

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
KHOA CÔNG NGHỆ THÔNG TIN 1**



Chủ biên: PGS. TS. Phạm Văn Cường  
Biên soạn: ThS. Nguyễn Xuân Anh

**GIÁO TRÌNH  
CÁC HỆ THỐNG PHÂN TÁN**  
(Dùng cho đề cương INT 1405)

Hà nội 02/2025

## LỜI GIỚI THIỆU

Hệ thống phân tán đang ngày càng phổ biến trên thế giới, có thể nói hầu hết các hệ thống thông tin đã chuyển sang cách tổ chức phân tán. Ngay từ khi thành lập khoa Công nghệ thông tin, môn học “Các hệ thống phân tán” đã được đưa vào chương trình đào tạo, tuy nhiên học liệu cho sinh viên mới chỉ dừng lại ở mức bài giảng. Sau nhiều lần chỉnh sửa, đây là lần đầu tiên Giáo trình các hệ thống phân tán được viết với kinh nghiệm giảng dạy và phát triển các hệ thống trong thực tế. Nội dung giáo trình được chia làm 10 chương, bắt đầu từ chương mở đầu giới thiệu chung về hệ thống phân tán và tóm chương còn lại trình bày tóm vấn đề cốt lõi cần giải quyết khi xây dựng hệ thống phân tán, chương cuối cùng đề cập đến một số cách tiếp cận xây dựng hệ thống.

Giáo trình này đã tham khảo hai tài liệu chính là *Distributed Systems: Principles and Paradigms*" của hai tác giả A. S. Tanenbaum, M. V. Steen và "*Distributed systems: Concept and Design*" của các tác giả G. Coulouris, J. Dollimore, T. Kinberg, G. Blair, cùng với đó là những bài báo về những vấn đề trong hệ thống phân tán. Nội dung trong giáo trình cũng đã bổ sung nhiều kinh nghiệm thực tiễn và những ví dụ của tác khi xây dựng các hệ thống phân tán như hệ thống thanh toán liên ngân hàng, hệ thống quản lý mạng viễn thông, hệ thống đào tạo trực tuyến.

PGS. TS. Phạm Văn Cường

## MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN	18
1.1 Định nghĩa hệ thống phân tán	18
1.2 Quan điểm về phân cứng	20
1.2.1 Hệ thống nhiều bộ vi xử lý	21
1.2.2 Hệ thống nhiều máy tính	22
1.3 Quan điểm về phần mềm	23
1.3.1 Hệ điều hành	23
1.3.2 Phần mềm ứng dụng	25
1.4 Phân loại hệ thống phân tán	25
1.4.1 Các hệ thống điện toán phân tán	26
1.4.1.1 Hệ thống điện toán cụm	26
1.4.1.2 Hệ thống điện toán lưới	27
1.4.2 Hệ thống thông tin phân tán	28
1.4.3 Hệ thống lan tỏa phân tán	29
1.4.4 Điện toán đám mây	31
1.5 Mục tiêu hệ thống phân tán	31
1.5.1 Tính sẵn sàng	32
1.5.2 Tính trong suốt	33
1.5.2.1 Phân loại tính trong suốt	33
1.5.2.2 Mức độ trong suốt	35
1.5.3 Tính mở của hệ thống	35
1.5.4 Qui mô hệ thống	37
1.5.4.1 Vấn đề hiệu năng khi mở rộng qui mô hệ thống	37
1.5.4.2 Các kỹ thuật xử lý trong hệ thống qui mô lớn	39
1.5.5 Hệ thống phân tán và mạng máy tính	41
1.6 Kiến trúc và mô hình hệ thống phân tán	42
1.6.1 Kiến trúc hệ thống phân tán	42
1.6.1.1 Kiến trúc phân cấp	42
1.6.1.2 Kiến trúc ngang hàng	43
1.6.1.3 Kiến trúc lai ghép	45
1.6.2 Mô hình hệ thống phân tán	45
1.6.2.1 Mô hình phân tầng	46
1.6.2.2 Mô hình đối tượng phân tán	49
1.6.2.3 Mô hình kênh sự kiện	51
1.6.2.4 Mô hình dữ liệu tập trung	52
1.6.3 Kiến trúc hệ thống và phần mềm trung gian	52
1.6.4 Tự quản lý trong các hệ thống phân tán	53
CHƯƠNG 2: TRAO ĐỔI THÔNG TIN TRONG HỆ THỐNG PHÂN TÁN	55
2.1 Cơ sở truyền thông	55
2.1.1 Giao thức mạng	56
2.1.1.1 Giao thức các tầng thấp	58

2.1.1.2 Giao thức tầng vận tải	58
2.1.1.3 Giao thức tầng cao	59
2.1.1.4 Giao thức tầng trung gian	60
2.1.2 Phân loại truyền thông	61
2.2 Chuyển thông điệp	62
2.2.1 Chuyển thông điệp dựa trên dịch vụ tầng vận tải	63
2.2.2 Giao diện truyền thông điệp	65
2.2.3 Chuyển thông điệp bền bỉ	67
2.3 Gọi thủ tục từ xa	69
2.3.1 Cơ chế hoạt động	69
2.3.2 Vấn đề truyền tham số	72
2.3.3 Các phương pháp gọi thủ tục từ xa	74
2.4 Truyền thông luồng	75
2.4.1 Hỗ trợ truyền thông liên tục	76
2.4.2 Luồng và chất lượng dịch vụ	77
2.4.3 Đồng bộ luồng	79
2.5 Truyền thông theo nhóm	81
2.5.1 Truyền thông dựa trên mạng phủ	82
2.5.1.1 Tổ chức hình cây	82
2.5.1.2 Quảng bá trong nhóm	84
2.5.2 Lan truyền ngẫu nhiên	87
<b>CHƯƠNG 3: ĐẶT TÊN TRONG HỆ THỐNG PHÂN TÁN</b>	<b>91</b>
3.1 Tên, định danh và địa chỉ	91
3.2 Đặt tên và các giải pháp tìm kiếm	91
3.2.1 Đặt tên phi cấu trúc	92
3.2.1.1 Giải pháp đơn giản	92
3.2.1.2 Giải pháp dựa trên nguồn gốc	94
3.2.1.3 Tìm kiếm dựa trên bảng băm phân tán	96
3.2.1.4 Giải pháp phân cấp	97
3.2.2 Đặt tên có cấu trúc	101
3.2.2.1 Không gian tên	101
3.2.2.2 Phân giải tên	103
3.2.2.3 Cài đặt không gian tên	105
3.2.2.4 Hệ thống tên miền trên Internet	110
3.2.3 Đặt tên dựa trên thuộc tính	113
3.2.3.1 Dịch vụ thư mục	114
3.2.3.2 Cài đặt theo kiến trúc phân cấp	114
3.2.3.3 Cài đặt theo kiến trúc ngang hàng	116
<b>CHƯƠNG 4: ĐỒNG BỘ VÀ CÁC GIẢI THUẬT PHÂN TÁN</b>	<b>120</b>
4.1 Đồng bộ đồng hồ vật lý	120
4.1.1 Giải thuật Cristian	123
4.1.2 Giải thuật Berkeley	125
4.1.3 Giải thuật trung bình	126

4.1.4 Giải thuật tham chiếu quảng bá	127
4.2 Thời gian và đồng hồ logic	129
4.2.1 Đồng hồ Lamport	130
4.2.2 Đồng hồ vector	132
4.2.3 Trạng thái toàn cục	136
4.3 Các giải thuật loại trừ tương hỗ phân tán	138
4.3.1 Giải thuật tập trung	138
4.3.2 Giải thuật không tập trung	139
4.3.3 Giải thuật phân tán	140
4.3.4 Giải thuật thẻ bài	141
4.3.5 So sánh các giải thuật loại trừ	142
4.4 Các giải thuật bầu chọn	142
4.4.1 Giải thuật nồi bọt	142
4.4.2 Giải thuật vòng	144
4.4.3 Bầu chọn trong môi trường không dây	145
4.4.4 Bầu chọn trong các hệ thống qui mô lớn	147
4.5 Hệ thống định vị	148
<b>CHƯƠNG 5: TIẾN TRÌNH TRONG CÁC HỆ THỐNG PHÂN TÁN</b>	<b>151</b>
5.1 Các luồng	151
5.1.1 Khái niệm luồng	151
5.1.2 Luồng trong trên máy tính độc lập	152
5.1.3 Cài đặt luồng	153
5.1.4 Luồng trong các hệ thống phân tán	154
5.2 Ảo hóa	157
5.2.1 Vai trò ảo hóa	157
5.2.2 Phân loại ảo hóa	158
5.3 Máy khách	160
5.3.1 Cung cấp dữ liệu cho máy khách	160
5.3.2 Tính trong suốt phân bố tài nguyên	161
5.4 Máy chủ	162
5.4.1 Thiết kế phần mềm	162
5.4.2 Lắp đặt máy chủ	163
5.4.3 Quản lý cụm máy chủ	166
5.5 Di trú mã	167
5.5.1 Các giải pháp di trú mã	167
5.5.2 Di trú và tài nguyên cục bộ	169
5.5.3 Di trú trong hệ thống không đồng nhất	171
<b>CHƯƠNG 6: QUẢN TRỊ GIAO TÁC VÀ ĐIỀU KHIỂN TƯƠNG TRANH</b>	<b>173</b>
6.1 Không sử dụng giao tác	173
6.2 Khái niệm giao tác	175
6.2.1 Giao tác phẳng	176
6.2.2 Các giao tác lồng nhau	177

6.2.3 Giao tác phân tán	177
6.3 Các phương pháp điều khiển tương tranh	178
6.3.1 Điều khiển tương tranh bằng khóa	179
6.3.2 Điều khiển tương tranh lạc quan	182
6.3.2.1 Phê chuẩn ngược	184
6.3.2.2 Phê chuẩn xuôi	184
6.3.3 Điều khiển tương tranh dựa trên nhãn thời gian	185
CHƯƠNG 7: PHỤC HỒI VÀ TÍNH CHỊU LỖI	187
7.1 Giới thiệu tính chịu lỗi	187
7.1.1 Khái niệm tính chịu lỗi	187
7.1.2 Phân loại lỗi	188
7.1.3 Các mô hình thất bại	189
7.2 Các biện pháp đảm bảo tính chịu lỗi	190
7.2.1 Che giấu lỗi bằng biện pháp dư thừa	190
7.2.2 Tiến trình bền bỉ	191
7.2.2.1 Tổ chức nhóm	192
7.2.2.2 Che giấu thất bại và nhân bản	193
7.2.2.3 Đồng thuận trong các hệ thống lỗi	193
7.2.2.4 Phát hiện lỗi	196
7.2.3 Truyền thông khách/chủ tin cậy	197
7.2.3.1 Truyền thông điểm – điểm	197
7.2.3.2 Những thất bại khi thủ tục từ xa	197
7.2.4 Truyền thông nhóm tin cậy	201
7.2.4.1 Truyền thông nhóm tin cậy	201
7.2.4.2 Truyền thông nhóm tin cậy trong các hệ thống lớn	202
7.3 Cam kết phân tán	204
7.3.1 Giao thức cam kết một pha	204
7.3.2 Giao thức cam kết hai pha	204
7.3.3 Giao thức cam kết ba pha	205
7.4 Phục hồi	207
7.4.1 Các biện pháp phục hồi	207
7.4.2 Điểm kiểm tra	208
7.4.2.1 Điểm kiểm tra độc lập	209
7.4.2.2 Điểm kiểm tra phối hợp	210
7.4.3 Ghi nhật ký thông điệp	211
CHƯƠNG 8: NHẤT QUÁN VÀ NHÂN BẢN	214
8.1 Giới thiệu chung	214
8.2 Các mô hình nhất quán lấy dữ liệu làm trung tâm	216
8.2.1 Nhất quán liên tục	217
8.2.2 Nhất quán theo thứ tự thao tác	220
8.2.2.1 Mô hình nhất quán nghiêm ngặt	220
8.2.2.2 Mô hình nhất quán tuần tự	221
8.2.2.3 Mô hình nhất quán nhân quả	224

8.2.2.4 Mô hình nhất quán hàng đợi	225
8.2.3 Nhất quán theo nhóm các thao tác	226
8.2.3.1 Mô hình nhất quán yếu	226
8.2.3.2 Mô hình nhất quán phát hành	227
8.2.3.3 Mô hình nhất quán mục dữ liệu	229
8.2.4 Tính nhất quán và gắn kết	230
8.3 Nhất quán lấy máy khách làm trung tâm	230
8.3.1 Khái niệm nhất quán sau cùng	230
8.3.2 Mô hình nhất quán đọc đều	232
8.3.3 Mô hình nhất quán ghi đều	232
8.3.4 Nhất quán đọc kết quả ghi	233
8.3.5 Nhất quán ghi sau khi đọc	233
8.4 Quản lý các bản sao	234
8.4.1 Vị trí máy chủ bản sao	234
8.4.2 Nhân bản nội dung và vị trí	234
8.4.3 Phân phát nội dung	236
8.4.3.1 Trạng thái hay thao tác	236
8.4.3.2 Hình thức đẩy hay kéo	237
8.4.3.3 Phương pháp lan truyền cập nhật	238
8.5 Các giao thức nhất quán	238
8.5.1 Nhất quán liên tục	239
8.5.1.1 Giới hạn sai số	239
8.5.1.2 Giới hạn chênh lệch trạng thái	240
8.5.1.3 Giới hạn độ lệch thứ tự	240
8.5.2 Các giao thức dựa trên bản chính	241
8.5.2.1 Giao thức ghi từ xa	241
8.5.2.2 Giao thức ghi cục bộ	242
8.5.3 Các giao thức nhân bản cập nhật	243
8.5.3.1 Nhân bản tích cực	243
8.5.3.2 Giao thức dựa trên đại diện	243
8.5.4 Giao thức gắn kết bộ nhớ cache	244
8.5.5 Cài đặt nhất quán lấy máy khách làm trung tâm	246
CHƯƠNG 9: BẢO MẬT	249
9.1 Giới thiệu chung	249
9.1.1 Các hình thức xâm phạm hệ thống thông tin	249
9.1.1.1 Tân công thăm dò	251
9.1.1.2 Truy nhập trái phép	251
9.1.1.3 Tân công từ chối dịch vụ	252
9.1.1.4 Phần mềm độc hại	252
9.1.2 Các vấn đề thiết kế	253
9.1.2.1 Xác định trọng tâm bảo mật	253
9.1.2.2 Phân tầng cơ chế bảo mật	254
9.1.2.3 Phân bố cơ chế bảo mật	255
9.1.2.4 Tính đơn giản trong thiết kế bảo mật	256

9.1.3	Mã hóa	257
9.1.3.1	Giải thuật mã hóa DES	258
9.1.3.2	Giải thuật mã hóa RSA	259
9.1.3.3	Hàm băm MD5	260
9.2	Các kênh bảo mật	261
9.2.1	Xác thực	261
9.2.1.1	Xác thực dựa trên khóa bí mật	261
9.2.1.2	Xác thực sử dụng trung tâm phân phối khóa	263
9.2.1.3	Xác thực dựa trên mã hóa khóa công khai	265
9.2.2	Toàn vẹn và bí mật thông điệp	266
9.2.2.1	Chữ ký số	266
9.2.2.2	Khóa phiên	267
9.2.3	Truyền thông nhóm bảo mật	268
9.2.3.1	Bí mật truyền thông trong nhóm	268
9.2.3.2	Bảo mật các máy chủ nhân bản	268
9.2.4	Xác thực bằng Kerberos	269
9.3	Kiểm soát truy nhập	271
9.3.1	Nguyên lý kiểm soát truy nhập	271
9.3.1.1	Ma trận kiểm soát truy nhập	271
9.3.1.2	Miền bảo vệ	272
9.3.2	Tường lửa	273
9.3.3	Bảo mật mã di động	273
9.3.3.1	Bảo vệ tác tử	273
9.3.3.2	Bảo vệ đích	274
9.3.4	Tù chối dịch vụ	278
9.4	Quản lý bảo mật	279
9.4.1	Quản lý khóa	279
9.4.1.1	Thiết lập khóa	280
9.4.1.2	Phân phát khóa	280
9.4.1.3	Thời gian sống của chứng chỉ	282
9.4.2	Quản lý bảo mật nhóm	282
9.4.3	Quản lý ủy quyền	283
9.4.3.1	Chứng chỉ quyền truy nhập	284
9.4.3.2	Ủy nhiệm	285
9.5	Một số vấn đề an toàn và bảo mật thông tin khác	287
<b>CHƯƠNG 10: PHÁT TRIỂN HỆ THỐNG PHÂN TÁN</b>		289
10.1	Đối tượng phân tán	289
10.1.1	Mô hình đối tượng thành phần phân tán	289
10.1.2	Gọi phương thức từ xa trong Java	290
10.1.3	Dịch vụ web	291
10.1.4	Kiến trúc môi trường yêu cầu đối tượng chung	292
10.2	Kiến trúc hướng dịch vụ	295
10.2.1	Giới thiệu kiến trúc hướng dịch vụ	296

10.2.2 Các dịch vụ	296
10.2.3 Mô hình cặp lồng	298
10.2.4 Chu kỳ sống dịch vụ	298
10.2.5 Phân loại dịch vụ	299
10.2.6 Trục dịch vụ doanh nghiệp	299
10.3 Kiến trúc MicroService	300
TÀI LIỆU THAM KHẢO	302

BẢNG DANH MỤC CHỮ VIẾT TẮT

<b>Viết tắt</b>	<b>Tiếng Anh</b>	<b>Tiếng Việt</b>
ACL	Access Control List	Danh sách kiểm soát truy nhập
ACP	Atomic Commit Protocol	Giao thức cam kết nguyên tử
ANIAC	Electronic Numerical Integrator and Computer	Máy tính tích hợp điện tử số
API	Application Programming Interface	Giao diện lập trình ứng dụng
BTS	Base Transceiver Station	Trạm thu phát sóng di động
CA	Certificate Authority	Cơ quan chứng nhận
CAN	Content addressable network	Mạng địa chỉ nội dung
COM	Component Object Model	Mô hình đối tượng thành phần
CORBA	Common Object Request Broker Architecture	Kiến trúc môi trường yêu cầu đối tượng chung
CPU	Central Processing Unit	Đơn vị xử lý trung tâm
CRL	Certificate Revocation List	Danh sách thu hồi chứng chỉ
DAP	Directory Application Protocol	Giao thức ứng dụng thư mục
DCOM	Distributed Component Object Model	Mô hình đối tượng thành phần phân tán
DDoS	Distributed Denial of Service	Từ chối dịch vụ phân tán
DES	Data Encryption Standard	Tiêu chuẩn mã hóa dữ liệu
DNS	Domain Name System	Hệ thống phân giải tên miền
DoS	Denial of Service	Từ chối dịch vụ
DRDA	Distributed Relational Database Architecture	Kiến trúc cơ sở dữ liệu quan hệ phân tán
DSA	Directory System Agent	Đại lý hệ thống thư mục
FTP	File Transfer Protocol	Giao thức truyền tập tin
GPS	Global Positioning System	Hệ thống định vị toàn cầu
HDLC	High-Level Data Link Control	Điều khiển liên kết dữ liệu mức cao
HTML	Hyper Text Markup Language	Ngôn ngữ đánh dấu siêu văn bản
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
IANA	Internet Assigned Numbers Authority	Tổ chức cấp phát số hiệu Internet
ICMP	Internet Control Message Protocol)	Giao thức thông điệp kiểm soát Internet
IDL	Interface Definition Language	Ngôn ngữ định nghĩa giao diện
IP	Internet Protocol	Giao thức liên mạng
IPRA	Internet Policy Registration Authority	Cơ quan đăng ký chính sách Internet
ISO	International Organization for Standardization	Tổ chức chuẩn hóa quốc tế
ITU	International Telecommunication Union	Liên minh viễn thông quốc tế
KDC	Key Distribution Center	Trung tâm phân phối khóa
LDAP	Lightweight Directory Access Protocol	Giao thức truy nhập thư mục hạng nhẹ
LWP	LightWeight Process	Tiến trình hạng nhẹ

MD5	Message-Digest 5	Chữ số hóa thông điệp loại 5
MPI	Message Passing Interface	Giao diện truyền thông điệp
MMU	Memory Management Unit	Đơn vị quản lý bộ nhớ
NTP	Network Time Protocol	Giao thức đồng bộ thời gian mạng
NFS	Network File System	Giao thức hệ thống tập tin mạng
NTSC	National Television System Committee	Ủy ban hệ thống truyền hình quốc gia (Hoa Kỳ)
OLE	Object Linking and Embedding	Liên kết và nhúng đối tượng
OMG	Object Management Group	Nhóm quản lý đối tượng
OSI	Open Systems Interconnection Reference Model	Mô hình tham chiếu liên kết các hệ thống mở
PCA	Policy Certification Authority	Cơ quan chứng nhận chính sách
PEM	Privacy Enhanced Mail	Thư được tăng cường tính riêng tư
PPP	Point-to-Point Protocol	Giao thức điểm – điểm
RSA		Viết tắt tên của ba nhà phát minh Rivest, Shamir và Adleman
RMI	Remote Method Invocation	Gọi phương thức từ xa
RPC	Remote Procedure Call	Gọi thủ tục từ xa
RAID	Redundant Array of Independent Disks	Mảng dư thừa các đĩa độc lập
RISSC	Reduced Interfaces for Secure System Components	Giao diện tối giản để bảo mật các thành phần hệ thống
SCORM	Sharable Content Object Reference Model	Mô hình tham chiếu đối tượng nội dung chia sẻ
SMDS	Switched Multi-megabit Data Service	Dịch vụ chuyển mạch tốc độ cao (Mega bit/s)
SMNP	Simple Network Management Protocol	Giao thức quản lý mạng
SMTP	Simple Mail Transfer Protocol	Giao thức chuyển thư đơn giản
SOA	Service-oriented Architecture	Kiến trúc hướng dịch vụ
SOAP	Simple Object Access Protocol	Giao thức truy nhập đối tượng đơn giản
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc
SRM	Scalable Reliable Multicasting	Nhóm tin cậy qui mô lớn
SSL	Secure Sockets Layer	Bảo mật tầng Socket
STP	Spanning Tree Protocol	Giao thức cây mở rộng
TAI	International Atomic Time	Thời gian nguyên tử quốc tế
TCP	Transmission Control Protocol	Giao thức điều khiển vận chuyển
TGS	Ticket Granting Service	Dịch vụ cấp thẻ
TLB	Translation Lookaside Buffer	Vùng đệm tra cứu chuyển đổi
UCLA	University of California, Los Angeles	Đại học California, Los Angeles
UDDI	Universal Description, Discovery and Integration	Mô tả vạn năng, khám phá và tích hợp
UDP	User Datagram Protocol	Giao thức đơn vị dữ liệu của người sử dụng
URL	Uniform Resource Locator	Định vị tài nguyên thống nhất

UTC	Universal Time Coordinated	Thời gian phối hợp quốc tế
XML	eXtensible Markup Language	Ngôn ngữ đánh dấu mở rộng
WSDL	Web Services Description Language	Ngôn ngữ mô tả dịch vụ Web

## DANH SÁCH HÌNH VẼ

Số thứ tự	Tên hình vẽ	Nguồn
Hình 1.1	Ba loại hệ thống nhiều bộ xử lý dựa trên một kênh truyền	
Hình 1.2	Hệ thống điện toán cụm bất đối xứng	
Hình 1.3	Mô hình phân tầng hệ thống điện toán lưới	
Hình 1.4	Phần mềm trung gian trong hệ thống thông tin phân tán	
Hình 1.5	Ưu tiên xử lý trên máy khách	
Hình 1.6	Mạng ngang hàng có cấu trúc	
Hình 1.7	Mạng địa chỉ nội dung	
Hình 1.8	Mô hình phân tầng	
Hình 1.9	Tổ chức phân tầng máy tìm kiếm trên Internet	Trang 61 tài liệu [1]
Hình 1.10	Phân chia xử lý giữa máy khách và máy chủ	
Hình 1.11	Mô hình đối tượng phân tán	
Hình 1.12	Mô hình khách/chủ	
Hình 1.13	Máy chủ đóng vai trò máy khách khi truy nhập cơ sở dữ liệu	
Hình 1.14	Mô hình kênh sự kiện	
Hình 1.15	Mô hình dữ liệu tập trung	
Hình 1.16	Tầng trung gian trong mô hình khách/chủ	
Hình 2.1	Mô hình tham chiếu liên kết các hệ thống mở	
Hình 2.2	Bao đóng dữ liệu tại các tầng của mô hình OSI	
Hình 2.3	Qui trình truyền dữ liệu của giao thức TCP	
Hình 2.4	Nguyên lý truyền thông sử dụng tầng trung gian	
Hình 2.5	Sử dụng hàm nguyên thủy trong chuyển thông điệp	
Hình 2.6	Các liên kết trong mô hình hàng đợi thông điệp	
Hình 2.7	Ngăn xếp khi gọi thủ tục trên máy tính	
Hình 2.8	Nguyên lý gọi thủ tục từ xa	
Hình 2.9	Các bước thực hiện trong gọi thủ tục từ xa	Trang 130 tài liệu [1]
Hình 2.10	Truyền tham số trong các hệ thống không đồng nhất	Trang 131 tài liệu [1]
Hình 2.11	Mã hóa tham số vào thông điệp	Trang 133 tài liệu [1]
Hình 2.12	Hai phương pháp gọi thủ tục từ xa	
Hình 2.13	Kiến trúc hệ thống truyền thông đa phương tiện	Trang 160 tài liệu [1]
Hình 2.14	Sử dụng bộ đệm trong truyền thông đa phương tiện	Trang 161 tài liệu [1]
Hình 2.15	Giải pháp xử lý mảng gói tin	Trang 162 tài liệu [1]

Hình 2.16	Nguyên lý đồng bộ mức đơn vị dữ liệu	Trang 164 tài liệu [1]
Hình 2.17	Nguyên lý đồng bộ mức giao diện	Trang 165 tài liệu [1]
Hình 2.18	Quan hệ giữa liên kết của mạng phủ và mạng vật lý	Trang 168 tài liệu [1]
Hình 2.19	Đồ thị quan hệ giữa số cạnh và số nút	
Hình 2.20	Đồ thị xác suất lan truyền ngẫu nhiên qua các vòng	
Hình 2.21	Đồ thị tương quan giữa xác suất dừng với xác suất chưa cập nhật	
Hình 3.1	Nguyên lý con trỏ chuyển tiếp	Trang 244 tài liệu [1]
Hình 3.2	Định hướng lại con trỏ chuyển tiếp	Trang 244 tài liệu [1]
Hình 3.3	Nguyên lý cách tiếp cận dựa trên nguồn gốc	Trang 246 tài liệu [1]
Hình 3.4	Tổ chức các nút mạng Chord	Trang 248 tài liệu [1]
Hình 3.5	Tổ chức phân cấp	Trang 252 tài liệu [1]
Hình 3.6	Lưu trữ thông tin cho thực thể có nhiều địa chỉ	Trang 253 tài liệu [1]
Hình 3.7	Tìm kiếm trong tổ chức phân cấp	Trang 253 tài liệu [1]
Hình 3.8	Thêm thực thể mới	Trang 254 tài liệu [1]
Hình 3.9	Tổ chức phân cấp logic trong hệ thống máy chủ vật lý	Trang 256 tài liệu [1]
Hình 3.10	Đồ thị tên đơn gốc	Trang 257 tài liệu [1]
Hình 3.11	Tổ chức hệ thống quản lý tập tin	Trang 259 tài liệu [1]
Hình 3.12	Liên kết trong đồ thị tên	Trang 261 tài liệu [1]
Hình 3.13	Gắn kết không gian tên	Trang 263 tài liệu [1]
Hình 3.14	Tổ chức tên miền Internet	Trang 265 tài liệu [1]
Hình 3.15	Phân giải tên miền tương tác	
Hình 3.16	Phân giải tên miền đệ qui	
Hình 3.17	So sánh hai giải pháp phân giải tên miền	
Hình 3.18	Các bước thực hiện trong phân giải tên miền	
Hình 3.19	Tổ chức bản ghi dịch vụ thư mục	
Hình 3.20	Cách đánh chỉ mục trong không gian Hilbert	Trang 290 tài liệu [1]
Hình 4.1	Hai máy tính có thời gian khác nhau	
Hình 4.2	Giải thuật Cristian	
Hình 4.3	Đồng bộ thời gian bằng giải thuật Berkeley	Trang 307 tài liệu [1]
Hình 4.4	Cách tính thời gian trễ khi chuyển thông điệp	Trang 309 tài liệu [1]
Hình 4.5	Đồng hồ logic tại tầng trung gian	Trang 313 tài liệu [1]
Hình 4.6	Đồng bộ nhãn thời gian Lamport	Trang 312 tài liệu [1]
Hình 4.7	Cập nhật phân tán	Trang 314 tài liệu [1]
Hình 4.8	Tính nhân quả trong chuyển thông điệp liên tiến trình	
Hình 4.9	Cập nhật nhãn thời gian vector	
Hình 4.10	Truyền thông nhân quả và nhãn thời gian vector	
Hình 4.11	Truyền thông nhân quả và nhãn thời gian vector cải tiến	
Hình 4.12	Lát cắt trong hệ thống phân tán	

Hình 4.13	Các loại lát cắt	
Hình 4.14	Nguyên lý giải thuật tập trung	
Hình 4.15	Nguyên lý giải thuật phân tán	Trang 325 tài liệu [1]
Hình 4.16	Nguyên lý giải thuật thẻ bài	
Hình 4.17	Nguyên lý giải thuật nồi bọt	
Hình 4.18	Nguyên lý bầu chọn bằng giải thuật vòng	Trang 333 tài liệu [1]
Hình 4.19	Bầu chọn trong môi trường không dây	
Hình 4.20	Nguyên lý di chuyển thẻ bài trong không gian hai chiều bằng lực đẩy	Trang 337 tài liệu [1]
Hình 4.21	Tính toán vị trí trong không gian hai chiều	
Hình 5.1	Chuyển ngữ cảnh giữa các tiến trình	
Hình 5.2	Kết hợp tiến trình hạng nhẹ và các luồng mức người sử dụng	Trang 110 tài liệu [1]
Hình 5.3	Xử lý đa luồng trên máy chủ	
Hình 5.4	Áo hóa tài nguyên phần cứng	
Hình 5.5	Các loại giao diện trên máy tính	
Hình 5.6	Hai hình thức áo hóa	
Hình 5.7	Hai hình thức cung cấp dữ liệu cho máy khách	
Hình 5.8	Tính trong suốt truy nhập tài nguyên nhân bản	
Hình 5.9	Cài đặt các tiến trình máy chủ	Trang 130 tài liệu [1]
Hình 5.10	Tổ chức cụm máy chủ theo mô hình ba lớp	Trang 142 tài liệu [1]
Hình 5.11	Định tuyến máy chủ phân tán	Trang 148 tài liệu [1]
Hình 5.12	Nguyên lý cấu hình động cho máy khách	Trang 154 tài liệu [1]
Hình 5.13	Phân loại di trú	
Hình 6.1	Các trường hợp thực hiện trong giao tác phẳng	
Hình 6.2	Cấu trúc giao tác lồng nhau	
Hình 6.3	Phê chuẩn giao tác	Trang 709 tài liệu [3]
Hình 6.4	Lưu đồ xử lý giao tác dựa trên nhãn thời gian	
Hình 7.1	Xây dựng hệ thống dư thừa theo kiểu chuyển mạch	Trang 434 tài liệu [1]
Hình 7.2	Nhóm các tiến trình trong hệ thống phân tán	
Hình 7.3	Các trường hợp đạt được sự đồng thuận	
Hình 7.4	Giải thuật đồng thuận phân tán	Trang 333 tài liệu [2]
Hình 7.5	Không xác định được sự đồng thuận	Trang 334 tài liệu [2]
Hình 7.6	Những khả năng lỗi khi gọi thủ tục từ xa	
Hình 7.7	Truyền thông nhóm tin cậy cơ bản	Trang 475 tài liệu [1]
Hình 7.8	Phản hồi không phân cấp	Trang 477 tài liệu [1]
Hình 7.9	Điều khiển phản hồi phân cấp	Trang 347 tài liệu [2]
Hình 7.10	Chuyển trạng thái trong giao thức cam kết hai pha	Trang 487 tài liệu [1]
Hình 7.11	Chuyển trạng thái trong giao thức cam kết ba pha	Trang 491 tài liệu [1]
Hình 7.12	Phục hồi lùi sử dụng trạng thái toàn cục	Trang 496 tài liệu [1]

	<u>nhất quán</u>	
Hình 7.13	<u>Điểm kiểm tra độc lập</u>	Trang 497 tài liệu [1]
Hình 7.14	<u>Không nhất quán sau khi phục hồi</u>	Trang 499 tài liệu [1]
Hình 8.1	<u>Tổ chức kho dữ liệu nhân bản</u>	
Hình 8.2	<u>Tính nhất quán với điều kiện nói lỏng thứ tự thao tác</u>	
Hình 8.3	<u>Số lượng thao tác trong lan truyền cập nhật</u>	
Hình 8.4	<u>Mô hình nhất quán nghiêm ngặt</u>	
Hình 8.5	<u>Mô hình nhất quán tuần tự</u>	
Hình 8.6	<u>Ba tiến trình thực hiện tương tranh</u>	
Hình 8.7	<u>Bốn trường hợp nhất quán tuần tự hợp lệ</u>	
Hình 8.8	<u>Tuyến tính hóa trong mô hình nhất quán tuần tự</u>	
Hình 8.9	<u>Mô hình nhất quán nhân quả</u>	
Hình 8.10	<u>Nhất quán nhân quả và vi phạm tính nhất quán nhân quả</u>	
Hình 8.11	<u>Nhất quán nhân quả trên các biến khác nhau</u>	
Hình 8.12	<u>Mô hình nhất quán hàng đợi</u>	
Hình 8.13	<u>Mô hình nhất quán yếu</u>	
Hình 8.14	<u>Mô hình nhất quán phát hành</u>	
Hình 8.15	<u>Mô hình nhất quán mục dữ liệu</u>	
Hình 8.16	<u>Người sử dụng truy nhập các bản sao dữ liệu nhân bản</u>	
Hình 8.17	<u>Mô hình nhất quán đọc đều</u>	
Hình 8.18	<u>Mô hình nhất quán ghi đều</u>	
Hình 8.19	<u>Mô hình nhất quán đọc kết quả ghi</u>	
Hình 8.20	<u>Mô hình nhất quán ghi sau khi đọc</u>	
Hình 8.21	<u>Đếm số lượng yêu cầu từ máy khách</u>	
Hình 8.22	<u>Giao thức ghi từ xa</u>	
Hình 8.23	<u>Giao thức ghi cục bộ</u>	
Hình 8.24	<u>Giao thức dựa trên đại biểu</u>	Trang 405 tài liệu [1]
Hình 9.1	<u>Các cách tiếp cận bảo mật</u>	Trang 509 tài liệu [1]
Hình 9.2	<u>Tổ chức phân tầng logic hệ thống phân tán</u>	
Hình 9.3	<u>Bảo mật liên mạng</u>	
Hình 9.4	<u>Nguyên lý RISC áp dụng cho hệ thống phân tán</u>	
Hình 9.5	<u>Xâm nhập và nghe trộm trong truyền thông</u>	Trang 514 tài liệu [1]
Hình 9.6	<u>Vòng lặp sản sinh khóa trong DES</u>	
Hình 9.7	<u>Qui trình mã hóa MD5</u>	
Hình 9.8	<u>Thao tác lặp vòng đầu của pha trong MD5</u>	
Hình 9.9	<u>Xác thực dựa trên khóa bí mật thực hiện 5 bước</u>	
Hình 9.10	<u>Xác thực dựa trên khóa bí mật thực hiện 3 bước</u>	
Hình 9.12	<u>Nguyên lý sử dụng trung tâm phân phối khóa</u>	
Hình 9.13	<u>KDC cho phép hai bên tự kết nối</u>	

Hình 9.14	Giao thức Needham-Schroeder	
Hình 9.15	Giao thức Needham-Schroeder chống lại tái sử dụng khóa phiên trước	
Hình 9.16	Xác thực lẫn nhau dựa trên mã hóa khóa công khai	
Hình 9.17	Chữ ký số sử dụng khóa mã hóa công khai	
Hình 9.18	Chữ ký số sử dụng bản thu gọn của thông điệp	Trang 526 tài liệu [1]
Hình 9.19	Chia sẻ chữ ký bí mật trong nhóm các máy chủ nhân bản	Trang 530 tài liệu [1]
Hình 9.20	Xác thực trong Kerberos	
Hình 9.21	Thiết lập kênh bảo mật trong Kerberos	
Hình 9.22	Nguyên lý kiểm soát truy nhập đối tượng	
Hình 9.23	Danh sách kiểm soát truy nhập	
Hình 9.24	Kiểm soát theo khả năng	
Hình 9.25	Cài đặt tường lửa	Trang 538 tài liệu [1]
Hình 9.26	Cấu trúc hộp cát Java	Trang 540 tài liệu [1]
Hình 9.27	Hộp cát và sân chơi	Trang 424 tài liệu [2]
Hình 9.28	Nguyên lý sử dụng tham chiếu đối tượng	Trang 541 tài liệu [1]
Hình 9.29	Nguyên lý kiểm tra bên trong ngăn xếp	
Hình 9.30	Nguyên lý trao đổi khóa Diffie – Hellman	
Hình 9.31	Phân phát khóa bí mật	Trang 548 tài liệu [1]
Hình 9.32	Phân phát khóa công khai	Trang 548 tài liệu [1]
Hình 9.33	Qui trình tiếp nhận thành viên mới	
Hình 9.34	Cấu trúc định danh của chứng chỉ	
Hình 9.35	Tạo chứng chỉ hạn chế từ chứng chỉ của chủ sở hữu	Trang 553 tài liệu [1]
Hình 9.36	Cấu trúc chung của đại diện dùng để ủy quyền	
Hình 9.37	Chứng minh sở hữu quyền truy nhập	

## CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG PHÂN TÁN

Từ khi máy tính điện tử đầu tiên ANIAC ra đời vào năm 1945 cho đến năm 1980 các máy tính có kích thước lớn và khá đắt tiền, do đó nhiều cơ quan chỉ có vài máy tính và thiếu phương tiện kết nối chúng, những máy tính này hoạt động độc lập với nhau. Bắt đầu từ năm 1980, những tiến bộ về công nghệ đã làm cho hệ thống máy tính phát triển ngày càng mạnh mẽ và thâm nhập vào mọi lĩnh vực đời sống xã hội. Tiến bộ thứ nhất là sự phát triển các bộ vi xử lý cho máy tính cá nhân, ban đầu thanh ghi chỉ có 8 bit rồi sau đó chuyển dần sang 16, 32 và hiện nay là 64 bit, chúng có khả năng tính toán như một máy tính lớn nhưng giá thành lại thấp hơn rất nhiều. Với việc trang bị bộ vi xử lý đa lõi trên các máy tính, những người phát triển phần mềm đang phải thích nghi để xây dựng các ứng dụng sao cho có thể khai thác lợi thế của xử lý song song.

Bước tiến bộ thứ hai là sự phát minh ra mạng máy tính tốc độ cao, các mạng cục bộ cho phép kết nối hàng trăm máy tính với băng thông lên tới hàng chục tỉ bit/s, các mạng diện rộng cho phép kết nối hàng triệu máy tính với băng thông từ vài chục ngàn bit/s đến hàng trăm triệu bit/s. Nhờ những thành tựu này, có thể kết nối số lượng lớn máy tính với nhau qua đường truyền tốc độ cao, chúng khác hẳn với các hệ thống tập trung chỉ bao gồm vài máy tính và thiết bị ngoại vi phục vụ cho nhóm nhỏ người sử dụng.

Năng lực xử lý của các thiết bị mạng ngày càng tăng nhưng kích thước của chúng ngày càng giảm, một chiếc điện thoại thông minh chỉ bé bằng bàn tay nhưng cũng có khả năng xử lý như một máy tính để bàn, điều này đã làm bùng nổ số lượng các thiết bị tham gia vào mạng máy tính vốn đã rất lớn. Các thiết bị này nằm rải rác trên phạm vi toàn cầu hình thành lên hệ thống phân tán, chúng có thể kết nối với nhau qua các kênh truyền dẫn hữu tuyến hoặc vô tuyến. Hơn nữa, hệ thống phân tán có tính động rất cao, nghĩa là một thành viên có thể tham gia hoặc rời bỏ bất cứ khi nào, hình trạng và hiệu năng của mạng cũng luôn thay đổi. Chương này sẽ đề cập đến một số khái niệm cơ bản, sau đó sẽ trình bày những mục tiêu thiết kế và các mô hình xây dựng hệ thống phân tán.

### 1.1 Định nghĩa hệ thống phân tán

Nhiều tài liệu đã đưa ra những định nghĩa khác nhau về hệ thống phân tán, nhưng chưa có một định nghĩa nào thỏa đáng. George Coulouris cho rằng hệ thống phân tán là hệ thống trong đó các thành phần đặt trên các máy tính kết nối mạng trao đổi thông tin và phối hợp các hoạt động chỉ bằng cách chuyển thông điệp. Andrew S. Tanenbaum định nghĩa hệ thống phân tán là tập hợp các phần tử điện toán tự trị xuất hiện trước người sử dụng như một hệ thống gắn kết.

Nhìn chung, các định nghĩa hệ thống phân tán đều đề cập đến hai đặc điểm quan trọng, đặc điểm thứ nhất là tập hợp các phần tử tính toán có thể hoạt động độc lập với nhau và đặc điểm thứ hai là chúng cần phải cộng tác để giải quyết một nhiệm vụ chung. Trong tài liệu này chúng ta thống nhất định nghĩa **hệ thống phân tán là hệ thống các phần mềm cài đặt trên các máy tính độc lập nhưng được phối hợp hoạt động với nhau như một thể thống nhất**.

Phần mềm ứng dụng trên các máy tính trao đổi thông tin với nhau qua mạng, chúng phối hợp các hoạt động chỉ bằng cách chuyển thông điệp làm cho người dùng cảm giác như đang sử dụng một máy tính đơn lẻ. Khái niệm phân tán được thể hiện bởi tính độc lập của từng máy tính nhưng phải phối hợp làm việc một cách đồng bộ với nhau chứ không phải tập hợp các thành phần rời rạc. Ví dụ, một hệ thống bán hàng gồm nhiều cửa hàng đặt tại những vị trí khác nhau, việc nhập thông tin hàng hóa được thực hiện tại nhiều địa điểm như cửa hàng, nhà kho..., tuy nhiên các nhân viên khai thác đều có thể tìm thấy thông tin theo yêu cầu như thế các thông tin đó đang được lưu trữ trên máy tính của mình.

Các thành viên của hệ thống phân tán có thể là những máy tính hiệu năng cao hoặc chỉ là những thiết bị nhỏ như điện thoại thông minh với cấu hình thấp hơn, chúng đều có thể hoạt động một cách độc lập nhưng như vậy thì việc tham gia vào hệ thống phân tán sẽ không còn ý nghĩa. Trong thực tế, các thiết bị này trao đổi thông tin với nhau dưới dạng thông điệp, mỗi khi có thông điệp gửi đến thì chúng phải xử lý để giải quyết nhiệm vụ chung theo cách đã được lập trình. Làm thế nào để phối hợp hoạt động giữa các phần tử nằm trên các máy tính khác nhau là trọng tâm của việc phát triển hệ thống phân tán.

Trên mỗi máy tính đều lắp đặt đồng hồ riêng của mình, chúng hoạt động độc lập với nhau, vì vậy không tồn tại thời gian chung cho toàn bộ hệ thống, do đó việc đồng bộ thời gian là một trong những nhiệm vụ quan trọng khi xây dựng hệ thống phân tán. Ví dụ hệ thống thanh toán quốc tế, thời gian đóng vai trò quan trọng trong việc xác định tỉ giá giữa các đồng tiền giao dịch, sai lệch thời gian giữa các bên có thể dẫn đến báo cáo tài chính không khớp nhau.

Hiệu năng hệ thống cũng ngày càng tốt hơn nhờ những tiến bộ về công nghệ mạng và tận dụng khả năng tính toán trên nhiều máy tính, nhưng lại nảy sinh những vấn đề liên quan đến việc quản lý các thành viên. Mỗi thành viên đều có thể tham gia hoặc rời bỏ hệ thống bất kỳ lúc nào, các thành viên đều có thể truy nhập dữ liệu nhưng chỉ được giới hạn trong phạm vi thẩm quyền của mình.

Tốc độ của bộ vi xử lý trung tâm, bộ nhớ và tốc độ kênh truyền trên bo mạch chủ là những yếu tố then chốt quyết định khả năng tính toán trên máy tính. Trong hệ thống phân tán, trao đổi thông tin giữa các máy tính được thực hiện trên môi trường mạng, thiếu yếu tố này thì nhiệm vụ tính toán sẽ không thể thực hiện được. Mặc dù tốc độ truyền dần ngày càng tăng đã tạo điều kiện cho sự phát triển các ứng dụng phân tán, các máy tính có thể trao đổi thông tin và chia sẻ dữ liệu với nhau không còn phụ thuộc vào khoảng cách địa lý. Bằng thông không phải là vô hạn và dữ liệu di chuyển trên đó sẽ cần một khoảng thời gian nhất định, do đó giảm thiểu thời gian trễ cũng là công việc quan trọng khi xây dựng các ứng dụng phân tán.

Nhìn chung, việc xây dựng các ứng dụng phân tán phức tạp hơn nhiều so với các ứng dụng chạy độc lập trên một máy tính, tuy nhiên đây là xu hướng tất yếu của các hệ thống thông tin. Lý do đầu tiên phải kể đến là nhu cầu chia sẻ thông tin và tài nguyên, người sử dụng trao đổi thông tin với nhau thông qua một ứng dụng chạy trên máy tính của mình. Dữ liệu cũng có thể được lưu trữ phân tán, điều này sẽ nảy sinh những vấn đề

về phân quyền truy nhập, ví dụ cho phép truy nhập dữ liệu từ xa nhưng không cho phép sao chép để lưu giữ cục bộ. Lý do thứ hai là vấn đề hiệu năng, trong nhiều trường hợp bắt buộc tính toán phân tán nhằm tận dụng khả năng tính toán song song hoặc nhằm mục đích khai thác khả năng tính toán của các máy tính chuyên dụng. Cuối cùng là xuất phát từ yêu cầu về khả năng chịu lỗi, hệ thống cần phải đảm bảo an toàn tuyệt đối ngay cả khi có sự cố xảy ra trên một máy chủ nào đó thì cũng không làm ảnh hưởng đến hoạt động của toàn bộ hệ thống. Điều này được thực hiện bằng cách sử dụng mọi biện pháp nhằm mục đích phát hiện kịp thời và xử lý lỗi, nếu máy chủ bị hỏng hoàn toàn thì sẽ chuyển sang các máy tính dự phòng khác để xử lý, như vậy dịch vụ của hệ thống không bị gián đoạn.

Trong các hệ thống phân tán, môi trường mạng đóng vai trò quan trọng trong việc phân phát dữ liệu đến các thành phần và tập hợp kết quả tính toán của các thành phần đó. Các máy tính kết nối với nhau trên mạng đảm nhiệm chức năng truyền thông cho các ứng dụng, chúng không chia sẻ bộ nhớ cho nhau do đó không thể sử dụng các biến toàn cục để trao đổi thông tin, thông tin trao đổi giữa các máy tính chỉ được thực hiện thông qua cơ chế trao đổi thông điệp. Mạng là tài nguyên chung của hệ thống do đó khi xây dựng hệ thống phân tán cần phải xem xét đến các vấn đề như băng thông, các điểm có thể xảy ra sự cố, bảo mật và an toàn dữ liệu, đồng bộ tiến trình.

Quá trình triển khai các ứng dụng trong hệ thống phân tán thường gặp một số khó khăn vì sự đa dạng của các thiết bị, trên mạng tồn tại nhiều chủng loại máy tính và thiết bị mạng của nhiều nhà sản xuất khác nhau và các máy tính được cài đặt các hệ điều hành khác nhau. Các thiết bị dòng Little endian và big endian thể hiện các bit trong mỗi byte như nhau nhưng sắp xếp thứ tự byte dữ liệu trong bộ nhớ lại ngược nhau. Việc tích hợp các phần mềm được phát triển bởi các ngôn ngữ lập trình khác nhau cũng gây trở ngại lớn cho việc triển khai các hệ thống phân tán. Thời gian phát triển phần mềm lâu hơn, số lượng dòng lệnh và các thủ tục nhiều hơn, do đó thường kéo theo chi phí xây dựng hệ thống cao.

Nền tảng cho các hệ thống phân tán dựa trên mạng máy tính, hệ điều hành đã giải quyết nhiệm vụ của các tầng thấp trong mô hình OSI, từ tầng vật lý đến tầng mạng. Các tính năng của ứng dụng phân tán thường chỉ nằm ở các tầng cao, chúng xử lý các thủ tục kết nối giữa các thành phần phân tán và thể hiện cấu trúc dữ liệu phức tạp của các ứng dụng. Để đơn giản hóa việc xây dựng hệ thống, những tính năng chung của các ứng dụng phân tán như xác định vị trí thành viên, đồng bộ, mã hóa thông tin... thường được đặt trong các thư viện và gọi là phần mềm trung gian, nó nằm giữa tầng vận tải và tầng ứng dụng.

## 1.2 Quan điểm về phần cứng

Những máy tính thuở ban đầu chỉ có một bộ vi xử lý và một kênh trao đổi thông tin dùng chung cho cả phần mã lệnh và dữ liệu. Sau đó người ta đã thiết kế máy tính với một bộ xử lý nhưng tách biệt kênh mã lệnh và kênh dữ liệu, tiếp đó là những máy tính có nhiều bộ vi xử lý và tất nhiên nhiều kênh truyền để chúng có thể đồng thời trao đổi thông tin. Michael J. Flynn đã chỉ ra hai đặc trưng cơ bản cho một hệ thống tính toán là số

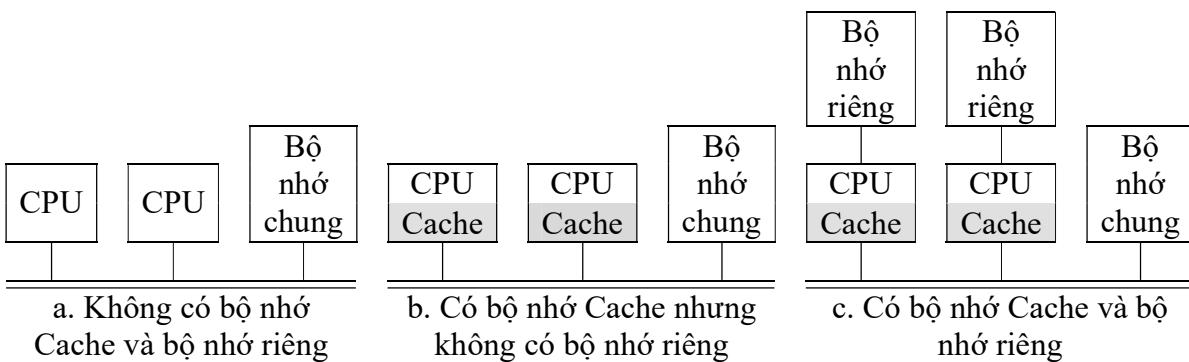
lượng luồng xử lý lệnh và số lượng kênh chuyển dữ liệu, như vậy về lý thuyết có thể xây dựng bốn loại sau:

- Đơn luồng xử lý, đơn luồng dữ liệu: Bộ vi xử lý chỉ có một luồng thực thi các lệnh và một kênh truyền dữ liệu, các máy tính cá nhân là ví dụ điển hình của loại này.
- Đơn luồng xử lý đa luồng dữ liệu: Bao gồm nhiều bộ xử lý nhưng chỉ có một bộ xử lý đóng vai trò tiếp nhận các chỉ thị để ra lệnh cho các bộ xử lý khác thực hiện song song.
- Đa luồng xử lý và đơn luồng dữ liệu: Nhiều bộ xử lý cùng chia sẻ một luồng dữ liệu, thực tế khó có thể tìm thấy máy tính thuộc loại này.
- Đa luồng xử lý và đa luồng dữ liệu: Nhiều bộ xử lý và nhiều luồng dữ liệu, mỗi bộ xử lý sở hữu bộ đếm chương trình, chương trình và dữ liệu.

Đa luồng xử lý và đa luồng dữ liệu thường tìm thấy trong hệ thống nhiều bộ vi xử lý và hệ thống nhiều máy tính. Trong hệ thống nhiều bộ vi xử lý, các bộ vi xử lý có quan hệ chặt chẽ với nhau, chúng dùng chung bộ nhớ chia sẻ, như vậy việc trao đổi thông tin giữa các bộ xử lý có thể thực hiện đơn giản bằng cách đọc/ghi các ô nhớ. Trong hệ thống nhiều máy tính, mỗi máy tính có thể lắp đặt một hoặc nhiều bộ vi xử lý và chúng sở hữu bộ nhớ riêng, các bộ xử lý ràng buộc không chia sẻ, mỗi máy tính đóng vai trò như một bộ xử lý và chúng trao đổi thông tin với nhau qua mạng. Hệ thống nhiều máy tính được coi là đồng nhất nếu các máy tính cùng chung nền tảng phần cứng, hệ điều hành và mạng ngược lại gọi là hệ thống không đồng nhất.

### 1.2.1 Hệ thống nhiều bộ vi xử lý

Hệ thống nhiều bộ vi xử lý đều có đặc điểm chung là các đơn vị xử lý trung tâm đều được kết nối vào các kênh chung trong bo mạch chủ và truy nhập trực tiếp vào bộ nhớ dùng chung. Ngoài các chức năng truyền thông của hệ điều hành như xử lý gọi hệ thống, quản lý bộ nhớ, quản lý tập tin, quản lý thiết bị vào ra..., hệ điều hành của hệ thống nhiều bộ vi xử lý phải thực hiện các chức năng đặc biệt như đồng bộ tiến trình, quản lý tài nguyên và lập lịch làm việc.



Hình 1.1 Ba loại hệ thống nhiều bộ xử lý dựa trên một kênh truyền

Hình 1.1-a minh họa hệ thống nhiều bộ vi xử lý dựa trên một kênh truyền, các đơn vị xử lý trung tâm và bộ nhớ dùng chung một kênh truyền để trao đổi thông tin. Trước khi truy nhập ô nhớ, đơn vị xử lý trung tâm phải kiểm tra xem kênh truyền có bận hay không, nếu rỗi thì đơn vị xử lý trung tâm đặt địa chỉ ô nhớ lên kênh, phát tín hiệu điều

khiến và chờ cho đến khi bộ nhớ đặt giá trị của ô nhớ đã yêu cầu lên kênh truyền. Cách tổ chức trên gọi là kiến trúc truy nhập bộ nhớ đồng nhất, các bộ vi xử lý sử dụng bộ nhớ dùng chung để lưu trữ mã lệnh và dữ liệu của mình, nó đáp ứng yêu cầu cho các ứng dụng lớn nhưng lại chỉ phù hợp với các hệ thống có ít bộ vi xử lý, khi số lượng bộ vi xử lý tăng lên thì chi phí cho vấn đề chuyển mạch cũng rất lớn.

Hiệu năng hệ thống sẽ bị hạn chế bởi băng thông của kênh truyền và hầu hết các đơn vị xử lý trung tâm sẽ lãng phí thời gian chờ để đến lượt đọc/ghi ô nhớ. Nếu chỉ có hai hoặc ba đơn vị xử lý trung tâm thì việc quản lý tương tranh khá đơn giản, vấn đề sẽ trở nên khá phức tạp khi số lượng của chúng tăng lên. Để giải quyết vấn đề này, người ta xây dựng kiến trúc truy nhập bộ nhớ không đồng nhất bằng cách thêm vùng đệm vào bo mạch của bộ vi xử lý. Vùng đệm trao đổi thông tin với bộ nhớ dùng chung theo phương pháp đọc/ghi từng khối 32 hoặc 64 byte, đơn vị xử lý trung tâm sẽ đọc/ghi các ô nhớ trong vùng đệm, như vậy sẽ tăng hiệu năng sử dụng kênh truyền.

Để mở rộng vùng đệm, có thể thêm bộ nhớ cục bộ được truy nhập bằng kênh riêng cho mỗi bộ vi xử lý, như vậy cần giải quyết vấn đề nhân bản ô nhớ. Mỗi khối ô nhớ được đánh dấu trạng thái đọc hoặc ghi, nếu đơn vị xử lý trung tâm muốn ghi một ô nhớ được nhân bản ở bộ nhớ riêng thì phần cứng của kênh truyền sẽ gửi tín hiệu ghi đến các bộ nhớ này. Trước khi thay đổi ô nhớ trong vùng đệm thì phải lưu lại giá trị cũ vào bộ nhớ dùng chung hoặc chuyển trực tiếp giá trị đó đến bộ điều khiển ghi qua kênh truyền. Để sử dụng một cách tối ưu, bộ nhớ riêng sẽ lưu mã chương trình, các hằng số, dữ liệu chỉ đọc và các biến số riêng, bộ nhớ dùng chung sẽ được cấp phát cho các biến số khác.

Công nghệ này cho phép người dùng bắt đầu với những máy chủ tương đối nhỏ và sau đó bổ sung bộ vi xử lý nếu cần mở rộng. Nếu sử dụng kiến trúc truy nhập bộ nhớ đồng nhất thì việc bổ sung thêm bộ vi xử lý không làm tăng hiệu năng như kỳ vọng, kiến trúc truy nhập bộ nhớ không đồng nhất khắc phục nhược điểm này và cho phép mở rộng lên đến 256 bộ vi xử lý trên một máy. Thông thường các bộ xử lý được phân thành những nhóm nhỏ, trong đó tất cả các bộ vi xử lý liên kết với nhau và có bộ nhớ riêng cho mỗi nhóm, nó làm giảm tình trạng tắc nghẽn vì các bộ vi xử lý trong một nhóm giao tiếp với nhau và với bộ nhớ cục bộ của chúng qua những kênh riêng.

### 1.2.2 Hệ thống nhiều máy tính

Hệ thống nhiều bộ vi xử lý đem lại hiệu năng xử lý rất tốt trên một máy tính nhưng mới chỉ giải quyết được vấn đề về tốc độ xử lý, hơn nữa rất khó tăng thêm số lượng bộ vi xử lý và không gian lưu trữ dữ liệu. Lắp đặt hệ thống gồm nhiều máy tính khá đơn giản, bộ vi xử lý trên mỗi máy tính sở hữu không gian bộ nhớ riêng, vấn đề còn lại cần phải giải quyết là việc trao đổi thông tin giữa chúng, đây là công việc vô cùng phức tạp. Giải pháp duy nhất cho đến nay là các máy tính được kết nối mạng, khi đó tốc độ trao đổi thông tin sẽ nhỏ hơn và thời gian trễ lớn hơn rất nhiều so với việc đọc/ghi bộ nhớ, thời gian truy nhập ô nhớ trên một máy tính chỉ dưới 1 nano giây trong khi nếu truy nhập ô nhớ trên máy tính khác phải cộng thêm trễ kênh truyền, giá trị của nó thường lên tới vài chục mili giây.

Tốc độ truyền dữ liệu trong mạng cục bộ phụ thuộc vào hình trạng mạng, các máy tính trong một vùng xung đột sẽ phải chia sẻ kênh truyền vật lý. Trong mạng cục bộ, nếu sử dụng kênh truyền dạng bus hoặc sử dụng thiết bị tập trung để kết nối các máy tính với nhau sẽ hình thành vùng xung đột, tại mỗi thời điểm chỉ cho phép 01 máy tính được gửi dữ liệu. Giải pháp tốt hơn là đấu nối các máy tính theo hình sao băng cách sử dụng thiết bị chuyển mạch, khi đó mỗi cổng của bộ chuyển mạch là một vùng xung đột và do đó có thể sử dụng tối đa băng thông. Băng thông của các kênh kết nối trên mạng diện rộng thường thấp hơn rất nhiều so với mạng cục bộ, dữ liệu phải đi qua nhiều thiết bị trung gian khác nhau và phải chi phí cho vấn đề đóng gói/bóc tách dữ liệu, chất lượng kênh truyền thấp và không ổn định..., tất cả những vấn đề đó làm tăng thời gian trễ.

Như vậy, độ trễ là điểm yếu cơ bản so với hệ thống nhiều bộ vi xử lý nhưng hệ thống nhiều máy tính lại thể hiện hàng loạt ưu điểm vượt trội và do đó đây là giải pháp phổ biến được lựa chọn để xây dựng các hệ thống phân tán. Ưu điểm thứ nhất phải kể đến là vấn đề giá thành, thay vì đầu tư một máy chủ với rất nhiều bộ vi xử lý với giá thành cao thì chỉ cần lắp đặt vài máy chủ với cấu hình vài bộ vi xử lý nhưng giá thành rẻ hơn. Ưu điểm thứ hai là vấn đề riêng tư, dữ liệu lưu trữ được quản lý độc lập trên mỗi máy tính nhưng vẫn có thể chia sẻ cho nhau. Ưu điểm thứ ba là khả năng dự phòng, nếu phần mềm có khả năng tự chuyển sang máy chủ khác khi gặp lỗi thì dịch vụ sẽ không bị gián đoạn hoặc chỉ ảnh hưởng một phần hệ thống. Ưu điểm thứ tư là khả năng mở rộng qui mô về địa lý cũng như số lượng người sử dụng, chỉ cần lắp đặt thêm máy chủ và vài thiết bị mạng trong khi hệ thống vẫn đang vận hành. Ưu điểm cuối cùng là tính linh hoạt, dễ dàng nâng cấp các phiên bản phần mềm hoặc tích hợp với các hệ thống khác mà không cần phải đầu tư thêm phần cứng.

### 1.3 Quan điểm về phần mềm

Phần cứng đóng vai trò quan trọng nhưng phần mềm đóng vai trò quan trọng hơn, sự hoạt động của máy tính hoàn toàn do phần mềm quyết định. Điều này cũng không phải là ngoại lệ đối với hệ thống phân tán, phần mềm được cài đặt trên các máy tính, chúng trao đổi thông tin qua môi trường mạng để phối hợp chặt chẽ với nhau như một thể thống nhất. Mỗi thiết bị trong hệ thống phân tán đều phải cài đặt hệ điều hành để làm nền tảng chạy các phần mềm ứng dụng.

#### 1.3.1 Hệ điều hành

Hệ điều hành là một phần mềm đặc biệt dùng để điều hành và quản lý toàn bộ tất cả thành phần bao gồm cả phần cứng lẫn phần mềm ứng dụng, nó đóng vai trò nền tảng cung cấp các dịch vụ truy nhập tài nguyên. Hệ điều hành đơn giản hóa sự phức tạp và đa dạng của phần cứng bằng kỹ thuật ảo hóa, như vậy nếu phần cứng được nâng cấp hoặc sản phẩm của các hãng khác nhau thì không cần phải thay đổi phần mềm ứng dụng. Hệ điều hành được thiết kế để phù hợp với nhiều chủng loại thiết bị khác nhau, từ máy tính cho đến điện thoại thông minh hoặc các thiết bị trung gian như thiết bị định tuyến và thiết bị chuyển mạch. Dựa trên cách xử lý trong môi trường phân tán người ta chia làm hai loại, hệ điều hành phân tán và hệ điều hành mạng, bảng 1.1 tóm tắt đặc điểm của những hệ điều hành này.

**Bảng 1.1** Hệ điều hành phân tán và hệ điều hành mạng

Hệ thống	Mô tả	Mục tiêu
Hệ điều hành phân tán	Các máy tính liên kết chặt chẽ với nhau, thường dùng cho các hệ thống đồng nhất	Quản lý và che giấu các tài nguyên phần cứng trên các máy tính khác nhau
Hệ điều hành mạng	Liên kết giữa các máy tính tương đối lỏng, thường dùng cho các hệ thống không đồng nhất	Cung cấp các dịch vụ cục bộ cho các máy tính khác truy nhập từ xa.

Hệ điều hành phân tán là phần mềm chạy trên tập các thiết bị độc lập được kết nối với nhau qua mạng, nó được cài đặt trên các máy tính khác nhau để tạo thành một cụm và vận hành như một máy tính có cấu hình mạnh hơn, ví dụ điển hình là sản phẩm LOCUS đã được đại học UCLA phát triển từ năm 1980 đến năm 1983. Như vậy hệ điều hành phân tán quản lý tổng thể tất cả các máy tính trong hệ thống, mỗi máy tính đặt ở những vị trí riêng biệt và được kết nối qua mạng nhưng hoạt động của chúng được gắn kết chặt chẽ để cung cấp các dịch vụ của hệ điều hành. Loại hệ điều hành này thường được cài đặt trong các hệ thống đồng nhất, thường là cụm máy chủ, nó mang lại những lợi ích về hiệu năng và độ tin cậy cho hệ thống. Nâng cao hiệu năng bằng cách phân rã một yêu cầu tính toán thành nhiều nhiệm vụ nhỏ hơn để thực hiện trên các máy tính khác nhau, do đó tăng tốc độ tính toán. Nâng cao hiệu năng cũng có thể được thể hiện dưới hình thức cân bằng tải, một yêu cầu sẽ được chuyển đến máy tính thích hợp để xử lý, có thể là máy tính đang ít yêu cầu trong hàng đợi hoặc máy chủ chuyên dụng. Độ tin cậy của hệ thống được tăng lên dựa trên cơ chế dự phòng nóng, nếu một máy tính bị hỏng thì đã có những máy tính khác xử lý, do đó không ảnh hưởng đến hoạt động của toàn bộ hệ thống.

Hệ điều hành mạng thường dùng cho các hệ thống không đồng nhất, nó cung cấp các tính năng để có thể kết nối mạng, mỗi máy tính tạo ra các dịch vụ cung cấp cho các máy tính khác. Ngoài các chức năng cơ bản của một hệ điều hành, các hệ điều hành mạng còn phải thực hiện việc chia sẻ và bảo vệ tài nguyên của mạng. Các hệ điều hành Microsoft Server, Linux, Unix được xếp vào loại hệ điều hành mạng, máy chủ quản lý tài nguyên của mình để cung cấp dịch vụ cho các máy khách. Quan hệ giữa các máy tính không chặt chẽ và sự gắn kết trong hệ thống do các ứng dụng phân tán đảm nhiệm, một số các dịch vụ chung do hệ điều hành cung cấp cho các ứng dụng phân tán gọi là phần mềm trung gian. Phần mềm trung gian thường được triển khai dưới dạng thư viện, những thư viện này cung cấp các phương tiện trao đổi thông tin dựa trên thông điệp.

Nếu máy tính chỉ lắp đặt một bộ vi xử lý thì nhiệm vụ của hệ điều hành chỉ đơn giản là quản lý tiến trình và ảo hóa tài nguyên để các phần mềm ứng dụng dễ dàng truy nhập. Các tiến trình sử dụng tài nguyên chung nhưng vẫn đảm bảo tính độc lập, do đó hệ điều hành cần phải giải quyết vấn đề tương tranh và ảo hóa tài nguyên là cách tiếp cận đơn giản nhất, nó cung cấp khả năng xử lý đa nhiệm cho các ứng dụng. Bên trong hệ điều hành thường có phần lõi, nó quản lý phần cứng, lập lịch cho các tiến trình và xử lý các yêu cầu tương tác giữa phần cứng và phần mềm, nghĩa là có thể truy nhập trực tiếp vào

bộ nhớ và các thanh ghi. Dựa trên các hàm do phần lõi cung cấp sẽ phát triển thêm các hàm giao diện lập trình để hạn chế truy nhập trực tiếp vào các thanh ghi hoặc bộ nhớ.

Phần mềm ứng dụng thường truy nhập đến tài nguyên thông qua các hàm do hệ điều hành cung cấp, chúng có thể thuộc loại phong tỏa không phong tỏa. Hàm phong tỏa nghĩa là chương trình gọi sẽ phải chờ kết quả thực hiện, nó không cho phép xử lý song song, như vậy toàn bộ tiến trình cũng sẽ bị phong tỏa. Hàm không phong tỏa cho phép chương trình gọi tiếp tục thực hiện mà không cần chờ đợi lời gọi hoàn thành, đây là điểm quan trọng để có thể áp dụng kỹ thuật lập trình song song nhằm nâng cao hiệu năng xử lý.

Đối với hệ thống gồm nhiều bộ vi xử lý, hệ điều hành có thể được thiết kế đối xứng hoặc bất đối xứng. Nếu thiết kế đối xứng thì mỗi tiến trình có thể chạy trên bất kỳ bộ vi xử lý nào, các bộ vi xử lý trao đổi thông tin qua một bộ nhớ dùng chung. Ưu điểm của nó là tính chịu lỗi và khả năng cân bằng tải tối ưu hơn, vì các tiến trình của hệ điều hành có thể chạy trên bất kỳ bộ xử lý nào nên nguy cơ tắc nghẽn giảm đi đáng kể. Tuy nhiên vấn đề đồng bộ giữa các bộ vi xử lý được đặt lên hàng đầu khi thiết kế hệ điều hành cho hệ thống đối xứng. Trong thiết kế bất đối xứng, hệ điều hành dành ra một hoặc hai bộ xử lý để sử dụng riêng, các bộ xử lý còn lại dùng để điều khiển các chương trình của người sử dụng. Hệ thống bất đối xứng đơn giản hơn nhiều so với hệ thống đối xứng, nhưng nếu bộ vi xử lý dành riêng cho hệ điều hành bị hỏng thì hệ thống có thể ngừng hoạt động.

### **1.3.2 Phần mềm ứng dụng**

Phần cứng cũng như hệ điều hành cài đặt trên các thiết bị của hệ thống phân tán không nhất thiết phải quan hệ chặt chẽ với nhau, không cần phải có sự phối hợp cộng tác ngoài việc tuân thủ qui tắc giao thức khi trao đổi thông tin. Bước phát triển tiếp theo là xây dựng phần mềm ứng dụng, chúng phải gắn kết chặt chẽ với nhau như một thực thể thống nhất chứ không phải là một tập hợp các máy tính. Phần mềm hệ thống phân tán phải thân thiện với người sử dụng, họ có thể thực hiện các thao tác giống như trên máy tính của mình. Nói cách khác, cần phải đảm bảo các máy tính phối hợp với nhau và hoạt động như một máy ảo với một bộ xử lý duy nhất, người sử dụng không có cảm giác về sự tồn tại của nhiều máy tính.

Khác với các ứng dụng chạy độc lập trên một máy tính, số lượng người sử dụng trên hệ thống phân tán rất lớn, do đó việc đảm bảo hiệu năng là một trong những mục tiêu quan trọng khi xây dựng hệ thống. Các yêu cầu của người sử dụng thường được thực hiện trên các máy tính khác nhau, dữ liệu lại thường xuyên phải di chuyển qua mạng..., những vấn đề đó đòi hỏi các ứng dụng phải đảm bảo tính an toàn và bảo mật thông tin. Tóm lại, nếu chỉ tuân thủ các yêu cầu của giao thức truyền thông thì chưa đủ, rất nhiều vấn đề cần phải giải quyết khi xây dựng các ứng dụng phân tán, đây cũng là trọng tâm được đề cập trong những phần tiếp theo của tài liệu này.

## **1.4 Phân loại hệ thống phân tán**

Hệ thống thông tin là tập hợp các yếu tố có mối quan hệ mật thiết với nhau cùng thực hiện các nhiệm vụ xử lý dữ liệu để đạt được mục đích cụ thể cho người sử dụng,

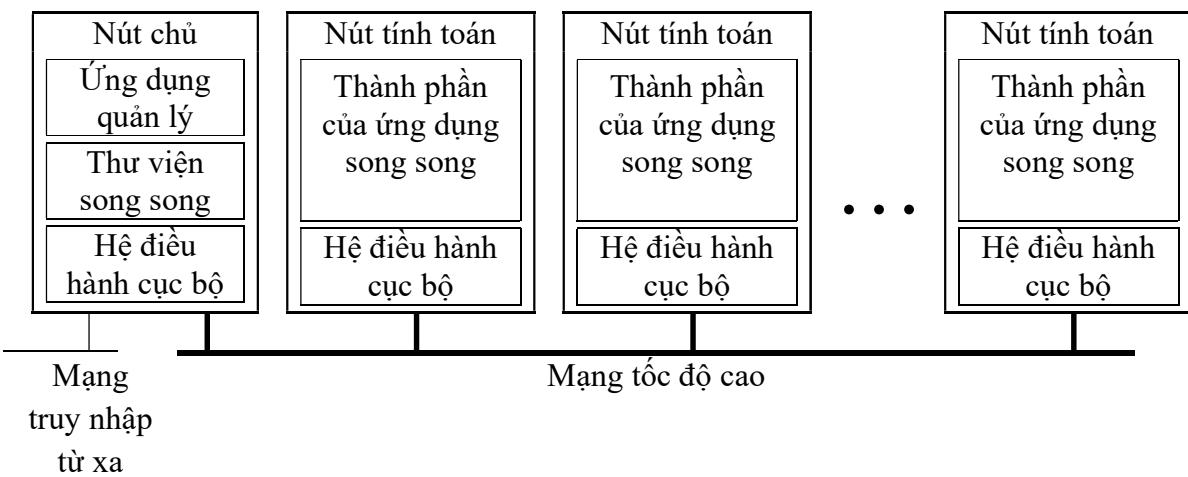
việc phân loại là cơ sở để phân tích và lựa chọn giải pháp phù hợp để phát triển hệ thống. Xét cho cùng, hệ thống phân tán cũng là một hệ thống thông tin, điểm làm nên khác biệt của nó là các thiết bị được nối mạng và việc lưu trữ dữ liệu cũng như các thao tác xử lý được thực hiện trên các thiết bị khác nhau. Dựa trên những đặc điểm này có thể chia hệ thống phân tán thành ba loại: điện toán phân tán, thông tin phân tán và lan tỏa phân tán.

### 1.4.1 Các hệ thống điện toán phân tán

Một trong những vấn đề cơ bản của hệ thống phân tán là phải đảm bảo vấn đề hiệu năng, tính toán phân tán thường được sử dụng trong các tác nghiệp yêu cầu hiệu năng cao. Tùy theo nền tảng hệ thống, điện toán phân tán có thể triển khai theo hình thức cụm hoặc lưới, dù theo hình thức nào thì điểm cốt lõi của nó vẫn là việc phân chia nhiệm vụ để xử lý trên nhiều máy tính. Trong hệ thống điện toán cụm, nền tảng phần cứng là tập hợp các máy tính tương tự nhau, chúng được cài đặt các hệ điều hành giống nhau và kết nối với nhau qua đường truyền tốc độ cao, thông thường là mạng cục bộ. Ngược lại, hệ thống điện toán lưới bao gồm các hệ thống phân tán thuộc về nhiều miền quản lý khác nhau và thường không đồng nhất về phần cứng, phần mềm, hệ điều hành và các hệ thống này có thể sử dụng những công nghệ mạng khác nhau.

#### 1.4.1.1 Hệ thống điện toán cụm

Giá thành máy tính ngày càng giảm trong khi khả năng xử lý ngày càng tăng là cơ hội để xây dựng hệ thống điện toán cụm, các máy tính được kết nối với nhau trong mạng tốc độ cao. Nguyên lý của nó vẫn là sử dụng kỹ thuật xử lý song song trên nhiều máy tính để tăng hiệu năng xử lý, người sử dụng nhìn nhận điện toán cụm như một máy tính ảo với hiệu năng cao. Trên mỗi máy tính có thể sử dụng hệ điều hành phân tán để liên kết các máy tính, nhưng cũng có thể chỉ cần cài đặt hệ điều hành mạng và chạy phần mềm hỗ trợ điện toán cụm.



Hình 1.2 Hệ thống điện toán cụm bất đối xứng

Một ví dụ khá quen biết của điện toán cụm là hệ thống Beowulf chạy trên hệ điều hành Linux, nó được thiết kế bất đối xứng, nghĩa là tồn tại nút chủ đảm nhiệm chức năng sắp đặt vị trí của các thành viên khác trong chương trình xử lý song song, quản lý hàng đợi các công việc và giao tiếp với người sử dụng. Hình 1.2 thể hiện loại hệ điều hành

này, nút chủ chỉ việc chạy phần mềm trung gian cần thiết cho các chương trình thực hiện và quản lý cụm, trong khi đó các nút tính toán sẽ không cần gì khác ngoài hệ điều hành chuẩn.

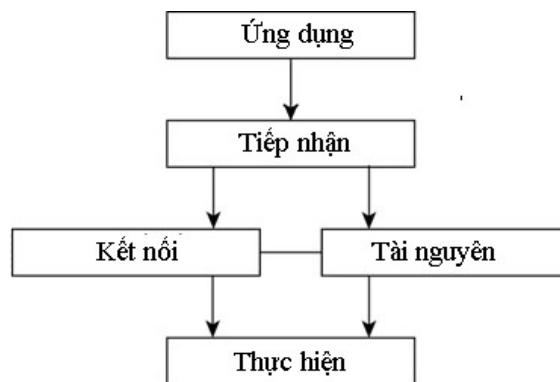
Một ví dụ khác là hệ thống MOSIX được xây dựng dựa trên cách tiếp cận đối xứng, nghĩa là không có máy tính nào đóng vai trò nút chủ. Độ trung suốt rất cao của MOSIX đạt được bằng cách di trú tiến trình, người sử dụng có thể khởi tạo tiến trình trên một máy tính nhưng tiến trình đó có thể di trú sang máy tính khác để thực hiện nhằm tận dụng tối đa tiềm năng xử lý của mỗi nút trong hệ thống.

Dù sử dụng cách nào thì điều kiện tiên quyết để có thể triển khai hệ thống điện toán cụm là nền tảng phải đồng nhất với kênh truyền tốc độ cao, nền tảng ở đây bao gồm phần cứng cũng như hệ điều hành và phần mềm hỗ trợ. Điện toán cụm thường áp dụng cho các trung tâm tính toán lớn, các máy tính kết nối với nhau trong mạng cục bộ đảm bảo kênh truyền tốc độ cao và ổn định.

#### 1.4.1.2 Hệ thống điện toán lưới

Không phải lúc nào cũng đảm bảo tính đồng nhất về hạ tầng để cài đặt hệ thống điện toán cụm, nếu nhìn xa hơn sẽ thấy điện toán cụm vẫn rơi vào cách xử lý tập trung, dù cho hiệu năng của nó rất cao nhưng vẫn không tránh khỏi việc tập trung các yêu cầu về một địa điểm, điều này có thể dẫn tới nghẽn mạng. Hơn nữa, việc giải quyết các yêu cầu của người sử dụng có thể liên quan đến nhiều đơn vị khác nhau, không thể bắt buộc máy tính của các đơn vị này cài đặt cùng một loại hệ điều hành và kết nối với nhau qua kênh truyền tốc độ cao, đó là chưa tính đến mỗi cơ quan lại có những chính sách quản lý riêng.

Điện toán lưới là mô hình tính toán dựa trên mạng để có khả năng xử lý một lượng lớn dữ liệu, một nhóm các máy tính phối hợp với nhau để giải quyết vấn đề chung. Tương tự như điện toán cụm, các máy tính trong hệ thống điện toán lưới liên kết với nhau để giải quyết một vấn đề chung bằng cách chia thành các đơn vị nhỏ hơn để xử lý, chúng hình thành một siêu máy tính ảo. Hệ thống điện toán lưới không đòi hỏi tính đồng nhất của tất cả các nút, mỗi thành viên có thể khác về cả phần cứng lẫn hệ điều hành và các chính sách quản lý.



Hình 1.3 Mô hình phân tầng hệ thống điện toán lưới

Vấn đề cốt lõi trong hệ thống điện toán lưới là việc quản lý tài nguyên của các cơ quan khác nhau nhưng phải cho phép các nhóm người sử dụng thuộc các cơ quan cộng tác với nhau, như vậy sự cộng tác được thực hiện dựa trên cơ quan ảo, người sử dụng thuộc về một cơ quan ảo thì có quyền truy nhập đến các tài nguyên của cơ quan ảo đó. Với đặc tính đó, nhiều phần mềm hệ thống điện toán lưới phát triển xung quanh việc truy nhập tài nguyên từ các vùng quản trị khác nhau cho người sử dụng và ứng dụng thuộc về một cơ quan ảo, do đó tiêu điểm của hệ thống điện toán lưới thường là những vấn đề liên quan đến mô hình hệ thống.

Hình 1.3 minh họa cách tổ chức hệ thống điện toán lưới, nó thường được xây dựng theo mô hình phân tầng, mỗi tầng thực hiện những chức năng riêng biệt. Tầng ứng dụng bao gồm các ứng dụng vận hành bên trong cơ quan ảo và sử dụng môi trường điện toán lưới. Tầng tiếp nhận xử lý các yêu cầu truy nhập đến nhiều tài nguyên khác nhau, thường cung cấp các chức năng như: thăm dò, định vị, lập lịch truy nhập, nhân bản tài nguyên... Các giao thức thuộc tầng này khá nhiều, để đảm bảo cung cấp dịch vụ theo yêu cầu của tầng ứng dụng nên chúng thường không phải là những giao thức đã được chuẩn hóa.

Tầng kết nối bao gồm các giao thức truyền thông để hỗ trợ cho các giao tác lưới bao trùm toàn bộ các tài nguyên, ví dụ các giao thức truy nhập để di chuyển tài nguyên hoặc đơn giản chỉ là truy nhập tài nguyên từ một vị trí nào đó. Tầng kết nối sẽ phải bao gồm các giao thức bảo mật, tính năng bảo mật có thể cho một tài khoản và cũng có thể cho một ứng dụng.

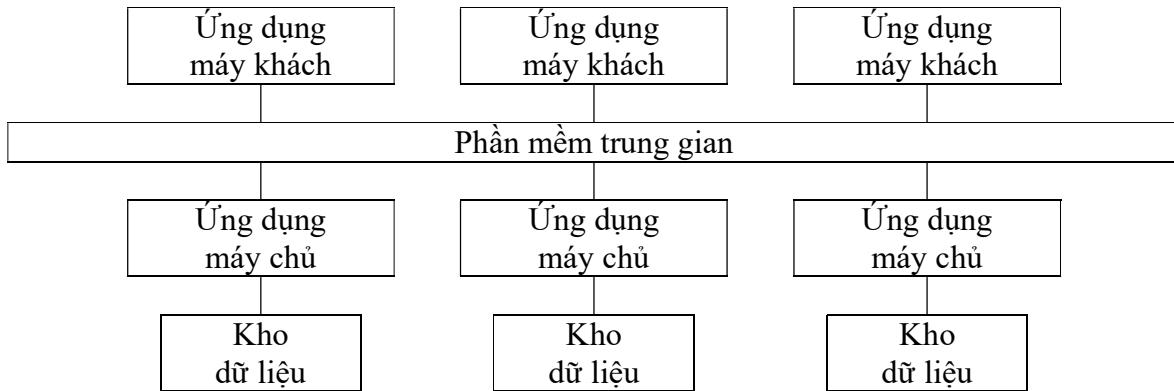
Tầng tài nguyên quản lý tài nguyên đơn lẻ, nó sử dụng các chức năng do tầng kết nối cung cấp và gọi trực tiếp các giao diện tầng thực hiện cung cấp để đảm bảo các chức năng điều khiển truy nhập, ví dụ các chức năng thiết lập cấu hình tài nguyên, khởi tạo tiến trình đọc/ghi dữ liệu. Tầng thực hiện cung cấp giao diện để truy nhập tài nguyên cục bộ, các giao diện này được xây dựng để có thể thích ứng với việc cho phép chia sẻ tài nguyên bên trong một cơ quan ảo, nó thường cung cấp các chức năng để truy vấn trạng thái và khả năng của tài nguyên, các chức năng quản lý tài nguyên thực.

Trong các hệ thống điện toán lưới, tầng tiếp nhận, tầng kết nối và tầng tài nguyên thường được gộp lại vào tầng trung gian, nó có nhiệm vụ quản lý và cung cấp chức năng truy nhập trong suốt đến tất cả các tài nguyên phân bố trên các trang mạng khác nhau. Quan sát cho thấy, việc cung cấp các thông tin riêng lẻ trong các hệ thống điện toán lưới khá phổ biến, điều này đã dần dẫn tới quan điểm về mô hình dịch vụ lưới mở.

#### **1.4.2 Hệ thống thông tin phân tán**

Thông tin phân tán là thực tế khá phổ biến trong hệ thống phân tán, dữ liệu được lưu trữ trên nhiều máy tính, xử lý dữ liệu trên một máy có thể sẽ liên quan đến những máy khác. Việc dữ liệu được lưu trữ phân tán xuất phát từ nhiều nguyên nhân, có thể do dung lượng dữ liệu lớn khi qui mô phát triển hoặc triển khai kỹ thuật nhân bản để tăng hiệu năng và đảm bảo tính tin cậy. Các phần mềm ngày nay thường được xây dựng theo mô hình ba bên, tách biệt các thành phần lưu trữ dữ liệu với xử lý nghiệp vụ và tiếp nhận yêu cầu.

Nếu dữ liệu được lưu trữ tập trung, khi nhận được yêu cầu từ máy khách thì máy chủ dịch vụ chỉ cần gửi yêu cầu truy vấn đến một máy chủ cơ sở dữ liệu, kết quả nhận được sẽ xử lý và trả về cho máy khách. Vấn đề sẽ phức tạp hơn nếu dữ liệu được nhân bản, các yêu cầu đọc sẽ không có nhiều điểm khác biệt, dù cho đôi khi phải thực hiện đọc phân tán, nhưng các yêu cầu ghi lại khác hoàn toàn, thay đổi dữ liệu phải được thực hiện thống nhất trên tất cả các bản sao.



Hình 1.4 Phần mềm trung gian trong hệ thống thông tin phân tán

Nếu máy khách truy nhập đến từng ứng dụng trên máy chủ để thực hiện trên các thao tác dữ liệu như các hệ thống hỏi/đáp thì sẽ rất phức tạp, thời gian thực hiện kéo dài và thậm chí khó có thể đạt được mục tiêu nhất quán. Hình 1.4 thể hiện cách truy nhập dữ liệu trong các hệ thống thông tin phân tán, đưa các yêu cầu truy vấn vào tầng trung gian sẽ đơn giản hóa cách phát triển hệ thống, tầng trung gian sẽ quyết định gửi yêu cầu đến những máy chủ nào và tổng hợp kết quả trả về cho máy khách. Nếu là yêu cầu đọc, nó có thể quyết định xem có cần đọc phân tán hay không, kỹ thuật lưu trữ cache cũng có thể được áp dụng để giảm thời gian xử lý và qua đó cải thiện vấn đề hiệu năng. Nếu là yêu cầu ghi thì phần mềm trung gian sẽ lựa chọn mô hình nhân bản để cân bằng tính nhất quán với các vấn đề liên quan thời gian đáp ứng yêu cầu của người sử dụng.

#### 1.4.3 Hệ thống lan tỏa phân tán

Các máy chủ thường được đặt ở vị trí địa lý cố định và kết nối với nhau qua kênh truyền chất lượng cao, tính ổn định này tạo điều kiện thuận lợi để đạt được tính trong suốt trong xử lý cũng như truy nhập tài nguyên. Các kỹ thuật xử lý phân tán và nhân bản dữ liệu không những nâng cao hiệu năng hệ thống mà còn có thể che giấu lỗi, nếu xảy ra trên một máy chủ nào đó. Vấn đề còn lại là các máy khách kết nối đến hệ thống máy chủ như thế nào và cách truyền dữ liệu ra sao.

Vấn đề trở nên phức tạp khi xuất hiện nhiều chủng loại thiết bị trong các gia đình, thậm chí các thiết bị di động ngày càng phổ biến, chúng thường được kết nối qua môi trường không dây với đặc trưng là tính không ổn định về vị trí cũng như tốc độ truyền dẫn. Hệ thống này thiếu vắng sự kiểm soát nhân công, cấu hình của các thiết bị do chủ sở hữu qui định hoặc thiết bị sẽ tự động khám phá môi trường và lựa chọn cấu hình phù hợp nhất. Môi trường như vậy gọi là lan tỏa phân tán, để tạo điều kiện lựa chọn cấu hình

chính xác nhất, Grimm đã đề xuất các thiết bị phải có khả năng bao quát ngữ cảnh, cung cấp giao diện cấu hình mặc định và tự động nhận biết chia sẻ tài nguyên.

Bao quát thay đổi ngữ cảnh nghĩa là thiết bị phải liên tục nhận biết được môi trường có thể thay đổi bất kỳ thời gian nào, ví dụ khi phát hiện mất kết nối mạng thì thiết bị sẽ tự động tìm mạng khác thay thế. Cung cấp giao diện cấu hình mặc định nghĩa là mỗi người sử dụng có thói quen riêng biệt, do đó cần phải cung cấp giao diện cấu hình sao cho đơn giản nhất phù hợp với tất cả mọi người hoặc một cấu hình được cài đặt tự động.

Tự động nhận biết chia sẻ tài nguyên là đặc điểm quan trọng của hệ thống lan tỏa, các thiết bị tham gia hệ thống theo thứ tự truy nhập thông tin, điều này đòi hỏi phải cung cấp các phương tiện để dễ dàng đọc, lưu trữ, quản lý và chia sẻ thông tin. Với quan niệm việc kết nối mạng của các thiết bị thường gián đoạn hoặc thay đổi thì không gian lưu trữ thông tin có thể truy nhập được cũng sẽ phải thay đổi theo thời gian. Với sự hiện diện của khả năng di động thì các thiết bị phải dễ dàng thích nghi với môi trường cục bộ, chúng phải có khả năng dễ dàng phát hiện các dịch vụ và phản hồi theo các dịch vụ đó. Sự trong suốt về mặt phân bố không chỉ ở trong hệ thống lan tỏa mà thực tế còn ở sự phân bố về dữ liệu, xử lý và điều khiển, vì lẽ đó tốt hơn hết là phải phơi bày chứ không nên che giấu chúng.

Cùng với xu thế Internet vạn vật, hệ thống lan tỏa phát triển ngày càng mạnh mẽ, ví dụ hệ thống thiết bị trong gia dụng, hệ thống chăm sóc sức khỏe, mạng cảm biến..., khó có thể yêu cầu người dùng cài đặt phần mềm trên những thiết bị này, chúng phải được thiết kế sao cho có thể tự động nhận biết được môi trường xung quanh. Lượng dữ liệu thu được từ các thiết bị này rất lớn, giải pháp lắp đặt cho mỗi thiết bị một bộ lưu trữ lên tới vài TB là không khả thi, cho dù có giải quyết được vấn đề lưu trữ thì vẫn gặp phải một loạt những vấn đề khác, tùy từng hệ thống cần phải xem xét các vấn đề sau:

- Bảo mật tính riêng tư của thông tin.
- Dữ liệu được lưu trữ ở đâu và bằng cách nào có thể khai thác được chúng.
- Các biện pháp phòng chống mất mát dữ liệu quan trọng.
- Hạ tầng cần thiết để tạo và lan tỏa các cảnh báo.
- Phương pháp phản hồi và kiểm soát trực tuyến các thiết bị.
- Sử dụng giải pháp nào để triển khai một lượng rất lớn các thiết bị.
- Giải pháp khắc phục khi gặp sự cố về đường truyền mạng.

Đối với các thiết bị gia dụng, vấn đề ưu tiên hàng đầu là các thiết bị phải có khả năng tự cấu hình quản lý và tính riêng tư của thông tin, ví dụ hệ thống chăm sóc sức khỏe lại đặt vấn đề cảnh báo và phản hồi trực tuyến là quan trọng, trong khi đó hệ thống cảm biến lại phải đương đầu với số lượng rất lớn các thiết bị. Lưu trữ dữ liệu cục bộ tạo điều kiện cho thiết bị hoạt động tương đối độc lập với môi trường mạng, tuy nhiên vấn đề khai thác dữ liệu sẽ trở nên phức tạp, hơn nữa nó đòi hỏi không gian lưu trữ đủ lớn trên mỗi thiết bị.

#### **1.4.4 Điện toán đám mây**

Những năm gần đây, khái niệm điện toán đám mây được nói nhiều trong cộng đồng xã hội, đó không phải là công nghệ mới, nó chỉ là sự thay đổi trong cách xây dựng các hệ thống thông tin, bản chất của nó là cung cấp các dịch vụ tiện ích dựa trên nền tảng điện toán phân tán. Với chi phí ban đầu rất thấp, điện toán đám mây được triển khai phổ biến cho cộng đồng người sử dụng là sự thật đã rất rõ ràng, nhưng áp dụng cho các doanh nghiệp lại là điều đáng phải cân nhắc. Câu hỏi về độ trễ xử lý và vấn đề an toàn bảo mật thông tin vẫn đang được thảo luận, xét cho cùng thì dữ liệu là tài sản đáng giá nhất của các doanh nghiệp.

Ý tưởng về điện toán đám mây đã được Jcr Licklider đề xuất trong một bài báo về “mạng máy tính giữa các thiên hà” vào năm 1969, chỉ đến khi Internet băng thông rộng phát triển mạnh mẽ thì điện toán đám mây mới có những thành tựu đáng kể. Một trong những cột mốc đầu tiên cho điện toán đám mây là sự xuất hiện của Salesforce.com vào năm 1999, tiên phong trong khái niệm các ứng dụng doanh nghiệp cung cấp thông qua một trang web đơn giản. Sự phát triển tiếp theo là Amazon Web Services vào năm 2002, trong đó cung cấp một bộ các dịch vụ dựa trên đám mây bao gồm lưu trữ, tính toán và ngay cả trí tuệ nhân tạo thông qua Amazon Mechanical Turk. Năm 2006, Amazon ra mắt điện toán đám mây Elastic Compute cung cấp dịch vụ web thương mại cho phép các công ty nhỏ hoặc cá nhân thuê để chạy các ứng dụng của mình. Năm 2009 đánh dấu một bước tiến triển lớn, một số hãng công nghệ lớn bắt đầu cung cấp các ứng dụng doanh nghiệp dựa trên trình duyệt, ví dụ dịch vụ như Google Apps.

Điện toán đám mây và điện toán lưới khá tương đồng với nhau, cả hai đều cung cấp dịch vụ cho người dùng bằng cách chia sẻ tài nguyên của mình để thực hiện các tác vụ khác nhau. Tuy nhiên hai loại này cũng có những điểm khác biệt cơ bản, trong khi điện toán lưới ảo hóa các tài nguyên tính toán lưu trữ dữ liệu thì điện toán đám mây không cho phép người dùng truy nhập trực tiếp vào tài nguyên, việc truy nhập được thực hiện thông qua các dịch vụ trên nền tảng Internet. Điện toán lưới chia một tác vụ thành nhiều tác vụ nhỏ hơn để thực hiện trên các máy tính khác nhau, điện toán đám mây tạo ra một thể hiện mới cho người dùng với tài nguyên riêng bằng cách cung cấp dịch vụ cho người dùng đó. Điện toán lưới là tập các tài nguyên máy tính từ nhiều vị trí khác nhau để xử lý một tác vụ, đó là một hệ thống chia sẻ tài nguyên cộng tác, các tài nguyên được quản lý phân tán. Điện toán đám mây là một dạng tính toán dựa trên tài nguyên đã ảo hóa, đặt ở các vị trí khác nhau trong một cụm và được quản lý tập trung.

### **1.5 Mục tiêu hệ thống phân tán**

Không cần hệ thống phân tán vẫn có thể trao đổi thông tin, đơn giản chỉ cần sao chép dữ liệu vào đĩa và chuyển sang máy tính khác, tuy nhiên công việc đó không còn phù hợp với thời đại khi mạng máy tính đã hiện diện ở khắp nơi trên thế giới. Giờ đây, người sử dụng chỉ cần có máy tính hoặc điện thoại thông minh là có thể giao tiếp với người khác hoặc thực hiện các công việc từ xa, làm việc nhưng không cần hiện diện tại cơ quan đã trở nên phổ biến.

Giống như khi sử dụng các phần mềm chạy độc lập trên máy tính, người sử dụng luôn mong muốn các dịch vụ phân tán phải thân thiện, tốc độ xử lý nhanh, dễ dàng cá nhân hóa các chức năng và đảm bảo an toàn thông tin. Thiết kế là công việc đầu tiên khi bắt tay vào việc xây dựng bất kỳ hệ thống thông tin nào, thiết kế sai sẽ dẫn đến những hậu quả khó có thể khắc phục. Phần mềm hệ thống phân tán trao đổi thông tin với nhau qua mạng, đó là môi trường không đáng tin cậy, điều này làm này sinh rất nhiều công việc khi phát triển hệ thống. Để đáp ứng những yêu cầu của người sử dụng, thiết kế phải đảm bảo hệ thống phân tán đạt được các mục tiêu sau:

- **Thân thiện với người dùng:** Tính thân thiện không những thể hiện ở giao diện thuận lợi nhất mà còn phải đáp ứng các yêu cầu đầy đủ thông tin. Hệ thống phải đảm bảo tính tin cậy, thông tin phải chính xác và luôn sẵn sàng đáp ứng yêu cầu của người sử dụng.
- **Hiệu năng hệ thống:** Phải đảm bảo thời gian đáp ứng khi người dùng đưa ra yêu cầu, nó không phụ thuộc vào qui mô về không gian cũng như số lượng người dùng. Cần chú ý đưa ra các giải pháp hạn chế tối đa độ trễ trên các kênh truyền vật lý, khắc phục hiện tượng nút cỗ chai hoặc nghẽn mạng.
- **Linh hoạt:** Dễ dàng thay đổi dịch vụ, có khả năng mở rộng và di chuyển các thành phần cấu thành hệ thống phân tán, tính chuyên môn hóa cao và khả năng tương tác tốt giữa các thành phần trong hệ thống cũng như các hệ thống phân tán khác.
- **Nhất quán:** Thống nhất trong việc sử dụng dịch vụ, dữ liệu trên toàn hệ thống. Dữ liệu có thể phân tán ở nhiều vị trí khác nhau, việc xử lý có thể thực hiện trên các máy tính khác nhau nhưng luôn phải đảm bảo tính nhất quán cho người dùng.
- **Chịu lỗi:** Khi gặp lỗi, hệ thống có thể tự khởi tạo lại trạng thái tốt nhất và đồng thời đảm bảo không mất mát thông tin, ở mức độ cao hơn hệ thống có thể tự xử lý các trường hợp lỗi.
- **An toàn và bảo mật thông tin:** thông tin đáng tin cậy, tránh mất mát và lộ thông tin của người dùng, bảo vệ tốt hệ thống kiểm soát truy nhập để đảm bảo an ninh và an toàn cho hệ thống.

