia tăng các mức độ chi tiết và sự gia tăng tính thực thi. Các hệ thống tốt hướng đến một sự cân bằng giữa những mục đích trái ngược nhau.

4.3.5 Những kết quả của mô hình

Một mô hình của một hệ thống có thể được dùng theo 2 cách riêng biệt [LaKe91]: (1) cung cấp các ước lượng về các giá trị đầu ra cho hoạt động của hệ thống trong "trạng thái vững vàng" và (2) cung cấp các ước lượng về các giá trị đầu ra cho hoạt động của hệ thống thông qua một khoảng thời gian được đặc tả (ví dụ cho một phép dịch 8 giờ). Những vấn đề về việc cung cấp các ước lương chính xác trong cả hai trường hợp đều như nhau, cũng như là các kỹ thuật để thoả mãn những vấn đề đó (Law and Kelton, 1991).

Các mô hình của những hệ thống máy tính có thể đưa ra thêm một vài vấn đề về nhiệm vụ cung cấp các ước lượng đáng tin về các biến đầu ra. Những điều này xuất phát từ thực tế rằng các hệ thống máy tính thật có thể lớn hơn và phức tạp hơn rất nhiều. Thêm vào đó, một vài thành phần quan trọng là không có thể nhìn thấy sẵn, và việc tạo ra các đặc trưng chính xác cho những cách hoạt động là rất khó. Vì thế việc xác nhận về cấu trúc và hoạt động được đặc tả trong một mô hình có thể là một tín hiệu ý nghĩa về sự phát triển của nó.
Trong một số trường hợp có thể đạt được những kết quả đầu ra cho việc kiểm tra tải làm việc thường dùng để tham số hoá mô hình của tải làm việc. Trong những trường hợp này, các kết quả từ mô hình có thể được xác nhận hoàn toàn. Trong những trường hợp khác, các nhà phát triển mô hình có thể bị bắt buộc phải xem lại thiết kế và việc bổ sung lại mô hình, nhằm kiểm tra tính xác thực của các kết quả.

4.4. Hệ thống truyền thông

Mọi dự án nhằm phát triển mô hình mô phỏng hệ thống truyền thông có thể được phân chia theo hai loại hình công việc chính: (1) xây dựng mô hình tải làm việc và (2) xây dựng mô hình hệ thống mạng.

4.4.1 Tải làm việc là gì?

Về cơ bản, mô hình tải làm việc của hệ thống truyền thông là sự mô tả về lưu lượng truyền tin trong một hệ thống truyền thông. Việc mô tả này có thể rõ ràng nếu dựa vào những nghiên cứu thông qua việc theo dõi một hệ thống thực bằng thiết bị đo, hoặc có khi chỉ mang tính thống kê nếu ngẫu nhiên đưa ra kết quả từ một nhóm các hàm phân phối xác suất. Dù ở dạng nào thì việc mô tả tải làm việc cũng phải biểu thị chính xác lưu lượng truyền tin trong một hệ thống xác định.

Việc xác nhận tải làm việc là quá trình đảm bảo được tính chính xác hợp lí trong việc lập phần mô tả tải. Thậm chí, những mô hình hệ thống mạng tin cậy nhất cũng không đem lại kết quả gì nếu chúng được thực hiện với tải làm việc không hợp lệ "nạp vào, đưa ra". Việc xây dựng và xác nhận tải làm việc là một phần rất quan trọng trong việc xây dựng mô hình mô phỏng bất kì hệ thống truyền thông nào.

Mô hình trình điều khiên tải làm việc đưa thông tin tới hệ thống truyền thông nhằm phân phối chúng trong hệ thống mạng. Vấn đề cần quan tâm là lưu lượng thông tin được truyền như thế nào trong không gian và thời gian mạng. Một lượng thông tin có thể là một phần của luồng thông tin lớn hơn với một điểm nguồn được xác định trước và một hay nhiều điểm đích. Thông tin được truyền tới chỉ 1 điểm đích được gọi " unicast traffic" trong khi thông tin được truyền tới nhiều điểm đích được gọi " multicast traffic", còn nếu truyền tới tất cả các điểm đích thì gọi là "broadcast traffic". Khi được gửi tới một hay nhiều điểm đích, một thông tin có thể ở hình thức là 1 mẩu thông tin, có thể được chia nhỏ hay được kết hợp thêm với các thông tin khác, cũng có thể những mẩu thông tin khác nhau được kết hợp hay phân tách nhiều lần trong quá trình truyền tin trong hệ thống.

Những gì xảy ra với 1 thông tin sau khi nó được truyền vào trong một hệ thống chính là công việc của mô hình hệ thống. Mô hình hệ thống mô tả loại hình thông tin cũng như thời gian tại thời điểm chúng đi vào hệ thống. Với cách hiểu như vậy, tải làm việc sẽ chịu trách nhiệm cung cấp *sự kích thích, và* hệ thống sẽ đưa ra *phản hồi*. Một phần của thiết kế quá trình mô phỏng là xác định loại hình kích thích phù hợp nhất, qua đó sẽ tạo tra sự phản hồi tương ứng.
Bởi vì các hệ thống mạng thực tế thường được cấu tạo theo các lớp cho nên đặc điểm của tải làm việc sẽ bị ảnh hưởng nhiều bởi lớp được chọn cho quá trình mô phỏng trực tiếp. Ví dụ, nếu chỉ nghiên cứu các mối liên kết vật lý thì việc kích thích sẽ được biểu hiện ở mức rất thấp của từng phân lớp (ví dụ như đơn vị dữ liệu vật lí) , trong khi đó, nếu nghiên cứu việc hoạt động của một cơ sở dữ liệu phân bố hoạt động trong hệ thống đó, thì việc kích thích sẽ ở mức cao (truy vấn cơ sở dữ liệu). Nếu lớp được chọn cho việc mô phỏng càng cao, ta càng phải biểu diễn nhiều phân lớp trong mô hình mạng, do đó, việc lựa chọn phân lớp sẽ ảnh hưởng lớn tới việc thực hiện toàn bộ mô hình.

Nếu ta chỉ tính toán đến các phân lớp thấp hơn trên lý thuyết, cần phải có kinh nghiệm để đưa ra quyết định chi tiết nào nên bỏ và phần nào nên xuất hiện trong mô hình.