Mục tiêu quan trọng nhất của hệ thống phân tán là đảm bảo khả năng sẵn sàng của tài nguyên cho các dịch vụ, làm sao để người sử dụng có thể tiếp cận các tài nguyên đó một cách thuận tiện nhất. Hệ thống phân tán phải đảm bảo tính trong suốt, nghĩa là người sử dụng không cần quan tâm tới những thao tác xử lý bên trong mà chỉ cần quan tâm đến kết quả xử lý yêu cầu và những yêu cầu đó phải được thực hiện chính xác trong thời gian ngắn nhất có thể. Tiếp theo là những mục tiêu mang tính chất kỹ thuật, hệ thống phải có khả năng mở rộng về qui mô, có khả năng chịu lỗi, khả năng bảo mật cao, dễ dàng nâng cấp và bảo trì hệ thống.

### 1.5.1 Tính sẵn sàng

Mục tiêu chính của hệ thống phân tán là kết nối người sử dụng với tài nguyên hệ thống, người sử dụng được tiếp cận tài nguyên tại bất kỳ thời điểm nào theo cách dễ dàng nhất mà không phụ thuộc vị trí địa lý. Tài nguyên ở đây có thể hiểu mọi thứ có thể chia sẻ hoặc cung cấp dịch vụ, đó có thể là tài nguyên phần cứng như máy in, máy tính, các phương tiện lưu trữ... và cũng có thể đó là những dữ liệu cung cấp thông tin hữu ích cho người sử dụng.

Có nhiều nguyên nhân người sử dụng muốn chia sẻ tài nguyên, lý do rõ ràng nhất là vì mục tiêu kinh tế, ví dụ trong một đơn vị có thể chia sẻ máy in để tiết kiệm chi phí. Nguyên nhân thứ hai là nhu cầu trao đổi thông tin, đó có thể là những thông tin cộng tác trong công việc hoặc có thể là những thông tin phục vụ cho giải trí. Với việc phát triển mạnh mẽ của mạng Internet, các hệ thống thương mại điện tử ngày càng mở rộng, người sử dụng có thể thực hiện các công việc mà không nhất thiết phải hiện diện tại nơi làm việc.

Tuy nhiên, khi nhu cầu kết nối và chia sẻ tài nguyên ngày càng tăng thì vấn đề bảo mật càng trở nên quan trọng, như vậy nảy sinh hàng loạt vấn đề liên quan đến việc khai thác và sử dụng tài nguyên như ai được phép truy nhập, mức độ truy nhập, thời gian được phép truy nhập, tần suất truy nhập .... Thực tế cho thấy các hệ thống ít được bảo vệ chống lại những hiểm họa nghe trộm hoặc xâm nhập dữ liệu được gửi trên mạng, không mã hóa mật khẩu và thường lưu trong máy tính. Xét về khía cạnh bảo mật sẽ có rất nhiều công việc cần phải thực hiện, ví dụ các giao dịch thương mại điện tử, người sử dụng chỉ cần nhập vào số hiệu thẻ tín dụng, lẽ tất nhiên sẽ cần thiết phải chứng minh đó thực sự là chủ sở hữu thẻ.

Một vấn đề bảo mật khác là việc lưu vết truyền thông để xây dựng hồ sơ thói quen của người sử dụng, điều này vi phạm tính riêng tư và nó đặc biệt nghiêm trọng nếu như không thông báo cho người sử dụng. Việc các thông tin rác trên mạng cũng gây khó chịu cho người sử dụng, như vậy cần phải thiết kế hệ thống sao cho có thể lọc được những thông tin dựa trên nội dung nhưng không ảnh hưởng đến mục tiêu đảm bảo tính sẵn sàng tài nguyên của hệ thống.

### 1.5.2 Tính trong suốt

Tính trong suốt là che giấu sự thật các yêu cầu được xử lý như thế nào và lấy dữ liệu ở đâu khi chúng được lưu trữ trên nhiều máy tính, người sử dụng không cần biết vị trí thực sự của tài nguyên và yêu cầu được xử lý trên máy tính nào. Nếu một hệ thống có thể tự trình diễn tạo cảm giác cho người sử dụng như thể đang chạy trên một máy tính thì ta nói hệ thống đó đáp ứng yêu cầu về tính trong suốt. Tính trong suốt tạo cho người sử dụng cách nhìn nhận hệ thống một cách đơn giản, dễ sử dụng và góp phần đạt được mục tiêu thân thiện.

#### 1.5.2.1 Phân loại tính trong suốt

Tính trong suốt của hệ thống phân tán có thể được áp dụng cho nhiều khía cạnh, tổ chức tiêu chuẩn quốc tế đã đưa ra khuyến nghị các vấn đề sau:

- Truy nhập: Che giấu việc thể hiện dữ liệu và cách truy nhập tài nguyên.
- Vị trí: Che giấu nơi đặt tài nguyên của hệ thống.
- Di trú: Che giấu việc tài nguyên có thể di chuyển sang vị trí khác, khi đặt lại vị trí tài nguyên không làm gián đoạn hoạt động của hệ thống.
- Nhân bản: Che giấu tài nguyên được nhân bản.
- Tương tranh: Che giấu việc chia sẻ tài nguyên cho nhiều người đồng thời sử dụng.
- Lỗi: Che giấu lỗi và vấn đề phục hồi sau khi lỗi xảy ra.

Trong suốt về truy nhập giải quyết vấn đề che giấu những điểm khác biệt trong việc thể hiện và cách thức người sử dụng có thể truy nhập đến tài nguyên. Ở mức cơ bản cần phải che giấu những điểm khác biệt trong kiến trúc của các máy, nhưng điều quan trọng hơn là phải đạt được sự đồng thuận về cách thể hiện trên các nền tảng khác nhau. Ví dụ hệ thống phân tán không đồng nhất, các máy tính cài đặt những hệ điều hành khác nhau, qui tắc đặt tên cho các tập tin cho mỗi loại hệ điều hành tất nhiên cũng sẽ khác nhau, như vậy cần phải che giấu những điểm khác biệt này đối với người sử dụng.

Trong suốt về vị trí thực chất là che giấu người sử dụng nơi tài nguyên trú ngụ, trong một số trường hợp việc đặt tên đóng vai trò rất quan trọng để đạt được yêu cầu này. Gán các tên cho tài nguyên một cách thân thiện và công khai cho người sử dụng nhưng vẫn đảm bảo tính bí mật vị trí của nó, ví dụ tên miền www.ptit.edu.vn không thể hiện bất kỳ thông tin nào về vị trí địa lý của máy chủ đang cài đặt trang này.

Tài nguyên có thể di chuyển trong hệ thống phân tán mà không ảnh hưởng tới việc cung cấp dịch vụ thì có thể nói rằng hệ thống đã đảm bảo tính trong suốt về di trú, thậm chí hệ thống có tính trong suốt về định vị lại vị trí nếu tài nguyên chuyển sang vị trí khác trong khi người sử dụng đang truy nhập mà không hề bị gián đoạn. Ví dụ mạng di động, người sử dụng có thể di chuyển trong khi vẫn đang thực hiện các giao dịch mà không hề bị mất kết nối mạng, điều này càng chứng tỏ vai trò quan trọng của tính trong suốt về đặt lại vị trí trong các hệ thống phân tán.

Để tăng tính sẵn sàng và qua đó tăng hiệu năng hệ thống người ta có thể áp dụng kỹ thuật nhân bản tài nguyên, làm cho tài nguyên đến gần vị trí truy nhập hơn, tính trong suốt trong nhân bản giải quyết vấn đề tồn tại nhiều bản sao của tài nguyên. Để che giấu việc nhân bản thì tên của các bản sao nên được đặt giống nhau, điều đó sẽ hỗ trợ tính trong suốt về vị trí, nếu không sẽ khó khăn trong việc tham chiếu đến các bản sao tại những vị trí khác nhau.

Một trong những mục tiêu quan trọng của hệ thống phân tán là cho phép chia sẻ tài nguyên, trong nhiều trường hợp những tài nguyên đó phải được chia sẻ theo cách cộng tác, nghĩa là phải cho phép nhiều người đồng thời sử dụng tài nguyên. Ví dụ cho phép chia sẻ đường truyền, chia sẻ tập tin và dữ liệu trên các máy chủ..., giải quyết tốt vấn đề tương tranh và hiệu năng sẽ góp phần nâng cao tính sẵn sàng của hệ thống. Sử dụng khóa là cơ chế thường được sử dụng để giải quyết vấn đề tương tranh, tuy nhiên nhược điểm có hưu của nó lại là hiện tượng khóa chét, đó là tình huống các tiến trình chờ nhau để được truy nhập tài nguyên. Nếu xung đột xảy ra thì hiệu năng bị suy giảm nghiêm trọng và thậm chí có thể làm hệ thống ngừng cung cấp dịch vụ.

Một vấn đề quan trọng khác trong hệ thống phân tán liên quan đến việc xử lý lỗi, người sử dụng không cần biết lỗi gì đã xảy ra mà chỉ cần quan tâm đến yêu cầu của họ có thực hiện đúng hay không. Tính trong suốt về lỗi nghĩa là hệ thống phải có khả năng chịu lỗi, nếu khắc phục kịp thời thì coi như lỗi chưa từng xảy ra nếu nhìn dưới góc độ của người sử dụng. Hạn chế tối đa những thông báo lỗi và cách phục hồi như thế nào, che giấu lỗi là một trong những vấn đề khó khăn nhất của hệ thống phân tán và thậm chí không thể thực hiện được nếu như không lường hết các tình huống gây lỗi.

Khi có lỗi xảy ra, hệ thống phải có khả năng phát hiện lỗi và kịp thời thông báo cho người quản trị về vị trí và các thông tin liên quan, đồng thời phải lưu vết để phục vụ công tác kiểm tra và nâng cấp phiên bản phần mềm. Khắc phục lỗi thể hiện ở ba mức độ: Tự động sửa lỗi nếu bảo đảm kết quả chính xác và thời gian thực hiện, nếu không thì đưa ra các khuyến nghị cho người sử dụng, cuối cùng là thông báo cho người sử dụng biết đồng thời khôi phục trạng thái về trước thời điểm thực hiện yêu cầu. Khó khăn chính trong việc che giấu lỗi nằm ở chỗ không có khả năng phân biệt giữa việc không thể truy nhập tài nguyên và truy nhập tài nguyên chậm, ví dụ trình duyệt gửi yêu cầu đến máy chủ nhưng quá thời gian vẫn không thấy trả lời, trình duyệt sẽ thông báo lỗi trang web không tồn tại, như vậy người sử dụng không thể kết luận lỗi đã xảy ra trên máy chủ hay mất kết nối mạng.

#### 1.5.2.2 Mức độ trong suốt

Mặc dù tính trong suốt được xem như một trong những mục tiêu của hệ thống phân tán, tuy nhiên trong nhiều trường hợp sự che giấu quá mức không phải là điều tốt. Các hệ thống phân tán dựa trên nền tảng của mạng máy tính, độ trễ của mạng bao gồm trễ truyền dẫn và trễ xử lý trên các thiết bị mạng, do đó việc trao đổi thông tin giữa các máy tính chắc chắn sẽ cần một khoảng thời gian nhất định. Tính trong suốt càng cao thì đòi hỏi hệ thống phải xử lý càng nhiều, điều này có thể dẫn tới sự suy giảm hiệu năng của hệ thống, do đó cần phải cân đối giữa mức độ trong suốt và hiệu năng.

Trong các hệ thống dữ liệu nhân bản, mỗi cập nhật thực hiện trên một bản sao thì sẽ phải lan tỏa đến tất cả các bản sao khác, như vậy để đảm bảo tính nhất quán thì thời gian thực hiện sẽ lâu hơn, điều này không nên che giấu người sử dụng, có thể hiển thị thông báo giao dịch đang được thực hiện hoặc thời gian dự kiến hoàn thành để người sử dụng yên tâm chờ đợi. Một ví dụ khác, khi muốn in một tập tin, việc chọn máy in nên được thực hiện thủ công, không nên để hệ thống tìm kiếm máy in đang rỗi và thông báo cho người sử dụng để gửi yêu cầu in.

Ví dụ cuối cùng là trường hợp hệ thống nhân bản nhưng có một bản sao nằm trên thiết bị của người sử dụng, thông thường bản sao trên thiết bị của người sử dụng và bản sao trên máy chủ của hệ thống phân tán phải được đồng bộ. Tuy nhiên, vì một lý do nào đó những bản sao này chưa được đồng bộ, trường hợp này không nên che giấu người sử dụng. Cũng có ý kiến cho rằng không nên che giấu, khó có thể đạt được sự trong suốt hoàn toàn cho nên sẽ tốt hơn nếu phơi bày các vấn đề người sử dụng biết và có biện pháp chủ động xử lý, tuy nhiên điều này lại ảnh hưởng đến tính thân thiện của hệ thống đối với người sử dụng. Như vậy tính trong suốt là mục tiêu rất quan trọng khi thiết kế hệ thống phân tán nhưng cần phải xem xét vấn đề hiệu năng và tính thân thiện, để đạt được tính trong suốt hoàn toàn có thể sẽ phải chi phí rất cao trong việc xử lý bên trong hệ thống.

#### 1.5.3 Tính mở của hệ thống

Tính mở của một hệ thống được hiểu đơn giản là có thể kết nối đến các hệ thống khác, dễ dàng nâng cấp/mở rộng dịch vụ, tương thích với các nền tảng phần cứng và hệ điều hành. Để đạt mục tiêu này thì phải tuân thủ các chuẩn giao tiếp đã được công bố, nghĩa là sản phẩm của các nhà sản xuất khác nhau có thể tương tác với nhau theo tập các luật và các qui tắc đã được chuẩn hóa. Trong mạng máy tính, các qui tắc chuẩn đê ra định

dạng, nội dung và ý nghĩa của các thông điệp gửi và nhận, những qui tắc như vậy gọi là giao thức. Tương tự như vậy, giao tiếp giữa các hệ thống lớn đã được các tổ chức quốc tế chuẩn hóa, ví dụ chuẩn ISO 8583 khuyến nghị cho thanh toán liên ngân hàng, các hệ thống đào tạo trực tuyến nên tuân thủ chuẩn SCORM...

Khi thiết kế hệ thống phân tán, các dịch vụ thường được mô tả bằng ngôn ngữ định nghĩa giao diện, nó thể hiện cú pháp của các dịch vụ, nghĩa là xác định chính xác tên của các hàm cùng với các tham số và giá trị trả về, thậm chí mô tả cả các trường hợp lỗi có thể xảy ra. Khó khăn nằm ở việc mô tả chính xác các dịch vụ đó làm gì, đó là ngữ nghĩa của các giao diện, trong thực tế những đặc tính kỹ thuật này được công bố không chính thức dưới hình thức ngôn ngữ tự nhiên. Nếu mô tả chính xác, định nghĩa giao diện sẽ cho phép tiến trình bất kỳ có thể tương tác với tiến trình cung cấp giao diện này, nó cũng cho phép các bên thực hiện cài đặt các giao diện hoàn toàn khác nhau nhưng chúng vận hành giống hệt nhau.

Các đặc tả chính xác phải đầy đủ và trung lập, tính đầy đủ thể hiện tất cả những việc cần phải thực hiện đều được xác định rõ ràng, tuy nhiên nhiều đặc tả chưa hoàn toàn đầy đủ và như vậy người phát triển phải thêm những chi tiết, tính trung lập thể hiện các đặc tả không qui định cách cài đặt cụ thể cho mỗi giao diện. Cả hai đặc tính này rất quan trọng để hệ thống có khả năng tương thích và khả năng dễ dàng chuyển đổi, khả năng tương thích thể hiện việc cài đặt các thành phần của những nhà sản xuất khác nhau có thể cùng làm việc chỉ dựa trên các dịch vụ của nhau như đã mô tả trong các tiêu chuẩn chung.

Khả năng dễ dàng chuyển đổi đặc trưng cho mức độ mà người phát triển những hệ thống khác nhau có thể tùy biến cùng một dịch vụ đã cung cấp. Một mục tiêu khác của tính mở trong hệ thống phân tán là khả năng dễ dàng cấu hình hệ thống từ các thành phần khác nhau, có thể dễ dàng thêm thành phần mới hoặc thực hiện những thao tác sửa đổi nhưng không ảnh hưởng đến các thành phần khác, mục tiêu này gọi là khả năng mở rộng của hệ thống. Hệ thống có khả năng mở rộng nghĩa là có thể tương đối dễ dàng thêm các thành phần chạy trên những nền tảng khác nhau, thậm chí có thể thay thế toàn bộ hệ thống. Những đặc điểm này đặc trưng cho tính linh hoạt của hệ thống, tất nhiên việc đề ra những mục tiêu như vậy rất dễ nhưng để thực hiện các mục tiêu đó lại là chuyện khác, rất nhiều công việc phải thực hiện khi nâng cấp hoặc chuyển đổi hệ thống mà không ảnh hưởng tới yêu cầu cung cấp dịch vụ liên tục.

Hệ thống phân tán mở thường được phân tách thành tập các thành phần tương đối nhỏ, dễ thay thế và dễ thích nghi, thông thường chúng được thiết kế dựa trên cách tiếp cận hướng dịch vụ. Điều này ngũ ý mỗi thành phần không chỉ cung cấp định nghĩa cho các giao diện mức cao nhất mà còn phải cung cấp định nghĩa cho những thành phần bên trong của nó và mô tả chính xác cách tương tác với nhau. Nhiều hệ thống cũ và ngay cả một số hệ thống hiện nay đều được xây dựng dựa trên cách tiếp cận khôi, nghĩa là các thành phần tách biệt nhau về mặt logic nhưng lại được biên dịch trong một chương trình lớn, do đó khó có thể thay thế hoặc thích nghi thành phần mà không ảnh hưởng đến toàn bộ hệ thống, các hệ thống như vậy có xu thế đóng chứ không có tính mở. Hệ thống phân tán thường phải được nâng cấp là sự thay đổi dịch vụ, các thành phần chưa lường hết được các chính sách thị trường, điều cần thiết là phải tách biệt giữa chính sách với các cơ chế xử lý, nên tổng quát hóa và cụ thể hóa bằng cách cung cấp các tham số để người sử dụng có thể tùy biến.

#### **1.5.4 Qui mô hệ thống**

Các ứng dụng phân tán được kết nối qua mạng Internet đã và đang nhanh chóng phủ khắp mọi hoạt động trên toàn thế giới, như vậy qui mô hệ thống là một trong những mục tiêu quan trọng nhất trong thiết kế hệ thống phân tán. Hệ thống được coi là đáp ứng yêu cầu về qui mô nếu nó đáp ứng các tiêu chí về tăng số lượng người sử dụng, mở rộng phạm vi địa lý và khả năng quản trị.

Tăng số lượng người sử dụng nhưng vẫn dựa trên hạ tầng hiện có, nghĩa là có thể dễ dàng thêm người sử dụng và tài nguyên vào hệ thống nhưng không cần phải sửa đổi phần mềm và có thể không cần phải đầu tư thêm hạ tầng phần cứng. Một dịch vụ nếu được thị trường chấp nhận thì theo thời gian số lượng người sử dụng sẽ tăng lên nhanh chóng, do đó số lượng giao dịch nhiều hơn nhưng thời gian xử lý các yêu cầu vẫn phải duy trì ổn định ở mức chấp nhận được mà không cần phải nâng cấp phần cứng. Năng lực của phần cứng cũng có giới hạn nhất định, nếu thấy vượt quá khả năng cho phép thì có thể nâng cấp, tuy nhiên việc này không nên quyết định tùy tiện, nên ưu tiên xử lý bằng giải pháp phần mềm. Thông thường khi số người sử dụng nhiều hơn thường dễ xảy ra tương tranh hoặc tắc nghẽn ở một điểm nào đó, do đó từ thiết kế đến phát triển phần mềm và qui trình vận hành hệ thống phải được xem xét kỹ lưỡng.

Tiêu chí thứ hai là hệ thống có thể mở rộng theo phạm vi địa lý, người sử dụng và tài nguyên có thể nằm cách xa nhau hoặc số lượng người sử dụng quá lớn làm cho thời gian đáp ứng vượt quá ngưỡng cho phép, khi đó phải lắp đặt thêm các thiết bị nhưng không cần phải nâng cấp phần mềm. Việc phân vùng người sử dụng sẽ giảm thời gian trễ trên kênh truyền và phân tải xử lý trên các máy chủ khác nhau, tuy nhiên điều này đòi hỏi phải thiết kế cơ chế và các chính sách nhân bản dữ liệu để đảm bảo tính nhất quán trong hệ thống.

Tiêu chí thứ ba là khả năng quản lý, qui mô hệ thống có thể mở rộng nhưng vẫn dễ dàng quản lý dù cho công việc này có thể do nhiều đơn vị thực hiện độc lập trong từng khu vực. Phần mềm quản lý thường phải được xây dựng sao cho có thể cấu hình các thiết bị, điều chỉnh hiệu năng, giám sát lỗi và an toàn bảo mật thông tin, quản lý các dịch vụ và tài khoản. Các tính năng quản lý phải thân thiện, tạo điều kiện thuận lợi cho người vận hành có thể dễ dàng xử lý khi có sự cố xảy ra, nếu gặp lỗi thì có thể khắc phục trong thời gian ngắn nhất.

##### **1.5.4.1 Vấn đề hiệu năng khi mở rộng qui mô hệ thống**

Qui mô hệ thống mở rộng có thể dẫn đến thời gian phản hồi kết quả xử lý các yêu cầu lớn hơn, đó là biểu hiện của hiện tượng suy giảm hiệu năng. Nâng cấp hạ tầng phần cứng là cách làm đúng nhưng không phải là giải pháp được ưu tiên lựa chọn, trước tiên nên xem xét cách vận hành của hệ thống phần mềm. Thiết kế hệ thống phân tán cần tránh cung cấp dịch vụ tập trung, lưu trữ dữ liệu tập trung và sử dụng các giải thuật tập trung.

Các dịch vụ được cài đặt tập trung trên một máy chủ tiềm ẩn nguy cơ nghiêm cấm, dù cho năng lực xử lý chủ rất lớn thì bằng thông kết nối đến máy chủ cũng là trở ngại khi mở rộng qui mô. Ví dụ hệ thống phân giải tên miền trên mạng Internet, nếu dịch vụ này tập trung trên một máy chủ, tại một thời điểm có hàng tỉ yêu cầu gửi đến thì chắc

chắn sẽ gây ra hiện tượng tắc nghẽn. Đôi khi không thể tránh dịch vụ tập trung, ví dụ quản lý thông tin mức độ bí mật cao như tài khoản ngân hàng, thuê bao điện thoại..., những trường hợp này nên đặt máy chủ tại nơi được bảo vệ nghiêm ngặt và tách khỏi các bộ phận khác của hệ thống. Có thể giải quyết vấn đề trên bằng cách phân biệt yêu cầu của người sử dụng, tách biệt dịch vụ truy vấn và dịch vụ cập nhật.

Lưu trữ dữ liệu tập trung thậm chí còn tồi hơn dịch vụ tập trung, công nghệ lưu trữ có thể đáp ứng không gian cho hàng tỉ bản ghi nhưng chắc chắn kênh truyền đến máy chủ cũng sẽ bị bão hòa khi có nhiều người đồng thời truy nhập. Hơn nữa, dữ liệu thường được quản lý bởi các hệ quản trị dữ liệu, thời gian truy vấn sẽ tăng lên đáng kể nếu số lượng bản ghi trong mỗi bảng quá lớn, ngay cả khi đã áp dụng kỹ thuật chỉ mục. Hệ quản trị cơ sở dữ liệu lưu trữ các bảng trên các tập tin, kích thước mỗi tập tin thường không quá 32 GB, vượt qua ngưỡng này thì phải tạo tập tin mới, điều này có thể dẫn đến hiện tượng phải đọc quá nhiều tập tin khi tìm kiếm, đó là một trong những nguyên nhân làm tăng thời gian tìm kiếm. Lưu trữ lượng dữ liệu quá lớn trong một hệ quản trị dữ liệu cũng gây trở ngại cho công tác sao lưu dự phòng, thời gian sao lưu 1 TB dữ liệu có thể lên tới vài giờ trong khi đó con số này chỉ nên dưới một phút. Kiểu dữ liệu đa phương tiện thường có kích thước lớn hơn rất nhiều so với các kiểu dữ liệu cơ bản, việc lưu trữ lẩn lộn hai loại này trong một bảng là điều tối kỵ.

Cuối cùng là vấn đề giải thuật tập trung, ví dụ xây dựng bảng định tuyến, dưới góc độ lý thuyết có thể tìm ra đường đi tối ưu bằng cách thu thập thông tin các kênh truyền sau đó tính toán để tìm ra tuyến đường tốt nhất, đây là ý tưởng rất tốt nhưng thực thi nó thì lại nảy sinh nhiều vấn đề. Thông tin về tải có thể lan truyền trên toàn bộ hệ thống để cải thiện vấn đề định tuyến, khó khăn ở chỗ việc thu thập và vận chuyển các thông điệp này lại làm tăng tải cho mạng. Thực tế cho thấy nên tránh các giải thuật thu thập thông tin từ nhiều nút để tính toán sau đó gửi kết quả cho các nút khác, nên thay thế bằng các giải thuật phi tập trung. Khác với giải thuật tập trung, các giải thuật không tập trung thường có những đặc điểm sau:

- Không một máy tính nào có đầy đủ thông tin về trạng thái hệ thống.
- Các máy tính ra quyết định chỉ dựa trên thông tin cục bộ.
- Lỗi xảy ra trên một máy sẽ không ảnh hưởng đến các máy khác và do đó không làm hỏng giải thuật.
- Không tồn tại đồng hồ chung cho toàn bộ hệ thống.

Đặc điểm thứ tư sẽ được đề cập chi tiết trong một chương riêng, các giải thuật cần phải tôn trọng sự thật không thể đồng bộ chính xác thời gian giữa các máy tính trong hệ thống phân tán. Trong phạm vi mạng cục bộ, việc đồng bộ thời gian có thể đạt chính xác đến vài  $\mu$ s, nhưng điều này sẽ không thể đạt được trên mạng diện rộng, sẽ rất mạo hiểm nếu sử dụng đồng hồ quốc gia hay đồng hồ quốc tế.

Qui mô về địa lý chứa đựng vấn đề riêng của nó, một trong những lý do chính rất khó mở rộng qui mô về địa lý cho những hệ thống phân tán được thiết kế chạy trong mạng cục bộ là do chúng dựa trên phương pháp truyền thông đồng bộ, nghĩa là tiến trình trên máy khách sẽ bị phong tỏa trong thời gian chờ kết quả trả về từ tiến trình trên máy

chủ. Truyền thông đồng bộ nói chung sẽ vận hành tốt trong mạng cục bộ, nơi mà độ trễ truyền thông chỉ vài µs. Vấn đề sẽ hoàn toàn khác khi áp dụng cho mạng diện rộng, độ trễ truyền thông có thể lên tới hàng trăm ms, nghĩa là chậm hơn hàng ngàn lần so với mạng cục bộ, do đó xây dựng các ứng dụng tương tác trong mạng diện rộng đòi hỏi phải nỗ lực rất lớn và cần thiết sự kiên nhẫn để tìm ra hướng giải quyết các vấn đề phát sinh.

Một vấn đề khác khi mở rộng qui mô địa lý là tính tin cậy trong các mạng diện rộng, nó hầu như chỉ hỗ trợ các kết nối điểm-điểm chứ không áp dụng được phương pháp quảng bá hoặc truyền tin theo nhóm. Ví dụ vấn đề xác định dịch vụ trong mạng cục bộ chỉ cần quản bá thông điệp đến các máy để hỏi xem có chạy dịch vụ hay không, chỉ những máy cung cấp dịch vụ mới trả lời, cách làm này chắc chắn không áp dụng được cho mạng diện rộng, như vậy sẽ phải thiết kế dịch vụ định vị đáp ứng yêu cầu cho hàng tỉ người sử dụng trên toàn thế giới.

Qui mô về địa lý cũng liên quan mạnh mẽ tới các vấn đề của giải pháp tập trung, chúng đều gây trở ngại cho việc mở rộng, nếu hệ thống gồm nhiều thành phần tập trung thì qui mô về địa lý sẽ bị giới hạn về hiệu năng và những vấn đề tin cậy phát sinh trong truyền thông trên mạng diện rộng. Hơn nữa, các thành phần tập trung sẽ dẫn đến sự lãng phí tài nguyên mạng, tương tác của người sử dụng với máy chủ sẽ phải đi qua nhiều thiết bị định tuyến, rõ ràng việc xử lý tập trung sẽ không thể đáp ứng yêu cầu mở rộng qui mô về địa lý.

Câu hỏi cuối cùng khá phức tạp, đó là làm sao có thể mở rộng hệ thống phân tán cho nhiều miền quản lý độc lập, vấn đề chính cần giải quyết là mâu thuẫn về chính sách đối với việc sử dụng tài nguyên, đó có thể là chính sách về thanh khoản và quản lý hay vấn đề bảo mật. Ví dụ về chính sách bảo mật, người sử dụng có thể tin cậy vào các thành phần của hệ thống phân tán thuộc cùng một miền quản lý, quản trị hệ thống có thể kiểm tra và chứng nhận cho các ứng dụng và thậm chí có thể thực hiện các biện pháp đặc biệt để bảo đảm các thành phần không bị giả mạo, tuy nhiên niềm tin này sẽ không thể mở rộng ra ngoài vùng quản lý. Nếu hệ thống phân tán mở rộng sang vùng khác thì phải thực hiện hai biện pháp bảo mật, thứ nhất phải tự bảo vệ chống lại tấn công phá hoại từ vùng mới, người sử dụng của vùng mới có thể chỉ được cấp quyền đọc tài nguyên, thậm chí không cấp quyền truy nhập đến một số tài nguyên cho người ngoài. Thứ hai, vùng mới cũng phải tự bảo vệ trước các cuộc tấn công phá hoại nhưng vẫn phải đảm bảo cấp quyền truy nhập cho người sử dụng thuộc vùng hiện hành.

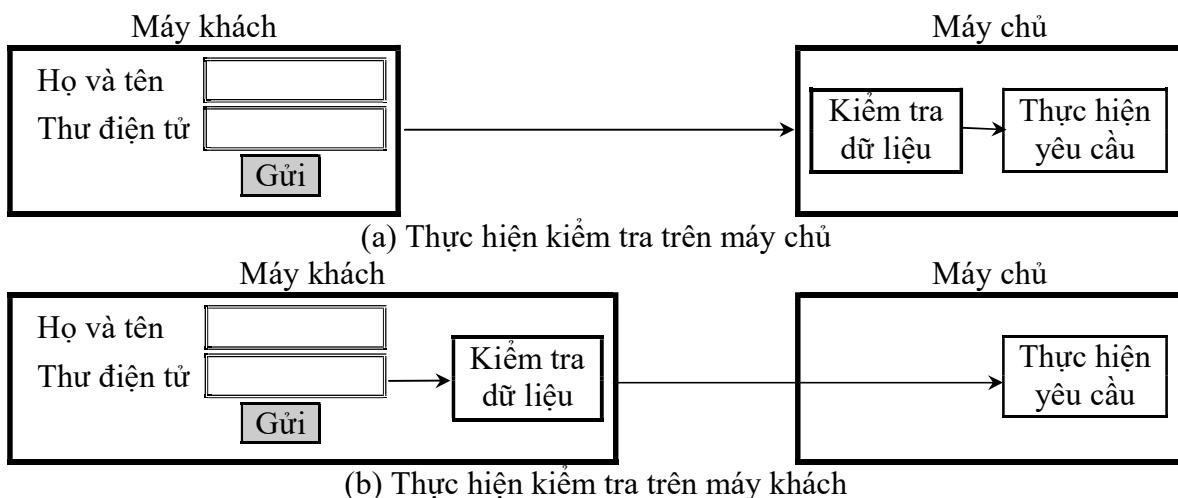
#### 1.5.4.2 Các kỹ thuật xử lý trong hệ thống qui mô lớn

Vấn đề qui mô hệ thống phức tạp, nhưng quan trọng hơn cần phải trả lời câu hỏi bằng cách nào có thể giải quyết những vấn đề này, hầu hết các trường hợp đều dẫn đến vấn đề về hiệu năng trong điều kiện hạn hẹp về năng lực của mạng và các máy chủ. Nếu hiệu năng suy giảm thì trước hết phải kiểm tra vấn đề tương tranh, tìm ra những điểm tắc nghẽn trong hệ thống và nên áp dụng phương pháp xử lý song song, nếu đã áp dụng cả ba biện pháp trên mà hiệu năng vẫn thấp thì mới nâng cấp hoặc bổ sung hạ tầng phần cứng. Để có thể sẵn sàng đáp ứng các yêu cầu về hiệu năng cho các hệ thống qui mô lớn thì cần

sử dụng các kỹ thuật che giấu trễ truyền thông, phân tán xử lý và nhân bản dữ liệu khi xây dựng phần mềm.

Che giấu trễ truyền thông đóng vai trò quan trọng trong việc mở rộng qui mô về địa lý, thực chất đơn giản chỉ là tránh chờ đợi kết quả trả về từ các dịch vụ từ xa càng nhiều càng tốt, nghĩa là bên gửi yêu cầu sẽ sử dụng thời gian chờ đợi cho những công việc hữu ích. Như vậy, chỉ có thể áp dụng cơ chế truyền thông không đồng bộ mới giải quyết được vấn đề này, ngay khi nhận được kết quả trả về sẽ tạm ngừng mọi công việc đang thực hiện để tiếp tục xử lý yêu cầu đã gửi trước đó. Truyền thông không đồng bộ thường được sử dụng trong các hệ thống xử lý lô và trong các ứng dụng xử lý song song, trong đó có thể lập lịch thực thi cho các nhiệm vụ độc lập trong khi chờ đợi hoàn thành truyền thông. Hoặc giả có thể khởi tạo một luồng mới để thực hiện yêu cầu, mặc dù nó bị phong tỏa để chờ kết quả trả về nhưng luồng khác của tiến trình vẫn có thể tiếp tục thực hiện.

Một số trường hợp không thể áp dụng truyền thông không đồng bộ, ví dụ trong các ứng dụng tương tác, sau khi gửi yêu cầu thì người sử dụng sẽ không làm gì hơn ngoài việc chờ đợi kết quả trả về. Trong những trường hợp này, giải pháp tốt nhất là giảm tối đa thời gian xử lý trên máy chủ bằng cách chuyển một số chức năng xử lý trên máy chủ về máy khách và giảm thiểu lượng dữ liệu di chuyển trên mạng.



Hình 1.5 Ưu tiên tiền xử lý trên máy khách

Hình 1.5 minh họa trường hợp điển hình về ưu tiên tiền xử lý trên máy khách khi đăng ký thông tin họ và tên và thư điện tử, người sử dụng nhập thông tin đăng nhập, chúng được chuyển thành thông điệp để gửi đến máy chủ và chờ đợi kết quả trả về, máy chủ phải kiểm tra tính hợp lệ của dữ liệu trước khi tiếp tục xử lý. Giải pháp tốt hơn sẽ là máy khách sẽ kiểm tra tính hợp lệ của dữ liệu trước khi gửi lên máy chủ, như vậy sẽ tiết kiệm thời gian xử lý trên máy chủ, thời gian đáp ứng nhanh hơn chỉ là biểu hiện bên ngoài, quan trọng hơn lại là tăng năng lực tiếp nhận yêu cầu của máy chủ. Trường hợp tương tự là hậu xử lý trên máy khách, sau khi yêu cầu được chuyển đến máy chủ, máy chủ sẽ lấy tập bản ghi từ cơ sở dữ liệu, thay vì xử lý trên máy chủ có thể chuyển nhiệm

vụ này cho máy khách. Tuy nhiên trường hợp này chỉ áp dụng cho tập nhỏ các bản ghi, nếu dung lượng dữ liệu lớn thì không nên áp dụng kỹ thuật này, giảm thời gian xử lý trên máy chủ là quan trọng nhưng giảm trao đổi thông tin lại là ưu tiên hàng đầu.

Kỹ thuật thứ hai là phân tán dịch vụ, nghĩa là một dịch vụ sẽ được chia thành nhiều phần nhỏ hơn và trải đều trên toàn bộ hệ thống. Ví dụ điển hình là hệ thống phân giải tên miền trên mạng Internet, không gian tên được tổ chức phân cấp thành các miền sau đó thành các khu vực, mỗi máy chủ chỉ cần lưu giữ tên miền trong khu vực phụ trách. Trang tin điện tử là hệ thống thông tin dựa trên văn bản, mỗi văn bản được xác định bằng một tên duy nhất dưới dạng đường dẫn URL, sử dụng URL giúp cho hệ thống có thể mở rộng qui mô. Để phân giải tên miền, máy khách phải gửi đường dẫn đến máy chủ tên miền, do cấu trúc phân cấp nên các yêu cầu sẽ không tập trung về một máy chủ nên vẫn luôn đảm bảo hiệu năng cho hệ thống.

Việc mở rộng qui mô thường làm suy giảm hiệu năng của hệ thống, giải pháp tốt nhất là thực hiện nhân bản, nhân bản không chỉ làm tăng tính sẵn sàng mà còn cân bằng tải và như vậy hiệu năng có thể sẽ tốt hơn, đối với các hệ thống qui mô địa lý rộng lớn việc các bản sao đặt ngay gần người sử dụng sẽ giảm độ trễ truyền thông. Tuy nhiên thực hiện nhân bản lại dẫn đến vấn đề về nhất quán, có thể chấp nhận tính nhất quán tới mức nào tùy thuộc vào việc sử dụng tài nguyên, những giao dịch thương mại điện tử đòi hỏi tính nhất quán rất cao. Nhất quán liên tục đòi hỏi các cập nhật phải được lan truyền ngay lập tức đến tất cả các bản sao, nghĩa là đòi hỏi cơ chế đồng bộ trên toàn hệ thống, những kỹ thuật như vậy rất khó triển khai trong các hệ thống lớn.

Dưới góc độ kỹ thuật, việc mở rộng qui mô người sử dụng được coi là đơn giản nhất, trong một số trường hợp thậm chí chỉ cần tăng năng lực xử lý của máy chủ. Mở rộng qui mô về địa lý sẽ phức tạp hơn nhưng vẫn có thể kết hợp ba kỹ thuật trên để giải quyết các vấn đề về tính nhất quán. Mở rộng qui mô về miền quản trị vẫn là vấn đề phức tạp nhất vì phải giải quyết cả những vấn đề phi kỹ thuật, ví dụ chính sách của các đơn vị và sự cộng tác giữa các nhân viên quản trị. Tuy nhiên có những tiến bộ nhất định đã đạt được trong vấn đề này, chỉ cần đơn giản bỏ qua các vùng quản trị, ví dụ có thể sử dụng kiến trúc ngang hàng để người sử dụng điều khiển cuối tự kiểm soát, tuy nhiên đó chỉ là một phần trong việc giải quyết mở rộng qui mô quản trị.

### 1.5.5 Hệ thống phân tán và mạng máy tính

Phát triển các hệ thống phân tán là công việc rất lớn, cùng một lúc phải cân nhắc rất nhiều vấn đề và đó đều là những vấn đề phức tạp. Tuy nhiên vấn đề sẽ được giải quyết nếu làm theo các nguyên tắc thiết kế và tuân thủ nghiêm ngặt các mục tiêu đã đề ra. Việc phát triển hệ thống trước hết phải tuân theo những qui tắc của công nghệ phần mềm, nhưng phải luôn lưu ý điểm khác biệt cơ bản so với hệ thống chạy độc lập trên một máy tính, các thành phần của hệ thống phân tán giao tiếp với nhau qua mạng. Khi phát triển hệ thống phân tán cần phải tránh những quan niệm chưa đúng sau:

- Mạng đáng tin cậy.
- Mạng đã được bảo mật.
- Mạng là hệ thống đồng nhất.

- Hình trạng mạng không bao giờ thay đổi.
- Độ trễ lưu chuyển dữ liệu trên mạng bằng không.
- Bandwidth của mạng là vô hạn.
- Chi phí vận chuyển bằng không.
- Chỉ có một người quản trị.

Những vấn đề trên không cần thiết phải để ý khi phát triển các ứng dụng chạy độc lập trên một máy tính, nhưng đó lại là những vấn đề cần phải giải quyết khi phát triển hệ thống phân tán. Mạng không đáng tin cậy đòi hỏi phải giải quyết vấn đề trong suốt về lỗi truyền thông, vấn đề bảo mật phải thực hiện bằng chính sách mã hóa và xác thực..., như vậy có thể thấy một lượng công việc rất lớn và phức tạp khi phát triển các hệ thống phân tán. Độ trễ truyền tin trên mạng thường từ vài chục đến hàng trăm mili giây, khắc phục nhược điểm này bằng cách mở nhiều kênh truyền thông logic và áp dụng kỹ thuật xử lý song song cũng như phương pháp truyền thông không đồng bộ. Bandwidth của mạng ngày càng được cải thiện nhưng nó không phải là tài nguyên vô hạn, cần phải giảm thiểu trao đổi thông tin giữa các thành phần trong hệ thống.

## 1.6 Kiến trúc và mô hình hệ thống phân tán

Hệ thống phân tán là tổ hợp các phần mềm chạy trên nhiều máy tính, để làm chủ các thành phần phức tạp này cần phải tổ chức quản lý và cách trao đổi thông tin giữa chúng. Kiến trúc thể hiện cấu trúc hệ thống, nó thể hiện mối quan hệ giữa các thành phần và các chính sách quản lý để có thể vận hành gắn kết như một thể thống nhất. Mô hình là hình mẫu để thực hiện triển khai trong thực tiễn, nó giải quyết vấn đề liên kết giữa các thành phần trong hệ thống.

Mỗi tiền trình thường phải tương tác với tiền trình trên các máy tính khác, nếu mỗi chức năng của phần mềm đều phải cài đặt tính năng chung phục vụ tương tác qua mạng thì rất lãng phí và khó có thể đạt được tính ổn định, do đó kiến trúc hệ thống phân tán thường tách biệt phần ứng dụng với nền tảng bằng cách cung cấp thêm tầng trung gian. Việc cài đặt thêm tầng trung gian không những đảm bảo tính trong suốt mà còn nâng cao tính mở và tính linh hoạt cho hệ thống.

### 1.6.1 Kiến trúc hệ thống phân tán

Kiến trúc thể hiện cách tổ chức các thành phần trong hệ thống phân tán, chúng được sắp xếp dưới dạng phân cấp hoặc ngang hàng. Mỗi kiến trúc đều có những ưu và nhược điểm riêng, do đó có thể kết hợp cả hai để hình thành kiến trúc lai ghép.

#### 1.6.1.1 Kiến trúc phân cấp

Kiến trúc phân cấp còn gọi là kiến trúc dọc, các thành phần trong hệ thống được tổ chức dưới dạng hình cây, nó bao gồm nhiều cấp phản ánh mối quan hệ giữa các thành phần trong hệ thống phân tán. Nút cao hơn gọi là nút cha và nút thấp hơn gọi là nút con, một nút không có nút con gọi là nút lá. Kiến trúc phân cấp thể hiện tầm quan trọng của các thành phần, những thành phần cao hơn trong cấu trúc thường có vai trò trung tâm hoặc quan trọng hơn.

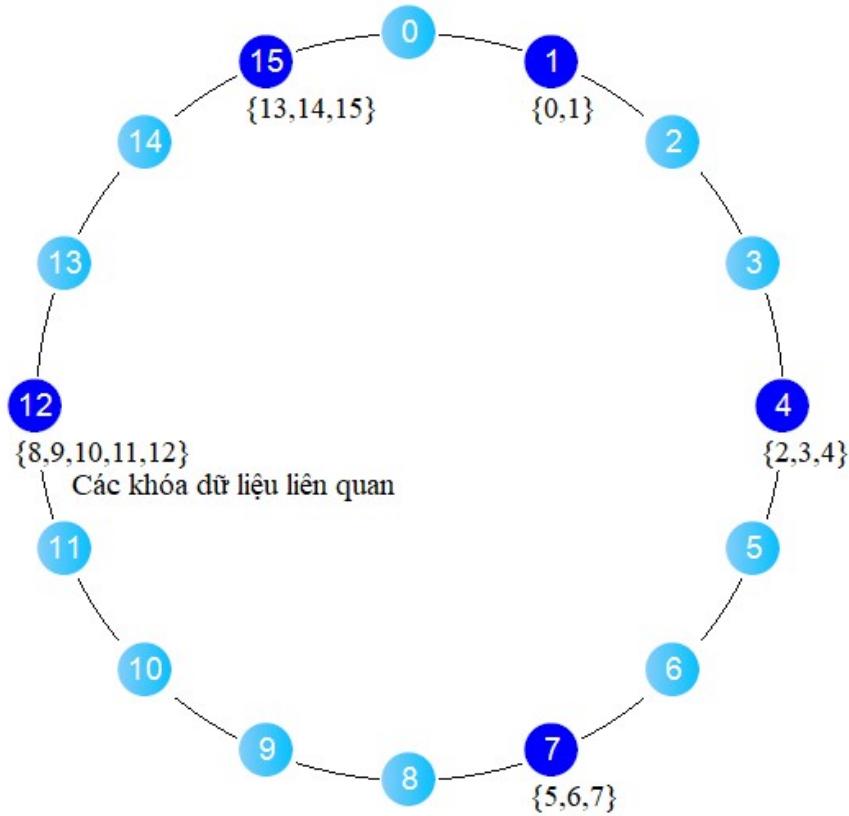
Kiến trúc phân cấp phân tách hệ thống thành các thành phần con ở các cấp độ khác nhau, thành phần con cấp thấp cung cấp dịch vụ cho các thành phần cha của nó. Kiến trúc phân cấp tạo điều kiện dễ dàng mở rộng hệ thống cũng như kiểm soát hoạt động của các thành phần, tuy nhiên nếu thành phần cha bị lỗi thì toàn bộ nhánh đó sẽ không có khả năng cung cấp dịch vụ. Kiến trúc phân cấp có thể được cấu hình tĩnh hoặc động, nếu cấu hình động thì cần thực hiện giải thuật bầu chọn mỗi khi hình trạng thay đổi.

Kiến trúc phân cấp cài đặt phần mềm trên các máy tính khác nhau, nó giúp cho các chức năng được tách xử lý trên nhiều máy và mỗi máy sẽ chuyên sâu vào một nhóm chức năng cụ thể. Một ứng dụng khác là cách lan tỏa thông tin dạng phân cấp, nút cha chuyển thông điệp cho nút con..., quá trình được thực hiện song song trên nhiều nút do đó giảm thiểu thời gian lan truyền.

#### *1.6.1.2 Kiến trúc ngang hàng*

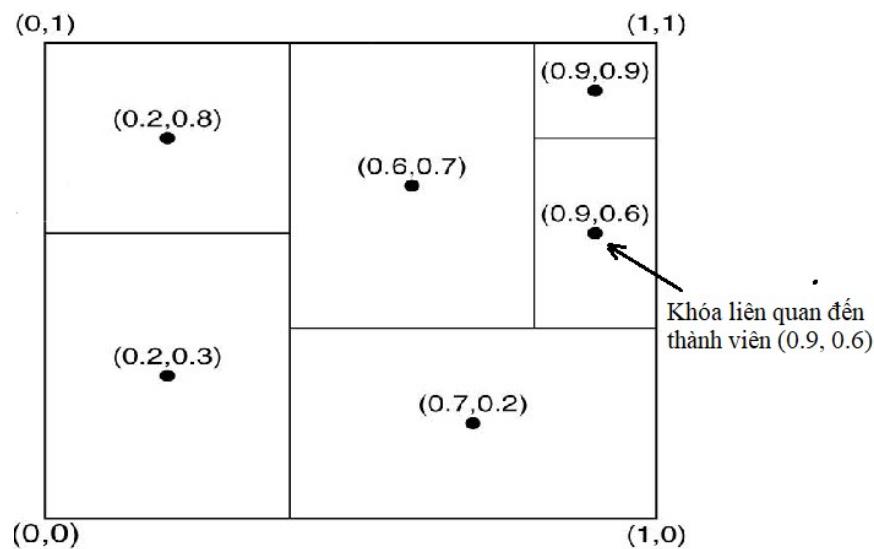
Kiến trúc ngang hàng còn gọi là kiến trúc không phân cấp, tất cả các thành viên đều có vai trò ngang nhau, các chức năng xử lý tương đương nhau nhưng mỗi thành viên hoạt động trên một tập dữ liệu hoàn chỉnh. Các chức năng cần thiết đều hiện diện trên mỗi thành viên, do đó tương tác giữa chúng thuộc loại đối xứng, chúng có thể đóng cả hai vai trò khách và chủ. Với cách thiết kế đối xứng, kiến trúc ngang hàng này sinh vấn đề tổ chức quản lý các thành viên sao cho chúng hoạt động gắn kết với nhau như một thể thống nhất, nghĩa là nếu nhìn từ bên ngoài thì nó chỉ là một thành viên, chúng có thể được tổ chức thành một nhóm có cấu trúc hoặc phi cấu trúc.

Kiến trúc ngang hàng có cấu trúc phân bố các thành viên dựa trên bảng băm phân tán, mỗi thành viên đều xây dựng bảng định tuyến trong đó các mục dữ liệu trong bảng được tính toán bằng hàm băm. Điểm mấu chốt của hệ thống dựa trên bảng băm phân tán là cài đặt lược đồ hiệu quả để ánh xạ duy nhất khóa của mục dữ liệu với định danh của thành viên dựa trên chỉ số khoảng cách. Yêu cầu dịch vụ có thể được chuyển đến bất kỳ thành viên nào, sử dụng bảng định tuyến để tìm kiếm thành viên có trách nhiệm xử lý.



Hình 1.6 Mạng ngang hàng có cấu trúc

Hình 1.6 minh họa nhóm kiến trúc ngang hàng có cấu trúc, 16 thành viên được tổ chức thành vòng, mỗi thành viên chứa bảng định tuyến của các khóa liên quan đến những dữ liệu. Các thành viên đều có vai trò ngang nhau, yêu cầu tìm kiếm được chuyển đến bất kỳ thành viên nào trong nhóm, sử dụng giải thuật băm để tính toán khóa cho mỗi mục dữ liệu, từ đó chiểu theo bảng định tuyến để quyết định chuyển tiếp yêu cầu đến thành viên tương ứng.



Hình 1.7 Mạng địa chỉ nội dung

Giải pháp khác là mạng địa chỉ nội dung CAN, nó dựa trên nguyên tắc phân mảnh các thành viên tham gia hệ thống trong không gian tọa độ đa chiều. Hình 1.7 minh họa một hệ thống lưu trữ dữ liệu dựa trên không gian hai chiều, mỗi khu vực đại diện cho một thành viên, mỗi mục dữ liệu được gán điểm duy nhất trong không gian này, mỗi thành viên có trách nhiệm cho một số mục dữ liệu ngoại trừ những mục thuộc vùng giáp gianh.

Kiến trúc ngang hàng phi cấu trúc thường dựa trên các giải thuật ngẫu nhiên để xây dựng hệ thống, mỗi thành viên duy trì danh sách hàng xóm theo cách ngẫu nhiên. Ví dụ hệ thống lưu trữ dữ liệu, các mục dữ liệu được đặt ngẫu nhiên trên mỗi thành viên, như vậy để xác định một mục dữ liệu cụ thể thì cần phải gửi nhiều thông điệp truy vấn tìm kiếm. Một trong những mục tiêu của nhiều hệ thống ngang hàng phi cấu trúc là xây dựng mạng phủ tương tự như đồ thị ngẫu nhiên, mỗi thành viên duy trì danh sách các hàng xóm đang hoạt động.

#### 1.6.1.3 Kiến trúc lai ghép

Nhiều hệ thống phân tán kết hợp cả hai loại kiến trúc để hình thành hệ thống với kiến trúc lai ghép, ví dụ hệ thống các máy chủ biên hoặc hệ thống phân tán cộng tác thuộc loại này. Sự hiện diện của hệ thống các máy chủ biên là cần thiết cho việc kết nối liên mạng, máy khách phải kết nối với máy chủ của nhà cung cấp dịch vụ Internet, điều đó thể hiện kiến trúc phân cấp bên trong mỗi nhà cung cấp dịch vụ. Không một nhà cung cấp dịch vụ Internet nào lưu trữ tất cả các nội dung, đa số các yêu cầu sẽ được định hướng đến máy chủ biên để chuyển đến các nhà cung cấp khác. Như vậy các máy chủ của nhà cung cấp dịch vụ Internet đóng vai trò là những máy chủ biên, chúng được tổ chức ngang hàng vì vai trò của các đơn vị này ngang nhau.

Các ví dụ về hệ thống phân tán cộng tác có thể kể đến như hệ thống chia sẻ tập tin ngang hàng BitTorrent và hệ thống phân phát nội dung cộng tác Globule. BitTorrent cho phép người dùng có thể phân phối dữ liệu theo kiến trúc ngang hàng với phương châm có đi có lại, nó có thể điều phối bằng tốc độ truyền dữ liệu. Với những hệ thống xây dựng tương tự như Globule, yêu cầu của người dùng được chuyển đến máy chủ, như vậy đó là kiến trúc phân cấp, nhưng nội dung thông tin sẽ được máy chủ tìm kiếm trên các hệ thống khác, như vậy đó là kiến trúc ngang hàng.

### 1.6.2 Mô hình hệ thống phân tán

Mô hình hệ thống phân tán thể hiện cách tương tác giữa các thành phần trong hệ thống, nó mô tả phương thức các thành phần liên kết với nhau, liên kết là cơ chế trung gian thực hiện nhiệm vụ trao đổi thông tin. Thành phần là một khối đơn vị với các giao diện rõ ràng, những thay đổi bên của khối không ảnh hưởng đến giao diện của nó. Các thành phần và liên kết giữa chúng hình thành bốn mô hình sau:

- Mô hình phân tầng
- Mô hình dựa trên đối tượng
- Mô hình dựa trên kênh sự kiện
- Mô hình dữ liệu tập trung

Mô hình đóng vai trò quan trọng khi xây dựng hệ thống phân tán vì chúng hướng tới tính trong suốt trong cách xử lý thông tin, cần phải cân bằng giữa những yêu cầu về

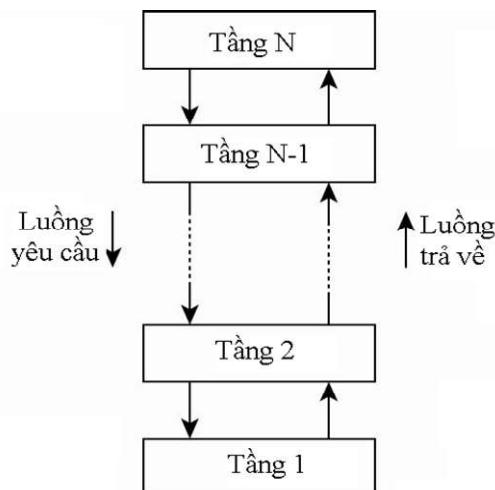
hiệu năng, khả năng chịu lỗi cũng như thời gian phát triển phần mềm. Không có mô hình nào đáp ứng yêu cầu cho tất cả các loại ứng dụng phân tán, do đó cần phải phân tích và lựa chọn mô hình thích hợp với yêu cầu của từng hệ thống.

#### 1.6.2.1 Mô hình phân tầng

Mô hình phân tầng đã được áp dụng để xây dựng mạng máy tính, các chức năng của mỗi tầng đều được qui định rõ ràng và không trùng lặp, các tầng liền kề ràng buộc chặt chẽ với nhau, tầng dưới cung cấp dịch vụ cho tầng trên. Thông điệp của bên gửi sẽ được chuyển từ tầng trên xuống tầng dưới, thông tin điều khiển sẽ được thêm vào thông điệp khi đi qua mỗi tầng. Khi thông điệp đến đích, bên nhận sẽ chuyển từ tầng dưới lên tầng trên, các thông tin điều khiển sẽ được loại bỏ khi đi qua mỗi tầng, cuối cùng sẽ có được thông điệp gốc ban đầu.

Học theo cách hoạt động của mạng máy tính, mỗi tầng của hệ thống phân tán cũng qui định thực hiện một số chức năng duy nhất, số lượng tầng càng nhiều chứng tỏ tính chuyên môn hóa càng cao, sự đa dạng và phức tạp của hệ thống sẽ được phân tách thành tập hợp những chức năng đơn giản hơn. Mô hình này đảm bảo tính chính xác rất cao, chỉ cần xảy ra lỗi tại một tầng thì coi như yêu cầu của bên gửi sẽ không được thực hiện. Ưu điểm nổi bật của mô hình này là khả năng nâng cấp, bảo trì cũng như mở rộng tính năng của mỗi tầng sẽ không ảnh hưởng đến các tầng khác.

Khi xem xét nhiều ứng dụng, hầu hết mọi người đều nhất trí phân tách thành ba tầng, đó là tầng giao tiếp với người sử dụng, tầng xử lý nghiệp vụ và tầng dữ liệu. Tầng giao diện tập trung giải quyết các chức năng giao tiếp với người sử dụng, thông tin hiển thị phải đầy đủ và được sắp xếp sao cho thuận tiện nhất, thậm chí cho phép cá nhân hóa. Các chức năng của tầng giao diện thường được cài đặt trên máy khách, cho phép người sử dụng tương tác với hệ thống một cách trong suốt. Có những điểm khác biệt đáng kể trong giao diện này, đơn giản nhất là màn hình hiển thị tương tác dạng ký tự thường xuất hiện trong các ứng dụng quản lý thiết bị cho đến các giao diện đồ họa thân thiện với người sử dụng.



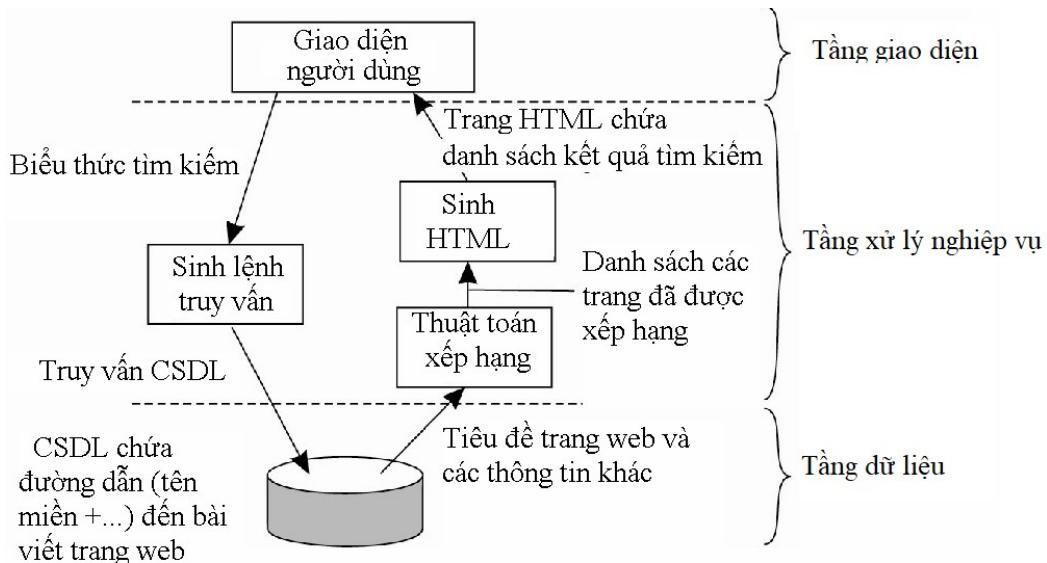
Hình 1.8 Mô hình phân tầng

Trước đây, đầu cuối chỉ thực hiện chức năng hiển thị mà không có chức năng xử lý, hiện nay giao diện của người sử dụng đã được cải thiện đáng kể, có thể sử dụng giao diện đồ họa để tương tác. Trong giai đoạn đầu phát triển hệ thống phân tán nên cung cấp các giao diện thể hiện bốn thao tác cơ bản truy vấn dữ liệu, những giao diện này có thể chưa thực sự thân thiện nhưng nó đảm bảo tính chính xác cho các nghiệp vụ xử lý, thời gian đưa các sản phẩm vào khai thác cũng sẽ được rút ngắn.

Tầng xử lý nghiệp vụ phải thực hiện rất nhiều chức năng, từ những chức năng tính toán.... cho đến các chức năng về an toàn và bảo mật thông tin. Nên tổng quát hóa các dịch vụ cung cấp cho tầng giao diện để có thể phục vụ nhiều thành phần khác nhau, cách làm này không những tiết kiệm chi phí mà còn rút ngắn thời gian xây dựng hệ thống. Rất nhiều giải thuật khác nhau được cài đặt ở tầng xử lý nghiệp vụ, cần phải sử dụng nhiều kỹ thuật khác nhau để giảm thiểu thời gian xử lý.

Dữ liệu có thể được lưu trữ trong hệ thống tập tin hoặc trên các hệ quản trị cơ sở dữ liệu, lưu trữ tại một nơi hay được nhân bản trên nhiều vị trí khác nhau, dù theo cách nào thì vẫn phải đảm bảo tính toàn vẹn của hệ thống. Tầng dữ liệu cung cấp dịch vụ cho tầng xử lý nghiệp vụ, nó bao gồm rất nhiều chương trình giải quyết các vấn đề về tính toàn vẹn, tương tranh và an toàn bảo mật thông tin, nếu dữ liệu được nhân bản thì còn phải đảm bảo tính nhất quán. Dạng đơn giản nhất là hệ thống tập tin do hệ điều hành quản lý, nhưng phổ biến dữ liệu được quản lý bởi các hệ quản trị cơ sở dữ liệu và thường được cài đặt trên các máy chủ riêng biệt.

Hầu hết các hệ thống hiện nay đều sử dụng cơ sở dữ liệu quan hệ, sự độc lập dữ liệu rất quan trọng, dữ liệu được tổ chức độc lập với ứng dụng và như vậy việc thay đổi tổ chức dữ liệu sẽ không ảnh hưởng đến các ứng dụng và ngược lại việc thay đổi các ứng dụng cũng không ảnh hưởng đến việc tổ chức dữ liệu. Sử dụng các cơ sở dữ liệu quan hệ giúp tách biệt tầng xử lý nghiệp vụ với tầng dữ liệu, do đó có thể coi hai chức năng này độc lập với nhau. Cơ sở dữ liệu quan hệ không phải là lựa chọn lý tưởng, nhiều ứng dụng hoạt động trên những loại dữ liệu phức tạp và được mô hình hóa thành các đối tượng thay cho các quan hệ. Trong những trường hợp này, các thao tác dữ liệu thường thể hiện bằng khái niệm thao tác với đối tượng, những cơ sở dữ liệu như vậy gọi là cơ sở dữ liệu hướng đối tượng hay cơ sở dữ liệu quan hệ đối tượng.



Hình 1.9 Tổ chức phân tầng máy tìm kiếm trên Internet

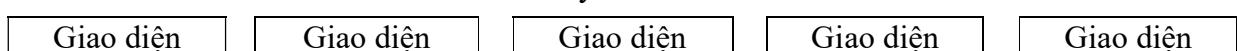
Hình 1.9 thể hiện cách hoạt động của ứng dụng tìm kiếm trên mạng Internet, bỏ qua các ảnh và những thứ tô điểm cho trang tìm kiếm thì giao diện sẽ chỉ còn là phần nhập nội dung và sau đó là phần hiển thị kết quả tìm kiếm. Khác với tầng giao diện và tầng truy nhập dữ liệu, tầng xử lý nghiệp vụ thường không có những đặc điểm chung, mỗi dịch vụ có những yêu cầu xử lý riêng, nếu có đặc điểm chung thì nó nên được tổng quát hóa. Tầng xử lý nghiệp vụ được cài đặt trên máy chủ, nhận được yêu cầu từ tầng giao diện, nó phải thực hiện một loạt các công việc để trả về kết quả tìm kiếm.

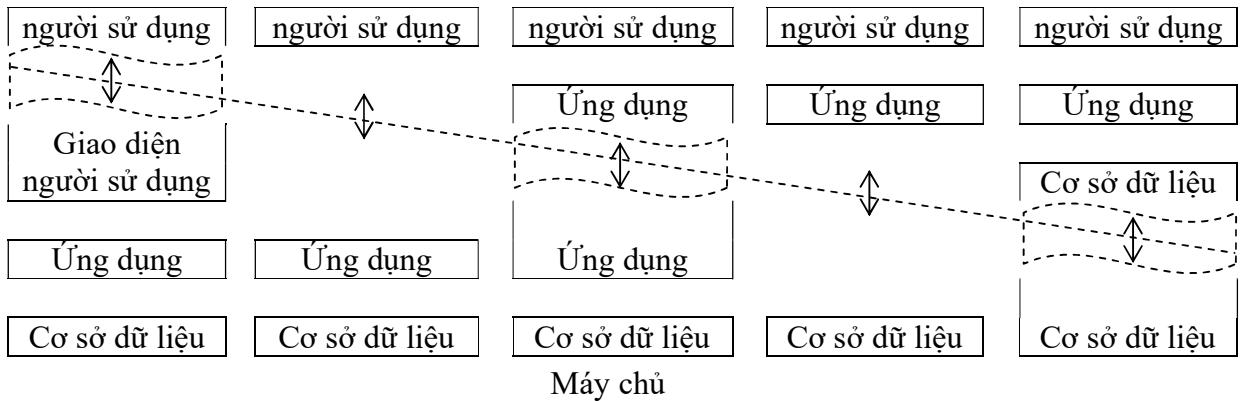
Phía sau của trang tìm kiếm đó là cả một cơ sở dữ liệu rất lớn chứa đường dẫn đến các trang tin khác nhau, để tăng hiệu năng thì chúng đều được đánh chỉ mục. Lõi của máy tìm kiếm là một chương trình chuyển đổi xâu ký tự tìm kiếm thành một hoặc nhiều câu lệnh truy vấn gửi đến cơ sở dữ liệu, tập bản ghi nhận được sẽ được sắp xếp theo thứ bậc và chuyển thành xâu ký tự định dạng HTML và trả về cho máy khách để hiển thị trên các trình duyệt.

Tầng dữ liệu là một kho dữ liệu phục vụ cho các yêu cầu tìm kiếm, nội dung của nó ít nhất cũng phải chứa tên miền và tiêu đề cũng như nội dung tóm tắt của các bài viết, nó thực hiện câu lệnh truy vấn và trả về cho tầng xử lý nghiệp vụ. Mặc dù chức năng sắp xếp được đặt ở tầng xử lý nghiệp vụ, trong một số trường hợp có thể cài đặt tại tầng dữ liệu, mặc dù làm tăng thời gian xử lý trên máy chủ dữ liệu nhưng lại giảm lượng thông tin lưu chuyển trên mạng.

Hình 1.10 liệt kê các phương án bố trí nhiệm vụ xử lý trên máy khách và máy chủ, cách tổ chức đơn giản nhất là máy khách chứa chương trình cài đặt các chức năng tầng giao diện và một số máy chủ thực hiện các chức năng tầng nghiệp vụ và tầng dữ liệu. Với cách tổ chức như vậy thì máy chủ thực hiện mọi công việc và về bản chất thì máy khách vẫn chỉ là đầu cuối cảm, có chăng chỉ là vài công việc hiển thị.

#### Máy khách





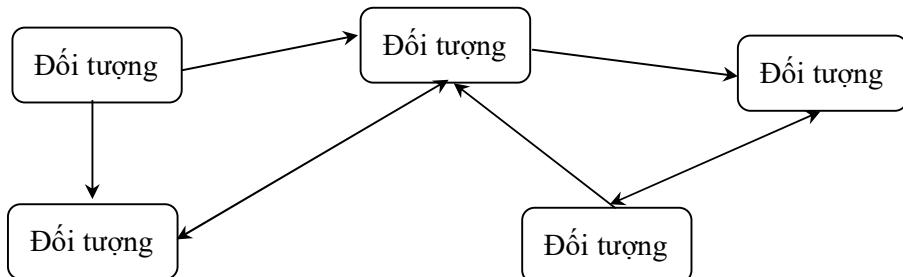
Hình 1.10 Phân chia xử lý giữa máy khách và máy chủ

Nếu chia nhỏ thêm mỗi tầng thì có nhiều lựa chọn khác nhau để phân chia nhiệm vụ giữa tiến trình trên máy khách và tiến trình trên máy chủ, tận dụng tối đa khả năng xử lý của các thiết bị trong hệ thống. Chuyển các chức năng của tầng xử lý nghiệp vụ và thậm chí một phần chức năng của tầng dữ liệu về cho máy khách không phải là giải pháp tốt, nguy cơ lộ thông tin truy nhập cơ sở dữ liệu và làm cho hệ thống phụ thuộc nhiều hơn vào máy khách.

#### 1.6.2.2 Mô hình đối tượng phân tán

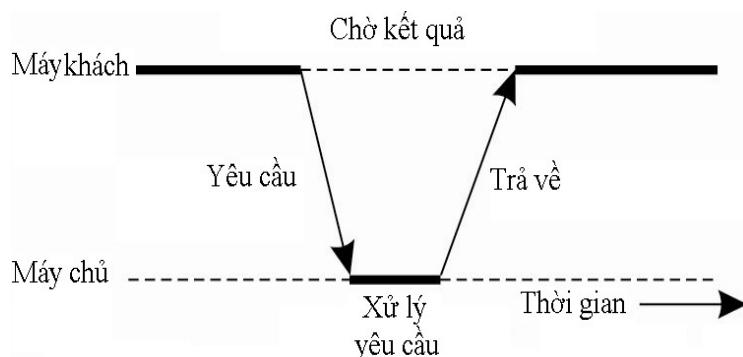
Cùng với mô hình phân tầng, mô hình đối tượng phân tán đóng vai trò quan trọng trong các hệ thống phân tán lớn. Mô hình đối tượng phân tán ràng buộc lỏng hơn mô hình phân tầng, mỗi đối tượng được coi là một thành phần và được kết nối thông qua cơ chế gọi thủ tục từ xa. Các đối tượng được gọi trực tiếp nên thời gian trễ thấp và do đó rất phù hợp với những ứng dụng yêu cầu thời gian thực, nhược điểm của nó nằm ở việc đối tượng gọi không biết đối tượng bị gọi có sẵn sàng đáp ứng yêu cầu dịch vụ hay không, nếu quá nhiều đối tượng gửi yêu cầu đến một đối tượng thì gây ra hiện tượng quá tải. Nhược điểm thứ hai là hiện tượng phong tỏa bên gọi, thời gian chờ đợi sẽ phụ thuộc vào thời gian vận chuyển dữ liệu cộng với thời gian xử lý yêu cầu, điều này có thể dẫn tới hiện tượng lỗi quá thời gian, tuy nhiên có thể khắc phục bằng cơ chế không đồng bộ.

Hình 1.11 minh họa mô hình đối tượng phân tán, mỗi đối tượng là một thể hiện của lớp, điều này giống như lập trình hướng đối tượng, mỗi lớp có các thuộc tính và phương thức xử lý, như vậy nó sẽ dễ hiểu và dễ dàng nâng cấp khi cần thiết. Mỗi tên hai chiều thể hiện các đối tượng có thể gọi các hàm của nhau, mỗi tên một chiều thể hiện một đối tượng chỉ có thể gọi các hàm của đối tượng khác. Ví dụ điển hình cho mô hình này là DCOM của Microsoft hoặc RMI trong Java, nó cho phép các ứng dụng dễ dàng trao đổi thông tin với nhau trên môi trường mạng. Trong thực tế, rất nhiều hệ thống phân tán đã được xây dựng theo cách kết hợp mô hình phân tầng và mô hình đối tượng phân tán.



Hình 1.11 Mô hình đối tượng phân tán

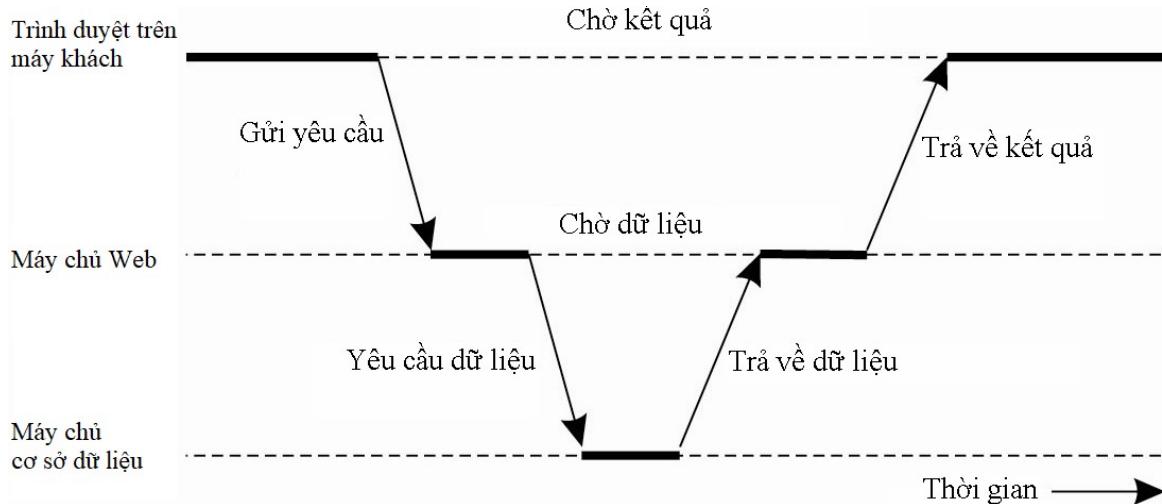
Các đối tượng trong mô hình này hoạt động tương đối độc lập, dễ dàng thay đổi và nâng cấp, đây là mô hình đã được lựa chọn để triển khai cho phương pháp tương tác giữa máy khách và máy chủ, do đó gọi là mô hình khách/chủ. Mô hình này được triển khai bằng cách cài đặt đối tượng cung cấp dịch vụ trên máy chủ, hình 1.12 thể hiện nguyên lý hoạt động của mô hình này, máy khách gọi các hàm trong đối tượng và chờ đợi kết quả trả về.



Hình 1.12 Mô hình khách/chủ

Khái niệm khách và chủ chỉ là tương đối, một đối tượng có thể đóng vai trò chủ đối với đối tượng này nhưng lại đóng vai trò khách đối với đối tượng khác, hình 1.13 thể hiện những vai trò này. Thông thường, hệ thống phân tán thường được xây dựng theo mô hình nhiều bên, máy khách không bao giờ có thể truy nhập trực tiếp vào cơ sở dữ liệu, quá trình phải thực hiện qua nhiều máy chủ xử lý nghiệp vụ. Cách làm này không những đảm bảo khả năng cân bằng tải và mở rộng qui mô hệ thống mà điều quan trọng hơn là đảm bảo an toàn cho dữ liệu.

Ví dụ khi truy nhập trang tin điện tử, trình duyệt trên máy khách gọi đối tượng nằm trên máy chủ web, để xử lý yêu cầu nó tiếp tục gọi đối tượng trên máy chủ cơ sở dữ liệu, như vậy đối tượng nằm trên máy chủ web đóng vai trò chủ đối với máy khách của người sử dụng nhưng lại là khách đối với đối tượng trên máy chủ cơ sở dữ liệu.

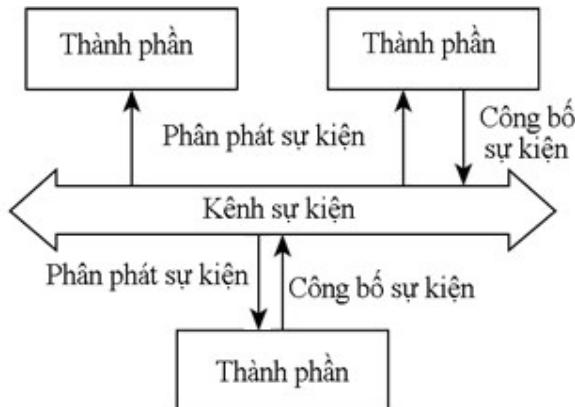


Hình 1.13 Máy chủ đóng vai trò máy khách khi truy nhập cơ sở dữ liệu

Xử lý phía máy chủ có xu hướng phân tải cho nhiều máy tính, đối tượng trên máy chủ có thể phân chia thành nhiều tác vụ nhỏ hơn, mỗi tác vụ lại gọi đến đối tượng nằm trên máy chủ khác. Tương tác khách/chủ có thể thực hiện dưới hình thức đồng bộ hoặc không đồng bộ, nếu sử dụng phương thức đồng bộ thì bên khách sẽ bị phong tỏa khi chủ đang thực hiện, do đó hiệu năng không cao. Nhiều ứng dụng phân tán sử dụng phương thức không đồng bộ, nó tạo điều kiện áp dụng kỹ thuật xử lý song song nhằm mục đích nâng cao hiệu năng nhưng vẫn đảm bảo tính tin cậy.

#### 1.6.2.3 Mô hình kênh sự kiện

Trong mô hình kênh sự kiện, các tiến trình trao đổi thông tin với nhau thông qua sự lan tỏa trên kênh sự kiện, sự kiện có thể mang theo dữ liệu và thường được gắn với các luật phân phát sự kiện. Các tiến trình phát tán sự kiện sau khi đã được phần mềm trung gian đảm bảo chỉ những tiến trình đã đăng ký mới nhận được sự kiện. Mức độ ràng buộc giữa các tiến trình của mô hình này tương đối thấp, về nguyên tắc thì các thành phần không cần phải tham chiếu đến nhau, như vậy quan hệ giữa các thành phần có thể coi như những cặp lồng.



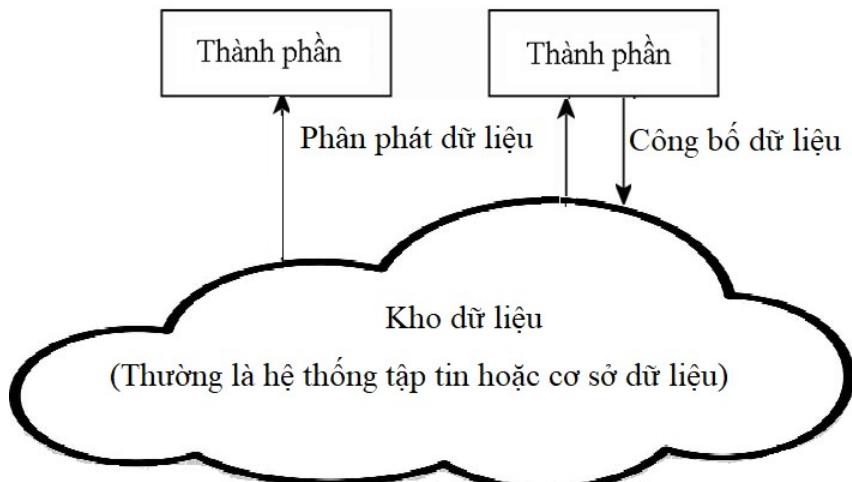
Hình 1.14 Mô hình kênh sự kiện

Hình 1.14 thể hiện nguyên lý hoạt động của mô hình này, kênh sự kiện thực chất là một tiến trình làm trung gian cho các tiến trình khác trong hệ thống, nó có một tác tử

luôn theo dõi vận hành của các thành viên trong hệ thống. Mỗi thành phần cung cấp dịch vụ sẽ đăng ký với kênh sự kiện, khi một tiến trình thành viên có yêu cầu dịch vụ, nó gửi đến kênh sự kiện, kênh sự kiện sẽ tiếp nhận và tìm kiếm tiến trình có thể đáp ứng yêu cầu và kết nối các thành viên đó để trao đổi với nhau. Đây là mô hình rất phù hợp để xây dựng các hệ thống có khả năng phân tải và dự phòng nóng, nên sử dụng CORBA hoặc JAVA RMI làm nền tảng cho các ứng dụng này.

#### 1.6.2.4 Mô hình dữ liệu tập trung

Ba mô hình đã đề cập trên đòi hỏi các tiến trình cung cấp dịch vụ luôn phải ở trạng thái hoạt động, hơn nữa nếu lượng dữ liệu lớn có thể dẫn tới hiện tượng quá thời gian. Mô hình dữ liệu tập trung phát triển dựa trên ý tưởng các tiến trình trao đổi thông tin với nhau qua kho dữ liệu chung theo phương thức chủ động hoặc thụ động. Mô hình này đảm bảo tính độc lập giữa các thành phần trong hệ thống và đồng thời tiện lợi cho việc chia sẻ dữ liệu lớn.



Hình 1.15 Mô hình dữ liệu tập trung

Mô hình kênh sự kiện có thể kết hợp với mô hình dữ liệu tập trung để hình thành nên các không gian dữ liệu chia sẻ, các tiến trình ràng buộc lỏng, chúng không cần phải đồng thời ở trạng thái hoạt động. Nhiều không gian dữ liệu chia sẻ cung cấp giao diện để truy nhập kho dữ liệu và như vậy có thể truy nhập bằng cách sử dụng mô tả chứ không cần phải tham chiếu, ví dụ điển hình là hệ thống chia sẻ tập tin.

#### 1.6.3 Kiến trúc hệ thống và phần mềm trung gian

Phần mềm trung gian hình thành tầng mới nằm giữa các ứng dụng và nền tảng của hệ thống phân tán, nó cung cấp tính năng trong suốt về phân phối dữ liệu, xử lý và điều khiển từ các ứng dụng. Sử dụng phần mềm trung gian làm cho việc thiết kế các hệ thống phân tán trở nên đơn giản hơn, tuy nhiên nó có thể không phải là giải pháp tối ưu nhất đối với các phần mềm ứng dụng, có thể khắc phục nhược điểm này bằng cách cung cấp các phiên bản phần mềm trung gian và phần mềm ứng dụng có thể dễ dàng cấu hình để tùy biến theo yêu cầu thực tiễn.

Phần mềm trung gian cung cấp nhiều tính năng khác nhau như thiết lập phiên làm việc giữa các tiến trình, bảo mật dữ liệu, nén/giải nén dữ liệu, xử lý lỗi. Tầng trung gian

đóng vai trò kết nối tiến trình máy khách với tiến trình máy chủ, nó giao tiếp với các tiến trình qua giao diện lập trình ứng dụng. Phần mềm trung gian trên máy khách thực hiện các chức năng cung cấp giao diện lập trình ứng dụng, thiết lập liên kết với tiến trình trên máy chủ bằng cách gửi các lệnh thông qua giao diện mạng và phần mềm trung gian của máy chủ. Phần mềm trung gian trên máy chủ giám sát các yêu cầu từ phía máy khách và gọi các tiến trình máy chủ tương ứng, nó tiếp nhận các yêu cầu từ phía máy khách và chuyển các yêu cầu đó cho tiến trình máy chủ, kiểm tra bảo mật hệ thống, xử lý tương tranh khi đồng thời nhận được nhiều yêu cầu từ phía máy khách, nhận kết quả xử lý của tiến trình máy chủ để trả về tiến trình trên máy khách, giám sát và xử lý lỗi.

Hình 1.16 thể hiện vị trí của tầng trung gian trong mô hình khách/chủ, các thành phần cơ bản trong mô hình này bao gồm tiến trình máy khách, máy chủ, phần mềm trung gian, dịch vụ mạng, dịch vụ cục bộ/dịch vụ mạng để quản lý dữ liệu và quản lý tiến trình, hệ điều hành và thiết bị phần cứng. Tiến trình máy khách bao gồm các chức năng của tầng ứng dụng, nó thực hiện giao tiếp với người sử dụng và các chức năng ứng dụng cần thiết như hiển thị thông tin, tính toán các bảng tính... Tiến trình máy khách giao tiếp với các phần mềm trung gian qua giao diện lập trình ứng dụng để gửi các yêu cầu đến máy chủ và nhận kết quả trả về. Tiến trình máy chủ thực hiện các chức năng tầng ứng dụng, nó cung cấp các dịch vụ cho máy khách với việc che giấu các thông tin riêng, đảm bảo cung cấp các dịch vụ xử lý lỗi và có thể thực hiện chức năng giám sát/điều phối.



Hình 1.16 Tầng trung gian trong mô hình khách/chủ

Chức năng chính của tầng trung gian là đảm bảo sự phối hợp và đồng bộ giữa các tiến trình trong hệ thống phân tán, các yêu cầu về tính trong suốt đòi hỏi những kỹ thuật khác nhau để tầng trung gian có khả năng thích nghi. Khả năng thích nghi có thể đạt được bằng cách xây dựng các tính năng giám sát và thực hiện các biện pháp điều chỉnh thích hợp, đạt được những yêu cầu này thì có thể coi đó là hệ thống tự trị. Các hệ thống tự trị thường được tổ chức dưới dạng các vòng lặp kiểm soát hồi qui, chúng được vận hành dựa trên bộ quy tắc quản lý duy nhất.

#### 1.6.4 Tự quản lý trong các hệ thống phân tán

Các hệ thống phân tán và phần mềm trung gian cần phải cung cấp giải pháp tổng thể đương đầu với những vấn đề ngoài mong muốn thường xảy ra trên mạng để chúng có thể hỗ trợ nhiều ứng dụng nhất có thể. Nhiều ứng dụng cũng không thực sự mong muốn tính trong suốt hoàn toàn về phân phối, điều đó đòi hỏi hệ thống phân tán phải hỗ trợ các giải pháp riêng cho từng ứng dụng, do đó hệ thống phân tán phải có khả năng thích ứng,

điều chỉnh hành vi thực hiện của chúng chứ không phải các phần mềm cấu thành hệ thống phân tán. Nếu sự thích ứng này cần được thực hiện tự động thì sẽ có sự tác động lẫn nhau giữa kiến trúc hệ thống và kiến trúc phần mềm. Một mặt, cần tổ chức các thành phần của hệ thống phân tán sao cho có thể thực hiện việc giám sát và điều chỉnh, mặt khác cần quyết định nơi thực hiện các tiến trình để xử lý việc điều chỉnh. Ví dụ các hệ thống điều khiển phản hồi cho phép tự động thích ứng với các thay đổi, chúng tự quản lý, tự cấu hình và tối ưu hóa, thậm chí có thể tự phục hồi.

## THẢO LUẬN

1. Liệt kê một số hệ thống phân tán thể hiện tập hợp các máy tính hoạt động độc lập nhưng gắn kết như một thể thống nhất.
2. Tìm hiểu các tiêu chuẩn mã hóa văn bản, âm thanh, hình ảnh và video.
3. Tìm hiểu các tiêu chuẩn mã hóa dữ liệu trong các hệ thống thanh khoản trực tuyến.
4. Hệ thống có khả năng chịu lỗi là gì, cho ví dụ.
5. Tìm hiểu các tính năng chịu lỗi trong các hệ điều hành cài đặt cho máy chủ.
6. Trình bày những khó khăn khi triển khai tính trong suốt.
7. Tìm hiểu những ưu điểm và nhược điểm của điện toán đám mây.
8. Trình bày các kiến trúc hệ thống phân tán, cho ví dụ.
9. Trình bày các mô hình hệ thống phân tán, đánh giá ưu điểm và nhược điểm của từng mô hình.
10. Nêu những nguyên nhân suy giảm hiệu năng, các giải pháp khắc phục.

## CHƯƠNG 2: TRAO ĐỔI THÔNG TIN TRONG HỆ THỐNG PHÂN TÁN

Bản thân định nghĩa hệ thống phân tán đã cho thấy tầm quan trọng của việc trao đổi thông tin giữa các tiến trình, chúng sử dụng cơ chế chuyển thông điệp do mạng máy tính đảm nhiệm, quá trình đó phức tạp hơn rất nhiều so với việc trao đổi thông tin giữa các tiến trình trên một máy tính. Về bản chất, cơ chế chuyển thông điệp vẫn sử dụng các giao thức đã qui định trong các mô hình OSI và TCP/IP, dữ liệu được đóng gói qua mỗi tầng trước khi chuyển qua mạng để gửi đến bên nhận.

Hệ điều hành đã tích hợp các chức năng của các tầng thấp, chỉ cần đưa thông điệp vào cổng đã qui định, mọi công việc vận chuyển đến đích đã có hệ điều hành và mạng thực thi. Cách làm này đã đơn giản hóa rất nhiều cho việc phát triển các ứng dụng, nhưng tính trong suốt của phương thức này khá thấp, nó vẫn đòi hỏi phải thực hiện rất nhiều tác vụ và xử lý các tình huống lỗi. Hơn nữa, có rất nhiều tiến trình tham gia hệ thống phân tán, nếu chỉ sử dụng các phương pháp truyền thông nguyên thủy thì sẽ rất khó phát triển các ứng dụng phân tán.

Cách tiếp cận tính trong suốt truyền thông kế thừa những thành tựu của công nghệ phần mềm, ban đầu người ta áp dụng phương pháp gọi thủ tục từ xa, bản chất của nó che giấu những thủ tục phức tạp trong việc chuyển thông điệp. Từ đầu những năm 1990, một loạt công nghệ đối tượng phân tán ra đời, Microsoft trình diễn mô hình đối tượng phân tán DCOM hỗ trợ các trình biên dịch C++, gọi phương thức từ xa RMI được tích hợp trong ngôn ngữ Java. Nhận thấy các công nghệ trên không hỗ trợ các trình biên dịch khác nhau, môi trường đối tượng chung CORBA ra đời cho phép liên kết các ứng dụng được phát triển trên các ngôn ngữ lập trình khác nhau. Cùng với sự phát triển của và hoàn thiện các giao thức tầng ứng dụng của Internet, khái niệm dịch vụ web xuất hiện đã nhanh chóng được cộng đồng phát triển phần mềm đón nhận, ngày nay truyền thông của rất nhiều hệ thống phân tán được xây dựng dựa trên cách tiếp cận này.

Truyền dữ liệu chính xác và giảm thiểu thời gian vận chuyển là những mục tiêu quan trọng, nhưng như thế chưa đủ, một số loại dữ liệu đòi hỏi các yêu cầu khác. Ví dụ, dữ liệu âm thanh đòi hỏi tính liên tục, dữ liệu video không những đòi hỏi tính liên tục mà còn yêu cầu tính đồng bộ, do đó cần phải nghiên cứu các giải pháp vận chuyển những loại dữ liệu này.

Vấn đề cuối cùng cần nghiên cứu là việc sử dụng phương pháp truyền thông điểm–điểm hay truyền thông theo nhóm. Truyền thông điểm–điểm sẽ chỉ có sự hiện diện của bên gửi và nhận, truyền thông theo nhóm sẽ có nhiều thành phần cùng tham gia, như vậy vấn đề sẽ phức tạp hơn rất nhiều, không những nảy sinh các vấn đề về hiệu năng mà còn xuất hiện nhiều vấn đề khác cần phải giải quyết.

### 2.1 Cơ sở truyền thông

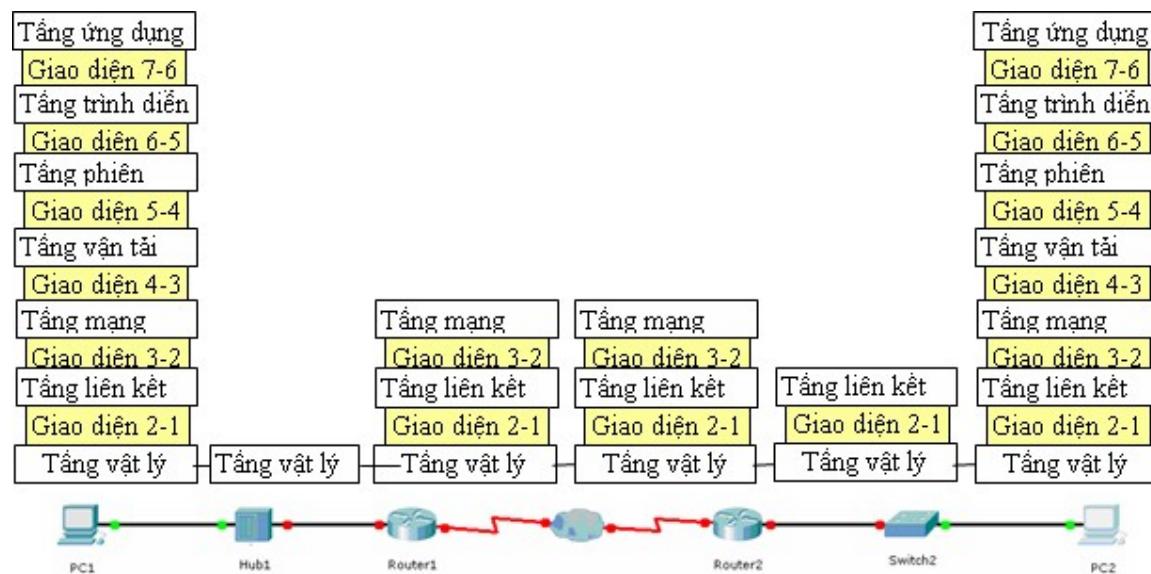
Trước khi giới thiệu các phương pháp truyền thông áp dụng trong các hệ thống phân tán, cần phải nhắc lại một số kiến thức cơ bản về các giao thức mạng để hiểu nguyên lý hoạt động của chúng. Nắm bắt nguyên lý hoạt động của chúng không những

giúp cho xây dựng các thành phần truyền thông mà còn có thể tối ưu hóa để hạn chế lỗi và nâng cao hiệu năng hệ thống.

### 2.1.1 Giao thức mạng

Các tiến trình trong hệ thống phân tán không dùng chung bộ nhớ, do đó việc trao đổi thông tin phải dựa hoàn toàn bằng phương pháp chuyển thông điệp. Khi một tiến trình A muốn trao đổi thông tin với tiến trình B, nó tạo một thông điệp trong vùng nhớ riêng của mình và thực hiện lời gọi hệ thống, khi đó hệ điều hành sẽ thực hiện chức năng chuyển thông điệp qua mạng đến hệ điều hành của máy B và từ đó chuyển cho tiến trình B. Về nguyên lý thì đơn giản như vậy, trong thực tế quá trình này khá phức tạp bởi trong hệ thống phân tán có thể có các máy tính thuộc các nhà sản xuất khác nhau và sử dụng tiêu chuẩn mã hóa thông tin khác nhau.

Để khắc phục vấn đề này, tổ chức chuẩn hóa Quốc tế ISO đã đưa ra mô hình liên kết các hệ thống mở OSI. Mặc dù các giao thức trong mô hình OSI ít được sử dụng, tuy nhiên đó là mô hình khá tốt để hiểu nguyên lý hoạt động của mạng máy tính. Mô hình OSI được phân thành 7 tầng như trên hình 2.1, mỗi tầng bao gồm các giao thức qui định khuôn dạng dữ liệu và các thủ tục xử lý như cách gửi/nhận đơn vị dữ liệu, cách xử lý khi xảy ra lỗi.



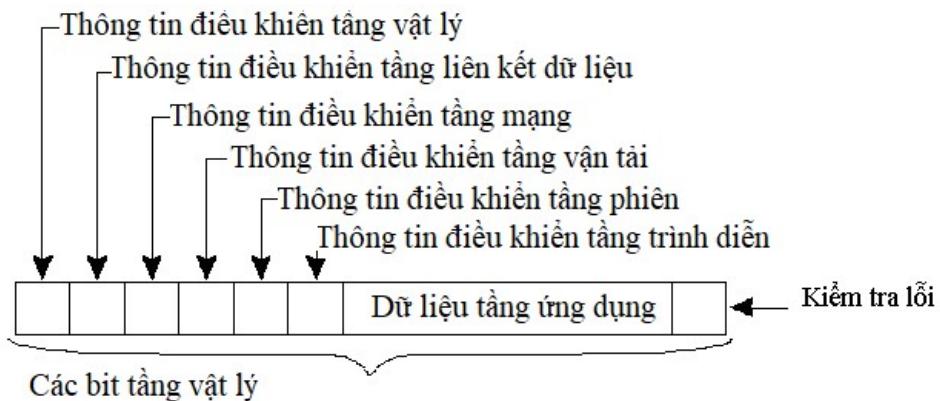
Hình 2.1 Mô hình tham chiếu liên kết các hệ thống mở

Mỗi tầng trong mô hình OSI thực hiện một số chức năng nhất định và không trùng lặp. Tầng ứng dụng không những cung cấp giao diện phục vụ cho người sử dụng mà còn cho các ứng dụng khác, các yêu cầu giao diện người sử dụng đã được Nielsen and Molich tổng kết thành 10 qui tắc khi thiết kế. Chức năng thứ nhất của tầng trình diễn là mã hóa/giải mã dữ liệu, dữ liệu dạng xâu ký tự thường được mã hóa dựa trên hai tiêu chuẩn ASCII hoặc Unicode, dữ liệu âm thanh được mã hóa theo tiêu chuẩn G711 do tổ chức viễn thông quốc tế qui định, dữ liệu video thường được mã hóa dựa trên tiêu chuẩn H264. Sau khi mã hóa, sự đa dạng các loại dữ liệu của người sử dụng sẽ được thể hiện thông nhất dưới dạng chuỗi các byte dữ liệu. Để giảm thiểu lượng thông tin lưu chuyển trên

mạng thì dữ liệu phải được nén, đó là chức năng thứ hai của tầng trình diễn. Tương tự như vậy, nếu người sử dụng có nhu cầu bảo mật thì chức năng mã hóa cũng được thực hiện ở tầng này, lưu ý đây là mã hóa bảo mật chứ không phải mã hóa để thống nhất các loại dữ liệu.

Tầng phiên cung cấp cơ chế để quản lý hội thoại giữa các tiến trình ứng dụng của người sử dụng, giao thức SIP thường được sử dụng để khởi tạo các phiên trong các ứng dụng hội thoại trực tuyến. Tầng vận tải tạo liên kết giữa các tiến trình trên các thiết bị đầu cuối của người sử dụng, nó còn có các chức năng điều khiển tốc độ truyền dữ liệu và xử lý lỗi truyền tin, hai giao thức TCP và UDP trong mô hình TCP/IP hoạt động ở tầng này. Tầng mạng thực hiện chức năng quản lý địa chỉ logic của các thiết bị tham gia vào mạng, duy trì bảng định tuyến và tìm đường đi tốt nhất cho mỗi gói tin, mạng Internet hiện nay sử dụng giao thức IP để vận chuyển các gói tin.

Tầng liên kết dữ liệu thực hiện thiết lập liên kết và đảm bảo truyền thông tin cậy giữa hai thiết bị vật lý kề cạnh nhau, nghĩa là hai thiết bị kết nối trực tiếp với nhau qua kênh truyền vật lý. Các tiêu chuẩn Ethernet thường được sử dụng trong các mạng cục bộ, liên kết giữa các thiết bị trên mạng diện rộng thường sử dụng giao thức HDLC hoặc PPP. Tầng vật lý chuyển đổi luồng byte dữ liệu thành các bit, chuyển đổi các bit thành các tín hiệu phù hợp với môi trường truyền dẫn và thực hiện thu phát các tín hiệu đó.



Hình 2.2 Bao đóng dữ liệu tại các tầng của mô hình OSI

Một tiến trình muốn gửi dữ liệu cho tiến trình trên máy tính khác, nó tạo một bản tin tại tầng ứng dụng và bản tin đó lần lượt được chuyển đến các tầng dưới trên máy tính đó. Khi đi qua mỗi tầng, thông tin điều khiển của giao thức được thêm vào bản tin, quá trình đó gọi là bao đóng dữ liệu. Ở chiều ngược lại, luồng các bit nhận được từ tầng vật lý được tập hợp lại thành các khung dữ liệu và lần lượt chuyển qua các tầng, đến mỗi tầng sẽ bóc tách các thông tin điều khiển và chỉ chuyển phần dữ liệu cho tầng trên cho đến khi tầng ứng dụng nhận được sẽ là thông điệp gốc của bên gửi.

Chức năng các tầng của mô hình OSI rất rõ ràng nhưng chúng chỉ để tham khảo, trong thực tế mạng Internet xây dựng dựa trên mô hình TCP/IP. Thêm thông tin điều khiển để hình thành đơn vị dữ liệu của giao thức chỉ là một công việc, tầng liên kết dữ liệu chỉ cho phép tối đa 1500 byte trong mỗi lần vận chuyển, kích thước tối đa cho các

đơn vị dữ liệu tầng mạng và tầng vận tải không quá 65536 byte, điều đó chứng tỏ hầu hết dữ liệu của tầng ứng dụng sẽ bị phân mảnh ở bên gửi và tập hợp lại ở bên nhận.

Dựa trên cơ chế vận hành giữa các thực thể truyền thông, người ta phân biệt giao thức có liên kết và không liên kết. Giao thức có liên kết nghĩa là phải thực hiện một số thủ tục bắt tay để đảm bảo hai bên đã sẵn sàng thì mới bắt đầu truyền dữ liệu, sau khi hoàn thành thì phải hủy bỏ liên kết để giải phóng tài nguyên mạng. Ưu điểm của loại này là tính tin cậy cao, dữ liệu chỉ được truyền đi khi các bên đã sẵn sàng, thậm chí phải phản hồi thông điệp xác nhận. Tuy nhiên, nhược điểm lộ rõ trong cơ chế hoạt động, lượng thông tin điều khiển nhiều hơn, bên gửi luôn phải chờ đợi bản tin xác nhận. Giao thức không liên kết thực hiện gửi dữ liệu mà không cần biết bên nhận đã sẵn sàng hay chưa, nó không cần duy trì kênh liên kết và cũng không cần biết thông điệp có đến đích hay không, tính tin cậy sẽ do các giao thức ở tầng trên đảm nhiệm. Không có ràng buộc giữa các bên tham gia và thông tin điều khiển ít hơn làm cho hiệu năng hoạt động của các giao thức loại này cao hơn.

#### 2.1.1.1 Giao thức các tầng thấp

Đây là những giao thức nằm ở ba tầng thấp nhất của mô hình OSI hoặc hai tầng thấp nhất của mô hình TCP/IP, chúng được tích hợp trong các hệ điều hành và các thiết bị trung gian mạng. Những giao thức trong các tầng này tập trung giải quyết hiệu năng vận chuyển dữ liệu, do đó chúng thường thuộc loại không liên kết. Mặc dù tầng liên kết dữ liệu có chức năng đảm bảo truyền thông tin cậy giữa hai thiết bị kề cạnh nhau, nhưng không có gì đảm bảo dữ liệu được truyền tin cậy giữa hai thiết bị đầu cuối. Các gói tin hoàn toàn có thể bị thất lạc hoặc bị bỏ lỡ không thể gửi đến thiết bị kế tiếp, tính tin cậy truyền thông sẽ do các giao thức tầng trên đảm nhiệm.

Thiết bị định tuyến hoạt động ở tầng mạng, gói tin phải đầy đủ thì mới có thể quyết định đường đi tiếp theo của nó, đơn vị dữ liệu tối đa lại dưới 1500 byte, nếu lớn hơn thì phải chia thành nhiều phần. Việc chia nhỏ gói tin sẽ đòi hỏi thiết bị định tuyến phải có vùng đệm đủ lớn để lưu trữ tạm thời các khung dữ liệu, nếu một khung bị lỗi hoặc thất lạc thì gói tin sẽ bị hủy bỏ, như vậy phải gửi lại dữ liệu của các giao thức tầng trên. Người phát triển các ứng dụng phân tán cần phải hiểu nguyên lý vận hành này để quyết định cách thức chia nhỏ dữ liệu ở các tầng trên, nếu chất lượng kênh truyền thấp thì nên giảm kích thước các đơn vị dữ liệu.

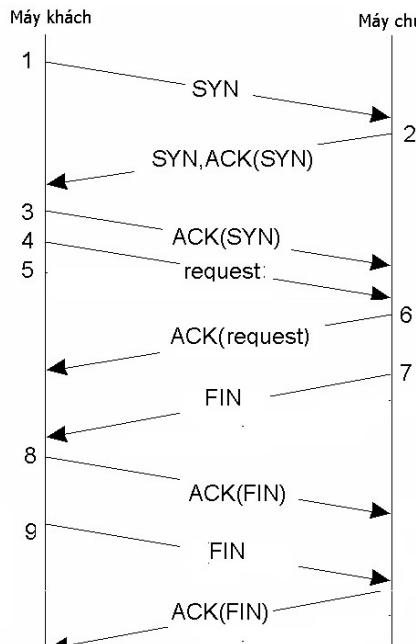
#### 2.1.1.2 Giao thức tầng vận tải

Tầng vận tải là tầng thấp nhất trong ngăn xếp các giao thức cơ sở để người lập trình viên sử dụng khi phát triển các ứng dụng phân tán, nó thực hiện trao đổi thông tin giữa các tiến trình trên thiết bị đầu cuối của người sử dụng. Mô hình TCP/IP cung cấp hai giao thức truyền thông ở tầng vận tải, giao thức có liên kết TCP và giao thức không liên kết UDP.

Các dịch vụ của những giao thức này được cung cấp thông qua các cổng, tại một thời điểm mỗi cổng chỉ phục vụ cho một tiến trình duy nhất, điều đó giải thích tại sao các giao thức tầng vận tải lại đảm bảo liên kết giữa các tiến trình trên thiết bị đầu cuối của người sử dụng. Số hiệu cổng của tiến trình cung cấp dịch vụ phải được cấp phát tĩnh

trong khi đó bên sử dụng dịch vụ thường được cấp phát động, mỗi cổng được cấp phát một vùng đệm để các tầng trên có thể gửi và nhận dữ liệu.

Giao thức UDP thuộc loại không liên kết, nó chỉ thêm thông tin điều khiển vào dữ liệu tầng và chuyển xuống tầng dưới để vận chuyển qua mạng mà không cần quan tâm đến bản tin có đến đích hay không. Giao thức TCP đòi hỏi phải thiết lập liên kết trước khi truyền dữ liệu tầng trên, khi kết thúc thì phải hủy liên kết để giải phóng tài nguyên mạng. Hình 2.3 thể hiện qui trình truyền dữ liệu của giao thức này, giai đoạn thiết lập liên kết được thực hiện qua ba bước bắt tay, hai bên sẽ thống nhất các tham số điều khiển phục vụ cho việc kiểm soát lỗi và điều khiển tốc độ truyền dữ liệu.



Hình 2.3 Qui trình truyền dữ liệu của giao thức TCP

Hệ điều hành cung cấp các hàm nguyên thủy cho hai giao thức này, nhưng truyền dữ liệu không đơn giản chỉ gửi và nhận các bản tin mà còn phải giải quyết các vấn đề lỗi và hiệu năng truyền thông. Nếu tần suất dữ liệu gửi từ các tầng trên quá lớn thì có thể dẫn đến hiện tượng tràn vùng đệm, 1518 byte là chiều dài tối đa của khung dữ liệu, do đó kích thước gói tin của tầng mạng nên nhỏ hơn 1480 byte để tránh phải phân mảnh. Tuy nhiên, các hệ điều hành thường đặt giá trị mặc định cho kích thước gói tin IP là 4096 byte, thông tin điều khiển tầng chiếm 20 byte do đó kích thước đơn vị dữ liệu tầng vận tải không nên vượt quá 4076 byte để tránh phân mảnh tại tầng này.

#### 2.1.1.3 Giao thức tầng cao

Theo mô hình OSI, các giao thức tầng cao thuộc về tầng phiên, tầng trình diễn và tầng ứng dụng, chúng sử dụng dịch vụ của các giao thức tầng vận tải thông qua các cổng, số hiệu cổng cho các dịch vụ do IANA qui định. Các cổng có giá trị nhỏ hơn 1024 dùng cho các dịch vụ chung, có thể sử dụng các cổng từ 1024 đến 49151 nhưng phải đăng ký, các cổng còn lại được phép sử dụng tự do.

Thông thường mỗi giao thức tầng cao thường chỉ sử dụng một loại giao thức tầng vận tải và một cổng dịch vụ, tuy nhiên một số giao thức có thể sử dụng cả hai loại giao thức tầng vận tải hoặc nhiều cổng dịch vụ, bảng 2.1 liệt kê một số dịch vụ cơ bản trên mạng Internet. Dịch vụ phân giải tên miền chỉ sử dụng giao thức DNS trên cổng dịch vụ số 53, khi thực hiện phân giải tên miền thì nó sử dụng giao thức UDP, giao thức TCP phục vụ cho việc đồng bộ giữa các máy chủ tên miền. Dịch vụ truyền tập tin chỉ sử dụng giao thức FTP nhưng nó lại sử dụng hai cổng, cổng 21 dành để vận chuyển thông tin điều khiển và cổng 20 dùng cho vận chuyển dữ liệu, cơ chế này tách biệt giữa vận chuyển dữ liệu và điều khiển và gọi là báo hiệu kênh ngoài.

**Bảng 2.1** Các dịch vụ và giao thức tầng cao

Dịch vụ	Giao thức	Giao thức tầng vận tải	Số hiệu cổng
Phân giải tên miền	DNS	TCP/UDP	53
Trang tin điện tử	HTTP, HTTPS	TCP	80, 443
Thư điện tử	SMTP, POP3, IMAP	TCP UDP	25, 110, 143
Cấp phát địa chỉ IP động	DHCP	UDP	67, 68
Truyền tập tin	FTP	TCP	20, 21
Truy nhập từ xa	Telnet	TCP	23

Việc sử dụng các giao thức tầng cao hoàn toàn do những người phát triển phần mềm quyết định, mỗi dịch vụ cho người sử dụng có thể phải sử dụng nhiều giao thức. Số lượng các giao thức tầng cao ngày càng nhiều, tùy theo yêu cầu phát triển hệ thống người ta có thể thêm một số các giao thức khác nhằm đơn giản hóa quá trình phát triển các ứng dụng phân tán, thông thường chúng được gộp chung vào nhóm tầng trung gian.

#### 2.1.1.4 Giao thức tầng trung gian

Phần mềm trung gian là thành phần nằm giữa ứng dụng phân tán và nền tảng hệ thống, nền tảng ở đây bao gồm phần cứng và các hệ điều hành. Hệ điều hành chỉ tích hợp các giao thức tầng thấp, các ứng dụng phân tán chỉ có thể thao tác với các giao thức tầng vận tải và tầng cao. Giao thức tầng trung gian nằm giữa tầng ứng dụng và tầng vận tải nhưng vẫn được xếp vào tầng ứng dụng, nó bao gồm nhiều giao thức đa năng để đảm bảo tính độc lập với các ứng dụng, chúng thường được thể hiện dưới dạng các dịch vụ.

Đối chiếu với mô hình phân tầng OSI, tầng trung gian thể hiện các tính năng của nó trong tầng trình diễn và tầng phiên. Việc thiết lập các tiêu chuẩn cho tầng trung gian nhằm mục đích cung cấp khả năng tương thích và tính mềm dẻo của các ứng dụng phân tán, nghĩa là phải chuẩn hóa giao diện lập trình ứng dụng và các giao thức trao đổi thông tin. Theo hai tiêu chí trên, tầng trung gian được chia thành bốn loại sau:

- Tầng trung gian mở hoàn toàn: sử dụng giao diện lập trình ứng dụng và giao thức chung để trao đổi thông tin giữa các thành viên, các sản phẩm loại này đảm bảo khả năng tương tác với nhau mà không phụ thuộc nền tảng của mỗi hệ thống, ví dụ DCE RPC, CORBA, OpenDoc.
- Tầng trung gian với giao diện lập trình ứng dụng mở: Cho phép các thành viên giao tiếp với nhau không phụ thuộc nền tảng hệ thống, các thành viên chỉ sử dụng chung các giao thức, ví dụ sản phẩm ODBC của hãng Microsoft.
- Tầng trung gian với giao thức mở: sử dụng giao thức chung nhưng cần phải chuyển đổi giao diện lập trình ứng dụng, ví dụ kiến trúc cơ sở dữ liệu quan hệ phân tán DRDA của hãng IBM là một tập hợp các giao thức cho phép giao tiếp giữa các ứng dụng và hệ thống cơ sở dữ liệu trên các nền tảng khác nhau và cho phép phân phối dữ liệu quan hệ giữa các nền tảng.
- Tầng trung gian riêng: các ứng dụng phân tán chỉ giao tiếp với nhau khi sử dụng sản phẩm của một nhà cung cấp, ví dụ ActiveX/OLE của hãng Microsoft.

Tùy theo yêu cầu sử dụng và khả năng phát triển, có thể xây dựng hệ thống phân tán theo các mức độ như truyền tập tin, mô hình khách/chủ hoặc mô hình ngang hàng. Phương pháp truyền tập tin là mức đơn giản nhất trong các hệ thống phân tán, thông tin cần trao đổi giữa các đối tượng trong hệ thống được lưu dưới dạng tập tin, các máy tính phải cùng sử dụng một giao thức để gửi và nhận tập tin. Các giao thức tầng trung gian đơn giản hóa sự phức tạp trong việc truyền dữ liệu trên mạng, nhờ có các giao thức này mà việc gọi các thủ tục từ xa sẽ được thể hiện tương tự như gọi các thủ tục trên máy cục bộ.

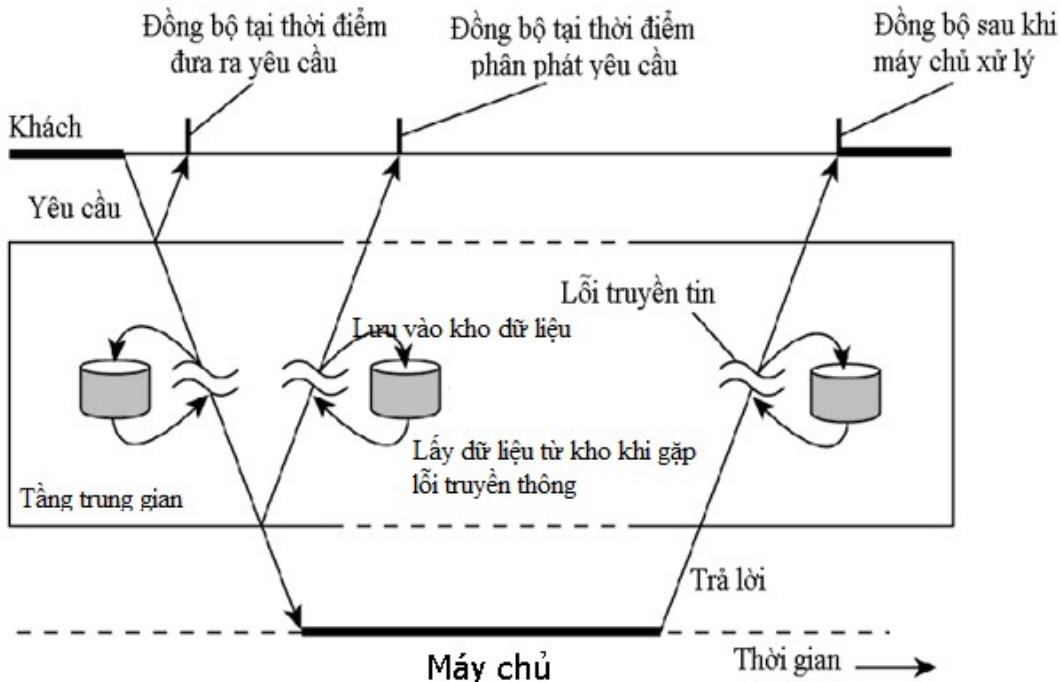
### **2.1.2 Phân loại truyền thông**

Để hiểu về các loại truyền thông mà tầng trung gian cung cấp cho các ứng dụng, chúng ta coi nó như một dịch vụ phụ trợ trong mô hình khách/chủ. Ví dụ hệ thống thư điện tử, bản chất của hệ thống này là dịch vụ truyền thông trung gian, trên mỗi máy của người dùng cài đặt phần mềm cho phép biên soạn, gửi và nhận thư điện tử. Người dùng biên soạn thư, gửi lên máy chủ thư điện tử và chờ đợi kết quả phân phát thư đó đến người nhận. Tương tự như vậy, người nhận kết nối đến hệ thống thư điện tử, kiểm tra xem có thư của mình hay không, nếu có thì hệ thống thư điện tử sẽ chuyển các bức thư đó tới máy của người dùng.

Hệ thống thư điện tử là một ví dụ về phương pháp truyền thông bền bỉ, các thông điệp của người dùng được lưu trữ trong hệ thống cho đến khi chuyển thành công đến bên nhận, bên gửi và bên nhận hoạt động hoàn toàn độc lập với nhau. Ngược lại, phương pháp truyền thông nhất thời chỉ lưu giữ thông điệp trong thời gian gửi và nhận, nghĩa là bên gửi và bên nhận phụ thuộc lẫn nhau, nếu bên nhận không hoạt động thì các thông điệp sẽ bị hủy bỏ.

Truyền thông cũng có thể được thực hiện dưới hình thức đồng bộ hoặc không đồng bộ. Trong phương thức truyền thông đồng bộ, bên gửi sẽ bị phong tỏa cho đến khi biết chắc chắn yêu cầu của mình đã được bên nhận xử lý, hình 2.4 thể hiện vai trò của tầng trung gian trong việc vận chuyển dữ liệu giữa các bên trong hệ thống. Phương pháp này đánh dấu ba thời điểm đồng bộ: Thời điểm thứ nhất bên gửi sẽ bị phong tỏa cho đến khi hệ thống trung gian tiếp nhận xong yêu cầu, thời điểm thứ hai hệ thống trung gian

thông báo đã chuyển yêu cầu cho bên nhận và thời điểm thứ ban bên gửi sẽ tiếp nhận kết quả bên nhận xử lý. Ngược lại, truyền thông không đồng bộ cho phép bên gửi tiếp tục thực hiện công việc của mình sau khi đã gửi thông điệp đến hệ thống trung gian.



Hình 2.4 Nguyên lý truyền thông sử dụng tầng trung gian

Trong thực tế, người ta thường kết hợp hai loại truyền thông trên để trao đổi thông tin giữa các tiến trình, phương pháp truyền bền bỉ và đồng bộ thường được áp dụng trong các trường hợp cần phải chuyển một lượng lớn dữ liệu trong khi đó phương pháp truyền thông tạm thời và đồng bộ sẽ chỉ phù hợp với lượng dữ liệu nhỏ, phương pháp này thường sử dụng trong kỹ thuật gọi thủ tục từ xa. Bên cạnh tính bền bỉ và tính đồng bộ người ta còn phân biệt tính rời rạc hay liên tục của truyền thông, những hệ thống mà mỗi thông điệp được truyền đi là những đơn vị dữ liệu độc lập sẽ được xếp vào nhóm rời rạc, nếu các thông điệp được truyền lần lượt và liên tục gọi là phương pháp truyền tin theo luồng.

## 2.2 Chuyển thông điệp

Thông điệp được hiểu là nội dung thể hiện thông tin cần vận chuyển cho đối tượng khác, chuyển thông điệp là phương pháp cơ bản vận chuyển dữ liệu giữa các ứng dụng trên mạng, mặc dù tính trong suốt thấp nhưng nó vẫn được ưa chuộng để phát triển nhiều hệ thống phân tán. Các phương pháp trao đổi thông tin được phát triển sau này đem lại tính trong suốt cao, nhưng bản chất của chúng vẫn sử dụng phương pháp chuyển thông điệp.

Hiểu một cách đơn giản, dữ liệu của người sử dụng được chuyển thành thông điệp và gửi qua các tầng đến bộ điều khiển giao diện mạng trên mỗi máy tính, tiếp tục các thiết bị trung gian mạng để đến máy tính bên nhận. Hệ điều hành đã tích hợp các giao thức của các tầng thấp, nó là trung gian giữa ứng dụng và bộ điều khiển giao diện mạng.

Nếu tần suất gửi thông điệp vượt quá năng lực xử lý của bộ điều khiển giao diện mạng thì sẽ dẫn đến hiện tượng mất thông điệp khi nó chưa kịp gửi lên mạng, vấn đề này được khắc phục bằng cách tạo ra vùng đệm lưu trữ tạm thời, do đó có những phương pháp chuyển thông điệp khác nhau.

### **2.2.1 Chuyển thông điệp dựa trên dịch vụ tầng vận tải**

Nhiều ứng dụng phân tán được xây dựng dựa trên dịch vụ chuyển thông điệp của tầng vận tải, trong mô hình TCP/IP đó là hai giao thức TCP và UDP, chúng đều sử dụng số hiệu cổng làm điểm truy nhập dịch vụ. Một tiến trình có thể sử dụng nhiều cổng nhưng tại một thời điểm mỗi cổng chỉ phục vụ cho một tiến trình, số hiệu cổng của các tiến trình cung cấp dịch vụ thường được cấp phát tĩnh.

Số hiệu cổng chỉ có ý nghĩa cục bộ, để có thể phân phát thông điệp sang máy tính khác thì cần địa chỉ IP. Cặp địa chỉ IP và số hiệu cổng hình thành điểm cuối liên kết giữa các tiến trình trên mạng, nó được hãng IBM đưa vào hệ điều hành Berkeley UNIX dưới tên gọi Socket. Socket trừu tượng hóa truyền thông thực tế giữa các điểm cuối cho các giao thức tầng vận tải, thông điệp được đặt vào Socket để hệ điều hành chuyển qua mạng, quá trình ngược lại được thực hiện ở bên nhận, hệ điều hành chuyển thông điệp đến cổng dịch vụ tầng vận tải để đưa lên tầng ứng dụng. Socket trở thành giao diện lập trình ứng dụng mạng, hầu hết các hệ điều hành và ngôn ngữ lập trình đều hỗ trợ, nó cung cấp các hàm nguyên thủy để thực hiện chuyển thông điệp.

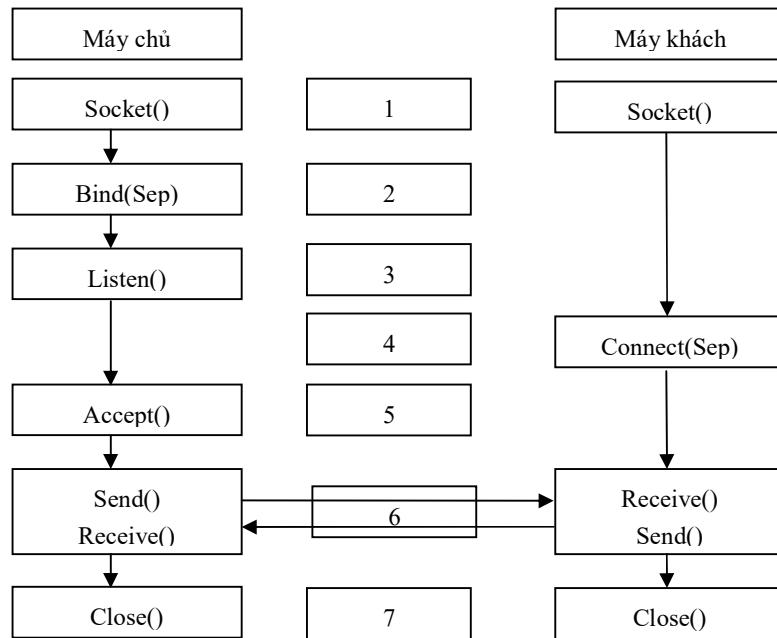
**Bảng 2.2** Các hàm nguyên thủy Socket

Tên hàm	Ý nghĩa
Socket	Tạo đối tượng điểm kết cuối
Bind	Nhúng điểm kết cuối vào socket để cung cấp dịch vụ
Listen	Đăng ký sẵn sàng tiếp nhận yêu cầu liên kết
Accept	Đồng ý tiếp nhận yêu cầu liên kết
Connect	Gửi yêu cầu thiết lập liên kết
Send	Gửi dữ liệu
Receive	Nhận dữ liệu
Close	Hủy bỏ liên kết

Ứng dụng chuyển thông điệp thường được xây dựng theo mô hình khách/chủ, tiến trình máy chủ phải luôn lắng nghe các yêu cầu của tiến trình máy khách. Hình 2.5 thể hiện lưu đồ các bước xử lý của ứng dụng khách/chủ đơn giản sử dụng giao thức TCP, trong đó Sep là điểm kết cuối, nó bao gồm địa chỉ IP của máy chủ và số hiệu cổng đăng ký dịch vụ.

Ứng dụng máy chủ khởi tạo thực thể điểm kết cuối bằng cách gọi hàm Socket() qui định giao thức sử dụng trên tầng vận tải và các tham số truyền thông, nghĩa là hệ điều hành cục bộ cấp phát vùng nhớ để phục vụ gửi/nhận các thông điệp cho giao thức đã xác định. Hàm Bind(Sep) liên kết điểm kết cuối cục bộ với đối tượng socket đã tạo, nó nhúng

địa chỉ IP của máy chủ và số hiệu cổng đăng ký dịch vụ vào đối tượng Socket để báo cho hệ điều hành biết điểm truy nhập dịch vụ. Hàm Listen() thuộc loại không phong toả, thông báo cho hệ điều hành biết điểm kết cuối đã sẵn sàng và có thể phục vụ tối đa bao nhiêu yêu cầu kết nối.



Hình 2.5 Sử dụng hàm nguyên thủy trong chuyển thông điệp

Ứng dụng máy khách cũng khởi tạo thực thể điểm kết cuối bằng cách gọi hàm Socket(), trong đó giao thức và tham số truyền thông phải phù hợp với máy chủ, không cần thiết phải nhúng địa chỉ cục bộ và số hiệu cổng vì hệ điều hành tự động gán những giá trị này khi thiết lập liên kết. Trước khi truyền dữ liệu, nó phải gọi hàm Connect(Sep), lưu ý Sep ở đây là điểm kết cuối của tiến trình máy chủ chứ không phải máy khách. Thực thể điểm kết cuối trên máy chủ luôn lắng nghe, nếu có yêu cầu gửi đến thì gọi hàm Accept() để tạo một Socket mới với các thuộc tính giống như Socket ban đầu và trả về cho tiến trình máy khách. Nó có thể được chuyển sang luồng khác để xử lý, cách tiếp cận này cho phép tiến trình máy chủ tách biệt các liên kết để có thể phục vụ nhiều yêu cầu, thực thể điểm kết cuối của tiến trình máy chủ tiếp tục đợi một yêu cầu liên kết khác.

Sau khi kênh truyền đã được thiết lập cả hai bên có thể bắt đầu trao đổi thông tin bằng cách sử dụng các hàm Send() và Receive(), khi kết thúc cả hai bên đóng liên kết bằng cách gọi hàm Close(). Nếu mất kết nối kênh truyền thì thông điệp sẽ bị hủy bỏ, do đó phương pháp sử dụng các hàm nguyên thủy được xếp vào loại truyền thông nhát thời, việc gửi lại dữ liệu hay không sẽ do các tính năng tầng trên quyết định.

Phương pháp trao đổi thông tin trong ví dụ trên thuộc loại đồng bộ, tiến trình máy khách sẽ bị phong tỏa khi gửi thông điệp đến tiến trình máy chủ, do đó tính tin cậy cao nhưng hiệu năng tương đối thấp. Dịch vụ của giao thức UDP đem lại hiệu năng cao hơn, không cần phải thiết lập liên kết trước khi truyền dữ liệu, không có ràng buộc giữa các bên nên khó kiểm soát lỗi. Vẫn sử dụng giao thức TCP nhưng áp dụng phương pháp

truyền thông không đồng bộ, hiệu năng cao hơn và có thể kiểm soát lỗi, điều này được thực hiện bằng cách truy nhập trực tiếp vùng đệm gửi/nhận của Socket.

### 2.2.2 Giao diện truyền thông điệp

Chuyển thông điệp nhất thời sử dụng các hàm nguyên thủy có mức độ trùu tượng thấp, nó chỉ đơn thuần cung cấp các hàm cơ bản phục vụ trao đổi thông tin. Hơn nữa, các Socket chỉ đáp ứng yêu cầu truyền thông trong mạng dựa trên mô hình TCP/IP, nó không phù hợp với các giao thức độc quyền dành riêng cho kênh truyền tốc độ cao. Các thiết bị mạng hiệu năng cao thường đính kèm trình điều khiển do nhà sản xuất cung cấp, chúng có thể không tương thích với nhau, nảy sinh vấn đề về tính khả chuyển của hệ thống.

Chuyển thông điệp với hiệu năng cao nhưng độc lập với phần cứng đòi hỏi phải được chuẩn hóa, một nhóm các nhà nghiên cứu bắt đầu thảo luận vấn đề này từ năm 1991 và thống nhất tên gọi là giao diện truyền thông điệp MPI. Giao diện truyền thông điệp MPI chưa được bất kỳ tổ chức tiêu chuẩn quốc tế nào xác nhận, nó được nhiều hãng công nghệ lớn hỗ trợ nhưng chưa được tích hợp trong các hệ điều hành, phát triển thư viện giao diện này thường do các tổ chức mã nguồn mở phát triển.

Giao diện truyền thông điệp MPI được thiết kế cho nhóm các ứng dụng xử lý song song, nó vẫn được xếp vào loại truyền thông nhất thời vì thông điệp có thể bị hủy nếu gặp lỗi, sự cố tiến trình hoặc mất kết nối mạng bị coi là nghiêm trọng. Giao diện truyền thông điệp giả thiết trao đổi thông tin diễn ra trong một nhóm các tiến trình đã biết, mỗi nhóm có một định danh và tiến trình trong nhóm cũng có định danh. Cặp định danh của nhóm và tiến trình xác định duy nhất nguồn hoặc đích của thông điệp, chúng được sử dụng thay cho cổng dịch vụ tầng vận tải, các nhóm và các tiến trình có thể chồng nhau và chúng đều có thể hoạt động song song.

**Bảng 2.3** Các hàm nguyên thủy giao diện truyền thông điệp

Tên hàm	Ý nghĩa
MPI_bsend	Không phong tỏa, chỉ cần chuyển thông điệp vào vùng đệm gửi
MPI_send	Phong tỏa cho đến khi thông điệp được lưu trong vùng đệm bên nhận hoặc vùng đệm tạm thời của hệ thống MPI
MPI_ssend	Phong tỏa cho đến khi thao tác nhận được kích hoạt
MPI_sendrecv	Phong tỏa cho đến khi chuyển thông điệp thành công
MPI_isend	Không phong tỏa, được coi là hoàn thành ngay cả khi thông điệp đang chuyển đến bên nhận
MPI_issend	Không phong tỏa, được coi là hoàn thành khi thông điệp đã đến hệ thống MPI của bên nhận
MPI_recv	Phong tỏa cho đến khi nhận được thông điệp
MPI irecv	Không phong tỏa ngay cả khi không có thông điệp

Bảng 2.3 tóm tắt ý nghĩa của 8 hàm nguyên thủy trong thư viện MPI, các tiến trình ứng dụng phân tán tương tác với nhau thông qua hệ thống trung gian MPI. Tiến trình đưa thông điệp vào vùng đệm, hệ thống MPI lấy thông điệp từ vùng đệm bằng cách sao chép

vào vùng đệm bên trong của nó sau đó chuyển thông điệp đến bên nhận, hệ thống MPI bên nhận lại đưa thông điệp vào vùng đệm giao tiếp với tiến trình bên nhận. Hệ thống MPI phải tốn thời gian sao chép thông điệp, nó cũng phải phục vụ cho nhiều tiến trình, do đó độ trễ có thể sẽ lớn hơn so với chuyển thông điệp bằng cách sử dụng dịch vụ của tầng vận tải.

Hàm MPI\_bsend hỗ trợ truyền thông không đồng bộ, tiến trình đưa thông điệp vào vùng đệm cục bộ sau đó có thể tiếp tục công việc khác, việc chuyển thông điệp đến bên nhận sẽ do hệ thống MPI thực hiện. Hàm MPI\_send thuộc loại phong tỏa, tiến trình gửi phải chờ cho đến khi thông điệp được chuyển đến vùng đệm của bên nhận hoặc vùng đệm tạm thời của hệ thống MPI. Việc tồn tại vùng đệm tạm thời bên trong hệ thống MPI giúp cho thông điệp được lưu trữ an toàn khi nó chưa được chuyển đến bên nhận, tránh hiện tượng ghi đè nếu tiến trình gửi thông điệp khác.

Hàm MPI\_ssend hỗ trợ truyền thông đồng bộ, tiến trình gửi bị phong tỏa cho đến khi yêu cầu của nó được chấp nhận, nghĩa là nó có thể tiếp tục chuyển thông điệp khác vào vùng đệm. Hàm MPI\_sendrecv thuộc loại truyền thông đồng bộ mạnh nhất, tiến trình chuyển thông điệp đến vùng đệm và chờ cho đến khi thông điệp đã được chuyển thành công đến tiến trình bên nhận.

Hai hàm MPI\_send và MPI\_ssend đều có các biến thể, chúng sử dụng con trỏ để tránh sao chép thông điệp từ vùng đệm tiến trình sang vùng đệm bên trong hệ thống MPI cục bộ, các biến thể này tương ứng với hình thức truyền thông không đồng bộ. Sử dụng hàm MPI\_isend, tiến trình gửi chuyển con trỏ đến thông điệp để hệ thống MPI xử lý truyền thông, sau đó ngay lập tức tiếp tục công việc khác, dù cho thông điệp có thể chưa đến bên nhận. Nhằm ngăn chặn việc ghi đè thông điệp trước khi hoàn thành chuyển thông điệp, thư viện MPI cung cấp các hàm kiểm tra xem đã hoàn thành hoặc thậm chí phong tỏa nếu cần thiết. Đối với hàm MPI\_send, thông điệp đã thực sự được chuyển đến bên nhận hay mới chỉ được sao chép vào vùng đệm bên trong hệ thống MPI vẫn chưa được qui định. Hàm MPI\_issend cũng là biến thể của hàm MPI\_ssend, bên gửi chỉ chuyển một con trỏ đến hệ thống MPI, khi hệ thống MPI cho biết thông điệp đã được xử lý, tiến trình gửi biết rằng đã chuyển thành công thông điệp đến bên nhận.

Hàm MPI\_recv đi kèm với hàm MPI\_send, dùng để nhận thông điệp, nó phong tỏa tiến trình gọi cho đến khi có thông điệp đi đến, nếu không có thông điệp thì tiếp tục chờ, nó chỉ trả về kết quả khi nhận được thông điệp. Hàm MPI irecv phục vụ cho truyền thông không đồng bộ, bên nhận có thể kiểm tra thông điệp thực sự đã đến hay chưa nhưng không phong tỏa tiến trình gọi, nó chỉ đơn thuần thông báo bên nhận đã sẵn sàng.

Ngữ nghĩa của các hàm giao diện truyền thông điệp không phải lúc nào cũng cứng nhắc, các hàm khác nhau đôi khi có thể được thay thế cho nhau mà không ảnh hưởng đến tính đúng đắn của luồng xử lý. Sự đa dạng của các hàm tạo điều kiện cho những người phát triển có những công cụ khác nhau để tối ưu hóa hiệu năng hệ thống, bản thân giao diện truyền thông điệp được thiết kế cho các ứng dụng xử lý song song.

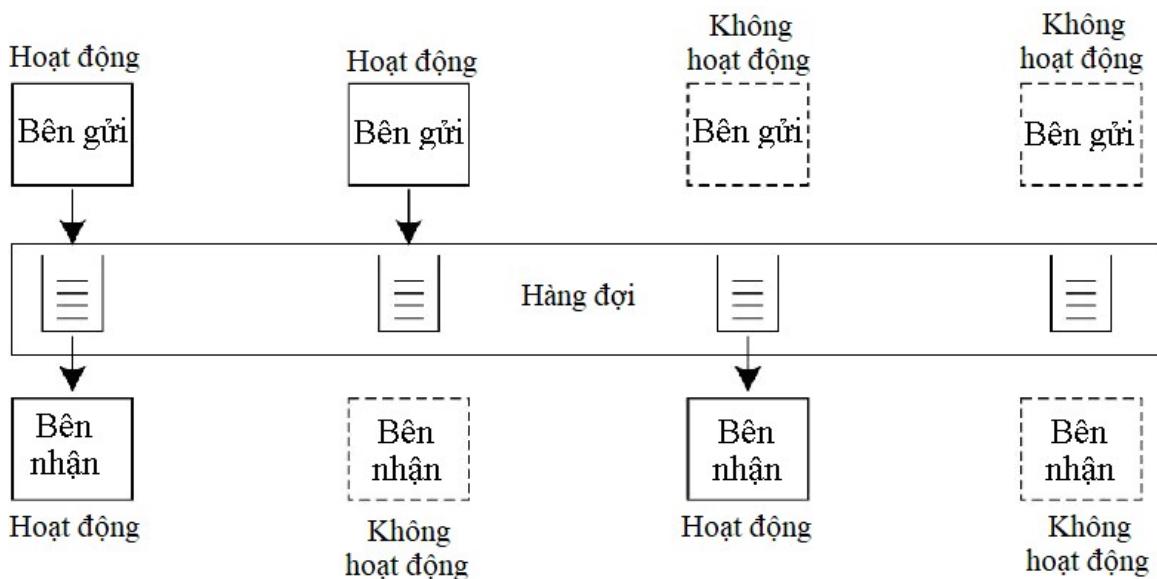
### 2.2.3 Chuyển thông điệp bền bỉ

Hai phương pháp chuyển thông điệp trên đều thuộc loại nhất thời, chúng đòi hỏi bên gửi và bên nhận đều phải cùng hoạt động, nếu bên nhận không sẵn sàng hoặc kênh truyền lỗi thì thông điệp sẽ bị loại bỏ. Dịch vụ tầng vận tải chỉ cho phép thông điệp kích thước nhỏ hơn 64 KB, giao diện truyền thông điệp cho phép lớn hơn nhưng thời gian vận chuyển cũng không được vượt quá vài giây. Theo định nghĩa truyền thông bền bỉ, thông điệp được lưu tại vùng đệm cho đến khi đảm bảo chắc chắn đã chuyển thành công đến bên nhận, để giải quyết vấn đề này người ta đã đưa ra mô hình hàng đợi thông điệp.

Hệ thống hàng đợi thông điệp hỗ trợ đặc lực truyền thông không đồng bộ, không bắt buộc các tiến trình đầu cuối phải ở trạng thái hoạt động, có thể chuyển thông điệp kích thước lớn tới mức thời gian vận chuyển lên đến vài phút. Về bản chất, khái niệm hàng đợi ở đây là mạng lưới trung gian truyền thông vận hành trên nguyên tắc đến trước thì xử lý trước, nó bao gồm các tiến trình chạy trên mạng lưới máy chủ truyền thông và máy khách.

Tiến trình ứng dụng chỉ cần chuyển thông điệp đến hàng đợi, cụ thể là tiến trình của hệ thống hàng đợi nhưng chạy trên máy tính cục bộ, trách nhiệm chuyển các thông điệp đến đích thuộc về mạng lưới các máy chủ. Nếu không thể chuyển thông điệp đến máy chủ truyền thông đầu tiên, thông điệp vẫn được lưu trong hàng đợi cục bộ và tiến trình ứng dụng vẫn có thể tiếp tục thực hiện các công việc khác, việc gửi lại thông điệp do tiến trình của hệ thống hàng đợi nhưng chạy trên máy tính cục bộ thực hiện.

Về nguyên tắc, mỗi tiến trình có một hàng đợi riêng để nhận thông điệp do các tiến trình khác gửi đến, tuy nhiên cũng có thể thiết kế để nhiều tiến trình trên một máy tính cùng chia sẻ một hàng đợi. Hệ thống hàng đợi chỉ đảm bảo thông điệp của bên gửi được chuyển đến hàng đợi của bên nhận mà không cam kết về thời gian cũng như tiến trình ứng dụng bên nhận có đọc hay không, việc đó hoàn toàn phụ thuộc vào cách xử lý của bên nhận.



Hình 2.6 Các liên kết trong mô hình hàng đợi thông điệp

Trao đổi thông tin qua hệ thống hàng đợi thông điệp không ràng buộc chặt chẽ giữa các bên, thông điệp không cần phải xử lý ngay sau khi nhận và bên gửi cũng không bị phong tỏa trong thời gian chuyển thông điệp, các tiến trình đầu cuối hoạt động hoàn toàn độc lập với nhau. Khi thông điệp đã được đưa vào hàng đợi, nó sẽ vẫn được lưu giữ ở đó, chỉ khi nào chuyển thành công đến bên nhận thì thông điệp mới bị xóa, dù cho bên gửi hay bên nhận còn hoạt động hay không, điều này cho thấy bốn khả năng xảy ra như trong hình 2.6.

Trường hợp thứ nhất, cả bên nhận và gửi đều đang ở trạng thái hoạt động, tiến trình gửi sẽ đưa thông điệp vào hàng đợi, sau đó hàng đợi sẽ gửi ngay thông điệp đến bên nhận. Trường hợp thứ hai, bên gửi đang hoạt động nhưng bên nhận lại không hoạt động, bên gửi chuyển thông điệp vào hàng đợi nhưng nó không thể phân phát đến bên nhận, thông điệp vẫn được lưu trong hàng đợi, bên gửi coi như đã hoàn thành công việc và có thể tiếp tục chuyển các thông điệp khác. Trường hợp thứ ba, bên nhận đang hoạt động nhưng bên gửi lại không ở trạng thái hoạt động, nghĩa là bên nhận vẫn có thể lấy các thông điệp từ hàng đợi. Trường hợp thứ tư, cả bên nhận và bên gửi đều ở trạng thái không hoạt động, nghĩa là thông điệp vẫn được hệ thống lưu trữ trong hàng đợi.

Về nguyên tắc, thông điệp có thể chứa bất kỳ loại dữ liệu nào, điều quan trọng mạng lưới máy chủ trung gian phải đảm bảo phân phát thông điệp chính xác, điều này được thực hiện bằng hệ thống định danh duy nhất trong hàng đợi. Nếu thông điệp có kích thước lớn, hàng đợi thông điệp có thể chia nhỏ thành nhiều phần để gửi và tập hợp chúng lại ở bên nhận, quá trình này hoàn toàn trong suốt đối với tiến trình ứng dụng. Để các tiến trình ứng dụng có thể trao đổi thông tin, hệ thống hàng đợi thông điệp cung cấp bốn hàm cơ bản, ý nghĩa của mỗi hàm được tóm tắt trong bảng 2.4.

**Bảng 2.4** Các hàm hàng đợi thông điệp

Tên hàm	Ý nghĩa
Put	Thêm thông điệp vào hàng đợi
Get	Lấy thông điệp đầu tiên và xóa, nếu không có thông điệp thì chờ
Poll	Lấy thông điệp đầu tiên và xóa, kết thúc ngay nếu không có thông điệp thì
Notify	Gọi trình xử lý khi có thông điệp trong hàng đợi

Tiến trình ứng dụng gửi thông điệp bằng cách gọi hàm Put() để chuyển thông điệp vào hệ thống hàng đợi, hàng đợi ở đây thực chất là tiến trình thuộc hệ thống hàng đợi nhưng chạy trên máy tính cục bộ. Đó là trao đổi thông tin liên tiến trình trên một máy tính nên thời gian chuyển thông điệp thường chỉ tính bằng nanô đến micro giây, do đó có thể coi hàm này thuộc loại không phong tỏa.

Hàm Get() lấy thông điệp đầu tiên từ hàng đợi sau đó thực hiện thao tác xóa, nếu hàng đợi rỗng thì chờ, do đó hàm này thuộc loại phong tỏa. Biến thể khác là hàm không phong tỏa Poll(), nó lấy thông điệp đầu tiên từ hàng đợi và xóa, nếu không tìm thấy thông điệp thì kết thúc để tiến trình ứng dụng tiếp tục thực hiện. Cuối cùng là hàm Notify(), nhiều hệ thống hàng đợi cho phép tiến trình cài đặt trình xử lý tương tự như hàm gọi lại, nó sẽ tự động được kích hoạt khi có thông điệp đưa vào hàng đợi. Cơ chế này cũng cho

phép tự động khởi tạo một tiến trình nếu như không có thành viên nào đang hoạt động, giải pháp này thường được cài đặt dưới dạng tiến trình giám sát hàng đợi ở bên nhận.

### 2.3 Gọi thủ tục từ xa

Trước năm 1990, phần mềm được xây dựng theo hướng thủ tục, nghĩa là nghiệp vụ phức tạp được chia nhỏ thành các thủ tục. Hàm cũng bao gồm những câu lệnh tương tự như thủ tục nhưng phải có giá trị trả về, do đó khi nói đến gọi thủ tục chúng ta ngầm hiểu đối tượng bị gọi có thể là thủ tục hoặc hàm. Các hàm/thủ tục có thể gọi nhau, chương trình chính là thủ tục cấp cao nhất, nó được kích hoạt ngay khi chương trình bắt đầu chạy. Thủ tục gọi chỉ cần truyền các tham số cho thủ tục con, trao đổi dữ liệu giữa chúng được thực hiện thông qua các ô nhớ trên một máy tính. Vấn đề trở nên phức tạp khi nhu cầu tính toán trên nhiều máy tính ngày càng trở nên cấp thiết, không thể gọi thủ tục trên máy tính khác theo cách truyền thống vì bộ nhớ thuộc sở hữu riêng của mỗi máy tính.

Phương pháp chuyển thông điệp mới chỉ tập trung vào các nhiệm vụ gửi và nhận dữ liệu và chưa quan tâm đến vấn đề xử lý, tính trong suốt thấp nên tiến trình ứng dụng phải xử lý nhiều công việc phức tạp. Ý tưởng gọi thủ tục chạy trên máy tính khác nhưng che giấu quá trình trao đổi thông tin, phương pháp xây dựng phần mềm gần như không thay đổi, từ đó hình thành khái niệm gọi thủ tục từ xa RPC. Về nguyên tắc, giải pháp này khá đơn giản cho việc cài đặt, tuy nhiên trong thực tế này sinh khá nhiều vấn đề như thực thi mã lệnh được thực hiện trên các vùng nhớ khác nhau hoặc nếu một trong hai máy tính bị lỗi trong quá trình thực thi mã lệnh cũng nảy sinh nhiều vấn đề phức tạp.

Các khuyến nghị về phương pháp gọi thủ tục từ xa đã được đưa ra từ năm 1975, nhưng mãi tới năm 1984 Birrell và Nelson mới hoàn thiện giải pháp triển khai thực tế, nó cũng là nền tảng để xây dựng các ứng dụng theo mô hình khách/chủ. Cơ chế gọi thủ tục từ xa đòi hỏi tiến trình máy chủ phải luôn ở trạng thái sẵn sàng tiếp nhận yêu cầu từ tiến trình máy khách, có thể áp dụng phương thức đồng bộ hoặc không đồng bộ.

#### 2.3.1 Cơ chế hoạt động

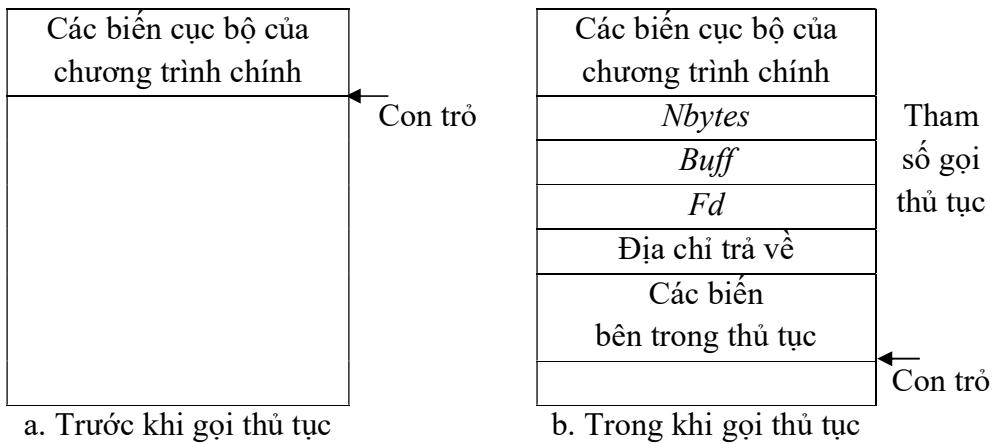
Nếu gọi thủ tục chạy trên máy tính cục bộ, hệ điều hành sẽ cấp phát không gian bộ nhớ để thực thi, nó được tổ chức theo kiểu ngăn xếp, tham số đầu tiên sẽ được chuyển cuối cùng. Sau khi thủ tục thực hiện xong, giá trị trả về sẽ được chuyển đến các thanh ghi, giải phóng vùng nhớ và chuyển quyền điều khiển cho chương trình gọi, chương trình gọi sẽ loại bỏ tham số ra khỏi ngăn xếp, trả ngăn xếp về trạng thái như trước khi gọi thủ tục. Giả sử cần đọc dữ liệu từ một tập tin, hệ điều hành đã cung cấp hàm này, chỉ cần gọi nó và kèm theo các tham số theo yêu cầu:

```
count = read(Fd, Buff, Nbytes);
```

trong đó *Fd* là con trỏ tập tin, *Buff* là vùng đệm để lưu trữ dữ liệu đọc từ tập tin, *Nbytes* là số lượng byte cần đọc.

Việc truyền tham số có thể được thực hiện bằng phương pháp truyền giá trị hay truyền tham chiếu, một số ngôn ngữ lập trình hỗ trợ cơ chế truyền con trỏ. Con trỏ là một kiểu dữ liệu đặc biệt, nó là địa chỉ của một đối tượng trong bộ nhớ, thông thường các đối tượng có thể được truy xuất trực tiếp bằng tên đại diện hoặc gián tiếp thông qua con trỏ.

Nếu một biến được định nghĩa là kiểu con trỏ thì nó sẽ chứa địa chỉ của vùng nhớ cấp cho đối tượng, nếu không sử dụng thì phải hủy con trỏ để thu hồi vùng nhớ. Con trỏ thường được sử dụng để tạo các đối tượng động trong thời gian chạy chương trình, chúng được cấp phát trong vùng nhớ heap.



Hình 2.7 Ngăn xếp khi gọi thủ tục trên máy tính

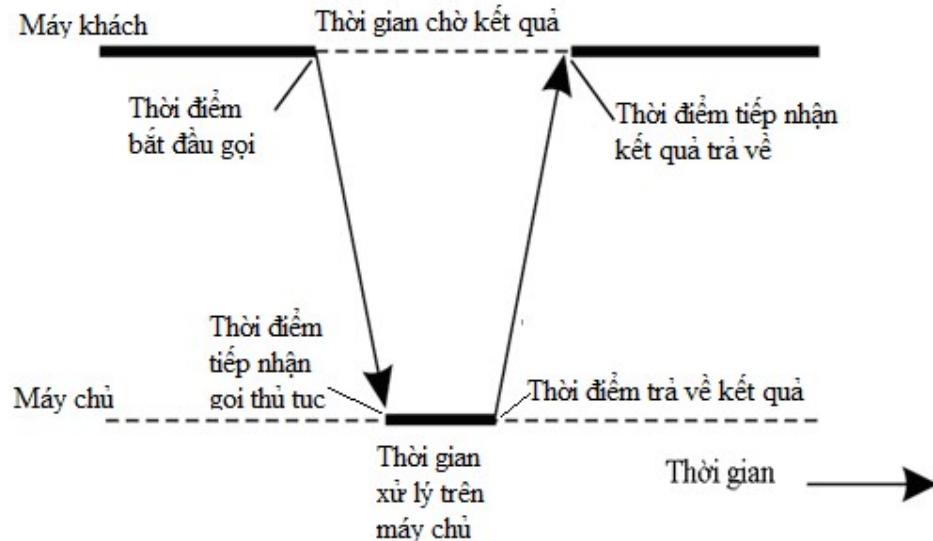
Truyền tham chiếu cung cấp một tên tượng trưng khác gọi là bí danh cho một đối tượng, truy xuất một đối tượng thông qua một tham chiếu giống như là truy xuất thông qua tên gốc của nó. Tham chiếu nâng cao tính hữu dụng của các con trỏ và sự tiện lợi của việc truy xuất trực tiếp các đối tượng, các câu lệnh bên trong thủ tục có thể thay đổi giá trị của các biến truyền này. Như vậy quá trình gọi một thủ tục được thực hiện trên môi trường đồng nhất, trao đổi thông tin thông qua bộ nhớ, quá trình xử lý hoàn toàn dưới quyền điều khiển của bộ vi xử lý của máy tính cục bộ.

Ý tưởng phương pháp gọi thủ tục từ xa là làm trong suốt quá trình xử lý trên máy tính khác, điều này được thực hiện bằng cách che giấu việc trao đổi thông tin giữa các máy tính. Danh sách tham số và cấu trúc bên trong thủ tục không hề thay đổi, nó chỉ chuyển vị trí thực hiện sang máy tính khác. Phương pháp gọi thủ tục từ xa tạo điều kiện cho việc xây dựng các ứng dụng theo mô hình khách/chủ trỏ nên đơn giản hơn, tiến trình máy khách kết nối với tiến trình máy chủ và sử dụng các dịch vụ của nó.

Gọi thủ tục từ xa xuất hiện lần đầu trên các máy tính của Sun Microsystems và Hewlett-Packard chạy trên nền hệ điều hành UNIX. Gọi thủ tục từ xa hoạt động theo thời gian thực vì chương trình gọi thường đợi đến khi nhận được trả lời từ chương trình được gọi, nghĩa là bên gọi sẽ bị phong tỏa cho đến khi thủ tục từ xa hoàn thành, sử dụng kỹ thuật đa luồng và cơ chế không đồng bộ sẽ khắc phục được nhược điểm này.

Các bước gọi thủ tục chạy trên máy chủ tương tự như gọi thủ tục chạy trên máy cục bộ, tiến trình máy khách chuyển các tham số đầu vào khi gọi thủ tục từ xa và tiến trình máy chủ sẽ kiểm tra tính hợp lệ của các tham số đó, thực hiện tính toán và trả về các giá trị theo yêu cầu của ứng dụng máy khách. Hình thức gọi thủ tục gần như không thay đổi nhưng thời gian xử lý có thể sẽ kéo dài hơn do dữ liệu được lưu chuyển qua mạng, hình 2.8 thể hiện các giai đoạn thực hiện của nguyên lý gọi thủ tục từ xa.

Tìm hiểu sâu hơn bên trong qui trình thực hiện khi gọi thủ tục từ xa, bản chất trao đổi thông tin giữa máy khách và máy chủ vẫn là chuyển thông điệp. Để tiến trình máy khách có thể gọi thủ tục từ xa, một tiến trình trên máy chủ phải cung cấp dịch vụ mô tả trong ngôn ngữ RPC. Mỗi máy chủ được gán tên và số hiệu tiến trình, tất cả các dịch vụ được khai báo trong danh sách với đầy đủ các tham số thể hiện dịch vụ. Với nguyên tắc này, ngôn ngữ RPC cho phép thực hiện các kiểu dữ liệu đơn giản cũng như các dữ liệu phức tạp như: struct, enum...



Hình 2.8 Nguyên lý gọi thủ tục từ xa

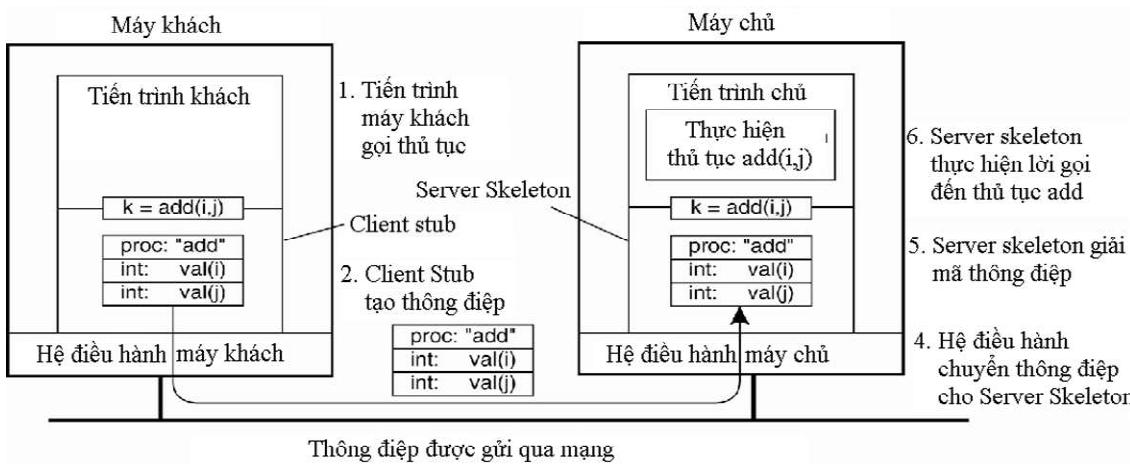
Một thành phần trên máy khách gọi là Stub sẽ mã hóa tên hàm và các tham số vào thông điệp, sau đó sẽ chuyển đến máy chủ. Máy chủ cũng có thành phần tương ứng với Stub gọi là Skeleton, nó giải mã thông điệp và chuyển đến thành phần thực thi như phương pháp gọi thủ tục truyền thống. Kết quả trả về lại được Skeleton mã hóa thành thông điệp để trả về cho máy khách, Stub trên máy khách sẽ giải mã thông điệp và trả về cho chương trình gọi các giá trị hoặc các tham số theo yêu cầu.

Các thành phần của RPC bao gồm tiến trình trên máy khách và máy chủ, tiến trình trên máy khách khởi tạo lời gọi thủ tục từ xa và tiến trình trên máy chủ cung cấp dịch vụ để trả lời yêu cầu của máy khách. Thành phần trung gian được cài đặt trong trường hợp này, máy khách sử dụng Stub để đóng gói các tham số và tên thủ tục thành những thông điệp gửi sang máy chủ, Skeleton trên máy chủ sẽ bóc tách thông tin từ thông điệp máy khách chuyển sang và gọi thủ tục cục bộ trên máy chủ sau đó lại đóng gói kết quả trả về để chuyển cho tiến trình trên máy khách.

Thành phần Stub sẽ giúp cho các máy khách xử lý dễ dàng hơn, tiến trình khách sẽ gọi Stub thay vì phải viết đoạn mã triển khai cho hàm đó, các Stub được biên soạn và liên kết với các ứng dụng máy khách trong quá trình phát triển. Thay vì chứa mã đoạn mã để thực hiện gọi thủ tục từ xa, các đoạn mã của Stub sẽ yêu cầu vấn những tham số từ không gian địa chỉ và sau đó chuyển chúng vào thư viện chạy thực trên máy khách. Tương tự như vậy, thành phần Skeleton trên máy chủ giúp cho việc gọi thủ tục của dịch vụ và trả về kết quả cho tiến trình máy khách được thực hiện dễ dàng hơn.

Phương pháp gọi thủ tục từ xa thể hiện quan điểm tách biệt giữa giao diện và phần cài đặt, xuất phát từ việc khai báo giao diện, phần mềm trung gian tạo các mã lệnh hỗ trợ cho việc xử lý phân tán bằng cách thể hiện các ứng dụng dưới dạng ngữ nghĩa truyền thông trên máy cục bộ, tuy nhiên nó phải đảm bảo nhiều nhiệm vụ phức tạp về truyền thông và đồng bộ. Hình 2.9 thể hiện bản chất bên trong quá trình gọi thủ tục từ xa, quá trình được thực hiện theo mười bước sau:

1. Thủ tục trên máy khách gọi stub như phương pháp gọi thủ tục truyền thống.
2. Stub tạo thông điệp và chuyển đến hệ điều hành của máy khách.
3. Hệ điều hành của máy khách gửi thông điệp đến hệ điều hành của máy chủ.
4. Hệ điều hành của máy chủ chuyển thông điệp đến Skeleton.
5. Skeleton giải mã thông điệp thành các tham số và gọi thủ tục xử lý tương ứng.
6. Máy chủ thực hiện lời gọi thủ tục và trả về giá trị cho Skeleton.
7. Skeleton đóng gói tham số giá trị trả về thành thông điệp và chuyển đến hệ điều hành của máy chủ.
8. Hệ điều hành của máy chủ chuyển thông điệp đến hệ điều hành của máy khách.
9. Hệ điều hành của máy khách nhận thông điệp và chuyển cho Stub.
10. Stub giải mã kết quả và trả về các tham số theo yêu cầu của thủ tục đã gọi.



Hình 2.9 Các bước thực hiện trong gọi thủ tục từ xa

Khác với gọi thủ tục trên một máy tính, gọi thủ tục từ xa phải di chuyển thông tin trên mạng, thời gian vận chuyển dữ liệu lâu hơn và tính tin cậy sẽ thấp hơn so với việc đọc/ghi bộ nhớ. Tổ chức lưu trữ trong bộ nhớ của máy khách và máy chủ có thể khác nhau, không thể trao đổi thông tin trực tiếp qua ô nhớ, đó là những nguyên nhân làm cho vấn đề truyền tham số trở nên phức tạp hơn.

### 2.3.2 Vấn đề truyền tham số

Gọi thủ tục trên máy tính cục bộ cho phép truyền tham số theo giá trị, tham chiếu và con trỏ. Hai phương pháp truyền tham chiếu và con trỏ đều truy nhập đến địa chỉ của ô nhớ nhằm mục đích tăng hiệu năng hoặc thay đổi giá trị của các biến số, con trỏ là biến số lưu trữ địa chỉ của một biến số khác. Nếu sử dụng phương pháp truyền tham chiếu thì các câu lệnh bên trong thủ tục sẽ thao tác trực tiếp trên địa chỉ của các biến số, nếu sử

dụng phương pháp truyền con trỏ thì nó sẽ sao chép địa chỉ của biến số, nghĩa là nó truy nhập vào các biến số một cách gián tiếp qua địa chỉ.

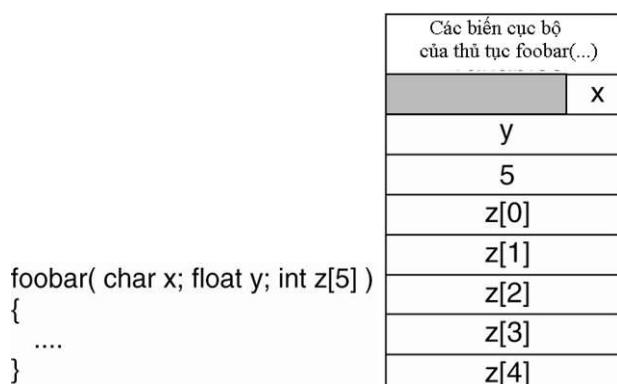
Địa chỉ ô nhớ chỉ có ý nghĩa cục bộ trên một máy tính, cùng một địa chỉ nhưng nếu đọc trên máy tính khác nhau sẽ cho các giá trị hoàn toàn khác nhau. Nếu áp dụng luật truy nhập qua địa chỉ như cách gọi các thủ tục trên một máy tính, địa chỉ của các biến số trên máy khách sẽ trở nên vô nghĩa, tiến trình máy chủ không thể sử dụng những địa chỉ này để thực hiện các thao tác trên biến số.

Gọi thủ tục từ xa vẫn cần phải cho phép thay đổi giá trị của các tham số, nghĩa là vẫn cần phương pháp truyền tham chiếu, yêu cầu này vẫn được đáp ứng nhưng chỉ là giả lập. Ưu điểm về hiệu năng sẽ không còn là lợi thế của phương pháp truyền địa chỉ, dữ liệu của tham số vẫn phải vận chuyển qua mạng. Phương pháp truyền tham chiếu trong gọi thủ tục từ xa được thực hiện bằng cách tạo biến tham chiếu trên Skeleton tương ứng với biến tham chiếu trên Stub, thủ tục trên máy chủ sẽ sử dụng địa chỉ của biến tham chiếu trên Skeleton như phương pháp truyền tham chiếu của một gọi thủ tục trên máy tính cục bộ.

<table border="1" style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td></td><td>3</td><td>2</td><td>1</td><td>0</td></tr> <tr><td>0</td><td></td><td></td><td></td><td></td></tr> <tr><td>7</td><td>6</td><td>5</td><td>4</td><td></td></tr> <tr><td>L</td><td>L</td><td>I</td><td>J</td><td></td></tr> </table>		3	2	1	0	0					7	6	5	4		L	L	I	J		<table border="1" style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>5</td><td></td><td>0</td><td>0</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>J</td><td>I</td><td>L</td><td>L</td></tr> </table>	0	1	2	3	5		0	0	4	5	6	7	J	I	L	L	<table border="1" style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>5</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>L</td><td>L</td><td>I</td><td>J</td></tr> </table>	0	1	2	3	0	0	0	5	4	5	6	7	L	L	I	J
	3	2	1	0																																																		
0																																																						
7	6	5	4																																																			
L	L	I	J																																																			
0	1	2	3																																																			
5		0	0																																																			
4	5	6	7																																																			
J	I	L	L																																																			
0	1	2	3																																																			
0	0	0	5																																																			
4	5	6	7																																																			
L	L	I	J																																																			
Thông điệp ban đầu trên máy Pentium	Thông điệp nhận được trên máy SPARC	Thông điệp sau khi đảo ngược																																																				

Hình 2.10 Truyền tham số trong các hệ thống không đồng nhất

Hệ thống phân tán có thể không đồng nhất, các bộ vi xử lý của Intel theo định dạng **little endian**, trong khi các bộ vi xử lý của Sun theo định dạng **big endian**, qui định lưu trữ các byte dữ liệu trong hai hệ thống này ngược nhau. Hình 2.10 minh họa 2 tham số (5, “LLIJ”) của thủ tục từ xa, các byte dữ liệu lần lượt được chuyển từ máy tính của hãng Intel sang máy tính của hãng Sun, khi đến máy Sun giá trị của các tham số theo thứ tự ngược lại và máy Sun sẽ hiểu số 5 là 0x5000. Tương tự như vậy, xâu ký tự LLIJ sẽ được máy Sun hiểu là JILL, do đó chỉ cần đảo ngược thứ tự các byte sẽ nhận được kết quả ban đầu.



Hình 2.11 Mã hóa tham số vào thông điệp

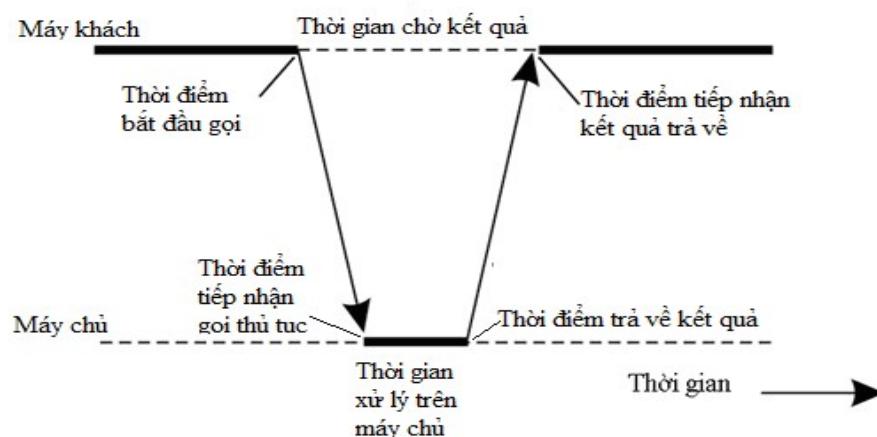
Phương pháp gọi thủ tục từ xa che giấu quá trình trao đổi thông tin trên mạng bằng cách đưa mã hóa các tham số vào thông điệp, cả máy khách và máy chủ đều phải tuân thủ quy tắc về định dạng tham số. Hình 2.11 minh họa Stub trên máy khách mã hóa các tham số để tạo thành thông điệp gửi sang máy chủ, thông điệp chứa giá trị của các biến x, y và mảng z. Ở chiều ngược lại, thành phần Skeleton trên máy chủ sẽ phải giải mã thông điệp để nhận được giá trị của các tham số, nó phải nắm được luật mã hóa mà Stub.

Để đơn giản hóa quá trình tạo Stub cho máy khách và Skeleton cho máy chủ, một ngôn ngữ mới được sử dụng gọi là ngôn ngữ định nghĩa giao diện IDL. Lập trình viên chỉ việc viết các hàm và các thủ tục theo qui định của ngôn ngữ, sau đó dùng chương trình dịch để chuyển sang ngôn ngữ lập trình tương ứng. Chương trình dịch IDL sẽ sinh ra hai loại tập tin, loại tập tin Stub dùng để tích hợp vào phần mềm của máy khách, loại tập tin Skeleton sẽ được tích hợp vào phần mềm của máy chủ.

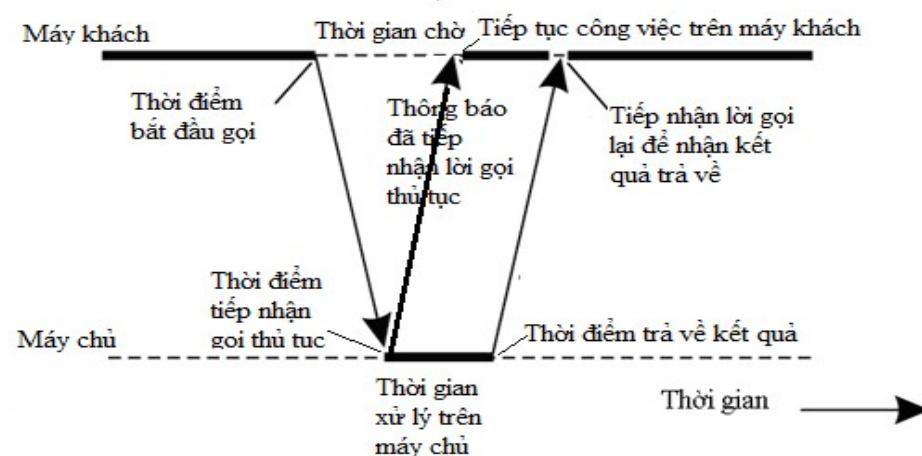
### 2.3.3 Các phương pháp gọi thủ tục từ xa

Tương tự như chuyển thông điệp dựa trên dịch vụ của tầng vận tải, gọi thủ tục từ xa có thể thực hiện bằng phương pháp đồng bộ hoặc không đồng bộ. Hình 2.12 thể hiện các giai đoạn gọi thủ tục từ xa, thời gian thực hiện gọi thủ tục từ xa là tổng của thời gian gửi yêu cầu, thời gian thực hiện trên máy chủ và thời gian vận chuyển kết quả trả về.

#### a. Phương pháp đồng bộ



#### b. Phương pháp không đồng bộ



Hình 2.12 Hai phương pháp gọi thủ tục từ xa

Nếu sử dụng phương pháp đồng bộ, tiến trình trên máy khách sẽ bị phong tỏa cho đến khi nhận được kết quả trả về, việc chờ đợi này đồng nghĩa với việc lãng phí năng lực xử lý của máy khách và khả năng vận chuyển dữ liệu của kênh truyền. Trong thời gian xử lý trên máy chủ, đáng lẽ tiến trình máy khách có thể thực hiện công việc khác, không có dữ liệu lưu chuyển trên kênh truyền vật lý dẫn tới hiệu suất sử dụng băng thông thấp. Hơn nữa, nếu thời gian xử lý trên máy chủ lớn có thể dẫn đến hiện tượng quá thời gian, khi đó sẽ phải thực hiện các thủ tục xử lý lỗi vốn đã rất phức tạp.

Nếu áp dụng phương pháp không đồng bộ, sau khi tiến trình máy khách gửi yêu cầu, chỉ cần nhận được phản hồi thành công từ máy chủ, tiến trình trên máy khách có thể tiếp tục xử lý các tác vụ khác. Như vậy thời gian tiến trình máy khách bị phong tỏa giảm đi đáng kể, nó chỉ còn phụ thuộc vào thời gian vận chuyển dữ liệu các tham số của thủ tục, dữ liệu xác nhận rất nhỏ nên thời gian vận chuyển không đáng kể.

Sau khi nhận được yêu cầu gọi thủ tục từ xa, tiến trình máy chủ sẽ thực hiện xử lý, khi hoàn thành sẽ chuyển kết quả thông qua cơ chế gọi lại. Hoạt động của máy khách và máy chủ hoàn toàn độc lập, máy khách có thể gửi liên tục nhiều yêu cầu đến máy chủ, hiệu quả sử dụng kênh truyền vật lý sẽ cao hơn. Nếu sử dụng phương pháp đồng bộ để gọi thủ tục từ xa thì tiến trình trên máy khách sẽ bị phong tỏa, hơn nữa tiến trình trên máy khách đều phải chuyển toàn bộ các tham số sang tiến trình máy chủ, nếu lượng dữ liệu lớn hoặc tốc độ kênh truyền thấp thì thời gian vận chuyển quá lâu sẽ phát sinh lỗi quá thời gian.

Như vậy, việc gọi thủ tục từ xa theo phương thức đồng bộ sẽ chỉ phù hợp cho các tác vụ với lượng dữ liệu nhỏ và thời gian thực hiện nhanh. Nếu lượng dữ liệu nhỏ nhưng thời gian xử lý quá lâu thì có thể áp dụng phương pháp không đồng bộ, phương pháp này cũng thường được áp dụng để nâng cao hiệu năng hệ thống. Cơ chế xử lý song song được triển khai trên cả máy khách lẫn máy chủ, hiệu năng hệ thống sẽ tăng lên đáng kể và đây là phương pháp được áp dụng khá phổ biến trong các hệ thống phân tán. Sau năm 1990, công nghệ phần mềm chuyển sang hướng đối tượng thay cho hướng thủ tục, gọi thủ tục từ xa cũng phát triển thành gọi đối tượng phân tán.

## 2.4 Truyền thông luồng

Tuần thông bằng cách gọi thủ tục từ xa hay chuyển thông điệp mới chỉ chú trọng đến việc phân phát dữ liệu mà chưa đề cập đến loại dữ liệu cần chuyển, thời gian thực hiện, tính liên tục và các yêu cầu về tính đồng bộ. Gọi một thủ tục từ xa hoặc chuyển thông điệp không phụ thuộc vào thời điểm thực hiện, dù thời gian thực hiện có thể nhanh hay chậm cũng không ảnh hưởng đến tính chính xác của dữ liệu.

Những loại dữ liệu mà tính chính xác của nó phụ thuộc vào thời gian thì phải được vận chuyển với độ trễ thấp nhất có thể nhưng vẫn phải đáp ứng các tiêu chuẩn về chất lượng dịch vụ. Ví dụ truyền âm thanh, tai người có thể nhận biết được âm thanh tần số từ 30 đến 20000 Hz, theo định luật lấy mẫu của Nyquist thì tần số lấy mẫu phải là 44100 Hz mới đảm bảo chất lượng, đó cũng là tiêu chuẩn CD 16 bit/44100 Hz. Bên gửi ghi lại âm thanh và chuyển qua mạng, bên nhận tổng hợp lại luồng bit để khôi phục âm thanh thì cũng phải đảm bảo tần số trên, rất tiếc tốc độ truyền tin trên mạng thường không ổn định,

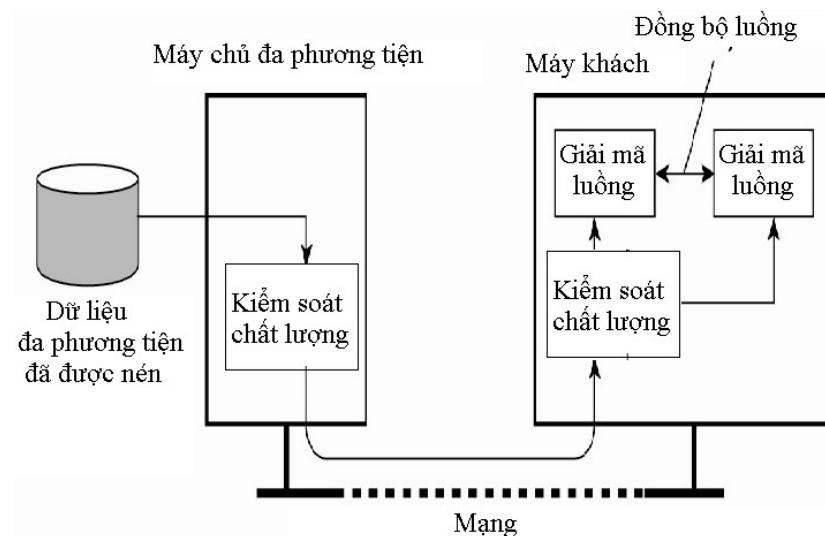
do đó có thể dẫn tới hiện tượng méo âm. Dịch vụ đa phương tiện chứa đựng nhiều luồng dữ liệu khác nhau như văn bản, âm thanh, hình ảnh..., chúng đòi hỏi tính đồng bộ giữa các luồng dữ liệu và tính liên tục cũng là thước đo của chất lượng dịch vụ.

#### 2.4.1 Hỗ trợ truyền thông liên tục

Mỗi loại thông tin có những tiêu chuẩn mã hóa riêng, văn bản thường được mã hóa theo chuẩn ASCII hoặc Unicode, chuẩn JPEG thường dùng để mã hóa hình ảnh, trong viễn thông chuẩn mã hóa âm thanh là G711 nhưng cũng có thể sử dụng nhiều chuẩn khác để chất lượng tốt hơn, mã hóa video thường sử dụng chuẩn H.265 hoặc MPEG-4. Văn bản và ảnh tĩnh không những đòi hỏi độ chính xác tuyệt đối mà còn phải đảm bảo thứ tự hiển thị của chúng, các yêu cầu về tính liên tục thường ít được quan tâm hơn. Âm thanh không những đòi hỏi tính chính xác và thứ tự thể hiện mà còn phải đảm bảo các yêu cầu khắt khe về tính liên tục, video cần phải đáp ứng tất cả các yêu cầu như âm thanh nhưng cộng thêm yêu cầu về tính đồng bộ.

Để vận chuyển dữ liệu của những thông tin mà tính đúng đắn của nó phụ thuộc thời gian, hệ thống phân tán sử dụng khái niệm luồng dữ liệu. Về bản chất, luồng dữ liệu là chuỗi các đơn vị dữ liệu rời rạc, trong khi đó việc thể hiện loại dữ liệu này cần phải đảm bảo tính liên tục. Luồng dữ liệu có thể là đơn hoặc phức hợp, nếu nó chỉ chứa một loại thông tin thì đó là luồng đơn, nếu luồng dữ liệu được cấu thành từ nhiều luồng con và mỗi luồng con thể hiện một loại thông tin thì gọi là luồng phức hợp.

Video bao gồm một luồng hình ảnh và hai luồng âm thanh, nhưng nếu có phụ đề thì cần luồng thứ tư là văn bản, những thông tin được cấu thành từ nhiều luồng dữ liệu khác nhau gọi là đa phương tiện. Như vậy luồng dữ liệu phức hợp không những phải đảm bảo tính liên tục trong mỗi luồng con mà còn cần phải duy trì tính đồng bộ giữa các luồng con với nhau. Thời gian là nhân tố đóng vai trò quan trọng trong luồng dữ liệu liên tục, có thể đảm bảo về thứ tự của các đơn vị dữ liệu nhưng không có gì đảm bảo tính ổn định của độ trễ truyền thông, ngay cả khi áp dụng phương pháp truyền đồng bộ.



Hình 2.13 Kiến trúc hệ thống truyền thông đa phương tiện

Hình 2.13 minh họa trường hợp đơn giản, các loại thông tin đều được đưa vào một luồng phức hợp và lưu trữ trong tập tin, cơ chế đồng bộ đã được giải quyết trong tiêu chuẩn mã hóa, chỉ cần giải quyết các yêu cầu về tính liên tục. Nhận được yêu cầu từ máy khách, tiến trình máy chủ chỉ cần đọc tập tin và chuyển dữ liệu qua mạng, tiến trình máy khách chỉ cần giải mã để được các luồng dữ liệu con và chuyển chúng đến các thành phần nghe nhìn tương ứng. Độ trễ thời gian cho việc đọc tập tin trên máy chủ và giải mã luồng dữ liệu trên máy khách tương đối ổn định, dữ liệu đã được tích hợp trong luồng phức nên không chúng đã được đồng bộ, yêu cầu duy nhất cần giải quyết trong trường hợp này là tính liên tục.

Dịch vụ hội thảo trực tuyến phức tạp hơn rất nhiều, dữ liệu được thu thập trực tiếp từ thiết bị ngoại vi, sẽ không hợp lý nếu trộn các luồng dữ liệu vào một luồng phức hợp. Giải quyết vấn đề đồng bộ giữa các luồng vốn đã phức tạp nhưng đáp ứng yêu cầu về tính liên tục kèm theo yếu tố thời gian thực còn phức tạp hơn, thời gian thu thập và xử lý thông tin từ thiết bị ngoại vi thường không ổn định, dung lượng dữ liệu rất lớn cũng là một trở ngại trong việc vận chuyển qua mạng.

#### 2.4.2 *Luồng và chất lượng dịch vụ*

Các dịch vụ phụ thuộc thời gian thường đòi hỏi những yêu cầu về chất lượng dịch vụ của hạ tầng truyền thông, đó là những tiêu chuẩn tối thiểu để đảm bảo quan hệ giữa các mục dữ liệu. Tiêu chuẩn thứ nhất là thông lượng, đó là lượng dữ liệu thực tế được chuyển đi trong một đơn vị thời gian, nó không những bị hạn chế bởi băng thông mà còn bị ảnh hưởng nhiều yếu tố khác như số lượng người sử dụng, các tác nhân môi trường... Tiêu chuẩn thứ hai là độ trễ tối đa để thiết lập liên kết, đó là thời gian kể từ khi đưa ra yêu cầu cho đến khi hoàn thành thiết lập phiên làm việc, nghĩa là một ứng dụng có thể bắt đầu gửi dữ liệu. Tiêu chuẩn thứ ba là độ trễ tối đa từ đầu cuối đến đầu cuối, nó thể hiện thời gian vận chuyển một đơn vị dữ liệu giữa các tiến trình trong hệ thống phân tán. Tiêu chuẩn thứ tư là biên độ tối đa thay đổi độ trễ, đó là phương sai độ trễ lớn nhất, nó thể hiện tính ổn định của kênh truyền. Tiêu chuẩn thứ năm là độ trễ toàn phần, nó thể hiện giá trị trễ lớn nhất kể từ khi gửi đơn vị dữ liệu cho đến khi nhận được bản tin phản hồi.

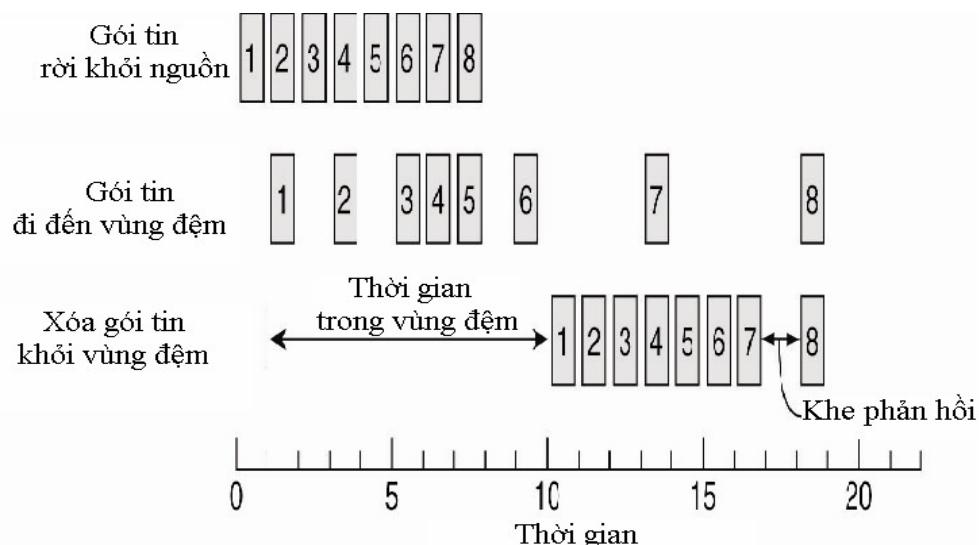
Các tiêu chuẩn trên không phải là bất biến, hệ thống phân tán hiện nay hầu hết đều được xây dựng dựa trên nền tảng của mô hình TCP/IP, thiết bị trung gian mạng có thể loại bỏ bất kỳ gói tin nào nếu không thể vận chuyển được đến thiết bị kế tiếp. Bản thân mô hình TCP/IP cũng đã qui định một loạt các tiêu chuẩn về chất lượng dịch vụ, tuy nhiên thực tế chúng ít khi được triển khai.

Các dịch vụ trực tuyến không những yêu cầu về tính liên tục mà còn phải đảm bảo yêu cầu thời gian thực, nghĩa là độ trễ nằm trong ngưỡng chấp nhận được. Âm thanh và hình ảnh của video được lấy mẫu theo tần suất rất cao, chu kỳ lấy mẫu âm thanh khoảng 20 mili giây và của hình ảnh video khoảng 40 mili giây, điều đó cho thấy dung lượng dữ liệu của chúng rất lớn.

Phương pháp truyền đồng bộ qui định thời gian trễ tối đa cho mỗi đơn vị dữ liệu trong luồng, thời gian truyền thực tế có thể nhỏ hơn ngưỡng cho phép nhưng không được phép lớn hơn khoảng thời gian giữa hai lần lấy mẫu. Hệ thống phân tán sử dụng phương

pháp truyền đẳng thời để vận chuyển dữ liệu loại âm thanh hoặc video, các đơn vị dữ liệu phải được chuyển đúng thời gian, nghĩa là phải đảm bảo ngưỡng thời gian tối đa và tối thiểu từ đầu cuối đến đầu cuối.

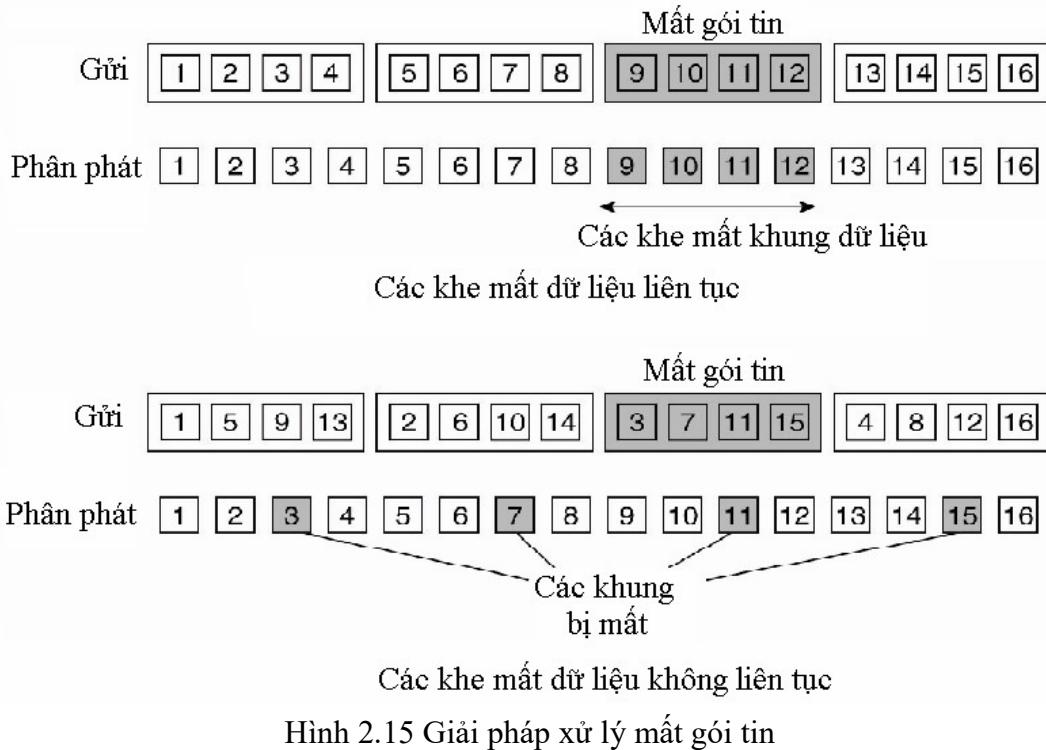
Chất lượng kém của hạ tầng mạng thường thể hiện ở độ trễ không ổn định và hiện tượng mất gói tin, do đó các ứng dụng phân tán phải khắc phục cả hai vấn đề này. Giải quyết vấn đề trễ bằng cách thiết lập giá trị ưu tiên trong gói tin IP, khi đó các gói tin sẽ được thiết bị định tuyến xử lý với độ trễ thấp nhất và độ tin cậy cao nhất. Một giải pháp khác là sử dụng bộ đệm để loại bỏ sự không ổn định của kênh truyền, dữ liệu nhận được sẽ đưa vào vùng đệm, kích thước của nó được tính toán sao cho đủ khả năng đảm bảo thể hiện thông tin liên tục nhưng thời gian trễ nhỏ nhất.



Hình 2.14 Sử dụng bộ đệm trong truyền thông đa phương tiện

Hình 2.14 minh họa nguyên lý sử dụng vùng đệm trong truyền thông, giả sử mỗi gói tin đại diện cho một đơn vị dữ liệu luồng ở tầng ứng dụng. Bên gửi lần lượt chuyển 8 gói tin lên mạng, nhận được gói tin số 6, bên nhận tính toán chỉ cần đặt kích thước vùng đệm đủ chứa 7 gói tin, như vậy nó bắt đầu hiển thị thông tin. Thật không may, khi đã hiển thị gói tin số 7 nhưng gói tin số 8 vẫn chưa đến, điều này sẽ làm gián đoạn thể hiện dữ liệu, giải pháp duy nhất ở đây là tăng kích thước vùng đệm.

Nếu một gói tin bị hủy bỏ, bên nhận có thể yêu cầu gửi lại, nhưng điều này không nằm trong cơ chế hoạt động của giao thức IP. Dữ liệu xuất phát từ tầng ứng dụng thường có kích thước lớn, nó thường được chia thành những đoạn tin trước khi chuyển đến tầng Internet. Giá trị mặc định của chiều dài gói tin trong hầu hết các hệ điều hành là 4096 byte, nếu kích thước đoạn tin lớn hơn chiều dài gói tin hệ điều hành qui định thì nó sẽ được phân mảnh, chỉ cần một gói tin thất lạc sẽ phải gửi lại toàn bộ đoạn tin. Giải pháp tối ưu sẽ là chia nhỏ dữ liệu ngay tại tầng ứng dụng sao cho kích thước mỗi đoạn tin không vượt quá 4076 byte, thông tin điều khiển của giao thức IP chiếm 20 byte.



Hình 2.15 Giải pháp xử lý mất gói tin

Cơ chế gửi lại không phù hợp với các dịch vụ trực tuyến, nó không đảm bảo yêu cầu thời gian thực, do đó cần phải áp dụng kỹ thuật sửa lỗi chuyển tiếp. Bên gửi sẽ mã hóa các gói dữ liệu để bên nhận có thể tái tạo lại K gói tin bị mất trên tổng số N gói tin nhận được. Đơn vị dữ liệu âm thanh thường nhỏ hơn 4076 byte, do đó chỉ cần 1 gói tin để vận chuyển, vì vậy không cần áp dụng kỹ thuật này. Mỗi ảnh thường phải chia thành nhiều khung hình nhỏ, mỗi khung hình tương ứng cho một gói tin, nếu gói tin bị thất lạc sẽ để lộ một khoảng trống khá lớn. Hình 2.15 thể sử dụng kỹ thuật trộn bằng cách đan xen các khung hình, các khung hình bị mất sẽ không liên tục, điều đó tạo điều kiện cho giải thuật sửa lỗi có thể phục hồi những khung hình đã thất lạc.

#### 2.4.3 Đồng bộ luồng

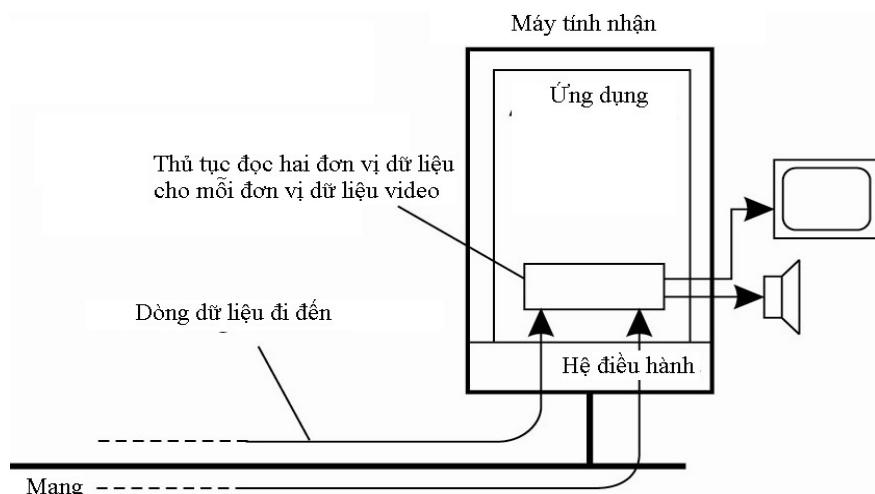
Đồng bộ luồng nhằm mục đích duy trì mối quan hệ nhịp độ giữa những thẻ thiện của các luồng dữ liệu khác nhau, dạng đơn giản nhất là đồng bộ luồng dữ liệu rời rạc với luồng dữ liệu liên tục, phức tạp hơn là đồng bộ dữ liệu của các luồng liên tục. Ví dụ hoạt động giảng dạy trực tuyến, hình ảnh màn hình máy tính thuộc loại dữ liệu rời rạc trong khi đó âm thanh và hình ảnh giảng viên là thuộc loại luồng dữ liệu liên tục.

Việc đồng bộ được thực hiện ở mức độ đơn vị dữ liệu của luồng, kích thước đơn vị dữ liệu phụ thuộc vào mức độ trừu tượng hóa để luồng dữ liệu thể hiện thông tin. Vấn đề cần bàn là quá trình đồng bộ nên đặt ở bên gửi hay bên nhận, nếu bên gửi thực hiện thì có thể hợp nhất các luồng con thành luồng duy nhất, mỗi đơn vị dữ liệu bao gồm các đơn vị dữ liệu của các luồng con. Nếu bên nhận thực hiện đồng bộ thì đơn vị dữ liệu của luồng con được vận chuyển độc lập trên mạng, độ trễ truyền thông khác nhau nên việc đồng bộ sẽ khó khăn hơn.

Ví dụ âm thanh mã hóa 16 bit trên một kênh và tần số lấy mẫu 44100 Hz, về mặt lý thuyết thì phải đồng bộ với các luồng dữ liệu khác theo chu kỳ lấy mẫu 22.6757369615 µs, đối với các hiệu ứng âm thanh nổi chất lượng cao thì việc đồng bộ hóa ở cấp độ này là thực sự cần thiết. Chu kỳ nhỏ như vậy thì không thể triển khai trên mạng, thực tế luồng dữ liệu âm thanh sẽ được chuyển vào vùng đệm tạo thành các đơn vị dữ liệu để chuyển sang máy tính khác. Gọi  $B_s$  là số lượng bit thể hiện mẫu âm thanh, số lượng kênh âm thanh  $C_h$ , tần số lấy mẫu  $F_s$  và khoảng thời gian đồng bộ  $T_s$  tính bằng mili giây. Như vậy để biểu diễn một mẫu sẽ cần  $(B_s * C_h)/8$  bytes và kích thước của đơn vị dữ liệu âm thanh  $S_{au}$  được tính theo công thức sau:

$$S_{au} = \frac{B_s * C_h * F_s * T_s}{8000} \text{ (byte)}$$

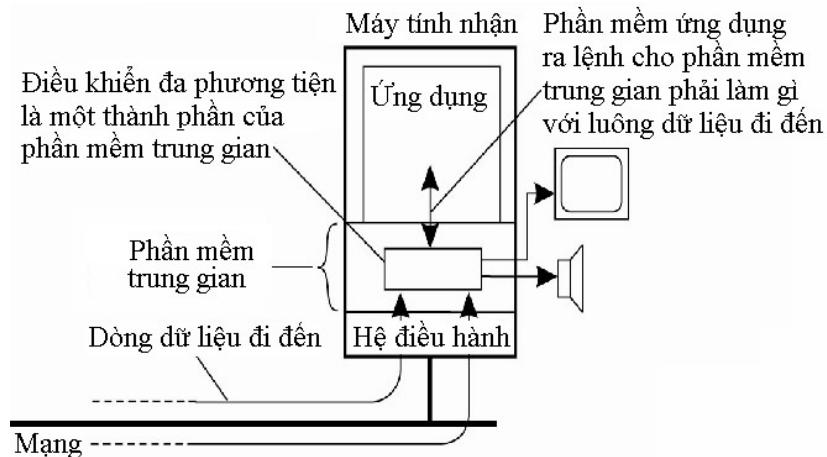
Tương tự như âm thanh, hình ảnh cũng phải liên tục, điều này khó có thể triển khai trên một máy tính chứ chưa cần nói đến việc phải chuyển dữ liệu qua mạng. Để có cảm giác liên tục, các khung hình phải được hiển thị với tần số tối thiểu 25 Hz, chuẩn NTSC với tần số 29.97 Hz thì chu kỳ lấy mẫu là 33.3667 mili giây. Nếu tần số lấy mẫu âm thanh 44100 Hz thì mỗi đơn vị dữ liệu âm thanh chứa 1471 mẫu, giả thiết sử dụng một kênh âm thanh và thì kích thước đơn vị dữ liệu âm thanh là 2942 byte đối với âm thanh đơn hoặc 5884 byte cho âm thanh nối.