Đơn vị dữ liệu logic có thể lớn hơn nhiều so với đơn vị dữ liệu vật lý được truyền qua đường liên kết, tuy nhiên cũng có thể xảy ra điều ngược lại. Lấy ví dụ về việc dữ liệu được truyền qua Ethernet và ATM: gói dữ liệu được truyền qua Ethernet có thể đạt kích thước hàng Kilobytes trong khi dữ liệu qua ATM chỉ một vài chục bytes. Mặt khác, khi so sánh lượng dữ liệu qua ATM và SONET ta thấy rằng cấu trúc SONET có thể lưu được một số lượng lớn các tế bào ATM. Nếu số lượng thông tin có thể giảm được 2 bậc về độ lớn thì điều đó rất được mong chờ nếu biết rằng việc giảm độ lớn như vậy có thể được thực hiện mà không làm mất đi sự chính xác trong quá trình mô phỏng. Việc xác định sự điều chỉnh này có thể phụ thuộc vào thuộc tính hoạt động nào của mạng là quan trọng để tính toán, do đó thật khó để đưa ra những quy luật cụ thể.

Mô hình tải làm việc tiếp nhận thông tin vào hệ thống mạng và mỗi thông tin đều phải có các thuộc tính chủ yếu sau:

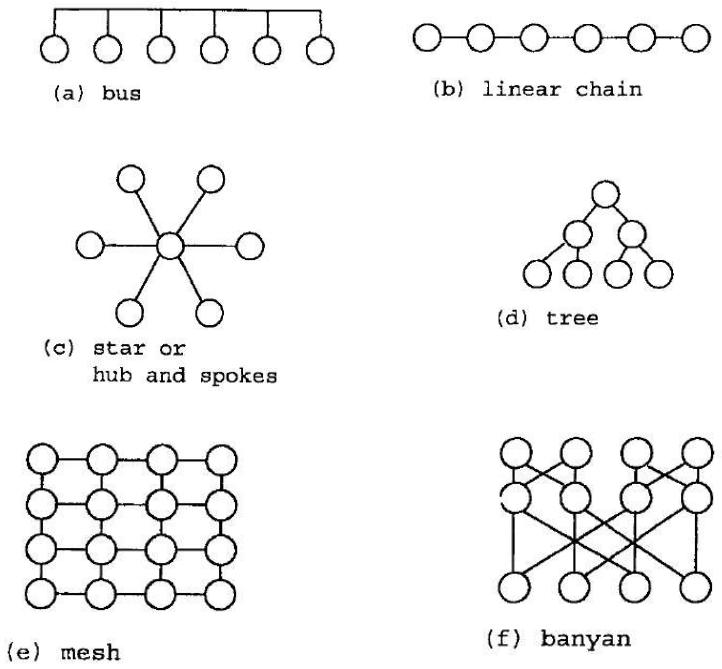
- Thời gian bắt đầu
- Điểm bắt đầu
- Điểm kết thúc
- Đặc điểm thông tin (kích thước, quyền ưu tiên.v.v.v)

Thông thường, thời gian xuất phát (và cung có thể là địa chỉ của điểm kết thúc) của lượng thông tin tiếp nhận được tính toán từ các hàm phân phối xác suất. Đúng hơn là khoảng thời gian giữa các điểm xuất phát thông tin này tại một thời điểm xác định trước được tính toán bằng các hàm phân phối xác suất. Việc lựa chọn chính xác sự phân bố giữa các khoảng thời gian xuất phát là một mặt quan trọng của việc xác nhận tải làm việc.

Bởi vì hệ thống truyền thông thường được phân tích như là các hệ thống chờ mà hầu hết đều được xử lý toán học với khoảng thời gian phân phối được tính theo hàm mũ. Việc phân phối theo hàm mũ thường được lựa chọn dù nó không hẳn là một sự lựa chọn đúng đắn. Điều này có thể ứng dụng được đối với địa điểm xuất phát thông tin là quá trình Poisson (không có nhớ) tuy nhiên lưu lượng thông tin đa phương tiện đa dạng như giọng nói và chuỗi video được nén lại sẽ chỉ ra các đặc điểm của quá trình Markov. Một quá trình Poisson và quá trình Markov *phi Poisson* với khoảng thời gian chuyển tiếp trung bình giữa các thông tin có thể mang lại các kết quả rất khác nhau về hoạt động của hệ thống mạng do lưu thông bị nghẽn tại các nút mạng.

4.4.2 Mạng và giao thức mạng

Mô hình mạng là nửa còn lại của vấn đề, nó cùng với mô hình tải cấu tạo nên mô hình của cả hệ thống truyền thông. Mạng thường được thể hiện là các biểu đồ với các nút, và liên kết. các liên kết nhiều điểm hay các bus có thể coi là các liên kết trong các mạng siêu biểu đồ. Thông thường, các nút đầu cuối là các nút nguồn và các nút đích, còn các nút khác là các nút chuyển mạch và nút định tuyến. cấu trúc biểu đồ này trở nên phức tạp hơn khi tính đến cả các giao thức mạng, đặc biệt khi tính đến việc phân lớp phần mềm như trong các cấu trúc mạng ngày nay.



Hình 4.1 Các topology thông thường của mạng lưới

Mô hình tương kết của các nút mạng và liên kết mạng cấu tạo nên topo mạng. viêc thiết kế các topo mạng phức tạp và đặc biệt bao gồm thời gian thiết kế và thời gian mô phỏng có thể sẽ rất đắt, nhưng không phải lúc nào cũng tránh đươc. Các topo mạng đơn giản và bình thường thường thấy được thể hiện trong **hình 20.1.**

Không cần thiết có sự phân biệt giữa topo mạng logic và topo mạng vật lý. Như ví dụ trong hình 20.2a, một mạng logic vòng có thể được thiết kế như một mạng vật lý vòng, chuỗi tuần tự (xen kẽ hoặc không xen kẽ) hay mạng sao. Mạng token ring phổ biến thường được xem là mạng hình sao, vì thiết kế vật lý mạng này theo kiểu vòng sẽ cồng kềnh và dễ sai sót.

Sự khác nhau giữa topo vật lý và topo logic có thể có hoặc không có ý nghĩa. Nếu sự trì hoãn tín hiệu truyền là quan trọng thì topo vật lý cần phải được chú ý trong mô hình mạng. ví dụ trong mạng ethernet thiết kế theo kiểu bus, sự tác động của các va chạm nhóm bị ảnh hưởng bởi vị trí tương đối và sự phân tách các nút trên đường truyền và bởi cả quá trì hoãn tín hiệu giữa các đầu nút.

4.4.3 Quy định dịch vụ đối với các bộ nhớ đệm, các kênh và bộ chuyển mạch.
Hoạt động của hệ thống truyền thông có thể bị ảnh hưởng rất lớn bởi việc lựa chọn các quy định dịch vụ. các quy định dịch vụ có thể thay đổi về mức độ đơn giản của việc thực hiện (phụ thuộc vào việc lựa chọn các công cụ và thư viện) và trong hoạt động tính toán trong thời gian mô phỏng. Việc cân bằng các yếu tố kỹ thuật có thể chống lại việc áp dụng các quy định dịch vụ tối ưu về mặt lý thuyết, hoặc trong thời gian mô phỏng hoặc trong việc thiết kế mạng thực tế.