Hình 2.16 Nguyên lý đồng bộ mức đơn vị dữ liệu

Cơ chế đồng bộ có thể xem ở một số mức độ trừu tượng khác nhau, ở mức thấp nhất đồng bộ hóa được thực hiện trên các đơn vị dữ liệu của các luồng con. Về nguyên tắc để đảm bảo tuân thủ ràng buộc về nhịp độ, hình 2.16 minh họa trường hợp dữ liệu video với âm thanh nổi, cứ mỗi luồng hình ảnh thì phải đọc hai luồng âm thanh. Giả sử ảnh 320x240 điểm, mỗi điểm ảnh được mã hóa bằng 3 bytes, như vậy đơn vị dữ liệu của ảnh là 230400 bytes, nếu sử dụng âm đơn sắc thì đơn vị dữ liệu luồng âm thanh là 2942 byte, cứ sau  $33366.7000333667 \mu s$  phải một đọc đơn vị dữ liệu luồng âm thanh và và một khung hình, bằng thông phải trên 56 Mbps mới có thể đảm bảo âm thanh và hình ảnh liên tục.

Hầu hết các hệ điều hành thiết lập giá trị mặc định cho chiều dài gói tin là 4 Kb, chỉ cần 1 gói tin vận chuyển âm thanh nhưng cần tới 17 gói tin vận chuyển hình ảnh. Điều đó giải thích vì sao không nên trộn hai luồng dữ liệu trên, nếu một gói tin bị thất lạc có thể dẫn đến hiện tượng méo âm vốn rất dễ nhận biết. Dữ liệu ảnh chiếm tỉ trọng lớn trong dung lượng tất cả các luồng, nếu băng thông thấp hoặc chất lượng kênh truyền kém thì có thể giảm tần suất lấy mẫu hình ảnh, chất lượng hình ảnh có thể chấp nhận được nếu khoảng thời gian đồng bộ xê dịch từ 40 đến 80 mili giây.



Hình 2.17 Nguyên lý đồng bộ mức giao diện

Cách tiếp cận này cho thấy ứng dụng hoàn toàn chịu trách nhiệm thực hiện đồng bộ trong khi nó chỉ có những tiện ích cấp thấp, cách tiếp cận tốt hơn là cung cấp cho ứng dụng một giao diện để có thể dễ dàng kiểm soát các luồng, ví dụ có thể cho phép điều khiển tốc độ hiển thị hình ảnh và âm thanh. Với giao diện điều khiển này có thể viết chương trình gồm hai trình xử lý phục vụ cho mỗi luồng và cùng kiểm tra tính đồng bộ giữa các luồng, nếu cần thì sẽ điều chỉnh tốc độ của các đơn vị hình ảnh hoặc âm thanh.

Hình 2.17 minh họa giải pháp cài đặt thêm phần mềm trung gian, nó cung cấp các giao diện để điều khiển luồng âm thanh và hình ảnh, mỗi thiết bị có giao diện riêng, bao gồm cả giao diện thông báo lỗi, như vậy phần mềm ứng dụng có thể viết các thủ tục để đồng bộ các luồng dữ liệu. Phân bổ cơ chế đồng bộ lại là một vấn đề khác cần xem xét, nhận được luồng dữ liệu phức hợp thì phải biết các đặc tả đồng bộ, thực tế thông tin này được gắn ngầm định khi ghép các luồng con vào đơn vị dữ liệu của luồng phức hợp.

Tiêu chuẩn MPEG qui định cách nén hình ảnh và âm thanh, MPEG-2 được thiết kế để truyền dữ liệu với tốc độ 4 đến 6 Mbit/s, nó có thể hợp nhất các luồng dữ liệu liên tục và các luồng dữ liệu rời rạc để tạo thành một luồng duy nhất, sau đó được chuyển thành các gói tin được gán nhãn thời gian dựa trên đồng hồ hệ thống tần số 90 KHz. Bên nhận sẽ tách luồng, nhãn thời gian của mỗi gói tin được sử dụng như một cơ chế cơ bản để đồng bộ các luồng.

## 2.5 Truyền thông theo nhóm

Truyền thông theo nhóm là phương pháp truyền dữ liệu từ một thành viên đến nhiều thành viên khác trong cùng một nhóm. Truyền thông theo nhóm được sử dụng

trong rất nhiều lĩnh vực của đời sống xã hội như phát thanh, truyền hình, hội thảo trực tuyến.... Vấn đề chính của giải pháp thực hiện yêu cầu này nằm ở cách thiết lập kênh truyền thông để phổ biến thông tin sao cho thời gian lan tỏa đến các thành viên nhanh nhất và lượng thông điệp lưu chuyển ít nhất.

Giải pháp thứ nhất dựa trên nguyên lý mạng phủ, bên gửi phải biết địa chỉ của tất cả các thành viên trong nhóm để thiết lập kênh truyền. Giải pháp thứ hai dựa trên nguyên lý lan truyền thông tin, mỗi thành viên không cần phải biết địa chỉ của tất cả các thành viên trong nhóm. Cả hai giải pháp đều được triển khai tại tầng ứng dụng, nói đúng hơn nên cài đặt ở tầng trung gian truyền thông, mỗi giải pháp đều có những ưu điểm và nhược điểm riêng.

### **2.5.1 Truyền thông dựa trên mạng phủ**

Ý tưởng cơ bản là xây dựng mạng phủ bao trùm tất cả các thành viên trong nhóm, nó được xây dựng ở tầng ứng dụng cho nên sẽ không có sự tham gia của các thiết bị định tuyến vốn chỉ hoạt động ở tầng mạng. Liên kết giữa các thành viên trong mạng phủ có thể đi qua nhiều kênh truyền vật lý, do đó việc định tuyến các thông điệp có thể sẽ không tối ưu như cơ chế định tuyến của tầng mạng.

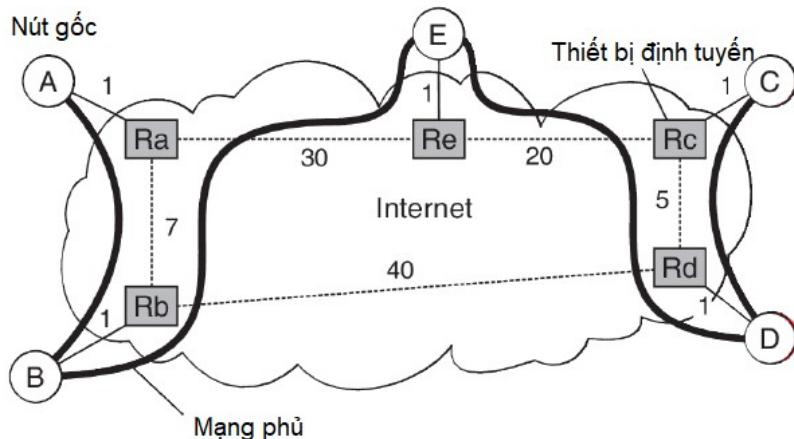
Vấn đề quan trọng là thiết kế mạng phủ, cách tiếp cận thứ nhất là tổ chức các nút theo kiến trúc hình cây, như vậy sẽ đảm bảo giữa hai nút bất kỳ sẽ chỉ có một đường đi duy nhất. Cách tiếp cận thứ hai là các thành viên được tổ chức thành một mạng lưới, mỗi thành viên sẽ có tập láng giềng, như vậy tồn tại nhiều đường đi giữa các thành viên, nếu một đường nào đó bị đứt hoặc quá tải thì có thể tìm đường đi khác mà không cần phải tổ chức lại mạng phủ, như vậy khả năng chịu lỗi sẽ cao hơn.

#### **2.5.1.1 Tổ chức hình cây**

Các thành viên được tổ chức phân cấp, việc đầu tiên phải lựa chọn thành viên đóng vai trò gốc của cây, cách đơn giản có thể sử dụng giải thuật bầu chọn, phức tạp hơn sử dụng giải thuật Chord. Giải thuật Chord để xây dựng cây cho một nhóm các thành viên bằng cách tạo mã định danh theo nhóm, giả sử  $M\_Id$  là khóa ngẫu nhiên 160 bit, sử dụng hàm  $succ(M\_Id)$  để tìm kiếm thành viên chịu trách nhiệm đối với khóa này và đề nghị thành viên đó là gốc của cây để gửi dữ liệu đến các thành viên của nhóm.

Nếu một thành viên muốn tham gia, nó chỉ cần thực hiện thao tác  $lookup(M\_Id)$  để gửi thông điệp tìm kiếm đến thành viên gốc  $succ(M\_Id)$ . Trên đường đến thành viên gốc, yêu cầu này có thể sẽ phải chuyển tiếp qua một số thành viên, mỗi thành viên chuyển tiếp sẽ được ghi nhận là cha của thành viên gửi yêu cầu. Nếu một thành viên đã là cha của thành viên gửi thì nó không cần phải chuyển tiếp yêu cầu đến thành viên cấp cao hơn vì nó đã được ghi nhận là con của thành này.

Tất cả các thành viên trong nhóm đã được tổ chức thành mạng phủ theo kiến trúc phân cấp, thành viên cha sẽ chuyển thông điệp cho các thành viên con, quá trình cứ tiếp tục như vậy trên tất cả các nhánh của cây. Việc xây dựng cây không quá khó, tuy nhiên nó chưa đề cập đến vấn đề hiệu năng, việc phân phát thông điệp hoàn toàn dựa trên cách định tuyến của mạng phủ chứ không phải cách thực hiện trên các thiết bị định tuyến.



Hình 2.18 Quan hệ giữa liên kết của mạng phủ và mạng vật lý

Hình 2.18 minh họa nhóm gồm 5 thành viên, chúng được kết nối mạng qua các kênh truyền vật lý được thể hiện bằng mạng lưới các thiết bị định tuyến (Ra, Rb, Rc, Rd, Re), các con số thể hiện chi phí vận chuyển dữ liệu. Theo một cách nào đó mạng phủ của nhóm được xây dựng theo dạng cây ABEDC với thành viên A là gốc, đường đậm nét thể hiện cấu trúc của cây. Nếu thành viên A muốn chuyển thông điệp cho các thành viên trong nhóm, đầu tiên nó sẽ chuyển đến B sau đó B chuyển đến E, E chuyển đến D và cuối cùng D chuyển đến C.

Dễ dàng nhận thấy sự lãng phí trong cách định tuyến của mạng phủ vì dữ liệu được chuyển hai lần qua các kênh vật lý giữa các thiết bị định tuyến Ra và Rb, Rc và Rd, chi phí truyền dữ liệu từ B qua E và D đến C là 73, nếu B chuyển trực tiếp qua kênh Rb và Rd đến Rc thì chi phí chỉ còn 47. Nguyên nhân là do cách định tuyến của mạng phủ không khớp với hình trạng vật lý của mạng, điều đó dẫn tới hiện tượng thông điệp sẽ phải chuyển nhiều lần qua một liên kết vật lý. Chất lượng của cây truyền thông nhóm thường được đo bằng các tham số ứng suất liên kết, độ trễ tương đối và chi phí cây.

Ứng suất liên kết là số lần một gói tin phải chuyển qua cùng một liên kết vật lý, giá trị của nó có thể lớn hơn 1 xuất phát từ thực tế gói tin có thể được vận chuyển trên hai liên kết logic khác nhau nhưng ở mức vật lý gói tin đã bị lặp trên một đoạn liên kết giữa các thiết bị kề cạnh nhau. Độ trễ tương đối đo bằng tỉ số giữa độ trễ giữa hai thành viên trong mạng phủ với độ trễ trong mạng vật lý, nếu coi chi phí trên hình vẽ là độ trễ thì tỉ lệ này là 1.553191, như vậy mục tiêu cần hướng đến là giảm thiểu độ trễ tương đối trung bình giữa tất cả các cặp thành viên trong mạng phủ.

Chi phí cây là thước đo chung, nó liên quan đến việc giảm thiểu tổng chi phí liên kết, nếu coi chi phí của một liên kết là độ trễ giữa hai thành viên cuối thì việc tối ưu hóa chi phí cây sẽ giảm xuống để tìm một cây phủ tất cả các thành viên của nhóm nhưng với tổng thời gian để lan truyền thông tin đến tất cả các thành viên là nhỏ nhất. Để đơn giản hóa vấn đề, một thành viên được giao nhiệm vụ theo dõi các thành viên trong nhóm, nếu thành viên mới muốn tham gia sẽ gửi yêu cầu đến thành viên này để lấy danh sách các thành viên, từ đó sẽ chọn ra thành viên tốt nhất với tư cách là cha của thành viên mới trong cây.

Xét một nhóm có một nguồn duy nhất, như vậy thành viên cha tốt nhất hiển nhiên là nguồn vì độ trễ tương đối bằng 1, kết quả là mạng phủ có dạng hình sao. Mặc dù cách xây dựng cây khá đơn giản nhưng dễ dàng nhận thấy thành viên gốc sẽ trở nên quá tải, việc lựa chọn một thành viên làm cha nên được hạn chế bởi số lượng các thành viên con, tuy nhiên điều này lại phức tạp hóa giải thuật thiết lập cây.

Giải pháp khác là xây dựng cây dựa trên nguyên lý giao thức STP đã được cài đặt trên các thiết bị chuyển mạch. Ý tưởng của giải thuật khá đơn giản, một nút P có thể chuyển cha bằng cách hủy bỏ liên kết với cha hiện hành và thay thế bằng liên kết đến nút khác với điều kiện cây mẹ không được phép là thành viên bắt nguồn từ nút P và cha của cây mới không có quá nhiều con nhầm tránh hiện tượng quá tải.

Có nhiều tiêu chí khác nhau để quyết định chuyển đổi nút cha, cách đơn giản là tối ưu hóa đường đi đến nguồn, giảm thiểu độ trễ khi thông điệp được truyền theo nhóm. Để thực hiện điều này thì mỗi nút phải thường xuyên nhận được thông tin về trạng thái của các nút khác, mỗi nút sẽ tự đánh giá, nếu tuyến mới tốt hơn thì chuyển đổi cha. Một tiêu chí khác là lấy độ trễ đến nút cha làm thước đo để so sánh, nếu tất cả các nút đều làm như vậy thì độ trễ tổng thể của cây trong trường hợp lý tưởng sẽ là nhỏ nhất. Cụ thể, mỗi nút sẽ lấy tập các nút láng giềng của cha, nghĩa là nút ông và các nút con khác của cha sau đó tính toán độ trễ của từng nút và chọn nút có độ trễ thấp nhất, khi đang thực hiện tìm cha thì sẽ từ chối các yêu cầu gửi đến để tránh hiện tượng lặp.

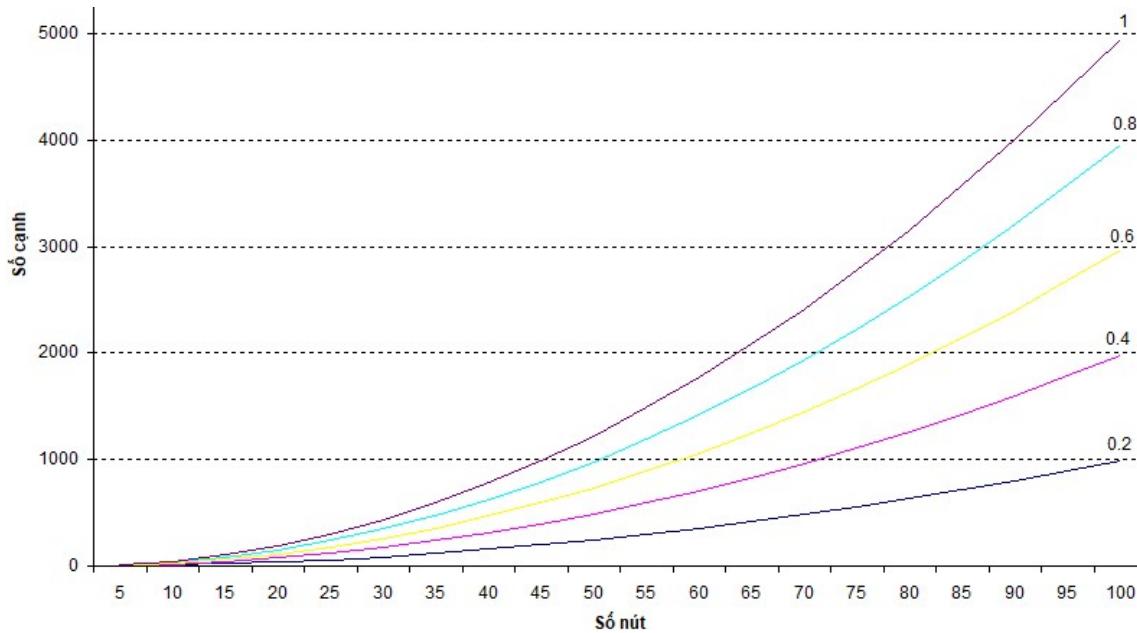
Vấn đề cuối cùng cần phải giải quyết là lỗi của một nút cha, khi đó nó sẽ tạm thời chọn nút gốc làm cha và khởi động lại giải thuật tối ưu hóa để gán các nút con của nút bị lỗi vào nút cha thích hợp, nếu nút gốc bị lỗi thì việc xây dựng cây sẽ phải khởi động lại từ đầu. Khó khăn lớn nhất khi triển khai thực tế nằm ở thu thập thông tin về hình trạng vật lý, các gói tin thường được vận chuyển trên các tuyến đường khác nhau. Rất khó xác định thời gian lan tỏa đến tất cả các thành viên, nó phụ thuộc vào cấu trúc của mạng phủ, trường hợp xấu nhất sẽ là  $T^*$  ( $N-1$ ), trong đó  $N$  là tổng số thành viên của nhóm và  $T$  là thời gian trung bình để vận chuyển thông điệp giữa hai thành viên.

#### 2.5.1.2 Quảng bá trong nhóm

Điều quan trọng trong truyền thông theo nhóm là giảm thiểu các nút trung gian mà thông điệp không được sử dụng, nút không sử dụng thông điệp gọi là nút chuyển tiếp, như vậy mạng phủ được xây dựng theo cách phân cấp thì chỉ nút lá mới là thành viên sử dụng thông điệp. Để tránh lãng phí truyền thông thì cần xây dựng mạng phủ cho mỗi nhóm, một thành viên có thể thuộc về nhiều nhóm, như vậy mỗi thành viên chỉ cần duy trì danh sách các thành viên của từng nhóm.

Giả sử mạng phủ tương ứng với một nhóm, mỗi thành viên đều có thể gửi thông điệp cho bất kỳ thành viên nào trong nhóm, khi đó việc chuyển thông điệp theo nhóm sẽ trở thành quảng bá thông điệp trong nhóm. Nhận được thông điệp, thành viên chỉ cần kiểm tra xem đó có phải là thông điệp mới hay không, nếu là thông điệp cũ thì bỏ qua, nếu là thông điệp mới chuyển tiếp đến các thành viên láng giềng ngoại trừ thành viên gửi thông điệp cho nó.

Như vậy số lượng thông điệp sau mỗi vòng gửi sẽ tăng gấp đôi, một thành viên có thể nhận được cùng một thông điệp nhưng từ nhiều thành viên khác, nghĩa là lãng phí kinh truyền. Có thể hình dung mạng phủ như một đồ thị  $N$  nút với  $M$  cạnh, nếu tổ chức hình cây thì trường hợp tối ưu nhất sẽ phải chuyển  $M=N-1$  thông điệp, trường hợp xấu nhất tất cả các thành viên được nối với nhau theo hình sao thì số lượng thông điệp lên tới  $M=N(N-1)/2$ .



Hình 2.19 Đồ thị quan hệ giữa số cạnh và số nút

Giả sử không có thông tin gì về cấu trúc mạng phủ, như vậy có thể xây dựng đồ thị ngẫu nhiên, ký hiệu  $p_{edge}$  xác suất hai thành viên bất kỳ được nối với nhau, đồ thị khi đó sẽ có  $M = p_{edge} N(N-1)/2$  cạnh. Có thể giảm số lượng thông điệp bằng cách sử dụng phương pháp xác suất Banaei-Kashani và Shahab, mỗi thành viên sẽ sử dụng xác suất  $p_{flood}$  để chọn  $p_{flood}N$  thành viên chuyển tiếp, trong đó  $N$  là số lượng thành viên láng giềng.

Ví dụ trên hình 2.19, nhóm gồm có 100 thành viên, nếu mỗi thành viên đều gửi thông điệp cho các thành viên khác thì số thông điệp lên tới 4950. Nếu áp dụng xác suất  $p_{flood}=0.2$  thì số lượng thông điệp chỉ còn 1000, đó là con số vẫn còn lớn. Mặc dù số lượng thông điệp sẽ giảm nhưng lại nảy sinh rủi ro không phải tất cả các thành viên đều nhận được thông điệp, vì vậy cần phải thêm yếu tố số lượng thành viên láng giềng trong quyết định chuyển tiếp.

Schlosser đã thiết kế lược đồ quảng bá hiệu quả dựa vào việc theo dõi các thành viên láng giềng trên mỗi chiều, số lượng chiều được xác định bằng cách lấy số bit tối thiểu có thể biểu diễn định danh của tất cả các thành viên. Định danh mỗi thành viên được thể hiện bằng chuỗi bit với chiều dài bằng số lượng chiều của nhóm, mỗi cạnh trong mạng phủ được gán nhãn với giá trị của nó thể hiện vị trí của bit đã thay đổi tính từ bên trái qua phải khi so sánh chuỗi bit định danh của hai thành viên.

Tập láng giềng được xác định bằng cách thay đổi lần lượt từng bit của định danh thành viên, ví dụ xét trường hợp số lượng chiều  $N=4$ , thành viên 0000 sẽ có tập láng giềng là {0001, 0010, 0100, 1000}, thành viên 0001 có tập láng giềng là {0000, 0011,

$0101, 1001\} \dots$ , cạnh giữa 0000 và 0001 được gán nhãn 4 tương ứng với vị trí bit thứ tư đã bị thay đổi, cạnh giữa 0000 và 0100 được gán nhãn bằng 2 tương đương với việc bit thứ hai bị thay đổi.

Thành viên ban đầu quảng bá thông điệp m đến tất cả các thành viên láng giềng và đính kèm nhãn của cạnh đã gửi thông điệp, ví dụ thành viên 1001 có tập láng giềng là {1000, 1011, 1101, 0001}. Thành viên này lần lượt gửi thông điệp (m,1) đến thành viên 0001, thông điệp (m,2) đến thành viên 1101, thông điệp (m,3) đến thành viên 1011 và thông điệp (m,4) đến thành viên 1000. Khi một thành viên nhận được thông điệp, nó chỉ chuyển tiếp đến các thành viên láng giềng trên cạnh được gán giá trị cao hơn, ví dụ thành viên 1101 có tập láng giềng là {1100, 1111, 1001, 0101}, nhãn của thông điệp gửi đến là 2 do đó nó sẽ chỉ chuyển tiếp thông điệp này cho thành viên 1111 có nhãn là 3 và thành viên 1100 có nhãn là 4.

Sử dụng lược đồ này cho thấy cần quảng bá  $2^N - 1$  thông điệp, trong đó N là số chiều lượng chiều, dưới góc độ số lượng thông điệp lưu chuyển trong mạng thì đây là cách tối ưu. Ví dụ, để lan tỏa thông điệp trong nhóm gồm 100 thành viên thì số lượng chiều  $N=7$ , nghĩa là số lượng thông điệp lưu chuyển trên mạng không vượt quá 127, con số này nhỏ hơn đáng kể ngay cả khi áp dụng xác suất  $p_{flood}=0.2$ .

Số lượng thông điệp lưu chuyển trên mạng không những giảm đáng kể mà còn dự đoán được chính xác thời gian lan tỏa đến các thành viên, giá trị của nó tỉ lệ thuận với số lượng chiều. Ví dụ trong bảng 2.5, số lượng chiều là 3 do đó cho phép tối đa 8 thành viên trong nhóm, định danh của các thành viên được viết dưới dạng nhị phân lần lượt từ 000 đến 111, mỗi ô giao điểm của hàng và cột thể hiện nhãn liên kết giữa các thành viên láng giềng.

Bảng 2.5 Láng giềng và nhãn các cạnh trong nhóm 8 thành viên

Thành viên	000	001	010	011	100	101	110	111
<b>000</b>		3	2		1			
<b>001</b>	3			2		1		
<b>010</b>	2			3			1	
<b>011</b>		2	3					1
<b>100</b>	1					3	2	
<b>101</b>		1			3			2
<b>110</b>			1		2			3
<b>111</b>				1		2	3	

Giả sử thông điệp xuất phát từ thành viên **011**, vòng thứ nhất thông điệp được chuyển đến thành viên **111**, nhãn là 1 nên chắc chắn thành viên **111** sẽ phải chuyển tiếp thông điệp đến thành viên **101** và **110**. Vòng thứ hai, thành viên **011** gửi thông điệp đến thành viên **001** trong khi đó thành viên **111** gửi thông điệp đến thành viên **101**, hai sự kiện được thực hiện song song, thành viên **101** sẽ chỉ cần chuyển thông điệp cho thành viên **100**. Vòng thứ ba, thành viên **011** chuyển thông điệp cho thành viên **010**, cùng thời gian đó thành viên **111** sẽ chuyển thông điệp cho thành viên **110** và thành viên **101** chuyển thông điệp cho thành viên **100**, ba sự kiện được thực hiện song song, thông điệp đã được lan tỏa đến tất cả các thành viên của nhóm.

### 2.5.2 Lan truyền ngẫu nhiên

Phương pháp lan truyền ngẫu nhiên dựa trên nguyên lý lan truyền bệnh dịch, không cần thành phần điều phối mà chỉ cần sử dụng thông tin cục bộ vẫn truyền thông nhanh chóng trong tập hợp lớn các thành viên. Khi cập nhật xảy ra tại một thành viên, giải thuật lan truyền được thực hiện sao cho thông tin đến được tất cả các thành viên khác một cách nhanh nhất có thể. Như vậy mỗi thành viên có thể nhận được thông điệp từ nhiều thành viên khác, để phân biệt dữ liệu cũ với dữ liệu mới, nhãn thời gian sẽ được đính kèm vào mỗi thông điệp lan truyền.

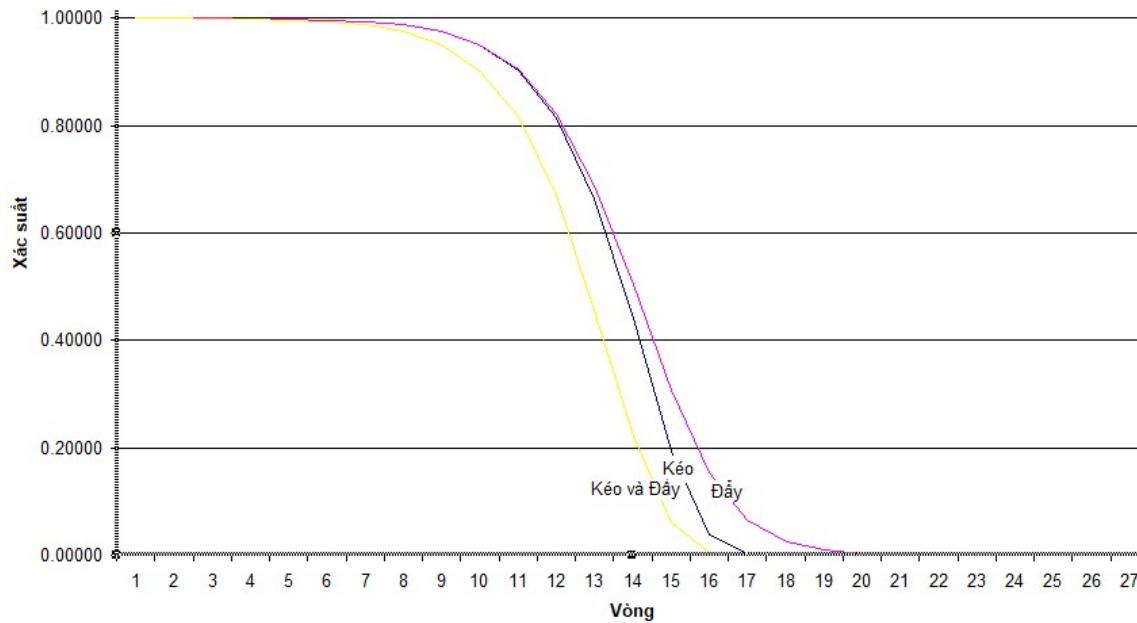
Thông điệp ở đây được hiểu là các yêu cầu, tuy nhiên nó có thể là dữ liệu để bổ sung vào kho lưu trữ của mỗi thành viên, nhưng nó cũng có thể là yêu cầu xóa một mục dữ liệu nào đó trên các thành viên. Một thành viên được gọi là đã cập nhật nếu nó nắm giữ dữ liệu muôn phổ biến cho các thành viên khác, thành viên chưa nhận được thông điệp gọi là thành viên chưa cập nhật, một thành viên không sẵn sàng hoặc không thể phát tán dữ liệu của mình bị coi là thành viên không tham gia.

Mô hình lan truyền thường dùng là mô hình chống thất thoát (anti-entropy), giả sử thành viên P chọn ngẫu nhiên thành viên Q, có ba cách để chúng trao đổi bản tin cập nhật. Cách thứ nhất P chỉ đẩy các bản tin cập nhật của mình sang Q, cách thứ hai P chỉ kéo các bản tin cập nhật mới từ Q và cách thứ ba P đẩy bản tin cập nhật của mình cho Q nhưng đồng thời kéo các bản tin cập nhật mới từ Q.

Để lan truyền cập nhật xảy ra nhanh chóng thì cách tiếp cận thứ nhất không phải là lựa chọn tốt, nhiều thành viên được đánh dấu là đã cập nhật, do đó xác suất để tìm ra thành viên chưa cập nhật rất thấp, vì vậy thành viên chờ cập nhật có thể phải chờ thời gian khá lâu vì không được thành viên nào chọn. Trong trường hợp này, cách tiếp cận thứ hai sẽ tốt hơn, một thành viên chưa cập nhật chọn ngẫu nhiên nút khác, nếu đó là thành viên đã cập nhật thì sẽ kéo dữ liệu về, xác suất chọn được thành viên đã cập nhật sẽ cao hơn. Nếu chỉ có duy nhất một thành viên được cập nhật thì có thể sử dụng cả hai cách tiếp cận, do đó cách tiếp cận thứ ba vẫn là lựa chọn tốt nhất.

Chu kỳ mỗi thành viên thực hiện một lần trao đổi cập nhật với thành viên khác được chọn ngẫu nhiên gọi là vòng, số vòng để lan truyền một cập nhật đến tất cả các thành viên là  $O(\log(N))$ , trong đó N là số lượng các thành viên trong nhóm, điều đó cho thấy phương pháp lan truyền khá nhanh chóng, nhưng quan trọng hơn cả là khả năng mở rộng qui mô. Giả sử nhóm gồm N thành viên, một thành viên bất kỳ bắt đầu chuyển thông điệp cho các thành viên khác, gọi  $p_i$  là xác suất thành viên P chưa nhận được thông điệp sau vòng thứ i. Nếu sử dụng phương pháp đẩy, sau vòng thứ i sẽ có  $N(1-p_i)$  thành viên đã nhận được thông điệp, ở vòng thứ  $i+1$  xác suất số lượng thành viên liên hệ với P là  $1 - 1/(N-1)$ , như vậy xác suất thành viên P không nhận được thông điệp được tính theo công thức  $p_{i+1} = p_i(1 - 1/(N-1))^{N(1-p_i)}$ . Nếu sử dụng phương pháp kéo, thành viên P liên hệ với thành viên khác nhưng thành viên đó cũng chưa nhận được thông điệp, do đó xác suất P không nhận được bản tin là  $p_{i+1} = p_i^2$ . Nếu sử dụng kết hợp cả đẩy và kéo, thành viên P liên hệ với thành viên chưa nhận được thông điệp và cũng không có thành viên nào liên lạc với P thì xác suất sẽ là  $p_{i+1} = (p_i)^2 p_i (1 - 1/(N-1))^{N(1-p_i)}$ .

Hình 2.20 thể hiện đồ thị xác suất theo ba phương pháp lan truyền đã nêu trên, trục hoành thể hiện thứ tự các vòng lan tỏa thông điệp và trục tung thể hiện xác suất mỗi thành viên chưa nhận được thông điệp. Giả sử nhóm gồm có 10000 thành viên, thể hiện bằng đồ thị cho thấy kể từ vòng thứ tám xác suất các thành viên chưa nhận được thông điệp giảm nhanh đáng kể, chỉ cần đến vòng thứ 17 thì xác suất một thành viên chưa nhận được thông điệp xấp xỉ bằng 0, điều đó minh chứng mô hình lan truyền ngẫu nhiên rất hiệu quả trong các hệ thống qui mô lớn.

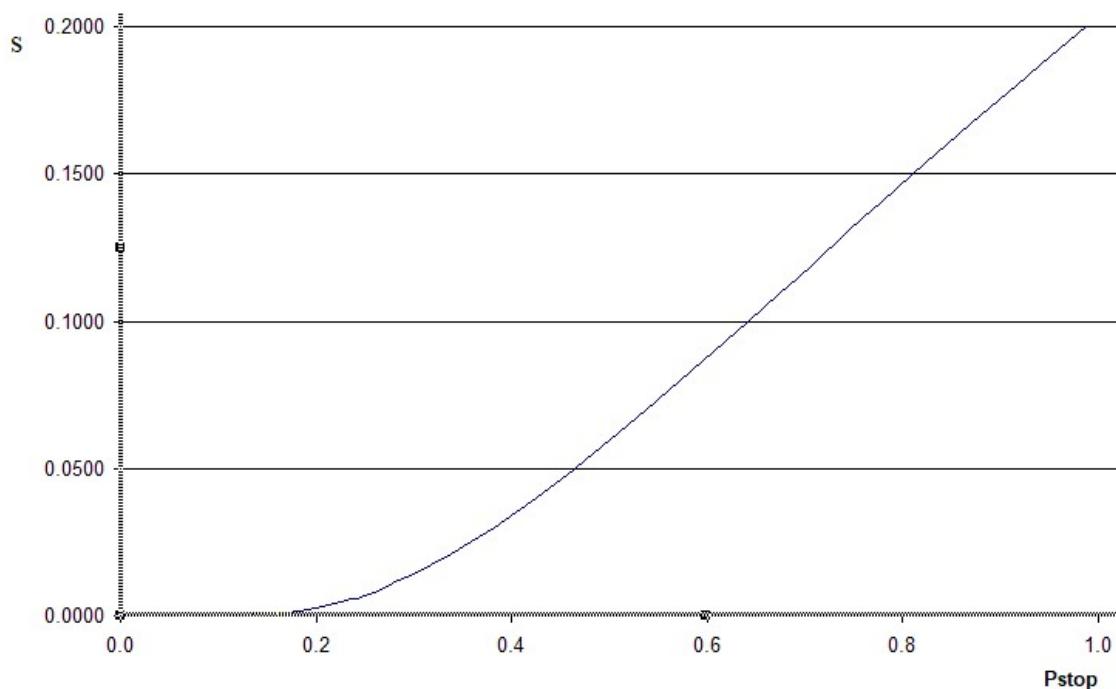


Hình 2.20 Đồ thị xác suất lan truyền ngẫu nhiên qua các vòng

Một biến thể khác của phương pháp lan truyền ngẫu nhiên gọi là lan truyền tin đồn, giả sử thành viên P gửi thông điệp cho thành viên Q, nếu nó đã nhận được thông điệp này từ thành viên khác thì nó sẽ thông báo cho thành viên P để thành viên này không tiếp tục gửi thông điệp cho các thành viên khác, gọi xác suất dừng lan truyền là  $p_{stop}$ . Như vậy số lượng thông điệp lan truyền sẽ giảm đi, tuy nhiên cách làm này không đảm bảo tất cả các thành viên đều nhận được thông điệp, ký hiệu s là xác suất các thành viên chưa nhận được thông điệp, u là xác suất các thành viên đã nhận được thông điệp và vẫn đang liên hệ với các thành viên khác để lan truyền thông điệp, r là xác suất các thành viên đã nhận được thông điệp nhưng đã từ bỏ lan truyền thông điệp, rõ ràng  $s+u+r=1$ . Sử dụng lý thuyết xác suất thống kê cho thấy kết quả quan hệ giữa s và  $p_{stop}$  thỏa mãn phương trình  $s=e^{-(1/p_{stop}+1)(1-s)}$ .

Đồ thị trên hình 2.21 biểu diễn tương quan giữa s và  $p_{stop}$  cho thấy, khi  $p_{stop}$  có giá trị xấp xỉ 0.2 thì  $s=0.0025$ , với những giá trị lớn hơn 0.2 thì giá trị s sẽ tăng nhanh và do đó cần các biện pháp bổ sung để đảm bảo tất cả các thành viên đều nhận được thông điệp. Ưu điểm chính của phương pháp lan truyền ngẫu nhiên là khả năng hội tụ nhanh, số lượng thao tác đồng bộ tương đối nhỏ so với các phương pháp khác. Đối với các hệ thống được kết nối qua mạng diện rộng thì cần phải xem xét cấu trúc liên kết mạng để có thể đạt kết quả tốt hơn, các thành viên sẽ chỉ liên hệ với vài thành viên khác với xác suất

tương đối cao, nên ưu tiên cập nhật cho các thành viên cầu nối giữa các mạng, giải pháp này gọi là lan truyền có định hướng.



Hình 2.21 Đồ thị tương quan giữa xác suất dừng với xác suất chưa cập nhật

Ván đè cuối cùng là lan truyền thông điệp chứa yêu cầu xóa mục dữ liệu nào đó, đây là công việc không hề đơn giản, bản chất của nó nằm ở chỗ thông điệp đã bị xóa sẽ lại được các thành viên khác gửi thông điệp cập nhật. Khi đã xóa một thông điệp ra khỏi kho lưu trữ, nó nhận được thông điệp từ các thành viên khác, kiểm tra trong kho dữ liệu không thấy thông điệp đã bị xóa, nó coi đó là thông điệp mới, do đó sẽ ghi vào kho lưu trữ và lại tiếp tục quá trình lan truyền cho các thành viên khác. Một giải pháp đơn giản là lưu lịch sử thao tác xóa, khi nhận được thông điệp cập nhật từ các thành viên khác, trước hết nó phải kiểm tra nhật ký để quyết định có thực hiện cập nhật hay không.

Cùng với thời gian, kho dữ liệu lịch sử thao tác xóa có thể rất lớn, điều này làm ảnh hưởng tới hiệu năng hệ thống, do đó cần phải tính toán thời gian hội tụ của các thông điệp lan truyền, sau khoảng thời gian này có thể loại bỏ những bản ghi lịch sử không còn giá trị sử dụng. Giả sử thành viên  $P$  đã xóa mục dữ liệu  $x$ , sau khoảng thời gian qui định cho phép xóa dữ liệu lịch sử nó ngẫu nhiên nhận được thông điệp cập nhật cho mục dữ liệu  $x$ , khi đó  $P$  sẽ chuyển thông điệp xóa mục dữ liệu  $x$  đến các thành viên khác. Để đảm bảo chắc chắn thông điệp yêu cầu xóa được chuyển đến tất cả các thành viên, hệ thống nên chọn một số thành viên không bao giờ xóa dữ liệu lịch sử.

## THẢO LUẬN

1. Nêu đặc điểm của truyền thông đồng bộ và không đồng bộ.
2. Tìm hiểu nguyên lý truyền thông của các phần mềm thu điện tử.
3. Thủ nghiệm triển khai Socket cho các giao thức TCP và UDP.

4. Liệt kê những khó khăn và giải pháp xử lý khi sử dụng giao thức TCP và UDP trên mạng Internet.
5. Thử nghiệm tiện ích nhắn tin có sẵn trong các hệ điều hành.
6. Nêu những ưu điểm và nhược điểm của truyền thông sử dụng các hàm nguyên thủy của hệ điều hành.
7. Cài đặt thư viện MPI và thử nghiệm ứng dụng nhắn tin đơn giản.
8. Xây dựng ứng dụng nhắn tin văn bản sử dụng giao thức TCP hoặc UDP, đánh giá vấn đề hiệu năng và lỗi.
9. Xây dựng ứng dụng video sử dụng giao thức TCP hoặc UDP, đánh giá vấn đề hiệu năng và lỗi.
10. Nêu những vấn đề về truyền thông cần giải quyết trong hệ thống hội thảo trực tuyến.

## **CHƯƠNG 3: ĐẶT TÊN TRONG HỆ THỐNG PHÂN TÁN**

Tên là một xâu ký tự dùng để phân biệt các thực thể, nó dùng để tham chiếu đến các đối tượng cụ thể hoặc trừu tượng, số lượng tên cho mỗi thực thể không bị hạn chế. Tên của thực thể đóng vai trò quan trọng trong tất cả các hệ thống máy tính, chúng xác định thực thể duy nhất hoặc tham chiếu đến vị trí của các thực thể cung cấp dịch vụ....

Dịch vụ đặt tên là nền tảng của bất kỳ hệ thống máy tính nào, nó cho phép tìm kiếm thực thể một cách chính xác và nhanh chóng. Tìm kiếm là một trong những nhiệm vụ quan trọng của các hệ thống thông tin, hiệu năng tìm kiếm phụ thuộc rất nhiều vào cách thức đặt tên và tổ chức dữ liệu. Chức năng chính của dịch vụ đặt tên là ánh xạ tên thân thiện với người sử dụng và thực thể, nó bao trùm phạm vi rộng lớn của hệ thống phân tán.

### **3.1 Tên, định danh và địa chỉ**

Để tìm kiếm một thực thể thì phải biết tên của nó, hệ thống đặt tên xác định cú pháp mà tên phải tuân thủ, đó là những quy tắc đặt tên. Tên thân thiện là các tên được đặt một cách dễ hiểu và thân thuộc với con người, nó gợi nhớ cho con người một số đặc điểm của thực thể.

Trong các hệ thống máy tính, tập hợp các tên tham chiếu đến nhiều loại thực thể khác nhau gọi là không gian tên, xét về mặt hình học, đây là đồ thị có hướng và gọi là đồ thị tên có cấu trúc. Đồ thị tên có cấu trúc là đồ thị mà mỗi nút là miêu tả một thực thể, mỗi nút thư mục gắn với nhiều nút khác và được lưu trữ trong bảng thư mục gồm tập các cặp nhãn và định danh của thực thể.

Định danh là một loại tên đặc biệt, nó bảo đảm tính duy nhất trong hệ thống máy tính, mỗi thực thể chỉ có duy nhất một định danh và mỗi định danh tham chiếu duy nhất một thực thể. Trong hệ thống phân tán, giá trị của định danh thường được tạo ngẫu nhiên không sử dụng lại để tránh những trường hợp nhập nhằng.

Để sử dụng dịch vụ của một thực thể thì phải biết điểm truy nhập của nó, điểm truy nhập này cũng phải được đặt tên và gọi là địa chỉ. Địa chỉ cũng là một loại tên đặc biệt, nó cung cấp thông tin về vị trí của thực thể. Một thực thể có thể sở hữu nhiều địa chỉ, nó có thể thay đổi địa chỉ trong quá trình tồn tại. Một địa chỉ có thể trỏ đến các thực thể khác nhau nhưng phải ở những thời điểm khác nhau, tại một thời điểm địa chỉ sẽ trỏ đến một thực thể duy nhất.

### **3.2 Đặt tên và các giải pháp tìm kiếm**

Tên chỉ là xâu ký tự nhưng việc đặt tên sao cho phù hợp nhất cho đối tượng sử dụng lại là vấn đề đòi hỏi những qui tắc nhất định, đối tượng sử dụng ở đây là con người hoặc máy tính. Con người quen với những tên thân thiện, đó là những tên dễ nhớ và phản ánh các thuộc tính của thực thể, những tên như vậy lại là trở ngại trong xử lý của các máy tính.

Tên của các thực thể trong hệ thống phân tán có thể được đặt phi cấu trúc, có cấu trúc hoặc dựa trên thuộc tính. Với mỗi cách đặt tên cần có các giải thuật tìm kiếm sao cho

có thể xác định các thực thể một cách chính xác và hiệu quả nhất. Trong hệ thống phân tán, dịch vụ đặt tên thường được cài đặt trên nhiều máy tính, do đó qui tắc đặt tên không những phải đảm bảo tính duy nhất mà còn phải đáp ứng yêu cầu về hiệu năng khi qui mô mở rộng.

### 3.2.1 Đặt tên phi cấu trúc

Tên phi cấu trúc còn gọi là tên phẳng, đơn thuần chỉ gồm chuỗi các bit ngẫu nhiên không phản ánh bất kỳ thông tin nào liên quan tới thực thể. Để tham chiếu đến thực thể thì phải xác định điểm truy nhập của nó, nghĩa là xác định địa chỉ của thực thể trong khi không có bất kỳ manh mối nào, điều đó đòi hỏi xây dựng những giải pháp tìm kiếm phù hợp cho từng hệ thống.

Tìm kiếm trực tiếp mà không cần tổ chức kho dữ liệu gọi là giải pháp đơn giản, nó sử dụng cơ chế truyền thông quảng bá/nhóm hoặc con trỏ chuyển tiếp. Tìm kiếm bằng cách truy vấn đến kho dữ liệu tập trung gọi là giải pháp nguồn gốc, giải pháp này khó có thể đáp ứng yêu cầu qui mô mở rộng. Với những hệ thống lớn, kho dữ liệu cần được phân tán, có thể tổ chức dựa trên hàm băm hoặc phân cấp.

#### 3.2.1.1 Giải pháp đơn giản

Nếu không có kho dữ liệu để truy vấn, cách đơn giản là gửi định danh của thực thể cần tìm tất cả các thành viên, có thể sử dụng cơ chế truyền thông quảng bá hoặc truyền thông theo nhóm. Nhận được thông điệp, thành viên nào có thực thể trùng khớp với định danh cần tìm thì gửi lại một thông điệp chứa địa chỉ của điểm truy nhập. Phương pháp này đòi hỏi tất cả các thành viên đều phải lắng nghe và tiếp nhận yêu cầu gửi đến, nó chỉ phù hợp với qui mô nhỏ, do đó thường chỉ được áp dụng trong các mạng cục bộ.

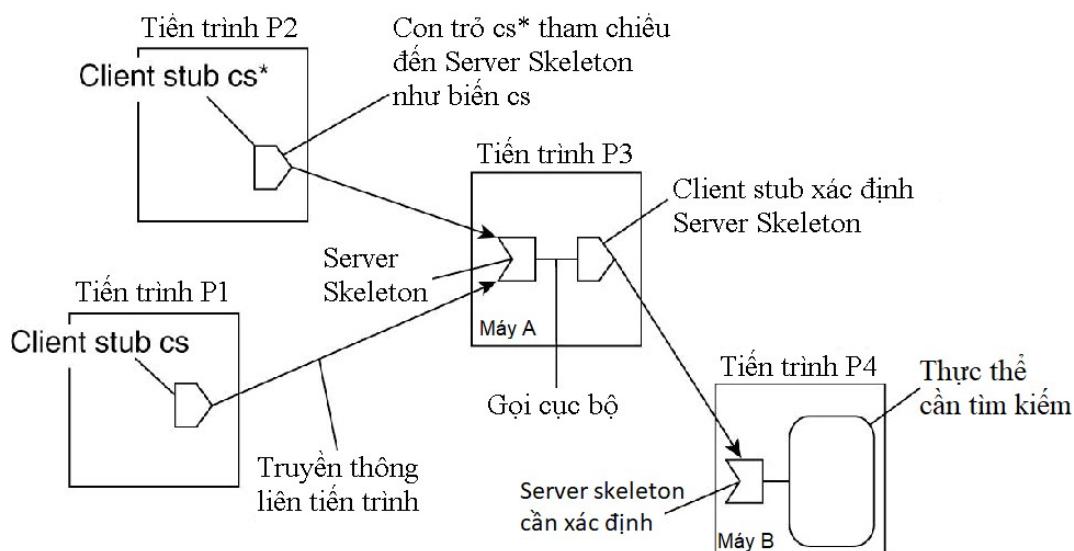
Ví dụ trong mạng cục bộ, việc tìm kiếm thực thể bằng phương pháp quảng bá rất đơn giản, định danh của thực thể cần tìm sẽ được gắn vào thông điệp gửi quảng bá đến tất cả các máy, máy nào đáp ứng yêu cầu thì phản hồi. Nguyên lý này là cơ sở để xây dựng giao thức ARP dùng để tìm kiếm địa chỉ vật lý khi biết địa chỉ IP, nó tạo khung dữ liệu với địa chỉ vật lý đích là địa chỉ quảng bá FFFFFFFFFF, khung dữ liệu sẽ được chuyển đến tất cả các máy trong mạng, chỉ máy nào có địa chỉ trùng khớp với địa chỉ IP cần tìm thì mới phản hồi.

Nếu qui mô mạng lớn hơn thì giải pháp quảng bá sẽ kém hiệu quả, nó không những lãng phí băng thông mà còn làm gián đoạn công việc của các máy khác. Để khắc phục nhược điểm này có thể hạn chế một số máy nhận được yêu cầu bằng cách sử dụng giải pháp truyền thông theo nhóm, tiêu chuẩn Ethernet 802.3 qui định bit thấp nhất của byte cao nhất bằng 1 thì đó là địa chỉ vật lý truyền thông theo nhóm. Truyền thông theo nhóm cũng có thể được sử dụng để xác định các thực thể trong mạng điểm-điểm, tầng mạng dành riêng địa chỉ IP lớp D dùng cho truyền thông theo nhóm.

Một phương pháp khác là dùng con trỏ chuyển tiếp, nó dựa trên nguyên tắc một thực thể khi rời sang vị trí khác thì phải để lại thông tin để có thể tham chiếu tới vị trí mới. Khi cần tìm một thực thể, máy khách có thể xác định ngay được địa chỉ hiện tại của thực thể này bằng cách dò tìm lưu vết qua các thành viên chuyển tiếp, tuy nhiên nếu chuỗi các con trỏ quá dài sẽ dẫn đến hiệu năng thấp và nguy cơ đứt đường truyền liên kết.

Phương pháp con trỏ chuyển tiếp thường dùng cho các thực thể di động, khi thực thể di chuyển, nó để lại vị trí cũ một tham chiếu đến vị trí mới, như vậy chỉ cần lùi theo chuỗi các con trỏ chuyển tiếp sẽ tìm thấy thực thể. Nếu một thực thể hay di chuyển thì chuỗi con trỏ sẽ rất lớn, điều đó dẫn đến hậu quả tăng chi phí tìm kiếm. Các nút trung gian sẽ phải duy trì con trỏ chuyển tiếp, nếu con trỏ chuyển tiếp bị lỗi trên một nút nào đó thì sẽ không thể tìm thấy thực thể.

Ví dụ xét trường hợp con trỏ chuyển tiếp trong gọi thủ tục từ xa trên hình 3.1, mỗi thủ tục từ xa sinh ra một cặp Stub dành cho máy khách và Skeleton dành cho máy chủ. Giả sử đối tượng di chuyển từ máy A sang máy B, khi tiến trình P<sub>1</sub> trên máy khách gọi thủ tục tìm kiếm thực thể, nó sẽ gửi yêu cầu đến tiến trình P<sub>3</sub> cài đặt trên máy A. Nếu đối tượng vẫn nằm trên máy A thì tiến trình P<sub>3</sub> chỉ cần gọi cục bộ và trả về kết quả cho máy khách, nhưng vì thực thể đã chuyển sang máy B, tiến trình P<sub>3</sub> sẽ gửi yêu cầu sang tiến trình P<sub>4</sub> trên máy B, sau khi nhận được kết quả từ P<sub>4</sub>, tiến trình P<sub>3</sub> sẽ trả về kết quả cho tiến trình P<sub>1</sub>.

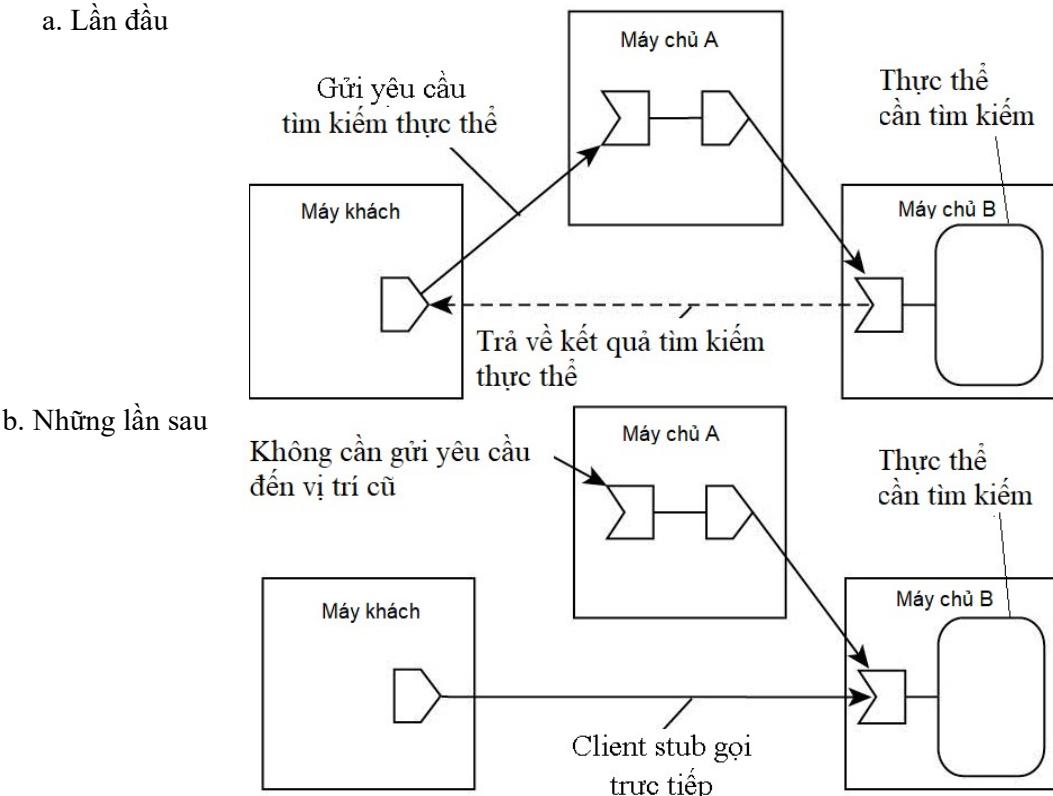


Hình 3.1 Nguyên lý con trỏ chuyển tiếp

Giải pháp con trỏ chuyển tiếp che giấu toàn bộ quá trình tìm kiếm thực thể, tiến trình máy khách chỉ biết đến tiến trình máy chủ đầu tiên mà không biết đến các tiến trình khác trong chuỗi con trỏ chuyển tiếp, quá trình tìm kiếm trên máy chủ hoàn toàn trong suốt đối với máy khách, tính bảo mật không những sẽ tốt hơn mà còn có thể tiết kiệm địa chỉ IP công cộng bằng cách gán cho các máy chủ địa chỉ IP riêng.

Hình 3.2 thể hiện một biến thể cài đặt phương pháp con trỏ chuyển tiếp, nó cài đặt thêm chức năng định hướng lại con trỏ chuyển tiếp. Tiến trình máy khách gửi yêu cầu đến tiến trình máy chủ A, nhưng tiến trình quản lý thực thể đã chuyển sang máy chủ B, do đó tiến trình máy chủ A sẽ chuyển tiếp đến tiến trình máy chủ B nhưng kèm theo thông tin để tiến trình máy chủ B có thể trực tiếp trả về kết quả cho tiến trình máy khách, thông tin ở đây thường là cặp địa chỉ IP và số hiệu cổng tiến trình máy khách đã sử dụng để gửi yêu cầu.

Nhận được kết quả, máy khách sẽ cập nhật cấu hình của mình để lần sau khi cần tìm thực thể sẽ gọi trực tiếp đến tiến trình máy chủ B, như vậy hiệu năng sẽ tăng lên nhưng không còn duy trì được tính trong suốt, không che giấu được vị trí của tiến trình máy chủ B và cũng không tiết kiệm được địa chỉ IP công cộng. Cách thực hiện như vậy tiềm ẩn nguy cơ an ninh, có thể cài tiến bằng cách tiến trình máy chủ A vẫn chuyển tiếp yêu cầu sang tiến trình máy chủ B, kết quả trả về vẫn phải qua tiến trình trên máy A để chuyển đến tiến trình máy khách. Trong thời gian tiến trình máy chủ B xử lý, tiến trình máy chủ A tạo bản tin thông báo vị trí mới của tiến trình máy chủ B để tiến trình máy khách cập nhật cấu hình của mình.



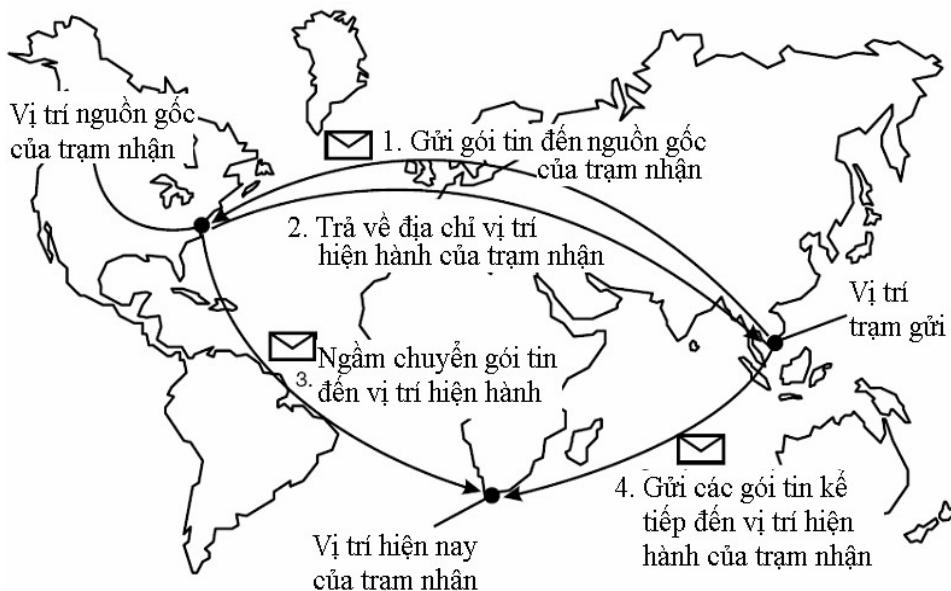
Hình 3.2 Định hướng lại con trỏ chuyển tiếp

Khi tất cả các máy khách đã được cập nhật vị trí mới của thực thể, máy A có thể loại bỏ bản ghi con trỏ chuyển tiếp liên quan đến thực thể, điều này về lý thuyết thì đơn giản nhưng thực tế sẽ phức tạp hơn nhiều. Sự cố phát sinh nếu một trong những máy chủ của chuỗi con trỏ chuyển tiếp gặp lỗi, có thể giải quyết vấn đề này bằng cách bố trí máy chủ luôn lưu giữ thông tin tham chiếu đến vị trí hiện hành của thực thể, nó được cài đặt theo cơ chế dự phòng, chỉ khi nào xảy ra lỗi trong chuỗi con trỏ chuyển tiếp thì mới tham chiếu đến máy chủ này.

### 3.2.1.2 Giải pháp dựa trên nguồn gốc

Giải pháp dựa trên nguồn gốc hỗ trợ các thực thể thường hay thay đổi vị trí trong các hệ thống qui mô lớn, địa chỉ của thực thể được đăng ký với một máy chủ gọi là nguồn gốc, khi di chuyển sang vị trí khác thực thể phải cập nhật địa chỉ mới. Cách tiếp cận này

được sử dụng như một cơ chế dự phòng khi một măt xích trong chuyến tiếp con trỏ gặp lỗi.



Hình 3.3 Nguyên lý cách tiếp cận dựa trên nguồn gốc

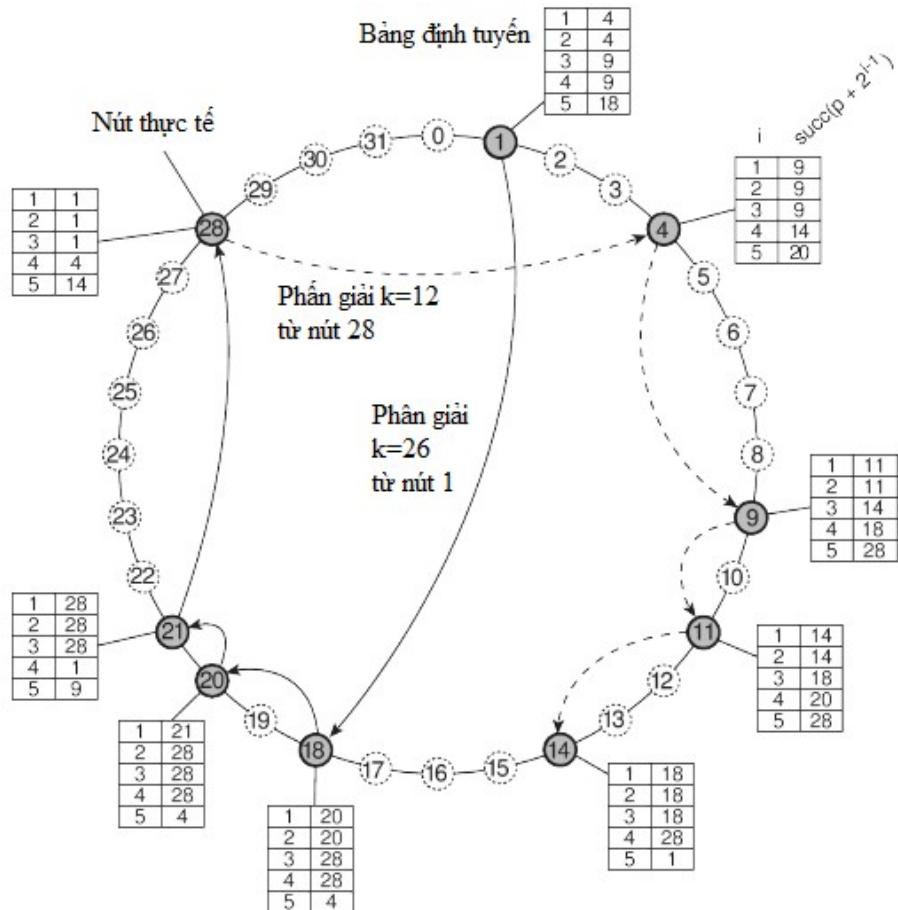
Hình 3.3 minh họa nguyên lý hoạt động của cách tiếp cận dựa trên nguồn gốc trong mạng IP di động, mỗi thực thể di động được máy chủ gốc cấp phát một địa chỉ IP, mọi liên lạc với địa chỉ IP này trước tiên sẽ được chuyển đến tiến trình đại diện mạng gốc. Nếu thiết bị di chuyển sang mạng khác, nó yêu cầu mạng mới cấp địa chỉ IP tạm thời để liên lạc, giá trị này sẽ được chuyển đến tiến trình đại diện mạng gốc để cập nhật cơ sở dữ liệu.

Khi tiến trình nhận được gói tin dành cho thực thể di chuyển, nó tra cứu cơ sở dữ liệu để tìm ra địa chỉ hiện hành của thực thể, nếu cùng mạng thì chỉ cần chuyển tiếp gói tin, ngược lại nó sẽ trả về cho bên gửi một gói tin thông báo địa chỉ hiện hành của thực thể và đồng thời tạo đường hầm chuyển tiếp gói tin nhận đến tiến trình đại diện cho mạng mới, từ đó gói tin sẽ được chuyển tiếp đến thực thể di động. Thực thể di chuyển sẽ trả lời bằng cách chuyển gói tin đến tiến trình đại diện của mạng mới, gói tin sẽ được chuyển tiếp đến bên yêu cầu, những gói tin kế tiếp sẽ được bên gửi chuyển trực tiếp đến tiến trình đại diện mạng mới mà không cần phải qua tiến trình đại diện của mạng gốc.

Cơ chế chuyển đổi hoàn toàn trong suốt đối với tiến trình xử lý tầng ứng dụng, bên gửi sẽ ghi nhớ địa chỉ mới của thiết bị di động để sử dụng cho việc gửi các gói tin kế tiếp, thiết bị di động nhận được gói tin bằng cách giải mã bản tin nhận được từ đường hầm như thể nó đang sử dụng địa chỉ gốc. Phải tham chiếu đến máy chủ gốc là nhược điểm của phương pháp này, nó không những làm tăng độ trễ truyền thông mà còn không thể thực hiện được khi máy chủ gốc bị lỗi, nếu một thực thể quyết định di chuyển tới mạng mới vĩnh viễn thì cũng nên đăng ký lại địa chỉ sang máy chủ mới. Thông số vị trí của các tiến trình đại diện cho các mạng tương đối ổn định, do đó có thể lưu vào vùng đệm để sử dụng cho những lần tìm kiếm kế tiếp.

### 3.2.1.3 Tìm kiếm dựa trên bảng băm phân tán

Nếu cơ sở dữ liệu về thực thể quá lớn, cách xử lý tập trung sẽ làm suy giảm hiệu năng, do đó nhiều hệ thống đã xây dựng chức năng tìm kiếm dựa trên mạng Chord. Mạng Chord sử dụng không gian định danh m-bit để gán định danh được lựa chọn ngẫu nhiên cho các nút và các khóa cho mỗi thực thể, giá trị m có thể bằng 128 hoặc 160 tùy thuộc vào hàm băm được sử dụng.



Hình 3.4 Tổ chức các nút mạng Chord

Thực thể có khóa  $k$  sẽ thuộc quản lý của nút liền sau với định danh nhỏ nhất sao cho định danh  $id \geq k$  và được ký hiệu là  $succ(k)$ , để đơn giản ký hiệu  $p$  là định danh của nút  $P$ . Vấn đề chính cần giải quyết là phân giải một cách hiệu quả khóa  $k$  thành địa chỉ của nút liền sau, cách làm đơn giản nhất là giải pháp tuyến tính, mỗi nút  $P$  sẽ ghi nhớ nút liền trước và liền sau của nó, khi nhận được yêu cầu nó chỉ cần chuyển tiếp cho một trong hai nút này, nếu  $pred(p) < k \leq p$  thì nút  $P$  sẽ trả về địa chỉ qua mình cho tiến trình khởi tạo yêu cầu phân giải.

Thay cho giải pháp tuyến tính, mạng Chord duy trì bảng định tuyến chứa  $s \leq m$  thực thể, nếu  $FT_p$  biểu thị tuyến của nút  $P$  thì  $FT_p[i] = succ(p+2^{i-1})$ , nghĩa là mục thứ  $i$  trả đến nút đầu tiên kế tiếp  $p$  ít nhất  $2^{i-1}$ , khoảng cách giữa các nút tăng lên theo cấp số nhân. Để tra cứu khóa  $k$ , nút  $P$  tìm kiếm trên bảng định tuyến, giả sử bản ghi thứ  $j$  cho biết phải chuyển tiếp đến nút  $Q$ , trong đó định danh  $q = FT_p[j] \leq k < FT_p[j+1]$  hoặc  $q = FT_p[1]$

khi  $p < k < FT_p$  [1], nếu bảng định tuyến của mỗi nút chỉ có một bản ghi thì trở thành tìm kiếm tuyến tính.

Xét ví dụ trên hình 3.4, qui trình phân giải  $k=26$  khởi nguồn từ nút 1, bảng định tuyến của nút này cho thấy  $FT_1[5]$  thỏa mãn điều kiện, như vậy nó sẽ chuyển yêu cầu đến nút 18. Tại nút 18, ta thấy  $FT_{18}[2] \leq k < FT_{18}[3]$  nên nó sẽ chọn nút 20, tương tự như vậy nút 20 chọn bản ghi đầu tiên nên chuyển cho nút 21, nút 21 chuyển cho nút 28 chịu trách nhiệm cho khóa  $k=26$ , nó trả về cho nút 1 địa chỉ cần tìm. Nếu yêu cầu phân giải khóa  $k=12$  xuất phát từ nút 28, bản ghi thứ tư cho thấy cần phải chuyển sang nút số 4, tại đây bản ghi thứ ba thấy rằng cần phải chuyển sang nút số 9, nó lại chuyển tiếp cho nút số 11 và sau đó đến nút 14 chịu trách nhiệm cho  $k=12$  nên trả về địa chỉ cho nút 28. Có thể thấy quá trình phân giải sẽ thực hiện trên  $O(\log N)$  bước, với  $N$  là số nút trong hệ thống.

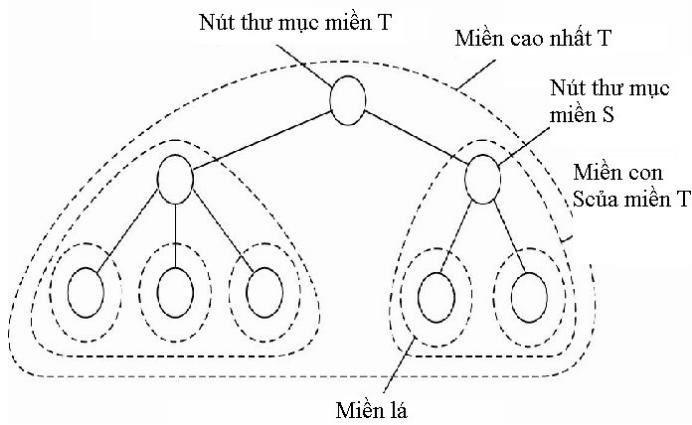
Số lượng các nút trong hệ thống có thể thay đổi, chúng có thể rời đi hoặc tham gia một cách tự nguyện nhưng cũng có thể do gặp lỗi nó phải rời đi, khắc phục xong lỗi lại tiếp tục tham gia. Trong mạng Chord, nếu một nút  $P$  muốn tham gia thì chỉ cần liên hệ với một nút bất kỳ và gửi thông điệp tìm kiếm nút liền sau  $\text{succ}(p+1)$ , nhận được kết quả sẽ tự chèn vào vòng logic của hệ thống. Quan trọng nhất là phải duy trì chính xác bản ghi đầu tiên, mỗi nút bất kỳ  $Q$  thường xuyên chạy một thủ tục để liên hệ với  $\text{succ}(q+1)$  và yêu cầu trả về liền trước của nút đó, nếu  $q=\text{pred}(\text{succ}(q+1))$  thì không cần phải cập nhật bảng định tuyến, ngược lại thì có thể hiểu một nút  $P$  đã được thêm vào hệ thống, với  $q < p \leq \text{succ}(q+1)$  thì  $Q$  sẽ điều chỉnh  $FT_q[1]$  thành  $p$ . Sau đó tiếp tục kiểm tra để đảm bảo chắc chắn  $P$  nhận  $Q$  là nút liền trước, nếu không đúng thì phải điều chỉnh lại  $FT_q[1]$ . Quá trình tương tự sẽ được thực hiện trên các bản ghi khác  $k = q + 2^{i-1}$ , những yêu cầu này sẽ được thường xuyên gửi đi bằng các phương tiện nền của hệ thống.

Các nút trong hệ thống luôn theo dõi các nút liền trước của chúng, nếu không liên lạc được với nút liền trước, nó chỉ cần đánh dấu trạng thái không xác định. Nếu một nút  $Q$  liên hệ với nút liền sau  $\text{succ}(q+1)$ , nhưng nút liền sau lại đã đặt  $Q$  không xác định thì nó thông báo cho  $\text{succ}(q+1)$  để ghi nhận  $Q$  là nút liền trước. Khi một nút rời khỏi mạng, tất cả các khóa được chuyển cho nút kế tiếp, sau đó thông báo cho nút kế tiếp và và nút liền trước, bảng định tuyến trên các nút khác sẽ được điều chỉnh.

Nếu một nút đột ngột rời khỏi mạng sẽ dẫn đến hiện tượng mất các khóa, các mục liên quan đến khóa, tất nhiên sẽ không truy vấn được một số khóa. Có thể giải quyết vấn đề này bằng cách mỗi nút lưu danh sách các nút liền sau gần nhất, nếu không liên lạc được với nút liền sau thì sẽ duyệt danh sách trên để tìm ra nút liền sau còn hoạt động và cập nhật lại bảng định tuyến.

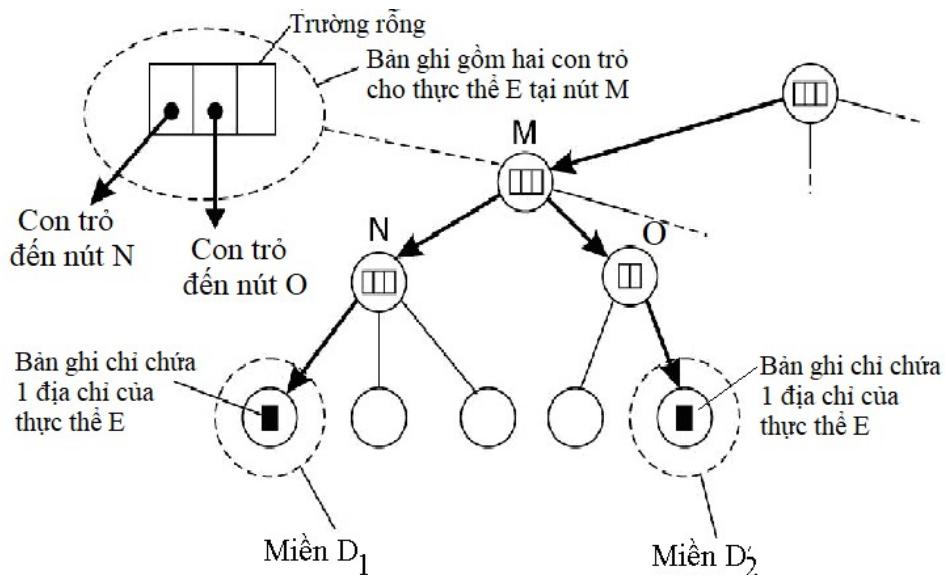
#### 3.2.1.4 Giải pháp phân cấp

Giải pháp phân cấp dựa trên dịch vụ định vụ toàn cầu, mạng được chia thành nhiều miền, trong đó có một miền cấp cao nhất bao trùm toàn bộ mạng. Mỗi miền lại có thể được chia thành nhiều miền con, miền thấp nhất gọi là lá, nó tương ứng với mạng cục bộ hoặc tế bào trong mạng điện thoại di động. Mỗi miền có một nút thư mục theo dõi các thực thể trong miền, do đó hình thành cây thư mục, nút thư mục của miền cấp cao nhất gọi là nút thư mục gốc, nó biết tất cả các thực thể.



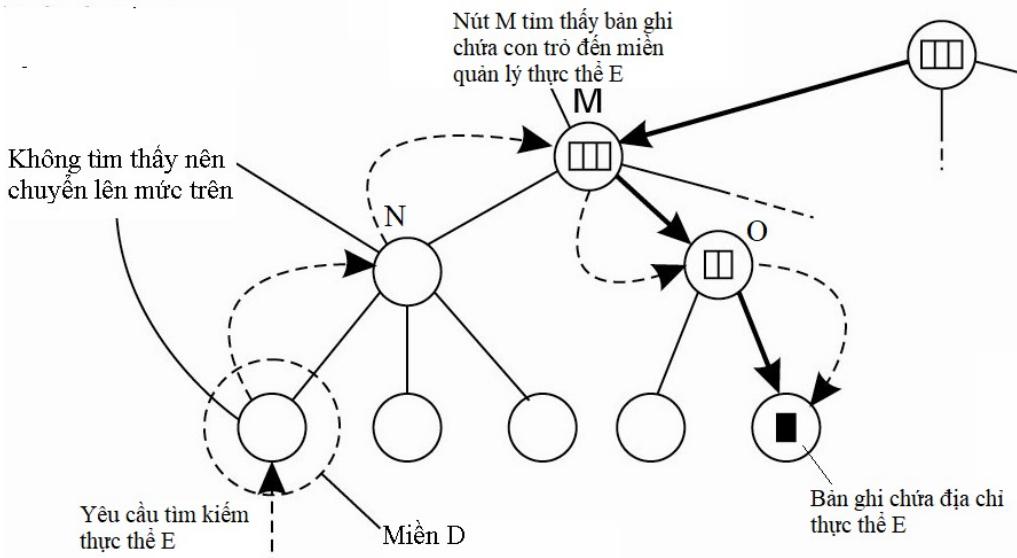
Hình 3.5 Tô chức phân cấp

Chỉ có nút lá mới lưu trữ địa chỉ hiện tại của thực thể, các nút thu mục lưu trữ con trỏ đến nút cấp thấp hơn. Một thực thể có thể sở hữu nhiều địa chỉ, mỗi địa chỉ lưu trữ tại một miền lá, do đó bản ghi của nút thu mục cấp trên sẽ phải lưu trữ nhiều con trỏ đến mỗi miền lá. Ví dụ thực thể E trên hình 3.6 có hai địa chỉ, một địa chỉ được lưu tại miền  $D_1$  của nút N và địa chỉ còn lại được lưu tại miền  $D_2$  của nút O, nút M là cấp trên của hai nút này nên bản ghi của nút M sẽ chứa hai con trỏ, nó thể hiện thực thể E được nhân bản tại hai vị trí khác nhau.



Hình 3.6 Lưu trữ thông tin cho thực thể có nhiều địa chỉ

Cách tiếp cận phân cấp ưu tiên tìm kiếm cục bộ, khu vực tìm kiếm được mở rộng mỗi khi yêu cầu phân giải được chuyển lên cấp cao hơn, trường hợp xấu nhất xảy ra khi yêu cầu đó được chuyển đến nút gốc. Vì nút gốc chưa bao giờ cho mỗi thực thể, do đó nó chỉ cần chuyển tiếp đọc theo các nút con để đến nút lá, ở đó sẽ tìm thấy địa chỉ của thực thể và trả về cho máy khách.



Hình 3.7 Tìm kiếm trong tổ chức phân cấp

Máy khách ở khu vực nào thì gửi yêu cầu tìm kiếm đến nút thư mục của khu vực đó, nếu nút thư mục cục bộ không tìm thấy thì mới chuyển lên nút cha. Như vậy, giải pháp phân cấp khai thác tính chất cục bộ, khu vực tìm kiếm ngày càng mở rộng nếu yêu cầu được chuyển lên nút cấp cao hơn, trường hợp xấu nhất sẽ lên tới nút gốc. Nút gốc chứa thông tin về tất cả các thực thể, do đó yêu cầu tìm kiếm lại được chuyển đến các nút lá theo đường dẫn trong bản ghi.

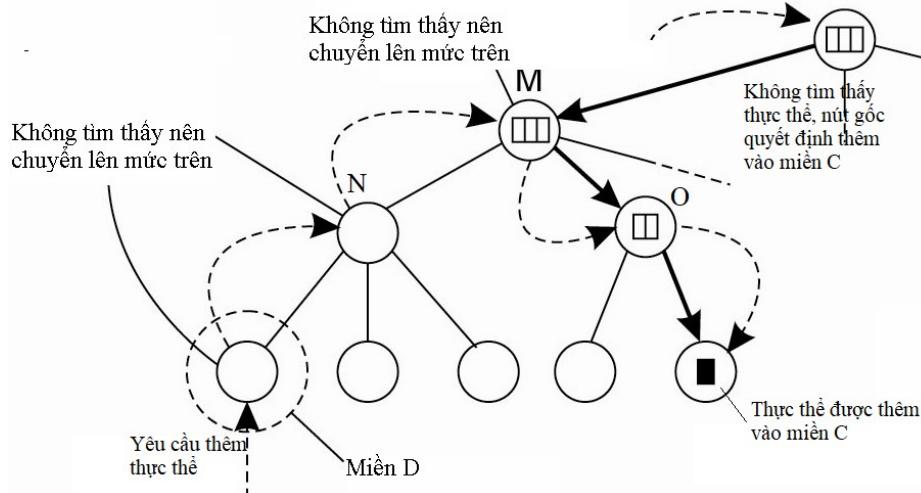
Ví dụ trên hình 3.7, máy khách nằm trong miền D, do đó yêu cầu tìm kiếm thực thể E được gửi đến nút lá quản lý miền D. Nút quản lý miền D không tìm thấy thực thể E nên chuyển yêu cầu đến nút cấp cao hơn, nút N cũng không chứa bản ghi về thực thể E nên yêu cầu lại được chuyển tiếp đến nút M. Nút M lưu bản ghi vị trí cho thực thể E, như vậy chỉ cần tìm kiếm trong miền do nút M phụ trách. Yêu cầu tìm kiếm được chuyển đến nút con O, cứ như vậy cho đến khi tìm thấy nút lá chứa địa chỉ của thực thể E, kết quả sẽ về cho máy khách.

Thao tác cập nhật cũng ưu tiên tính chất cục bộ, đầu tiên sẽ tìm kiếm thực thể trong miền máy khách đang hoạt động, nếu không tìm thấy thì mới chuyển yêu cầu lên các nút cấp cao hơn. Nếu không tìm thấy thực thể, nghĩa là cần thao tác thêm bản ghi mới, việc thêm bản ghi cũng xảy ra trong trường hợp một thực thể đang tồn tại nhưng yêu cầu được nhận bản.

Thao tác thêm bản ghi mới không những thực hiện ở miền quản lý thực thể mà còn phải cập nhật bản ghi con trỏ trên tất cả các nút trên đường dẫn đến nút gốc, có thể tiếp cận từ trên xuống dưới hoặc từ dưới lên trên. Sử dụng cách tiếp cận từ trên xuống dưới, khi máy khách muốn thêm một thực thể, ngay lập tức nút lá sẽ chuyển tiếp yêu cầu lên nút cha, cứ tiếp tục như vậy cho đến nút gốc, nếu không tìm thấy thực thể, nút gốc sẽ quyết định đường dẫn đến miền chịu trách nhiệm quản lý thực thể.

Hình 3.8 minh họa máy khách ở miền D gửi yêu cầu thêm thực thể, việc tìm kiếm được thực hiện qua các nút cấp cao hơn, cho đến tận nút gốc vẫn không tìm thấy bản ghi

nào liên quan đến thực thể này, do đó nút gốc quyết định lưu trữ tại miền C. Như vậy nút gốc sẽ thêm bản ghi cho thực thể này trở đến nút M trong cơ sở dữ liệu của nó, sau đó chuyển yêu cầu thêm thực thể đến nút M. Quá trình thêm bản ghi con trở tiếp tục được thực hiện trên các nút M và O, nút lá của miền C mới chính thức thêm bản ghi chưa định danh của thực thể và các thông tin liên quan.



Hình 3.8 Thêm thực thể mới

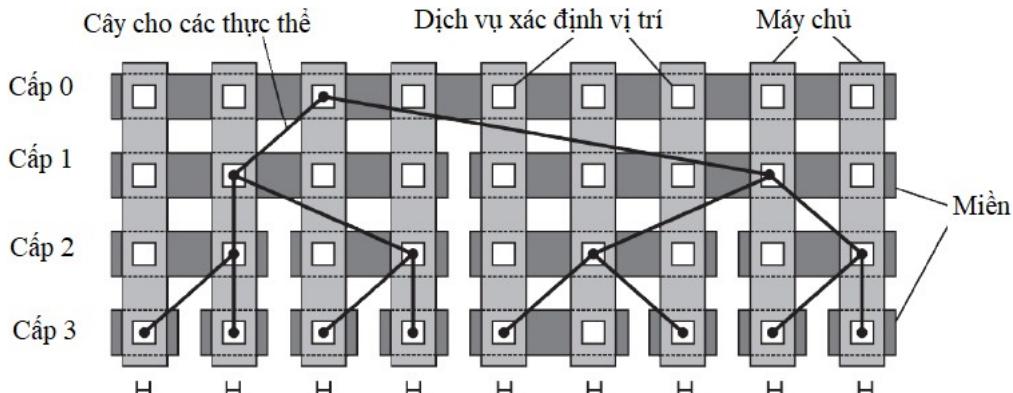
Thêm địa chỉ theo cách trên dẫn tới việc cài đặt chuỗi con trả theo kiểu từ trên xuống dưới, bắt đầu từ nút gốc cho đến nút lá có chứa bản ghi vị trí của thực thể, nếu yêu cầu gửi lên nút không thực hiện được thì thao tác thêm thực thể thất bại. Miền quản lý thực thể không nằm cùng miền truy nhập của máy khách cũng là điểm hạn chế cho thao tác tìm kiếm, nó không những mở rộng phạm vi tìm kiếm đòi hỏi các nút lá luôn phải hoạt động.

Cách thứ hai tiếp cận từ dưới lên trên, khi nhận được yêu cầu thêm thực thể, nút lá tại miền máy khách đang truy nhập sẽ thêm bản ghi chưa định danh của thực thể và các thông tin liên quan, sau đó nó mới gửi yêu cầu đến các nút cấp cao hơn để yêu cầu thêm bản ghi con trả. Như vậy nút lá cục bộ đã tạo bản ghi cho thực thể trước khi chuyển yêu cầu đến nút cha, nếu chưa cập nhật nút cha thì vẫn có thể tìm kiếm cục bộ tài miền của nút lá hiện hành.

Thao tác xóa cũng tương tự như cách thêm, trước hết phải thực hiện thao tác tìm kiếm, nếu tìm thấy bản ghi chưa thông tin về thực thể thì phải phân biệt hai trường hợp. Nếu trong bản ghi có nhiều con trả đến các nút lá thì xóa bản ghi tại nút lá và loại bỏ con trả trong bản ghi tại các nút xuất hiện trong đường dẫn kể từ nút gốc. Nếu bản ghi chỉ chứa một con trả thì có thể xóa bản ghi tại nút lá và tất cả các nút cấp cao hơn trên đường dẫn đến nút gốc. Giải pháp phân cấp đòi hỏi nút gốc phải theo dõi tất cả các định danh, do đó khó có thể đáp ứng yêu cầu về hiệu năng nếu qui mô lớn, dường như đây là một lỗ hỏng trong thiết kế.

Giả sử mỗi thực thể được gán định danh trong không gian m-bit, hệ thống gồm N máy chủ cung cấp dịch vụ tìm kiếm, mỗi máy chủ có khả năng chạy một hoặc nhiều dịch

vụ xác định vị trí, mỗi dịch vụ đại diện cho một nút ở các cấp độ khác nhau của cây. Ký hiệu  $D_k(A)$  là miền cấp k chứa địa chỉ A, giá trị  $k = 0$  biểu thị miền gốc,  $LS_k(E, A)$  là dịch vụ xác định vị trí duy nhất trong  $D_k(A)$  chịu trách nhiệm theo dõi vị trí hiện tại của thực thể E. Tập miền  $D_k = \{D_{k,1}, D_{k,2}, \dots, D_{k,N_k}\}$  biểu thị  $N_k$  miền ở cấp k, hiển nhiên  $N_0 = |D_0| = 1$ . Đối với mỗi cấp k, các máy chủ được phân vùng thành các tập con  $N_k$ , mỗi máy chủ chạy dịch vụ xác định vị trí đại diện chính xác một trong các miền  $D_{k,t}$  từ tập  $D_k$ .



Hình 3.9 Tổ chức phân cấp logic trong hệ thống máy chủ vật lý

Mỗi nút trong giải pháp phân cấp không đòi hỏi phải cài đặt riêng rẽ trên các máy tính, hình trạng của nó mang tính chất logic, hình 3.9 minh họa cách bố trí các nút trên 9 máy chủ vật lý. Hệ thống được chia thành bốn cấp, nút gốc đặt tại máy chủ  $H_3$ , Hai miền cấp 1 đặt tại các máy  $H_2$  và  $H_6$ , bốn miền cấp 2 và tám miền lá, địa chỉ của các thực thể được lưu trữ tại các máy chủ thuộc miền cấp 3. Giả sử địa chỉ của thực thể E được lưu trữ tại nút lá của máy chủ  $H_3$ , như vậy dịch vụ xác định vị trí gốc cho thực thể E cũng chạy trên máy chủ này. Cần thiết phải lựa chọn một cách thận trọng dịch vụ xác định vị trí cho thực thể E nên đặt trên máy chủ nào, nếu kết hợp nguyên tắc ưu tiên cục bộ với phân bổ đầy đủ các máy chủ mức cao hơn thì vẫn có thể đáp ứng yêu cầu về hiệu năng khi mở rộng qui mô hệ thống.

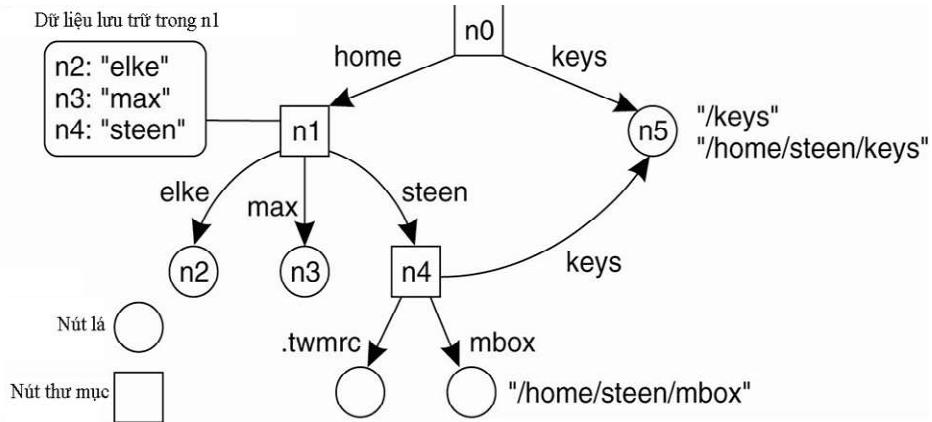
### 3.2.2 Đặt tên có cấu trúc

Đặt tên phi cấu trúc phù hợp các máy tính nhưng nó hoàn toàn không thân thiện với người sử dụng, con người cần những tên đơn giản dễ đọc và dễ nhớ. Cách đặt tên có cấu trúc được áp dụng phổ biến trên mạng Internet, cho dù đó là tên của máy tính hay tên của các tập tin. Tên có cấu trúc lại không thuận tiện cho việc xử lý trong các hệ thống máy tính, do đó cần phải có những giải pháp ánh xạ hai loại tên này.

#### 3.2.2.1 Không gian tên

Các tên thường được tổ chức thành không gian tên, không gian tên có cấu trúc có thể được thể hiện bằng đồ thị gán nhãn có hướng với nút lá và nút thư mục. Nút lá không có cạnh đi ra, nó đại diện cho thực thể được đặt tên và chứa thông tin liên quan đến thực thể, ví dụ địa chỉ để máy khách có thể truy nhập. Ngoài ra, nó có thể lưu trữ trạng thái của thực thể, ví dụ hệ thống tập tin, nút lá thực sự chứa toàn bộ dữ liệu biểu diễn tập tin.

Nút có các cạnh đi ra gọi là nút thư mục, mỗi cạnh được gán nhãn với tên cụ thể. Mỗi nút trong đồ thị tên được coi là một thực thể khác trong hệ thống phân tán, đặc biệt nó có định danh liên kết. Nút thư mục lưu trữ bảng thư mục biểu diễn thông tin của các cạnh đi ra từ nút đó dưới dạng một cặp định danh và nhãn, nếu chỉ có các cạnh đi ra mà không có cạnh đi vào gọi là nút gốc, mỗi đồ thị có thể có nhiều nút gốc.

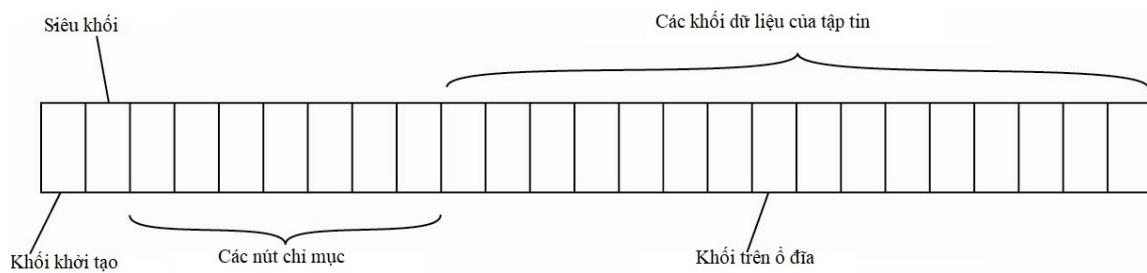


Hình 3.10 Đồ thị tên đơn gốc

Đường dẫn là chuỗi các nhãn của các cạnh liên tiếp giữa các nút, nếu nút đầu tiên là gốc thì gọi là đường dẫn tuyệt đối, ngược lại gọi là đường dẫn tương đối. Tên toàn cục là tên biểu thị cùng một thực thể trong toàn bộ hệ thống, tên cục bộ là tên mà ý nghĩa của nó phụ thuộc vào nơi đang được sử dụng, chỉ những thư mục chứa nó mới biết.

Cách mô tả đồ thị trên gần giống với hệ thống quản lý tập tin trên một máy tính, các nhãn trong đường dẫn được phân cách nhau bởi dấu gạch chéo, nếu tồn tại nhiều đường dẫn đến một nút thì nó sẽ biểu diễn bằng các tên đường dẫn khác nhau. Tên của các tài nguyên cũng được đặt giống như tên của các tập tin, cách tiếp cận này tương tự như việc cài đặt đồ thị tên cho các tài nguyên trong hệ thống phân tán.

Hầu hết không gian tên chỉ có duy nhất một nút gốc, trong nhiều trường hợp nó được tổ chức chặt chẽ dưới dạng cây, khi đó mỗi nút chỉ có một cạnh đến, ngoại trừ nút gốc, kết quả là mỗi nút có chính xác một tên đường dẫn liên kết tuyệt đối. Trong thực tế cũng có thể có những không gian tên được biểu diễn dưới dạng đồ thị không tuần hoàn có hướng, có thể đến một nút bằng nhiều đường khác nhau nhưng không được phép tạo thành vòng tuần hoàn. Ví dụ trên hình 3.10, không gian tên là đồ thị có hướng không tuần hoàn, nút lá n5 có hai cạnh đi vào đến từ các nút thư mục n0 và n4.



Hình 3.11 Tổ chức hệ thống quản lý tập tin

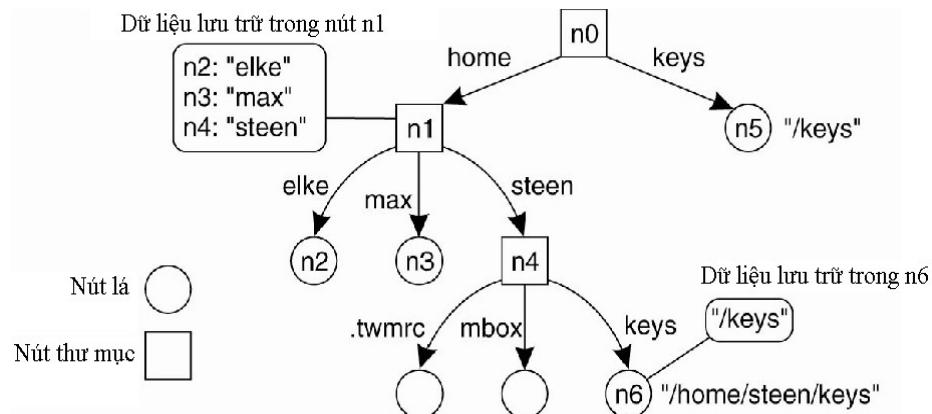
Ví dụ đồ thị đặt tên trong hệ điều hành Unix, nút thư mục đại diện cho thư mục chứa tập tin trong khi đó nút lá đại diện cho tập tin cụ thể, nút gốc là tên ổ đĩa. Hình 3.11 thể hiện cách tổ chức ổ đĩa trên máy tính, nó được chia thành nhiều khối kế tiếp nhau, bao gồm khối khởi động dùng để nạp hệ điều hành sau khi bật máy tính, siêu khối để mô tả hệ thống tập tin, các nút chỉ mục và cuối cùng là các khối chứa dữ liệu của tập tin. Các nút chỉ mục chứa thông tin phục vụ cho tìm kiếm tập tin, nó cũng có thể chứa thông tin về thuộc tính của tập tin như chủ sở hữu, thời gian tạo, thời gian cập nhật cuối cùng..., nó cũng lưu trữ thông tin về các thư mục, do đó có thể thấy số chỉ mục tương ứng với một định danh của nút trong đồ thị tên.

### 3.2.2.2 Phân giải tên

Không gian tên cung cấp cơ chế thuận tiện để lưu trữ và truy xuất thông tin về các thực thể bằng tên, quá trình tra cứu thông tin dựa trên đường dẫn của tên gọi là phân giải tên. Đầu tiên nó phải truy vấn bảng thư mục để tìm ra dữ liệu thực được lưu trữ ở đâu, quá trình tìm kiếm tiếp diễn từ nút thư mục này qua nút thư mục khác, cuối cùng sẽ trả về định danh một nút.

Phân giải tên chỉ có thể diễn ra nếu biết được bắt đầu từ nút nào trong không gian tên, gọi là cơ chế đóng, điều đó làm cho cơ chế này đôi khi khó hiểu là chúng nhất thiết phải ngầm hiểu từng phần của tên. Ví dụ nhìn vào chuỗi số “84986677028” chúng ta sẽ không hiểu đó là cái gì nếu như không biết rằng đó là số điện thoại di động, thông tin đó là đủ để bắt đầu quá trình phân giải.

Hệ thống quản lý tên trong hệ điều hành Unix phân biệt tên cục bộ và tên toàn cục, biến môi trường HOME được sử dụng để tham chiếu đến thư mục gốc của người dùng, mỗi người dùng có bản sao riêng của biến này được khởi tạo thành tên toàn cục tương ứng với thư mục chính của người dùng, cơ chế đóng gắn kết với các biến môi trường đảm bảo việc phân giải tên chính xác bằng cách tra cứu bảng dành riêng cho người dùng. Thực tế Unix coi nút chỉ mục đầu tiên của ổ đĩa là nút gốc, độ lệch thực tế của nó được tính toán từ dữ liệu lấy từ siêu khối. Ví dụ trên hình 3.10, để phân giải tên /home/steen/mbox cần phải truy nhập vào bảng thư mục của nút gốc trong đồ thị tên, bản thân nút gốc đó cũng được tra cứu trong một đồ thị khác.

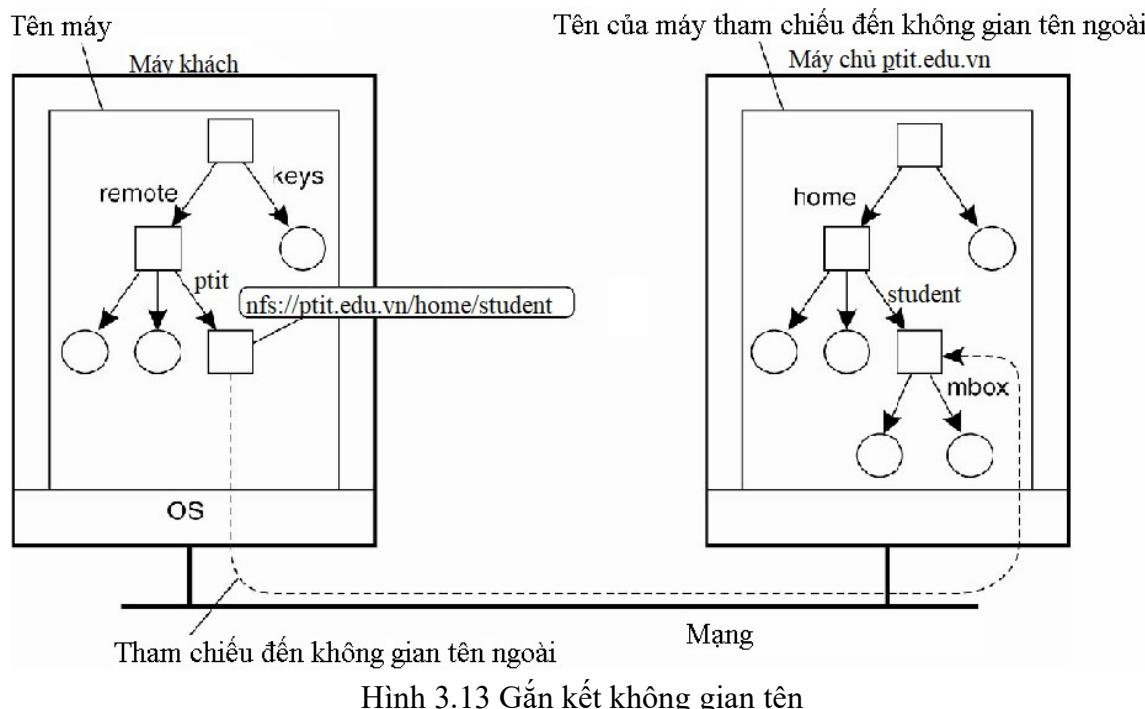


Hình 3.12 Liên kết trong đồ thị tên

Bí danh là một tên khác của cùng một thực thể, về cơ bản có hai cách triển khai trong đồ thị tên. Cách thứ nhất là cho phép nhiều đường dẫn tuyệt đối tham chiếu đến cùng một nút, ví dụ thực thể các đường dẫn /keys và /home/steen/keys đều trỏ đến nút n5. Cách thứ hai là biểu diễn một thực thể bằng nút lá nhưng nó không lưu trữ địa chỉ hay trạng thái mà lưu trữ đường dẫn tuyệt đối đến thực thể. Như vậy việc phân giải tên sẽ được thực hiện qua hai bước, bước thứ nhất lấy đường dẫn tuyệt đối được lưu trữ trong nút lá, sau đó mới phân giải đường dẫn này, cơ chế như vậy gọi là liên kết tên. Ví dụ trên hình 3.12, bước thứ nhất phân giải để tìm nút lá n6, ở đó lưu trữ đường dẫn /keys, có được giá trị này sẽ tiếp tục phân giải lần thứ hai để đến được nút n5.

Phân giải tên cũng có thể được thực hiện trên hai không gian khác nhau, để thực hiện nhiệm vụ này thì cần phải có một nút thư mục được gọi là điểm gắn kết lưu giữ định danh từ một không gian khác, nút thư mục bên phía không gian tên cần gắn kết được gọi là điểm gắn. Điểm gắn kết được coi là một nút thư mục và thông thường nó là gốc của không gian tên bên ngoài, như vậy quá trình phân giải sẽ bắt đầu bằng việc truy nhập bảng thư mục của nút này.

Để gắn kết không gian tên của các máy tính khác nhau thì cần tối thiểu tên giao thức sử dụng, tên/địa chỉ của máy chủ và tên của điểm truy nhập. Tên của giao thức truy nhập dùng để giao tiếp với máy tính của không gian tên bên ngoài, tên của máy chủ dùng để xác định địa chỉ của máy tính quản lý không gian tên bên ngoài. Tên của điểm truy nhập thể hiện nút gắn kết của không gian tên bên ngoài, nó sẽ được phân giải thành định danh của nút bắt đầu trong không gian tên bên ngoài.



Hình 3.13 Gắn kết không gian tên

Tên của điểm gắn kết sẽ được máy chủ của không gian tên bên ngoài phân giải, nhưng cũng cần không gian tên cho giao thức truy nhập và tên của máy chủ, có thể sử dụng đường liên kết URL để đại diện cho cả ba thông tin này. Giao thức NFS cho phép

máy khách có thể truy nhập đến các tập tin cài đặt trên máy chủ, ví dụ đường liên kết nfs://ptit.edu.vn/home/student cho biết giao thức sử dụng là NFS, tên máy chủ là ptit.edu.vn và điểm truy nhập là /home/student. Tên đầu tiên, trong trường hợp này là nfs, đã được cộng đồng công nghệ thông tin trên thế giới qui định giao thức sử dụng, nó hiểu rằng cần phải triển khai giao thức NFS, tên máy chủ ptit.edu.vn sẽ được phân giải bằng dịch vụ tên miền, như vậy chỉ còn /home/student sẽ được máy chủ của không gian tên bên ngoài phân giải.

Tổ chức tập tin trên máy khách do người sử dụng định nghĩa, nó chứa thư mục con /remote gồm các điểm gắn kết với không gian tên bên ngoài, hình 3.13 thể hiện gắn kết không gian tên của máy chủ, thư mục chính của PTIT sẽ sử dụng nút /ptit để lưu trữ đường liên kết nfs://ptit.edu.vn/home/student. Tên /remote/ptit/mbox được phân giải trên máy khách cho đến khi gặp nút /remote/ptit, sau đó tiếp tục bằng cách trả về đường liên kết nfs://ptit.edu.vn/home/student, máy khách sẽ sử dụng giao thức NFS để liên lạc với máy chủ để có thể truy nhập vào thư mục /home/student, quá trình phân giải tên sẽ tiếp tục trên máy chủ bằng cách đọc tập tin mbox.

Việc gắn kết hệ thống tập tin từ xa như đã mô tả trên cho phép máy khách có thể thao tác với các tập tin như thể chúng đang ở trên máy tính của mình, người sử dụng hoàn toàn không biết thực tế đã lấy những dữ liệu đó từ máy chủ ngoại trừ thời gian có thể chậm hơn đôi chút, đường như không gian tên bắt đầu từ điểm gắn kết đã hòa nhập vào không gian tên trên máy khách.

### 3.2.2.3 Cài đặt không gian tên

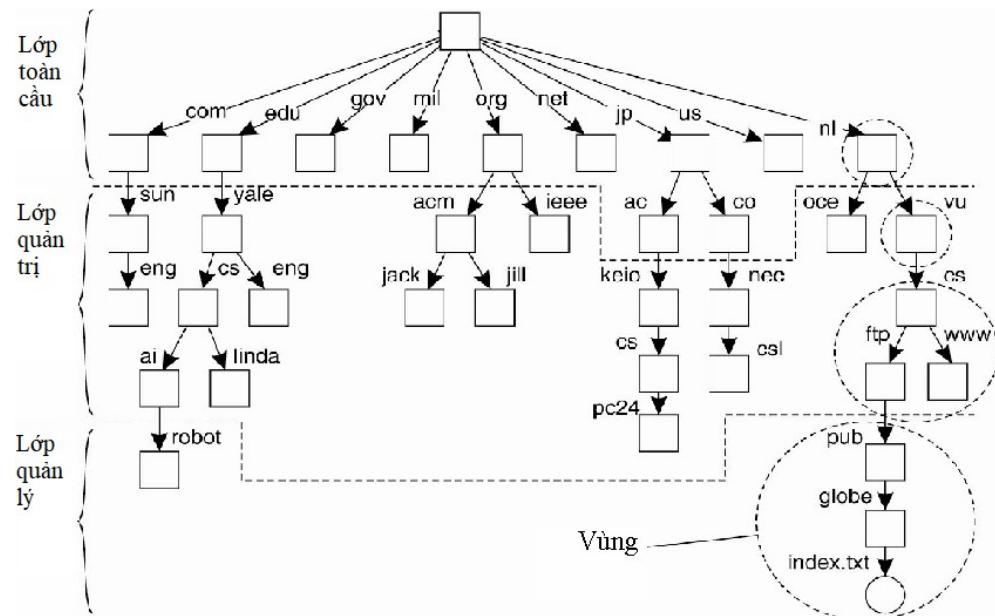
Không gian tên là trái tim của dịch vụ đặt tên, dịch vụ này được cài đặt trên máy chủ định danh, nếu ở qui mô nhỏ thì chỉ cần một máy chủ, với qui mô lớn thì cần cài đặt trên nhiều máy chủ. Không gian tên cho hệ thống qui mô lớn như mạng Internet được tổ chức theo dạng phân cấp, để tổ chức một cách hiệu quả người ta chia không gian tên thành các lớp toàn cầu, quản trị và quản lý.

Hình 3.14 thể hiện cách phân lớp trong không gian tên miền trên mạng Internet, chúng được chia thành các vùng không trùng lặp do một máy chủ quản lý. Xét về tính sẵn sàng và hiệu năng thì mỗi máy chủ phân giải tên miền sẽ phải đáp ứng các yêu cầu khác nhau, máy chủ ở lớp toàn cầu đòi hỏi tính sẵn sàng cao, nếu nó bị lỗi thì một lượng lớn tên miền sẽ không thể phân giải, bảng 3.1 tóm tắt đặc điểm của các lớp này.

Lớp toàn cầu được hình thành bởi các nút mức cao nhất, nghĩa là chỉ gồm nút gốc và các nút con của nó, tính ổn định của chúng khá cao, nghĩa là bảng thư mục của các nút này ít thay đổi, các nút như vậy thường đại diện cho các tổ chức. Lớp quản trị được hình thành từ các nút thư mục do cùng một đơn vị quản lý, chúng đại diện cho các nhóm thực thể thuộc cùng một tổ chức, các nút ít khi thay đổi nhưng tính ổn định của nó thấp hơn lớp toàn cục. Cuối cùng là lớp quản lý bao gồm các nút có thể thay đổi thường xuyên, đó là các nút đại diện cho dịch vụ nào đó, khác với hai lớp trên, người dùng đầu cuối cũng có thể quản trị các nút thư mục thuộc lớp này.

Vấn đề hiệu năng có thể giải quyết bằng cách sử dụng vùng đệm, khi nhận được yêu cầu phân giải, máy tính sẽ tìm kiếm trong vùng đệm trước khi truy vấn cơ sở dữ liệu.

Như vậy chỉ là lần đầu tiên truy nhập máy khách mới phải sử dụng dịch vụ phân giải tên, do đó các máy chủ lớp toàn cầu không cần phải đáp ứng nhanh các yêu cầu tra cứu, tuy nhiên phải cung cấp đủ băng thông phục vụ cho qui mô hàng triệu yêu cầu gửi đến cùng một lúc. Để đáp ứng yêu cầu về tính sẵn sàng và hiệu năng thì các máy chủ lớp toàn cầu phải được nhân bản, do các tên miền không đòi hỏi cập nhật ngay lập tức cho nên sẽ vẫn đáp ứng các yêu cầu về tính nhất quán cho các bản sao.



Hình 3.14 Tô chúc tên miền Internet

Tính sẵn sàng của các máy chủ ở lớp quản trị chủ yếu quan trọng đối với máy khách trong cùng một đơn vị với máy chủ phân giải tên, nếu bị lỗi thì không thể tra cứu tài nguyên trong tổ chức đó. Yêu cầu về hiệu năng cũng tương tự như ở lớp toàn cầu, các nút nói chung ít thay đổi, tuy nhiên nó đòi hỏi thời gian đáp ứng chỉ vài mili giây, như vậy các máy chủ lớp này cũng cần phải sử dụng vùng đệm, thời gian cập nhật thông tin cũng phải nhanh hơn ở lớp toàn cầu, do đó phải sử dụng các máy chủ cấu hình mạnh kết hợp với các chiến lược nhân bản nhưng vẫn phải đảm bảo tính nhất quán. Các yêu cầu về tính sẵn sàng của các máy chủ ở lớp quản lý thường ít khắt khe hơn nhưng hiệu năng lại là vấn đề rất quan trọng, thông tin các nút thay đổi thường xuyên đòi hỏi phải được cập nhật ngay lập tức.

Việc cài đặt các máy chủ phân giải tên tại lớp toàn cầu và lớp quản trị là phức tạp nhất, khó khăn không những phải đáp ứng các yêu cầu về tính sẵn sàng và hiệu năng mà còn ở khía cạnh nhất quán trong nhân bản. Vấn đề sẽ trở nên khó khăn hơn khi vùng đệm và các bản sao được trải rộng trên qui mô lớn, điều này có thể gây nên sự chậm trễ truyền thông trong quá trình tra cứu.

Mỗi thực thể đều có tên và địa chỉ tương ứng, việc ánh xạ từ tên đến địa chỉ của thực thể được thực hiện theo mô hình một lớp hoặc hai lớp. Mô hình một lớp chỉ có một mức ánh xạ giữa tên và thực thể, mỗi lần thực thể thay đổi vị trí, ánh xạ cần phải được thay đổi theo. Mô hình hai lớp phân biệt tên và địa chỉ nhờ định danh thực thể, nó gồm

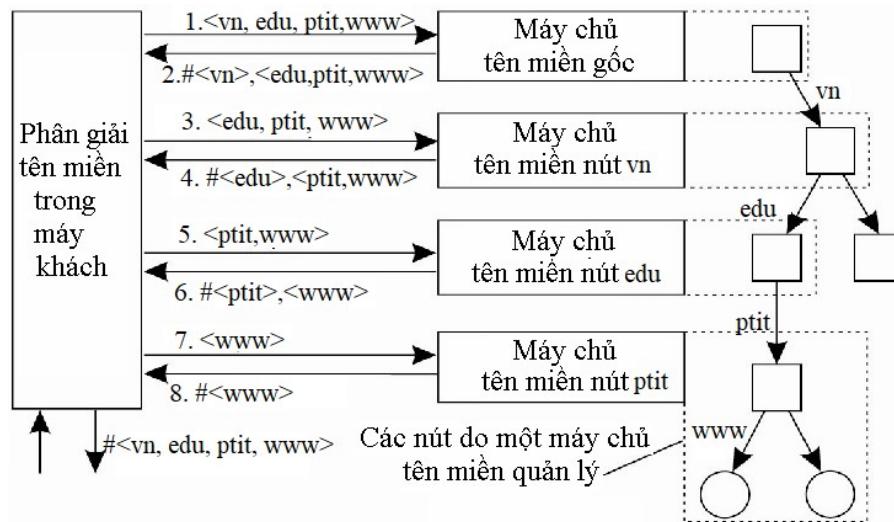
quá trình tìm định danh thực thể tương ứng từ tên của thực thể được thực hiện bằng dịch vụ tên và quá trình xác định vị trí của thực thể từ định danh được thực hiện bởi dịch vụ định vị.

**Bảng 3.1** So sánh đặc điểm máy chủ tên miền theo các mức

Mục	Toàn cầu	Quản trị	Quản lý
Phạm vi địa lý	Toàn thế giới	Tổ chức	Đơn vị
Số lượng nút	Ít	Nhiều	Rất nhiều
Thời gian đáp ứng	Giây	Mili giây	Ngay lập tức
Cập nhật lan truyền	Ít	Ngay lập tức	Ngay lập tức
Số lượng bản sao	Nhiều	Ít	Không có
Lưu yêu cầu máy khách	Có	Có	Không cần

Không gian tên được phân bố trên nhiều máy chủ có ảnh hưởng đến việc phân giải tên, có thể sử dụng giải pháp tương tác hoặc đệ qui. Phân giải tên tương tác thực hiện bằng cách truyền và nhận qua lại giữa máy khách và các máy chủ quản lý tên ở các mức khác nhau. Các máy chủ quản lý tên không trao đổi trực tiếp với nhau, mỗi máy chủ chỉ phân giải nhãn tương ứng với lớp để xác định địa chỉ của máy chủ tiếp theo, kết quả trả lại cho máy khách là địa chỉ của máy chủ quản lý tên kế tiếp và việc liên kết với máy chủ tiếp theo là do máy khách đảm nhiệm. Phân giải tên đệ quy thực hiện bằng cách mỗi máy chủ quản lý tên sẽ gửi kết quả đến máy chủ quản lý tên tiếp theo mà nó tìm thấy và cứ như vậy cho đến khi hoàn thành phân giải toàn bộ đường dẫn.

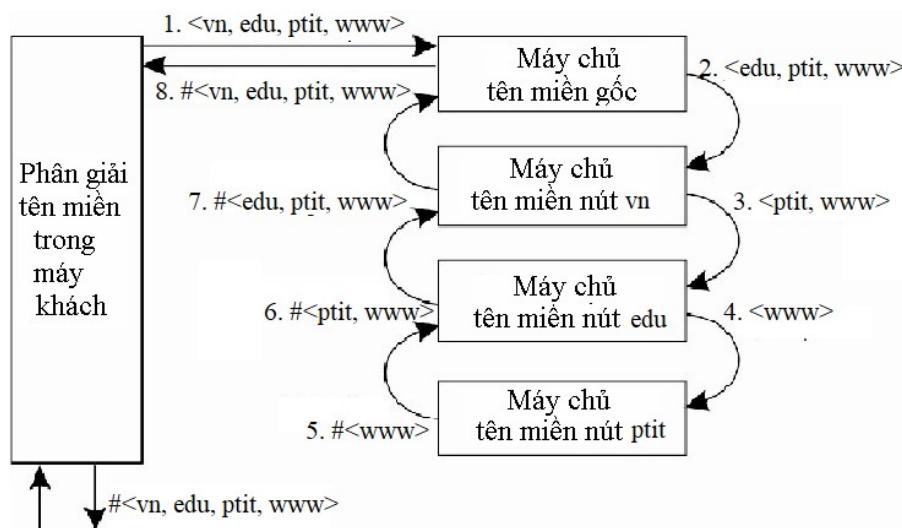
Khi người sử dụng truy nhập trang <http://www.ptit.edu.vn/index.html>, nghĩa là sử dụng giao thức http để lấy dữ liệu từ tập tin index.html trên dịch vụ web (www) đặt tại máy chủ “ptit.edu.vn”, rõ ràng mạng máy tính không thể định tuyến dựa trên tên thân thiện này, nó cần phải chuyển đổi sang địa chỉ IP. Như vậy, cần phải phân giải tên miền www.ptit.edu.vn, theo lý thuyết đã trình bày trên, phân tích từ bên phải qua bên trái đường dẫn tuyệt đối sẽ là <vn, edu, ptit, www>. Để hiểu rõ hơn về qui trình phân giải tên miền, giả thiết không có máy tính nào chứa bản ghi tên miền này trong bộ nhớ đệm, do đó nó phải đi qua tất cả các nút trong đồ thị tên để lấy địa chỉ của tên miền.



Hình 3.15 Phân giải tên miền tương tác

Hình 3.15 thể hiện các bước thực hiện phân giải tên miền theo giải pháp tương tác, tiến trình máy khách sẽ gửi thông điệp <vn, edu, ptit, www> đến máy chủ cấp cao nhất, máy chủ sẽ trả về cho máy khách địa chỉ tham chiếu đến máy chủ quản lý nút vn. Nhận được địa chỉ này, máy khách gửi thông điệp đến máy chủ quản lý nút vn để yêu cầu phân giải <edu, ptit, www>, máy chủ quản lý nút vn trả về địa chỉ tham chiếu đến máy chủ quản lý nút edu. Máy khách tiếp tục gửi thông điệp yêu cầu phân giải <ptit, www> đến máy chủ quản lý nút edu, máy chủ này lại trả về địa chỉ tham chiếu đến máy chủ quản lý nút ptit. Máy khách tiếp tục gửi đến máy chủ quản lý nút ptit, yêu cầu cho biết địa chỉ của dịch vụ web, máy chủ trả về địa chỉ của tên miền www.ptit.edu.vn. Như vậy tiến trình phân giải đã hoàn thành, nó trả về cho tiến trình ứng dụng địa chỉ của tên miền www.ptit.edu.vn để có thể truy vấn nội dung của trang index.html.

Nếu phân giải tên miền sử dụng giải pháp đệ qui, máy khách gửi yêu cầu địa chỉ của tên miền <vn, edu, ptit, www> đến máy chủ tên miền gốc, máy chủ tên miền gốc sẽ gửi yêu cầu đến máy chủ quản lý nút vn, yêu cầu tiếp tục được chuyển đến máy chủ quản lý nút edu và sau đó chuyển đến máy chủ quản lý nút ptit. Máy chủ quản lý nút tìm kiếm tại nút lá www để lấy địa chỉ của tên miền, kết quả lần lượt chuyển ngược lại cho đến máy chủ tên miền gốc và trả về cho máy khách, khi đó tiến trình phân giải tên miền sẽ trả về cho tiến trình ứng dụng đã yêu cầu.



Hình 3.16 Phân giải tên miền đệ qui

Ưu điểm của giải pháp phân giải tên miền tương tác là sự phân tải được dải đều cho các máy chủ, tuy nhiên nhược điểm của nó là hiệu quả sử dụng tính năng lưu trữ địa chỉ trong bộ nhớ đệm tương đối thấp. Ngoại trừ máy chủ tại nút lá, tất cả các máy chủ khác đều không biết địa chỉ của tên miền nên không thể lưu trữ vào vùng đệm để giảm thời gian truy nhập tìm kiếm, qua đó tăng hiệu năng xử lý.

Ưu điểm của phương pháp phân giải đệ qui nằm ở hiệu quả sử dụng bộ nhớ đệm tốt hơn và có thể giảm lượng thông tin trao đổi trên mạng, địa chỉ của các máy chủ định danh cấp thấp hơn lần lượt được lưu trong bộ nhớ đệm, điều đó sẽ cải thiện đáng kể hiệu năng cho hệ thống. Khi nhận được yêu cầu từ máy khách, máy chủ tên miền gốc chỉ cần

truy nhập vào bộ nhớ, thời gian chỉ mất vài nano giây, nếu không tìm thấy trong vùng đệm thì mới phải chuyển yêu cầu đến máy chủ khác.

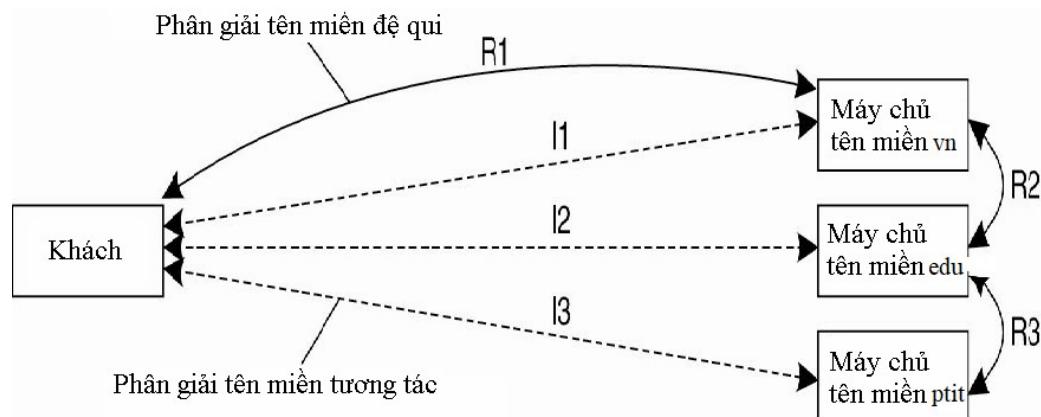
**Bảng 3.2** Khả năng lưu dữ liệu vào vùng đệm trên các máy chủ

Máy chủ	Yêu cầu Phân giải	Tìm kiếm	Gửi cấp thấp hơn	Nhận và lưu bộ nhớ đệm	Trả kết quả cho máy yêu cầu
ptit	<www>	#<www>	-	-	#<www>
edu	<ptit, www>	#<ptit>	<www>	#<www>	#<ptit> #<ptit, www>
vn	<edu, ptit, www>	#<edu>	<ptit, www>	#<www> #<ptit, www>	#<edu> #<edu, ptit> #<edu, ptit, www>
Gốc	<vn, edu, ptit, www>	#<vn>	<edu, ptit, www>	#<vn> #<vn, edu> #<vn, edu, ptit>	#<vn> #<vn, edu> #<vn, edu, ptit> #<vn, edu, ptit, www>

Các nút mức toàn cầu và mức quản trị thường ít khi thay đổi, máy chủ gốc sử dụng bộ nhớ đệm hiệu quả hơn so với phương pháp tương tác, các máy chủ cấp thấp hơn cũng quản lý bộ nhớ đệm theo cách tương tự. Kết quả tra cứu trung gian cũng có thể được lưu vào bộ nhớ đệm, bảng 3.2 tóm tắt kết quả tổng thể quá trình phân giải và lưu kết quả trong bộ nhớ đệm.

Lợi ích chính của phương pháp lưu trữ này là các hoạt động tra cứu có thể được xử lý khá hiệu quả, máy chủ cấp trên có thể liên lạc với bất kỳ máy chủ nào cấp dưới mà không cần phải đi tuần tự theo các nhánh của cây. Lợi ích thứ hai là chi phí về truyền thông, các gói tin có thể sẽ đi qua ít thiết bị định tuyến hơn nếu các máy chủ phân giải tên miền được lắp đặt trong cùng khu vực.

Nhược điểm của giải pháp đệ qui nằm ở việc yêu cầu hiệu năng cao đối với các máy chủ phân giải tên miền, mỗi máy chủ sẽ phải đảm nhiệm việc phân giải hoàn chỉnh đối với đường dẫn đã yêu cầu. Tải cao nhất xảy ra tại máy chủ tên miền gốc, yêu cầu của máy khách đều tập trung về máy chủ này, ngay cả khi sử dụng bộ đệm thì cũng khó có thể đảm bảo hiệu năng phục vụ cho hàng tỉ máy khách, hơn nữa việc lưu trữ tất cả các tên miền trong bộ nhớ cũng là điểm không hợp lý.



Hình 3.17 So sánh hai giải pháp phân giải tên miền

Hình 3.17 thể hiện qui trình thực hiện của hai giải pháp phân giải tên miền, giải pháp đệ qui được thể hiện bằng đường liên kết và giải pháp tương tác được thể hiện bằng đường đứt nét. Đối với phân giải tên miền tương tác, bộ nhớ đệm bị giới hạn cho tiến trình phân giải, nếu nhiều máy tính trong mạng cục bộ đều có nhu cầu phân giải một tên miền thì quá trình thực hiện sẽ như nhau trên tất cả các máy, cho dù đó là những yêu cầu giống nhau. Cho dù sử dụng giải pháp nào thì tất cả máy khách đều gửi yêu cầu đến các máy chủ tên miền gốc, điều này không khả thi khi trên mạng Internet có hàng tỉ máy tính.

Phân giải tên miền trên mạng Internet kết hợp cả hai giải pháp đệ qui và tương tác, một máy chủ trung gian tiếp nhận yêu cầu của máy khách, máy khách sẽ không liên lạc trực tiếp đến các máy chủ tên miền. Khi nhận được yêu cầu phân giải tên miền từ các máy khách, máy chủ phân giải tên miền mặc định sẽ tìm kiếm cục bộ, nghĩa là truy vấn vùng đệm hoặc cơ sở dữ liệu của nó, nếu tìm thấy bản ghi tên miền hợp lệ thì trả ngay địa chỉ cho máy khách, nếu không nó thực hiện giải pháp tương tác để phân giải tên miền. Như vậy đoạn từ máy khách đến máy chủ phân giải tên miền đầu tiên hoạt động dựa trên giải pháp đệ qui, phần còn lại hoạt động theo cơ chế tương tác.

Trên mỗi giao diện mạng của máy tính đều phải thiết lập địa chỉ của máy chủ phân giải tên miền mặc định, đó là địa chỉ để máy tính gửi yêu cầu phân giải tên miền, nếu không thiết lập tham số này thì sẽ không thể sử dụng dịch vụ. Có rất nhiều địa chỉ máy chủ giải tên miền, nên sử dụng địa chỉ thuộc nhà cung cấp dịch vụ Internet để giảm thiểu kinh phí truyền vật lý và tận dụng khả năng lưu tên miền trong vùng đệm của máy chủ.

Quá trình phân giải tên bắt đầu từ máy khách, nó kiểm tra xem tên miền đã có trong vùng đệm hay chưa, nếu chưa có thì sẽ gửi yêu cầu đến máy chủ phân giải tên miền mặc định. Sau khi nhận được yêu cầu của máy khách, máy chủ phân giải tên miền mặc định tìm trong vùng đệm nếu tìm thấy thì trả về ngay địa chỉ. Nếu không tìm thấy, máy chủ tên miền mặc định sử dụng phương pháp phân giải tên miền tương tác, yêu cầu được chuyển đến máy chủ cấp cao nhất và quá trình tiếp tục như đã trình bày trên.

Nhận được địa chỉ tương ứng với tên miền, máy chủ tên miền mặc định không những trả kết quả cho máy khách, nó còn lưu trong vùng đệm để phục vụ cho những yêu cầu tương tự từ máy tính khác. Số lượng người sử dụng trong phạm vi nhà cung cấp dịch vụ Internet thường nhỏ hơn rất nhiều so với số lượng người sử dụng trên mạng Internet, hơn nữa họ thường truy nhập vào những tên miền giống nhau, do đó số yêu cầu gửi lên máy chủ tên miền mức toàn cầu sẽ giảm đi đáng kể. Dữ liệu trong vùng đệm có thể không phản ánh thực tế nếu tên miền thay đổi địa chỉ, do đó các bản ghi này sẽ được thiết lập thời gian quá hạn, tên miền cũng thường xuyên được đồng bộ giữa các máy chủ.

#### 3.2.2.4 Hệ thống tên miền trên Internet

Một trong những dịch vụ đặt tên phán tán lớn nhất hiện nay là hệ thống tên miền của mạng Internet, hệ thống này được phát minh vào năm 1984 cho phép thiết lập tương ứng giữa địa chỉ IP và tên miền. Hệ thống tên miền là một hệ thống đặt tên theo thứ tự cho máy tính, dịch vụ hoặc bất kỳ nguồn lực nào tham gia vào mạng Internet. Hệ thống phân giải tên miền liên kết nhiều thông tin đa dạng với tên miền được gán cho những

thành phần tham gia và chuyển tên miền thân thiện với con người vào định danh của tên đó. Ví dụ tên miền www.ptit.edu.vn dễ nhớ đối với con người nhưng lại không áp dụng được cho việc định tuyến, trước khi tham gia định tuyến tên miền này sẽ phải được chuyển đổi thành địa chỉ của máy chủ đang cài đặt dịch vụ web cho trang này.

Tên miền nên được đặt đơn giản và có tính chất gợi nhớ với mục đích và phạm vi hoạt động của chủ sở hữu, mỗi nhãn tên miền tối đa 63 ký tự, các nhãn phân tách nhau bằng dấu chấm. Tên miền được đặt bằng các ký tự (a-z A-Z 0-9) không phân biệt chữ thường hay chữ hoa và ký tự gạch ngang “-” nhưng không được bắt đầu hoặc kết thúc bằng ký tự này, một tên miền đầy đủ có chiều dài không vượt quá 253 ký tự.

Hệ thống tên miền sẽ phân chia trách nhiệm gán tên miền và lập bản đồ ánh xạ tên với địa chỉ IP bằng cách định rõ những máy chủ có thẩm quyền cho mỗi loại tên miền. Những máy chủ có thẩm quyền được phân công chịu trách nhiệm đối với tên miền riêng của họ và lần lượt có thể chỉ định tên máy chủ khác cho các tên miền phụ. Ban đầu, tất cả các tên miền và địa chỉ IP tương ứng được lưu giữ trong tập tin tại trung tâm thông tin mạng của Hoa Kỳ. Tuy nhiên khi hệ thống Internet phát triển, việc lưu giữ thông tin trong một tập không thể đáp ứng nhu cầu phân phối và cập nhật, vì vậy hệ thống tên miền đã phát triển dưới dạng các cơ sở dữ liệu phân tán, mỗi cơ sở dữ liệu sẽ quản lý một phần trong hệ thống tên miền.

Hệ thống tên miền Internet được phân bố theo cấu trúc hình cây, tên miền cấp cao nhất là tên miền gốc được thể hiện bằng dấu chấm. Dưới tên miền gốc có hai loại tên miền là tên miền cấp cao dùng chung và tên miền cấp cao quốc gia như .vn, .jp, .kr, ..... Tên miền cấp cao dùng chung hiện nay do hiệp hội Internet gán tên và địa chỉ ICANN quản lý, tham khảo tại địa chỉ <https://www.icann.org/resources/pages/listing-2012-02-25-en>, bảng 3.3 phân loại tên miền dựa trên lĩnh vực hoạt động.

**Bảng 3.3** Tên miền cho các lĩnh vực hoạt động

Loại	Lĩnh vực hoạt động
COM, BIZ	Các tổ chức, cá nhân hoạt động thương mại.
EDU	Các tổ chức, cá nhân hoạt động trong lĩnh vực giáo dục, đào tạo
GOV	Các cơ quan, tổ chức nhà nước ở trung ương và địa phương
NET	Các tổ chức, cá nhân hoạt động trong lĩnh vực thiết lập và cung cấp dịch vụ mạng
ORG	Các tổ chức hoạt động trong lĩnh vực chính trị, văn hoá, xã hội
INT	Các tổ chức quốc tế
AC	Các tổ chức, cá nhân hoạt động trong lĩnh vực nghiên cứu
PRO	Các tổ chức, cá nhân hoạt động trong lĩnh vực chuyên ngành cao
INFO	Các tổ chức, cá nhân hoạt động trong lĩnh vực cung cấp thông tin
HEALTH	Các tổ chức, cá nhân hoạt động trong lĩnh vực dược, y tế
NAME	Tên riêng của cá nhân tham gia hoạt động Internet

Hệ thống tên miền được sắp xếp theo cấu trúc phân cấp, mức trên cùng được gọi là gốc và ký hiệu là “.”, ICANN đảm nhiệm vai trò quản lý hệ thống tên miền mức cao nhất do đó nó có quyền cấp phát các tên miền dưới mức này. Giả sử có tên miền www.ptit.edu.vn sẽ được nhận dạng từ phải qua trái, mục đầu tiên của tên miền này là vn nghĩa là thuộc về Việt Nam, tiếp theo đó edu nghĩa là đơn vị giáo dục, ptit là tên chủ sở hữu yêu cầu và www nghĩa là trang tin điện tử.

Theo cấu trúc và cách phân chia trong không gian tên miền đã trình bày ở trên, người dùng khi gặp một tên miền hoàn toàn có thể biết tổ chức quản lý tên miền này thuộc lĩnh vực gì, hay tên miền này do quốc gia nào quản lý... Tên miền tận cùng bằng .vn do đó tên miền này thuộc vùng do Việt Nam quản lý, cụ thể là Trung tâm Internet Việt Nam VNNIC, tiếp sau là tên miền cấp 2 edu, nghĩa là tên miền này được phân cho tổ chức hoạt động trong lĩnh vực giáo dục.

Máy chủ tên miền là máy chủ cung cấp dịch vụ phân giải tên miền, giống như cách phân cấp của hệ thống tên miền, tương ứng với mỗi cấp và mỗi loại tên miền có máy chủ tên miền phục vụ tên miền ở cấp và loại tên miền. Máy chủ tên miền gốc cung cấp dịch vụ phân giải tên miền ở mức toàn cầu, ở mức quốc gia sẽ có máy chủ tên miền quản lý tên miền mức quốc gia.

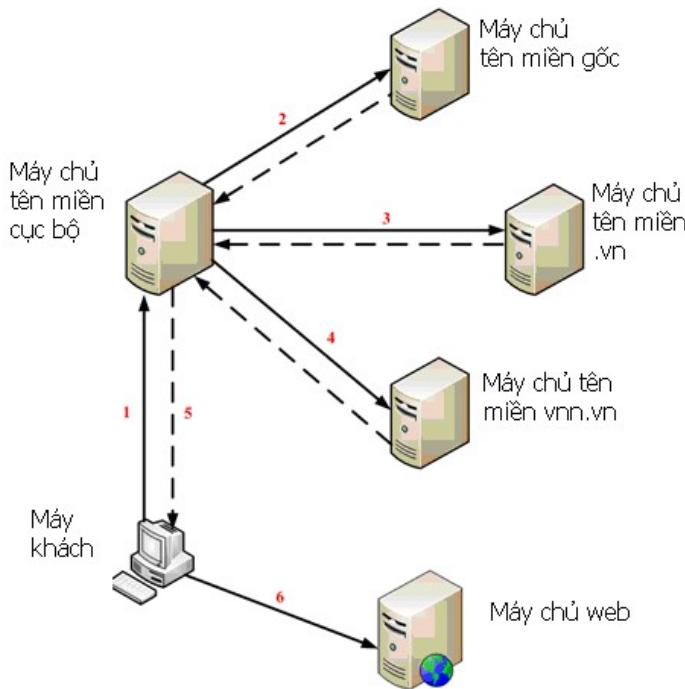
Hệ thống tên miền định nghĩa hai kiểu máy chủ tên miền là máy chủ tên miền chính và máy chủ tên miền phụ. Máy chủ tên miền chính là máy chủ tên miền lấy dữ liệu cho các vùng của nó từ các tập tin có sẵn trên máy, máy chủ tên miền phụ là máy chủ tên miền lấy dữ liệu cho các khu vực của nó từ một máy chủ tên miền chính khác. Khi máy chủ phụ khởi động nó sẽ kết nối đến máy chủ chính để lấy dữ liệu từ máy này về cho các khu vực mà nó quản lý, quá trình lấy dữ liệu từ máy chính về máy phụ được gọi là chuyển vùng.

Máy chủ tên miền ở mức cao nhất là máy chủ tên miền chứa các thông tin để tìm kiếm các máy chủ tên miền ủy quyền cho các tên miền thuộc mức cao nhất. Máy chủ gốc có thể đưa ra các truy vấn để tìm kiếm tối thiểu là các thông tin về địa chỉ của các máy chủ tên miền ủy quyền thuộc mức cao nhất chứa tên miền muốn tìm. Sau đó, các máy chủ tên miền ở mức cao nhất có thể cung cấp các thông tin về địa chỉ của máy chủ ủy quyền cho tên miền ở mức thứ cấp chứa tên miền muốn tìm, quá trình tìm kiếm tiếp tục cho đến khi chỉ ra được máy chủ tên miền ủy quyền cho tên miền muốn tìm.

Máy chủ tên miền đóng vai trò quan trọng trong việc tìm kiếm một tên miền bất kỳ trên không gian tên miền, quá trình tìm kiếm tên miền luôn được bắt đầu bằng các truy vấn gửi cho máy chủ gốc, nếu như các máy chủ tên miền gốc không hoạt động, quá trình tìm kiếm tên miền sẽ không được thực hiện. Để tránh điều này không xảy ra, trên mạng Internet hiện tại có 13 hệ thống máy chủ tên miền gốc, các máy chủ tên miền này nói chung và ngay trong cùng một hệ thống cũng được đặt tại nhiều vị trí khác nhau trên mạng Internet.

Giả sử người sử dụng muốn truy nhập vào trang tin điện tử có tên miền là www.vnn.vn, hình 3.18 thể hiện các bước thực hiện quá trình phân giải tên miền. Trước hết chương trình trên máy người sử dụng gửi yêu cầu tìm kiếm địa chỉ IP ứng với tên

miền www.vnn.vn đến máy chủ tên miền cục bộ, địa chỉ của máy chủ này đã được thiết lập trong giao diện mạng. Máy chủ tên miền cục bộ kiểm tra trong cơ sở dữ liệu của nó có chứa tên miền mà người sử dụng yêu cầu không, nếu có thì trả lại địa chỉ IP cho máy khách, nếu không thì gửi yêu cầu lên máy chủ tên miền gốc, tức máy chủ mức toàn cầu. Máy chủ tên miền gốc sẽ trả về địa chỉ của máy chủ tên miền .VN, máy chủ tên miền cục bộ sẽ gửi yêu cầu đến máy chủ tên miền .VN để yêu cầu phân giải tên miền www.vnn.vn, máy chủ tên miền .VN sẽ trả lại địa chỉ của máy chủ quản lý tên miền vnn.vn.



Hình 3.18 Các bước thực hiện trong phân giải tên miền

Máy chủ tên miền cục bộ sẽ hỏi máy chủ tên miền vnn.vn này địa chỉ IP của tên miền www.vnn.vn, máy chủ quản lý tên miền vnn.vn trả về địa chỉ IP của tên miền www.vnn.vn cho máy chủ tên miền cục bộ. Máy chủ tên miền cục bộ chuyển thông tin tìm được đến máy tính của người sử dụng, trình duyệt trên máy tính của người sử dụng dùng địa chỉ IP này để kết nối đến máy chủ cài đặt trang www.vnn.vn. Trong quá trình trên, máy chủ tên miền cục bộ sẽ lưu địa chỉ của máy chủ cài đặt trang www.vnn.vn vào cơ sở dữ liệu để phục vụ cho những lần truy nhập tiếp theo mà không cần phải tham chiếu đến máy chủ tên miền gốc. Điều đó cho thấy người sử dụng nên chọn máy chủ DNS sao cho chi phí thấp nhất, chi phí ở đây được hiểu là độ trễ và số lượng thiết bị định tuyến trên kênh truyền.

### 3.2.3 Đặt tên dựa trên thuộc tính

Đặt tên phi cấu trúc hay đặt tên có cấu trúc đều bảo đảm tham chiếu đến một thực thể duy nhất một cách độc lập với vị trí của thực thể đó. Trong thực tế, hai tiêu chí thân thiện với người sử dụng và độc lập vị trí là chưa đủ, người sử dụng muốn tìm kiếm các thực thể dựa trên các thuộc tính của chúng, mô tả càng nhiều thuộc tính càng tốt. Có nhiều cách mô tả thực thể nhưng cách phổ biến là mô tả thực thể theo cặp thuộc tính và

giá trị, phương pháp này gọi là đặt tên theo thuộc tính. Một thực thể bao gồm nhiều thuộc tính liên quan với nhau, mỗi thuộc tính thể hiện đặc điểm của thực thể, bằng cách xác định giá trị của các thuộc tính sẽ hạn chế được tập các thực thể cần tìm.

Nếu tên được phân ra thành nhiều thuộc tính thì có thể lợi dụng thứ tự các thuộc tính, ví dụ người sử dụng với thuộc tính tên là <A>, thuộc tính tổ chức là <FOT> và thuộc tính quốc gia là <VN> thì có thể tạo ra thành một thuộc tính tổng hợp là <A.FOT.VN>. Tổng quát hơn, khi không cho thứ tự các thuộc tính thì có thể xác định thực thể nhờ tập hợp tất cả các thuộc tính, ví dụ U=A, C=FOT, O=VN, trong đó U, C và O đại diện cho các thuộc tính tên, cơ quan và quốc gia. Những tên này được xác định dựa vào việc ghép nối tiếp các thuộc tính dựa vào tên cấu trúc phân cấp hoặc dựa vào chỉ tập hợp các thuộc tính với cấu trúc tùy ý. Việc phân chia thuộc tính của thực thể có thể căn cứ vào tính chất vật lý, theo tổ chức hoặc theo chức năng.

### 3.2.3.1 Dịch vụ thư mục

Hệ thống đặt tên dựa trên thuộc tính còn gọi là dịch vụ thư mục, mỗi thực thể có tập các thuộc tính liên quan dùng để tìm kiếm. Xác định các thuộc tính cho thực thể không phải là vấn đề đơn giản, trong nhiều trường hợp cần phải thực hiện thủ công. Dịch vụ thư mục tương tự với một từ điển, nó cho phép tra cứu tên và thông tin liên kết với tên đó. Mỗi từ có thể hiểu theo nhiều nghĩa, do đó một tên có thể được kết hợp với nhiều thông tin khác nhau.

Thực tế cho thấy ngay cả khi có sự đồng thuận về tập các thuộc tính thì vẫn chưa chắc đạt được tính nhất quán cho một nhóm người dùng. Để giảm bớt sự phức tạp này, các nghiên cứu đã được tiến hành nhằm thống nhất cách mô tả các thực thể, một khung mô tả tài nguyên đã được sử dụng trong các hệ thống phân tán, về cơ bản nó là cặp ba {chủ thể, thuộc tính và giá trị}, nếu lưu trữ bản ghi này thì có thể truy vấn theo thuộc tính để tìm thông tin liên quan đến chủ thể.

Mỗi tài nguyên trên mạng được xem là một thực thể, thông tin về một tài nguyên được lưu giữ dưới dạng thuộc tính của thực thể đó. Khác với hệ thống đặt tên có cấu trúc, việc tìm kiếm giá trị trong hệ thống đặt tên dựa trên thuộc tính đòi hỏi tìm kiém toàn diện thông qua tất cả các bộ mô tả, có thể tránh được tìm kiém toàn diện bằng cách lập các bảng chỉ mục. Khi tìm kiếm chỉ cần gửi yêu cầu đến các máy chủ dịch vụ thư mục, nhưng số lượng rất lớn, gửi hàng trăm câu lệnh truy vấn không phải là cách hay, do đó cần phải có các giải pháp tìm kiếm đối với loại hình này.

### 3.2.3.2 Cài đặt theo kiến trúc phân cấp

Cách tiếp cận phổ biến để giải quyết dịch vụ thư mục phân tán là kết hợp đặt tên có cấu trúc với đặt tên dựa trên thuộc tính, nó đã được áp dụng rộng rãi trong các dịch vụ hệ thống phân tán. Nhiều hệ thống sử dụng giao thức truy nhập thư mục hạng nhẹ LDAP, nó bắt nguồn từ dịch vụ thư mục X.500 do hai tổ chức quốc tế ITU và ISO xây dựng. Ban đầu tiêu chuẩn này dùng để hỗ trợ các yêu cầu của các nhà cung cấp dịch vụ nhắn tin và tra cứu tên mạng.

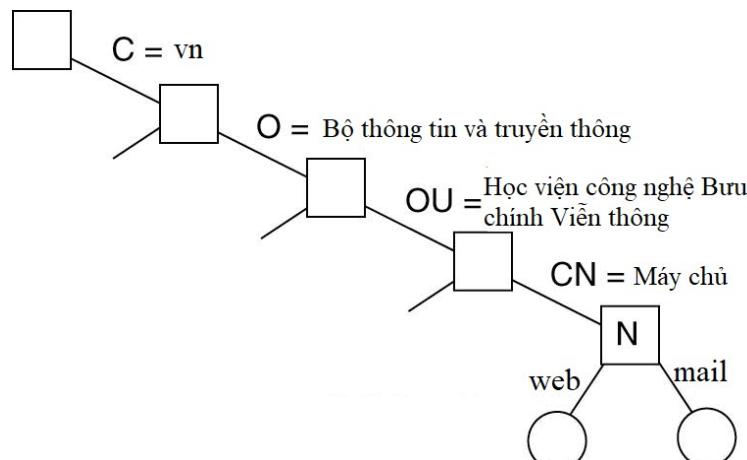
Giao thức LDAP dựa trên các dịch vụ thư mục thông tin của X.500 nhưng sử dụng mô hình TCP/IP và một chuỗi mã hóa chương trình của giao thức X.500 DAP, Dịch vụ

thư mục bao gồm các bản ghi thể hiện các phần trong thư mục, mỗi bản ghi là tập hợp của các cặp {thuộc tính, giá trị}, mỗi thuộc tính có một kiểu liên kết, nó có thể là đơn trị hoặc đa trị.

**Bảng 3.4** Cặp thuộc tính và giá trị trong dịch vụ thư mục

Thuộc tính	Viết tắt	Giá trị
Country	C	Vn
Organization	O	Bộ thông tin và truyền thông
Organization Unit	OU	Học viện công nghệ Bưu chính Viễn thông
CommonName	CN	Máy chủ
WebServer	-	222.255.113.97, 203.162.88.119, 203.162.10.122
MailServer	-	123.30.182.167

Bảng 3.4 mô tả địa chỉ các máy chủ trang tin và thư điện tử của Học viện Công nghệ Bưu chính Viễn thông, thuộc tính WebServer nhận dữ liệu đa trị, các thuộc tính khác nhận dữ liệu đơn trị. Tập hợp tất cả các mục trong dữ liệu của LDAP gọi là cơ sở thông tin thư mục, mỗi bản ghi được xác định duy nhất để có thể tra cứu, một tên duy nhất trên toàn cầu xuất hiện dưới dạng chuỗi các thuộc tính đặt tên trong mỗi bản ghi, mỗi thuộc tính đặt tên gọi là tên phân biệt tương đối. Trong ví dụ này, năm thuộc tính đầu tiên đều là các thuộc tính đặt tên, chúng tạo thành tên duy nhất trên toàn cầu, tương tự như tên miền có cấu trúc www.ptit.edu.vn.



Hình 3.19 Tổ chức bản ghi dịch vụ thư mục

Giống như hệ thống tên miền, việc sử dụng các tên duy nhất trên toàn cầu bằng cách liệt kê các tên tương đối theo thứ tự dẫn đến hệ thống phân cấp của tập hợp các mục dữ liệu và được gọi là cây thông tin thư mục, đó là đồ thị tên của dịch vụ thư mục. Ví dụ trên hình 3.19 mô tả cách lưu trữ thông tin trong dịch vụ thư mục, nút N đóng vai trò là nút thư mục và là cha của hai nút lá thuộc tính bổ sung HostName, nó đóng vai trò như một nút thư mục trên đồ thị tên và cũng là bản ghi của LDAP.

Bảng 3.5 thể hiện cặp các thuộc tính và giá trị, của hai nút lá trong ví dụ này, rõ ràng cách tổ chức phân cấp đã tiết kiệm không gian lưu trữ. Việc cài đặt dịch vụ thư mục tương tự như cách triển khai dịch vụ đặt tên phân cấp, tuy nhiên nó hỗ trợ nhiều thao tác tra cứu hơn. Khi xử lý thư mục có quy mô lớn, cây thông tin thư mục sẽ được cài đặt trên

máy chủ gọi là đại lý hệ thống thư mục DSA, đó là tập hợp các dịch vụ và tiến trình cung cấp truy nhập đến dữ liệu. Mỗi phần của cây thông tin thư mục sẽ tương ứng với một vùng trong hệ thống đặt tên, mỗi DSA hoạt động như một máy chủ định danh thông thường nhưng có thêm các tính năng tìm kiếm nâng cao. Tiến trình máy khách kết nối đến DSA theo giao thức LDAP để sử dụng dịch vụ thư mục.

**Bảng 3.5** Cặp thuộc tính và giá trị cây thư mục

Thuộc tính	Giá trị	Thuộc tính	Giá trị
Country	Vn	Country	Vn
Organization	Bộ thông tin và truyền thông	Organization	Bộ thông tin và truyền thông
Organization Unit	Học viện công nghệ Bưu chính Viễn thông	Organization Unit	Học viện công nghệ Bưu chính Viễn thông
CommonName	Máy chủ	CommonName	Máy chủ
HostName	Web	HostName	Mail
HostAddress	222.255.113.97, 203.162.88.119, 203.162.10.122	HostAddress	123.30.182.167

Điểm khác biệt giữa dịch vụ thư mục và hệ thống phân giải tên miền nằm ở thực thể lưu trữ và các phương tiện tìm kiếm, trong khi hệ thống phân giải tên miền chỉ tập trung quản lý tên thân thiện và địa chỉ của thực thể thì dịch vụ thư mục lại quản lý rất nhiều thuộc tính khác. Dịch vụ thư mục dựa vào cơ sở thông tin thư mục, nó tìm kiếm các thực thể dựa trên các tiêu chí của các thuộc tính. Ví dụ, để tìm kiếm các máy chủ của các đơn vị trực thuộc Bộ thông tin và truyền thông chỉ cần thực hiện lệnh sau:

search("(C=vn)(O= Bộ thông tin và truyền thông) (OU=\*)(CN=Máy chủ)")

Tìm kiếm trong dịch vụ thư mục thường là hoạt động tốn kém, câu lệnh trên đòi hỏi phải tìm kiếm các mục dữ liệu cho mỗi đơn vị của Bộ thông tin và truyền thông sau đó kết hợp kết quả tìm kiếm với nhau, nghĩa là phải truy nhập vào một số nút lá của cây thông tin thư mục, trong khi đó dịch vụ tên miền chỉ cần truy nhập một nút lá duy nhất. Dịch vụ thư mục cho phép một số cây cùng tồn tại trong khi chúng được liên kết với nhau, điều này dẫn đến một rùng các miền dịch vụ thư mục, do đó cần cài đặt một máy chủ danh mục chung để chỉ ra cần tìm kiếm trên những miền nào.

Bản thân dịch vụ thư mục sử dụng hệ thống phân cấp để đáp ứng yêu cầu về quy mô, có thể kết hợp dịch vụ thư mục với hệ thống phân giải tên miền, khi đó mọi cây trong dịch vụ thư mục đều có thể truy nhập từ gốc. Tiêu chuẩn LDAP/X.500 đang được áp dụng trong các sản phẩm như Novell eDirectory, Red Hat Directory Server, Microsoft Active Directory, Apple's Mac Open Directory, Apache Directory Server, Oracle Internet Directory.

### 3.2.3.3 Cài đặt theo kiến trúc ngang hàng

Với sự ra đời của các hệ thống ngang hàng, các nhà nghiên cứu cũng đang tìm kiếm những giải pháp phi tập trung hóa hệ thống đặt tên dựa trên thuộc tính. Hệ thống ngang hàng thường được sử dụng để lưu trữ các tập tin, nó chỉ cung cấp tính năng tra cứu

dựa trên khóa mà không hỗ trợ các tính năng tìm kiếm. Tuy nhiên, nếu có thể tìm kiếm được tập tin dựa trên bộ mô tả thì đó là điều rất tiện lợi, trong đó bộ mô tả là cặp {thuộc tính, giá trị}, vấn đề mấu chốt ở đây là phải tổ chức sao cho quá trình tìm kiếm thực hiện tốt nhất. Việc truy vấn mọi nút trong hệ thống ngang hàng để xem nó có chứa tập tin khớp với các cặp thuộc tính hay không là điều không khả thi, những gì cần làm là ánh xạ các cặp {thuộc tính, giá trị} đến các máy chủ chỉ mục trỏ đến các tập tin phù hợp với các cặp thuộc tính và giá trị đó.

Ý tưởng xây dựng cây chỉ mục phân tán là truy vấn tìm kiếm được thể hiện dưới dạng danh sách các cặp {thuộc tính, giá trị}, kết quả trả về là danh sách tham chiếu đến các thực thể phù hợp với những giá trị của các thuộc tính. Ví dụ hệ thống ngang hàng lưu trữ tập tin, danh sách trả về là các khóa của các tập tin có liên quan, từ đó máy khách có thể sử dụng lần lượt từng khóa để tra cứu các tập tin. Giả sử có M thuộc tính khác nhau, có thể sử dụng máy chủ nào đó cho mỗi thuộc tính, máy chủ cho thuộc tính A duy trì tập hợp các cặp {E, Val<sub>A</sub>} cho mỗi thực thể E có giá trị Val<sub>A</sub> cho thuộc tính A, ví dụ câu truy vấn sau:

```
search("(C=vn)(O= Bộ thông tin và truyền thông) (OU=*)(CN=Main server)")
```

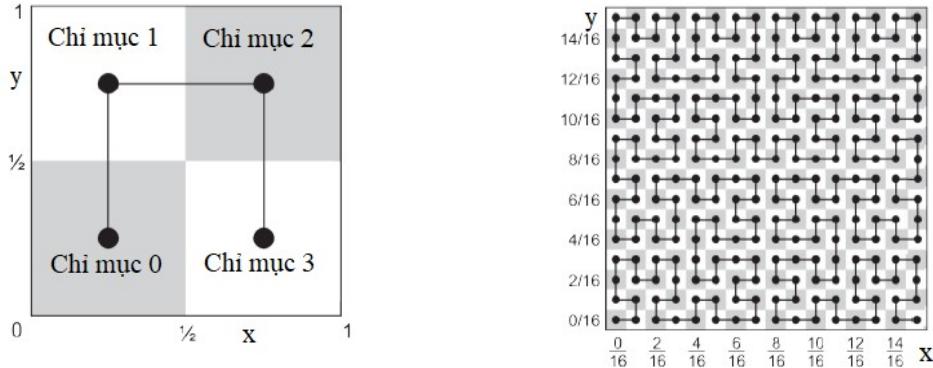
Câu lệnh truy vấn trên sẽ được gửi đến các máy chủ tương ứng cho các thuộc tính C, O và CN, sau đó máy khách sẽ xem thực thể nào xuất hiện trong cả ba tập hợp do các máy chủ trả về, mỗi máy chủ có thể phân vùng thêm để giảm số lượng tập thực thể trên mỗi máy. Giả sử có tập thuộc tính {a<sup>1</sup>, a<sup>2</sup>, ..., a<sup>N</sup>}, mỗi thuộc tính a<sup>k</sup> liên kết với tập S<sup>k</sup> = {S<sup>k</sup><sub>1</sub>, ..., S<sup>k</sup><sub>n<sub>k</sub></sub>} của n<sub>k</sub> máy chủ và a<sup>k</sup> nhận giá trị từ tập R<sup>k</sup>, xây dựng ánh xạ toàn cục F(a<sup>k</sup>, v) = S<sup>k</sup><sub>j</sub> với S<sup>k</sup><sub>j</sub> ∈ S<sup>k</sup> và v ∈ R<sup>k</sup>. Áp dụng cho câu lệnh tìm kiếm trong ví dụ trên, máy chủ S<sup>k</sup> theo dõi khóa liên quan đến tập tin có a<sup>k</sup> = v. Nếu L(a<sup>k</sup>, v) là tập các khóa trả về từ máy chủ F(a<sup>k</sup>, v) thì câu lệnh truy vấn có thể viết dưới dạng biểu thức logic (F(a<sup>1</sup>, v<sup>1</sup>) ∧ F(a<sup>2</sup>, v<sup>2</sup>)) ∨ F(a<sup>3</sup>, v<sup>3</sup>), nó có thể được máy khách xử lý bằng cách xây dựng tập (L(a<sup>1</sup>, v<sup>1</sup>) ∩ L(a<sup>2</sup>, v<sup>2</sup>)) ∪ L(a<sup>3</sup>, v<sup>3</sup>).

Giải pháp trên có ưu điểm đơn giản nhưng nó lại tiềm ẩn ba yếu điểm cơ bản. Thứ nhất chi phí về truyền thông có thể rất cao, bất kỳ truy vấn nào liên quan đến k thuộc tính thì phải liên hệ với k máy chủ chỉ mục. Nhược điểm thứ hai là tập kết quả trả về cho máy khách có thể rất lớn, thực tế tập khóa trả về cho thuộc tính C=vn có thể lên tới hàng triệu bản ghi. Nhược điểm thứ ba là nó không hỗ trợ cho việc tìm kiếm theo phạm vi, giả sử có thêm thuộc tính số lượng nhân viên, khó có thể thực hiện truy vấn những đơn vị có số nhân viên trong khoảng từ 1000 đến 2000.

Một cách tiếp cận khác để cài đặt hệ thống đặt tên dựa trên thuộc tính theo kiến trúc ngang hàng là sử dụng phương pháp đường cong lấp đầy không gian. Ý tưởng cơ bản là lập bản đồ không gian N chiều bao phủ bởi N thuộc tính {a<sup>1</sup>, a<sup>2</sup>, ..., a<sup>N</sup>} cho mỗi chiều, sau đó sử dụng kỹ thuật băm để phân phối không gian kết quả giữa các máy chủ chỉ mục, điều quan trọng ở đây là phải có các cặp {thuộc tính, giá trị} gần với nhau để xử lý trên cùng một máy chủ.

Hình 2.20 minh họa ví dụ đường cong lấp đầy không gian Hilbert, điều này dễ giải thích bằng không gian hai chiều tượng trưng cho hai thuộc tính, mỗi thuộc tính tương ứng

với một trục của không gian. Giả sử mỗi thuộc tính chỉ nhận các giá trị trong khoảng  $[0,1]$ , đầu tiên chia không gian thành bốn phần tương ứng với bốn chỉ số. Tất cả các dữ liệu  $(x, y)$  với  $0 \leq x, y < 0,5$  liên kết với chỉ số 0, các dữ liệu  $(x, y)$  với  $0,5 \leq x, y < 1,0$  liên kết với chỉ số 2. Có thể lặp lại qui trình này theo phương pháp đệ qui cho mỗi phần con và nối các phần bằng một đường duy nhất, đường cong Hilbert bậc k sẽ nối  $2^{2k}$  ô tương ứng với  $2^{2k}$  chỉ mục.



Hình 3.20 Cách đánh chỉ mục trong không gian Hilbert

Tính chất quan trọng của đường cong lấp đầy không gian là chúng bảo toàn tính cục bộ, hai chỉ mục gần nhau trên đường cong tương ứng với hai điểm cũng gần nhau trong không gian đa chiều, điều ngược lại chưa chắc đã đúng. Các giá trị của thuộc tính cần phải được lập chỉ mục, giả sử có N thuộc tính  $\{a^1, a^2, \dots, a^N\}$  và mỗi thực thể gán một giá trị cho từng thuộc tính, để đơn giản chúng ta coi các giá trị này đã được chuẩn hóa nằm trong khoảng  $[0, 1]$ . Như vậy mỗi thực thể E sẽ có bộ giá trị  $\{v^1, v^2, \dots, v^N\}$  được liên kết với một tọa độ có giá trị thực trong không gian N chiều, kết quả là nó được liên kết duy nhất với một ô con trong không gian này. Tâm của ô con đó tương ứng với chỉ mục trên đường cong lấp đầy không gian và giờ đây nó liên kết với thực thể E. Tất nhiên nhiều thực thể có tọa độ rơi vào cùng một ô sẽ có cùng chỉ mục, để tránh xung đột này thì sử dụng bậc cao hơn, bậc 32 hay 64 không phải là hiếm.

Nguyên lý tìm kiếm các thực thể dựa trên các giá trị thuộc tính cũng đã rõ ràng, giả sử các giá trị thuộc tính  $a^1, a^2$  nằm trong khoảng  $[v^1_l, v^1_u]$  và  $[v^2_l, v^2_u]$  với  $v^i_l < v^i_u$ , các tọa độ đó phân định các ô mà đường cong đi qua, tất cả các tập tin được đánh chỉ mục bởi những đoạn đường cong giao với các ô đó phù hợp với tiêu chí tìm kiếm. Do đó cần phải có thao tác trả về dãy các chỉ mục liên quan đến đường cong cho một vùng trong không gian N chiều có liên quan, thao tác như vậy phụ thuộc vào đường cong lấp đầy không gian nào đã được sử dụng mà không phụ thuộc vào các thực thể. Cuối cùng là việc tham chiếu đến các thực thể liên quan tới các chỉ mục, một trong những giải pháp lựa chọn là sử dụng mạng Chord.

## THẢO LUẬN

- Trình bày vai trò của hệ thống tên trong hệ phân tán, nêu ví dụ về một hệ thống tên trong thực tế.

2. Tìm hiểu giao thức RARP.
3. Sử dụng tiện ích ARP cài đặt trong các hệ điều hành, tìm hiểu các thông tin do tiện ích này cung cấp.
4. Lấy một số ví dụ về tên miền phân cấp.
5. Nêu giải pháp đặt tên đối tượng trong môi trường toàn cầu.
6. Thử nghiệm ánh xạ thư mục giữa các máy tính khác nhau.
7. Trình bày cơ chế phân giải tên miền đang áp dụng trên mạng Internet.
8. Sử dụng tiện ích NSLOOKUP cài đặt trong các hệ điều hành, tìm hiểu các thông tin do tiện ích này cung cấp.
9. Thử nghiệm sử dụng địa chỉ IP của một số máy chủ tên miền của Việt Nam và so sánh hiệu năng với máy chủ tên miền của Google, đưa ra khuyến nghị cho người sử dụng.
10. Thử nghiệm cài đặt dịch vụ thư mục có sẵn trong các hệ điều hành.

## CHƯƠNG 4: ĐỒNG BỘ VÀ CÁC GIẢI THUẬT PHÂN TÁN

Trao đổi thông tin giữa các tiến trình là nhiệm vụ quan trọng trong các hệ thống phân tán, tuy nhiên đó mới chỉ là một trong những nhiệm vụ cơ bản, vấn đề đặt ra là làm thế nào để các tiến trình có thể phối hợp và đồng bộ với nhau. Khái niệm phối hợp được định nghĩa là sự kết nối hoạt động của các tiến trình nhằm giải quyết nhiệm vụ chung. Khái niệm đồng bộ được định nghĩa là những hoạt động có cùng chu kỳ hoặc cùng tốc độ, được tiến hành trong cùng một thời gian, nó tạo ra sự phối hợp nhịp nhàng giữa các tiến trình trong hệ thống.

Phối hợp và đồng bộ là hai vấn đề liên quan mật thiết với nhau, mục tiêu của phối hợp hoạt động là quản lý các tương tác và phụ thuộc lẫn nhau giữa các tiến trình, đồng bộ đòi hỏi một tiến trình phải chờ tiến trình khác hoàn thành thì mới thực hiện công việc của mình. Chỉ có sự ăn khớp giữa các bước thực hiện trên tất cả các tiến trình mới tạo nên hoạt động nhịp nhàng, như thế mới hình thành một hệ thống gắn kết như một thể thống nhất. Tuy nhiên khái niệm đồng bộ trong hệ thống dữ liệu nhân bản lại tập trung giải quyết tính nhất quán chứ không phải thứ tự thực hiện của các tiến trình.

Đồng bộ trong hệ thống phân tán phức tạp hơn rất nhiều so với các hệ thống xử lý trên một máy tính, nhất là đối với các ứng dụng yêu cầu xử lý theo thời gian thực. Mỗi thành viên trong hệ thống phân tán sử dụng đồng hồ vật lý riêng, không có gì đảm bảo rằng tại một thời điểm đồng hồ vật lý của tất cả các thành viên trong hệ thống đều có giá trị giống nhau. Mỗi tiến trình nhận nhiều thông điệp từ các tiến trình khác, mạng máy tính cũng không đảm bảo phân phát thông điệp theo đúng thứ tự đã gửi, quyết định ưu tiên xử lý thông điệp nào trước cũng là vấn đề quan trọng.

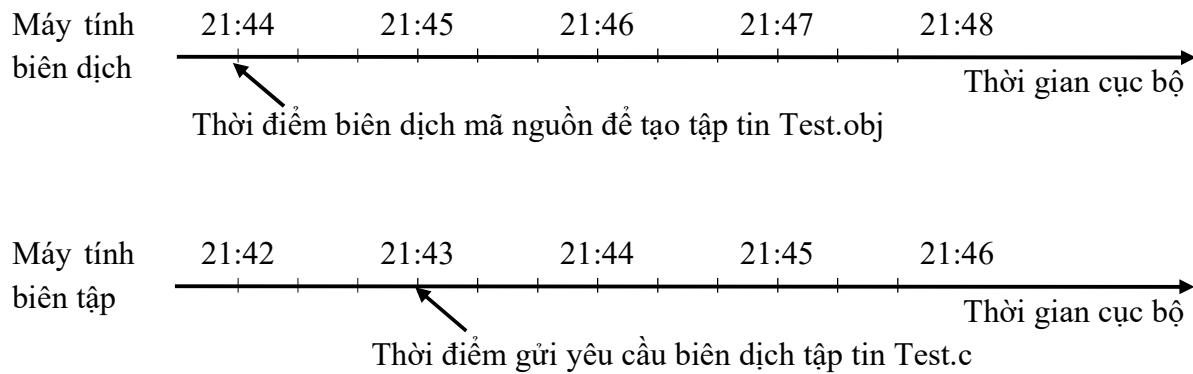
Dưới góc độ tương tranh, nếu nhiều tiến trình đồng thời truy nhập vào một tài nguyên vật lý sẽ dẫn hiện tượng xung đột, do đó phải có giải pháp để tất cả các tiến trình đều được phép sử dụng tài nguyên theo cách hợp lý nhất. Để tạo sự công bằng thì mỗi tiến trình chỉ được phép sử dụng tài nguyên trong một khoảng thời gian nhất định, như vậy đòi hỏi phải có các giải thuật phân phối quyền sử dụng tài nguyên chung. Để các tiến trình có thể phối hợp với nhau một cách đồng bộ, trước hết phải giải quyết các vấn đề liên quan đến thời gian, sau đó là những vấn đề về tương tranh.

### 4.1 Đồng bộ đồng hồ vật lý

Thời gian trên một máy tính thể hiện rất rõ ràng vì nó chỉ sử dụng duy nhất một đồng hồ vật lý, một tiến trình khi cần biết thời gian chỉ cần thực hiện lời gọi và bộ đếm thời gian cài đặt trong phần lõi của hệ thống sẽ trả lời. Nếu hai tiến trình yêu cầu thời gian hệ thống thì giá trị thời gian của tiến trình gọi trước bao giờ cũng nhỏ hơn giá trị thời gian của tiến trình gọi sau.

Đồng thuận về thời gian giữa các tiến trình trong hệ thống phân tán là vấn đề không đơn giản, ngay cả trường hợp lấy thời gian từ một máy chủ thì độ trễ truyền thông vẫn gây ra những sai lệch. Sự sai lệch của thời gian làm cho việc xác định thứ tự các sự kiện trở nên khó xác định, dẫn đến xử lý không chính xác.

Để hiểu về nhu cầu đồng bộ thời gian giữa các thành phần trong hệ thống phân tán, xét ví dụ về dự án phát triển phần mềm, hai lập trình viên cùng biên soạn một tập tin mã nguồn và biên dịch. Hình 4.1 minh họa đồng hồ trên hai máy tính không khớp nhau, máy tính biên dịch có thời gian nhanh hơn máy tính biên tập và cả hai máy tính này đều không sử dụng đồng hồ toàn cục, các lập trình viên cùng biên soạn tập tin Test.c sau đó sẽ biên dịch để được tập tin Test.obj. Phần mềm biên dịch thực hiện theo nguyên tắc, trước hết nó kiểm tra tất cả các tập tin nguồn, nếu thời gian tập tin mã nguồn nhỏ hơn thời gian tập tin Test.obj tương ứng đã được biên dịch gần nhất thì tập tin mã nguồn đó sẽ không được biên dịch.



Hình 4.1 Hai máy tính có thời gian khác nhau

Tại thời điểm máy tính thứ nhất biên dịch, đồng hồ là 2144, trong khi đó đồng hồ trên máy tính thứ hai là 2142, như vậy tập tin Test.obj sẽ ghi nhận thời gian được biên dịch là 2144. Sau một đơn vị thời gian, lập trình viên trên máy tính thứ hai thực hiện biên dịch, chương trình dịch kiểm tra thời gian cuối cùng biên dịch tập tin Test.obj là 2044 cao hơn thời gian hiện hành 2143 của máy tính thứ hai, do đó nó không thực hiện biên dịch mặc dù tập tin output.c đã được chỉnh sửa. Như vậy đã xảy ra vấn đề ghi nhận thời gian cập nhật gần nhất đối với tập tin Test.c và lập trình viên trên máy tính thứ hai khó có thể xác định nguyên nhân lỗi trong trường hợp này. Ví dụ trên chỉ là một trong những minh chứng cho thấy vai trò quan trọng của thời gian trong các hệ thống phân tán, lỗi trên có thể sẽ không xảy ra nếu như đồng hồ thời gian trên các máy tính khớp nhau. Vậy câu hỏi đặt ra là có một cơ chế nào để đồng bộ tất cả các đồng hồ trong hệ thống phân tán hay không, câu trả lời tưởng chừng đơn giản nhưng thực tế lại khá phức tạp.

Thời gian đóng vai trò rất quan trọng trong các hệ thống thông tin, nếu trên một máy tính thì giá trị thời gian đọc sau không bao giờ nhỏ hơn lần đọc trước. Hầu hết các máy tính đều có một vi mạch để duy trì thời gian, ngôn ngữ phổ thông gọi là đồng hồ nhưng chính xác hơn phải gọi là bộ đếm thời gian máy tính. Bộ đếm thời gian trong máy tính là một bộ dao động tinh thể thạch anh, tần suất dao động của nó phụ thuộc vào loại tinh thể và hiệu điện thế cài đặt trong vi mạch. Trong vi mạch này có hai thanh ghi, một thanh ghi dùng làm bộ đếm và một thanh ghi dùng để lưu trữ, sau mỗi dao động của tinh thể bộ đếm sẽ giảm 1 đơn vị, khi đạt giá trị 0 sẽ sinh ra một ngắt thời gian sau đó được nạp lại giá trị lấy từ thanh ghi lưu trữ. Mặc dù tần số của bộ dao động tinh thể thạch anh

khá ổn định, nó không thể đảm bảo các tinh thể trong các máy tính khác nhau đều chạy chính xác cùng tầm số.

Mỗi thành viên của hệ thống phân tán lại có đồng hồ đo thời gian riêng, sự tồn tại nhiều đồng hồ vật lý như vậy sẽ phát sinh yêu cầu đồng bộ thời gian giữa các thành viên với nhau. Đồng hồ đo thời gian thường dựa vào bộ dao động tinh thể thạch anh, tầm số dao động của nó khác nhau chút ít trên các máy tính dẫn đến thời gian khác nhau. Sự khác nhau về giá trị thời gian được gọi là sự sai lệch của đồng hồ và kết quả sẽ dẫn đến những sai lệch trong những tính toán có liên quan đến thời gian thực.

Theo thiên văn học, ngày mặt trời thực là khoảng thời gian giữa hai lần mặt trời đi qua điểm cao nhất trên bầu trời địa phương, chính xác hơn phải là trái đất tự quay khoảng một vòng và di chuyển quanh mặt trời sao cho mặt trời ở điểm cao nhất trên bầu trời. Tốc độ trái đất tự quay, vận tốc di chuyển và góc nghiêng so với mặt phẳng quỹ đạo không phải là những giá trị cố định, do đó ngày mặt trời thực cũng là giá trị thay đổi. Quan sát nhiều năm, các nhà thiên văn học thống nhất sử dụng giá trị trung bình của ngày mặt trời thực, vì vậy thang đo thời gian chúng ta đang sử dụng hiện nay mỗi ngày dài 24 giờ và mỗi giờ có 3600 giây.

Năm 1948, các nhà vật lý đã phát minh ra phương pháp đo thời gian chính xác hơn và không phụ thuộc vào sự dao động và lắc lư của trái đất bằng cách đếm các chuyển đổi của nguyên tử Cesium 133. Một giây được định nghĩa là thời gian nguyên tử Cesium 133 thực hiện chính xác  $9192631770$  lần chuyển đổi, giá trị này bằng với một giây mặt trời trung bình quan sát trong năm. Thời gian nguyên tử quốc tế TAI chính thức có hiệu lực từ ngày 01/01/1958, nó được đo bằng giá trị trung bình của một số đồng hồ nguyên tử Cesium 133.

Đồng hồ nguyên tử được coi là chính xác và ổn định nhưng mỗi ngày nó vẫn lệch khoảng 3 mili giây so với ngày trung bình mặt trời thực, ủy ban thời gian quốc tế giải quyết vấn đề bằng cách đưa ra giây nhuận, bất cứ khi nào chênh lệch giữa thời gian TAI và thời gian mặt trời thực tăng lên đến 800 mili giây. Sự hiệu chỉnh này tạo ra một hệ thống thời gian dựa trên giây TAI không đổi nhưng vẫn cùng pha với chuyển động thực của trái đất quanh mặt trời, hệ thống thời gian này được gọi là thời gian phối hợp quốc tế UTC.

Đồng hồ vật lý của các thiết bị cần phải được đồng bộ với thời gian chuẩn, nếu các máy tính có các bộ thu thời gian và có thể kết nối mạng Internet thì nên đồng bộ theo thời gian UTC, nếu không thể liên lạc được thì sử dụng các biện pháp tốt nhất có thể để đảm bảo chênh lệch thời gian giữa các thành viên ít nhất. Tất cả các giải thuật đều sử dụng nguyên lý cơ bản, mỗi máy tính đều có một bộ đếm thời gian, nó tạo ra một ngắt H lần trong một giây.

Gọi giá trị của đồng hồ trên máy tính là  $C$ , nếu thời gian UTC là  $t$  thì giá trị của đồng hồ trên máy tính p sẽ là  $C_p(t)$ , trường hợp lý tưởng  $C_p(t) = t$  cho tất cả p và t. Bộ đếm thời gian không ngắt chính xác H lần trong một giây, về lý thuyết, bộ đếm thời gian với  $H = 60$  cần phát ra 216000 nhịp trong một giờ. Thực tế những sai số tương đối đạt được với các vi mạch đếm thời gian hiện đại đạt khoảng  $10^{-5}$ , nghĩa là một máy nào đó có thể

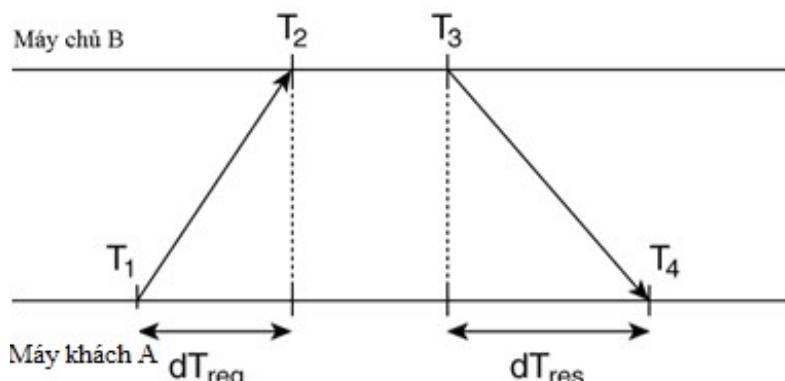
lấy giá trị từ 215998 đến 216002 nhịp trong một giờ. Dù cho đồng hồ của máy tính đã được điều chỉnh chính xác với thời gian UTC thì sau một thời gian nhất định cũng sẽ bị sai lệch, đến một ngưỡng nào đó thì phải đồng bộ lại.

Nếu  $p$  và  $q$  là hai máy tính của một hệ thống phân tán, giải thuật đồng bộ phải duy trì độ lệch thời gian nằm trong giới hạn nhất định  $\pi$ , gọi là độ chính xác bên trong, nghĩa là  $|C_p(t) - C_q(t)| \leq \pi$ . Nếu mỗi máy tính trong hệ thống phân tán được đồng hồ theo thời gian quốc tế thì sai số cũng không được phép vượt quá giá trị  $\alpha$ , gọi là độ chính xác bên ngoài, nghĩa là  $|C_p(t) - t| \leq \alpha$ , sai số thời gian của đồng hồ giữa hai máy tính bất kỳ trong hệ thống sẽ không vượt quá  $2\alpha$ .

Tần số nhịp đồng hồ trên mỗi máy tính không ổn định dẫn đến hiện tượng chênh lệch thời gian, tồn tại một hằng số  $\rho$  thoả mãn  $1 - \rho \leq dC_p(t)/dt \leq 1 + \rho$ , hằng số  $\rho$  sẽ được nhà sản xuất công bố và được gọi là tỉ lệ dung sai tối đa. Nếu hai đồng hồ trôi từ UTC theo hướng ngược nhau, sau khoảng thời gian  $\Delta t$  sau khi đồng bộ, chênh lệch có thể lên tới  $2\rho \times \Delta t$ . Nếu muốn chênh lệch thời gian giữa các máy tính trong một hệ thống không vượt quá  $\pi$  giây thì sau mỗi  $\pi/(2\rho)$  giây phải thực hiện đồng bộ lại. Bốn giải thuật phổ biến đang sử dụng hiện nay bao gồm Cristian, Berkeley, trung bình và tham chiêu quảng bá.

#### 4.1.1 Giải thuật Cristian

Giải thuật này do Cristian đề xuất vào năm 1989 và đã được áp dụng cho đồng bộ thời gian máy khách với máy chủ cung cấp thời gian UTC bằng cách sử dụng giao thức đồng bộ thời gian mạng NTP, gọi là máy chủ NTP. Đặc tả của giao thức NTPv3 cho địa chỉ IPv4 được mô tả trong RFC-1305 và NTPv4 cho địa chỉ IPv6 trong RFC-5905, chúng mô tả khuôn dạng thông điệp và cách thức trao đổi giữa máy khách và máy chủ thời gian UTC.



Giả thiết thời gian lưu chuyển bản tin yêu cầu và bản tin trả lời như nhau. Sai số thời gian giữa máy trạm và máy chủ thời gian được tính theo công thức:

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

Hình 4.2 Giải thuật Cristian

Máy chủ NTP cung cấp dịch vụ trên cổng 123 và sử dụng giao thức UDP ở tầng vận tải, dữ liệu thời gian trong thông điệp loại Big Endian, do đó những máy tính loại

Little Endian cần phải thực hiện thao tác chuyển đổi. Trên mạng Internet có rất nhiều máy chủ NTP, các máy tính cài đặt hệ điều hành của hãng Microsoft thường đồng bộ thời gian với máy chủ có tên miền là time.windows.com.

Tiến trình máy khách tạo một Socket chứa thông điệp yêu cầu đồng bộ thời gian, máy chủ sẽ trả lại thông điệp bao gồm thời gian nhận được yêu cầu và thời gian trước khi phản hồi, hai tham số này thường có giá trị giống nhau. Không thể lấy những giá trị này để điều chỉnh đồng hồ trên máy tính, trễ trên kênh truyền sẽ làm sai lệch giá trị thời gian thực tế. Mục đích của giải thuật Cristian là tính toán để loại bỏ ảnh hưởng của độ trễ này, nó có thể đạt độ chính xác 50 mili giây, đây là giá trị có thể chấp nhận được nếu so sánh với độ lệch có thể lên tới 800 mili giây giữa thời gian UTC và TAI.

Hình 4.2 thể hiện các bước thực hiện, tại thời điểm  $T_1$  tiến trình máy khách A gửi yêu cầu và nhận được kết quả trả về từ tiến trình máy chủ B tại thời điểm  $T_4$ , thời gian máy chủ nhận được là  $T_2$  và thời gian phản hồi là  $T_3$ . Giả thiết độ trễ mạng chiều gửi và nhận xấp xỉ bằng nhau, nghĩa là  $|T_2 - T_1| \approx |T_4 - T_3|$ , tiến trình máy khách ước tính độ lệch thời gian với máy chủ  $\Theta = T_3 + (T_2 - T_1 + T_4 - T_3)/2 - T_4 = (T_2 - T_1 + T_3 - T_4)/2$ , thời gian mới của máy khách sẽ phải cộng thêm giá trị tính toán theo công thức trên.

Nếu giá trị  $\Theta < 0$ , nghĩa là đồng hồ của máy khách chạy nhanh hơn thì cần phải điều chỉnh lùi, nhưng thời gian không thể quay lui, hành động này có thể dẫn đến những hậu quả nghiêm trọng. Nếu trong khoảng thời gian quay lui đến thời điểm hiện hành không có sự kiện cập nhật nào thì có thể thiết lập thời gian mới, nhưng nếu có sự kiện đã xảy ra thì sẽ dẫn đến nghịch lý thời gian ghi nhận cho sự kiện xảy ra trước lại có giá trị lớn hơn thời gian ghi nhận sự kiện xảy ra sau. Nếu trong hệ thống cài đặt lịch cho một số tiến trình, việc điều chỉnh lùi đồng hồ cũng có thể dẫn đến hiện tượng lặp. Nên tạo vòng lặp để điều chỉnh dần dần, chu kỳ điều chỉnh thường 10 mili giây, nghĩa là cứ đến mili giây thứ 10 thì không tăng thời gian của máy tính, quá trình lặp lại cho đến khi đạt được độ chính xác chấp nhận được, như vậy thời gian trên máy tính không bao giờ quay lui.

Tương tự như vậy đối với trường hợp  $\Theta > 0$ , nghĩa là đồng hồ của máy khách chạy chậm hơn, do đó phải điều chỉnh tiến nhưng cần kiểm tra xem trong khoảng thời gian đó có nhiệm vụ theo lịch hay không, nếu có thì phải tránh hiện tượng bỏ sót. Các hệ điều hành thường đặt lịch chính xác đến giây, do đó chu kỳ thường phải lớn hơn 1 giây, chu kỳ nhỏ có thể dẫn đến hiện tượng lặp, tốt nhất nên tránh đồng bộ trong khoảng thời gian xảy ra sự kiện theo lịch của hệ điều hành.

Giao thức NTP sử dụng dịch vụ không liên kết ở tầng vận tải, độ trễ truyền thông mỗi chiều ước chừng sẽ bằng  $\delta = ((T_4 - T_1) - (T_3 - T_2))/2$ . Để chính xác hơn thì cần thực hiện tám lần, tám cặp giá trị  $(\Theta, \delta)$  được lưu vào vùng đệm, cuối cùng lấy giá trị tối thiểu cho  $\delta$  làm độ trễ ước lượng tốt nhất và sau đó giá trị  $\Theta$  tương ứng là độ lệch thời gian đáng tin cậy nhất.

Các máy chủ thời gian NTP cũng cần phải được đồng bộ, về nguyên tắc thì có thể áp dụng NTP đối xứng, nghĩa là máy B cũng có thể đồng bộ theo thời gian của máy A, các máy chủ NTP tham chiếu thời gian của nhau sẽ gây ra sự hỗn loạn. Để giải quyết vấn đề này, các máy chủ NTP được tổ chức phân tầng, những máy chủ tham chiếu đến đồng

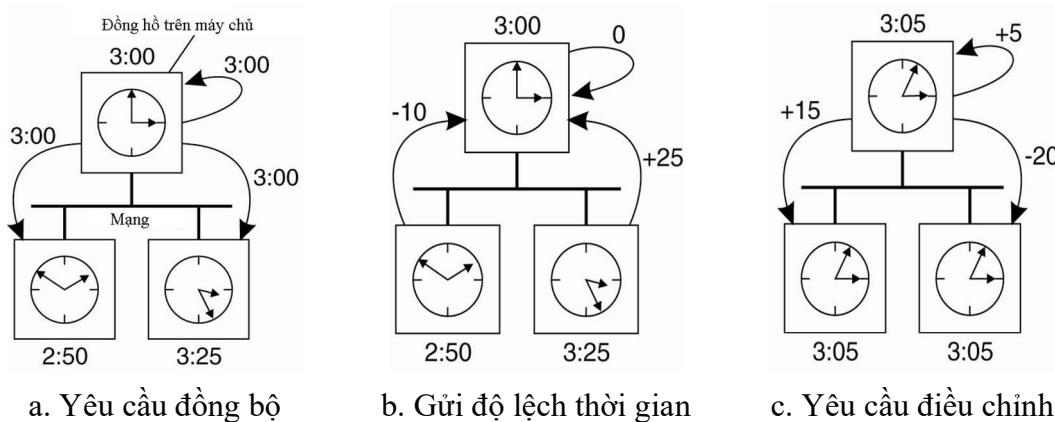
hồ nguyên tử được xếp vào tầng 1, nó chỉ điều chỉnh đồng hồ khi mức tầng của nó cao hơn máy chủ tham chiếu.

#### 4.1.2 Giải thuật Berkeley

Giải thuật Cristian đòi hỏi phải có một máy chủ cung cấp thời gian cho các thành viên trong hệ thống, thời gian của mỗi thành viên sẽ hoàn toàn phụ thuộc vào máy chủ này, như vậy có thể nói máy chủ thời gian hoàn toàn thụ động trong quá trình đồng bộ. Giải thuật Berkeley tiếp cận theo cách ngược lại, tiến trình máy chủ thăm dò thời gian trên các máy khách để tính toán thời gian thời gian cần phải điều chỉnh cho mỗi máy tính trong nhóm, nó phù hợp cho những hệ thống không có khả năng kết nối đến máy chủ thời gian quốc tế.

Giải thuật Berkeley giả thiết không máy tính nào có thời gian chính xác, nó được Gusella và Zatti phát triển tại Đại học California, Berkeley vào năm 1989. Qui trình đồng bộ được thực hiện bằng cách tham khảo thời gian của tất cả các máy tính trong hệ thống, máy chủ thời gian chỉ đóng vai trò như một thành viên điều phối.

Thành viên điều phối gửi thời gian của mình cho các thành viên trong nhóm và yêu cầu từng thành viên trả về độ lệch thời gian giữa thành viên với máy chủ. Sau khi nhận được kết quả từ các thành viên, máy chủ tính trung bình của các giá trị trả về và coi đó là thời gian cần điều chỉnh, máy chủ tính toán độ lệch thời gian cho mỗi thành viên và gửi yêu cầu để các thành viên cập nhật lại thời gian. Thời gian ban đầu trên máy chủ có thể được điều chỉnh thủ công, thời gian sau khi đồng bộ có thể khác với thời gian quốc tế, nhưng tất cả các thành viên cùng đồng thuận với sai lệch này, như vậy giải thuật chỉ được áp dụng trong nội bộ nhóm.



Hình 4.3 Đồng bộ thời gian bằng giải thuật Berkeley

Hình 4.3 minh họa quá trình thực hiện đồng bộ, thành viên điều phối yêu cầu đồng bộ bằng cách gửi thông điệp chứa thời gian hiện hành 03:00 cho các thành viên trong nhóm, bản thân nó cũng là một thành viên. Nhận được yêu cầu, từng thành viên sẽ phản hồi thông điệp chứa độ lệch thời gian được tính bằng cách lấy thời gian của mình trừ đi thời gian được tách từ thông điệp. Tách thông tin từ thông điệp phản hồi của các thành viên, thành viên điều phối sẽ nhận được các giá trị  $\{0, -10, 15\}$ , trung bình cộng của các

độ lệch này là 5. Tiến trình điều phối tính toán thời gian mà mỗi thành viên cần phải điều chỉnh, như vậy bản thân nó phải cộng thêm 5 phút, thành viên bên trái phải cộng thêm 15 phút còn thành viên bên phải sẽ phải điều chỉnh lùi 20 phút, tất nhiên việc điều chỉnh đồng hồ trên máy tính phải tuân thủ những vấn đề đã đề cập trong giải thuật Cristian.

Cách thực hiện như trên đã bỏ qua độ trễ truyền thông và thời gian xử lý trên các thành viên, để đảm bảo chính xác hơn thì phải tính đến các yếu tố này. Thành viên điều phối hoàn toàn có thể ước lượng giá trị của chúng dựa trên thời gian gửi yêu cầu đồng bộ và thời gian nhận thông điệp phản hồi từ các thành viên, giá trị đó cũng là tham số khi tính toán thời gian mỗi thành viên phải điều chỉnh.

Giải thuật yêu cầu một máy tính đóng vai trò điều phối, nếu máy tính này bị lỗi thì không thể thực hiện đồng bộ. Tuy nhiên bất cứ máy tính nào cũng có thể đóng vai trò điều phối, quá trình này có thể thực hiện thông qua các giải thuật bầu chọn. Xét về độ chính xác, giải thuật toán tính toán thời gian mỗi máy tính cần điều chỉnh dựa trên độ lệch thời gian nhận được từ tất cả các máy tính khác trong nhóm, điều này có thể dẫn đến sai sót độ trễ truyền thông lớn và không ổn định, chỉ cần một thành viên không phản hồi thì giải thuật sẽ thất bại.

Vấn đề bảo mật là điểm yếu của giải thuật này, không có biện pháp nào ngăn chặn máy tính độc hại tham gia vào quá trình đồng bộ, điều này có thể làm sai lệch kết quả đồng bộ. Khả năng thích ứng của giải thuật thấp, nó không thích ứng với những thay đổi trong nhóm, ví dụ một máy tính thêm vào hoặc rời bỏ mạng.

#### 4.1.3 Giải thuật trung bình

Giải thuật Berkeley đòi hỏi phải bầu chọn thành viên điều phối, trong quá trình đồng bộ chỉ cần một thành viên không phản hồi yêu cầu của thành viên điều phối thì giải thuật sẽ thất bại. Giải thuật trung bình giả thiết không tồn tại thành viên điều phối và cũng không đòi hỏi tất cả các thành viên đều phải tham gia, các thành viên trong hệ thống thống nhất chu kỳ đồng bộ thời gian, khoảng thời gian thứ i sẽ bắt đầu từ thời điểm  $T_0 + iR$  và chạy đến khi  $T_0 + (i+1)R$ , trong đó  $T_0$  là thời điểm xác định trước và  $R$  là khoảng thời gian của một chu kỳ đồng bộ.

Ở mỗi chu kỳ, các thành viên quảng bá thời gian của mình, sau đó bắt đầu nhận bản tin quảng bá từ các thành viên khác trong khoảng thời gian S. Cách đơn giản nhất để tính thời gian mới là lấy trung bình của tất cả các giá trị nhận được, để tránh những giá trị gây nhiễu có thể loại bỏ M giá trị cao nhất và thấp nhất và lấy thời gian trung bình của phần còn lại.

Bảng 4.1 minh họa nhóm gồm tám máy tính, tại một thời điểm qui định trước, tất cả các tiến trình đồng bộ quảng bá thời gian trên máy tính của mình, sau đó nghe trong khoảng thời gian đã qui định. Giả sử tiến trình đồng bộ trên tất cả các máy tính của nhóm đều nhận được thông điệp quảng bá từ các tiến trình khác, tiến trình trên máy tính 1 nhận được thông điệp từ các máy tính {2, 3, 4, 5, 6, 7, 8}, nó sẽ loại bỏ giá trị nhận được từ máy tính số 4 và số 5 vì chúng là những giá trị nhỏ nhất và lớn nhất, như vậy thời gian sau đồng bộ của máy tính này sẽ là giá trị trung bình thời gian trên các máy tính {2, 3, 6,

7, 8}. Thực hiện tính toán tương tự cho các máy tính khác, dễ dàng nhận thấy chênh lệch thời gian giữa các máy tính đã thu hẹp hơn rất nhiều so với trước chu kỳ đồng bộ.

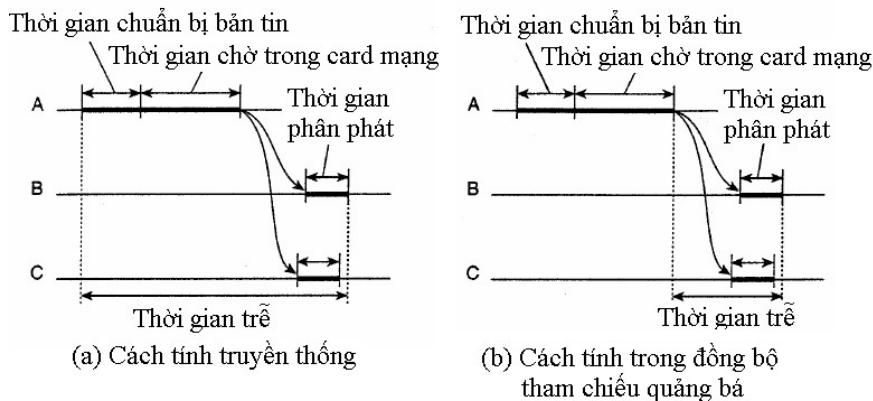
**Bảng 4.1** Ví dụ đồng bộ thời gian sử dụng giải thuật trung bình

Máy tính	Thời gian trước đồng bộ	Thời gian sau đồng bộ
1	22/04/2024 13:10:24.180	22/04/2024 12:56:18.700
2	22/04/2024 12:55:15.030	22/04/2024 12:57:20.540
3	22/04/2024 13:00:24.080	22/04/2024 12:56:18.730
4	22/04/2024 12:50:15.130	22/04/2024 12:58:20.490
5	22/04/2024 13:00:24.230	22/04/2024 12:56:18.700
6	22/04/2024 12:55:15.180	22/04/2024 12:57:20.510
7	22/04/2024 13:00:23.930	22/04/2024 12:56:18.760
8	22/04/2024 12:50:15.280	22/04/2024 12:58:20.490

Để chính xác hơn có thể cộng thêm độ trễ thời gian di chuyển trên mạng cho mỗi bản tin, độ trễ này có thể ước lượng bằng cách gửi thông điệp và đo thời gian phản hồi. Giải thuật trung bình có thể hạn chế nhiễu bằng cách loại bỏ các giá trị biên, tuy nhiên hiện tượng trôi thời gian trên mỗi máy tính làm cho các máy tính không thể bắt đầu chu kỳ đồng bộ cùng một thời điểm, khoảng thời gian S phải nhỏ hơn ngưỡng chênh lệch cho phép, do đó một máy tính có thể sẽ không bao giờ được đồng bộ.

#### 4.1.4 Giải thuật tham chiếu quảng bá

Nếu các máy tính được kết nối với nhau bằng các kênh truyền hữu tuyến thì có thể dễ dàng thực hiện chuyển thông điệp với độ trễ mạng tương đối ổn định. Trong mạng không dây, việc định tuyến qua nhiều điểm truy nhập dẫn đến độ trễ lớn và thường không ổn định, các thiết bị không dây cũng thường tối ưu hóa giải thuật để tiết kiệm năng lượng. Không thể lường trước thời gian thông điệp chờ đợi trước khi gửi lên kênh truyền vật lý, do đó phải thiết kế giải thuật đồng bộ khác loại bỏ ảnh hưởng của độ trễ này.