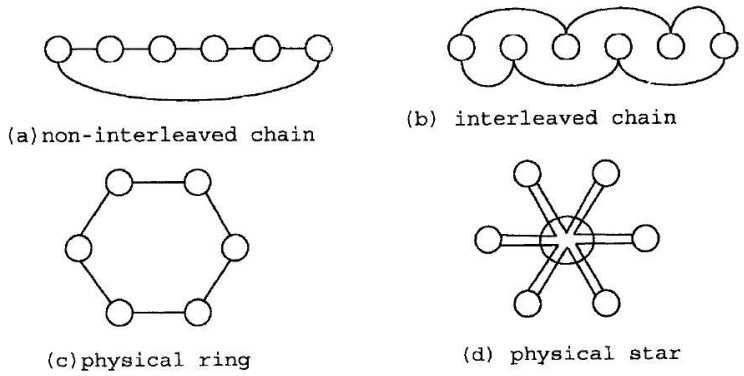


Figure 20.2 Logical versus physical network topologies.

Hình 4.2: Các topology logic và topology vật lý của mạng lưới

Một số quy định dịch vụ phổ biến có thể áp dụng cho các bộ nhớ đệm truyền thông:

- First come, first served (FCFS), cũng gọi là First in, first out (FIFO)
- Thứ tự ưu tiên, dựa trên việc ưu tiên thông điệp đơn giản hoạc trên các thuộc tính của thông điệp (vd thời gian hình thành hay một số tham số chất lượng dịch vụ (QoS)liên quan đến thông điệp, kênh hay đích)
- Thứ tự ngẫu nhiên, có thể là hoàn toàn ngẫu nhiên(phân phối đồng bộ) hoặc thiên về các thuộc tính của thông điệp hoặc thời gian chờ)

Điều quan trọng là FCFS là một trường hợp đặc biệt của thứ tự ưu tiên, quy định này ưu tiên thời gian chờ. Tương tự, thứ tự ưu tiện cũng có thể coi là một trường hợp đăc biệt của thứ tự tiền định, ngược lại với thứ tự ngẫu nhiên. Các kiến trúc sư của hệ thống truyền thông cũng quan tâm đến việc lựa chọn các thứ tự dịch vụ làm tối ưu các trị số mạng. do cả hoạt động của mạng và các chi phí cho mạng đều là các lượng đa chiều, thông số kỹ thuật của một trị số phù hợp không phải đơn giản (đơn chiều)

Trong một số trường hợp, có một số thứ tự dịch vụ được chứng minh là tối ưu cho một mục tiêu thiết kế nhất định, và thường thì người ta mong muốn mô phỏng được hoạt động của các thứ tự tối ưu và không tối ưu. Điều này đặc biệt đúng nếu như chi phí hay mức độ phức tạp của việc áp dụng thứ tự tối ưu là cao.

Đối với việc quản lý bộ nhớ đệm, chúng ta cần xem xét (1) gửi thông điệp tới bộ nhớ đệm, (2) sự ghi nhớ thông điệp tại bộ nhớ đệm, và (3) sự chuyển thông điệp ra khỏi bộ nhớ đệm (một cách bình thường hoặc không bình thường)

Việc chuyển thông điệp tới bộ nhớ đệm có thể phụ thuộc vào trạng thái hiện tại của bộ nhớ đệm (ví dụ.trọn vẹn hay không chọn vẹn, việc ưu tiên các thông điệp đang trong vùng đệm hay các thông điệp sắp được chuyển đến bộ nhớ đệm, vv) và các thuộc tính của các thông điệp sắp chuyển đến bộ nhớ đệm. Việc ghi nhớ các thông điệp có thể bao gồm việc sắp xếp lại nội dung bộ nhớ đệm dựa trên các tham số động, bao gồm thời gian chờ của các thông điệp đang trong bộ nhớ đệm. việc chuyển thông điệp ra khỏi bộ nhớ đệm có thể xuất hiện một các bình thường như khi thông điệp được thực hiện hoặc bất bình thường như khi thông điệp bị từ chối do quá hạn dịch vụ, quyền ưu tiên của thông điệp, hay bất cứ yếu tố gì.

Thông thường, các bộ nhớ đệm được sử dụng kết hợp với các bộ chuyển mạch để lưu trữ các thông điệp trong quá trình chuyển các thông điệp này từ một kênh đến tới một hoặc nhiều kênh đi. Những điều cản trở và sự phức tạp liên quan đến việc chuyển mạch thông điệp có thể có ảnh hưởng lớn đến chiến lược quản lý bộ nhớ đệm. các bộ nhớ đệm có thể được sử dụng ở các cổng đầu vào(input-bufered switch), các cổng đầu ra(output-bufered switch), cả đầu vào và đầu ra của bộ chuyển mạch, hay ở bất cứ giai đoạn trung gian nào trong chuyển mạch nhiều giai đoạn.

Phụ lục : So sánh, đánh giá và lựa chọn chương trình mô phỏng mạng
Tác giả: Tổng hợp từ nhiều nguồn
Dịch thuật: Tô Thành Công, Hà Tất Thành
Biên tập: Vũ Thúy Vân

1. Các tiêu chí đánh giá chương trình mô phỏng mạng.
Cần xem xét các đặc điểm sau đây của các chương trình mô phỏng mạng để lựa chọn được chương trình mô phỏng mạng thích hợp nhất cho các dự án nghiên cứu cá nhân.

Loại mô phỏng được hỗ trợ
Có 3 loại mô phỏng phổ biến: mô phỏng sự kiện rời rạc, mô phỏng theo vết (trace driven) và mô phỏng Monte. Một loại khác là đưa lưu lượng trực tiếp vào chương trình mô phỏng và/hoặc đưa lưu lượng từ một chương trình mô phỏng vào mạng thật. Trên thực tế, các đặc điểm này giống bộ giả lập hơn là mô phỏng.

Hỗ trợ chạy trên các nền tảng khác nhau
Sẽ hưu ích nếu có một chương trình mô phỏng chạy trên các nền tảng khác nhau, cụ thể là trên cả các hệ điều hành linux và windows. Trong trường hợp mô phỏng có tải tính toán nặng, cần quan tâm tới các khả năng xử lí đa nhiệm, song song, hay phân tán

Hỗ trợ trong việc tạo các topo mạng.
Hai loại tạo topo mạng chính là dùng các tập lệnh đặc biệt, hoặc các ngôn ngữ lập cấu hình và giao diện đồ họa. Các chương trình mô phỏng cần có khả năng tạo các topo mạng phân cấp, không phân cấp cũng như các topo mạng ngẫu nhiên.