Hình 4.4 Cách tính thời gian trễ khi chuyển thông điệp

Giải thuật đồng bộ tham chiếu quảng bá thường được sử dụng trong mạng không dây, nguyên lý vận hành của nó khác hẳn với ba giải thuật đồng bộ đã trình bày trên. Giống như giải thuật Berkeley và giải thuật trung bình, giải thuật đồng bộ tham chiếu quảng bá không hướng tới mục tiêu đồng bộ theo thời gian quốc tế UTC mà chỉ đồng bộ theo thời gian của các thành viên nội bộ trong hệ thống. Tương tự như giải thuật trung

bình, giải thuật đồng thuận tham chiểu quảng bá không giả thiết tính sàng của mỗi thành viên.

Các giải thuật trước đây đều sử dụng giao thức trao đổi thông tin hai chiều để đồng bộ, nghĩa là bên gửi và bên nhận đều tham gia vào quá trình đồng bộ. Giải thuật đồng bộ tham chiểu quảng bá tách khỏi quan điểm này, nó loại bỏ thành viên gửi và chỉ cho phép đồng bộ các thành viên nhận thông điệp. Hình 4.4 thể hiện cách tính thời gian trễ khi chuyển thông điệp, giải thuật đồng bộ tham chiểu quảng bá loại bỏ thời gian thông điệp chờ đợi trong giao diện mạng, chỉ có thời gian trễ lan truyền trên môi trường vật lý mới ảnh hưởng đến tính chính xác của giải thuật.

Trong tham đồng bộ tham chiểu quảng bá, bên gửi quảng bá thông điệp tham chiểu với mong muốn những thành viên nhận được sẽ tự điều chỉnh đồng hồ của mình. Điểm quan trọng là ở chỗ trong mạng cảm biến, với giả thiết không cần định tuyến qua nhiều điểm truy nhập thì thời gian lan tỏa tín hiệu đến các nút khác xấp xỉ là hằng số. Độ trễ chuyển thông điệp trong mạng không dây phụ thuộc nhiều yếu tố, dữ liệu phải di chuyển qua bộ định tuyến không dây với tốc độ không ổn định và phụ thuộc nguồn năng lượng cũng như khoảng cách vật lý.

Các thành viên sẽ không lường trước được thời gian chờ để được phục vụ, giá trị của nó thường lớn hơn 10 mili giây trong khi thời gian lan truyền chỉ dưới 1 mili giây, vì vậy một giải pháp được đưa ra là loại bỏ thời gian trễ chuẩn bị bản tin và thời gian chờ đợi trên giao diện mạng. Thời gian lan truyền trong trường hợp này được tính từ thời điểm thông điệp rời khỏi giao diện mạng của bên gửi, như vậy hai nguyên nhân quan trọng làm thay đổi thời gian chuyển thông điệp không còn đóng vai trò trong ước lượng độ trễ, đó là thời gian chi phí cho xây dựng thông điệp và thời gian chi phí cho việc truy nhập mạng.

Các giải thuật đồng bộ thời gian trước đây đều thêm nhãn thời gian vào thông điệp trước khi chuyển đến giao diện mạng, mạng không dây dựa trên giao thức tương tranh và không thể biết trước được thông điệp sẽ phải chờ bao lâu để được thực sự gửi lên kênh truyền vật lý. Những yếu tố không tự quyết sẽ bị loại bỏ trong giải thuật đồng bộ tham chiểu quảng bá, vấn đề còn lại chỉ là thời gian lan tỏa trên môi trường vật lý, nó nhỏ hơn đáng kể so với thời gian truy nhập mạng.

Ý tưởng đồng bộ tham chiểu quảng bá rất đơn giản, một thành viên quảng bá thông điệp tham chiểu m, thông điệp này không chứa thời gian của nó mà chỉ là tín hiệu yêu cầu các thành viên thực hiện đồng bộ. Mỗi thành viên nhận được thông điệp m sẽ ghi nhận thời gian máy tính của mình, ví dụ thành viên P ghi nhận thời gian  $T_{p,m}$  và Q ghi nhận thời gian  $T_{q,m}$ . Các thành viên sau đó sẽ trao đổi thông điệp cho nhau để biết độ lệch thời gian giữa chúng, như vậy thời gian chờ đợi trên giao diện mạng đã được cân bằng trên tất cả các thành viên, vấn đề còn lại chỉ là chênh lệch thời gian xử lý trên mỗi máy tính. Bỏ qua chênh lệch về thời gian xử lý trên mỗi máy tính, nếu thực hiện đồng bộ liên tục M lần thì độ lệch trung bình giữa hai thành viên P và Q được tính theo công thức sau:

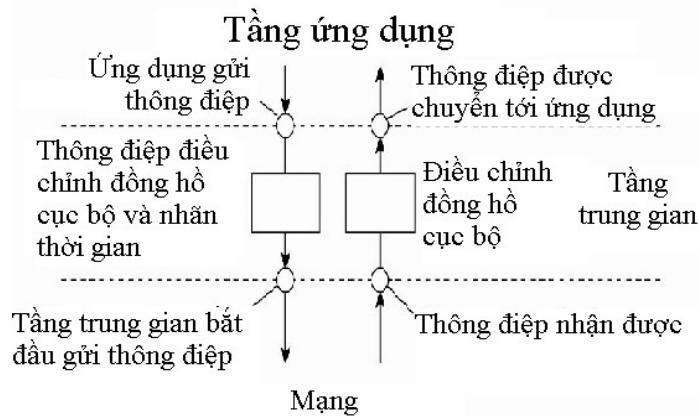
$$Offset[p, q] = \frac{\sum_{k=1}^M (T_{p,k} - T_{q,k})}{M}$$

Nếu chỉ đơn giản lưu trữ các độ lệch này thì không cần điều chỉnh đồng hồ của mình, đáng tiếc đồng hồ trên các máy tính chạy với tốc độ khác nhau, kết quả là việc tính toán đơn giản độ lệch trung bình như trên không còn phù hợp, các giá trị lần cuối cùng sẽ kém chính xác hơn các giá trị đầu tiên. Hơn nữa, thời gian trôi đi thì độ lệch cũng sẽ tăng lên. Elson đã sử dụng giải thuật đơn giản để cân bằng điều này, thay vì tính toán giá trị trung bình thì áp dụng phương pháp hồi quy tuyến tính để xác định độ lệch  $\text{Offset}[p,q](t) = \alpha t + \beta$ , trong đó các hằng số  $\alpha$  và  $\beta$  được tính từ cặp các giá trị  $\{T_{p,k}, T_{q,k}\}$ , công thức mới sẽ cho phép thành viên P tính toán chính xác hơn thời gian hiện hành của Q và ngược lại.

## 4.2 Thời gian và đồng hồ logic

Trong hệ thống phân tán, rất khó duy trì tính đồng bộ về thời gian vật lý giữa các máy tính, mỗi tiến trình sẽ chấp nhận những giá trị chênh lệch khác nhau. Một số hệ thống sử dụng đồng hồ vật lý để giải quyết xung đột, thao tác cập nhật cuối cùng sẽ thắng, chúng chấp nhận rủi ro khi mất đồng bộ thời gian. Bên giải thuật đồng bộ thời gian vật lý không thể đảm bảo chính xác tuyệt đối, chỉ dựa vào thời gian vật lý sẽ không đảm bảo tính nhất quán cũng như độ tin cậy.

Nếu hai tiến trình không tương tác với nhau thì có đồng bộ thời gian hay không cũng không ảnh hưởng đến hoạt động của chúng, nhưng đó là các ứng dụng chạy độc lập trên một máy tính chứ không phải hệ thống phân tán. Các tiến trình chỉ cần xác định thứ tự các sự kiện mà không quan tâm đến thời gian vật lý, do đó cần phải có cơ chế theo dõi các sự kiện này.



Hình 4.5 Đồng hồ logic tại tầng trung gian

Leslie Lamport là người đầu tiên đã đưa ra khái niệm đồng hồ logic vào năm 1978, đó là cơ chế ghi nhận các sự kiện xảy ra trên các tiến trình để xác định thứ tự của chúng. Đồng hồ logic hoạt động độc lập với đồng hồ vật lý trên các máy tính và cũng không phụ thuộc vào độ trễ của mạng. Đồng hồ logic được cài đặt trong các tiến trình, nó được cài đặt trong tầng trung gian và thực hiện nhiệm vụ gán thời gian logic cho mỗi sự kiện, dựa trên chuỗi các giá trị này có thể suy luận tính nhân quả giữa các sự kiện, nó cũng được sử dụng để phát hiện tranh chấp.

Trong phần mở đầu chúng ta đã nói về mô hình phân tầng, tầng trung gian đảm nhiệm vai trò truyền thông giữa các ứng dụng phân tán trên môi trường mạng, tất cả những thao tác đồng bộ đồng hồ logic đều được thực hiện ở tầng này. Mỗi tiến trình có thể sở hữu riêng đồng hồ logic hoặc có thể xây dựng đồng hồ logic dùng chung cho tất cả mỗi tiến trình trên máy tính. Hình 4.5 thể hiện nhiệm vụ của tầng trung gian trong việc gán nhãn thời gian logic cho mỗi sự kiện, đồng hồ logic sẽ được điều chỉnh mỗi khi nhận được thông điệp từ tầng ứng dụng hoặc từ mạng.

#### 4.2.1 Đồng hồ Lamport

Để đồng bộ đồng hồ logic, Lamport dựa trên quan điểm nhân quả và đưa ra định nghĩa mối quan hệ gọi được gọi là xảy ra trước. Nếu  $a$  và  $b$  là hai sự kiện xảy ra trên một tiến trình nhưng  $a$  xảy ra trước  $b$  thì  $a$  là nguyên nhân của  $b$ , ký hiệu  $a \rightarrow b$ . Nếu  $a$  và  $b$  là những sự kiện xảy ra trên hai tiến trình khác nhau nhưng  $a$  là sự kiện gửi thông điệp và  $b$  là sự kiện nhận chính thông điệp đó thì  $a$  là nguyên nhân của  $b$ . Quan hệ xảy ra trước có tính chất bắc cầu, nếu  $a \rightarrow b$  và  $b \rightarrow c$  thì  $a \rightarrow c$ . Nếu  $x$  và  $y$  là hai sự kiện xảy ra trong hai tiến trình khác nhau nhưng không xác định được mối tương quan giữa chúng thì không thể nói  $x \rightarrow y$  hay  $y \rightarrow x$ , trường hợp này ta nói  $x$  và  $y$  là những sự kiện tranh giành vì chúng hoàn toàn có thể xảy ra đồng thời với nhau.

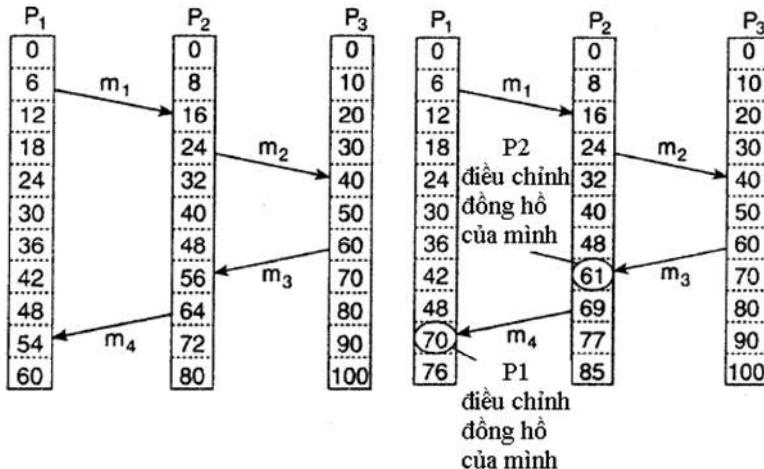
Ký hiệu  $C$  là bộ đếm đồng hồ logic và  $C(x)$  là nhãn thời gian của xảy ra sự kiện  $x$ , mọi sự kiện phân biệt  $a$  và  $b$  thì  $C(a) \neq C(b)$ . Trong cùng một tiến trình, nếu sự kiện  $a$  xảy ra trước sự kiện  $b$  thì  $C(a) < C(b)$ . Nếu  $a$  là sự kiện gửi thông điệp và  $b$  là sự kiện nhận chính thông điệp đó, sự kiện nhận thông điệp phải xảy ra sau sự kiện gửi thông điệp vì cần một khoảng thời gian nhất định để lan tỏa trên mạng, do đó  $C(a) < C(b)$ .

Đo thời gian tương ứng với một sự kiện bằng cách gán một giá trị cho bộ đếm đồng hồ logic  $C$  tại thời điểm sự kiện xuất hiện, các giá trị này phải đảm bảo nếu  $a \rightarrow b$  thì  $C(a) < C(b)$ . Bộ đếm đồng hồ logic  $C$  chỉ được phép tiến chứ không được lùi, nghĩa là nếu cần điều chỉnh thì chỉ được phép cộng thêm giá trị số dương.

Mỗi tiến trình  $P_i$  có bộ đếm  $C_i$  để đánh dấu thời điểm xảy ra các sự kiện trong tiến trình, sự kiện ở đây có thể là nội tại trong tiến trình hoặc là những sự kiện gửi và nhận thông điệp từ tiến trình khác. Trước khi thực hiện một sự kiện, tiến trình  $P_i$  tăng bộ đếm thêm 1 đơn vị, nghĩa là  $C_i \leftarrow C_i + 1$ . Nếu tiến trình  $P_i$  gửi thông điệp  $m$  cho tiến trình  $P_j$  thì nó đính kèm nhãn thời gian  $ts(m) = C_i$ . Nhận được thông điệp  $m$ , tiến trình  $P_j$  tăng bộ đếm  $C_j$  thêm 1 đơn vị để đánh dấu đã có sự kiện xảy ra, nhãn thời gian mới của bộ đếm  $C_j$  là giá trị lớn nhất khi so sánh với nhãn thời gian trong thông điệp nhận được, nghĩa là  $C_j \leftarrow \max\{C_j, ts(m)\}$ .

Hình 4.6 minh họa cách điều chỉnh đồng hồ logic, mỗi tiến trình có bộ đếm thời gian logic riêng nhưng nhịp đếm không giống nhau, ban đầu giá trị đồng hồ trên mỗi tiến trình đều bằng 0. Cùng khoảng thời gian vật lý, tiến trình  $P_1$  chỉ đếm được 6 nhịp trong khi đó tiến trình  $P_2$  đếm được 8 nhịp và tiến trình  $P_3$  đếm được 10 nhịp. Tại nhãn thời gian 6, tiến trình  $P_1$  gửi thông điệp  $m_1$  cho tiến trình  $P_2$ , mất bao lâu để thông điệp này đến đích tùy thuộc vào việc lựa chọn đồng hồ nào, cụ thể trường hợp này  $P_2$  ghi nhận tại nhãn 16 và nó hiểu rằng thời gian di chuyển thông điệp hết 10 đơn vị. Tương tự như vậy,

thời gian di chuyển của thông điệp  $m_2$  là 16 đơn vị, cả hai giá trị này đều là những số dương cho nên  $P_2$  và  $P_3$  đều chấp nhận.



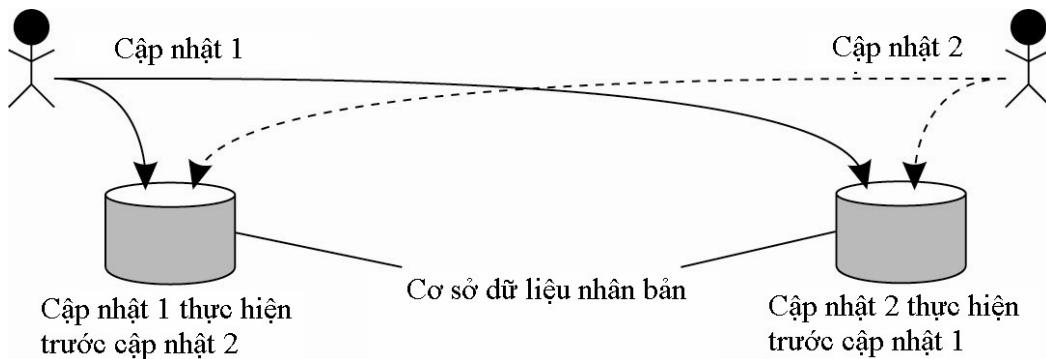
Hình 4.6 Đồng bộ nhãn thời gian Lamport

Xét thông điệp  $m_3$ , thời điểm rời khỏi tiến trình  $P_3$  là 60 và đến đích là 56, như vậy tiến trình  $P_2$  nhận thấy thời gian lưu chuyển là -4, đây là giá trị không thể chấp nhận được và cần phải có biện pháp ngăn chặn hiện tượng này. Giải pháp của Lamport suy luận trực tiếp từ quan hệ xảy ra trước, vì  $m_3$  rời đi tại thời điểm 60 thì thời điểm đến nhanh nhất phải là 61. Nếu giá trị nhãn thời gian tại điểm nhận nhỏ hơn hoặc bằng nhãn thời gian bên gửi thì bên nhận sẽ cộng thêm một đơn vị vào nhãn thời gian gửi, như vậy tiến trình  $P_2$  phải điều chỉnh từ 56 thành 61. Sau 8 đơn vị thời gian, thông điệp  $m_4$  rời khỏi  $P_2$ , như vậy thời điểm gửi đi sẽ là 69 và đến đích tại thời điểm 54, do đó  $P_1$  sẽ phải điều chỉnh đồng hồ thành giá trị 70.

Trong một số trường hợp, cần phải bổ sung thêm định danh của tiến trình để đảm bảo không xảy ra hiện tượng hai sự kiện có nhãn thời gian giống hệt nhau. Ký hiệu  $\langle C_i, P_i \rangle$  được hiểu là sự kiện xảy ra tại thời điểm  $C_i$  trên tiến trình  $P_i$ . Cho hai nhãn thời gian  $\langle C_i, P_i \rangle$  và  $\langle C_j, P_j \rangle$ , nếu  $C_i = C_j$  thì xét tiếp điều kiện giá trị định danh của tiến trình, nếu  $P_i < P_j$  thì  $\langle C_i, P_i \rangle$  nhỏ hơn  $\langle C_j, P_j \rangle$ . Cách mở rộng trên chỉ là biện pháp tình thế, nó chỉ là qui ước chứ không phản ánh thực tế, khó có thể kết luận về mối nhân quả giữa hai sự kiện trong trường hợp này.

Nhãn thời gian Lamport được áp dụng cho truyền thông hàng đợi, ví dụ cập nhật dữ liệu trong các hệ thống nhân bản, thứ tự cập nhật các bản sao phải được thực hiện rất nghiêm ngặt. Giả sử tài khoản của khách hàng có 1000 đồng, tại một thời điểm khách hàng gửi thêm 100 đồng và ngân hàng cũng tính lãi suất 1% cho tài khoản này, hai lệnh trên được gửi đến các bản sao nhân bản. Do độ trễ trên mạng nên mỗi bản sao nhận được hai lệnh cập nhật theo thứ tự khác nhau, lệnh gửi thêm tiền được thực hiện trước trên bản sao thứ nhất thì số dư tài khoản sẽ là 1111 VND, còn trên bản sao thứ hai lệnh tính lãi suất được thực hiện trước thì số dư tài khoản của sẽ là 1110 VND.

Vấn đề ở đây là các thao tác cập nhật phải được thực hiện theo cùng một thứ tự trên tất cả các bản sao thì mới có thể đạt được tính nhất quán, dù cho giá trị có thể khác nhau phụ thuộc vào việc lệnh nào được thực hiện trước. Những trường hợp như vậy đòi hỏi phải đảm bảo truyền thông theo nhóm có thứ tự, cách đơn giản là gán nhãn thời gian Lamport cho mỗi câu lệnh cập nhật trước khi gửi đến các bản sao, tầng trung gian trên mỗi bản sao có nhiệm vụ sắp xếp lại theo thứ tự đã phân phát trước khi gửi cho ứng dụng xử lý.



Hình 4.7 Cập nhật phân tán

Qui trình truyền thông nhóm theo thứ tự hàng đợi được thực hiện bằng cách gán nhãn thời gian Lamport cho mỗi thông điệp trước khi gửi, giả sử không bị mất thông điệp nào. Tầng trung gian của tiến trình nhận sẽ đưa các thông điệp vào vùng đệm cục bộ, sau đó sẽ gửi thông điệp xác nhận đến tất cả các tiến trình gửi, tất nhiên thông điệp xác nhận cũng được gán nhãn thời gian Lamport và giá trị của nó luôn lớn hơn nhãn thời gian trong thông điệp nhận được, cuối cùng tất cả các tiến trình sẽ có cùng bản sao nhưng trong vùng đệm cục bộ.

Tầng trung gian của tiến trình nhận có thể chuyển thông điệp lấy từ vùng đệm đến tầng ứng dụng để xử lý khi và chỉ khi nó đã được tất cả các tiến trình trong nhóm xác nhận. Để đảm bảo yêu cầu truyền thông theo thứ tự hàng đợi, nghĩa là tuân thủ thứ tự của bên gửi đã qui định, các thông điệp trong vùng đệm phải được sắp xếp theo nhãn thời gian chính kèm. Sau khi thông điệp được chuyển đến tầng ứng dụng để xử lý thì có thể xóa khỏi vùng đệm, như vậy đã đạt được tính nhất quán theo thứ tự hàng đợi.

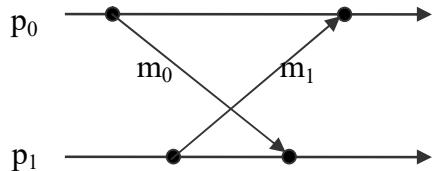
Truyền thông nhóm theo thứ tự hàng đợi là giải pháp quan trọng cho các dịch vụ nhân bản, các bản sao đạt được tính nhất quán bằng cách thực hiện các thao tác giống nhau theo cùng một thứ tự ở mọi nơi, tất nhiên với giả thiết không sử dụng các hàm thời gian thực. Các bản sao tuân thủ qui tắc chuyển đổi giống nhau trong cùng một máy trạng thái hữu hạn nên nó còn được gọi là nhân bản máy trạng thái.

#### 4.2.2 Đồng hồ vector

Đồng hồ Lamport mới chỉ đề cập đến kết quả mà chưa nói đến nguyên nhân, nếu sự kiện a xảy ra trước sự kiện b thì nhãn thời gian xảy ra sự kiện a phải nhỏ hơn nhãn thời gian xảy ra sự kiện b, nghĩa là  $C(a) < C(b)$ . Sẽ chưa thể kết luận về mối quan hệ giữa hai sự kiện nếu biết giá trị nhãn thời gian của chúng, nếu  $C(a) < C(b)$  và chúng đều

là hai sự kiện trong một tiến trình hoặc của cùng một thông điệp thì có thể cam kết  $a \rightarrow b$ , ngược lại thì không thể cam kết sự kiện nào xảy ra trước.

Ký hiệu  $T_{\text{snd}}(m_i)$  là nhãn thời gian gửi thông điệp  $m_i$  và  $T_{\text{rev}}(m_i)$  là nhãn thời gian nhận được thông điệp đó, hiển nhiên ta có  $T_{\text{snd}}(m_i) < T_{\text{rev}}(m_i)$  bởi vì chúng đều là các sự kiện liên quan tới thông điệp  $m_i$ . Nếu  $T_{\text{snd}}(m_i) < T_{\text{rev}}(m_j)$  thì không thể có kết luận gì về tính nhân quả giữa hai thông điệp  $m_i$  và  $m_j$ , ví dụ trên hình 4.8, các tiến trình  $p_0$  và  $p_1$  gửi thông điệp cho nhau trước khi nhận được thông điệp, không thể khẳng định thông điệp  $m_0$  hay  $m_1$  được gửi trước.



Hình 4.8 Tính nhân quả trong truyền thông điệp liên tiến trình

Vấn đề nằm ở chỗ đồng hồ Lamport không nắm bắt được quan hệ nhân quả, đó là lý do vì sao phải đưa ra khái niệm đồng hồ vector. Trong một số trường hợp, áp dụng đồng hồ vector có thể đưa ra kết luận về tính nhân quả của hai sự kiện trên, do đó người ta nói đồng hồ vector là nhân hóa các sự kiện phi nhân quả. Để theo dõi quan hệ nhân quả thì chỉ cần gán cho mỗi sự kiện một tên duy nhất, chẳng hạn tổ hợp định danh tăng tuần tự của các tiến trình, ký hiệu  $p_k$  là sự kiện thứ  $k$  trong tiến trình  $P$ . Vấn đề nhân quả sẽ được giải quyết bằng việc theo dõi lịch sử hoạt động, đó là những sự kiện được ghi lại trong bộ nhớ ổn định khi thực hiện các thao tác trên mỗi tiến trình.

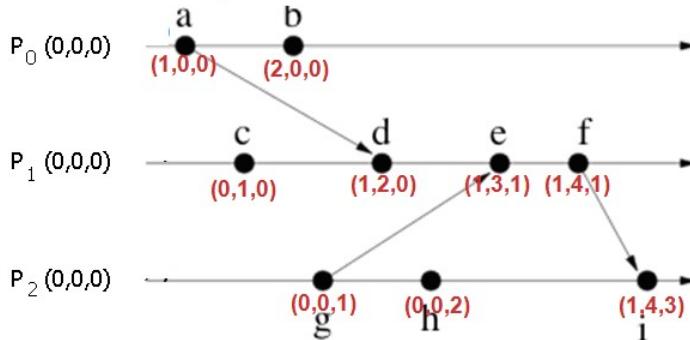
Nếu sử dụng chỉ mục cho mỗi tiến trình thì có thể biểu diễn lịch sử nhân quả dưới dạng vector, số lượng phần tử của vector bằng số lượng tiến trình và mỗi phần tử tương ứng với một tiến trình, giá trị ban đầu của các phần tử thường đặt bằng 0. Mỗi tiến trình  $P_i$  duy trì một vector  $VC_i$ , giá trị mỗi phần tử cho biết số lượng sự kiện đã xảy ra trên tất cả các tiến trình của hệ thống. Giá trị  $VC_i[i]$  thể hiện số sự kiện đã xảy ra cho đến thời điểm hiện tại trên tiến trình  $P_i$ , nếu  $VC_i[j] = k$  thì  $P_i$  hiểu rằng đã có  $k$  sự kiện xảy ra trên tiến trình  $P_j$ .

Tính chất đầu tiên được duy trì bởi việc tăng  $VC_i[i]$  thêm một đơn vị khi sự kiện mới xảy ra ở  $P_i$ , tính chất thứ hai được duy trì bằng cách kết hợp vector nhãn thời gian hiện hành cùng với thông điệp nhận được. Mỗi khi có sự kiện mới xảy ra ở tiến trình  $P_i$  thì phải tăng  $VC_i[i]$  và phải đảm bảo vector này được gửi cùng thông điệp suốt trong quá trình. Đồng hồ vector sẽ được điều chỉnh khi xảy ra sự kiện nào đó, có thể đó là sự kiện gửi hay nhận thông điệp và cũng có thể chỉ là những sự kiện nội tại bên trong mỗi tiến trình.

Trước khi tiến trình  $P_i$  gửi thông điệp, giá trị phần tử tương ứng với tiến trình sẽ được tăng thêm một đơn vị, nghĩa là  $VC_i[i] \leftarrow VC_i[i] + 1$ , điều đó thể hiện đã có một sự kiện xảy ra. Khi tiến trình  $P_i$  gửi thông điệp  $m$  cho tiến trình  $P_j$ , nó đính kèm nhãn thời gian vector  $ts(m) = VC_i$ . Tiến trình  $P_j$  nhận được thông điệp, đó cũng là sự kiện nên giá trị

của phần tử tương ứng với tiến trình này cũng được tăng thêm một đơn vị, bóc tách nhãn thời gian trong thông điệp nhận được, giá trị nhãn thời gian mới sẽ là giá trị lớn nhất khi so sánh lần lượt từng phần tử của hai vector. Giá trị mới của phần tử thứ k trong nhãn thời gian vector của tiến trình  $P_j$  sẽ là  $V_j[k] \leftarrow \max\{V_j[k], ts(m)[k]\}$ .

Giá trị  $ts(m)[i]$  biểu thị  $ts(m)[i]-1$  sự kiện đã xảy ra trên tiến trình  $P_i$  trước khi gửi thông điệp, như vậy tiến trình  $P_j$  biết được số lượng sự kiện xảy ra không những trên tiến trình  $P_i$  mà còn cả trên các tiến trình khác. Nói cách khác, nhãn thời gian  $ts(m)$  thể hiện số lượng sự kiện trong tất cả các tiến trình xảy ra trước khi tiến trình  $P_j$  gửi thông điệp m.



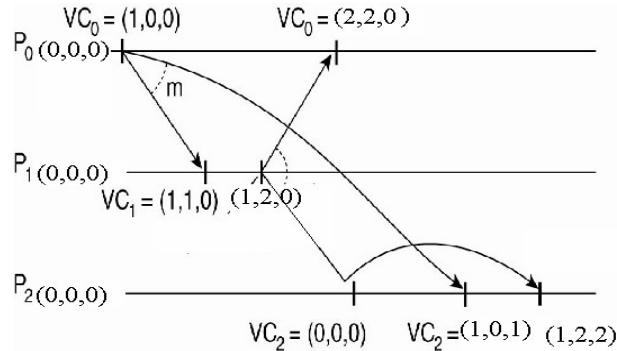
Hình 4.9 Cập nhật nhãn thời gian vector

Hình 4.9 minh họa quá trình điều chỉnh đồng hồ vector trong hệ thống gồm ba tiến trình, ký hiệu  $P_0$ ,  $P_1$  và  $P_2$ . Mỗi tiến trình xây dựng một vector gồm ba phần tử, mỗi phần tử tương ứng với nhãn thời gian của một tiến trình, ban đầu giá trị của tất cả các phần tử đều bằng 0. Tiến trình  $P_0$  gửi thông điệp cho tiến trình  $P_1$ , đồng hồ vector tăng giá trị của phần tử tương ứng với tiến trình của nó thêm 1 đơn vị, do đó nhãn thời gian vector của nó là  $(1,0,0)$ , giá trị này được đính kèm thông điệp để gửi đến tiến trình  $P_1$ . Tiến trình  $P_1$  thực hiện sự kiện nội tại c, phần tử tương ứng với tiến trình  $P_1$  được tăng thêm một đơn vị, do đó nhãn thời gian vector là  $(0,1,0)$ . Nhận được thông điệp từ tiến trình  $P_0$ , phần tử tương ứng với tiến trình  $P_1$  cũng được tăng thêm một đơn vị và giá trị mới là  $(0,2,0)$ . Tiến trình  $P_1$  bóc tách thời gian  $(1,0,0)$  đính kèm với thông điệp nhận được, duyệt từng phần tử của các nhãn thời gian  $(1,0,0)$  và  $(0,2,0)$  để lấy giá trị lớn nhất,  $\max\{(1,0,0), (0,2,0)\}$  là  $(1,2,0)$ , đó là giá trị mới của đồng hồ vector trên tiến trình  $P_1$ , quá trình tương tự như vậy cũng được thực hiện cho các sự kiện khác xảy ra trên cả ba tiến trình.

Để xác định mối tương quan giữa các sự kiện khi biết nhãn thời gian vector của chúng thì chỉ cần so sánh giá trị của chúng. Nhắc lại kiến thức về so sánh vector, cho hai vector  $V$  và  $W$ , chỉ có thể so sánh nếu số lượng phần tử của chúng bằng nhau, khi đó các mệnh đề sau là đúng:

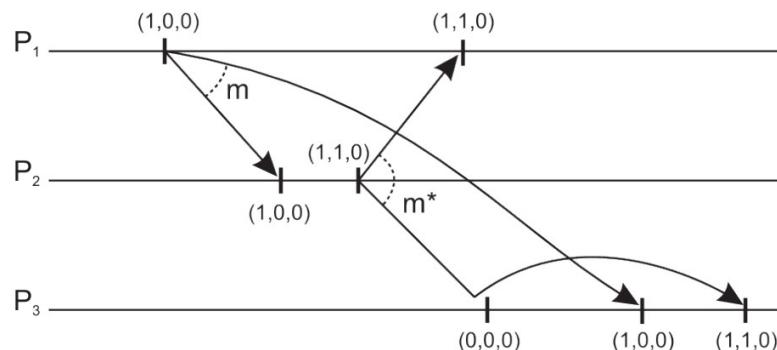
- **Ngang bằng:**  $V = W$  nếu với mọi  $i$ ,  $V[i] = W[i]$ , ví dụ:  $(3,2,4) = (3,2,4)$
- **Nhỏ hơn hoặc bằng:**  $V \leq W$  nếu với mọi  $i$ ,  $V[i] \leq W[i]$ , ví dụ:  $(2,2,3) \leq (3,2,4)$ ,  $(3,2,4) \leq (3,2,4)$
- **Nhỏ hơn:**  $V < W$  nếu  $V \leq W$  nhưng  $V \neq W$ , ví dụ:  $(2,2,3) < (3,2,4)$
- **Không thể so sánh:**  $V \parallel W$  nếu  $!(V \leq W)$  và  $!(W \leq V)$ , ví dụ:  $(3,2,4) \parallel (4,1,4)$

Sử dụng đồng hồ vector, giờ đây có thể đảm bảo tính nhân quả trong truyền thông, nghĩa là một thông điệp được gửi đi khi và chỉ khi đã nhận được tất cả các thông điệp trước nó, thậm chí đó là những thông điệp được gửi từ bất kỳ tiến trình nào vào bất kỳ thời gian nào. Tầng trung gian trì hoãn việc chuyển thông điệp lên tầng ứng dụng, sẽ lưu giữ thông điệp trong một khoảng thời gian nhất định để sắp xếp chúng theo thứ tự nhân quả.



Hình 4.10 Truyền thông nhân quả và nhãn thời gian vector

Ví dụ, hệ thống gồm ba tiến trình  $P_0$ ,  $P_1$  và  $P_2$  thực hiện chuyển thông điệp theo nhóm nhân quả,  $P_0$  gửi thông điệp đến các thành viên khác,  $P_1$  nhận được thông điệp sau đó gửi theo nhóm và thông điệp này đến  $P_2$  sớm hơn thông điệp được gửi từ  $P_0$ . Thời điểm  $P_2$  nhận được thông điệp từ  $P_1$ , nhãn thời gian của nó vẫn là  $(0,0,0)$  trong khi nhãn thời gian của thông điệp do  $P_1$  gửi đến là  $ts(m)=(1,2,0)$ , kiểm tra nhãn thời gian vector, theo luật nhân quả nó hiểu rằng cần phải chờ thông điệp từ  $P_0$ . Nhận được thông điệp từ  $P_0$  với nhãn thời gian  $ts(m)=(1,0,0)$  nó điều chỉnh nhãn thời gian  $(1,0,1)$  lấy từ vùng đệm thông điệp từ  $P_1$  gửi đến để xử lý, nhãn thời gian được gán sẽ là  $(1,2,2)$ .



Hình 4.11 Truyền thông nhân quả và nhãn thời gian vector cải tiến

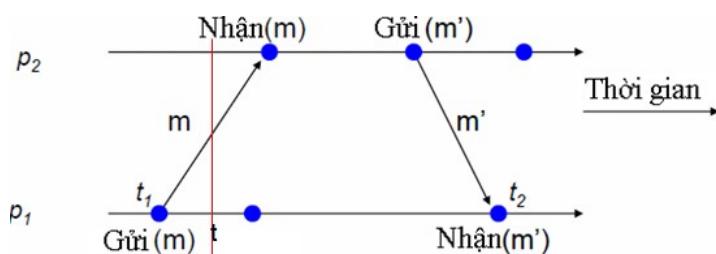
Phần mềm trung gian sắp xếp các thông điệp theo thứ tự nhân quả chỉ dựa thuận tiện vào nhãn thời gian đánh kèm, cách tiếp cận này có một số hạn chế nhất định dẫn đến các vấn đề về hiệu năng. Nếu điều chỉnh đồng hồ vector theo cách truyền thống, tiến trình sẽ mất rất nhiều thời gian để quyết định thứ tự nhân quả của các sự kiện. Một biến thể khác cho giải thuật, mỗi tiến trình không tăng giá trị phần tử của mình thêm một đơn vị nếu đó là sự kiện nội tại trong tiến trình hoặc nhận thông điệp.

Khi tiến trình  $P_j$  nhận được một thông điệp  $m$  từ tiến trình  $P_i$  với nhãn thời gian vector  $ts(m)$ , nó sẽ tạm thời đưa thông điệp vào vùng đệm, thông điệp sẽ được chuyển lên tầng ứng dụng nếu  $ts(m)[i] = VC_j[i] + 1$  và với mọi  $k \neq i$  thì  $ts(m)[k] \leq VC_j[k]$ . Điều kiện thứ nhất thể hiện  $m$  là thông điệp tiếp theo mà  $P_j$  mong nhận được từ tiến trình  $P_i$ , điều kiện thứ hai cho thấy  $P_j$  đã xử lý tất cả các thông điệp đã được  $P_i$  chuyển đến trước khi nhận được thông điệp  $m$ .

Hình 4.11 thể hiện nhãn thời gian vector của các tiến trình theo cách cải tiến này, nhận được thông điệp từ tiến trình  $P_0$ , tiến trình  $P_1$  không tăng giá trị của phần tử thứ hai trong vector, như vậy đồng hồ vector trên tiến trình  $P_1$  sẽ có giá trị là  $(1,0,0)$ , khi quang bá thông điệp thì đồng hồ vector của nó có giá trị là  $(1,1,0)$ . Trên tiến trình  $P_2$ , nhãn thời gian vector của nó là  $(0,0,0)$  khi nhận được thông điệp  $P_1$  với nhãn thời gian đính kèm  $(1,1,0)$ , quyết định chuyển lên tầng ứng dụng hay không phụ thuộc vào việc kiểm tra hai điều kiện trên. Điều kiện thứ nhất thỏa mãn vì  $ts(m)[1] = 1$  và  $VC_2[1] + 1 = 0 + 1 = 0$ . Điều kiện thứ hai với mọi  $k \neq 1$ , xét  $k=0$  ta có  $ts(m)[0]=1$  trong khi đó  $VC_j[0] = 0$ , không thỏa mãn  $ts(m)[k] \leq VC_j[k]$ , do đó sẽ không chuyển lên tầng ứng dụng. Nhận được thông điệp từ tiến trình  $P_0$  với nhãn thời gian  $(1,0,0)$ , điều kiện thứ nhất thỏa mãn, điều kiện thứ hai chỉ cần xét  $k=\{1,2\}$ , giá trị của  $ts(m)[k]$  và  $VC_j[k]$  đều bằng 0, do đó thông điệp được chuyển lên tầng ứng dụng, đồng hồ vector của tiến trình  $P_2$  điều chỉnh thành  $(1,0,0)$ . Tiến trình  $P_2$  lấy thông điệp nhãn thời gian  $(1,1,0)$  từ vùng đệm, giá trị  $VC_j[0]$  lúc này bằng 1, thỏa mãn điều kiện thứ hai nên thông điệp này được chuyển lên tầng ứng dụng để xử lý.

#### 4.2.3 Trạng thái toàn cục

Tại một thời điểm bất kỳ, trạng thái toàn cục của hệ thống phân tán được đặc trưng bởi trạng thái của từng tiến trình và các thông điệp đang lưu chuyển trên mạng. Một trong những phương pháp để xác định trạng thái toàn cục là sử dụng lát cắt, lát cắt mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiến trình, bằng cách này nó có thể kiểm tra tất cả các thông điệp nhận đều tương ứng với các thông điệp gửi được ghi lại trên lát cắt.

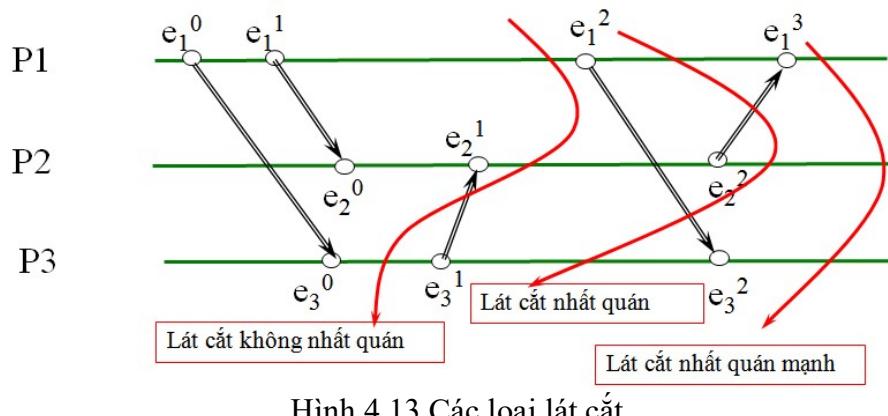


Hình 4.12 Lát cắt trong hệ thống phân tán

Hình 4.12 thể hiện ý tưởng lát cắt, hệ thống có thể qui định thời điểm  $t$  tất cả các tiến trình đồng loạt ghi trạng thái, nhưng rất tiếc hệ thống lại không có đồng hồ tuyệt đối. Cũng có thể triển khai bằng cách tiến trình điều phối gửi thông điệp yêu cầu ghi trạng thái đến tất cả các thành viên, tuy nhiên trễ kênh truyền làm cho giải pháp này không thực hiện được. Cả hai giải pháp đều không thể ghi nhận trạng thái của các kênh truyền, không thể biết có bao nhiêu thông điệp và vị trí của chúng.

Khái niệm lát cắt toàn cục không hề tồn tại trong thực tế, vì vậy cần phải nói lỏng điều kiện, thay cho trạng thái toàn cục của hệ thống sẽ sử dụng trạng thái toàn cục nhất quán. Mỗi tiến trình đều ghi lại sự kiện xảy ra trong quá trình hoạt động, thông tin bao gồm nhãn thời gian vector và các thao tác đã thực hiện. Nếu tiến trình  $P_i$ , ghi nhận các sự kiện  $e_i^0, e_i^1, \dots$  thì toàn bộ hoạt động của tiến trình này đều lưu lại lịch sử các sự kiện  $h_i = \langle e_i^0, e_i^1, \dots \rangle$ . Tiến sử tiến trình  $P_i^k$  được định nghĩa là tất cả những sự kiện từ khi bắt đầu hoạt động cho đến sự kiện thứ k, nghĩa là  $h_i^k = \langle e_i^0, e_i^1, \dots, e_i^k \rangle$ , trạng thái tiến trình  $S_i^k$  là trạng thái của tiến trình  $P_i$  ngay trước khi xảy ra sự kiện thứ k.

Sử dụng lịch sử các sự kiện trên mỗi tiến trình, khái niệm lát cắt giờ đây được định nghĩa lại, nó mô tả sự kiện cuối cùng mà sự kiện này được ghi lại cho mỗi tiến trình. Cho tập các tiến trình  $P_1, \dots, P_i, \dots$ , lịch sử toàn cục của hệ thống được hiểu là lịch sử sự kiện xảy ra trên tất cả các tiến trình, nghĩa là  $H = \cup_i (h_i)$ . Trạng thái toàn cục được định nghĩa là hợp trạng thái của các tiến trình,  $S = \cup_i (S_i^k)$ . Lát cắt C phải nằm trong lịch sử toàn cục của hệ thống, nghĩa là  $C \subseteq H = h_1^{c1} \cup h_2^{c2} \cup \dots \cup h_n^{cn}$ , ranh giới của lát cắt là tập hợp mỗi sự kiện lấy từ các tiến trình  $C = \{e_i^{ci}, i = 1, 2, \dots n\}$ .



Hình 4.13 Các loại lát cắt

Hình 4.13 thể hiện ba loại lát cắt, trạng thái toàn cục S của hệ thống sẽ tương ứng với từng loại lát cắt. Lát cắt C là nhất quán khi và chỉ khi với mọi sự kiện e nằm trong lát cắt, nếu sự kiện f là nguyên nhân sự kiện e thì f cũng phải nằm trong lát cắt, nghĩa là nếu  $\forall e \in C$  và tồn tại  $f \rightarrow e$  thì  $f \in C$ . Lát cắt là nhất quán mạnh khi và chỉ khi mọi sự kiện trong các quan hệ nhân quả đều thuộc về lát cắt, nghĩa là với mọi quan hệ  $f \rightarrow e$  thì  $f, e \in C$ . Các trường hợp khác gọi là lát cắt không nhất quán, chỉ cần một sự kiện nằm trong lát cắt nhưng nguyên nhân của nó không nằm trong lát cắt.

Lát cắt nhất quán mạnh thể hiện tất cả các sự kiện gửi và nhận thông điệp đều đã được lưu trong lịch sử tiến, lát cắt nhất quán thể hiện đã lưu sự kiện gửi nhưng chưa lưu sự kiện nhận thông điệp, lát cắt không nhất quán thể hiện đã lưu sự kiện nhận nhưng chưa lưu sự kiện gửi thông điệp. Lát cắt đóng vai trò quan trọng trong việc phục hồi hệ thống sau khi lỗi xảy ra, cách xác định nó dựa trên nhãn thời gian vector được lưu lại trong lịch sử tiến trình.

### 4.3 Các giải thuật loại trừ tương hỗ phân tán

Tương tranh là một trong những vấn đề trong các hệ thống phân tán, nếu nhiều tiến trình cùng truy nhập tài nguyên có thể dẫn đến hiện tượng xung đột, dẫn đến suy giảm hiệu năng hoặc bỏ sót các yêu cầu xử lý. Nguyên tắc của các giải thuật dựa trên quan điểm loại trừ lẫn nhau, đó là biện pháp kiểm soát tương tranh để ngăn chặn xung đột. Để giải quyết vấn đề này, một số giải thuật loại trừ tương hỗ đã được đề xuất dựa trên phương pháp sử dụng thẻ bài hoặc cấp quyền truy nhập. Các giải thuật đều đáp ứng yêu cầu công bằng cho mọi tiến trình, mỗi tiến trình đều có quyền sử dụng tài nguyên sau một khoảng thời gian chờ đợi hữu hạn.

Phương pháp cấp quyền truy nhập hoạt động theo hai bước, bước thứ nhất gửi yêu cầu đề nghị được cấp quyền sử dụng, sau khi được cấp quyền mới chuyển sang bước truy nhập tài nguyên. Phương pháp này tiềm ẩn nguy cơ khóa chết, đó là hiện tượng các tiến trình chờ nhau để được sử dụng tài nguyên, hậu quả là một yêu cầu nào đó sẽ bị bỏ qua hoặc thậm chí dẫn đến hiện tượng treo hệ thống.

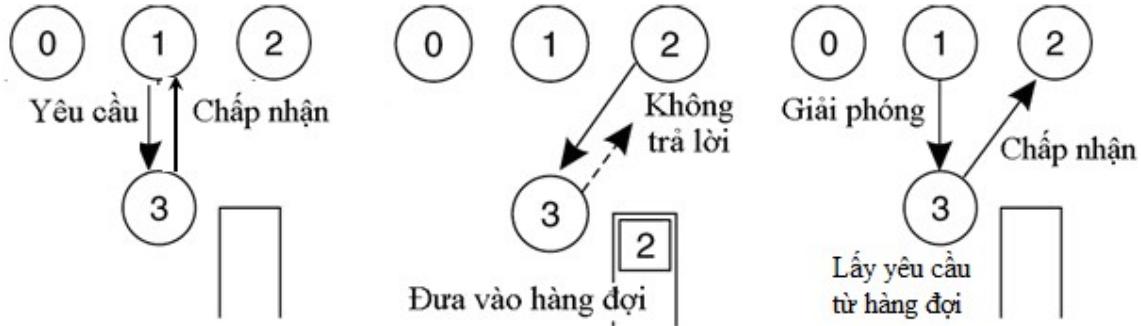
Bản chất của phương pháp thẻ bài là duy trì một thông điệp duy nhất gọi là thẻ bài di chuyển trong nhóm các thành viên của hệ thống. Thành viên nào nắm giữ thẻ bài thì có quyền truy nhập tài nguyên, nếu không có nhu cầu truy nhập tài nguyên thì sẽ chuyển cho thành viên kế tiếp. Phương pháp này luôn bảo đảm tránh được tương tranh vì tại một thời điểm chỉ có tối đa một thành viên nắm giữ thẻ bài, nó cũng giải quyết triệt để hiện tượng khóa chết, tuy nhiên nếu thẻ bài bị mất thì không có thành viên nào được sử dụng tài nguyên, hệ thống phải tái tạo một thẻ bài mới.

#### 4.3.1 Giải thuật tập trung

Giải thuật tập trung đòi hỏi phải có một tiến trình đóng vai trò điều phối để quản lý việc truy nhập tài nguyên, có thể chỉ định tiến trình điều phối bằng phương pháp tĩnh hoặc động. Nếu sử dụng phương pháp động thì phải áp dụng giải thuật bầu chọn, ví dụ bầu chọn tiến trình có số hiệu định danh cao nhất. Khi một tiến trình muốn truy nhập tài nguyên, nó sẽ gửi thông điệp xin được cấp quyền đến tiến trình điều phối. Nếu không có tiến trình nào đang sử dụng tài nguyên thì tiến trình điều phối sẽ phản hồi thông điệp cấp phép, ngược lại sẽ gửi chuyển yêu cầu vào hàng đợi. Khi tiến trình sử dụng xong tài nguyên, nó sẽ gửi thông điệp đến tiến trình điều phối thông báo trả lại quyền truy nhập, nếu hàng đợi không rỗng, tiến trình điều phối sẽ gửi thông điệp cấp phép truy nhập tài nguyên cho tiến trình đầu tiên trong hàng đợi.

Hình 4.14 minh họa hệ thống gồm bốn tiến trình, định danh của chúng từ 0 đến 3, tiến trình số 3 được bầu chọn là tiến trình điều phối. Tiến trình số 1 muốn sử dụng tài nguyên, nó gửi yêu cầu đến tiến trình số 3, do tài nguyên đang rỗi nên tiến trình số 3 gửi thông điệp cấp phép cho tiến trình số 1. Trong khi tiến trình số 1 đang sử dụng tài nguyên thì yêu cầu từ tiến trình số 2 gửi đến, tiến trình số 3 sẽ chuyển yêu cầu của tiến trình số 2 vào hàng đợi. Chờ đến khi nhận được thông điệp giải phóng tài nguyên từ tiến trình số 1, tiến trình số 3 sẽ lấy phần tử đầu tiên trong hàng đợi, đó là yêu cầu của tiến trình số 2, vì vậy nó gửi thông điệp cấp phép sử dụng tài nguyên cho tiến trình số 2.

Tại một thời điểm sẽ chỉ có tối đa một tiến trình được phép truy nhập tài nguyên, giải thuật khá đơn giản, nó chỉ cần ba thông điệp: yêu cầu, cấp phép và giải phóng. Tính công bằng của giải thuật được đảm bảo, yêu cầu đến trước sẽ được phục vụ trước. Giải thuật này cũng có hai nhược điểm cơ bản, nếu tiến trình điều phối bị hỏng thì sẽ không có một tiến trình nào được cấp phép, hoặc khi một tiến trình đang sử dụng mà bị phong tỏa lỗi thì tiến trình điều phối sẽ không bao giờ nhận được thông điệp giải phóng tài nguyên.



Hình 4.14 Nguyên lý giải thuật tập trung

Nếu tiến trình điều phối nhận được thông điệp trong khi tài nguyên đang bận thì nó không phản hồi thông điệp, do đó tiến trình gửi không biết khi nào mới được phục vụ. Qui mô hệ thống ngày càng mở rộng, số lượng tiến trình tăng lên, nếu chỉ có một tiến trình điều phối thì sẽ xuất hiện hiện tượng thắt cổ chai, hiệu năng hệ thống sẽ bị suy giảm.

#### 4.3.2 Giải thuật không tập trung

Giải thuật tập trung chỉ có một thành phần điều phối, đó là cách tiếp cận nghèo nàn, các thành viên chỉ có duy nhất một lựa chọn. Để khắc phục nhược điểm này, mỗi tài nguyên sẽ được nhân bản N lần và mỗi bản sao sẽ có tiến trình điều phối riêng để kiểm soát truy nhập. Một tiến trình muốn truy nhập tài nguyên thì phải gửi yêu cầu đến N tiến trình điều phối đang hoạt động và sẽ được cấp quyền truy nhập khi và chỉ khi nhận được số phiếu đồng ý lớn hơn  $N/2$ . So với giải thuật tập trung, giải thuật này ít bị lỗi hơn, gọi là xác suất bị lỗi của mỗi tiến trình điều phối, xác suất  $k$  trong  $m$  tiến trình điều phối bị lỗi sẽ là:

$$P[k] = \binom{m}{k} p^k (1-p)^{m-k}$$

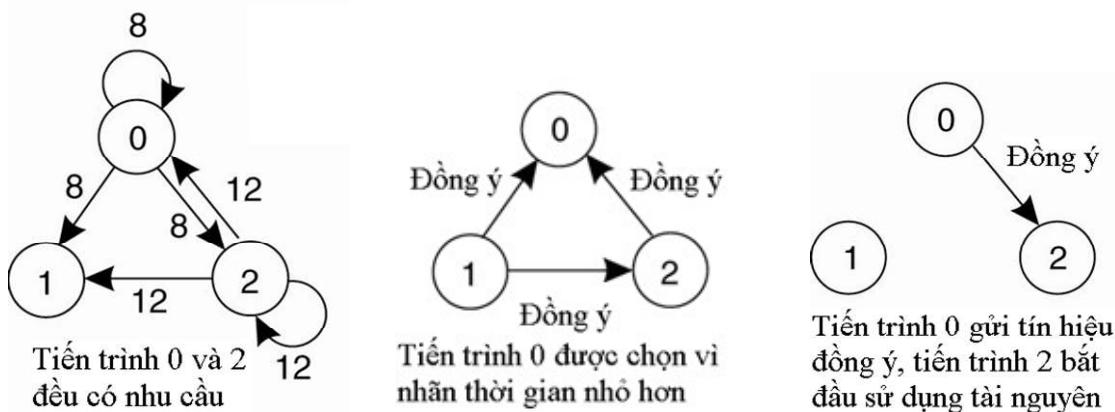
Nếu tiến trình nhận được số phiếu đồng ý nhỏ hơn hoặc bằng  $N/2$ , tức là quyền truy nhập bị từ chối, nó sẽ chờ đợi trong một khoảng thời gian trước khi gửi lại yêu cầu. Nếu nhiều tiến trình cạnh tranh để được truy nhập, các tiến trình bị từ chối đều phải chờ đợi sau đó gửi lại yêu cầu, chu kỳ đó liên tục xảy ra đến mức không một tiến trình nào có thể nhận được đủ số phiếu đồng ý khiến tài nguyên không được sử dụng. Có thể giải quyết vấn đề này bằng cách qui định thời gian chờ đợi khác nhau cho mỗi tiến trình sau mỗi lần bị từ chối. Mỗi tiến trình lưu biến đếm  $R$ , giá trị của nó tăng thêm một đơn vị sau

mỗi lần bị từ chối, thời gian chờ đợi được tính theo công thức  $K*T$ , trong đó  $T$  là hằng số thông nhất cho tất cả các tiến trình trong hệ thống,  $K$  là giá trị ngẫu nhiên trong khoảng từ 0 đến  $2^{\min(R,10)} - 1$ .

#### 4.3.3 Giải thuật phân tán

Khác với giải thuật tập trung và không tập trung, giải thuật phân tán không có tiến trình điều phối. Trước khi chính thức sử dụng tài nguyên thì tiến trình phải được cấp phép để được vào vùng tới hạn, đó là vùng truy nhập độc quyền dành cho một tiến trình. Nếu một tiến trình muốn vào vùng tới hạn, trước hết nó sẽ tạo ra một nhãn thời gian và gắn vào thông điệp gửi đến tất cả các tiến trình khác, nên sử dụng nhãn thời gian Lamport mở rộng. Tiến trình chỉ được phép vào vùng tới hạn để sử dụng tài nguyên nếu nhận được sự chấp thuận của tất cả các tiến trình trong hệ thống.

Tiến trình nhận được thông điệp yêu cầu vào vùng tới hạn, nếu nó không ở trong vùng tới hạn và cũng không có nhu cầu vào vùng tới hạn thì sẽ phản hồi thông điệp chấp nhận. Nếu tiến trình đang ở trong vùng tới hạn thì đưa yêu cầu vào hàng đợi, chờ đến khi thoát khỏi vùng tới hạn sẽ phản hồi thông điệp chấp nhận cho tất cả các tiến trình đã gửi và xóa toàn bộ hàng đợi. Nếu tiến trình cũng có nhu cầu vào vùng tới hạn thì áp dụng luật nhân quả bằng cách so sánh nhãn thời gian của mình với nhãn thời gian gắn trong thông điệp nhận được. Nếu nhãn thời gian của tiến trình lớn hơn nhãn thời gian đính kèm trong thông điệp thì phải phản hồi bằng thông điệp chấp nhận, ngược lại sẽ chuyển yêu cầu vào hàng đợi, chờ đến khi ra khỏi vùng tới hạn hoặc không có nhu cầu vào vùng tới hạn thì sẽ lấy tất cả thông điệp từ hàng đợi và phản hồi đồng ý cho các tiến trình tương ứng và đồng thời xóa các thông điệp này trong hàng đợi.



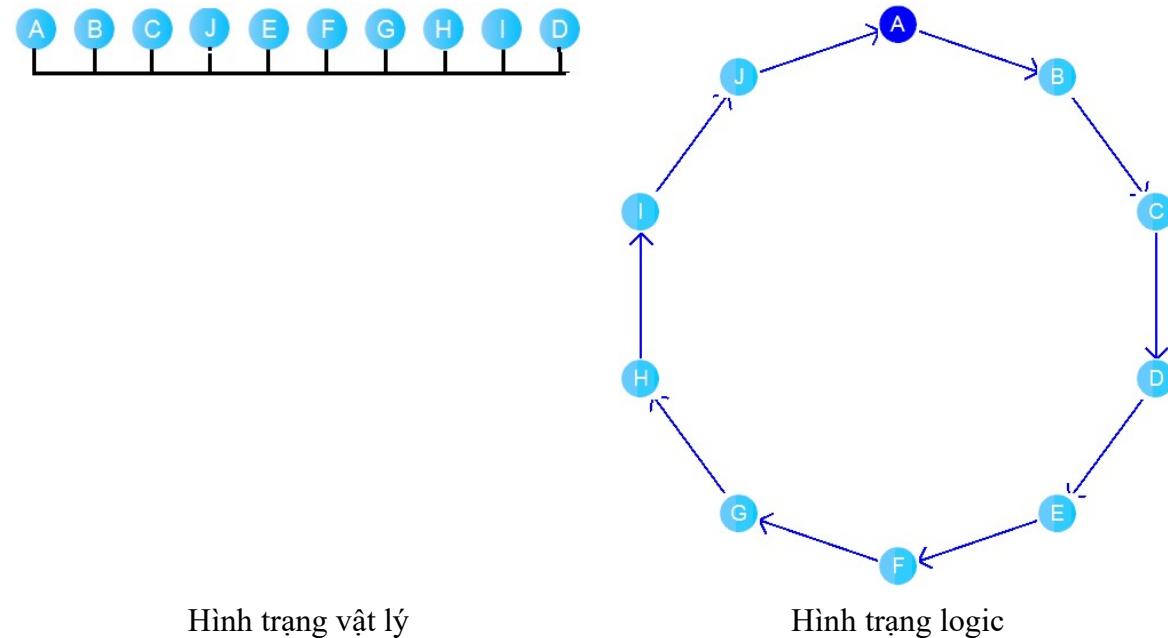
Hình 4.15 Nguyên lý giải thuật phân tán

Hình 4.15 minh họa hoạt động của giải thuật phân tán trong hệ thống gồm ba tiến trình, tiến trình số 1 không có nhu cầu sử dụng tài nguyên. Tiến trình số 0 và tiến trình số 2 đều có nhu cầu sử dụng tài nguyên, chúng đính kèm nhãn thời gian của mình vào thông điệp cấp phép truy nhập gửi đến các tiến trình khác. Nhận được thông điệp, tiến trình số 1 phản hồi thông điệp chấp thuận cho cả hai tiến trình số 0 và 2, tiến trình số 2 nhận thấy nhãn thời gian 12 của nó lớn hơn nhãn thời gian 8 đính kèm thông điệp của tiến trình số 0 nên nó gửi thông điệp chấp thuận đến tiến trình số 0. Tiến trình số 0 đang có nhu cầu vào

vùng tối hạn, do đó nó chuyển yêu cầu của tiến trình số 2 vào hàng đợi, bản thân nó đã được cả hai thành viên còn lại của hệ thống chấp nhận, nghĩa là nó được cấp phép vào vùng tối hạn để truy nhập tài nguyên. Sau khi tiến trình số 0 ra khỏi vùng tối hạn, nó lấy yêu cầu từ hàng đợi và gửi thông điệp chấp nhận cho tiến trình số 2, lúc đó tiến trình số 2 mới được phép vào vùng tối hạn.

#### 4.3.4 Giải thuật thẻ bài

Giả thiết tất cả các tiến trình được sắp xếp trên một vòng tròn logic như minh họa trên hình 4.16, các tiến trình đều được gán định danh và đều biết đến các tiến trình cạnh nó. Khi hệ thống bắt đầu khởi tạo, tiến trình 0 sẽ được trao thẻ bài, thẻ bài này sẽ不停地 chuyền quanh vòng tròn logic, nó được chuyền từ tiến trình k đến tiến trình kế tiếp k+1 theo kiểu chuyền thông điệp điểm – điểm.



Hình 4.16 Nguyên lý giải thuật thẻ bài

Khi một tiến trình nhận được thẻ bài từ tiến trình liền trước, nó sẽ kiểm tra xem có cần thiết vào vùng tối hạn hay không. Nếu tiến trình không có nhu cầu truy nhập tài nguyên thì chuyển thẻ bài cho tiến trình kế tiếp, ngược lại sẽ vào vùng tối hạn, sau khi hoàn thành phần việc của mình nó sẽ trả thẻ bài cho tiến trình kế tiếp. Vấn đề lớn nhất của giải thuật này là thẻ bài có thể bị thất lạc, khi đó hệ thống sẽ phải tạo lại thẻ bài bởi vì việc dò tìm lại thẻ bài là công việc không hề đơn giản. Một tiến trình có thẻ bị lỗi sụp đổ, do đó một tiến trình có thẻ không thể chuyển thẻ bài cho tiến trình liền sau. Có thể giải quyết vấn đề này bằng cách mỗi tiến trình lưu trữ danh sách các tiến trình liền sau, nếu không thể chuyển thẻ bài cho tiến trình liền sau thì sẽ lần lượt chuyển cho tiến trình kế tiếp trong danh sách.

### 4.3.5 So sánh các giải thuật loại trừ

Để so sánh các giải thuật cần phải dựa trên số lượng thông điệp cần lưu chuyển, độ trễ và những vấn đề tiềm ẩn trong giải thuật, bảng 4.2 tóm tắt những đặc điểm cơ bản của bốn giải thuật loại trừ. Giải thuật tập trung đơn giản và hiệu quả nhất, nó chỉ cần ba thông điệp để vào, ra và giải phóng khỏi vùng tối hạn. Nếu mỗi tài nguyên được nhân bản m lần thì giải thuật phi tập trung cần tới  $3mk$  thông điệp, trong đó  $k=1,2,3\dots$  tương ứng với số lần gửi yêu cầu. Nếu hệ thống gồm n thành viên thì giải thuật phân tán cần  $2(n - 1)$  thông điệp, số lượng thông điệp cần gửi trong giải thuật thẻ bài luôn thay đổi.

**Bảng 4.2** So sánh các giải thuật loại trừ

Giải thuật	Số thông điệp	Độ trễ (số thông điệp)	Nhược điểm
Tập trung	3	2	Tiến trình điều phối lỗi
Phi tập trung	$3mk$ , $k=1,2\dots$	$2m$	Kém hiệu quả
Phân tán	$2(n - 1)$	$2(n - 1)$	Lỗi của bất kỳ tiến trình nào
Thẻ bài	$1\text{đến }\infty$	$0\text{ đến }n - 1$	Mất thẻ bài

Khoảng thời gian chờ đợi tính từ thời điểm một tiến trình muốn vào vùng tối hạn cho đến khi thực sự được cấp phép cũng thay đổi, nếu được cấp phép vào vùng tối hạn thì thời gian đó không nhỏ hơn tổng thời gian chuyển thông điệp. Trường hợp xấu nhất, các thông điệp được gửi lần lượt, tức là không bao giờ có nhiều thông điệp được truyền đi cùng một lúc. Nếu thời gian sử dụng tài nguyên ngắn, yếu tố chi phối thời gian chờ đợi là tổng số thông điệp được gửi qua hệ thống trước khi có thể cấp quyền truy cập. Nếu tài nguyên được sử dụng trong thời gian dài, yếu tố chi phối sẽ là chờ đợi các tiến trình khác để đến lượt mình.

Tất cả các giải thuật đều bị ảnh hưởng nghiêm trọng nếu xảy ra sự cố, do đó cần phải bổ sung các giải pháp để tránh làm sập toàn bộ hệ thống. Mặc dù giải thuật tập trung gấp phải vấn đề về mở rộng qui mô, tuy nhiên nó ít nhạy cảm với các sự cố lỗi, nếu tiến trình điều phối bị lỗi thì có thể bầu chọn lại, do đó nó là giải thuật được áp dụng rộng rãi nhất.

## 4.4 Các giải thuật bầu chọn

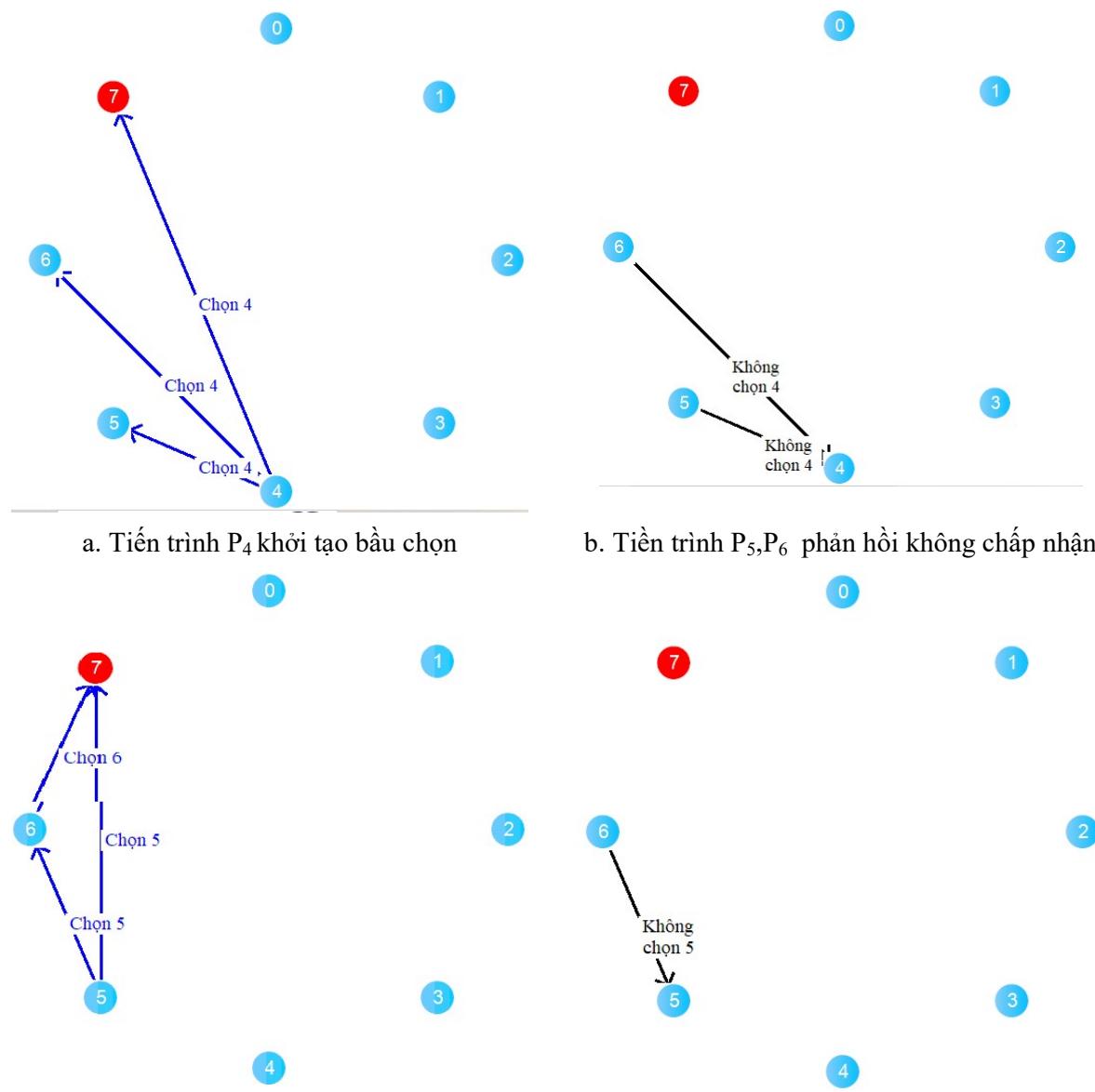
Nhiều tác vụ trong hệ thống phân tán yêu cầu phải tìm ra một tiến trình bất kỳ để thực hiện vai trò điều phối hay thực hiện vai trò đặc biệt nào đó. Như vậy, cần thiết phải có giải thuật để tìm ra một tiến trình duy nhất đóng vai trò điều phối hoặc khi tiến trình điều phối gặp lỗi thì sẽ phải có quá trình bầu chọn để tìm ra một tiến trình khác thay thế.

Tùy từng nhiệm vụ, hệ thống có thể qui định tiêu chí bầu chọn khác nhau, nó có thể là tập hợp trạng thái đang chạy của mỗi tiến trình, ví dụ tập các thông số về phần cứng hoặc thời gian trung bình xử lý mỗi yêu cầu... Bất kỳ tiến trình nào cũng có quyền ứng cử, giả thiết mỗi tiến trình được gán định danh duy nhất, tiến trình có định danh cao nhất sẽ được bầu chọn.

### 4.4.1 Giải thuật nối bợt

Giải thuật nối bợt được Garcia-Molina đề xuất vào năm 1982, giả thiết các tiến trình trong hệ thống đều biết định danh và địa chỉ của nhau, cần phải chọn một tiến trình

có định danh cao nhất. Giả sử tiến trình  $P_k$  phát hiện tiến trình điều phối bị lỗi, nó khởi tạo giải thuật bầu chọn, tự ứng cử bằng cách gửi thông điệp “Bầu chọn” đến các tiến trình có định danh cao hơn. Sau một khoảng thời gian chờ đợi, nếu không nhận được thông điệp “Không chấp nhận” từ tiến trình có định danh cao hơn thì tiến trình  $P_k$  sẽ trở thành tiến trình được bầu chọn, tiến trình  $P_k$  sẽ gửi thông điệp đến tất cả các tiến trình còn đang hoạt động để thông báo nó đã được bầu chọn. Chỉ cần nhận được một thông điệp “Không chấp nhận” từ tiến trình có định danh cao hơn, tiến trình  $P_k$  sẽ từ bỏ ý định ứng cử. Tại bất kỳ thời điểm nào, nếu tiến trình nhận được thông điệp “Bầu chọn” từ những tiến trình có định danh thấp hơn, nếu không muốn tham gia thì không cần trả lời, nếu tham gia thì phải gửi thông điệp “Không chấp nhận” và đồng thời khởi tạo lại giải thuật bầu chọn.



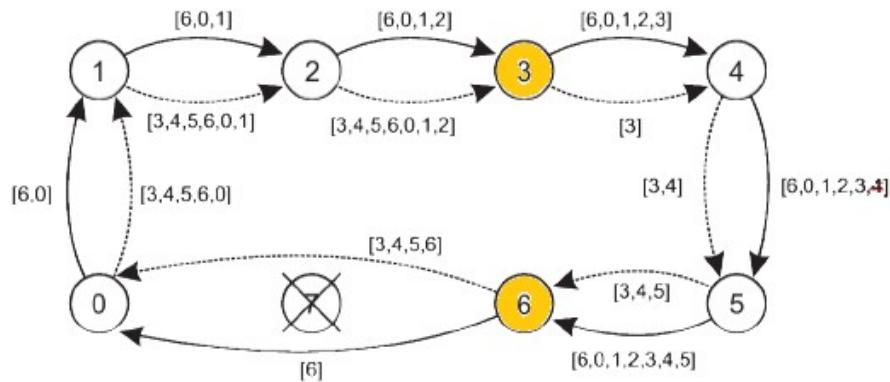
Hình 4.17 Nguyên lý giải thuật nỗi bọt

Hình 4.17 minh họa các bước thực hiện của giải thuật, giả sử hệ thống gồm 8 tiến trình với các định danh từ 0 đến 7, tiến trình P<sub>7</sub> đóng vai trò điều phối nhưng bị lỗi. Tiến trình P<sub>4</sub> phát hiện thấy tiến trình điều phối bị lỗi, nó tự ứng cử bằng cách gửi thông điệp “Bầu chọn” đến các tiến trình {P<sub>5</sub>, P<sub>6</sub>, P<sub>7</sub>}. Nhận được thông điệp yêu cầu “Bầu chọn”, tiến trình P<sub>5</sub> và P<sub>6</sub> vẫn đang hoạt động nên chúng phản hồi thông điệp “Không chấp nhận” và cả hai đồng thời khởi tạo lại quá trình bầu chọn, tiến trình P<sub>4</sub> ngay lập tức từ bỏ tranh cử. Tiến trình P<sub>5</sub> gửi thông điệp “Bầu chọn” đến tiến các tiến trình {P<sub>6</sub>, P<sub>7</sub>}, tiến trình P<sub>6</sub> gửi thông điệp “Bầu chọn” đến tiến trình P<sub>7</sub>, tất nhiên tiến trình P<sub>6</sub> sẽ gửi thông điệp “Không chấp nhận” đến tiến trình P<sub>5</sub>, do đó tiến trình P<sub>5</sub> sẽ từ bỏ tranh cử. Tiến trình P<sub>7</sub> bị lỗi nên không phản hồi, sau một thời gian chờ đợi, tiến trình P<sub>6</sub> hiểu rằng yêu cầu tự ứng cử của nó không có tiến trình nào phản đối, vì vậy nó là tiến trình được bầu chọn, tiến trình P<sub>6</sub> chỉ cần gửi thông báo đến các tiến trình khác đang hoạt động {P<sub>0</sub>, P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub>, P<sub>4</sub>, P<sub>5</sub>}.

Nếu trong quá trình bầu chọn, tiến trình P<sub>7</sub> sống lại, nó sẽ phản hồi thông điệp “Không chấp nhận” đến các tiến trình đã khởi sướng giải thuật, ít nhất sẽ có một thông điệp được chuyển đến tiến trình P<sub>6</sub>, kết quả là tiến trình P<sub>7</sub> vẫn đóng vai trò điều phối. Nếu quá trình bầu chọn đã hoàn thành, một tiến trình khác đã được bầu chọn thì tiến trình điều phối mới sống lại, nó không cần phải gửi thông điệp “Bầu chọn” cho bất kỳ tiến trình nào, trong hệ thống tồn tại hai tiến trình điều phối, đây là điểm sơ hở của giải thuật và cần phải có những biện pháp bổ sung.

#### 4.4.2 Giải thuật vòng

Giả thiết các tiến trình có một định danh duy nhất và được sắp xếp trên một vòng tròn logic, mỗi tiến trình có thể nhận biết địa chỉ của một số tiến trình liền sau. Một tiến trình bất kỳ khởi sướng giải thuật bằng cách tạo thông điệp “Bầu chọn” chứa định danh của nó sau đó gửi đến tiến trình đang hoạt động gần nhất, quá trình gửi theo một hướng nhất định, thăm dò liên tiếp trên vòng cho đến khi tìm được một tiến trình còn hoạt động. Khi nhận được thông điệp “Bầu chọn”, mỗi tiến trình sẽ thêm định danh của mình và chuyển cho tiến trình kế tiếp. Thông điệp “Bầu chọn” sẽ đi qua tất cả các tiến trình đang hoạt động và cuối cùng sẽ trở về tiến trình khởi sướng, khi đó sẽ chọn được một tiến trình có định danh cao nhất và gửi thông điệp đến tiến trình này. Tiến trình được chọn sẽ gửi thông điệp đến tất cả các tiến trình khác để thông báo kết quả bầu chọn.



Hình 4.18 Nguyên lý bầu chọn bằng giải thuật vòng

Hình 4.18 minh họa hoạt động của giải thuật vòng, giả sử tiến trình P<sub>3</sub> và P<sub>6</sub> phát hiện tiến trình điều phối P<sub>7</sub> bị lỗi, chúng đồng loạt khởi tạo giải thuật. Tiến trình P<sub>3</sub> tạo thông điệp “Bầu chọn” [3] và chuyển đến tiến trình kế tiếp P<sub>4</sub>, tiến trình P<sub>3</sub> thêm định danh của mình và chuyển thông điệp [3,4] đến tiến trình P<sub>5</sub>, tiến trình P<sub>5</sub> thêm định danh của mình và chuyển thông điệp [3,4,5] đến tiến trình P<sub>6</sub>. Tiến trình P<sub>7</sub> bị lỗi nên tiến trình P<sub>6</sub> không thể chuyển thông điệp cho nó, tiến trình P<sub>6</sub> thăm dò các tiến trình kế tiếp trong danh sách, nó chuyển thành công thông điệp [3,4,5,6] đến tiến trình P<sub>0</sub>. Quá trình tương tự đến xảy ra trên các tiến trình P<sub>1</sub> và P<sub>2</sub>, cuối cùng thông điệp [3,4,5,6,0,1,2] sẽ trở về tiến trình P<sub>3</sub>.

Nhận được bản tin chứa định danh của các tiến trình đang còn hoạt động, tiến trình P<sub>3</sub> biết định danh cao nhất, nó tạo thông điệp chứa định danh của tiến trình bầu chọn [6] và gửi đến tiến trình P<sub>4</sub>. Thông điệp được chuyển tiếp qua các tiến trình, nhận được thông điệp này, tiến trình được bầu chọn P<sub>6</sub> không cần chuyển tiếp thông điệp, nó tạo một thông điệp mới thông báo tiến trình điều phối mới và gửi đến tiến trình P<sub>0</sub>, thông điệp sẽ được chuyển tiếp đến các tiến trình khác theo vòng.

Cùng thời điểm tiến trình P<sub>3</sub> khởi sướng bầu chọn, tiến trình P<sub>6</sub> cũng thực hiện tương tự, dù có hai thông điệp bầu chọn song song tồn tại nhưng nó không làm thay đổi kết quả bầu chọn. Giải thuật vòng đã khắc phục nhược điểm của giải thuật nỗi bọt, sau khi các tiến trình hoàn thành quá trình bầu chọn, tiến trình P<sub>6</sub> gửi thông điệp đến các tiến trình khác. Nếu tiến trình P<sub>7</sub> sống lại, nó sẽ nhận được thông điệp này và tất nhiên không chấp nhận kết quả bầu chọn, giải thuật được khởi động lại.

#### **4.4.3 Bầu chọn trong môi trường không dây**

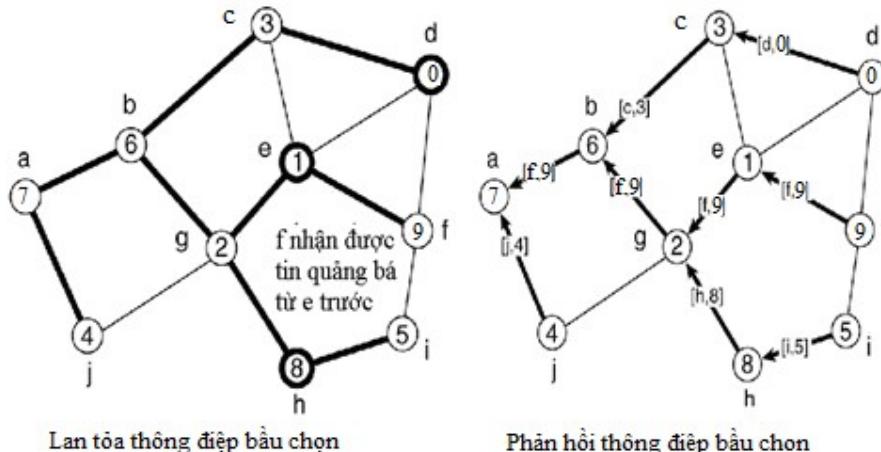
Giải thuật nỗi bọt và giải thuật vòng dựa trên giả thiết việc phân phát các thông điệp đáng tin cậy và hình trạng mạng không thay đổi, đó là những yêu cầu khó có thể đáp ứng được trong môi trường không dây. Môi trường không dây thường chịu ảnh hưởng của nhiều yếu tố về môi trường dẫn đến độ tin cậy truyền thông không cao, tồn tại nhiều đường đi đến các nút, điều đó có thể dẫn đến hiện tượng vòng lặp vô hạn. Hình 4.19 minh họa mạng không dây gồm 10 tiến trình với định danh khác nhau, khó có thể áp dụng giải thuật nỗi bọt vì một số tiến trình không thể liên lạc trực tiếp với nhau, cũng không thể sắp xếp các tiến trình để áp dụng giải thuật vòng.

Vasudevan đề xuất giải thuật bầu chọn trong môi trường không dây vào năm 2004, thay vì lựa chọn một cách chính xác như các giải thuật bầu chọn nỗi bọt hoặc giải thuật vòng thì chỉ cần chọn một thành viên tốt nhất. Ý tưởng của giải thuật là xây dựng cây tiến trình, nghĩa là về mặt logic sẽ chỉ tồn tại một đường đi duy nhất giữa các nút. Giải thuật chia làm hai giai đoạn, giai đoạn thứ nhất quảng bá thông điệp “Bầu chọn”, giai đoạn thứ hai phản hồi thông điệp “Bầu chọn”.

Bất cứ tiến trình cũng có quyền khởi tạo bầu chọn, các tiến trình không cần biết định danh của các tiến trình khác trong hệ thống. Nếu một tiến trình phát hiện tiến trình điều phối bị lỗi, nó gửi thông điệp “Bầu chọn” đến tất cả những tiến trình láng giềng, láng giềng là những tiến trình có thể kết nối trực tiếp. Nhận được thông điệp “Bầu chọn”, tiến trình phải kiểm tra xem đó có phải là thông điệp đầu tiên hay không. Nếu là thông

điệp đầu tiên thì sẽ đánh dấu tiến trình gửi là cha, sau đó tiếp tục chuyển thông điệp “Bầu chọn” đến các tiến trình láng giềng khác ngoại trừ tiến trình cha và những tiến trình đã bị từ chối làm cha. Nếu không phải là thông điệp đầu tiên thì chỉ cần phản hồi đã nhận được thông điệp, không nhận tiến trình gửi là cha và cũng cần chuyển tiếp thông điệp cho các tiến trình láng giềng khác.

Nếu tiến trình không có con thì chỉ cần gửi định danh của mình cho tiến trình cha, ngược lại nó phải so sánh định danh của mình với định danh của các con để chọn tiến trình đại diện cho nhánh của nó. Mỗi tiến trình cha sẽ phải gửi thông tin về tiến trình đại diện đến tiến trình cha, tiến trình khởi sướng xác định được tiến trình nào được bầu chọn và quảng bá thông điệp kết quả bầu chọn cho tất cả các thành viên của hệ thống.



Hình 4.19 Bầu chọn trong môi trường không dây

Ví dụ trên hình 4.19, giả sử tiến trình  $a_7$  khởi sướng bầu chọn, nó sẽ gửi thông điệp “Bầu chọn” đến tiến trình  $b_6$  và  $j_4$ , chúng ghi nhận tiến trình  $a_7$  là cha, trên hình vẽ được thể hiện bằng nét đậm. Tiến trình trên nút  $b_6$  chuyển tiếp thông điệp đến tiến trình  $c_3$  và  $g_2$ , tiến trình  $j_4$  chuyển tiếp thông điệp đến tiến trình  $g_2$ . Tiến trình  $c_3$  nhận tiến trình  $b_6$  làm cha, trong khi đó tiến trình  $g_2$  nhận được hai thông điệp “Bầu chọn” nên nó phải cân nhắc nhận tiến trình nào làm cha. Giả sử  $g_2$  nhận được thông điệp “Bầu chọn” từ tiến trình  $b_6$  trước tiến trình  $j_4$ , nó ghi nhận tiến trình  $b_6$  là cha, tiến trình  $g_2$  chỉ gửi cho  $j_4$  thông điệp xác nhận. Tiến trình  $g_2$  chuyển tiếp thông điệp “Bầu chọn” đến tiến trình  $h_8$  và  $e_1$ , cả hai tiến trình này đều chọn tiến trình  $g_2$  là cha. Nếu tại thời điểm này mà thông điệp từ tiến trình  $j_4$  chưa chuyển đến tiến trình  $g_2$  thì nó cũng chuyển thông điệp “Bầu chọn” cho tiến trình  $j_4$ , tất nhiên chắc chắn tiến trình  $j_4$  sẽ không nhận tiến trình  $g_2$  làm cha, do đó không ảnh hưởng đến kết quả bầu chọn.

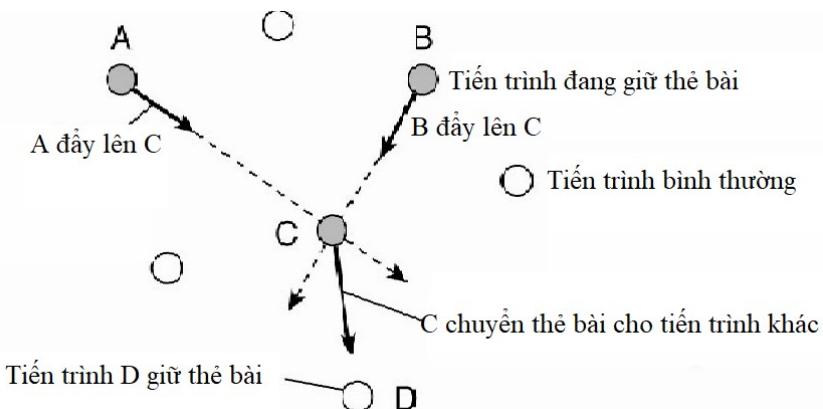
Quá trình tương tự xảy ra trên các tiến trình khác, tiến trình  $c_3$  chuyển tiếp thông điệp cho các tiến trình  $\{d_0, e_1\}$  và được  $d_0$  nhận làm cha, tiến trình  $e_1$  chuyển tiếp thông điệp “Bầu chọn” đến các tiến trình  $\{f_9, d_0, c_3\}$  và tiến trình  $f_9$  nhận làm cha. Tiến trình  $d_0$  chuyển tiếp thông điệp đến tiến trình  $f_9$  và có thể cả đến tiến trình  $e_1$  nhưng không được chấp nhận làm cha. Tiến trình  $h_8$  và  $f_9$  đều chuyển thông điệp “Bầu chọn” đến tiến trình  $i_5$ , nhưng thông điệp từ  $h_8$  đến trước nên tiến trình  $i_5$  nhận tiến trình  $h_8$  làm cha, thông điệp bầu chọn cũng có thể được tiến trình  $i_5$  chuyển đến tiến trình  $f_9$ .

Kết quả vòng gửi thông điệp “Bầu chọn“, các tiến trình  $\{j_4, f_9, d_0, i_5\}$  đều không có con, chúng chuyển định danh của mình cho các tiến trình cha để bắt đầu giai đoạn phản hồi thông điệp “Bầu chọn“. Tiến trình  $h_8$  thấy định danh của nó cao hơn nên chuyển cho tiến trình  $g_2$  giá trị đại diện là chính nó [ $h_8$ ], tiến trình  $e_1$  nhận thấy định danh của nó nhỏ hơn định danh của con  $f_9$  nên gửi cho tiến trình  $g_9$  giá trị đại diện [ $f_9$ ], tiến trình  $g_9$  tổng hợp nhánh của nó và gửi lên tiến trình  $b_6$  giá trị [ $f_9$ ]. Tương tự như vậy, tiến trình  $c_3$  cử đại diện [ $c_3$ ], tiến trình  $b_6$  cử đại diện [ $f_9$ ], tiến trình  $j_4$  gửi định danh của mình cho tiến trình  $a_7$ . Các thông điệp phản hồi được qui tụ về tiến trình khởi sướng  $a_7$ , nhận thấy tiến trình  $f_9$  có định danh cao nhất, do đó tiến trình  $a_7$  sẽ quảng bá thông điệp để thông báo tiến trình  $f_9$  đã được bầu chọn.

#### 4.4.4 Bầu chọn trong các hệ thống qui mô lớn

Các giải thuật bầu chọn đã trình bày trên thường chỉ áp dụng cho các hệ thống qui mô nhỏ, hơn nữa những giải thuật đó mới chỉ bầu chọn một tiến trình. Một số tình huống đòi hỏi bầu chọn nhiều tiến trình, ví dụ cần một số tiến trình đại diện trong các mạng ngang hàng. Hệ thống lớn sẽ được chia thành nhiều nhóm nhỏ, mỗi nhóm có một tiến trình đại diện, như vậy việc bầu chọn tiến trình cho toàn bộ hệ thống sẽ chỉ diễn ra giữa các tiến trình đại diện. Nên phân bổ các tiến trình đại diện một cách đồng đều sao cho độ trễ mạng đến các thành viên nhỏ, mỗi nhóm không nên quá nhiều tiến trình và nên xác định trước số lượng nhóm phù hợp với hệ thống.

Một cách chia nhóm đơn giản học theo cách chia mạng con trong môn học mạng máy tính, định danh của các tiến trình được biểu diễn bằng  $m$  bit, như vậy hệ thống hỗ trợ tối đa  $2^m$  tiến trình. Hệ thống được chia thành  $2^k$  nhóm, giả sử tiến trình đại diện là tiến trình có định danh nhỏ nhất trong nhóm, định danh của các tiến trình đại diện sẽ là những giá trị  $(0 \text{ đến } 2^k-1)*2^{m-k}$ . Một tiến trình muốn tìm định danh của tiến trình đại diện trong nhóm thì chỉ cần lấy định danh của mình thực hiện phép toán AND nhị phân trong đó k bit bên trái có giá trị bằng 1 và các bit còn lại có giá trị bằng 0. Ví dụ, hệ thống có tối đa 256 tiến trình được chia thành 8 nhóm, trong trường hợp này chỉ cần lấy  $m=8$  và  $k=3$ , định danh của các tiến trình sẽ từ 0 đến 255, định danh của các tiến trình đại diện mỗi nhóm  $(0 \text{ đến } 7)*32$ , nghĩa là tập  $\{0,32,64,96,128,160,192,224\}$ . Nếu tiến trình định danh 139 cần tìm đại diện của nhóm, viết dưới dạng nhị phân ta có 10001011 AND 11100000 cho kết quả 10000000 chuyển sang hệ thập phân ta có 128.



Hình 4.20 Nguyên lý di chuyển thẻ bài trong không gian hai chiều bằng lực đẩy

Một cách tiếp cận hoàn toàn khác dựa trên việc định vị các tiến trình trong không gian hình học m chiều, giả sử cần đặt N tiến trình đại diện đều khắp hệ thống. Ý tưởng rất đơn giản, chọn ngẫu nhiên N tiến trình, mỗi tiến trình được cấp một thẻ bài. Mỗi thẻ bài biểu thị một lực đẩy khiến thẻ bài khác có xu hướng di chuyển ra xa, hiệu ứng ròng là nếu tất cả các thẻ bài tác dụng cùng một lực đẩy, chúng sẽ di chuyển ra xa nhau và phân bổ đều trong không gian hình học.

Cách tiếp cận này yêu cầu các tiến trình giữ thẻ bài tìm hiểu về các thẻ bài khác, có thể sử dụng phương pháp lan truyền trong đó lực của thẻ bài được truyền đi khắp hệ thống. Nếu một tiến trình phát hiện ra rằng tổng lực tác động lên nó vượt quá ngưỡng thì nó sẽ di chuyển thẻ bài theo hướng của các lực kết hợp. Khi một thẻ bài được một tiến trình giữ trong một khoảng thời gian nhất định, tiến trình đó đó sẽ được coi như đại diện của nhóm. Ví dụ trên hình hình 4.20, thẻ bài trên các tiến trình A và B biểu thị lực đẩy lên tiến trình C đang giữ thẻ bài, tới ngưỡng nhất định nó chuyển thẻ bài cho tiến trình D, như vậy tiến trình C trở lại là tiến trình bình thường trong khi đó tiến trình D trở thành tiến trình đại diện.

#### 4.5 Hệ thống định vị

Các hệ thống phân tán qui mô lớn thường trải dài trên phạm vi địa lý rộng lớn, các tiến trình láng giềng trong mạng phủ nhưng thực tế khoảng cách vật lý của chúng rất xa nhau. Độ trễ truyền thông bao gồm thời gian xử lý trên các thiết bị và thời gian lan tỏa thông tin, khoảng cách càng lớn sẽ dẫn đến độ trễ càng cao. Nếu tần suất giao dịch giữa hai tiến trình lớn, cách tốt nhất là lựa chọn phương án cài đặt để vị trí của chúng gần nhau nhất có thể.

Khi số lượng các nút trong hệ phân tán tăng lên thì nhiệm vụ duy trì thông tin của các nút khác trở nên rất phức tạp. Trong mạng lưới địa lý, mỗi nút được được đặt tại một tọa độ địa lý nhất định và có thể tính toán được khoảng cách giữa các nút, nhưng khoảng cách càng xa thì càng cần nhiều thời gian để lưu chuyển thông điệp. Để giảm thiểu thời gian truy nhập người ta thường nhân bản các máy chủ dịch vụ, như vậy cần phải có các giải thuật để tìm kiếm máy chủ gần nhất đối với máy khách. Cách xác định vị trí của mỗi thiết bị trong mạng thông thường dựa trên hệ thống định vị toàn cầu GPS, tuy nhiên cũng có thể dựa vào một số phương pháp khác.

Hệ thống định vị toàn cầu GPS là hệ thống các vệ tinh có khả năng xác định vị trí trên toàn cầu với độ chính xác khá cao do Bộ quốc phòng Hoa Kỳ xây dựng từ những năm 1970. Ban đầu, GPS được xây dựng để phục vụ cho các mục đích quân sự, tuy nhiên sau này cho phép sử dụng cả trong lĩnh vực dân sự. GPS bao gồm một mạng lưới 31<sup>1</sup> vệ tinh đang vận hành, để có khả năng hoạt động tốt thì số lượng vệ tinh trong mạng lưới phải luôn luôn lớn hơn 24.

---

<sup>1</sup> Theo trang

[https://www.faa.gov/about/office\\_org/headquarters\\_offices/ato/service\\_units/techops/navservices/gnss/gps#:~:text=Currently%2031%20GPS%20satellites%20orbit,Department%20of%20Defense%20\(DoD\).](https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps#:~:text=Currently%2031%20GPS%20satellites%20orbit,Department%20of%20Defense%20(DoD).)

Để đảm bảo vùng phủ sóng liên tục trên toàn thế giới, các vệ tinh GPS được sắp xếp sao cho 4 vệ tinh sẽ nằm cùng nhau trên 1 trong 6 mặt phẳng quỹ đạo. Với cách sắp xếp này sẽ có 4 đến 10 vệ tinh được nhìn thấy tại bất kỳ điểm nào trên trái đất với góc ngang là 100 nhưng thực tế chỉ cần 4 vệ tinh là có thể cung cấp đầy đủ các thông tin về vị trí. Các quỹ đạo vệ tinh GPS là những đường vòng elip với độ lệch tâm cực đại là 0.01, nghiêng khoảng 55° so với đường xích đạo. Độ cao của các vệ tinh so với bề mặt trái đất là khoảng 20.200 km, chu kỳ quỹ đạo các vệ tinh GPS khoảng 12 giờ, chính xác là 11 giờ 58 phút.

Mỗi vệ tinh có bốn đồng hồ nguyên tử, được đồng bộ thường xuyên với các máy chủ thời gian nguyên tử trên Trái Đất, nó liên tục quảng bá vị trí của mình và kèm theo nhãn thời gian cục bộ trong mỗi thông điệp. Việc phát sóng này cho phép mọi thiết bị trên Trái Đất có bộ tiếp nhận sóng vệ tinh có thể tính toán chính xác vị trí của mình, về nguyên tắc cần sử dụng thông tin của bốn vệ tinh. Tọa độ mỗi thực thể trong không gian ba chiều được xác định bằng cặp ba giá trị ( $x, y, z$ ), do đó khoảng cách từ thiết bị R đến vệ tinh  $S_i$  có thể tính theo công thức:

$$d_i = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Tuy nhiên thực tế lại phức tạp hơn nhiều vì cần phải có một khoảng thời gian nhất định để chuyên thông điệp từ vệ tinh về đến thiết bị và thời gian của thiết bị thường chưa được đồng bộ với vệ tinh. Ký hiệu  $T_i$  là nhãn thời gian đính kèm thông điệp quảng bá của vệ tinh  $S_i$ ,  $\Delta_{si}$  là độ lệch thời gian của vệ tinh  $S_i$  so với đồng hồ nguyên tử gốc,  $T_r$  là thời gian trên thiết bị R,  $\Delta_r$  là độ lệch thời gian của thiết bị R so với với đồng hồ nguyên tử gốc, độ lệch thời gian giữa thiết bị đến mỗi vệ tinh là  $\Delta_i = (T_r - T_i) + (\Delta_r - \Delta_{si})$ . Giả thiết đồng hồ trên các vệ tinh được đồng bộ chính xác tuyệt đối với đồng hồ nguyên tử gốc trên Trái Đất và tín hiệu được lan truyền với tốc độ ánh sáng c, khoảng cách giữa thiết bị đến vệ tinh  $S_i$  là  $d_i = c * [(T_r - T_i) - \Delta_r]$ , ta có phương trình sau:

$$c * [(T_r - T_i) - \Delta_r] = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$$

Phương trình trên chứa 4 ẩn số bao gồm các tọa độ  $(x_r, y_r, z_r)$  và độ lệch thời gian  $\Delta_r$  của thiết bị R so với đồng hồ nguyên tử, do đó cần phải có bốn vệ tinh để tìm được đáp số duy nhất. Thực tế đồng hồ trên các vệ tinh và vị trí của chúng chưa hoàn toàn chính xác, tốc độ lan truyền tín hiệu không phải là hằng số, tốc độ ánh sáng 300 000 km/s chỉ là ở môi trường chân không, do đó sai số tính toán sẽ khoảng 10 m. Để tính toán một cách chính xác hơn người ta sử dụng GPS vi sai, tuy nhiên điều này không cần thiết đối với việc xác định vị trí của một thiết bị trong hệ thống phân tán.