Hỗ trợ việc tạo và quản lí các profile lưu lượng.
Các chương trình mô phỏng cần các bộ phát dữ liệu tuân theo một vài phân bố cụ thể (ví dụ Poisson hay các phân bố nhận được từ các quan sát thực tế về lưu lượng). Một chương trình mô phỏng tốt thường có một hệ thống các bộ phát lưu lượng như vậy.
Mặt khác, cũng cần phải có các công cụ nhằm mô tả sơ lược lưu lượng đã phát và tính toán các con số thống kê cần thiết để đưa ra kết luận

Hỗ trợ giám sát.
Trong quá trình mô phỏng, sẽ rất có lợi nếu giám sát hoạt động của mạng trên mỗi luồng, mỗi nút mạng hay tổng quát hơn là trên một số các tiêu chí tổng hợp. Việc giám sát có thể được trợ giúp bởi giao diện đồ họa.
Kết quả giám sát có thể ghi vào các file nhằm tạo ra các dữ liệu để so sánh về sau hay chạy lại mô phỏng phục vụ mục đích nghiên cứu các hoạt đã xảy ra.

Module cho các giao thức OSI, các mô hình di động và truyền lan sóng vô tuyến
Các module OSI rất đáng chú ý: Các giao thức định tuyến ở lớp mạng, hoạt động ổn định của lớp MAC, lới vật lí và lớp đường truyền (link). Tại lớp ứng dụng hay lớp phiên, có lẽ chỉ cần ứng dụng kiểu FTP là đủ. Tại lớp vận chuyển, cần hỗ trợ TCP và UDP.
Sẽ có lợi nếu chương trình mô phỏng có module di động bởi nghiên cứu so sánh sẽ dễ hơn do các nhà nghiên cứu khác hiện cũng đang có xu hướng sử dụng module này. Truyền lan sóng vô tuyến trong các chương trình mô phỏng thường được thực hiện trực tiếp dựa trên các kết quả chuẩn từ các nghiên cứu về điện từ. Một mô hình truyền lan sóng vô tuyến chính xác cần xem xét đến yếu tố hình thái và những đặc tính vật lý của môi trường.

Các vấn đề về sự cân đối, khả năng mở rộng, khả năng điều chỉnh và tính mềm dẻo/ linh hoạt.

Các chương trình mô phỏng phải có một kiến trúc module thực sự mở mà trong đó, các thành phần có thể dễ dàng chuyển đổi giữa hai trạng thái bật tắt, các mô hình hiện thời có thể được thay thế hay chỉnh sửa, và kiến trúc module này có thể bổ sung cả các mô hình mới.
Khả năng mở rộng cũng là một vấn đề quan trọng. Về nguyên lí, tất cả các chương trình mô phỏng đều cho phép thêm nhiều nút mạng để tăng tải tính toán của các mẫu lưu lượng và các mặt động của mạng (các thành phần di động, các nút mạng đang xuất hiện/ biến mất, các đường bị lỗi vân vân).
Tuy nhiên, trong thực tế, chỉ một vài chương trình mô phỏng là cân đối tốt giữa các chỉ tiêu trên, do đó, cần nhận ra các hạn chế của các chương trình mô phỏng này.

Tính khả dụng rộng rãi
Thiết kế phần mềm của một chương trình mô phỏng cũng như là các công cụ lập trình dùng trong chương trình mô phỏng đó có tác động lớn đến tính khả dụng của nó.
Lợi ích của giao diện đồ họa có thể tăng tốc thao tác trên chương trình mô phỏng và do đó được coi là một ưu điểm.
Cuối cùng, chất lượng chung của các tài liệu sẵn có và hỗ trợ về kĩ thuật cũng là các yếu tố quan trọng để có thể học cách sử dụng và điều khiển chương trình mô phỏng đó một cách hiệu quả và nhanh chóng.

Mức độ cộng đồng mạng chấp nhận một chương trình mô phỏng
Việc chọn lựa một chương trình mô phỏng được sử dụng rộng rãi và được chấp nhận bởi cộng đồng mạng cho phép tạo ra các kết quả dễ so sánh với các kết quả được trình bày trong các tài liệu đã có, và ở khía cạnh nào đó, các kết quả này cũng dễ được giới khoa học chấp nhận hơn.

Loại bản quyền phần mềm
Rõ ràng là theo lí thuyết thì các chương trình mô phỏng miễn phí luôn tiện lợi hơn. Tuy nhiên cũng cần xem xét, lựa chọn các chương trình mô phỏng thương mại nếu chúng có nhiều ưu điểm chiến lược hơn các chương trình mô phỏng miễn phí.
OPNET, GloMoSim/QualNet, NS-2 và OMNeT++ được xem như các ứng cử viên sáng giá nhất. Tiếp theo đây, chúng ta sẽ thảo luận về các đặc điểm chính của mỗi phần mềm trên ở cả mức độ tổng quát và trên từng quan điểm về các tiêu chí lựa chọn ở bảng 1.

2- Giới thiệu các chương trình mô phỏng mạng
2.1 OPNET

OPNET (Optimized Network Engineering Tools) là công cụ "có phí" của tập đoàn công nghệ OPNET [2] dùng cho việc mô hình hóa và mô phỏng các mạng thông tin, các thiết bị và các giao thức.
OPNET là chương trình mô phỏng trên nền Windows được sử dụng rộng rãi. Nó được xây dựng dựa trên ngôn ngữ C++ và cung cấp môi trường ảo cho việc mô hình hóa, phân tích và dự đoán hiệu năng mạng, giúp mô hình hóa chính xác các ứng dụng, các máy chủ và nhiều công nghệ mạng.. Các giao thức và thiết bị mới thường xuyên được cập nhật nhằm theo kịp xu hướng phát triển nhanh chóng của công nghệ mạng.
OPNET được sử dụng bơi nhiều các tổ chức thương mại, các tổ chức chính phủ và các trường đại học trên toàn thế giới. Có nhiều các tính năng đa dạng cho người dùng OPNET. Bao gồm:
- Tạo và chỉnh sửa các mạng và các nút mạng.
- Tạo và chỉnh sửa các quá trình đang chạy trên các nút đó.
- Phân tích kết quả mô phỏng và tạo biểu đồ hiệu năng.
- Định nghĩa các quá trình toán học trong việc dùng các công cụ phân tích.

Các tính năng này làm cho OPNET rất linh hoạt và cung cấp khả năng mô phỏng hầu như mọi loại hình mạng truyền thông. Tạo mô phỏng topo mạng đơn giản là dễ dàng với việc sử dụng các thành phần kéo & thả và các cài đặt được định trước. Có thể quản lí được một mạng với hàng trăm nút

mang. Tuy nhiên, với người mới bắt đầu, học để sử dụng triệt để OPNET nhằm triển khai một giao thức mới có đôi chút khó khăn; người dùng cần làm quen với phương pháp và ngôn ngữ hướng đối tượng như C++, cũng như là kiến thức cơ bản về mạng.