## THẢO LUẬN

1. Nêu vai trò của thời gian vật lý, lấy ví dụ về hậu quả của sự mất đồng bộ thời gian vật lý.
2. Trình bày các giải thuật đồng bộ thời gian vật lý.
3. Sử dụng tính năng đồng bộ thời gian quốc tế có sẵn trong hệ điều hành của các máy tính.

4. Lập chương trình đồng bộ thời gian vật lý sử dụng giao thức NTP.
5. Nêu vai trò của nhãn thời gian logic trong các hệ thống phân tán.
6. Tìm hiểu cách ứng dụng nhãn thời gian Lamport trong hệ quản trị cơ sở dữ liệu Oracle.
7. Xây dựng ứng dụng truyền thông điệp đơn giản có sử dụng nhãn thời gian Vector.
8. Trình bày các giải thuật loại trừ tương hỗ phân tán và so sánh, đưa ra khuyến nghị áp dụng trong thực tiễn.
9. So sánh các giải thuật bầu chọn và đưa ra khuyến nghị áp dụng trong thực tiễn.
10. Xây dựng ứng dụng lựa chọn thành viên điều phối với tiêu chí bầu chọn là giá trị cao nhất của địa chỉ vật lý của giao diện mạng.

## CHƯƠNG 5: TIẾN TRÌNH TRONG CÁC HỆ THỐNG PHÂN TÁN

Khái niệm tiến trình xuất phát từ lĩnh vực hệ điều hành, một chương trình đang chạy thì được gọi là tiến trình. Một trong những nhiệm vụ chính của hệ điều hành là quản lý và lập lịch cho các tiến trình, tuy nhiên trong hệ thống phân tán sẽ phát sinh nhiều vấn đề phức tạp hơn khi hoạt động của nó liên quan đến những tiến trình chạy trên các máy tính khác. Tiến trình tạo nên các khối chức năng trong hệ thống phân tán, thực tế cho thấy việc phân chia như vậy chưa đủ, cần thiết phải nhìn nhận vấn đề một cách chi tiết hơn. Ví dụ, trong hệ thống phân tán sẽ phải thường xuyên áp dụng kỹ thuật đa luồng nhằm nâng cao hiệu năng hoạt động của hệ thống, với kỹ thuật đó quá trình trao đổi thông tin khách/chủ và các xử lý cục bộ được thực hiện đồng thời với nhau.

Trong những năm gần đây, khái niệm ảo hóa ngày càng trở nên phổ biến. Bằng kỹ thuật ảo hóa, một ứng dụng có thể chạy song song với các ứng dụng khác, có thể cài đặt các hệ điều hành trên nhiều nền tảng khác nhau. Các ứng dụng và thậm chí hệ điều hành hoạt động độc lập với phần cứng, điều đó làm cho hệ thống đáp ứng dễ thích nghi với các môi trường khác nhau. Ảo hóa tạo điều kiện cách ly các lỗi phát sinh trên mỗi tiến trình hoặc lỗi về bảo mật, nó cũng giải quyết tốt vấn đề tương tranh.

Mô hình khách/chủ đóng vai trò quan trọng trong các hệ thống phân tán, do đó cách triển khai mô hình này sẽ được đề cập kỹ lưỡng, cách tổ chức máy khách và máy chủ. Những vấn đề chung về thiết kế máy chủ cũng sẽ được đề cập dưới nhiều góc độ khác nhau, từ kiến trúc hệ thống phần mềm cho đến cách bố trí mạng lưới thiết bị phần cứng.

Một vấn đề quan trọng nữa là vấn đề di trú, tiến trình có thể di chuyển giữa các máy tính khác nhau trong hệ thống phân tán. Di trú tiến trình hay cụ thể hơn là di trú mã nhằm đảm bảo các yêu cầu về hiệu năng khi qui mô hệ thống mở rộng, nhưng cũng có thể đạt được mục tiêu cấu hình động cho máy khách và máy chủ, hệ thống trở nên linh hoạt hơn.

### 5.1 Các luồng

Theo quan điểm hệ điều hành, việc quản lý và lập lịch cho các tiến trình đóng vai trò quan trọng nhất. Trong hệ thống phân tán, vấn đề trao đổi thông tin giữa các tiến trình này sinh nhiều vấn đề khá phức tạp, ngoài việc quản lý các tiến trình cần phải đảm bảo hiệu suất hoạt động của hệ thống. Mặc dù các tiến trình hình thành nên các khối xây dựng hệ thống phân tán nhưng thực tiễn cho thấy cần thiết phải chia tiến trình thành các đơn vị nhỏ hơn để dễ dàng xây dựng các ứng dụng phân tán và đồng thời đạt được hiệu năng cao hơn.

#### 5.1.1 Khái niệm luồng

Để hiểu về vai trò của luồng trong hệ thống phân tán thì phải nắm được bản chất của tiến trình và mối quan hệ của nó với các luồng. Mỗi chương trình khi chạy cần phải được cung cấp một bộ xử lý, để đáp ứng yêu cầu cùng một lúc chạy được nhiều chương trình thì hệ điều hành phải tạo ra một số bộ xử lý ảo cho mỗi chương trình. Nhằm mục đích quản lý các bộ xử lý ảo này, hệ điều hành tạo bảng tiến trình chứa thông tin giá trị các thành ghi của đơn vị xử lý trung tâm, bản đồ bộ nhớ, các tập tin đang dùng, thông tin

tài khoản và các quyền thực hiện có liên quan..., tất cả những thứ đó tương tự như ngữ cảnh của một bộ xử lý và được gọi là ngữ cảnh tiến trình. Như vậy tiến trình có thể được hiểu như là một chương trình đang chạy trên bộ xử lý ảo của hệ điều hành. Nhiều tiến trình chạy đồng thời và dùng chung đơn vị xử lý trung tâm cũng như các tài nguyên khác mà không ảnh hưởng lẫn nhau, do đó sẽ đảm bảo tính trong suốt của hệ thống.

Để đạt được tính trong suốt, hệ điều hành phải trả giá khá cao trong các thao tác xử lý, nó phải tạo một không gian bộ nhớ riêng cho mỗi tiến trình. Không gian đó được mô phỏng như một hệ vi xử lý thực sự, phải có các thanh ghi và vùng nhớ để lưu trữ dữ liệu. Tại một thời điểm chỉ có duy nhất một tiến trình được phép sử dụng đơn vị xử lý trung tâm và vùng nhớ thực thi chương trình, chi phí về thời gian xử lý sẽ phát sinh trong di chuyển dữ liệu giữa bộ xử lý ảo và bộ xử lý vật lý.

Hệ điều hành phải sao chép toàn bộ dữ liệu của tiến trình đang chạy vào vùng nhớ đã qui định của tiến trình đó, sau đó mới sao chép dữ liệu của tiến trình kế tiếp vào bộ xử lý vật lý. Khi các tiến trình sử dụng hết bộ nhớ chính, nó sẽ sử dụng một phần ổ đĩa để làm bộ nhớ, như vậy sẽ phát sinh rất nhiều các thao tác đọc/ghi đĩa, điều này sẽ làm suy giảm nghiêm trọng hiệu năng xử lý của hệ thống.

Giống như tiến trình, luồng thực hiện đoạn mã chương trình của mình độc lập với các luồng khác. Tuy nhiên, điểm khác biệt cơ bản so với tiến trình, luồng không cố định độ trong suốt cao nếu thao tác đó làm suy giảm hiệu năng hoạt động, do đó hệ thống luồng thường chỉ duy trì thông tin tối thiểu nhằm chia sẻ đơn vị xử lý trung tâm. Cụ thể, vấn đề quản lý luồng được thực hiện theo cơ chế đóng/mở, tính toàn vẹn của dữ liệu do người phát triển phần mềm ứng dụng đảm nhiệm.

### **5.1.2 Luồng trong trên máy tính độc lập**

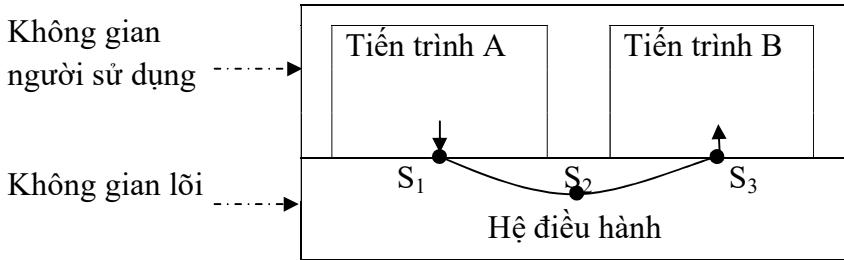
Nếu chỉ sử dụng một luồng đơn, bất cứ khi nào thực thi lời gọi hệ thống loại phong tỏa thì sẽ tiến trình cũng sẽ bị phong tỏa. Một số phần mềm chạy trên máy tính độc lập nhưng vẫn sử dụng nhiều luồng để tạo cơ chế xử lý song song để nâng cao hiệu năng xử lý. Ví dụ ứng dụng đa luồng trong kỹ thuật xử lý song song, trên máy tính nhiều bộ vi xử lý, mỗi luồng được gán cho một bộ xử lý nhưng chúng vẫn dùng chung bộ nhớ của máy tính.

Một ví dụ khác là bảng tính Microsoft Excel, các ô trong bảng tính độc lập với nhau, nếu người sử dụng thay đổi giá trị của một ô thì sẽ làm thay đổi giá trị của các ô có liên quan với ô dữ liệu đó. Nếu chỉ sử dụng một luồng thì công việc tính toán sẽ không được thực hiện trong khi nhập dữ liệu và ngược lại khi đang tính toán thì người dùng sẽ không nhập được dữ liệu. Như vậy ở đây cần phải có hai luồng xử lý, một luồng giao tiếp với người sử dụng và một luồng khác cập nhật thông tin của trang, thậm chí cần thiết phải có luồng thứ ba thực hiện nhiệm vụ sao lưu dữ liệu vào ổ đĩa.

Áp dụng đa luồng để đạt được tính song song trong xử lý đang ngày càng trở nên quan trọng khi phát triển phần mềm chạy trên máy tính nhiều bộ vi xử lý, chúng thường được sử dụng để đóng vai trò máy chủ cung cấp dịch vụ cho các máy khách. Nếu được thiết kế đúng cách, cách xử song song có thể đạt được tính trong suốt, tiến trình sẽ chạy

tốt ngay cả khi chạy trên máy tính chỉ lắp đặt một bộ vi xử lý, tất nhiên chậm hơn đôi chút so với máy tính nhiều bộ vi xử lý.

Kỹ thuật xử lý đa luồng cũng được áp dụng để xây dựng những ứng dụng qui mô lớn, đó là những ứng dụng bao gồm nhiều chương trình cộng tác, mỗi chương trình sử dụng một tiến trình riêng, tương tác giữa các chương trình được thực hiện bằng cơ chế truyền thông liên tiến trình, ví dụ như kỹ thuật đường ống, hàng đợi thông điệp hay vùng nhớ dùng chung, nhược điểm cơ bản của cơ chế này là vẫn đề luân chuyển từ tiến trình này sang tiến trình khác.



Hình 5.1 Chuyển ngữ cảnh giữa các tiến trình

Hình 5.1 minh họa việc chuyển tiến trình, trước tiên tiến trình A phải chuyển từ không gian người sử dụng sang không gian lõi tại điểm  $S_1$ , điều này sẽ làm thay đổi bản đồ đơn vị quản lý bộ nhớ MMU và xây dựng lại vùng đệm tra cứu chuyển đổi TLB, đó là bảng lưu trữ thông tin chuyển đổi từ địa chỉ ảo sang địa chỉ vật lý giúp giảm thiểu thời gian truy nhập bộ nhớ. Bên trong không gian lõi sẽ xảy ra quá trình chuyển ngữ cảnh xử lý tại điểm  $S_2$  và sau đó tại điểm  $S_3$  sẽ chuyển tiến trình B sang không gian người sử dụng, quá trình này cũng đòi hỏi phải thay đổi MMU và TLB.

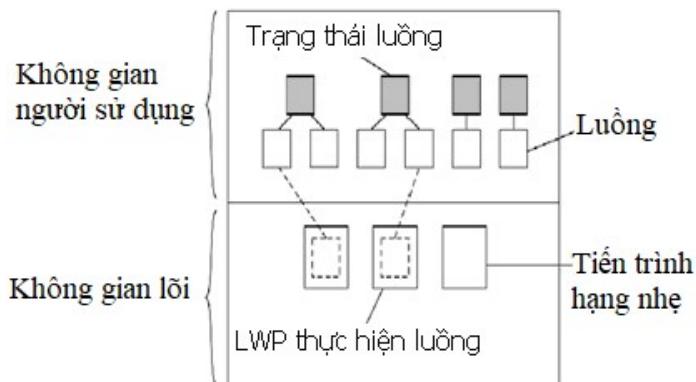
Thay cho việc xây dựng nhiều tiến trình, ứng dụng có thể được phân thành nhiều phần khác nhau, mỗi phần tương ứng với một luồng. Trao đổi thông tin giữa các luồng được thực hiện bằng cách chia sẻ vùng nhớ, việc chuyển đổi từ luồng này sang luồng khác được thực hiện trong không gian người sử dụng, rất ít khi sử dụng cơ chế lập lịch trong không gian lõi của hệ điều hành, vì vậy hiệu năng của hệ thống sẽ tăng lên đáng kể. Cuối cùng, lý do áp dụng kỹ thuật đa luồng xuất phát từ công nghệ phần mềm, tổ chức các luồng phối hợp với nhau trong một tiến trình sẽ đơn giản và thuận tiện hơn, người sử dụng sẽ không cần phải chuyển từ cửa sổ này sang cửa sổ khác.

### 5.1.3 Cài đặt luồng

Luồng thường được cung cấp dưới dạng gói luồng, những gói như vậy bao gồm các thao tác tạo và hủy luồng cũng như các thao tác trên biến đồng bộ, ví dụ các biến cung cấp cơ chế khóa trong điều khiển tương tranh và các biến điều kiện. Có thể cài đặt thư viện luồng chạy hoàn toàn ở mức người sử dụng, các thao tác tạo và hủy hoặc chuyển luồng được thực hiện tương đối dễ dàng với chi phí khá thấp. Thao tác tạo luồng chỉ đơn giản là cấp phát bộ nhớ, hủy luồng chỉ cần giải phóng vùng nhớ đã cấp phát. Chi phí thực hiện chuyển luồng cũng không lớn, về cơ bản chỉ cần vài chỉ thị lệnh chuyển dữ liệu trong các thanh ghi của đơn vị xử lý trung tâm mà không cần phải cập nhật lại bản đồ MTU và vùng đệm TLB. Cách tiếp cận này có điểm hạn chế, nếu một luồng thực thi lời

gọi hệ thống thuộc loại phong tỏa thì tất cả các luồng khác trong tiến trình sẽ phải tạm ngừng.

Cách tiếp cận thứ hai, cài đặt luồng ở mức lõi của hệ điều hành, nghĩa là luồng sẽ được hệ điều hành đưa vào lịch thực hiện, các thao tác tạo và hủy hoặc chuyển luồng đều do mức lõi thực hiện, hệ điều hành coi luồng tương tự như tiến trình, chúng được gọi là tiến trình hạng nhẹ LWP. Như vậy, nếu một luồng thực hiện lời gọi hệ thống thì chỉ luồng đó bị phong tỏa, các luồng khác của tiến trình không bị phong tỏa, nó vẫn duy trì cơ chế chạy song song giữa các luồng của tiến trình.



Hình 5.2 Kết hợp tiến trình hạng nhẹ và các luồng mức người sử dụng

Kết hợp luồng ở mức người sử dụng và tiến trình hạng nhẹ, tiến trình chính bao gồm một số tiến trình hạng nhẹ và thư viện luồng chạy ở mức người sử dụng, thao tác chuyển luồng được thực hiện thông qua các biến đồng bộ mà không cần lõi của hệ điều hành can thiệp. Các tiến trình hạng nhẹ chia sẻ gói luồng, chúng vẫn có thể tạo một số luồng riêng ở mức người sử dụng, nếu gọi hàm thuộc loại phong tỏa thì tiến trình hạng nhẹ bị phong tỏa trong khi các luồng khác của tiến trình chính vẫn xử lý.

Các ứng dụng đa luồng được xây dựng bằng cách tạo ra các luồng chạy ở mức người sử dụng và các tiến trình hạng nhẹ, mỗi tiến trình hạng nhẹ sẽ tạo ra một ngăn xếp chứa các luồng riêng, các luồng được quản lý trong bảng luồng và được bảo vệ bằng cơ chế biến đồng bộ. Như vậy, mỗi tiến trình con khi gọi đến luồng nào đó chỉ cần tìm kiếm trong bảng luồng và thực hiện việc chuyển ngữ cảnh hoàn toàn trong chế độ người sử dụng. Trường hợp một luồng nào đó thực hiện lời gọi phong tỏa hệ thống thì nó chuyển từ chế độ người dùng sang chế độ lõi của hệ điều hành nhưng vẫn nằm trong tiến trình hạng nhẹ.

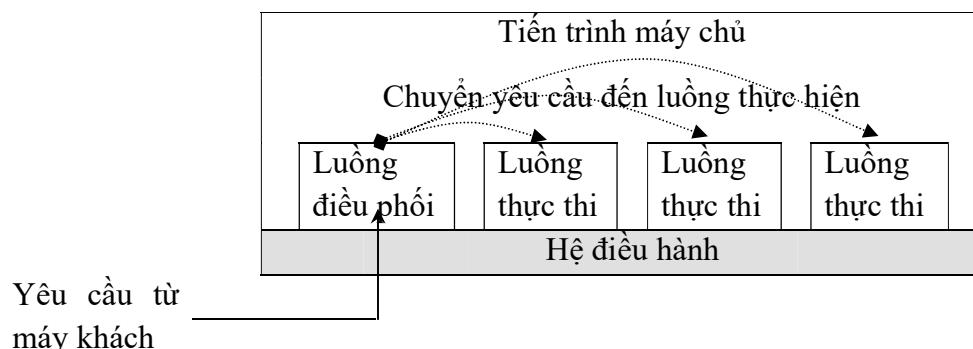
#### **5.1.4 Luồng trong các hệ thống phân tán**

Một đặc tính quan trọng của luồng là chúng cung cấp các phương tiện dễ dàng cho phép gọi phong tỏa hệ thống nhưng không phong tỏa toàn bộ tiến trình đang chạy, do đó kỹ thuật này thường được sử dụng trong các hệ thống phân tán để duy trì truyền thông dưới dạng các kênh logic. Ví dụ, trong mô hình khách/chủ, máy khách tạo ra nhiều luồng độc lập chạy song song với nhau, máy chủ cũng tạo ra nhiều luồng để có thể đồng thời tiếp nhận và xử lý các yêu cầu của máy khách.

Trên mạng diện rộng, thông tin di chuyển từ máy tính này sang máy tính khác sẽ mất một khoảng thời gian nhất định, đôi khi có thể lên tới vài giây. Để đảm bảo tính trong suốt phân bố tài nguyên thì các ứng dụng phân tán cần phải che giấu thời gian lan truyền các thông điệp, cách thông thường là khởi tạo kênh truyền và trong thời gian đó vẫn thực hiện các nhiệm vụ khác. Ví dụ, khi truy nhập trang tin điện tử, trình duyệt sử dụng giao thức HTTP để lấy dữ liệu từ máy chủ về máy khách, nếu hiển thị thông tin cho người dùng sau khi đã tải toàn bộ dữ liệu của trang thì thời gian chờ đợi sẽ tương đối dài. Để giải quyết vấn đề này, sau khi lấy được khung dữ liệu của toàn bộ trang, máy khách sẽ tạo ra nhiều luồng riêng biệt, mỗi luồng sẽ lấy dữ liệu một phần của trang, nhận được dữ liệu của phần nào thì hiển thị ngay thông tin của phần đó.

Như vậy, mỗi luồng trên máy khách sẽ thiết lập một kênh logic mới gửi đến máy chủ, điều này làm tăng số lượng yêu cầu lên máy chủ và khi số lượng yêu cầu vượt quá khả năng xử lý thì các yêu cầu này sẽ được đưa vào hàng đợi chờ xử lý, như vậy hiệu năng của hệ thống sẽ không được cải thiện. Trong nhiều trường hợp, máy chủ được tổ chức thành từng cụm, mỗi yêu cầu của máy khách không phải chỉ xử lý trên một máy chủ mà có thể sẽ được chuyển đến máy khác xử lý, do đó đòi hỏi phải triển khai kỹ thuật xử lý song song trên cả máy khách lẫn máy chủ.

Kỹ thuật xử lý đa luồng trên máy khách đóng vai trò rất quan trọng nhưng thực tế cho thấy việc triển khai kỹ thuật này trên máy chủ mới là nhân tố quyết định hiệu năng của hệ thống phân tán. Kỹ thuật xử lý đa luồng không những đơn giản hóa đáng kể cách phát triển phần mềm mà còn làm cho việc triển khai kỹ thuật xử lý song song để đạt được hiệu năng cao nhất, dù cho đó là máy tính một hay nhiều bộ vi xử lý.



Hình 5.3 Xử lý đa luồng trên máy chủ

Ví dụ hệ thống cung cấp dịch vụ truyền tập tin, tập tin được lưu trữ trên ổ đĩa và quá trình đọc các khối dữ liệu trên ổ đĩa chiếm thời gian đáng kể so với các thao tác khác trong phiên làm việc. Hình 5.3 thể hiện kỹ thuật xử lý đa luồng, máy chủ tạo luồng chuyên tiếp nhận các yêu cầu của máy khách và nhiều luồng khác có nhiệm vụ đọc dữ liệu trên ổ đĩa. Khi nhận yêu cầu đọc tập tin, thành phần tiếp nhận yêu cầu sẽ quyết định giao cho tiến trình đọc/ghi nào đó thực hiện sau đó lại tiếp tục trở về trạng thái sẵn sàng tiếp nhận các yêu cầu khác từ máy khách. Nhận được yêu cầu đọc dữ liệu, một luồng thực thi đọc/ghi sẽ được khởi tạo bằng cách mở tập tin và chờ cho đến khi hoàn thành

quá trình đọc, sau khi đọc xong sẽ gửi kết quả về cho tiến trình phần tiếp nhận để trả về cho máy khách.

Nếu chỉ sử dụng một luồng xử lý, vòng lặp của chương trình chính sẽ tiếp nhận và hoàn thành việc thực thi nhiệm vụ trước khi tiếp nhận yêu cầu kế tiếp, như vậy trong khi chờ đọc dữ liệu từ ổ đĩa máy chủ tuy ở trạng thái nghỉ nhưng vẫn không tiếp nhận thêm yêu cầu nào, kết quả là nhiều yêu cầu của máy khách vẫn phải xếp hàng để được xử lý. Như vậy có thể thấy, việc áp dụng kỹ thuật đa luồng đã làm tăng đáng kể hiệu năng xử lý nhưng mỗi luồng vẫn được lập trình theo phương pháp thông thường.

Một giải pháp khác áp dụng kỹ thuật đa luồng cho hiệu năng cao gọi là máy trạng thái hữu hạn, nó thường được áp dụng trong các hệ thống đọc nhiều ghi ít. Nguyên tắc của nó là sử dụng vùng đệm, kết quả yêu cầu đọc dữ liệu sẽ được đưa vào vùng đệm gọi là bộ nhớ cache để cung cấp cho các yêu cầu tương tự, việc truy nhập vào bộ nhớ cache sẽ nhanh hơn rất nhiều so với đọc dữ liệu trên ổ đĩa.

Khi có nhận được yêu cầu đọc, tiến trình sẽ kiểm tra xem dữ liệu trong bộ nhớ cache có đáp ứng yêu cầu hay không, nếu đáp ứng được yêu cầu thì sẽ trả về cho máy khách, nếu không đáp ứng thì phải đọc từ ổ đĩa. Tuy nhiên, hàm đọc dữ liệu từ ổ đĩa thuộc loại phong tỏa, như vậy sẽ phong tỏa luôn cả tiến trình tiếp nhận dù có áp dụng kỹ thuật đa luồng, vì vậy nó đưa yêu cầu vào một bảng và thiết lập trạng thái “Chờ xử lý” cho bản ghi đó và tiếp tục nhận các yêu cầu mới, bảng quản lý yêu cầu cũng thường được lưu trong bộ nhớ để tăng tốc độ truy nhập. Một tiến trình hạng nhẹ hoặc thậm chí một tiến trình khác sẽ lấy từng bản ghi trong bảng quản lý yêu cầu để xử lý, kết quả sẽ đưa vào bộ nhớ cache và đồng thời cập nhật trạng thái của bản ghi “Đã xử lý”. Sau mỗi lần tiếp nhận yêu cầu mới, tiến trình sẽ kiểm tra bảng quản lý yêu cầu, nếu thấy trạng thái “Đã xử lý” thì sẽ lấy dữ liệu từ bộ nhớ cache để trả về cho máy khách tương ứng và đồng thời có thể xóa bản ghi này.

Với phương pháp này, tiến trình tiếp nhận yêu cầu sẽ không cần phải thực hiện các lời gọi hệ thống thuộc loại phong tỏa khi gửi và nhận yêu cầu từ máy khách và đồng thời loại bỏ được đặc điểm xử lý tuần tự của hai phương pháp trước. Trạng thái xử lý được lưu tường minh khi gửi hoặc nhận mỗi yêu cầu, tiến trình vận hành như máy trạng thái hữu hạn, nó lấy sự kiện và phản hồi tùy thuộc vào cái gì chứa trong tiến trình đó.

**Bảng 5.1** So sánh ba kỹ thuật xử lý trên máy chủ

Kỹ thuật	Đặc điểm
Đơn luồng	Tuần tự, phong tỏa hệ thống
Đa luồng	Song song, phong tỏa hệ thống
Máy trạng thái	Song song, không phong tỏa hệ thống

Bảng 5.1 tóm tắt ba kỹ thuật xử lý trên máy chủ, có thể áp dụng ba kỹ thuật xử lý đa luồng, đơn luồng hoặc máy trạng thái hữu hạn. Kỹ thuật đa luồng cho phép xử lý song song nhưng có thể phải tạm ngừng tiếp nhận yêu cầu nếu gặp phải hàm thuộc loại phong tỏa. Kỹ thuật đơn luồng không những có thể bị phong tỏa mà còn không hỗ trợ xử lý song song, hiệu năng hệ thống thấp. Kỹ thuật máy trạng thái hữu hạn cho phép xử lý song song

và không phong tỏa hệ thống, nó rất phù hợp cho các hệ thống lớn, đặc biệt là các hệ thống đọc nhiều ghi ít.

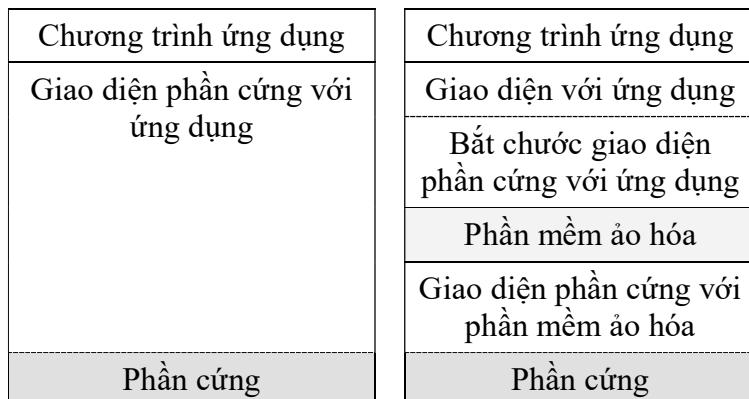
## 5.2 Ảo hóa

Xử lý đa luồng và đa tiến trình có thể được nhìn nhận như một cách thực hiện nhiều công việc trong cùng một thời điểm, làm cho hiệu năng của hệ thống cao hơn. Nếu máy tính chỉ có một bộ vi xử lý thì việc chạy đồng thời nhiều tiến trình chỉ là ảo giác, bởi vì tại một thời điểm bộ vi xử lý chỉ có thể thực thi được một chỉ thị lệnh, bằng cách chuyển rất nhanh từ tiến trình này sang tiến trình khác hoặc từ luồng này sang luồng khác tạo ra cảm giác các chúng chạy song song với nhau.

Ý tưởng ảo hóa xuất phát từ nhu cầu thực tế, tài nguyên vật lý chỉ có một nhưng nhiều thành phần cùng có nhu cầu sử dụng tài nguyên chung và thành phần nào cũng muốn tài nguyên đó thuộc về mình. Như vậy sẽ nảy sinh vấn đề tranh chấp giữa các tiến trình, vì vậy kỹ thuật ảo hóa đã được áp dụng vừa để đáp ứng nhu cầu sở hữu riêng tài nguyên của các tiến trình và đồng thời không để xảy ra tranh chấp trong hệ thống.

### 5.2.1 Vai trò ảo hóa

Các hệ thống máy tính được tổ chức theo mô hình phân tầng, tầng thấp hơn sẽ cung cấp giao diện lập trình cho các tầng cao hơn. Có nhiều loại giao diện, thấp nhất là giao diện với bộ xử lý trung tâm cho đến thư viện giao diện lập trình ứng dụng do các hệ thống trung gian cung cấp. Sự ảo hóa ở đây thể hiện bằng cách mở rộng hoặc thay thế giao diện hiện hành để bắt chước hành vi của hệ thống khác. Hình 5.4 thể hiện nguyên lý ảo hóa, phần mềm ảo hóa bắt chước giao diện phần cứng để cung cấp giao diện cho các chương trình ứng dụng, điều này làm cho phần mềm ứng dụng tách biệt với hạ tầng phần cứng.



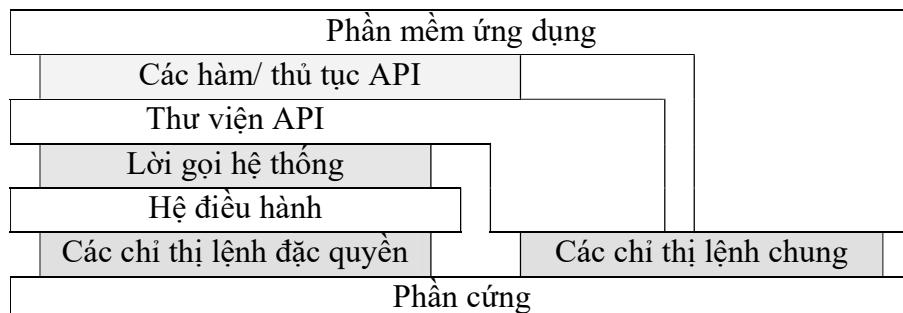
Hình 5.4 Ảo hóa tài nguyên phần cứng

Những năm 70 của thế kỷ trước, giá thành của máy tính rất đắt mà số lượng phần mềm chưa nhiều và các hệ điều hành thường phát triển cho một loại phần mềm nào đó. Ý tưởng ảo hóa được IBM triển khai nhằm tiết kiệm chi phí và cho phép các hệ thống phần mềm cũ có thể chạy trên máy chủ thế hệ mới. Khi giá thành phần cứng giảm dần, công nghệ phần cứng thay đổi nhanh hơn các ứng dụng phần mềm, vai trò ảo hóa chuyển dần sang nhiệm vụ đảm bảo tính tương thích với các hệ thống cũ.

Đặc biệt, với sự bùng nổ của mạng Internet, nhiều hệ thống phần mềm lớn được xây dựng trên các nền tảng hệ thống khác nhau, việc ảo hóa nhằm tạo điều kiện cho các hệ thống dễ dàng tương tác với nhau. Ảo hóa sẽ giúp tăng cường tính khả chuyển và tính linh hoạt của hệ thống, ví dụ hỗ trợ cho việc nhân bản dữ liệu, góp phần nâng cao tính trong suốt của hệ thống. Ảo hóa giúp nâng cao tính năng bảo mật, các thành phần bên ngoài không được phép truy nhập trực tiếp vào các đối tượng bên trong hệ thống. Ngoài ra, kỹ thuật ảo hóa có thể hạn chế thậm chí loại bỏ các sự cố tương tranh, điều này giúp cho hệ thống hoạt động hiệu quả hơn.

### 5.2.2 Phân loại ảo hóa

Có nhiều cách để thực hiện việc ảo hóa, để hiểu các hình thức này trước hết cần phải biết các hệ thống máy tính thường cung cấp ba loại giao diện. Thứ nhất, giao diện giữa phần cứng và phần mềm bao gồm các chỉ thị lệnh, nó bao gồm các chỉ thị lệnh chung mà bất kỳ chương trình nào cũng có thể thực hiện và các chỉ thị lệnh đặc quyền chỉ có hệ điều hành mới được thực hiện. Thứ hai là giao diện gồm các lời gọi hệ thống do hệ điều hành cung cấp, nó cho phép phần mềm ứng dụng yêu cầu các dịch vụ lõi của hệ điều hành, ví dụ các yêu cầu truy nhập phần cứng hay quản lý các tập tin trên ổ đĩa. Cuối cùng là giao diện gồm các lời gọi thư viện hình thành trong thư viện giao diện lập trình ứng dụng API, trong nhiều trường hợp các lời gọi hệ thống ẩn trong thư viện này. Ba loại giao diện kể trên được thể hiện trên hình 5.5 và đó là những vấn đề cốt yếu của ảo hóa để bắt chước hành vi của các giao diện.



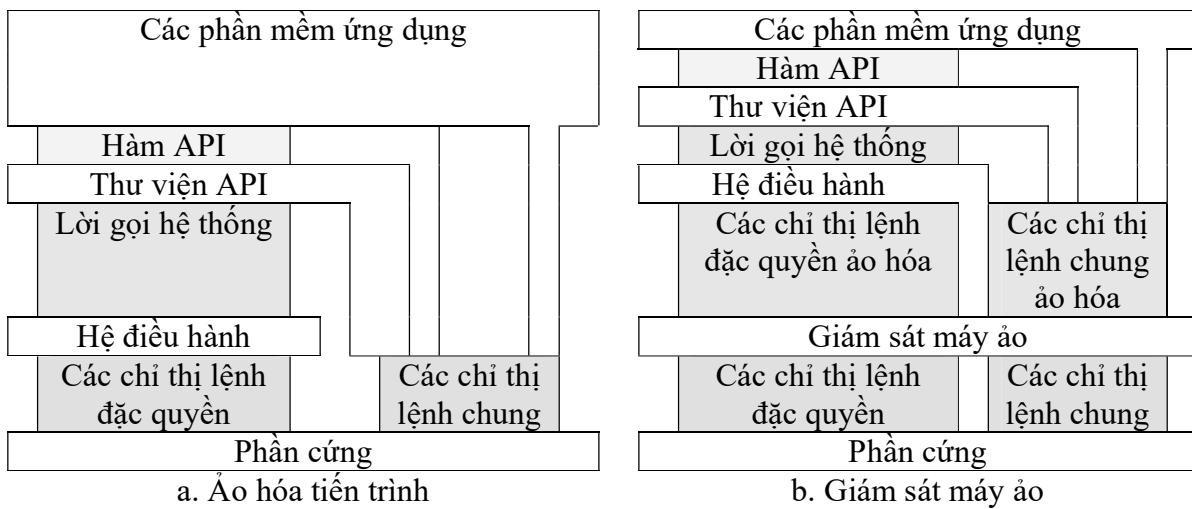
Hình 5.5 Các loại giao diện trên máy tính

Hình 5.6 thể hiện hai hình thức ảo hóa, thứ nhất là máy ảo tiền trình và thứ hai là giám sát máy ảo. Máy ảo tiền trình xây dựng hệ thống thực hiện theo thời gian, về cơ bản nó cung cấp tập chỉ thị lệnh trừu tượng dùng để thực thi phần mềm ứng dụng, các chỉ thị lệnh này sẽ được thông dịch trực tiếp. Ví dụ các ứng dụng viết bằng ngôn ngữ Java sẽ chạy trên môi trường Runtime Java, hoặc cũng có thể được mô phỏng như chạy các ứng dụng Windows trên nền tảng hệ điều hành Unix, hình thức ảo hóa này chủ yếu áp dụng cho một tiền trình.

Giám sát máy ảo dựa trên mô hình phân tầng, nó che đậy hoàn toàn phần cứng nhưng cung cấp đầy đủ các chỉ thị lệnh của phần cứng gọi là giao diện. Điểm quan trọng trong hình thức này là giao diện của nó cung cấp đồng thời cho nhiều chương trình khác nhau, nghĩa là nhiều hệ điều hành có thể cùng chạy trên một nền tảng phần cứng.

Giám sát máy ảo cung cấp và sắp đặt quyền truy nhập đến nhiều tài nguyên khác nhau, ví dụ thiết bị lưu trữ bên ngoài và hạ tầng vật lý của mạng máy tính giống như bất kỳ hệ điều hành nào, điều này cho thấy nó phải cài đặt trình điều khiển thiết bị cho các tài nguyên đó. Ví dụ, sử dụng phần mềm máy ảo VMware thì có thể cài đặt đồng thời nhiều hệ điều hành trên cùng một nền tảng phần cứng.

Như vậy, hệ điều hành giao tiếp với phần cứng qua lớp máy ảo, hình thức này ngày càng trở nên quan trọng trong việc giải quyết vấn đề tin cậy và bảo mật các hệ thống phân tán. Giám sát máy ảo cho phép cách ly toàn bộ các ứng dụng với môi trường của chúng, nếu xảy ra lỗi do tấn công bảo mật thì không còn ảnh hưởng đến toàn bộ máy. Ngoài ra, giám sát máy ảo còn nâng cao tính khả chuyển của hệ thống, nó tách biệt phần cứng với phần mềm và như vậy có thể chuyển toàn bộ phần mềm từ nền tảng phần cứng này sang nền tảng phần cứng khác.



Hình 5.6 Hai hình thức ảo hóa

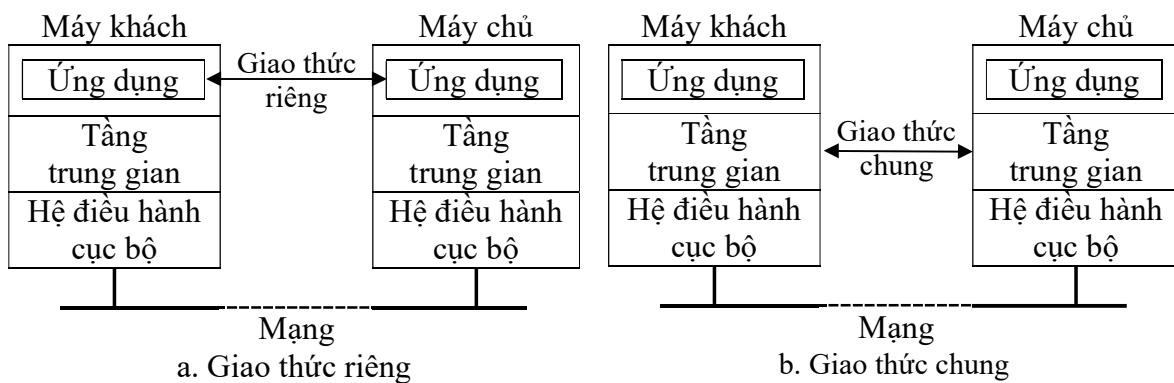
Mô hình như vậy đòi hỏi giám sát máy ảo phải xây lại toàn bộ những những chỉ thị lệnh như hệ điều hành đã thực hiện với thiết bị phần cứng, một giải pháp triển khai khác gọi là giám sát máy ảo lưu trữ. Thay vì làm lại toàn bộ những công việc như hệ điều hành khi giao tiếp với phần cứng, một cách tiếp cận khác gọi là giám sát máy ảo lưu trữ. Giám sát máy ảo lưu trữ chạy trên nền tảng của một hệ điều hành, nó có thể sử dụng các tiện ích và được cấp các đặc quyền như mức lỗi của hệ điều hành. Phía trên máy ảo lưu trữ có thể cài đặt nhiều hệ điều hành khác nhau để phục vụ cho các tiến trình ứng dụng, giám sát máy ảo lưu trữ được sử dụng rất phổ biến trong các hệ thống phân tán hiện đại như trung tâm dữ liệu và điện toán đám mây. Ứng dụng quan trọng nhất của ảo hóa trong hệ thống phân tán nằm ở điện toán đám mây, nó cung cấp các dịch vụ theo nhiều mức độ khác nhau. Phổ biến nhất là dịch vụ hạ tầng, khách hàng thuê máy chủ ảo, thực chất đó là cách chia sẻ máy chủ vật lý, mỗi khách hàng đều có cảm giác sở hữu máy chủ riêng, giá thành thấp hơn và tất nhiên hiệu suất cũng kém hơn.

### 5.3 Máy khách

Mô hình khách chủ thường được sử dụng để xây dựng các hệ thống phân tán, nhiệm vụ chính của máy khách là cung cấp giao diện cho người sử dụng để tương tác với máy chủ. Giao diện thân thiện với người sử dụng là điều quan trọng, nhưng quan trọng hơn là lại là vấn đề hiệu năng, yêu cầu của người sử dụng phải được đáp ứng ngay lập tức, làm cho người sử dụng không còn cảm giác làm việc trên mạng. Phần đầu chương này đã đề cập tới kỹ thuật xử lý đa luồng, phần này sẽ giới thiệu thêm một số kỹ thuật nhằm đáp ứng yêu cầu trên. Để nâng cao hơn nữa hiệu năng tổng thể của hệ thống, máy khách có thể thực hiện nhiệm vụ tiền xử lý dữ liệu trước khi chuyển yêu cầu đến máy chủ.

#### 5.3.1 Cung cấp dữ liệu cho máy khách

Việc cung cấp dữ liệu cho máy khách để hiển thị cho người sử dụng có thể thực hiện theo hình thức tạo một bản sao dữ liệu trên máy khách hoặc lấy dữ liệu từ máy chủ khi người sử dụng yêu cầu, nguyên lý hoạt động của chúng được minh họa trên hình 5.7. Với hình thức thứ nhất, mỗi khi người sử dụng yêu cầu dịch vụ, tiến trình máy khách chỉ cần truy nhập đến dữ liệu cục bộ và hiển thị cho người sử dụng. Cách làm này đảm bảo thời gian đáp ứng ngay lập tức, không bị ảnh hưởng độ trễ truyền thông và như thời gian xử lý trên máy chủ, tuy nhiên vẫn đề nhát quán lại là một thách thức rất lớn.



Hình 5.7 Hai hình thức cung cấp dữ liệu cho máy khách

Để giải quyết trở ngại này, với mỗi dịch vụ trên máy chủ sẽ có thành phần tương ứng trên máy khách để có thể liên lạc với dịch vụ qua mạng, như vậy tầng ứng dụng trên máy khách sử dụng giao thức riêng để đồng bộ với dữ liệu trên máy chủ. Đại diện cho hình thức này có thể kể đến các ứng dụng điện thoại di động và các ứng dụng dạng cửa sổ trên máy tính, các ứng dụng này sẽ sao chép dữ liệu của dịch vụ về máy khách để xử lý và hiển thị theo cách riêng của máy khách, việc đồng bộ dữ liệu của dịch vụ với máy chủ được thực hiện theo cách riêng của từng hệ thống. Cần phải thiết kế sao cho mỗi tiến trình tận dụng thời gian rỗi trên máy khách để quá trình đồng bộ hoàn toàn trong suốt với người sử dụng.

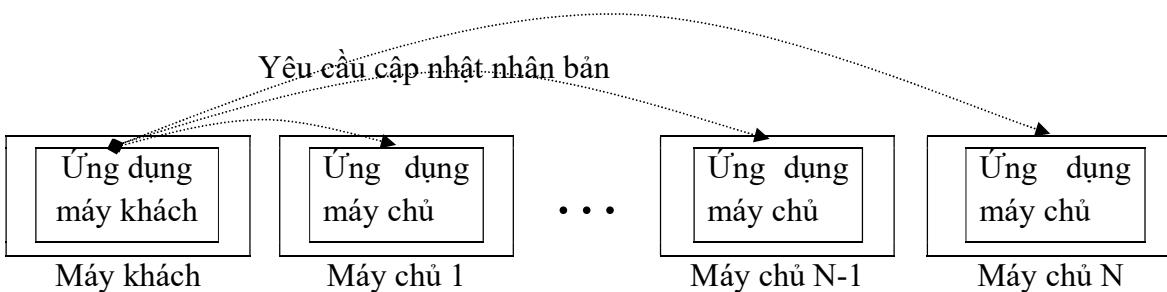
Hình thức thứ hai, máy khách không lưu trữ dữ liệu cục bộ mà sử dụng giao thức chung để lấy dữ liệu từ máy chủ và hiển thị cho người sử dụng, như vậy cả dữ liệu và các dạng hiển thị cho người sử dụng đều được lưu trữ trên máy chủ. Ví dụ thông tin của dịch

vụ được hiển thị trên các trình duyệt của máy khách, máy khách chỉ đóng vai trò hiển thị mà không có chức năng xử lý dữ liệu, cách tiếp cận này ngày càng trở nên phổ biến trong môi trường Internet.

Phân tích kỹ hơn qui trình thực hiện bên trong hệ thống khi người sử dụng truy nhập trang tin điện tử, trình duyệt gửi yêu cầu đến tiến trình máy chủ web, máy chủ web gửi một số yêu cầu đến máy chủ cơ sở dữ liệu và tạo trang web theo định dạng html để trả về cho trình duyệt. Như vậy tiến trình máy chủ web đảm nhiệm hai vai trò, vai trò chủ đối với trình duyệt của máy khách và vai trò khách đối với máy chủ cơ sở dữ liệu. Tạo bản sao dữ liệu trên máy khách là điều không phù hợp thực tế, số lượng máy khách thường rất lớn, khó có thể đồng bộ với dữ liệu trên máy chủ. Tuy nhiên, tạo bản sao dữ liệu trên máy chủ web lại là cách nên làm, nên tạo bản các trang dưới dạng tập tin html, mỗi khi trình duyệt gửi yêu cầu thì chỉ cần đọc tập tin này, không cần phải gửi yêu cầu truy vấn đến máy chủ cơ sở dữ liệu.

### 5.3.2 Tính trong suốt phân bố tài nguyên

Tiến trình trên máy khách không những đảm nhiệm các chức năng giao tiếp người sử dụng mà còn phải thực hiện nhiều nhiệm vụ khác như xử lý dữ liệu, truy nhập tài nguyên trên máy chủ..., những chức năng này càng trong suốt đối với người dùng càng tốt, trái ngược hẳn với quan điểm xây dựng các ứng dụng trên máy chủ.



Hình 5.8 Tính trong suốt truy nhập tài nguyên nhân bản

Tính trong suốt truy nhập được xử lý bằng cách tạo thành phần có giao diện như trên máy chủ nhưng che giấu kiến trúc máy và hình thức trao đổi thông tin. Vấn đề trong suốt phân bố tài nguyên thường được xử lý bằng hệ thống đặt tên, tuy nhiên trong những trường hợp máy khách đã kết nối tới máy chủ nào đó thì máy chủ có thể gửi thông tin trực tiếp đến tầng trung gian cài đặt trên máy khách để thực hiện kết nối đến máy chủ mới, điều này có thể tạm thời làm suy giảm hiệu năng của máy khách.

Hình 5.8 thể hiện hệ thống máy chủ được nhân bản, máy khách có thể gửi yêu cầu đến tất cả máy chủ nhưng sẽ chỉ sử dụng kết quả trả về từ 01 máy. Để đảm bảo tính trong suốt về lỗi, đối với lỗi truyền thông thì phần mềm trung gian trên máy khách có thể gửi lại một vài lần hoặc gửi yêu cầu đến máy chủ khác xử lý, trường hợp xấu nhất có thể lấy dữ liệu đã được phiên liền trước ghi nhớ trong vùng đệm dữ liệu. Tính trong suốt tương tranh thường được giải quyết trên các máy chủ, ví dụ sử dụng hệ thống giám sát tương tranh, máy khách ít khi phải xử lý vấn đề này.

## 5.4 Máy chủ

Máy chủ là máy chạy các tiến trình cung cấp dịch vụ theo yêu cầu của máy khách, chúng thường là những máy tính có cấu hình đủ lớn để luôn sẵn sàng đáp ứng các yêu cầu dịch vụ phục vụ cho người sử dụng. Các hệ thống qui mô lớn thường bố trí nhiều máy chủ cung cấp dịch vụ để phân tải xử lý và đồng thời đảm bảo khả năng chịu lỗi.

Thực chất, các máy chủ đều được tổ chức theo cách giống nhau, nó chờ yêu cầu từ máy khách gửi tới và thực thi yêu cầu và sau đó lại chờ yêu cầu tiếp theo. Thiết kế hệ thống máy chủ phải nhìn nhận cả hai góc độ, thứ nhất là cách tổ chức phần mềm và thứ hai là hình trạng vật lý.

### 5.4.1 Thiết kế phần mềm

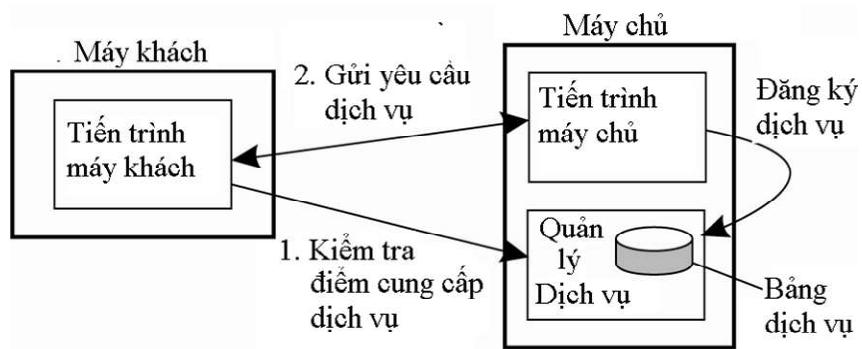
Trên quan điểm tổ chức phần mềm, nhiều hệ thống phân tán sử dụng mô hình phân tầng, nó tách biệt giữa các chức năng tiếp nhận yêu cầu với xử lý nghiệp vụ và lưu trữ dữ liệu. Tương tác giữa các tầng trong hệ thống thường sử dụng mô hình khách/chủ, thành phần máy chủ có thể thuộc loại tương tác hoặc tương tranh. Máy chủ tương tác tiếp nhận yêu cầu, sau khi xử lý xong và trả về kết quả cho máy khách thì mới tiếp tục nhận yêu cầu khác, như vậy tại một thời điểm chỉ có duy nhất một yêu cầu được xử lý. Máy chủ tương tranh tiếp nhận yêu cầu nhưng không xử lý mà chuyển yêu cầu đó cho luồng khác hoặc cho một tiến trình khác để xử lý và sau đó lại tiếp tục nhận các yêu cầu mới.

Kỹ thuật xử lý đa luồng thường được áp dụng cho cả hai loại máy chủ trên, tuy nhiên cách cài đặt hoàn toàn khác nhau. Đối với máy chủ tương tác, sau khi tiếp nhận yêu cầu của máy khách, tiến trình trên máy chủ có thể chia nhỏ nhiệm vụ xử lý cho nhiều luồng sau đó tổng hợp kết quả trả về cho tiến trình máy khách. Đối với máy chủ tương tranh, đương nhiên phải áp dụng kỹ thuật đa luồng, yêu cầu được chuyển cho luồng hoặc tiến trình khác xử lý, chúng có thể trả về kết quả cho tiến trình máy khách mà không cần gửi qua thành phần đã tiếp nhận yêu cầu.

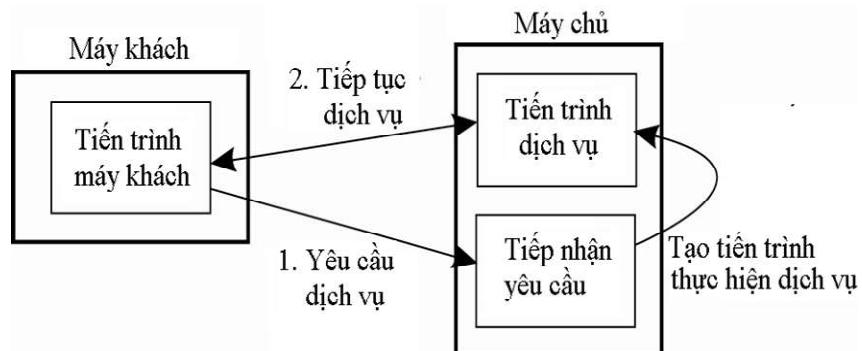
Một vấn đề quan trọng cần phải giải quyết đó là làm thế nào máy khách biết được điểm truy nhập dịch vụ, có thể cài đặt bằng cách sử dụng trực tiếp cổng dịch vụ hoặc thông qua một tiến trình khác. Tổ chức IANA đã qui định trên mỗi máy tính có 65636 cổng dịch vụ, các cổng từ 0-1024 là dành cho những dịch vụ công cộng. Như vậy, mỗi dịch vụ trên máy chủ sẽ được gán cho một cổng và tiến trình máy chủ thường xuyên nghe cổng đó. Hình 5.9.a thể hiện tiến trình máy chủ cài đặt trực tiếp trên cổng dịch vụ, máy khách muốn sử dụng dịch vụ thì phải cung cấp cặp thông tin địa chỉ máy chủ và số hiệu cổng dịch vụ.

Hình thức thứ hai, một số dịch vụ không sử dụng đến cổng, trong trường hợp này máy chủ phải gán điểm truy nhập dịch vụ động và gán cho mỗi hệ điều hành và đồng thời phải có một tiến trình đặc biệt gọi là daemon luôn theo dõi điểm truy nhập dịch vụ này. Thông thường mỗi điểm cuối sẽ được gán cho một dịch vụ riêng biệt, nếu cài đặt mỗi dịch vụ lại sử dụng các máy chủ riêng biệt sẽ lãng phí tài nguyên. Vấn đề này có thể giải quyết bằng cách cung cấp một tiến trình chuyên tiếp nhận yêu cầu của máy khách sau đó chuyển cho tiến trình khác tiếp tục xử lý dịch vụ, điều này được thể hiện trên hình 5.9.b.

a. Tiền trình máy chủ  
cài đặt trên công  
dịch vụ



b. Sử dụng tiền trình  
tiếp nhận yêu cầu



Hình 5.9 Cài đặt các tiến trình máy chủ

Một khía cạnh khác cần phải tính đến khi thiết kế phần mềm trên máy chủ là vấn đề làm thế nào máy chủ có thể ngừng khi chưa hoàn thành yêu cầu xử lý dịch vụ. Ví dụ trường hợp máy khách đột ngột hủy trong khi máy chủ đang thực thi yêu cầu, khi đó kết nối giữa máy khách và máy chủ sẽ bị hủy bỏ, máy chủ sẽ chờ một khoảng thời gian nhất định sau đó sẽ hủy bỏ liên kết. Tuy nhiên, giải pháp này sẽ không tốt trong trường hợp liên kết giữa máy khách và máy chủ bị gián đoạn do nguyên nhân khách quan như chất lượng kênh truyền kém..., giải pháp tốt hơn sẽ là cung cấp báo hiệu kênh ngoài cho liên kết giữa máy khách và máy chủ hoặc thiết kế giao thức có khả năng điều khiển tương tác.

Điểm cuối cùng trong thiết kế máy chủ là vấn đề có lưu vết trạng thái hay không. Nếu không lưu giữ thông tin trạng thái của máy khách thì máy chủ có thể tùy ý thay đổi trạng thái của mình mà không cần báo lại cho máy khách. Nếu lưu trạng thái của máy khách thì máy chủ không những không được phép tùy ý thay đổi trạng thái mà còn tăng dung lượng lưu trữ, do đó có thể dung hòa hai cách tiếp cận trên bằng cách lưu trạng thái của máy khách trong một khoảng thời gian nhất định.

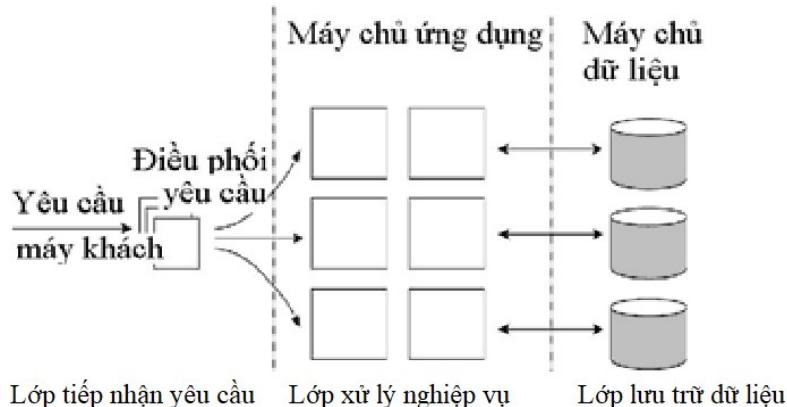
#### 5.4.2 Lắp đặt máy chủ

Đối với những hệ thống qui mô nhỏ thì chỉ cần một máy chủ, phần mềm theo kiến trúc phân tầng đều được cài đặt trên một máy, điều đó tiềm ẩn một loạt nguy cơ về khả năng chịu lỗi và khả năng bảo mật. Nếu qui mô hệ thống mở rộng thì phải phân tải xử lý trên nhiều máy chủ, điều đó không những đảm bảo yêu cầu về hiệu năng mà còn tăng khả năng chịu lỗi cũng như các vấn đề an toàn bảo mật thông tin.

Nếu mở rộng qui mô cho số lượng người sử dụng thì chỉ cần triển khai hệ thống điện toán cụm, đó là một trong ba loại cơ bản của hệ thống phân tán. Nếu qui mô rộng về

phạm vi địa lý thì triển khai các hệ thống máy chủ phân tán, mỗi máy chủ sẽ phụ trách cung cấp dịch vụ cho người sử dụng trong phạm vi địa lý nhất định. Việc lắp đặt các máy chủ phân tán không những rút ngắn đường đi của các gói tin từ máy khách đến máy chủ mà còn giải quyết vấn đề tắc nghẽn mạng, do đó thời gian đáp ứng các yêu cầu của người sử dụng.

Cụm máy chủ là tập hợp các máy được kết nối qua mạng, trên mỗi máy chạy một hoặc nhiều tiến trình máy chủ. Thông thường cụm máy chủ được kết nối trong mạng cục bộ để bảo đảm băng thông đủ lớn nhằm mục đích giảm thiểu độ trễ khi trao đổi thông tin giữa các máy chủ trong cụm. Cụm máy chủ thường được bố trí thành ba lớp để phù hợp với mô hình phân tầng của phần mềm, tùy theo qui mô người sử dụng, mỗi lớp có thể gồm nhiều máy chủ được tổ chức theo kiến trúc phân cấp hoặc kiến trúc ngang hàng.



Hình 5.10 Tổ chức cụm máy chủ theo mô hình ba lớp

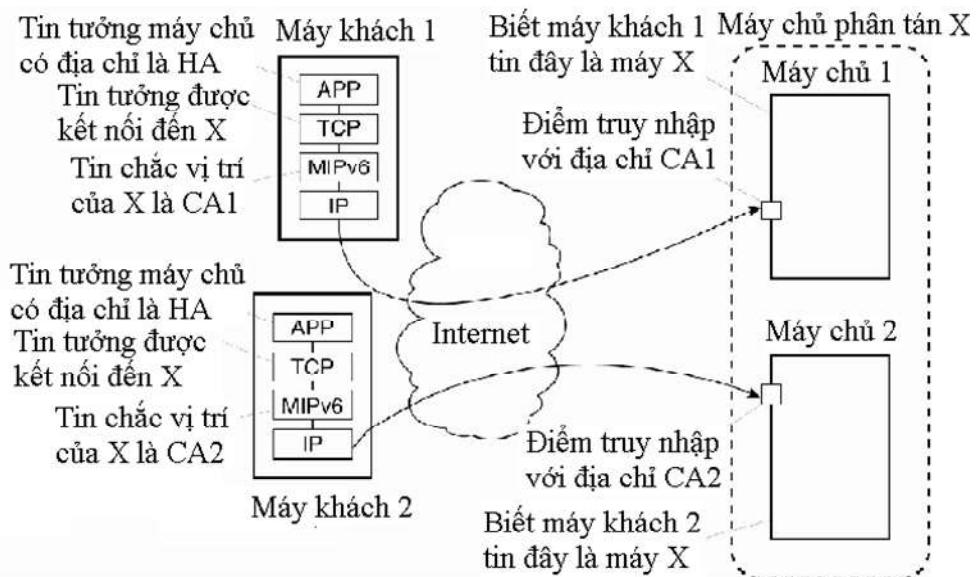
Hình 5.10 thể hiện cách tổ chức cụm máy chủ theo mô hình ba lớp, nó tách biệt cách chức năng chính của ba khối phần mềm trong hệ thống phân tán, cách lắp đặt này đáp ứng tất cả các yêu cầu không những làm tăng hiệu năng xử lý mà còn đảm bảo độ tin cậy cho hệ thống. Máy khách chỉ giao tiếp với thành phần tiếp nhận yêu cầu, hệ thống che đậy toàn bộ các máy chủ xử lý nghiệp vụ và các máy chủ lưu trữ dữ liệu, điều đó không những tăng khả năng bảo mật hệ thống mà còn góp phần tiết kiệm địa chỉ IP công cộng vốn đang hạn hẹp. Lớp tiếp nhận yêu cầu chỉ cần máy tính có bộ vi xử lý tốc độ cao để có khả năng tiếp nhận số lượng lớn yêu cầu của các máy khách, nếu máy chủ này ngừng hoạt động hoặc bị quá tải thì sẽ ảnh hưởng đến toàn bộ hệ thống.

Các máy chủ tại lớp xử lý nghiệp vụ thì cần phải có những bộ vi xử lý tốc độ cao và bộ nhớ đủ lớn để đáp ứng các tác vụ tính toán, mỗi máy chủ sẽ cài đặt phần mềm nghiệp vụ riêng nhằm nâng cao tính chuyên môn hóa. Để giải quyết các vấn đề phân tải xử lý cũng như khả năng dự phòng nóng, mỗi chức năng nghiệp vụ nên được cài đặt trên một số máy chủ, các hệ điều hành hiện nay đều đã hỗ trợ cài đặt theo kiểu chính phủ.

Các máy chủ tại lớp lưu trữ dữ liệu thường lưu trữ các tập tin hoặc cài đặt các hệ quản trị cơ sở dữ liệu, chúng đòi hỏi dung lượng ổ đĩa lớn, khả năng đọc/ghi nhanh và đồng thời có khả năng chịu lỗi. Hầu hết các ổ đĩa trên các máy chủ này đều sử dụng công nghệ RAID, đó là hình thức ảo hóa ghép nhiều đĩa cứng vật lý thành một hệ thống ổ đĩa

cứng có chức năng gia tăng tốc độ đọc/ghi dữ liệu và tăng thêm tính an toàn cho dữ liệu. Nhiều hệ thống áp dụng kỹ thuật nhân bản dữ liệu, điều đó không những tăng cường khả năng an toàn mà còn nâng cao hiệu năng hệ thống, giảm tính tương tranh. Một số dịch vụ không đòi hỏi truy nhập đến máy chủ tại lớp này, ví dụ dịch vụ hội thảo trực tuyến, dung lượng dữ liệu lớn và yêu cầu thời gian thực nên khó khả thi trong thực tế.

Nếu hệ thống mở rộng theo qui mô địa lý, người sử dụng nằm rải rác ở những vị trí khác nhau, việc tập trung tiếp nhận yêu cầu tại một cụm máy chủ có thể dẫn đến hiện tượng nghẽn mạng. Hơn nữa, cụm máy chủ với cấu hình cao dẫn đến chi phí đầu tư lớn, trường hợp này nên tổ chức máy chủ máy chủ cấu hình thấp hơn nhưng chúng phân tán rải rác theo vị trí địa lý. Thay vì chỉ có một thành phần tiếp nhận yêu cầu sẽ bỏ chí nhiều thành phần tiếp nhận yêu cầu, đây là mô hình đã áp dụng trong hệ thống phân giải tên miền.



Hình 5.11 Định tuyến máy chủ phân tán

Hệ thống máy chủ phân tán đòi hỏi phải cài đặt thêm bộ tối ưu hóa định tuyến nhằm tạo điều kiện cho các máy khách tìm được máy chủ dịch vụ một cách nhanh nhất. Ngoài ra, máy khách cũng phải được cấu hình để biết thông tin về các máy chủ tiếp nhận yêu cầu, ví dụ trên giao diện mạng của mỗi máy khách qui định địa chỉ của máy chủ DNS mặc định, máy khách sẽ liên lạc với máy chủ này mỗi khi có yêu cầu phân giải tên miền.

Hệ thống máy chủ phân tán bao gồm tập các máy chủ cung cấp dịch vụ cho người sử dụng và một máy chủ gốc đóng vai trò điểm truy nhập, địa chỉ HA của nó công khai cho máy khách. Máy chủ gốc phải có cấu hình rất mạnh với hiệu suất cao là những yêu cầu dễ dàng đạt được, nhưng để hoạt động ổn định trong thời gian dài là vấn đề rất khó. Nên lắp đặt cụm máy tính có cấu hình thấp với giá thành thấp hơn, chúng hoạt động theo cơ chế dự phòng nóng, nếu lỗi xảy ra trên một máy tính thì có ngay máy tính khác thay thế. Mỗi máy chủ cung cấp dịch vụ có địa chỉ riêng, đây là địa chỉ hoàn toàn trong suốt với máy khách, các chức năng xử lý của chúng y hệt nhau, nghĩa là yêu cầu gửi đến máy

chủ nào thì cũng nhận về kết quả giống nhau. Nếu máy chủ cung cấp dịch vụ thay đổi địa chỉ thì phải thông báo cho máy chủ gốc để nó cập nhật thông tin trên các máy khách.

Hình 5.11 minh họa cơ chế hoạt động của hệ thống máy chủ phân tán, máy khách nhìn nhận hệ thống phân tán chỉ là một “máy chủ X“ với địa chỉ điểm truy nhập HA, máy chủ phân tán bao gồm hai máy chủ dịch vụ với địa chỉ là CA1 và CA2. Khi máy khách truy nhập vào “máy chủ X“, yêu cầu của nó được gửi đến điểm truy nhập và sau đó được chuyển tiếp đến một trong những máy chủ cung cấp dịch vụ để khởi tạo quá trình tối ưu hóa định tuyến. Trong trường hợp máy khách 1, kết quả quá trình tối ưu hóa định tuyến cho thấy kết nối đến máy chủ CA1 là phương án tối ưu nhất, máy khách sẽ lưu cắp giá trị (HA, CA1) dùng để truy nhập cho những lần sau. Máy khách 2 truy nhập hệ thống, quá trình tương tự được thực hiện để máy khách 2 tin tưởng dịch truy nhập đến máy chủ CA2, nó lưu trữ cắp giá trị (HA, CA2) dùng để truy nhập cho những lần sau. Như vậy các máy khách khác nhau sẽ được những máy chủ vật lý khác nhau phục vụ, trong khi tiến trình trên máy khách vẫn có cảm giác như chỉ đang truy nhập vào một máy chủ.

#### **5.4.3 Quản lý cụm máy chủ**

Người quản trị dụng luôn mong muốn có thể quản lý cụm máy chủ phải tương tự như trên một máy tính, thực tế cho thấy công việc này đang được thực hiện bằng tiện ích truy nhập từ xa. Giả sử cụm có N nút và xác suất xảy ra lỗi trên mỗi nút là p, xác suất m nút đồng thời xảy ra lỗi được tính theo công thức của lý thuyết xác suất thống kê như sau:

$$P(X = m) \binom{N}{m} p^m (1-p)^{N-m}$$

Ví dụ cụm máy chủ gồm N=1000 nút, xác suất xảy ra lỗi trên mỗi máy chủ là p=0.001, áp dụng công thức trên để tính xác suất không có máy nào bị lỗi sẽ cho kết quả như sau:

$$\begin{aligned} P(X = 0) & \binom{1000}{0} p^0 (1-p)^{1000} \\ & = \frac{1000!}{0!1000!} (0.001)^0 (0.999)^{1000} \\ & = 0.367695 \end{aligned}$$

Xác suất không máy chủ nào bị lỗi chỉ khoảng 37%, điều đó chứng tỏ một tỉ lệ khá cao người quản trị phải truy nhập vào mỗi máy chủ để quản lý. Một giải pháp khác là xây dựng phần mềm cài đặt trên máy tính của người quản trị, mỗi máy chủ trong cụm được thể hiện bằng một nút và người quản trị có thể thêm hoặc bớt bất kỳ máy chủ nào.

Như vậy, thay vì phải truy nhập vào từng máy thì phần mềm quản lý sẽ thu thập thông tin từ mỗi máy chủ và cung cấp giao diện quản lý cho người quản trị. Nhiều phần mềm như vậy đã được xây dựng, có cả mã nguồn mở, tuy nhiên giải pháp này chỉ phù hợp với mô hình nhỏ, vấn đề sẽ phức tạp hơn rất nhiều đối với cụm máy chủ có quy mô lớn. Ngay cả phần mềm quản lý cụm máy chủ của hãng IBM cũng đưa ra khuyến nghị chỉ nên quản lý 128 nút, vì vậy giải pháp quản lý cụm máy chủ lớn vẫn đang tiếp tục nghiên cứu.

## 5.5 Di trú mã

Từ đầu chương trình chúng ta mới chỉ tập trung vào vấn đề trao đổi dữ liệu trong hệ thống phân tán mà chưa đề cập đến vấn đề di chuyển các thành phần khác của tiến trình. Để tăng hiệu năng và tính linh hoạt của hệ thống, đôi khi phải di chuyển di chuyển các chương trình hoặc đoạn mã chương trình, quá trình đó gọi là di trú mã, đoạn mã có thể là mã nguồn hoặc mã nhị phân.

Cần phân biệt di trú mã với gọi thủ tục từ xa, tuy chúng đều thực hiện trên máy tính khác nhưng gọi thủ tục từ xa đòi hỏi kết nối mạng phải liên tục, nếu mất kết nối mạng trong khi thủ tục từ xa đang thực hiện thì sẽ phát sinh lỗi. Di trú mã hoạt động độc lập với nguồn của nó, sau khi đã chuyển sang máy tính khác, đoạn mã vẫn tiếp tục chạy ngay cả khi đã ngắt kết nối mạng.

Thông thường, di trú mã trong các hệ thống phân tán thực hiện dưới dạng di trú tiến trình, toàn bộ tiến trình được di chuyển từ máy này sang máy khác, đây là công việc đòi hỏi chi phí cao và phức tạp. Lý do phải di chuyển tiến trình vẫn là vấn đề hiệu năng của hệ thống, các yêu cầu xử lý cần phải được chuyển tới những máy ít tải hơn, tải ở đây được hiểu là tỉ lệ sử dụng bộ vi xử lý, bộ nhớ và một số yếu tố khác.

Các giải thuật phân tải đưa ra quyết định liên quan đến vấn đề xác định và phân phối lại nhiệm vụ dựa trên số lượng bộ xử lý đóng vai trò quan trọng trong các hệ thống tính toán với tần suất lớn. Tuy nhiên, trong các hệ thống phân tán hiện đại, vấn đề truyền dữ liệu được coi trọng hơn việc tối ưu hóa khả năng tính toán. Ngoài ra, vì các lý do liên quan đến vấn đề nền tảng không thông nhất hoặc mạng máy tính mà quyết định di trú mã có tính chất định tính chứ không dựa trên các mô hình toán học.

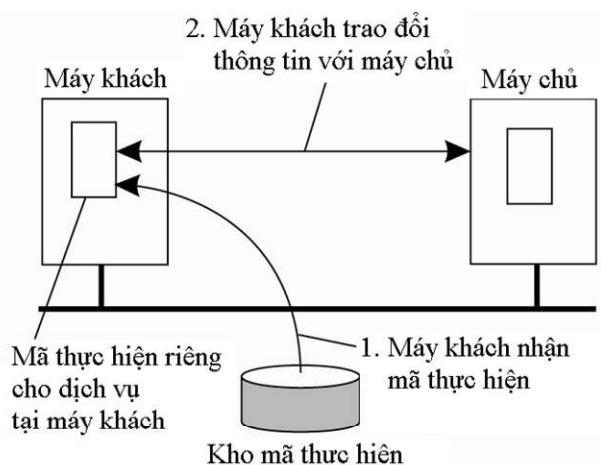
Nếu máy khách cần phải tải một lượng dữ liệu lớn từ máy chủ về để xử lý thì có thể di trú đoạn mã trên máy khách về máy chủ và khi đó máy khách chỉ nhận kết quả đã được tính toán trên tập dữ liệu máy khách đã yêu cầu, như vậy đã giảm lượng dữ liệu lưu chuyển trên mạng. Ngược lại, đối với những trường hợp người sử dụng phải cung cấp thông tin để máy chủ xử lý, thao tác kiểm tra tính hợp lệ của thông tin đầu vào nên được thực hiện trên máy khách, như vậy sẽ giảm tải cho máy chủ.

Di trú mã cũng cần thiết trong các trường hợp xử lý song song, nhưng không phải trên một máy tính. Ví dụ, để thực hiện nhiệm vụ tìm kiếm thông tin trên mạng, người ta sử dụng một đoạn mã di trú gọi là Agent, đoạn mã này sẽ di chuyển từ trang này sang trang khác, khi dừng lại ở trang nào nó sẽ tiếp tục nhận bản đoạn mã đó. Ngoài ra, di trú mã còn làm cho hệ thống phân tán linh hoạt hơn, ví dụ mô hình nhiều bên trong các ứng dụng khách/chủ.

### 5.5.1 Các giải pháp di trú mã

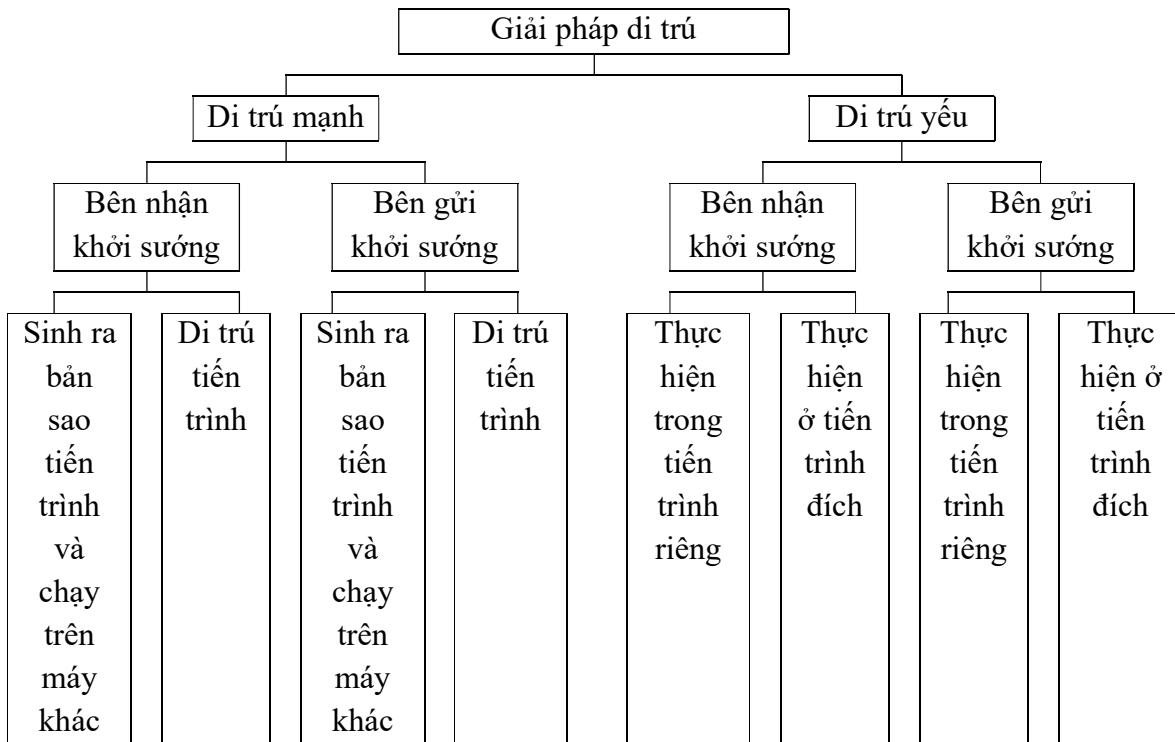
Cách phát triển phần mềm trước đây thường sử dụng phương pháp tĩnh, các thủ tục xử lý thông tin trên máy khách được biên dịch cùng với chương trình chính, khi nâng cấp phần mềm đòi hỏi người sử dụng phải cài đặt lại. Ví dụ, nếu máy khách muốn tải tập tin từ máy chủ thì phải xây dựng giao thức riêng trong đó qui định cấu trúc từng loại tập tin, máy khách sẽ dựa trên giao thức này để liên kết với các ứng dụng trên máy chủ. Như vậy,

giao thức được xây dựng trước khi phát triển phần mềm trên máy khách, các thủ tục xử lý cấu trúc tập tin được đặt trong thư viện và biên dịch tĩnh.



Hình 5.12 Nguyên lý cấu hình động cho máy khách

Cách phát triển phần mềm ngày nay sử dụng giải pháp động, các đoạn mã được đặt trong kho lưu trữ trên máy chủ, máy khách sẽ tải những đoạn mã cần thiết khi khởi tạo tiến trình. Hình 5.12 minh họa quá trình di trú mã từ máy chủ về máy khách, một thư viện liên kết động hoặc các đoạn mã chương trình được đặt trong kho “mã thực hiện” tại máy chủ. Khi tiến trình máy khách kết nối đến máy chủ, đầu tiên nó kiểm tra phiên bản thư viện, nếu cần thiết thì tải các tập tin hoặc một số đoạn mã cần thiết về máy khách.



Hình 5.13 Phân loại di trú

Theo quan điểm của Fuggetta, mỗi tiến trình bao gồm ba phần: phần mã, phần tài nguyên và phần đang thực thi. Phần mã bao gồm tập các lệnh của chương trình đang chạy, đây là thành phần xử lý nghiệp vụ. Phần tài nguyên bao gồm các tham chiếu đến tất cả các tài nguyên bên ngoài mà tiến trình đang sử dụng, ví dụ các tập tin hay các thiết bị ngoại vi khác. Phần đang thực thi chứa các trạng thái hiện hành của tiến trình, nó bao gồm dữ liệu riêng, các ngăn xếp và tất nhiên bộ đếm chương trình.

Di trú mã được hiểu là di chuyển một phần hay toàn bộ tiến trình, người ta phân làm hai loại di trú yếu và di trú mạnh, hình 5.13 liệt kê chi tiết các loại di trú. Di trú yếu chỉ truyền phần mã và một số các dữ liệu khởi động, một chương trình được chuyển đi luôn bắt đầu từ một số vị trí khởi động đã xác định trước. Loại di trú này khá đơn giản, nó chỉ yêu cầu máy đích thực hiện đoạn mã chương trình đã di trú. Di trú mạnh truyền cả phần mã và phần thực thi, tiến trình đang chạy có thể tạm dừng để chuyển sang máy tính khác và tiếp tục thực hiện. Di trú mạnh tổng quát hơn di trú yếu, nó không cần bắt đầu lại từ đầu mà chỉ tiếp tục công việc đã tạm ngừng, việc triển khai tương đối phức tạp. Ví dụ tác tử di động, nó di chuyển từ máy tính này sang máy tính khác mà không cần phải khởi động lại, việc thực thi tiếp tục ở lệnh tiếp theo trong tác tử.

Dù là di trú yếu hay mạnh, nếu máy tính lưu trữ đoạn mã khởi tạo di trú gọi là bên gửi khởi sướng, ví dụ các tác tử tìm kiếm qua mạng Internet đến máy chủ web. Ngược lại, nếu máy tính chủ động lấy đoạn mã từ máy tính khác gọi là di trú do bên nhận khởi sướng, đây là giải pháp thường được sử dụng để nâng cao hiệu năng. Ví dụ, các đoạn mã tiền kiểm tra dữ liệu thường được tải từ máy chủ khi người sử dụng truy nhập trang tin điện tử nhưng nó lại thực hiện trên máy khách. Không phụ thuộc bên nào khởi sướng, đoạn mã di trú yếu có thể chạy ở tiến trình đích hoặc khởi tạo một tiến trình riêng. Trong di trú mạnh, thay vì di trú tiến trình đang chạy, có thể tạo một bản sao của tiến trình đó và chạy trên máy tính khác song song với tiến trình hiện hành.

### 5.5.2 *Di trú và tài nguyên cục bộ*

Tài nguyên cục bộ thuộc sở hữu của một máy tính, di trú đoạn mã hay tiến trình đang thực thi giữ nguyên giống với bản gốc nhưng tài nguyên cục bộ có thể thay đổi. Ví dụ, một tiến trình đang sử dụng một cổng nào đó, khi di trú sang máy tính khác, nếu cổng đó đã được sử dụng thì phải thiết lập cổng mới thay cho cổng đang sử dụng. Tuy nhiên, không phải tài nguyên cục bộ nào cũng bị thay đổi khi di trú, ví dụ tham chiếu đến tài nguyên thông qua đường dẫn tuyệt đối sẽ luôn hợp lệ dù cho nó được sử dụng trên bất kỳ máy tính nào.

Fuggetta phân biệt ba mức tham chiếu giữa tiến trình và tài nguyên: Mạnh, yếu và rất yếu. Ở mức thứ nhất, tiến trình tham chiếu đến tài nguyên thông qua định danh, ví dụ các tiến trình sử dụng đường dẫn của một trang tin điện tử hoặc máy chủ truyền tập tin. Mức thứ hai, tiến trình chỉ đòi hỏi giá trị của tài nguyên, hoạt động của tiến trình không hề bị ảnh hưởng nếu cung cấp các giá trị như nhau, điều này tương tự như việc gọi thủ tục trong các thư viện lập trình. Mức thứ ba, tiến trình chỉ tham chiếu đến loại tài nguyên, ví dụ tham chiếu đến như máy in và màn hình hoặc các thiết bị ngoại vi khác.

Khi di trú mã thì chỉ có thể thay đổi tham chiếu đến tài nguyên mà không được phép thay đổi mức tham chiếu, điều này phụ thuộc tài nguyên có thể di chuyển sang máy tính khác hay không, nghĩa là phụ thuộc vào mức gắn kết giữa tài nguyên với máy tính. Những tài nguyên không gắn kết với máy tính thì có thể di trú với những mức độ phức tạp khác nhau, một số trường hợp có thể di trú nhưng với chi phí rất cao. Mức gắn kết tập tin với máy tính thường rất thấp, do đó có thể di chuyển chúng đến bất kỳ máy tính nào. Hệ quản trị cơ sở dữ liệu có mức gắn kết cao hơn, có thể di trú SQL Server tương đối dễ dàng nhưng khó có thể di trú Oracle, nó liên quan đến nhiều biến môi trường trên mỗi máy tính. Khó có thể di trú những tài nguyên gắn chặt với máy tính, ví dụ các thiết bị ngoại vi gắn vào máy tính hoặc các cổng dịch vụ.

**Bảng 5.2** Tổ hợp tham chiếu tài nguyên và mức độ gắn kết với máy tính

Mức gắn kết Mức tham chiếu	Gắn kết thấp	Gắn kết cao	Cố định
<b>Mạnh</b>	MV hoặc GR	GR hoặc MV	GR
<b>Yếu</b>	CP, MV hoặc GR	GR hoặc CP	GR
<b>Rất yếu</b>	RB, MV hoặc CP	RB, GR hoặc CP	RB hoặc GR

*Chú thích viết tắt:*

GR Thiết lập tham chiếu toàn cục, ví dụ đường dẫn liên kết URL

MV Di chuyển tài nguyên

CP Sao chép giá trị của tài nguyên

RB Tiến trình nhúng lại sang tài nguyên cục bộ

Bảng 5.2 tổng kết tổ hợp các khả năng tham chiếu và mức độ gắn kết tài nguyên với máy tính, cần phải xem xét sử dụng mỗi loại khi cài đặt di trú mã, mức độ ưu tiên sử dụng từ trái sang phải. Khi tiến trình tham chiếu đến tài nguyên ở mức độ mạnh, nếu tài nguyên không được gắn kết thì tốt nhất di chuyển nó cùng với mã di trú, nếu phải chia sẻ cho các tiến trình khác thì nên thiết lập tham chiếu toàn cục. Nếu tài nguyên gắn kết cao thì nên sử dụng tham chiếu toàn cục hoặc di chuyển tài nguyên nhưng nếu là tài nguyên cố định thì chỉ có thể sử dụng tham chiếu toàn cục.

Tham chiếu toàn cục không đơn thuần chỉ sử dụng đường dẫn liên kết, việc sử dụng tham chiếu này đôi khi trả giá về lưu lượng truyền thông, như trên đã đề cập vấn đề này, để nâng cao hiệu năng thì phải giảm thiểu trao đổi thông tin trên mạng. Ví dụ, để tạo hình ảnh chất lượng cao thì có thể chuyển đến máy tính chuyên dụng, có thể thiết lập tham chiếu toàn cục đến máy tính này lại dẫn đến nhu cầu băng thông mạng cao khi truyền hình ảnh.

Thiết lập tham chiếu toàn cục không phải lúc nào cũng dễ dàng, ví dụ khi di trú tiến trình đang sử dụng điểm kết cuối truyền thông, nghĩa là tiến trình bị ràng buộc địa chỉ của máy tính và số hiệu cổng. Có thể giải quyết vấn đề này bằng cách tiến trình di trú sang máy tính khác nhưng vẫn duy trì liên lạc với máy tính gốc, mỗi khi có thông điệp chuyển đến thì máy tính gốc sẽ chuyển tiếp sang máy tính mới, nhưng điều này dẫn đến độ trễ cao hơn và tiềm ẩn rủi ro máy tính gốc bị lỗi. Giải pháp tốt hơn là tiến trình di trú

tạo điểm kết cuối mới, tiến trình gốc sẽ thông báo cho tất cả những tiến trình liên quan cập nhật tham chiếu toàn cục.

Nếu tham chiếu ở mức độ yếu, nghĩa là chỉ ràng buộc về giá trị, nếu đó là tài nguyên gắn kết thấp hoặc gắn kết cao thì chỉ cần sao chép dữ liệu, nhưng đối với tài nguyên cố định thì phức tạp hơn nhiều và thậm chí khó khả thi trong thực tế. Ví dụ việc chia sẻ bộ nhớ giữa các tiến trình trên các máy tính khác nhau, nghĩa là xây dựng bộ nhớ dùng chung phân tán, đây thực sự là điều khó thực hiện. Tài nguyên gắn kết cao được tham chiếu theo giá trị của chúng thường là những thư viện liên kết động, bao sao của các tài nguyên như vậy thường có sẵn trên máy đích hoặc nếu không thì phải sao chép trước khi di trú mã. Đối với những tài nguyên ở mức độ gắn kết thấp, cách tốt nhất vẫn là sao chép sang máy tính mới, tuy nhiên nếu tài nguyên đó được dùng chung cho nhiều tiến trình thì nên tạo tham chiếu toàn cục.

Trường hợp cuối cùng là tham chiếu rất yếu, nó chỉ ràng buộc loại tài nguyên, bất kể đó là loại gắn kết hay cố định thì chỉ cần tiến trình nhúng lại trên tài nguyên cục bộ cùng loại. Trường hợp tài nguyên không có trên máy tính mới, những tài nguyên cố định thì phải tạo tham chiếu toàn cục, những tài nguyên khác thì chỉ cần sao chép hoặc di trú tài nguyên gốc sang máy tính mới hoặc cũng có thể tạo tham chiếu toàn cục.

### 5.5.3 *Di trú trong hệ thống không đồng nhất*

Di trú mã trên các hệ thống đồng nhất sẽ không gặp những trở ngại lớn là do các máy tính cùng chạy trên một nền tảng phần cứng và hệ điều hành, vấn đề sẽ trở nên khó khăn hơn rất nhiều nếu hệ thống bao gồm các máy tính sử dụng nền tảng khác nhau. Nhiều máy tính dòng Big Endian và Little Indian hay những máy tính 64 bit và 32 bit vẫn song hành tồn tại, các máy tính trên thế giới có thể cài đặt nhiều loại hệ điều hành với những phiên bản khác nhau, việc di trú ngay cả những tài nguyên ở mức gắn kết thấp cũng rất khó khăn. Ngoài ra, vấn đề ngôn ngữ lập trình cũng gây trở ngại lớn trong di trú mã, các ứng dụng thường được lập trình bằng các ngôn ngữ lập trình bậc cao như Pascal, C hay Java..., và tiếp tục sẽ còn nhiều ngôn ngữ lập trình mới ra đời.

Việc di trú trong các hệ thống như vậy đòi hỏi phải hỗ trợ từng nền tảng, nghĩa là đoạn mã có thể được thực thi đúng trên các nền tảng. Mặc dù ý tưởng ảo hóa đã được đề xuất từ những năm 1970, tuy nhiên nó chưa bao giờ là giải pháp chung cho vấn đề di trú mã, đặc biệt là đối với những phần mềm được viết bằng ngôn ngữ C, ngôn ngữ này được coi là ngôn ngữ lập trình bậc cao nhưng hiệu năng của nó gần giống như ngôn ngữ lập trình bậc thấp. Những phát triển gần đây đã giảm sự phụ thuộc vào ngôn ngữ lập trình, các giải pháp đã được đề xuất không chỉ để di trú các tiến trình còn để di trú toàn bộ môi trường điện toán. Ý tưởng cơ bản là phân chia môi trường tổng thể và cung cấp cho các tiến trình trong cùng một phần gốc nhìn riêng về môi trường điện toán của chúng, việc phân chia đó diễn ra dưới dạng các giám sát máy ảo chạy hệ điều hành và một bộ ứng dụng.

Với việc di trú máy ảo, có thể tách môi trường điện toán khỏi hệ thống cơ bản và thực sự di chuyển nó sang một máy tính khác, các tiến trình có thể không biết về bản thân quá trình di chuyển, chúng không cần phải bị gián đoạn trong quá trình thực thi và cũng

không gặp bất kỳ trở ngại nào với các tài nguyên đã sử dụng. Clark đề xuất di trú hệ điều hành ảo theo thời gian thực, vấn đề này quan đến di chuyển toàn bộ hình ảnh bộ nhớ và di chuyển các ràng buộc tham chiếu sang tài nguyên cục bộ.

Đối với vấn đề di chuyển toàn bộ hình ảnh bộ nhớ có thể thực hiện theo ba cách. Cách thứ nhất, đẩy các trang bộ nhớ sang máy mới và sẽ cập nhật lại những trang bị thay đổi trong quá trình di trú. Cách thứ hai, ngừng hoạt động của máy ảo, di trú bộ nhớ và khởi tạo máy ảo mới, như vậy sẽ có một khoảng thời gian ngừng hoạt động, điều này không thể chấp nhận đối với các hệ thống yêu cầu dịch vụ liên tục. Cách thứ ba, cho phép máy ảo mới kéo các trang bộ nhớ mới khi cần thiết, nghĩa là khởi tạo các tiến trình trên máy ảo mới và sao chép các trang bộ nhớ theo yêu cầu của tiến trình đó, điều này dẫn tới hiện tượng di trú kéo dài và do đó hiệu năng thấp.

Đối với tài nguyên cục bộ, chỉ cần chuyển ràng buộc tham chiếu sang tài nguyên mới, nghĩa là thiết lập lại kết nối mạng. Nếu hệ thống được tổ chức theo mô hình cụm máy chủ với các lớp xử lý riêng biệt thì việc thiết lập lại liên kết khá đơn giản. Như vậy, khái niệm di trú không phải chỉ bó hẹp trong phạm vi tiến trình mà còn có thể mở rộng sang cả hệ điều hành.

## THẢO LUẬN

1. Nêu những nhược điểm khi áp dụng luồng khi xây dựng phần mềm.
2. Xây dựng ứng dụng đa luồng chế độ người sử dụng và chế độ lõi của hệ điều hành cho một ứng dụng khách chủ đơn giản, ví dụ gọi đối tượng từ xa.
3. Xây dựng ứng dụng máy khách gồm 2 truy nhập đồng thời vào hai trang tin điện tử khác nhau, đánh giá thời gian đáp ứng.
4. Xây dựng ứng dụng máy chủ đơn luồng tìm kiếm các xâu ký tự trong các tập tin văn bản.
5. Xây dựng ứng dụng máy chủ đa luồng tìm kiếm các xâu ký tự trong các tập tin văn bản.
6. Xây dựng ứng dụng máy chủ trạng thái tìm kiếm các xâu ký tự trong các tập tin văn bản, đánh giá hiệu năng so với phương pháp đơn luồng và đa luồng.
7. Xây dựng thư viện ảo hóa truy nhập các hệ quản trị cơ sở dữ liệu khác nhau như Oracle, SQL Server, MySql.
8. Viết hàm kiểm tra tính hợp lệ của thư điện tử bằng Java Script, giải thích nguyên lý hoạt động khi cài đặt.
9. Xây dựng ứng dụng truy vấn bảng khoảng 20 000 bản ghi nhưng chỉ cần hiển thị trên màn hình 50 bản ghi và thanh định hướng sang những trang tiếp theo bằng hai phương pháp như đã trình bày trong phần di trú mã, so sánh thời gian đáp ứng.
10. Tìm hiểu cơ chế hoạt động của điện toán đám mây AWS, đánh giá ưu điểm và nhược điểm của sản phẩm này.

## CHƯƠNG 6: QUẢN TRỊ GIAO TÁC VÀ ĐIỀU KHIỂN TƯƠNG TRANH

Tương tranh là hiện tượng tại một thời điểm có nhiều yêu cầu sử dụng tài nguyên dùng chung, theo Edsger Dijkstra: "Tương tranh xảy ra khi nhiều hơn một luồng thực thi có thể chạy đồng thời". Nhiều tiến trình truy nhập đồng thời đến các tài nguyên dùng chung như bộ nhớ, kênh truyền thông, các tập tin trên ổ đĩa... thì có thể dẫn tới hiện tượng xung đột và nó là một trong những nguyên nhân cơ bản làm suy giảm hiệu năng hệ thống, thậm chí có thể làm treo hệ thống.

Các giải thuật loại trừ lẫn nhau giải quyết vấn đề tương tranh, nhưng có thể dẫn đến hiện tượng khóa chết hoặc đói tài nguyên. Thiết kế các hệ thống tương tranh thường là kết quả của việc tìm kiếm các kỹ thuật đáng tin cậy cho việc phối hợp hoạt động thực thi, trao đổi dữ liệu, cấp phát bộ nhớ và lập lịch thực thi để giảm thiểu thời gian đáp ứng và tăng tối đa thông lượng. Ví dụ, trong mạng cục bộ, nếu các máy tính cùng gửi dữ liệu trên một kênh truyền vật lý thì sẽ dẫn đến hiện tượng xung đột, người ta đã hạn chế hiện tượng này bằng cách đấu nối các máy tính với nhau qua thiết bị chuyển mạch.

Dữ liệu trong hệ thống phân tán có thể được lưu trữ tại một hoặc nhiều vị trí khác nhau, các tiến trình truy nhập dữ liệu để thực hiện các công việc tính toán theo yêu cầu nghiệp vụ. Nếu nhiều tiến trình đồng thời truy nhập dữ liệu mà không giải quyết tốt vấn đề tương tranh thì có thể dẫn tới hiện tượng xung đột, hậu quả tất yếu sẽ làm suy giảm hiệu năng hệ thống và thậm chí có thể dẫn đến sai lệch thông tin, hai hiện tượng phổ biến là mất cập nhật và đọc không nhất quán. Điều khiển tương tranh là cơ chế cho phép nhiều tiến trình đồng thời truy nhập đến dữ liệu mà không xảy ra xung đột, nó đảm bảo tính toàn vẹn trong hệ thống.

### 6.1 Không sử dụng giao tác

Tài khoản X muốn chuyển cho tài khoản Y một lượng tiền bằng Z% số dư tài khoản hiện hành của Y, theo nghiệp vụ ngân hàng, trước hết sẽ lấy số dư hiện nay của tài khoản Y nhân với tỉ lệ Z sẽ cho lượng tiền phải chuyển khoản, công việc còn lại chỉ là thao tác cập nhật số dư mới của các tài khoản Y và X. Đoạn chương trình sau thực hiện nhiệm vụ trên:

```
MoneyTransfer(Accounts X, Accounts Y, float Z )  
{  
    float Balance= Y.GetBalance();           //Lấy số dư tài khoản  
    float Trans= Balance*Z;                 //Tính số tiền cần chuyển  
    Y.SetBalance(Balance +Trans);           //Cập nhật số dư tài khoản  
    X.Withdraw(Trans);                     // Trừ tiền tài khoản X  
}
```

Giả sử ba tài khoản A, B và C đều thuộc về một ngân hàng, nghĩa là các tài khoản này đều được lưu trữ trong một cơ sở dữ liệu, số dư tài khoản của chúng có giá trị lần lượt là 100\$, 200\$ và 300\$. Hai tài khoản A và C cùng chuyển cho tài khoản B số tiền bằng 10% số dư tài khoản hiện có của B. Nếu sự kiện chuyển tiền xảy ra tại những thời điểm khác nhau thì giá trị tài khoản của B sẽ là 242\$, không phụ thuộc tài khoản nào thực

hiện chuyển tiền trước. Tuy nhiên, sẽ có xác suất cả hai thao tác chuyển tiền được thực hiện cùng một lúc, đoạn mã mô tả nghiệp vụ chuyển tiền trên sẽ thực hiện như sau:

A chuyển cho B	C chuyển cho B
MoneyTransfer(A,B,0.1)	MoneyTransfer(C,B,0.1)
<pre>float Balance= B.GetBalance(); float Trans= Balance*0.1; B.SetBalance(Balance+Trans); A.Withdraw(Trans);</pre>	<pre>float Balance= B.GetBalance(); float Trans= Balance*0.1; B.SetBalance(Balance+Trans); C.Withdraw(Trans);</pre>

Hai yêu cầu chuyển tiền trên được gửi đến máy chủ cùng một lúc, tại một thời điểm máy chủ chỉ có thể thực hiện một lệnh. Tám câu lệnh của cả hai yêu cầu trên sẽ cho rất nhiều tổ hợp khác nhau và một trong số những tổ hợp đó có thể như sau (phản ứng thích sử dụng ký pháp ngôn ngữ C, nó thể hiện kết quả thực hiện câu lệnh):

<pre>float Balance=B.GetBalance(); //200\$ float Trans= Balance/0.1; //20\$  B.SetBalance(Balance+Trans); //220\$ A.Withdraw(Trans); // 80\$</pre>	<pre>float Balance= B.GetBalance(); //200\$ float Trans= Balance*0.1; // 20\$ B.SetBalance(Trans); //220\$  C.Withdraw(Trans); //280\$</pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

Kết quả thực hiện cho thấy, số dư của tài khoản B là 220\$ trong khi mỗi tài khoản A và C đều bị trừ 20\$, số dư tài khoản B theo đúng nghiệp vụ phải là 242\$ hoặc ít nhất phải là 240\$ vì mỗi tài khoản A và C chuyển cho nó 20\$. Như vậy thủ tục tính toán không những đã thực hiện sai nghiệp vụ mà còn gây thất thoát cho tài khoản A hoặc C, hiện tượng này gọi là mất cập nhật.