Phương pháp mô hình hóa trong OPNET được tổ chức theo cấu trúc phân cấp. Ở mức thấp nhất, là mức độ hoàn toàn có thể điều chỉnh được, các mô hình Process được xây dựng như các FSM (Finite State Machine - cơ cấu có số trạng thái giới hạn). Trạng thái và các chuyển dịch có thể được cụ thể hóa một cách sinh động bằng cách sử dụng hình vẽ các lưu đồ chuyển dịch trạng thái, trong khi các điều kiện xác định diễn biến trong mỗi trạng thái được lập trình bởi Proto-C, một ngôn ngữ tương tự C.

Các mô hình Process, và các module đi kèm trong OPNET (các module nguồn và đích, các bộ phát lưu lượng, các hàng đợi, các module vô tuyến...) sau đó được cấu hình với các bảng lựa chọn và được tổ chức thành các lưu đồ dòng dữ liệu thể hiện các nút mạng bằng cách dùng bộ biên tập nút kiểu đồ họa (graphical Node Editor). Trình biên tập mạng đồ họa, các nút và link được lựa chọn để xây dựng topo mạng truyền thông.

The Modeling libraries are included with OPNET Modeler and OPNET Planner and contain protocols and environments (e.g., ATM, TCP, IP, Frame Relay, FDDI, Ethernet, IEEE 802.11, support for wireless), link models such as point-to-point and bus, queuing service disciplines such as First-in-First-Out (FIFO), Last-In-First-Out (LIFO), priority non-preemptive queuing, shortest first job, round-robin or preempt and resume.

Các thư viện mô hình hóa được kèm trong OPNET Modeler và OPNET Planner, có chứa các giao thức và môi trường (Ví dụ: ATM, TCP,IP Frame Relay, FDDI, Ethernet, IEEE 820.11 trong thông tin không dây), các mô hình đường truyền như điểm - điểm, bus, các quy tắc dịch vụ hàng đợi như FIFO, LIFO, Ưu tiền không chèn, round robin v…v

2.2 GloMoSim/QuaNet

GloMoSim [3] là một thư viện mô phỏng có khả năng mở rộng được thiết kế tại phòng thí nghiệm máy tính UCLA để hỗ trợ việc nghiên cứu các mô hình mạng quy mô lớn có thể lớn tới hàng triệu nút, bằng cách sử dụng đồng thời quá trình thực hiện phân phối and/or trên một tập hợp các máy tính song song khác nha (có nghĩa là cả 2 cùng phân phối hoặc chia sẻ bộ nhớ). Nó được thiết kế cho các mạng vô tuyến và hệ thống mạng diện rộng. Nhưng tại thời điểm này nó chỉ hỗ trợ cho các giao thức trong mạng vô tuyến. GloMoSim là một thư viện cho ngôn ngữ mô phỏng theo sự kiện rời rạc song song dựa trên C – gọi là PARSEC [4]. Một đặc trưng quan trọng và nổi bật nhất của PARSEC là khả năng thực hiện một mô hình mô phỏng sự kiện rời rạc bằng cách mô phỏng một vài giao thức không đồng bộ khác nhau trên một loạt các kiến trúc song song.

GloMoSim được xây dựng theo hướng tiếp cận các tầng đã được phân chia trong mô hình OSI. Thông thường, API giữa hai mô hình thuộc chồng giao thức gần nhau đã được định nghĩa trước để hỗ trợ cho thành phần cấu trúc của chúng. Các API chỉ rõ việc trao đổi và cung cấp các thông số giữa các tầng liền kề nhau. Nó cho phép tích hợp một cách nhanh chóng các mô hình đã được phát triển tại các tầng khác nhau bởi những người khác nhau. Hiện nay, các mã hoạt động thực tế có thể được tích hợp một các dễ dàng vào GloMoSim.. Nếu tất các các mô hình giao thức đều tuân thủ một cách chặt chẽ theo các API đã được định nghĩa cho từng tầng thì nó có thể hoán đổi các mô hình giao thức trong một tần nào đó mà không phải chỉnh sửa các mô hình còn lại trong ngăn xếp. Đó là việc thực hiện các modul có khả năng so sánh tính nhất quán của các giao thức và mức độ khác nhau chi tiết tại một tầng nào đó.

Để thực hiện một chương trình GloMoSim, cần phải có trình biên dịch PARSEC. Để phát triển các giao thức mới trong GloMoSim, người sử dụng nên có những hiểu biết rõ ràng về PARSEC, nhưng

người dùng cũng không cần phải hiểu biết ở mức độ chuyên môn sâu. Đối với đa số các giao thức nó khả năng thêm vào một vài hàm PARSEC được viết hoàn toàn bằng mã C. Một số các giao thức đã được phát triểncho từng lớp, các giao thức này hoặc các tầng có thể được triển khai với mức độ chi tiết khác nhau và đặc điểm này là có thể thực hiện được trong GloMoSim. Telnet ftp, CBR, HTTP đã sao chép các file hệ thống có liên quan đến các úng dụng . Trong khi đó các mô hình được đề cập đến trong tầng truyền tải (transport layer) là có khả năng thực hiện được, ví dụ như: TCP (FreeBSD), UDP, TCP (tahoe). Đối với định tuyến theo kiểu unicast thì các thuật toán có thể sử dụng được là: AODV, Bellman-ford, DSR, Fisheye, Flooding, LAR, NS DSDV, OSPF, WRP/ Các mô hình có thể sử dụng được cho tầng MAC là CSMA, IEEE 802.11, FAMA, MACA. Sóng vô tuyến chung sử dụng ăng ten đa hướng có hoặc không có khả năng chiếm giữ. Sự truyền sóng vô tuyến có thể được mô phỏng thông qua các vùng trống, mô hình Rayleigh, mô hình Ricean hoặc mô hình SIRCIM (bao gồm các vật cản, các tình huống trong nhà).. Mô hình di động bao gồm nhiều loại hình như basic Random waypoint, Random drunken, ECRV, BBN, Pathloss matrix, và di động nhóm.

GloMoSim có kiến trúc phân lớp. Quá trình mô phỏng là sự tập hợp các nút mạng, với mỗi nút mạng là các tham số giao thức và số liệu thống kê. Đổi lại, mỗi lớp là một đối tượng với các biến và cấu trúc của chúng. Các bản tin có thể trao đổi chéo gữa các nút và các tầng với bất kỳ mức yêu cầu nào của nhóm và mức độ chi tiết. Sự đồng bộ giữa các sự kiện được đảm bảo bởi việc tự lập lịch (bộ định thời – timer) của các bản tin. Thư viện với tập hợp các hàm đơn giản được xây dựng trên các API chuẩn, dùng để tạo ra, truyền tải và thao tác thông tin. Các kiểu sự kiện mới như là : các bản tin. Các nút, các lớp có thể được tạo ra một cách khác dễ dàng.

Kiến trúc bên trong của GloMoSim đã được thiết kế để phát triển môi trường mô phỏng theo modul có khả năng mở rộng mạng với hàng nghìn/hàng triệu nút khác nhau và dễ dàng chuyển đổi giữa môi trường tính toán song song/phân phối. Các yêu cầu đối với tính chất mở rộng và tính modul hóa làm cho việc thiết kế thư viện gặp khó khăn. GloMoSim cho rằng hệ thống mạng được phân tách thành các nhóm phân vùng, một thực thể đơn được định nghĩa để mô phỏng một lớp đơn trong toàn bộ chồng giao thức của tất cả các nút trong hệ thống mạng thuộc cùng một phân vùng. Sự tương tác giữa các thực thể tuân theo các API tương ứng.

QualNet: Phiên bản thương mại của GloMoSim

GloMoSim không phải là miễn phí, nhưng nó được sử dụng tự do mà không cần trả phí cho mục đích giáo dục, nghiên cứu hoặc các tổ chức phi lợi nhuận. Tuy nhiên các tài liệu được gửi cùng với GloMoSim là khá ít (thậm chí không có một tài liệu hướng dẫn cho người sử dụng). Tập hợp các công cụ có khả năng tăng tốc để tạo ra các topo, để giám sát hoạt động hệ thống, để phân tích kết quả mô phỏng bổ xung hay tổng quát hơn là để thiết kế các đặc đặc điểm cho sự tương tác đối với các môi trường mô phỏng cũng rất hạn chế.

QualNet bổ sung phần nào thiếu sót của GloMoSim. QualNet[5] là một sản phẩm thương mại có nguồn gốc từ Scalable Network Technologies, cũng là nơi bắt nguồn của GloMoSim. Tuy nhiên, QualNet được nạp thêm vào nhiều tính năng được đánh giá cao hơn so với GloMoSim. QualNet có một bộ mở rộng các mô hình và các giao thức thực thi chính xác cho cả mạng hữu tuyến và mạng vô tuyến/ không dây (nội hạt, vệ tinh, mạng ad hoc, mạng cellular) , cũng như số lượng lớn các tư liệu và hỗ trợ kỹ thuật. Ba thư viện có thể dùng được là:

+) Một thư viện chuẩn cung cấp hầu hết các mô hình và các giao thức cần thiết giành cho cả hai hướng nghiên cứu và hướng thương mại, hoạt động trong mạng hữu tuyến và mạng không dây.

+) Một thư viện MANET cung cấp thêm các thành phần rất cụ thể cho các mạng ad hoc khác nhau hơn là các thư viện chuẩn đã tồn tại khác .

+) Một thư viện QoS bao gồm các giao thức chuyên dùng cho chất lượng dịch vụ. QualNet cũng bao gồm mô hình độ cao số hoá - DEM (Digital Elevation Model) để tạo các nút và các sóng vô tuyến chuyển động trong những địa hình không bằng phẳng với các đặc tính bức xạ radio. Mô phỏng, hay bản thân chương trình mô phỏng được thiết kế để cung cấp các mô hình mạng có độ trung thực cao với hàng vạn nút có lưu lượng và tính di động cao. Thêm vào đó QualNet đưa ra những phát triển song song đối với việc quản lý và thực thi song song.

2.3 OMNeT++

OMNet++ là một mã nguồn mở, thành phần cơ bản của chương trình mô phỏng này được xây dựng trên nền tảng C++. Nó cung cấp một thư viện mô phỏng C++ và hỗ trợ giao diện đồ hoạ (cho phép chỉnh sửa mạng bằng hình vẽ, hỗ trợ hình ảnh động).

Chương trình mô phỏng này có thể được sử dụng để: thực hiện mô phỏng lưu lượng trong mạng viễn thông, tạo mô hình các giao thức, xây dựng mô hình các hàng đợi trong mạng, xây dựng các bộ vi xử lý đa nhiệm và các hệ thống phân tán khác, kiểm tra lại kiến trúc phần cứng, đánh giá khía cạnh về hiệu suất của các hệ thống phần mềm phúc tạp , và nói chung là việc tạo mô hình bất kì một hệ thống nào khác đều gắn với các thành phần tích cực mà chúng giao tiếp với nhau bằng việc gửi các bản tin. Về môt số mặt thì OMNet++ khác so với các phần mềm mô phỏng đã nói trên. OMNet++ không được thiết kế chuyên cho mạng viễn thông, mà nó tổng quát hơn nhiều. Thực tế là OMNet++ vẫn là một phần mềm mới (được phát triển từ năm 1998), điều đó cho thấy OMNet++ có ít các modul và các giao thức (về mạng) đi kèm hơn các chương trình mô phỏng đã nói trước đây khá nhiều

Trên một khía cạnh khác, OMNeT++ đã được thiết kế cẩn trên phương diện phần mềm, kết quả là được một sản phẩm được tổ chức tốt hơn, mêm dẻo và dễ dàng sử dụng hơn so với các phần mềm mô phỏng khác, mà các phần mềm này đều dựa trên các sản phẩm phần mềm và các khái niệm thiết kế cũ.

Một mô hình OMNeT++ của một hệ thống (vd: một mạng…) bao gồm các modul phân cấp lồng nhau. Độ sâu của việc lồng ghép các modul này là không có giới hạn, nó cho phép người sử dụng ánh xạ đế cấu trúc logic trong thực tế bằng các mô hình cấu trúc trong mô phỏng. Các modul giao tiếp với nhau thông qua việc truyền những thông báo. Các thông báo có thể chứa các cấu trúc dữ liệu phức tạp tùy ý. Các modul có thể gửi các thông báo trực tiếp đến đích hoặc dọc theo một đường dẫn đã được định trước thông qua các cổng hoặc các kết nối, mà các đường này đã được gán cho các đặc tính như băng thông, độ trễ và tỉ lệ lỗi. Các module có thể có các thông số mà được dùng để tùy chỉnh hoạt động của nó, điều này tạo nên tính linh hoạt trong các topo mạng và khi đó các module giao tiếp với nhau giống như việc chia sẻ giữa các biến.

Các modul thấp nhất trong module phân cấp thì được cung cấp bởi người sử dụng, các modul đó chứa các thuật toán. Trong quá trình thực hiện mô phỏng, các modul đơn này sẽ chạy song song với các modul khác, bởi vì chúng đã được thi hành như các thường trình phụ (đôi khi gọi là quá trình nhẹ tải)

Để viết một vài module đơn, đòi hỏi người sử dụng phải biết lập trình C++. OMNeT++ có định hướng thiết kế hướng đối tượng chặt chẽ. Người dùng có thể dùng các khái niệm của việc lập trình hướng đối tượng một cách tự do để mở rộng chức năng của chương trình mô phỏng

Các chương trình mô phỏng trong OMNeT++ có thể có đặc tính khác nhau về giao diện người người dùng cho các mục đích khác nhau như: gỡ rối, thể hiện và thực thi các khối công việc. Giao diện đồ họa giúp cho người sử dụng có thể nhìn thấy được bên trong của mô hình, cho phép họ có thể bắt đầu/ ngừng thực hiện mô phỏng và can thiệp vào bên trong mô hình bằng cách thay đổi các

biến hoặc các đối tượng. Thư viện đồ họa được liên kết với các chương trình gỡ rối/bám vết trở thành các chương trình mô phỏng có thể thực thi được. Lựa chọn này cho phép tất cả người dùng – người đã tạo ra đối tượng đó có thể nhìn thấy được đối tượng (và có thể sửa đổi nó) trong giao diện đồ họa thông qua cửa sổ kiểm soát.

OMNeT++ cũng hỗ trợ mô phỏng song song thông qua việc sử dụng các thư viện giao tiếp MPI hoặc PVM3. Trong trường hợp tổng quát, OMNeT++ giống với PARSEC (Parallel Simulation Environment for Complex Systems), nhưng nó mềm dẻo hơn, linh hoạt hơn và môi trường phong phú hơn.

OMNeT++ cùng với một tập hợp rộng lớn các lớp bao hàm đại diện cho cấu trúc dữ liệu (vd: hàng đợi, mảng…), các lớp thu thập dữ liệu, lớp xác xuất và lớp thống kê (vd: Các biểu đồ cột, các phân phối theo luật số mũ), các lớp phát hiện nhanh và các lớp phát hiện kết quả chính xác. Các file đầu ra là các file text mà sau một vài quá trình xử lí đơn giản, có thể được chuyển thành các tập tin mang tính thống kê và toán học chuẩn.

OMNeT++ cùng với bộ giao thức internet bao gồm: IP, TCP, UDP, RTP và một vài giao thức hỗ trợ cho chất lượng dịch vụ cơ bản, trong khi đó đề liên quan đến mạng vô tuyến, đây là một modul mô phỏng các tầng thấp hơn của mạng GSM, một vài modul mở đầu dành cho các nút có tính chuyển động, sự lan truyền sóng vô tuyến, các thuật toán định tuyến (AODV) và sự truyền thông vô tuyến nói chung.

2.4 NS-2

Chương trình mô phỏng mạng NS [7] có nguồn gốc từ U.C.Berkely/LBNL, NS là một chương trình mô phỏng sự kiện rời rạc hướng đối tượng có mục tiêu tiến hành nghiên cứu hoạt động mạng và là một phần mềm miễn phí. NS-2 được sử dụng rộng rãi trong cộng đồng nghiên cứu hoạt động mạng, dựa trên sự thừa nhận rộng rãi NS-2 như là một công cụ thử nghiệm cho các ý tưởng mới, các giao thức, và các thuật toán phân phối. NS2 luôn luôn nhận được bao gồm sự đóng góp quan trọng từ các nhà nghiên cứu.

Ngày nay, NS-2 thích hợp với mạng chuyển mạch gói và mạng vô tuyến (ad hoc, local and satellite), và được sử dụng chủ yếu cho các việc mô phỏng quy mô nhỏ của hàng đợi, thuật toán định tuyến, các giao thức truyền tải, điều khiển tắc nghẽn và một vài các công việc có liên qua đến đa đường (multicast). Nó cung cấp các hỗ trợ quan trọng dành cho mô phỏng TCP, định tuyến và các giao thức đa đường (multicast) thông qua các mạng hữu tuyến và mạng vô tuyến

NS-2 thích hợp không chỉ hợp cho việc mô phỏng mà cho cả sự giả lập, điều này có nghĩa là nó có thể đưa chương trình mô phỏng vào trong mạng thực tế. Những đối tượng trong chương trình mô phỏng có khả năng đưa các lưu lượng thực vào trong chương trình mô phỏng và đưa một phần lưu lượng trong chương trình mô phỏng vào trong mạng thực tế.

Nhưng tiếc rằng, với kiến trúc phần mềm của nó thì việc thêm các thành phần mới hoặc chỉnh sửa những điểm hạn chế không dễ dàng. Điều này có nghĩa là về khả năng thực hiện kiểm tra thuật toán hay các kịch bản mô phỏng mới thì NS-2 yếu thế so với các phần mềm mô phỏng khác. Ngoài ra NS-2 không cân đối (scale) tốt đối với số lượng nút mạng và theo như các báo cáo thì NS-2 tính toán khá chậm.

Kiến trúc của NS-2 bám sát mô hình OSI. Mô hình mạng này đặc trưng cho quan hệ của các yếu tố mạng. Bao gồm các nút và các liên kết. Các bộ phát lưu lượng loại đơn lưu lượng hay đa lưu lượng bao gồm các bộ phát kiểu thống kê và cá kiểu điển hình khác như FTP và telnet có thể được gán cho mọi nút. Thêm vào đó, hoạt động mạng và giao thức truyền tải được mô phỏng bởi thao tác gán các tác nhân thích hợp vào các nút có liên quan. Mã nguồn của ns 2 được phân chia thành C++(đóng vai trò lõi chương trình) và OTcL, đối tượng mở rộng của MIT với TcL, dành cho việc cấu hình và

các đoạn mã mô phỏng

Sự kết hợp của hai ngôn ngữ này tạo ra sự thỏa hiệp giữa tính hiệu quả và tính dễ dàng sử dụng. Gói phần mềm này cung cấp một trình biên dịch lớp phân cấp của các đối tượng viết bằng C++ và việc dịch các lớp phân cấp của các đối tượng này được viết bằng TCL, sau đó được thống nhất thành một trình biên dịch duy nhất. Người sử dụng tạo ra các đối tượng mới này bằng bộ biên dịch TCL. Các đối tương mới này được ánh xạ bằng quá trình trao đổi các đối tương trong trình biên dịch phân cấp. Các chỉ thị TCL được cung cấp một cách linh hoạt và có quyền hạn mạnh mẽ thông qua việc mô phỏng(bắt đầu và dừng các sự kiện, sự cố trong mạng, quá trình tập hợp số liệu và cấu hình mạng). Bộ biên dịch Tcl còn cung cấp các câu lệnh để tạo các liên kết và các nút trong các topo mạng, và kết hợp các agents với các nút.

Việc thực hiện và mô phỏng NS2 bao gồm 4 bước:

(1) Thực hiện các giao thức bằng cách thêm một tổ hợp giữa mã C++ và mã Tcl vào trong mã nguồn của NS-2.

(2) Mô tả việc mô phỏng bằng kịch bản OTcl;

(3) chạy chương trình mô phỏng và

(4) phân tích các tệp tin bám vết đã được sinh ra. Việc thực hiện một giao thức mới đòi hỏi phải bổ xung thêm mã C++ cho chức năng của các giao thức này, điều này tương tự như việc cập nhật thêm các từ khóa cho các file cấu hình NS-2 Otcl một cách hợp lệ dành cho NS-2, điều này giúp nhận biết ra các giao thức mới và các tham số mặc định của nó. Mã C++ cũng miêu tả các tham số và các phương thức, điều này tạo nên tính sẵn sàng thực thi được khi thực hiện kịch bản OTcl.

Quá trình mô phỏng được cấu hình, điều khiển và vận hành thông qua việc sử dụng giao diện đồ họa được cung cấp bởi lớp mô phỏng OTcl. Lớp này cung cấp các thủ tục để tạo và quản lý các topo, khởi tạo định dạng các gói và lựa chọn lịch làm việc. Nó lưu trữ các chỉ dẫn bên trong mỗi phần tử của topo. Người sử dụng tạo ra topo bằng các sử dụng OTcl thông qua việc sử dụng nút các lớp độc lập và liên kết, các lớp độc lập và liên kết này đưa ra một vài bước khởi đầu đơn giản.

Các topo mạng tùy ý, gồm có các bộ định tuyến, các liên kết và kênh truyền chung, có thể được xác định bằng cách liệt kê các nút mạng và các đường nối các nút trong một tập tin topo hoặc sử dụng một vài bộ phát/tạo nút sinh của mạng đã được xây dựng cho NS-2. (Tiers và GT-ITM)

Để thu thập dữ liệu đầu ra hay dữ liệu vết trong một chương trình mô phỏng, NS-2 sử dụng, cả hai loại thông tin vết là các bản ghi của mỗi gói tin khi nó tới, đi, hoặc bị loại khỏi đường truyền hoặc hàng đợi, và thông tin giám sát - là các số đếm bản ghi của nhiều chỉ số đáng chú ý như số gói tin và byte tới, đi v…v.. Chúng có thể được kết hợp lại với cả gói tin và luồng. Chương trình NAM là một Tcl/Tk được dựa trên công cụ hoạt hình động, nó có thể sử dụng cho việc quan sát các tập tin ghi sự kiện của quá trình xử lý bổ xung, phân tích và thực hiện lại của các quá trình mô phỏng (thậm chí NAM có thể sử dụng với mọi chương trình mô phỏng miễn là các định dạng dữ liệu c được quan tâm).

NS-2 là phần mềm mô phỏng phổ biến nhất được sử dụng cho việc nghiên cứu trong các lĩnh vực về mạng Ad hoc. NS-2 có đầy đủ các trang thiết bị cho các giao thức, các mô hình, các thuật toán, các công cụ thêm vào, và nó còn miễn phí. Bởi vậy, xét về tính chấp nhận khoa học, về số lượng các công cụ/ modul và về giá thành thì NS2 có lẽ là một lựa chọn lý tưởng.

3. Các gợi ý cho sự lựa chọn chương trình mô phỏng mạng thích hợp

Có một điều chắc chắn rằng, không có chương trình mô phỏng nào đã được xem xét ở trên là thực sự đáp ứng được toàn bộ các yêu cầu đặc trưng phù hợp với các kế hoạch và các mục tiêu riêng lẻ. Tuy nhiên, chương trình mô phỏng tốt nhất là chương trình có khả năng hài hòa các khía cạnh như

số các thành phần được xây dựng từ trước, khả năng modul hóa, khả năng mở rộng, và khả năng chỉnh sửa, giá thành ...vv.

Cụ thể, một chương trình mô phỏng tốt thường đi kèm với:

1 Một tập hợp lớn các mô hình, giao thức và thuật toán được xây dựng từ trước

2 Một mức độ thừa nhận cao từ cộng đồng khoa học

3 Có khả năng cân đối (scaleability) tốt

4 Thiết kế phần mềm cũng phải tốt và có tính modul cao

5 Mức độ thỏa đáng của tính tiện lợi, tính có thể sửa đổi được và tính mở rộng

6 Sự tiên tiến về đồ hoạ và các công cụ toán học dành cho việc xây dựng thử nghiệm, giám sát và xử lý bổ xung

7 Tài liệu hướng dẫn cụ thể

8 Khả năng thực hiện song song và/hoặc phân tán

9 Khả năng xác định một mô hình 3D thực tế cho môi trường

10 Chi phí hợp lí trong các trường hợp ngân sách bị giới hạn

Bảng 2. So sánh định tính của OPNET, QualNet, OMNeT++ và NS-2

Đặc điểm của các chương trình mô phỏng mạng	**OPNET**	**QualNet**	**OMNET++**	**NS2**
Các kiểu mô phỏng được hỗ trợ	Sự kiện rời rạc	Sự kiện rời rạc	Sự kiện rời rạc	Sự kiện rời rạc
Nền tảng tính toán	Song song/phân tán	Song song / phân tán	Song song	Phân tán
Các topo	Cấu trúc phân cấp	Cấu trúc phân cấp	Cấu trúc phân cấp	Cấu trúc thống nhất
Công cụ định nghĩa các topo	Đồ họa	Đồ họa	Đồ họa	Kịch bản cấu hình
Bộ sinh dữ liệu lưu lượng	Có	Có	Có	Có
Quá trình tạo lưu lượng	Có	Có	Có	Có
Hỗ trợ giám sát	Đồ họa	Đồ họa	Đồ họa	Đồ họa
Các modul dành cho các tầng trong mô hình OSI	Có	Có	Có	Có
Tính linh hoạt của các mô hình	Có	Có	Có	Có
Mô hình dành cho sự truyền sóng	Có	Có	Có	không
Tính có thể sửa đổi và mở rộng	Có	Có	Có	Khó thay đổi
Độ co giãn	**Có**	**Hỗ trợ thay đổi tỉ lệ lớn**	**Có**	**Khó thay đổi**
Tính dễ sử dụng	Khó với những người mới bắt đầu		Bình thường	Khó với những người mới bắt đầu
Sự thừa nhận của khoa học	Được thừa nhận	Được thừa nhận	Được thừa nhận	Được thừa nhận
Loại giấy phép của phần mềm	Thu phí	Thu phí	Miễn phí	Miễn phí

Tài liệu tham khảo

[1] Gianni A. Di Caro, "Analysis of simulation environments for mobile ad hoc networks"
[2] OPNET Simulator, http://www.opnet.com
[3] GloMoSim, http://pcl.cs.ucla.edu/projects/glomosim/
[4] Parallel Simulation Environment for Complex Systems (PARSEC), http://pcl.cs.ucla.edu/projects/parsec/
[5] QualNet Simulator, http://www.scalable-networks.com
[6] OMNET++, http://www.omnetpp.org/ …
[7] Network Simulator, http://www.isi.edu/nsnam/ns/