Tương tự như vậy, giả sử tài khoản A chuyển cho tài khoản B 10\$, như vậy sẽ phải trừ 10\$ trong số dư của tài khoản A và cộng thêm 10\$ vào số dư của tài khoản B. Tại thời điểm đang xảy ra giao dịch, nhân viên ngân hàng thực hiện thao tác kiểm kê số dư của tất cả các tài khoản, các lệnh của các thao tác như sau:

A chuyển cho B 10\$	Nhân viên kiểm tra số dư
<pre>A.Withdraw(10); B.Deposit(10);</pre>	<pre>float Total=A.GetBalance(); Total= Total+ B.GetBalance();</pre>

Một trong những tổ hợp các lệnh sẽ được thực hiện trên máy chủ như sau:

<pre>A.Withdraw(10); // 90\$  B.Deposit(10); // 210\$</pre>	<pre>float Total=A.GetBalance(); // 90\$ Total= Total+ B.GetBalance(); // 290\$</pre>
---------------------------------------------------------------------	-------------------------------------------------------------------------------------------

Kết quả thực hiện cho thấy tổng số dư của các tài khoản phải là 290\$, đây là kết quả không đúng, nếu ngay sau đó nhân viên ngân hàng kiểm kê lại thì kết quả là 300\$, đây là hiện tượng đọc kết quả không nhất quán. Nguyên nhân của hai lỗi trên được xác định là do trình tự thực hiện các thao tác không theo mong muốn và xung đột đã xảy ra trong khi thực hiện.

Để giải quyết vấn đề này thì phải tuân tự hóa thực hiện các câu lệnh, đưa các câu lệnh vào trong một khối gọi là giao tác, khi đó thứ tự thực hiện các câu lệnh sẽ nhất quán theo ý đồ của người phát triển. Trong ví dụ chuyển tiền giữa các tài khoản, nếu gộp ba câu lệnh đầu tiên vào một giao tác thì thứ tự thực hiện của chúng sẽ như sau:

A chuyển cho B	C chuyển cho B
MoneyTransfer(A,B,0.1)	MoneyTransfer(C,B,0.1)
<pre>float Balance= B.GetBalance(); //200\$ float Trans= Balance/0.1; // 20\$ B.SetBalance(Balance+Trans); //220\$  A.Withdraw(Trans); // 80\$</pre>	<pre>float Balance= B.GetBalance(); //220\$ float Trans= Balance*0.1; // 22\$ B.SetBalance(Trans); //242\$  C.Withdraw(Trans); //280\$</pre>

Trường hợp thực hiện trên bắt buộc ba câu lệnh đầu tiên phải được thực hiện cùng với nhau, không quan trọng giao dịch chuyển tiền nào được thực hiện trước, kết quả số dư tài khoản của B sau khi thực hiện chuyển tiền là 242\$, số lượng tăng thêm đúng bằng tổng số tiền đã trừ trong các tài khoản A và C. Tuy nhiên dễ nhận thấy sơ hở nếu chỉ gộp ba câu lệnh đầu tiên vào giao tác, thực hiện thành công cộng thêm tiền vào tài khoản của B nhưng lại thất bại khi trừ tiền trong tài khoản của A hoặc C. Giải pháp tốt nhất sẽ phải đưa cả bốn câu lệnh vào giao tác, như vậy việc thực hiện chuyển tiền sẽ được hoàn tất lần lượt cho từng yêu cầu, dù cho các yêu cầu đó được gửi đến máy chủ tại cùng một thời điểm.

Hầu hết các hệ quản trị cơ sở dữ liệu hiện nay đều cài đặt cơ chế khóa để điều khiển tương tranh, điều này có thể dẫn tới hiện tượng khóa chết, nó làm hệ thống bị “treo”. Giải quyết tương tranh chỉ là công việc của quá trình phát triển phần mềm, sẽ tốt hơn nếu ngay từ giai đoạn thiết kế đã lường trước vấn đề này sao cho không thể xảy ra tương tranh.

Ví dụ, để đếm số lượng người đọc bài viết trên các trang tin điện tử, người thiết kế bảng Articles(ArticleId, Counter...), trong đó Counter sẽ được tăng thêm 1 khi có người đọc bài viết, cách thiết kế như vậy tiềm ẩn nguy cơ tương tranh. Sẽ tốt hơn nếu tạo thêm bảng ArticleVisits(ArticleVisitId, ArticleId, ...), mỗi khi có người đọc bài viết thì thêm một bản ghi, câu lệnh thêm bản ghi không bao giờ gây nên tương tranh. Tương tự như vậy, đối với bài toán chuyển tiền trong ngân hàng, chỉ cần thiết kế bảng lịch sử biến động số dư tài khoản, mỗi lần thực hiện giao dịch sẽ thêm một bản ghi, nguy cơ tương tranh sẽ giảm đi rất nhiều. Thao tác thực hiện không những nhanh hơn mà còn quản lý được nhiều thông tin hơn, ví dụ có thể quản lý tiền đã được chuyển từ tài khoản và thời gian cũng như địa điểm nào.

## 6.2 Khái niệm giao tác

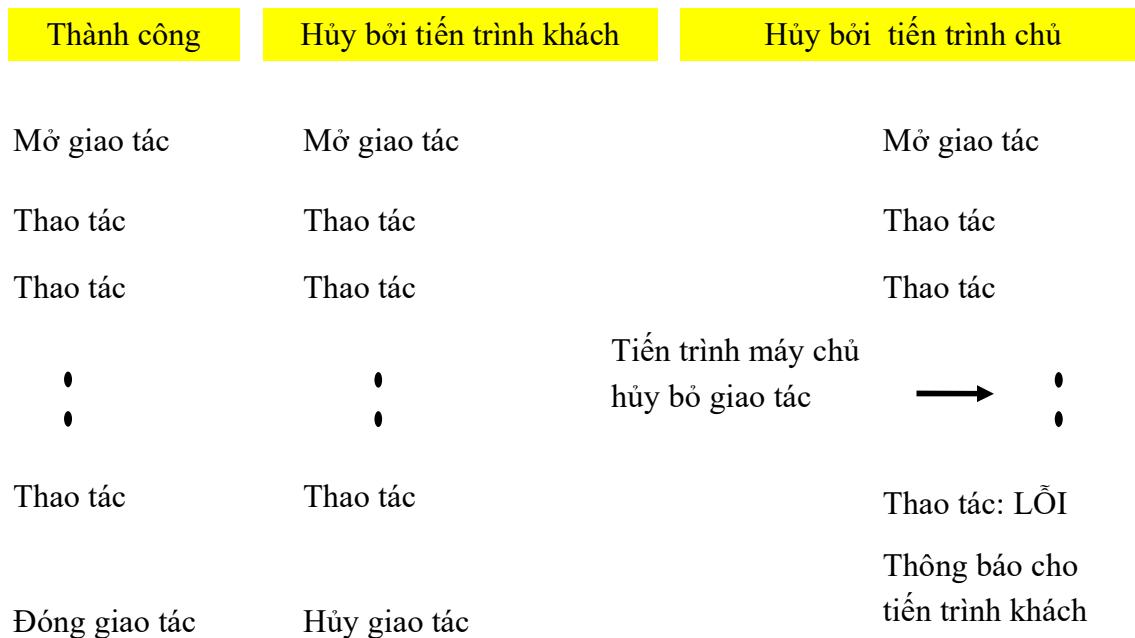
Giao tác gồm một hoặc nhiều câu lệnh truy nhập dữ liệu và chúng được thực hiện như một đơn vị thống nhất, nghĩa là không thể thêm hay bớt bất kỳ câu lệnh nào. Giao tác được chia thành giao tác phẳng, giao tác lồng ghép và giao tác phân tán. Giao tác có thể

gồm nhiều câu lệnh đọc/ghi dữ liệu và phải đảm bảo bốn tính chất: nguyên tử, nhất quán, cô lập và bền vững.

Tính chất nguyên tử nghĩa là đối với thế giới bên ngoài thì giao tác không thể chia nhỏ hơn được, các lệnh trong giao tác đều được thực hiện hoặc không có lệnh nào được thực hiện, nếu giao tác bị hủy giữa chừng thì kết quả của các lệnh đã thực hiện sẽ bị hủy bỏ. Tính chất nhất quán nghĩa là luôn phải đảm bảo tính toàn vẹn dữ liệu, giao tác không vi phạm tính chất bất biến của hệ thống. Tính chất cô lập nghĩa là mỗi giao tác sẽ hoạt động độc lập và không làm ảnh hưởng đến các giao tác khác. Tính chất bền vững nghĩa là khi giao tác đã cam kết thì các thay đổi đối với nó không phải là tạm thời mà là những thay đổi bền vững, nếu không cam kết thì sẽ tự động phục hồi về giá trị cũ, coi như chưa từng thực hiện các lệnh trong giao tác.

### 6.2.1 Giao tác phẳng

Giao tác thỏa mãn bốn tính trên gọi là giao tác phẳng, hạn chế chính của giao tác phẳng là chúng không cho phép tách riêng các kết quả được cam kết hay hủy bỏ. Mỗi giao tác bắt đầu bằng lệnh mở giao tác, tiếp theo là các chuỗi các lệnh thao tác với dữ liệu và kết thúc đóng giao tác bằng lệnh cam kết, nghĩa là đồng ý với những lệnh đã thực hiện hoặc hủy bỏ nghĩa là không đồng ý với những lệnh đã thực hiện và đề nghị phục hồi lại các giá trị trước khi bắt đầu giao tác.



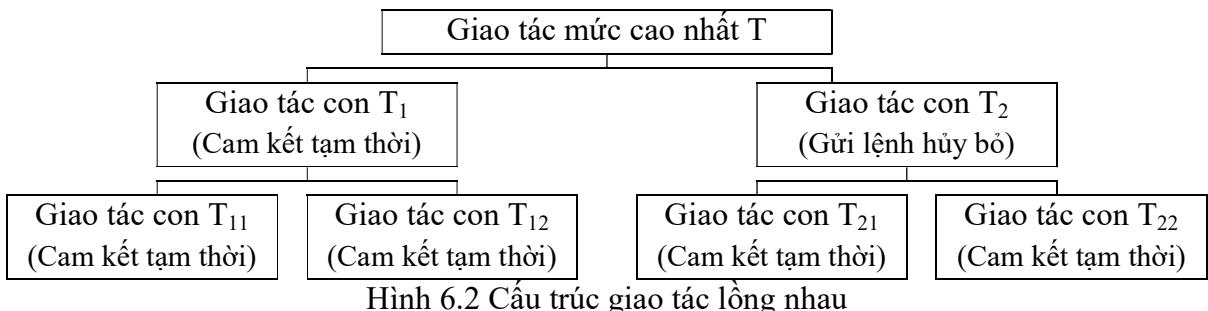
Hình 6.1 Các trường hợp thực hiện trong giao tác phẳng

Hình 6.1 thể hiện các trường hợp thực hiện giao tác phẳng, cả ba tình huống đều bắt đầu bằng lệnh mở giao tác. Trường hợp thứ nhất thực hiện thành công, sau lệnh mở giao tác thì từng lệnh được thực hiện và không gặp bất kỳ lỗi nào, lệnh cuối cùng là đóng giao tác, thông thường đó là lệnh cam kết (COMMIT), những thay đổi chính thức được ghi vào bộ nhớ ổn định. Trường hợp thứ hai, khi đang thực hiện một câu lệnh nào đó thì gặp

câu lệnh hủy bỏ (ABORT hoặc ROLLBACK), giao tác lập tức sẽ bị đóng lại, dữ liệu sẽ được phục hồi về giá trị trước khi bắt đầu giao tác. Trường hợp thứ ba, khi đang thực hiện thì gặp lỗi và giao tác không thể thực hiện lệnh đó, ngay lập tức tiến trình máy chủ sẽ hủy bỏ giao tác và dữ liệu được phục hồi về giá trị cũ trước khi bước vào giao tác. Trong trường hợp này, một số hệ thống sẽ tự động thực hiện câu lệnh ABORT hoặc ROLLBACK nhưng có một số hệ thống không tự động thực hiện những câu lệnh này, điều đó có thể dẫn tới hiện tượng “treo” hệ thống.

### 6.2.2 Các giao tác lồng nhau

Giao tác phẳng thường được sử dụng nhiều nhất, trong thực tế còn có những giao tác được cấu thành từ một số giao tác con gọi là giao tác lồng nhau, nói cách khác là trong giao tác lại gồm nhiều giao tác con, mỗi giao tác con cũng có thể thực thi một hay nhiều giao tác con của chính nó. Giao tác lồng nhau dùng để tách một giao dịch lớn thành nhiều phần và mỗi phần hoạt động độc lập. Độ sâu của giao tác lồng nhau không giới hạn, nó chỉ phụ thuộc vào dung lượng bộ nhớ cho phép trên máy tính.



Hình 6.2 Cấu trúc giao tác lồng nhau

Hình 6.2 thể hiện giao tác lồng nhau, giao tác lớn nhất T có hai giao tác con T<sub>1</sub> và T<sub>2</sub>, mỗi giao tác con lại chứa đựng các con của chúng. Các giao tác con thể hiện tính nguyên tử đối với cha của nó, các giao tác con trên cùng một mức có thể chạy đồng thời nhưng nếu chúng truy nhập cùng một tài nguyên thì sẽ được tuân tự hóa. Trong khi giao tác con đang hoạt động thì giao tác cha không được thực hiện bất kỳ thao tác nào ngoài việc cam kết hay hủy bỏ hoặc tạo thêm giao tác con, nó chỉ đóng giao tác khi tất cả các con của nó cũng đã hoàn thành cam kết hoặc hủy bỏ. Nếu giao tác cha thực hiện lệnh hủy bỏ thì các con của nó sẽ bị hủy bỏ, nhưng nếu giao tác cha thực hiện lệnh cam kết thì chưa chắc tất cả các giao tác con cũng thực hiện cam kết. Giao tác con thực hiện lệnh hủy bỏ hay cam kết đều không ảnh hưởng đến giao tác cha của nó.

### 6.2.3 Giao tác phân tán

Giao tác phân tán là những giao tác thực hiện thao tác với dữ liệu được đặt trên nhiều máy tính khác nhau, nó phải đảm bảo tính thống nhất trong việc thực hiện các thao tác trên tất cả các máy tính. Về bản chất, trên mỗi máy tính sẽ thực hiện các giao tác phẳng, nó bắt đầu bằng lệnh mở giao tác và kết thúc bằng lệnh cam kết hoặc hủy bỏ, như vậy giữa các máy tính phải có sự đồng thuận về cam kết hay hủy bỏ, vấn đề này sẽ được bàn luận chi tiết trong phần sau.

### 6.3 Các phương pháp điều khiển tương tranh

Các giải thuật loại trừ tương hỗ đều dựa trên nguyên lý tuần tự hóa, nếu một tiến trình đang sử dụng dữ liệu mà tiến trình khác muốn sử dụng thì phải chờ. Truy nhập cơ sở dữ liệu có những đặc điểm riêng so với các tài nguyên vật lý khác, thao tác với dữ liệu chỉ có câu lệnh đọc hoặc ghi, câu lệnh INSERT trong các hệ quản trị cơ sở dữ liệu không tạo nên xung đột trong khi các câu lệnh UPDATE và DELETE tiềm ẩn nguy cơ xung đột.

**Bảng 6.1** Luật xung đột đọc/ghi

Giao tác 1	Giao tác 2	Xung đột	Nguyên nhân
Đọc	Đọc	Không	Kết quả của các cặp thao tác đọc không phụ thuộc vào thứ tự thực hiện của chúng
Đọc	Ghi	Có	Kết quả của thao tác đọc và ghi phụ thuộc vào thứ tự thực hiện của chúng
Ghi	Ghi	Có	Kết quả của các cặp thao tác ghi phụ thuộc vào thứ tự thực hiện của chúng

Luật xung đột đọc/ghi liệt kê trong bảng 6.1 cho thấy, nếu chỉ đọc dữ liệu thì sẽ không bao giờ xảy ra xung đột, như vậy sẽ không hạn chế số lượng tiến trình cùng đọc một thực thể, xung đột chỉ xảy ra khi xuất hiện lệnh ghi. Hai giao tác được gọi là tương tranh nếu kết quả tổ hợp thực hiện các lệnh của chúng phụ thuộc vào thứ tự thực hiện, do đó cần phải xác định các lệnh có thể gây ra xung đột và phải tuần tự hóa thứ tự thực hiện của chúng. Giả sử hai giao tác T và U thực hiện các câu lệnh sau:

Giao tác T	Giao tác U
<pre>x = read(i); write(i, 10); write(j, 20);</pre>	<pre>y = read(j); write(j, 30); z = read(i);</pre>

Khi thực hiện, hai câu lệnh đầu tiên đều có thể thực hiện hợp lệ, chúng chỉ đọc hoặc ghi dữ liệu vào những thực thể khác nhau. Xung đột có thể xảy ra khi thực hiện câu lệnh thứ ba, giao tác T muốn ghi dữ liệu vào thực thể j trong khi giao tác U đã ghi dữ liệu vào thực thể này nhưng chưa hoàn thành giao tác, giao tác U muốn đọc dữ liệu từ thực thể i nhưng nó lại được giao tác T đã cập nhật nhưng cũng chưa hoàn thành giao tác. Điều này dẫn tới hiện tượng cả hai giao tác T và U đều phải chờ nhau và sẽ không có giao tác nào được hoàn thành, như vậy chỉ có luật xung đột đọc/ghi chưa đủ, để giải quyết vấn đề tương tranh cần phải bổ sung thêm qui tắc về tương đương tuần tự.

Hai giao tác được gọi là tương đương tuần tự theo thứ tự khi và chỉ khi tất cả các cặp thao tác xung đột của chúng được thực hiện theo thứ tự trên tất cả các đối tượng chúng truy nhập. Trong ví dụ trên, tương đương tuần tự theo thứ tự nếu thỏa mãn nếu giao tác T truy nhập các thực thể i và j trước giao tác U hoặc giao tác U truy nhập các thực thể i và j trước giao tác T. Không phân biệt giao tác nào thực hiện trước, tại một thời điểm chỉ được phép thực hiện một giao tác, nếu tuân thủ luật này thì các câu lệnh trong hai giao tác trên sẽ như sau:

Giao tác T	Giao tác U
<pre>x = read(i); write(i, 10); write(j, 20);</pre>	<pre>y = read(j); write(j, 30); z = read (i);</pre>

Có ba cách tiếp cận giải quyết vấn đề tương tranh: Điều khiển tương bằng khóa, điều khiển tương tranh lạc quan và điều khiển tương tranh dựa trên nhãn thời gian. Hai cách tiếp cận đầu tiên trái ngược nhau về quan điểm, điều khiển tương tranh bằng khóa nhìn nhận hệ thống luôn tiềm ẩn tương tranh do đó còn có tên gọi khác là điều khiển tương tranh bi quan, trong khi đó điều khiển tương tranh lạc quan nhìn nhận không xảy ra tương tranh và khi nào xảy ra thì mới giải quyết.

### 6.3.1 Điều khiển tương tranh bằng khóa

Để tránh xung đột khi truy nhập dữ liệu dùng chung thì cách đơn giản nhất là tuần tự hóa theo kiểu loại trừ tương hỗ, nghĩa là khi một giao tác đang thao tác với dữ liệu thì các giao tác khác phải chờ. Tư tưởng chính của giải pháp này là các thao tác trên dữ liệu nếu có xung đột thì tại một thời điểm chỉ cho phép một giao tác thực hiện, điều này có thể thực hiện dựa trên cơ chế khóa, một giao tác được phép truy nhập dữ liệu khi và chỉ khi nó đang giữ khóa trên dữ liệu đó.

Trước khi thao tác trên một mục dữ liệu thì phải thiết lập khóa và khi thực hiện xong thì phải giải phóng khóa. Nếu mục dữ liệu đã bị khóa bởi một giao tác khác và khóa đó không tương thích thì bộ điều khiển tương tranh sẽ không cấp khóa cho đến khi khóa này được giải phóng. Giao tác phải giữ một khóa trên mục dữ liệu chừng nào mà nó còn truy nhập mục dữ liệu này, việc nhả khóa ngay sau khi truy nhập cuối cùng đến mục dữ liệu không phải là điều mong muốn vì như vậy tính tuần tự có thể không được đảm bảo cho đến khi tất cả các khóa không tương thích do các giao tác khác giải phóng.

Cơ chế khóa bao gồm hai pha, pha thứ nhất giữ khóa và pha thứ hai nhả khóa. Nếu khóa được nhả ngay sau khi hoàn thành truy nhập dữ liệu thì gọi là khóa hai pha thường, nếu khóa chỉ được nhả khi kết thúc gọi là khóa hai pha nghiêm ngặt. Khóa ghi được hình thành từ lúc thực hiện thao tác ghi cho đến khi gấp chỉ thị COMMIT hoặc ROLLBACK/ABORT, như vậy đó là khóa hai pha nghiêm ngặt. Khóa đọc được hình thành từ khi thực hiện thao tác đọc và giải phóng ngay sau khi đọc xong, như vậy đó là khóa hai pha thường, muốn thay đổi mức độ tuần tự thì phải qui định một cách tường minh khi bắt đầu giao tác. Khi một thao tác truy nhập đổi tượng bên trong giao tác, cơ chế quản lý khóa như sau:

- Nếu đổi tượng chưa bị khóa, đổi tượng sẽ được khóa và thao tác tiếp tục.
- Nếu đổi tượng được khóa trong cùng một giao tác thì nâng mức khóa nếu cần thiết và thao tác tiếp tục.
- Nếu đã có giao tác khác khóa đổi tượng và khóa đó không xung đột với thao tác thì khóa ở chế độ chia sẻ và thao tác tiếp tục.

- Nếu đã có giao tác khác khóa đối tượng và khóa đó xung đột với thao tác thì thao tác đó phải chờ cho đến khi mở khóa

Khi thực hiện COMMIT hoặc ABORT, máy chủ sẽ mở khóa tất cả các đối tượng đã khóa cho giao tác. Trở lại với ví dụ về giao dịch chuyển tiền đã trình bày ở đầu chương, nếu các câu lệnh được đặt trong giao tác với mức cách ly dạng tuần tự, các khóa sẽ được thiết lập như sau:

A chuyển cho B	C chuyển cho B
MoneyTransfer(A,B,0.1)	MoneyTransfer(C,B,0.1)
<pre>Mở giao tác mức tuần tự float Balance= B.GetBalance(); //R<sub>b</sub> float Trans= Balance/0.1; B.SetBalance(Balance+Trans); //W<sub>b</sub> A.Withdraw(Trans); //W<sub>a</sub> Đóng giao tác // Giải phóng W<sub>a</sub>, W<sub>b</sub></pre>	<pre>Mở giao tác mức tuần tự float Balance= B.GetBalance(); // R<sub>b</sub> float Trans= Balance*0.1; B.SetBalance(Trans); // W<sub>b</sub> C.Withdraw(Trans); // W<sub>c</sub> Đóng giao tác // Giải phóng W<sub>c</sub>, W<sub>b</sub></pre>

Vì giao tác được mở ở mức tuần tự, nghĩa là tất cả các giao tác truy nhập vào các mục dữ liệu chung đều phải được tuần tự hóa, như vậy mới đảm bảo tương đương tuần tự, do đó giao tác chuyển tiền của tài khoản A hoặc C thực hiện xong thì giao tác còn lại mới bắt đầu. Trong giao tác tài khoản A chuyển tiền cho tài khoản B, bản ghi tài khoản B sẽ được thiết lập khóa đọc  $R_b$  trước khi đọc số dư tài khoản của nó. Trước khi cập nhật giá trị mới cho số dư của tài khoản B, bản ghi này đang bị khóa ở mức đọc, do cùng nằm trong một giao tác nên nó nâng cấp khóa lên mức ghi  $W_b$ , vì đây là khóa nghiêm ngặt nên phải chờ đến khi kết thúc giao tác mới được giải phóng. Lệnh tiếp theo thực hiện trừ tiền trong số dư của tài khoản A, tất nhiên khóa ghi  $W_a$  sẽ được thiết lập trước khi thực hiện thao tác này. Cuối cùng khi đóng giao tác, những thay đổi sẽ được đưa vào vùng nhớ ổn định và các khóa  $W_a$ ,  $W_b$  cũng sẽ được giải phóng, khi đó giao tác tài khoản C chuyển tiền cho tài khoản B mới bắt đầu.

Sử dụng khóa có thể dẫn đến hiện tượng khóa chết, đó là trạng thái mỗi thành viên của nhóm các giao tác phải chờ lẫn nhau giải phóng khóa. Giả sử tài khoản A chuyển 50\$ cho tài khoản B và cùng thời điểm đó tài khoản B chuyển 100\$ cho tài khoản A, hai giao tác chuyển tiền được máy chủ thực hiện như sau:

A chuyển 50\$ cho B	B chuyển 100\$ cho A
MoneyTransfer(A,B,0.1)	MoneyTransfer(B,A,0.1)
<pre>Mở giao tác A.Withdraw(50); //W<sub>a</sub> B.Deposit(50); //Chờ khóa W<sub>b</sub> Đóng giao tác</pre>	<pre>Mở giao tác B.Withdraw(100); // W<sub>b</sub> A.Deposit(100); // Chờ khóa W<sub>a</sub> Đóng giao tác</pre>

Theo thứ tự thực hiện trên, khóa  $W_a$  được thiết lập trước khi thực hiện lệnh thứ trù tiền của tài khoản A, khi chưa thực hiện lệnh cộng thêm tiền vào tài khoản B thì giao tác thứ hai thực hiện lệnh trù tiền của tài khoản B, do đó khóa  $W_b$  được thiết lập. Đây là những khóa nghiêm ngặt, do đó chúng chỉ được giải phóng khi kết thúc giao tác, giao tác chưa kết thúc cho nên hai khóa ghi này vẫn tồn tại. Chuyển sang lệnh cộng thêm tiền vào số dư tài khoản, giao tác về trái muốn thêm 50\$ vào tài khoản của B, nó phải chờ vì khóa  $W_b$  đã được giao tác về phải thiết lập, tương tự như vậy giao tác về phải muốn thêm tiền vào tài khoản của A cũng phải chờ vì khóa  $W_a$  đã được giao tác về trái thiết lập. Như vậy giao tác về trái và giao tác về phải chờ nhau và không một giao tác nào có thể kết thúc, hiện tượng này gọi là khóa chết, biểu hiện của nó hệ thống dường như bị “treo”, nghĩa là vẫn có thể thao tác với các bản ghi khác ngoại trừ hai bản ghi này.

Có thể sử dụng đồ thị “Chờ đợi” để thể hiện các giao tác này chờ nhau, đó là vòng tròn chờ đợi vô hạn, chỉ cần hủy bỏ một giao tác thì các giao tác khác có thể tiếp tục thực hiện và như vậy sẽ kết thúc hiện tượng khóa chết. Có ba giải pháp hạn chế hiện tượng khóa chết, giải pháp thứ nhất, khi bắt đầu giao tác thì thiết lập khóa tất cả các mục dữ liệu cần truy nhập, cách làm này dẫn đến suy giảm hiệu năng, nó ngăn cản việc truy nhập đến các mục dữ liệu có thể chia sẻ. Giải pháp thứ hai, mỗi giao tác yêu cầu thiết lập khóa trên các mục dữ liệu theo thứ tự đã định nghĩa trước, giải pháp này sinh vấn đề lãng phí, một giao tác giữ khóa trên mục dữ liệu nhưng lại chưa bắt đầu truy nhập nó. Giải pháp thứ ba, mỗi khóa có khoảng thời gian giới hạn, sau thời gian đó sẽ không được bảo vệ, nếu xảy ra hiện tượng khóa chết thì chấp nhận giao tác có thể bị hủy bỏ.

Mặc dù giải pháp thứ ba hủy bỏ giao tác nào đó nếu xảy ra hiện tượng khóa chết nhưng nó lại đáp ứng các yêu cầu về hiệu năng, một yêu cầu quan trọng hàng đầu trong bất kỳ hệ thống thông tin nào, do đó nó là giải pháp được lựa chọn để cài đặt trong các hệ quản trị cơ sở dữ liệu. Trong hệ thống tồn tại một tiến trình giám sát, nó luôn theo dõi thời gian hoạt động của các giao tác, nếu phát hiện thấy khóa chết thì sẽ hủy một giao tác nào đó mà thời gian bảo vệ đã quá hạn, giá trị của các mục dữ liệu sẽ được phục hồi và do đó vẫn bảo toàn tính nhất quán. Trường hợp hai tài khoản A và B chuyển tiền cho nhau, giả sử tiến trình giám sát hủy bỏ giao tác A chuyển tiền cho B, các câu lệnh sẽ thực hiện như sau:

A chuyển 50\$ cho B	B chuyển 100\$ cho A
Mở giao tác A.Withdraw(50); // $W_a$ B.Deposit(50); // Chờ khóa $W_b$ Hủy giao tác // Giải phóng khóa $W_a$	Mở giao tác B.Withdraw(100); // $W_b$ A.Deposit(100); Đóng giao tác

Giao tác về trái đã bị hủy, nghĩa là số dư của tài khoản A vẫn là 100\$, câu lệnh B.Deposit(50) vẫn chưa được phép thực hiện cho nên không cần phải phục hồi giá trị cũ của nó. Hiện tượng khóa chết đã được giải quyết, giao tác về phải tiếp tục thực hiện, số

dư của tài khoản A sẽ là 200\$ và của B sẽ chỉ còn 100\$. Mặc dù hiện tượng khóa chết đã được giải quyết, nhưng phải hủy một giao tác không phải là cách làm hay, nếu thay đổi thiết kế bằng cách thêm bảng lịch sử giao dịch thì sẽ không bao giờ xảy ra hiện tượng tương tranh.

### 6.3.2 Điều khiển tương tranh lạc quan

Điều khiển tương tranh bằng khóa có một số nhược điểm cơ bản, một số hệ thống không hỗ trợ truy nhập đồng thời đến các dữ liệu chia sẻ, ví dụ hệ thống tập tin nhật ký<sup>1</sup>. Những thao tác đọc không hề ảnh hưởng đến tính toàn vẹn của dữ liệu nhưng vẫn phải duy trì khóa để đảm bảo trong thời gian đang đọc thì giao tác khác không được phép ghi. Điều khiển tương tranh bằng khóa tiềm ẩn nguy cơ khóa chết, để phòng hiện tượng này làm giảm nghiêm trọng tính tương tranh, do đó phải sử dụng thời gian quá hạn và phát hiện khóa chết, cách làm này không phù hợp với các chương trình tương tác. Khóa hai pha nghiêm ngặt chỉ được giải phóng khi kết thúc giao tác, điều đó làm suy giảm đáng kể tính tương tranh.

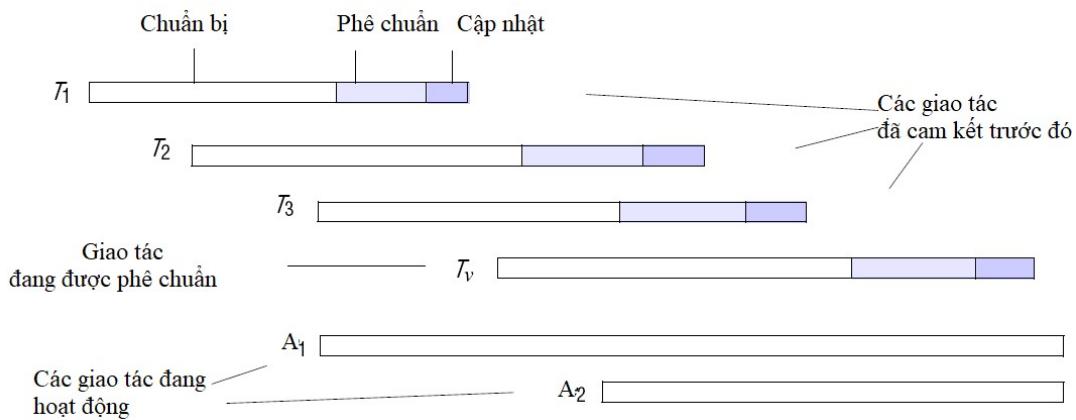
Điều khiển tương tranh bằng khóa được coi là cách nhìn bi quan, Kung và Robinson đề xuất cách nhìn lạc quan vì cho rằng xác suất xung đột rất thấp. Điều khiển tương tranh lạc quan cho phép các giao tác được phép tiếp tục nếu không có xung đột với các giao tác khác, nếu phát hiện xung đột thì sẽ hủy bỏ giao tác nào đó và máy khách sẽ phải thực hiện lại. Mỗi giao tác được chia thành ba giai đoạn, bắt đầu bằng pha chuẩn bị sau đó đến phê chuẩn và cuối cùng là pha cập nhật, sự phân chia này dựa trên giả định pha chuẩn bị thường chiếm thời gian nhiều nhất.

Trong pha chuẩn bị, mỗi giao tác sở hữu phiên bản tạm thời của tất cả các đối tượng mà nó phải cập nhật, thực chất đó là bản sao của phiên bản được cam kết gần nhất. Các thao tác kiểm tra ràng buộc cũng được thực hiện trong pha này, nếu vi phạm ràng buộc thì giao tác sẽ bị hủy bỏ ngay lập tức. Việc sử dụng các phiên bnr tạm thời sẽ tạo điều kiện để dễ dàng hủy bỏ giao tác nếu gặp lỗi hoặc không được phê chuẩn mà không ảnh hưởng đến các đối tượng, nếu được phê chuẩn thì đó là giá trị mới của đối tượng. Các thao tác đọc sẽ được thực hiện ngay lập tức, nếu có phiên bản tạm thời thì sử dụng nó, nếu không sẽ đọc giá trị của phiên bản đã được cam kết gần nhất. Nếu có nhiều giao tác tương tranh thì có thể tồn tại một số giá trị tạm thời khác nhau cho cùng một đối tượng, một giao tác không thể nhìn thấy phiên bản tạm thời của giao tác khác.

Trong pha phê chuẩn, các thao tác trên đối tượng dùng chung sẽ được kiểm tra xem có vi phạm luật xung đột hay không, nếu phê chuẩn thành công thì giao tác có thể kết cam kết. Nếu phê chuẩn thất bại thì phải áp dụng một số giải pháp, có thể hủy giao tác hiện hành, nhưng cũng có thể hủy các giao tác khác. Trong pha cập nhật, nếu giao tác đã được phê chuẩn thì tất cả những thay đổi đã xảy ra trên bản lưu tạm thời sẽ chuyển sang vĩnh cửu.

---

<sup>1</sup> Hệ thống tập tin có khả năng phục hồi riêng trong trường hợp xảy ra lỗi, thông tin về các thay đổi được ghi lại trong nhật ký trước khi cập nhật các chỉ mục của tập tin. Nếu mất điện hoặc lỗi hệ thống khác làm hỏng các chỉ mục khi đang ghi tập tin, hệ điều hành có thể sử dụng nhật ký để sửa chữa tập tin khi khởi động lại máy tính.



Hình 6.3 Phê chuẩn giao tác

Phê chuẩn giao tác sử dụng các luật xung đột đọc/ghi để đảm bảo tương đương tuần tự theo thứ tự, mỗi giao tác được gán số hiệu khi bước vào pha phê chuẩn, đó là số nguyên tăng tuần tự. Nếu giao tác được phê chuẩn và kết thúc thành công thì nó giữ lại số này, nếu phê chuẩn thất bại hay giao tác bị hủy hoặc giao tác thuộc loại chỉ đọc thì sẽ nhả lại số hiệu để gán lại. Số hiệu giao tác thể hiện vị trí của nó theo thời gian, giao tác luôn luôn kết thúc pha chuẩn bị của mình sau tất cả các giao tác có số hiệu thấp hơn.

Hình 6.3 minh họa cách phê chuẩn giao tác, các giao tác  $\{T_1, T_2, T_3\}$  đã hoàn thành cam kết trước khi giao tác  $T_v$  bước vào pha phê chuẩn, các giao tác  $\{A_1, A_2\}$  chưa bước vào pha phê chuẩn. Kiểm tra phê chuẩn trên giao tác  $T_v$  dựa trên xung đột giữa các thao tác trên những cặp giao tác  $T_i$  và  $T_v$ , bảng 6.2 liệt kê các luật cần phải tuân thủ để đạt được tương đương tuần tự theo thứ tự.

**Bảng 6.2** Luật tuần tự trong pha phê chuẩn

Số thứ tự	$T_v$	$T_i$	Luật
1.	Ghi	Đọc	$T_i$ không được đọc các đối tượng $T_v$ ghi
2.	Đọc	Ghi	$T_v$ không được đọc các đối tượng $T_i$ ghi
3.	Ghi	Ghi	$T_i$ không được ghi các đối tượng $T_v$ ghi và $T_v$ không được ghi các đối tượng $T_i$ ghi

Thời gian pha phê chuẩn và cập nhật nói chung nhỏ so với thời gian pha chuẩn bị, nếu qui định tại một thời điểm chỉ cho phép không quá một giao tác trong pha phê chuẩn và pha cập nhật thì luật thứ 3 thỏa mãn. Để tránh chồng chéo giao tác, toàn bộ các pha phê chuẩn và cập nhật được đưa vào vùng tối hạn, để tăng tính tương tranh thì một phần của hai pha này được cài đặt ở bên ngoài vùng tối hạn. Việc phê chuẩn giao tác phải đảm bảo tuân thủ luật 1 và 2 bằng các kiểm tra sự chồng nhau giữa các đối tượng của các cặp giao tác  $T_i$  và  $T_v$ , hai giải thuật phê chuẩn ngược và phê chuẩn xuôi thực hiện nhiệm vụ này. Phê chuẩn ngược kiểm tra với các giao tác đã bước vào giai đoạn phê chuẩn trước nó trong khi đó phê chuẩn xuôi lại kiểm tra với các giao tác sau nhưng vẫn đang hoạt động.

### 6.3.2.1 Phê chuẩn ngược

Lệnh ghi trong giao tác đang chờ phê chuẩn không thể ảnh hưởng đến các lệnh đọc của các giao tác đã bước vào giai đoạn phê chuẩn trước nó, do đó luật 1 được thỏa mãn, như vậy chỉ cần kiểm tra luật 2. Phê chuẩn giao tác  $T_v$  chỉ cần kiểm tra tập đọc của giao tác  $T_v$  với tập ghi của các giao tác trước, nếu phát hiện thấy chòng chéo thì phê chuẩn thất bại. Gọi  $startTn$  là số hiệu giao tác lớn nhất đã được gán trước khi giao tác  $T_v$  bước vào pha chuẩn bị,  $finishTn$  là số hiệu giao tác lớn nhất đã được gán tại thời điểm  $T_v$  bước vào pha phê chuẩn, đoạn chương trình sau thể hiện giải thuật phê chuẩn ngược:

```
boolean valid = true;
for (int Ti=startTn+1; Ti>=finishTn; Ti++)
{
    if (Tập đọc của  $T_v$  giao với tập ghi của  $T_i$ )
    {
        valid = false;
        break;
    }
}
```

Giao tác  $T_1$  kết thúc trước khi giao tác  $T_v$  bắt đầu vào pha chuẩn bị, các giao tác  $T_2$  và  $T_3$  cam kết trước khi giao tác  $T_v$  bước vào pha phê chuẩn, do đó  $startTn+1$  là giao tác  $T_2$  và  $finishTn$  là giao tác  $T_3$ , nghĩa là tập đọc của giao tác  $T_v$  phải so sánh với tập ghi của các giao tác  $T_2$  và  $T_3$ .

Phê chuẩn ngược chỉ kiểm tra tập đọc của giao tác đang chờ phê chuẩn với tập ghi của các giao tác đã được cam kết, nếu phát hiện thấy xung đột thì chỉ có cách hủy giao tác đang chờ phê chuẩn. Nếu giao tác đang chờ phê chuẩn chỉ có những lệnh ghi thì không cần phải thực hiện phê chuẩn ngược. Phê chuẩn ngược đòi hỏi phải lưu lại tập ghi của các giao tác đã được cam kết cho đến khi không còn nguy cơ xảy ra xung đột, giao tác  $A_1$  vẫn đang trong pha chuẩn bị, do đó vẫn phải lưu tập ghi của các giao tác  $\{T_1, T_2, T_3, T_v\}$ .

### 6.3.2.2 Phê chuẩn xuôi

Phê chuẩn xuôi kiểm tra giao tác đang phê chuẩn với các giao tác đang hoạt động bước vào pha phê chuẩn sau nó, luật 2 đương nhiên thỏa mãn vì các giao tác được gán số hiệu sau chỉ được thực hiện lệnh ghi sau khi giao tác trước đã kết thúc thao tác đọc. Tập ghi của giao tác đang phê chuẩn được đối chiếu với tập đọc của tất cả các giao tác vẫn đang hoạt động, những giao tác này vẫn đang ở pha chuẩn bị, đoạn mã chương trình sau thể hiện cách cài đặt giải thuật này:

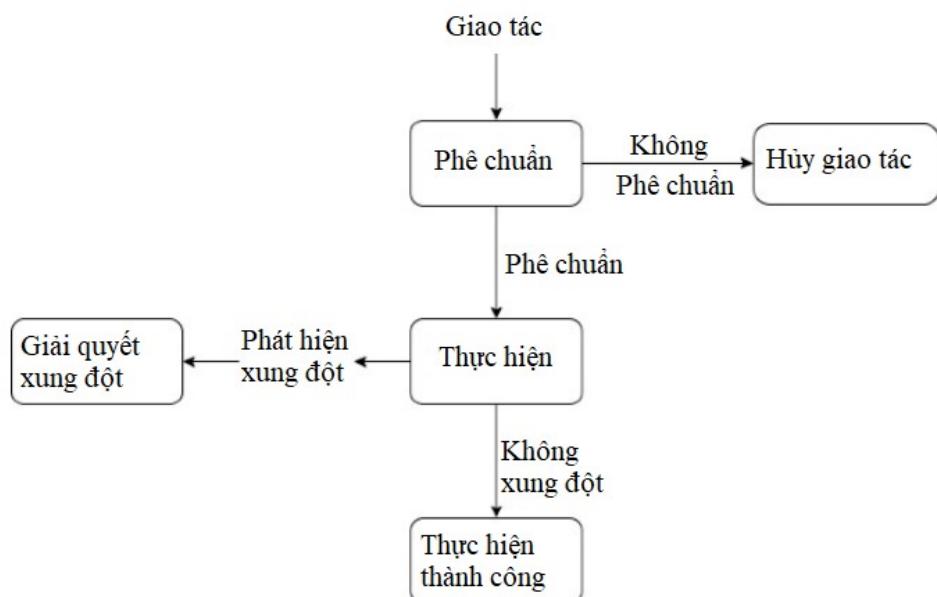
```
boolean valid = true;
for (int Tid = A1; Tid <= AN; Tid++)
{
    if (Tập ghi của  $T_v$  giao với tập đọc của  $T_{id}$ )
    {
        valid = false;
        break;
    }
}
```

Trên hình 6.3, phải đổi chiểu tập ghi của giao tác  $T_v$  với tập đọc của các giao tác  $\{A_1, A_2\}$ , ngay cả khi đang phê duyệt thì tập này cũng có thể bổ sung giao tác mới, nếu giao tác  $T_v$  chỉ chứa các lệnh đọc thì không cần phê chuẩn xuôi. Nếu giao tác đang phê chuẩn xung đột với giao tác khác thì đơn giản nhất là hủy bỏ, tuy nhiên không có gì đảm bảo phê chuẩn thành công cho giao tác của các yêu cầu gửi lại. Giải pháp khác là hủy bỏ tất cả các giao tác xung đột, có thể phải hủy nhiều giao tác, thật khó xác định lý do biện minh cho việc làm này nếu không chứng minh được giao tác hiện đang phê chuẩn có tầm quan trọng cao hơn. Giải pháp thứ ba là chờ một thời gian và kiểm ra lại, không có gì đảm bảo lần phê chuẩn tiếp sẽ thành công, hơn nữa có thể những giao tác mới lại xuất hiện.

### 6.3.3 Điều khiển tương tranh dựa trên nhãn thời gian

Một cách tiếp cận khác trong điều khiển tương tranh là gán nhãn thời gian cho các giao tác, mỗi giao tác khi bắt đầu sẽ được gán nhãn thời gian duy nhất. Việc sắp xếp thứ tự các giao tác theo nhãn thời gian là biện pháp tuần tự hóa, giao tác có nhãn thời gian nhỏ hơn sẽ được ưu tiên thực hiện, đó là biện pháp để đảm bảo tính nhất quán. Luật thứ tự nhãn thời gian dựa trên các thao tác có thể xảy ra xung đột, nó coi mỗi đối tượng chỉ có một phiên bản và hạn chế nhiều giao tác đồng thời truy nhập đến đối tượng. Yêu cầu ghi của một giao tác chỉ hợp lệ nếu giao tác trước đã hoàn thành lệnh đọc hoặc ghi cuối cùng, yêu cầu đọc của một giao tác chỉ hợp nếu giao tác trước đã hoàn thành lệnh ghi cuối cùng.

Nếu mỗi giao tác tạo phiên bản tạm thời cho đối tượng thì nhiều giao tác tương tranh có thể truy nhập đến đối tượng, luật thứ tự nhãn thời gian tinh chỉnh để đảm bảo mỗi giao tác truy nhập đến các phiên bản nhất quán của đối tượng, các phiên bản tạm thời của mỗi đối tượng sẽ được cam kết theo thứ tự nhãn thời gian. Có thể thực hiện ngay các lệnh ghi mà không để máy khách phải chờ, nhưng nếu là lệnh đọc thì máy khách phải chờ để các giao tác trước hoàn thành, giải pháp này giúp tránh được hiện tượng khóa chết.



Hình 6.4 Lưu đồ xử lý giao tác dựa trên nhãn thời gian

Hình 6.4 thể hiện lưu đồ các bước của giải thuật, quá trình xử lý một giao tác được chia thành hai giai đoạn, giai đoạn phê chuẩn và giai đoạn thực hiện. Giai đoạn phê chuẩn kiểm tra nhãn thời gian của từng giao dịch để đảm bảo các giao tác được thực hiện theo đúng trình tự, giao tác có nhãn thời gian nhỏ hơn phải được thực hiện trước.

Mỗi lệnh mang nhãn thời gian của giao tác đã được cấp phát và được phê chuẩn khi thực hiện, nếu câu lệnh không được thực hiện thì giao tác sẽ bị hủy bỏ ngay lập tức. Trong giai đoạn thực hiện, giải thuật toán sắp xếp nhãn thời gian thực hiện các giao dịch theo thứ tự được xác định trong giai đoạn phê chuẩn. Nếu xảy ra xung đột thì có thể hủy giao có tác nhãn thời gian thấp hơn hoặc chờ một thời gian để giao tác xung đột hoàn thành và sẽ tiếp tục thực hiện.

## THẢO LUẬN

1. Phân biệt khái niệm tương tranh và xung đột.
2. Giao tác là gì, tại sao tất cả các hệ quản trị cơ sở dữ liệu đều cung cấp cơ chế này?
3. Lấy một số ví dụ về giao tác phẳng và giao tác phân tầng.
4. Trung tâm dịch vụ gồm nhiều nhân viên luôn túc trực trả lời khách hàng, nêu giải pháp thực hiện để không xảy ra hiện tượng một yêu cầu được phân phát cho nhiều nhân viên.
5. Tìm hiểu cơ chế quản lý giao tác trong hệ quản trị cơ sở dữ liệu Oracle.
6. Tại sao nên đưa các câu lệnh SQL vào thủ tục trong mỗi hệ quản trị cơ sở dữ liệu?
7. Tìm hiểu cơ chế điều khiển tương tranh trong hệ thống quản lý tập tin.
8. Một hệ thống thường xuyên gặp lỗi khóa chết, nguyên nhân là gì và nêu giải pháp khắc phục triệt để vấn đề này.
9. So sánh phương pháp điều khiển tương tranh bằng khóa với phương pháp điều khiển tương tranh lạc quan.
10. So sánh phương pháp điều khiển tương tranh bằng khóa với phương pháp điều khiển tương tranh dựa trên nhãn thời gian.

## CHƯƠNG 7: PHỤC HỒI VÀ TÍNH CHỊU LỖI

Lỗi có thể xảy ra trong bất kỳ thời điểm nào, nếu hệ thống phụ thuộc hoàn toàn vào một máy chủ thì sẽ ảnh hưởng đến toàn bộ hệ thống. Một trong những mục tiêu quan trọng khi thiết kế hệ thống sao cho nó có khả năng tự phục hồi khi gặp lỗi và không ảnh hưởng nghiêm trọng đến hiệu năng tổng thể. Bất cứ khi nào lỗi xảy ra, hệ thống vẫn tiếp tục hoạt động theo cách chấp nhận được trong thời gian sửa lỗi, nghĩa là hệ thống có khả năng chịu lỗi.

Chương này sẽ giới thiệu một số phương pháp để đáp ứng mục tiêu chịu lỗi, bắt đầu bằng phương pháp dư thừa, sau đó sẽ trình bày các giải pháp tiến trình bền bỉ. Phương pháp gọi thủ tục từ xa vốn được coi là phương pháp truyền thông tin cậy, tuy nhiên nó cũng sẽ được phân tích kỹ lưỡng về các tình huống lỗi có thể xảy ra và các giải pháp khắc phục. Khả năng chịu lỗi liên quan chặt chẽ với truyền thông, vấn đề này đã được giới thiệu, phần này sẽ tập trung đào sâu những lỗi có thể xảy ra và biện pháp khắc phục. Giao tác phân tán là đã được giới thiệu sơ bộ trong phần điều khiển tương tranh, chương này sẽ đề cập chi tiết cách thực hiện trong các hệ thống phân tán. Cuối cùng sẽ đề cập đến vấn đề phục hồi sau khi xảy ra lỗi, những yêu cầu để có thể đảm bảo ít ảnh hưởng nhất đến hoạt động của hệ thống.

### 7.1 Giới thiệu tính chịu lỗi

Trong hệ thống phân tán, lỗi có thể xảy ra trên bất cứ thành phần nào, dù cho đó lỗi máy chủ hay lỗi mạng thì đều làm suy giảm hiệu năng và thậm chí một phần của hệ thống có thể gián đoạn dịch vụ. Một yêu cầu quan trọng khi xây dựng hệ thống phân tán là phải lường trước được các lỗi có thể xảy ra và sẵn sàng phương án xử lý sao cho tối thiểu hóa các ảnh hưởng của nó đến hệ thống. Chủ đề về tính chịu lỗi của hệ thống phân tán đã được nghiên cứu khá nhiều trong khoa học máy tính. Phương pháp dư thừa vẫn thường được áp dụng để hệ thống có khả năng chịu lỗi, những thành phần dư thừa luôn sẵn sàng phục vụ khi gặp lỗi.

#### 7.1.1 Khái niệm tính chịu lỗi

Khả năng chịu lỗi của một hệ thống liên quan mật thiết tới khái niệm độ tin cậy, một hệ thống được coi là có khả năng chịu lỗi nếu đáp ứng được bốn yêu cầu cơ bản về tính sẵn sàng, đáng tin cậy, an toàn và khả năng bảo trì hệ thống. Tính sẵn sàng là đặc tính có thể phục vụ ngay khi có yêu cầu dịch vụ, hệ thống có khả năng hoạt động chính xác tại bất kỳ thời điểm nào.

Tính đáng tin cậy liên quan đến mức độ chính xác của thông tin mà không phụ thuộc vào yếu tố thời gian, hệ thống có thể chạy liên tục mà không bị lỗi. Khác với tính sẵn sàng, tính đáng tin cậy của hệ thống được định nghĩa là hệ thống có thể chạy trong một khoảng thời gian chứ không phải mọi thời điểm, một hệ thống có độ đáng tin cậy cao nếu nó có khả năng chạy liên tục trong thời gian dài mà không xảy ra lỗi. Nếu mỗi giờ chỉ cần gián đoạn dịch vụ 1 mili giây thì tính sẵn sàng đạt 99.99997%, đó là tỉ lệ khá cao nhưng vẫn không được coi là đáng tin cậy. Tính đáng tin cậy cũng đòi hỏi cung cấp mức

độ bảo mật cao và tính toàn vẹn, tuy nhiên đây là những vấn đề lớn nên sẽ được đề cập sau.

Tính an toàn của hệ thống thể hiện ở khả năng bảo đảm an toàn ngay cả khi hệ thống tạm thời gặp lỗi dẫn đến hoạt động không chính xác, không gây ra những hậu quả nghiêm trọng. Những hệ thống điều khiển đòi hỏi độ an toàn phải rất cao, một số thảm họa hàng không đã từng xảy ra được ghi nhận do lỗi phần mềm điều khiển, hệ thống dữ liệu toàn cầu đã từng ngừng hoạt động 24 giờ chỉ vì lỗi khi nâng cấp một phần mềm bảo mật.

Khả năng bảo trì hệ thống thể hiện ở mức độ dễ dàng sửa chữa hệ thống sau khi xảy ra lỗi, nó phải được thực hiện một cách đơn giản và trong thời gian nhanh nhất và tổn thất ở mức thấp nhất. Hệ thống có khả năng bảo trì sẽ giúp nâng cao tính sẵn sàng, nếu có thể phát hiện và sửa lỗi tự động thì có thể coi như lỗi chưa từng xảy ra. Phần cuối chương này sẽ giới thiệu về các chiến lược phục hồi tự động sau khi gặp lỗi, đây là vấn đề phức tạp và không dễ dàng thực hiện.

### 7.1.2 Phân loại lỗi

Một hệ thống bị coi là thất bại nếu nó không cung cấp chính xác các dịch vụ như đã thiết kế, nếu một hệ thống công bố các dịch vụ nhưng tại một thời điểm nào đó không thể cung cấp cho người sử dụng thì coi như hệ thống đã thất bại. Nếu coi hệ thống phân tán là tập hợp máy tính trao đổi thông tin với nhau thì lỗi có thể xảy ra ở trên các máy tính hoặc trên hạ tầng mạng. Một hệ thống bị thất bại thì phải có nguyên nhân nào đó, một số nguyên nhân có thể dễ dàng loại bỏ, nhiều trường hợp đã khắc phục nhưng vẫn lỗi vẫn lặp lại. Ví dụ truyền một thông điệp, nếu đó là lỗi tràn vùng đệm thì có thể dễ dàng xử lý, nhưng đó là do ảnh hưởng của thời tiết hoặc bão từ trường thì khó có thể sửa chữa môi trường truyền dẫn.

Hệ thống phân tán không những phụ thuộc vào phần cứng máy tính và hạ tầng viễn thông mà còn phụ thuộc vào các phần mềm, nhiều trường hợp lập trình viên thiếu kinh nghiệm hoặc không lường trước các tình huống có thể xảy ra. Việc vận hành sai qui cách cũng có thể gây ra lỗi, nếu các máy chủ hoạt động phụ thuộc lẫn nhau thì một máy chủ gặp lỗi có thể ảnh hưởng đến các máy chủ khác.

Tính đáng tin cậy liên quan chặt chẽ với kiểm soát lỗi, lỗi là chuyện bình thường nhưng nếu không phát hiện được lỗi mới là điều bất bình thường. Phần mềm trước hết phải có khả năng phát hiện lỗi, sau đó mới nói đến các mức độ cao hơn như ngăn ngừa, dung thứ, loại bỏ và thậm chí dự báo lỗi. Một trong những mục tiêu quan trọng khi xây dựng hệ thống phân tán là khả năng chịu lỗi, nghĩa là hệ thống có thể cung cấp dịch vụ ngay cả khi lỗi xảy ra.

Lỗi được chia thành ba loại: lỗi nhất thời, lỗi chập chờn và lỗi vĩnh cửu. Lỗi nhất thời chỉ xảy ra một lần sau đó biến mất, nếu thực hiện lại thì lỗi đó không xuất hiện lại. Ví dụ chức năng quản lý truy nhập, người sử dụng thất bại trong lần đăng nhập thứ nhất, nhưng thực hiện lại thì thành công mặc dù không hề thay đổi các thông tin về tài khoản và mật khẩu. Lỗi chập chờn là loại lỗi mà chúng xuất hiện nhiều lần, có thể theo chu kỳ hoặc không, nó có thể gây hậu quả nghiêm trọng vì chúng rất khó xác định được. Lỗi

vĩnh cửu là loại lỗi vẫn tồn tại ngay cả khi thành phần gây lỗi đó đã được sửa chữa, ví dụ lỗi phần cứng như ổ đĩa trên máy tính hoặc cáp truyền thông, lỗi khóa chết trong các hệ quản trị cơ sở dữ liệu sử dụng cơ chế khóa.

### **7.1.3 Các mô hình thất bại**

Khi hệ thống thất bại, nhiều dịch vụ sẽ không được cung cấp, nếu có cung cấp thì chất lượng cũng không đảm bảo như thiết kế. Hoạt động của hệ thống phụ thuộc rất nhiều thành phần, từ hạ tầng mạng cho đến phần mềm chạy trên các máy chủ cũng như máy tính của người sử dụng. Nếu máy khách bị hỏng thì phạm vi ảnh hưởng rất nhỏ, nhưng nếu hệ thống máy chủ cơ sở dữ liệu bị sập thì sẽ ảnh hưởng đến rất nhiều người sử dụng. Mô hình thất bại xác định tỷ lệ, tần suất thất bại và các chi tiết thống kê khác được quan sát thấy trong các hệ thống thực, các mô hình này chủ yếu được sử dụng trong các hệ thống mô phỏng và dự đoán để tái tạo các thất bại. Để hiểu rõ hơn về mức độ nghiêm trọng thực sự của mỗi loại thất bại, bảng 7.1 liệt kê năm mô hình thường gặp.

**Bảng 7.1** Các mô hình thất bại

Mô hình thất bại	Mô tả hiện tượng
Sụp đổ	Đang hoạt động bình thường nhưng đột ngột dừng lại
BỎ sót	Không phản hồi được các yêu cầu gửi đến: <ul style="list-style-type: none"> <li>- Gửi yêu cầu không thành công</li> <li>- Nhận yêu cầu không thành công</li> </ul>
Thời gian	Phản hồi ngoài khoảng thời gian cho phép
Kết quả sai	Giá trị trả về không đúng như thiết kế
Tùy tiện	Không phản hồi yêu cầu hoặc có phản hồi nhưng kết quả không dựa trên bất kỳ cơ sở nào vào những thời điểm không lường trước

Thất bại dạng sụp đổ nghĩa là một thành phần trong hệ thống vẫn hoạt động bình thường trước khi dừng hoạt động một cách đột ngột, không còn nhận được bất kỳ thông tin nào về thành phần đó, thành phần ở đây có thể là máy tính hoặc các tiến trình của hệ thống phân tán, cách duy nhất khắc phục duy nhất là khởi động lại. Ví dụ điển hình cho loại này là hệ điều hành dừng đột ngột và chỉ có một giải pháp duy nhất là khởi động lại máy tính. Khởi động lại máy tính không phải là giải pháp ưu tiên lựa chọn, nó có thể dẫn tới những hậu quả rất nghiêm trọng. Ví dụ, hệ quản trị cơ sở dữ liệu bị lỗi thì nên tìm hiểu nguyên nhân và đưa ra giải pháp xử lý, ưu tiên các biện pháp an toàn để tắt dịch vụ sau đó chạy lại. Khởi động lại máy tính trong trường hợp này có thể dẫn tới hiện tượng mất toàn bộ thông tin điều khiển, toàn bộ dữ liệu sẽ không thể sử dụng được, tuy nhiên thời gian xử lý lỗi kéo dài có thể ảnh hưởng đến tính sẵn sàng của hệ thống.

Thất bại dạng bỏ sót xảy ra khi máy khách gửi yêu cầu nhưng không nhận được phản hồi từ máy chủ, đã có lỗi xảy ra trong quá trình gửi yêu cầu từ máy khách đến máy chủ và ngược lại. Có thể máy khách không thể gửi yêu cầu đến máy chủ do lỗi máy khách hoặc lỗi mạng, có thể số lượng yêu cầu quá lớn tới mức vượt quá khả năng tiếp nhận của máy chủ. Ví dụ dịch vụ dạy học trực tuyến, lượng dữ liệu không lồ cho âm

thanh cũng như hình ảnh được lưu chuyển trên mạng giữa máy tính của giảng viên và các học viên có thể dẫn đến hiện tượng tràn vùng đệm hoặc vượt quá băng thông mạng.

Thất bại về thời gian xảy ra khi phản hồi nằm ngoài khoảng thời gian vật lý đã qui định, mỗi yêu cầu gửi đến máy chủ cơ sở dữ liệu thường qui định thời gian quá hạn, vượt quá thời gian này mà không nhận được kết quả thì phần mềm sẽ thông báo lỗi, thất bại trong truy vấn. Những thất bại này thường do năng lực xử lý của máy chủ, thời gian thực hiện quá lâu hoặc quá nhiều yêu cầu phải chờ đợi để được xử lý.

Kết quả sai là một thất bại nghiêm trọng, đó là kết quả không chính xác như đã thiết kế. Trong các hệ thống điều khiển, kết quả sai có thể dẫn đến hậu quả rất nghiêm trọng. Đây là vấn đề rất nhạy cảm, nhiều hệ thống hiện nay đã áp dụng trí tuệ nhân tạo, cách nhận dạng sai có thể gây nên những thảm họa. Ví dụ điển hình là việc xây dựng bảng định tuyến, thông tin cung cấp không sát với thực tế có thể dẫn đến vòng lặp vô hạn, nếu tiến trình P không liên lạc được với tiến trình Q thì không nên vội vàng kết luận tiến trình Q bị lỗi.

Nghiêm trọng nhất là các thất bại dạng tùy tiện, nó còn được gọi là thất bại Byzantine, máy khách nên chuẩn bị cho tình huống xấu nhất. Máy chủ có thể không trả lời máy khách nhưng cũng có thể trả về kết quả mà máy khách khó có thể phát hiện đó là kết quả không chính xác. Mô hình thất bại này có thể chia làm ba loại, thứ nhất là thất bại dừng hoạt động, tiến trình máy chủ không phản hồi và các tiến trình khác có thể phát hiện ra điều này, trường hợp tốt nhất nó sẽ thông báo về sự cố, nếu không sẽ chỉ đơn giản ngừng hoạt động. Loại thứ hai là thất bại nhưng im lặng, tiến trình máy chủ đột ngột hoạt động chậm lại các tiến trình không thể biết được nó còn hoạt động hay không, trường hợp này chỉ ảnh hưởng đến hiệu năng của hệ thống. Loại thứ ba là thất bại an toàn, tiến trình máy chủ trả về kết quả nào đó để các tiến trình khác nhận dạng nhưng nó không có giá trị sử dụng.

## 7.2 Các biện pháp đảm bảo tính chịu lỗi

Lỗi có thể xuất hiện bất cứ ở đâu tại bất cứ thời điểm nào nhưng hệ thống có thể phát hiện kịp thời và xử lý để không xảy ra thất bại, những biện pháp đó đảm bảo cho hệ thống có khả năng chống chịu với lỗi. Hệ thống phân tán hoạt động dựa trên hạ tầng mạng, điều đó cho thấy cần phải phân biệt lỗi trên mỗi máy tính hay lỗi mạng để có những biện pháp xử lý phù hợp.

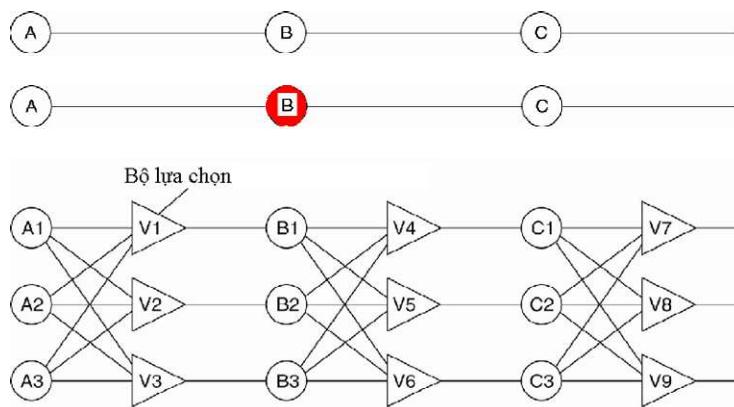
### 7.2.1 Che giấu lỗi bằng biện pháp dư thừa

Một hệ thống chịu được lỗi thì cần phải có khả năng che giấu được các lỗi, nói cách khác người sử dụng không hề biết hoặc ít biết về sự cố đối với mạng và dịch vụ. Một phương pháp phổ biến thường được áp dụng trong các hệ thống phân tán là xây dựng các thành phần dư thừa, dư thừa ở đây có thể là thông tin, thời gian và vật lý.

Dư thừa thông tin thực hiện bằng cách bổ sung thêm các dữ liệu dư thừa để phát hiện lỗi và hồi lỗi, ví dụ trong truyền dữ liệu thường thêm vào các bit kiểm tra chẵn lẻ, mã Hamming, CRC.... Nhân bản dữ liệu cũng được xếp vào loại dư thừa thông tin, nếu hệ thống chỉ dựa trên một bản sao dữ liệu thì nguy cơ thất bại rất cao, nếu xảy ra lỗi sụp đổ thì có thể dẫn đến mất mát thông tin.

Nếu dư thừa thời gian, một thao tác đã thực hiện nhưng gặp lỗi thì có thể thực hiện lại, kỹ thuật này phù hợp để khắc phục lỗi nhất thời và lỗi chập chờn. Có thể sửa lỗi bằng cách thực hiện lại các giao tác bị lỗi, tuy nhiên phải chú ý tính đúng đắn đối với các thao tác thực hiện theo thời gian thực.

Dư thừa vật lý được triển khai bằng cách bổ sung thêm tài nguyên vật lý, thay vì sử dụng một thiết bị thì sử dụng thêm nhiều thiết bị khác để dự phòng, nếu thiết kế hệ thống tốt thì có thể tận dụng để nâng cao hiệu năng hệ thống. Bổ sung thêm thiết bị hoặc kenh truyền vật lý dẫn đến chi phí đầu tư lớn hơn, việc quản lý cũng phức tạp hơn, do đó cần phải tính toán sao cho hợp lý phù hợp với tiềm năng tài chính.



Hình 7.1 Xây dựng hệ thống dư thừa theo kiểu chuyển mạch

Hình 7.1 minh họa kỹ thuật dư thừa tương tự như hệ thống chuyển mạch trong mạng viễn thông, bình thường hệ thống chỉ bao gồm 3 thành phần chính A, B, C. Để đảm bảo khả năng chịu lỗi, các thành phần dư thừa  $A_i, B_i, C_i$  tương ứng với A, B, C được thêm vào hệ thống và duy trì ở trạng thái dự phòng. Khi có lỗi xảy ra, hệ thống sẽ tự động chuyển đến một trong các thành phần dư thừa tương ứng, sau khi khắc phục xong lỗi, hệ thống sẽ chuyển về xử lý trên các thành phần chính của hệ thống. Cài đặt dư thừa vật lý hoàn toàn đơn giản khi các thiết bị chuyển mạch và thiết bị định tuyến đều đã cung cấp các tính năng này, tuy nhiên về mặt phần mềm thì cần phải có tiến trình luôn sẵn sàng phục vụ, chúng phải hoạt động gắn kết với nhau, điều đó dẫn đến khái niệm tiến trình bền bỉ.

### 7.2.2 Tiến trình bền bỉ

Nhiệm vụ cơ bản trong việc cung cấp tính năng chịu lỗi trong hệ thống phân tán là cách xử lý khi gặp lỗi, dư thừa vật lý là điều cần thiết nhưng quan trọng hơn là những phần mềm điều khiển. Phần mềm phải có khả năng phát hiện lỗi một cách chính xác và tự động xử lý, điều này sẽ nâng cao tính sẵn sàng của hệ thống. Tư tưởng của giải pháp tiến trình bền bỉ là xây dựng một nhóm các tiến trình, tất cả các tiến trình đều sẵn sàng phục vụ, nhưng nếu nhiều tiến trình cùng xử lý thì dẫn tới hiện tượng lặp. Có thể giải quyết bằng cách tắt cả các tiến trình đều xử lý nhưng chỉ lấy kết quả của một tiến trình, giải

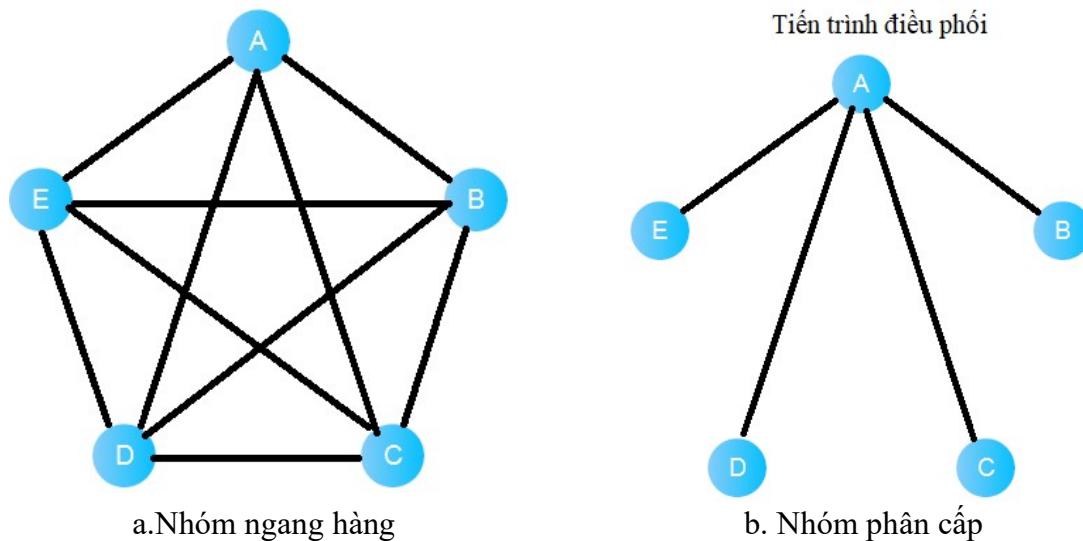
pháp này khá phức tạp và tốn kém. Giải pháp tốt hơn, chọn ra một tiến trình duy nhất để xử lý, nếu tiến trình đó bị lỗi đã có ngay tiến trình khác sẵn sàng thay thế.

Cần phải có cơ chế quản lý nhóm và các thành viên, quan hệ giữa nhóm và tiến trình ở dạng nhiều – nhiều, nghĩa là mỗi nhóm gồm nhiều tiến trình và mỗi tiến trình có thể tham gia nhiều nhóm. Nhiều tiến trình được gộp vào một nhóm là cách trừu tượng hóa, dưới góc nhìn của tiến trình khác thì nhóm các tiến trình cũng chỉ là một tiến trình. Ví dụ, nhóm A gồm tập các tiến trình  $\{A_1, A_2, \dots, A_n\}$  chạy trên nhiều máy tính, nếu tiến trình  $A_i$  lỗi thì đã có  $N-1$  tiến trình khác sẵn sàng thay thế, tiến trình B nhận N tiến trình như một tiến trình đơn A, nó chỉ cần gửi yêu cầu đến một thành viên của nhóm, việc lan tỏa đến các thành viên khác hoàn toàn do nội bộ nhóm qui định.

Như vậy, giải pháp chính để xây dựng tính năng chịu lỗi là nhân bản tiến trình và đưa chúng vào một nhóm, khi có yêu cầu được gửi đến thì tất cả các thành viên trong nhóm đều có trách nhiệm xử lý. Vấn đề mới nảy sinh là việc quản lý và phối hợp hoạt động giữa các tiến trình trong nhóm, điều này liên quan đến cách tổ chức nhóm.

#### 7.2.2.1 Tổ chức nhóm

Mỗi nhóm có thể được tổ chức theo kiến trúc ngang hàng hoặc phân cấp, một tiến trình có thể là thành viên của nhiều nhóm khác nhau. Hình 7.2 thể hiện hai cách tổ chức nhóm tiến trình trong các hệ thống phân tán, mỗi cách tổ chức có những ưu điểm và nhược điểm riêng.



Hình 7.2 Nhóm các tiến trình trong hệ thống phân tán

Nếu tổ chức nhóm theo kiến trúc ngang hàng, tất cả các tiến trình đều có vai trò như nhau và quyết định của chúng mang tính chất tập thể. Khi có yêu cầu gửi đến thì tất cả các thành viên đều nhận được, việc lan tỏa đến từng thành viên là công việc của nội bộ nhóm chứ không phải của tiến trình gửi yêu cầu, tiến trình gửi yêu cầu nhận nhóm như một tiến trình đơn như đã đề cập trên. Chỉ một thành viên được phép xử lý yêu cầu, nếu không sẽ dẫn đến lỗi lặp, do đó các tiến trình phải thực hiện giải thuật bầu chọn. Nếu

một tiến trình bị lỗi thì chỉ làm cho kích thước của nhóm giảm đi chứ không ảnh hưởng đến hoạt động của cả nhóm, tuy nhiên thời gian xử lý yêu cầu sẽ lâu hơn do phải mất thời gian lan tỏa thông điệp đến từng thành viên và thực hiện giải thuật bầu chọn.

Nếu tổ chức nhóm theo kiến trúc phân cấp, một tiến trình trong nhóm được chọn để đóng vai trò điều phối. Khi nhận được yêu cầu xử lý, tiến trình điều phối sẽ chuyển cho tiến trình nào đó trong nhóm thực hiện. Thời gian xử lý nhanh hơn so với kiến trúc ngang hàng, nếu thành viên điều phối bị lỗi thì khởi động lại giải thuật bầu chọn để tìm thành viên mới thay thế.

Nhiệm vụ tiếp theo là quản lý thành viên, cần phải có những qui định về tạo và xóa nhóm, cũng như cho phép các tiến trình tham gia hoặc rời khỏi nhóm. Cách tiếp cận tập trung được coi là khả thi, một máy chủ có chức năng duy trì danh sách nhóm và các thành viên của chúng, tiến trình muốn tham gia hoặc rời khỏi nhóm đều phải gửi yêu cầu đến máy chủ này. Đây là cách làm đơn giản và dễ triển khai, tuy nhiên cách tiếp cận này bộc lộ những nhược điểm của hệ thống tập trung, rủi ro tiềm ẩn nếu máy chủ nhóm bị lỗi.

Cách tiếp cận khác nhìn nhận các tiến trình dưới góc độ phân tán, tiến trình muốn tham gia hay rời khỏi nhóm đều phải gửi yêu cầu đến tất cả các tiến trình trong nhóm. Nếu một tiến trình bị sụp đổ và ngừng hoạt động thì không thể gửi yêu cầu rời khỏi nhóm, các tiến trình khác không thể liên lạc với nó, sẽ phải khởi động giải thuật đồng thuận phân tán để xác định trạng thái của tiến trình này. Vấn đề thứ hai là vấn đề đồng bộ, từ khi tham gia cho đến khi rời bỏ, mỗi tiến trình đều phải nhận được các yêu cầu gửi đến nhóm. Vấn đề cuối cùng là cách tạo hoặc hủy nhóm như thế nào, mỗi tiến trình đều có quyền tạo nhóm mới, nếu nhiều tiến trình cùng gửi yêu cầu tạo nhóm mới thì giải thuật sẽ trở nên rất phức tạp.

#### 7.2.2.2 *Che giấu thất bại và nhân bản*

Xây dựng nhóm tiến trình giống hệt nhau thực chất là nhân bản tiến trình, ứng dụng sẽ được nhân bản để chạy trên nhiều máy tính, nếu một số tiến trình sụp đổ thì sẽ có những tiến trình khác thay thế. Cách tổ chức phân cấp được triển khai trong thực tế với tên gọi “Nhân bản dựa trên thành phần chính”, một máy tính được chỉ định làm thành phần chính và các máy tính khác đóng vai trò dự phòng, nếu máy chính sụp đổ thì một trong những máy tính dự phòng sẽ thay thế. Cách tổ chức nhóm theo kiến trúc ngang hàng được triển khai trong thực tế với tên gọi “Nhân bản chủ động”, các máy tính đều có vai trò ngang nhau.

Câu hỏi đặt ra, cần phải nhân bản bao nhiêu tiến trình là đủ, nếu K tiến trình bị sụp đổ theo mô hình “im lặng” thì chỉ cần K+1 tiến trình là đủ vì một tiến trình vẫn còn hoạt động. Nếu K tiến trình sụp đổ theo mô hình dừng lại ngay mà không có bất kỳ thông báo nào cho nhóm thì cần nhân bản ít nhất 2K+1 tiến trình vì số tiến trình phản hồi chiếm đa số. Nếu lớn hơn K tiến trình bị lỗi, dù cho vẫn còn những tiến trình khác hoạt động, nhưng mọi kết quả thực hiện của chúng đều không đáng tin cậy.

#### 7.2.2.3 *Đồng thuận trong các hệ thống lỗi*

Tổ chức các tiến trình nhân bản thành nhóm làm tăng khả năng chịu lỗi, nếu máy khách có thể dựa trên quyết định của nó thông qua cơ chế bầu chọn thì vẫn có thể chấp nhận kết quả sai lệch của K trong số 2K+1 tiến trình. Vấn đề trở nên phức tạp nếu muốn

các tiến trình trong nhóm đạt được sự đồng thuận trong một số trường hợp như bầu chọn thành viên điều phối, ra quyết định cam kết trong các giao tác phân tán, phân chia công việc cho các tiến trình thực thi, thực hiện đồng bộ ... Nếu các tiến trình và truyền thông đều ở trạng thái hoạt động tốt thì sẽ không có vấn đề gì xảy ra, ngược lại sẽ nảy sinh vấn đề đạt được sự đồng thuận giữa các tiến trình. Mục đích chung của giải thuật đồng thuận phân tán là để tất cả các tiến trình không lỗi đạt được sự đồng thuận về một vấn đề nào đó sau một số bước hữu hạn. Vấn đề trở nên phức tạp hơn bởi thực tế các giả định khác nhau về hệ thống nền tảng đòi hỏi các giải pháp khác nhau, Turek và Shasha đã chỉ ra bốn trường hợp sau:

1. Hệ thống đồng bộ với hệ thống không đồng bộ, một hệ thống được coi là đồng bộ khi và chỉ khi các tiến trình vận hành trong chế độ theo sát nhau. Về hình thức, phải có hằng số  $s \geq 1$ , nếu bất kỳ tiến trình nào thực hiện  $s+1$  bước thì tiến trình khác cũng phải thực hiện ít nhất một bước, ngược lại sẽ là không đồng bộ.
2. Trễ truyền thông bị giới hạn hay không giới hạn, độ trễ bị giới hạn khi và chỉ khi bất kỳ một thông điệp nào cũng sẽ được phân phát trong một khoảng thời gian nhất định.
3. Phân phát thông điệp theo thứ tự hay không theo thứ tự, phân phát thông điệp theo thứ tự nghĩa là bên nhận phải nhận được các thông điệp đúng theo thứ tự bên gửi đã phát đi. Thậm chí phân phát thông điệp theo thứ tự còn phải tính đến sự nhất quán thứ tự các thông điệp trên mỗi tiến trình của hệ thống phân tán.
4. Chuyển thông điệp theo phương pháp điểm – điểm hay truyền theo nhóm.

Trên hình 7.3 các trường hợp có thể đạt được sự đồng thuận được đánh dấu X, tất cả những trường hợp khác đều không có giải pháp thực hiện. Cần lưu ý rằng, các tiến trình của hệ thống phân tán trong thực tế hầu hết đều vận hành theo phương thức không đồng bộ, truyền thông điểm điểm và có ràng buộc thời gian khi Chuyển thông điệp.

		Thứ tự thông điệp							
		Không		Có					
Phương thức xử lý	Đồng bộ			X		Ràng buộc	Độ trễ truyền thông		
				X					
	Không đồng bộ	X	X	X	X	Không ràng buộc			
				X	X				
		Điểm – Điểm	Nhóm	Điểm – Điểm	Nhóm				

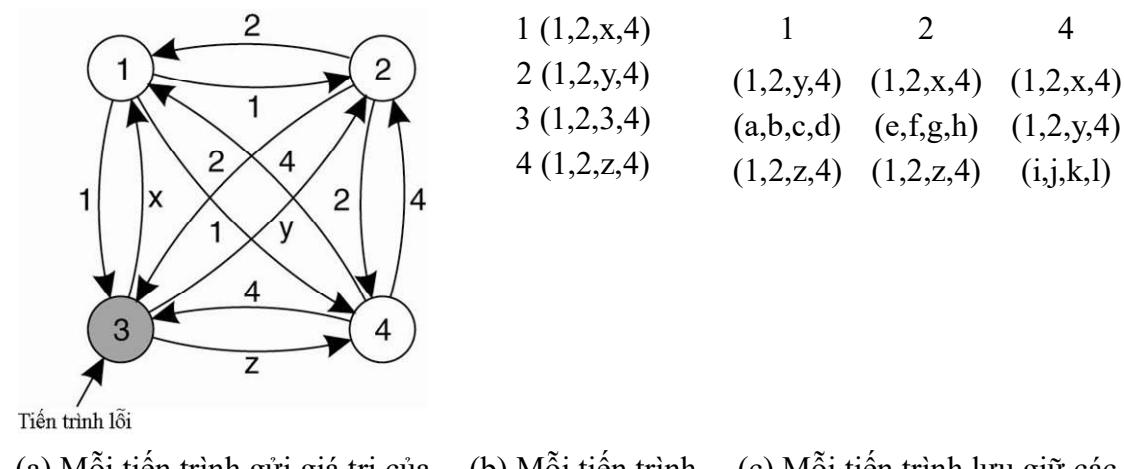
Hình 7.3 Các trường hợp đạt được sự đồng thuận

Vấn đề trên đã được Lamport nghiên cứu và đưa ra bài toán đồng thuận Bazantine, đề cập đến nhiều cuộc chiến mà một số đội quân cần phải đạt được sự đồng thuận, ví dụ về quân số, trong khi phải đối mặt với những thông tin sai lệch. Trong ví dụ này ta coi các tiến trình vận hành theo phương thức đồng bộ, thông điệp truyền theo phương pháp

điểm-điểm, có thứ tự và giới hạn thời gian truyền thông. Giả sử hệ thống gồm N tiến trình, mỗi tiến trình i cung cấp giá trị  $V_i$  cho các tiến trình khác, mục tiêu cần đạt trong đồng thuận lỗi sẽ được thực hiện bằng cách cho phép mỗi tiến trình xây dựng vector V gồm N phần tử với  $V[i]=V_i$  nếu tiến trình i không lỗi và  $V[i]$  không xác định nếu tiến trình i bị lỗi. Quá trình tiến tới sự đồng thuận sẽ được thực hiện theo bốn bước sau:

1. Mỗi tiến trình i không lỗi gửi giá trị  $V_i$  cho các tiến trình khác sử dụng phương pháp truyền thông điểm-điểm tin cậy, tiến trình lỗi gửi giá trị bất kỳ.
2. Kết quả nhận được từ bước 1 sẽ tập hợp lại thành vector V.
3. Mỗi tiến trình gửi vector V của bước 2 cho các tiến trình khác.
4. Mỗi tiến trình kiểm tra phần tử thứ i trong các Vector nhận được ở bước 3, nếu kết quả kiểm tra chiếm đa số thì đặt giá trị vào Vector kết quả đồng thuận, nếu không thì đặt giá trị không xác định UNKNOWN.

Ví dụ, hệ thống gồm bốn thành viên như trên hình 7.4 (a), các tiến trình 1, 2, 4 đều gửi giá trị tương ứng 1, 2 và 4 cho các tiến trình khác, tiến trình số 3 bị lỗi nên gửi các giá trị bất kỳ x, y và z cho các tiến trình 1, 2 và 4. Các thành viên khác trong hệ thống cần phải đảm bảo sự đồng thuận về việc tiến trình 3 có thực sự bị lỗi hay không. Kết quả thực hiện ở bước 2, mỗi tiến trình sẽ lưu giữ giá trị vector lần lượt là  $(1,2,x,4)$ ,  $(1,2,y,4)$ ,  $(1,2,3,4)$ ,  $(1,2,z,4)$ . Thực hiện bước 3, các tiến trình 1, 2 và 4 gửi giá trị vector đã nhận được ở bước 2 cho các tiến trình khác, trong khi đó tiến trình 3 tiếp tục gửi những giá trị không xác định, kết quả là các tiến trình 1, 2 và 4 sẽ nhận được các vector như hình 7.4 (c).



- (a) Mỗi tiến trình gửi giá trị của chúng cho các tiến trình khác      (b) Mỗi tiến trình xây dựng Vector      (c) Mỗi tiến trình lưu giữ các vector nhận được ở bước 3

Hình 7.4 Giải thuật đồng thuận phân tán

Tại bước 4, mỗi tiến trình sẽ tự tổng hợp cho kết quả cuối cùng cho mỗi phần tử thứ i của các vector nhận được trong bước thứ ba, nếu giá trị nào chiếm đa số thì sẽ đánh dấu giá trị đó trong phần tử tương ứng, ngược lại sẽ điền giá trị không xác định. Giá trị cuối cùng trong các tiến trình 1, 2 và 4 đều có giá trị là  $(1,2,\text{Unknown},4)$ , như vậy các tiến trình 1, 2 và 4 có thể đảm bảo chắc chắn các tiến trình số 1, 2, 4 không bị lỗi nhưng lại không thể đảm bảo tiến trình số 3 có bị lỗi hay không.

Xét lại ví dụ trên trong trường hợp có 3 tiến trình và 1 tiến trình bị lỗi, áp dụng thuật toán trên sẽ cho kết quả như trên hình 7.5, kết quả cuối cùng sau khi hoàn bước 3 không có bất kỳ phần tử nào chiếm đa số, vì vậy các tiến trình không lỗi 1 và 2 đều không đạt được sự đồng thuận. Lamport đã chứng minh rằng, một hệ thống có k tiến trình lỗi thì phải có  $2k+1$  tiến trình không lỗi, như vậy tổng số tiến trình phải là  $3k+1$ , nói cách khác phải có trên  $2/3$  số tiến trình không lỗi.

	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>1 (1,2,x)</td><td>1</td><td>2</td></tr> <tr> <td>2 (1,2,y)</td><td>(1,2,y)</td><td>(1,2,x)</td></tr> <tr> <td>3 (1,2,3)</td><td>(a,b,c)</td><td>(d,e,f)</td></tr> </table>	1 (1,2,x)	1	2	2 (1,2,y)	(1,2,y)	(1,2,x)	3 (1,2,3)	(a,b,c)	(d,e,f)	
1 (1,2,x)	1	2									
2 (1,2,y)	(1,2,y)	(1,2,x)									
3 (1,2,3)	(a,b,c)	(d,e,f)									
(a) Mỗi tiến trình gửi giá trị của chúng cho các tiến trình khác	(b) Mỗi tiến trình xây dựng Vector	(c) Mỗi tiến trình lưu giữ các vector nhận được ở bước 3									

Hình 7.5 Không xác định được sự đồng thuận

Một cách khác nhìn nhận vấn đề trên, hệ thống cần phải đạt được đa số phiếu bầu của các tiến trình không lỗi bất chấp sự hiện diện của các tiến trình lỗi. Với K tiến trình lỗi, hệ thống cần phải được đảm bảo rằng bầu chọn của chúng, cùng với bất kỳ tiến trình không lỗi nào bị nhầm lẫn vì các tiến trình lỗi, đều không trái với đa số bầu chọn của các tiến trình không lỗi. Với  $2K + 1$  tiến trình không có lỗi, điều này có thể đạt được bằng cách yêu cầu tiến tới đồng thuận chỉ khi nào có trên hai phần ba số phiếu bầu giống nhau.

Nói cách khác, nếu trên hai phần ba tiến trình đồng thuận về cùng một quyết định, quyết định này tương ứng với đa số phiếu của nhóm các tiến trình không lỗi. Tuy nhiên, tiến tới sự đồng thuận có thể sẽ còn tồi tệ hơn, Fischer đã chứng minh rằng, trong các hệ thống phân tán mà các thông điệp không đảm bảo được phân phát trong một khoảng thời gian hữu hạn thì đồng thuận cũng không thể đạt được ngay cả khi chỉ có một tiến trình bị lỗi, mặc dù tiến trình đó bị lỗi thầm lặng, những hệ thống này làm chậm các tiến trình một cách tùy tiện không thể phân biệt với các tiến trình bị lỗi sụp đổ.

Nhiều kết quả lý thuyết khác cũng chỉ ra khi nào có thể đạt được sự đồng thuận, ví dụ các kết quả nghiên cứu của Barborak, Turek và Shasha. Ngoài mô hình đồng thuận đã đề cập trên còn phải kể tới vấn đề đồng thuận trong các môi trường cộng tác, điều này rất khó đạt được trong các hệ thống thuộc về nhiều miền quản lý khác nhau, Aiyer đã đề cập tới vấn đề này trong các bài viết về tính chịu lỗi BAR (Byzantine, Altruism, Rationality).

#### 7.2.2.4 Phát hiện lỗi

Phát hiện lỗi là một trong những nhiệm vụ nền tảng để đảm bảo khả năng chịu lỗi trong các hệ thống phân tán, các tiến trình trong nhóm phải có khả năng phát hiện kịp thời những tiến trình nào đang bị lỗi hoặc có khả năng gây ra lỗi, từ đó thông báo đến các thành phần khác có liên quan hoặc thông báo cho người quản trị hệ thống. Việc phát hiện lỗi có thể dựa trên các thông tin trạng thái, ngưỡng giới hạn hoặc các tiến trình sử dụng

kỹ thuật trí tuệ nhân tạo, khi có lỗi xảy ra cần phải thực hiện các thao tác trong qui trình sửa lỗi.

Để phát hiện lỗi của các tiến trình, chỉ cần sử dụng hai cơ chế chủ động hoặc thụ động, trong cơ chế chủ động các tiến trình trong nhóm gửi cho nhau thông điệp “IS ALIVE” và tất nhiên sẽ chờ đợi phản hồi từ các tiến trình khác. Với cơ chế thụ động, mỗi tiến trình sẽ chờ nhận thông điệp từ các tiến trình khác, trong thực tế thường dùng cơ chế chủ động.

Đã có nhiều công trình nghiên cứu lý thuyết phát hiện lỗi, tự trung đều sử dụng cơ chế thời gian quá hạn để kiểm tra xem một tiến trình có bị lỗi hay không. Cơ chế này có hai điểm yếu cơ bản, thứ nhất mạng không đáng tin cậy và thứ hai thời gian quá hạn chỉ là dữ liệu thô. Khi gửi thông điệp đến tiến trình khác, không thể vội vàng kết luận tiến trình đó đã sụp đổ chỉ vì lý do không nhận được phản hồi. Mỗi tiến trình cần thường xuyên trao đổi thông tin với các tiến trình có liên quan, phải phân biệt được lỗi mạng hay lỗi trên mỗi nút mạng, bất kỳ tiến trình nào phát hiện tiến trình khác sụp đổ thì phải thông báo tới các tiến trình có liên quan.

### **7.2.3 Truyền thông khách/chủ tin cậy**

Tính chịu lỗi của hệ thống phân tán mới chỉ tập trung giải quyết trường hợp tiến trình sụp đổ, tuy nhiên cũng cần phải chú ý đến các lỗi truyền thông. Lỗi truyền thông có thể do mất kênh truyền, thất lạc thông tin, quá thời gian, lặp thông điệp..., trong thực tế khi xây dựng các kênh truyền thông tin cậy tiêu điểm tập trung vào việc che giấu lỗi mất kênh truyền và thất lạc thông tin. Để khắc phục lỗi truyền thông, người ta thường sử dụng phương pháp truyền thông điểm-điểm hoặc truyền thông theo nhóm, truyền thông tin cậy điểm-điểm dựa trên các giao thức truyền tin cậy như TCP.

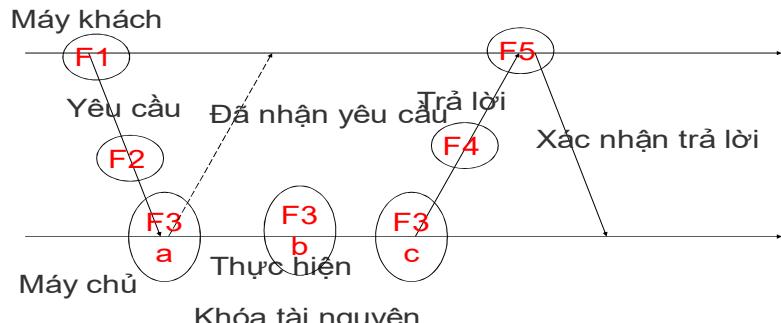
#### **7.2.3.1 Truyền thông điểm – điểm**

Trong hệ phân tán, truyền thông điểm - điểm tin cậy được thiết lập bằng cách sử dụng các giao thức truyền thông tin cậy TCP, nó che giấu được lỗi bỏ sót bằng số tuần tự của đoạn tin và gửi lại, những lỗi như vậy hoàn toàn trong suốt đối với thành phần TCP trên máy khách. Tuy nhiên, sử dụng giao thức truyền thông tin cậy không khắc phục được lỗi sụp đổ, lỗi này xuất hiện khi liên kết TCP đột ngột bị ngắt và các thông điệp không thể tiếp tục gửi qua kênh truyền, thông thường máy khách sẽ nhận được thông báo lỗi kênh truyền. Khi hệ thống gặp lỗi sụp đổ thì liên kết TCP sẽ bị hủy, cách duy nhất là hệ thống tự động tạo một liên kết mới bằng cách gửi yêu cầu thiết lập lại liên kết.

#### **7.2.3.2 Những thất bại khi thủ tục từ xa**

Gọi thủ tục từ xa là một trong những phương pháp sử dụng phổ biến trong mô hình truyền thông khách/chủ, nó che giấu quá trình truyền thông bằng cách làm cho việc gọi thủ tục từ xa giống như gọi trên máy cục bộ. Nếu máy khách, máy chủ và môi trường mạng vận hành bình thường thì gọi thủ tục từ xa thực hiện rất tốt vai trò của nó, tuy nhiên vấn đề sẽ phát sinh khi một trong ba thành phần trên gặp lỗi và tất nhiên việc che giấu lỗi sẽ không thể thực hiện được nữa, gọi thủ tục từ xa gồm nhiều công đoạn, hình 7.6 thể hiện những thất bại có thể xảy ra.

Thất bại thứ nhất, tiến trình trên máy khách không thể xác định được tiến trình cung cấp dịch vụ trên máy chủ. Nguyên nhân gây lỗi có thể do tiến trình cung cấp dịch vụ trên máy chủ và tiến trình máy khách dùng các phiên bản phần mềm khác nhau hoặc do chính tiến trình cung cấp dịch vụ trên máy chủ bị lỗi. Tuy nhiên, nhiều trường hợp dịch vụ tên miền bị lỗi cũng làm cho tiến trình máy khách không nhận được địa chỉ hoặc địa chỉ không chính xác của máy chủ, do đó người phát triển hệ thống cần phải lưu ý đến vấn đề này. Khắc phục bằng cách sử dụng tính năng phát hiện lỗi trong ngôn ngữ lập trình ứng dụng, tuy nhiên không phải ngôn ngữ nào cũng hỗ trợ phát hiện lỗi, nếu tự viết một ngoại lệ hay điều khiển tín hiệu thì sẽ vi phạm tính trong suốt.



Hình 7.6 Những khả năng lỗi khi gọi thủ tục từ xa

Thất bại thứ hai, mất thông điệp yêu cầu từ tiến trình trên máy khách gửi đến tiến trình cung cấp dịch vụ trên máy chủ. Đây là loại lỗi dễ xử lý nhất, hệ điều hành hay Stub của tiến trình trên máy khách kích hoạt một bộ đếm thời gian khi gửi đi một yêu cầu, nếu bộ đếm thời gian đã trở về giá trị 0 mà vẫn không nhận được thông điệp phản hồi từ tiến trình cung cấp dịch vụ trên máy chủ thì nó sẽ gửi lại yêu cầu đó. Nếu máy khách nhận thấy có quá nhiều yêu cầu phải gửi lại thì nó sẽ xác nhận rằng máy chủ không hoạt động và sẽ quay lại thành kiểu lỗi "không xác định được máy chủ".

Thất bại thứ ba, tiến trình máy chủ bị sụp đổ. Sau khi nhận được yêu cầu từ tiến trình máy khách, tiến trình máy chủ sẽ thực hiện yêu cầu và phản hồi kết quả thực hiện cho máy khách. Tuy nhiên, tiến trình máy chủ có thể bị sụp đổ ngay khi chưa kịp xử lý yêu cầu, đang xử lý hoặc đã xử lý xong nhưng chưa phản hồi kết quả cho máy khách. Nếu tiến trình máy chủ bị sụp đổ thì máy khách sẽ không nhận được kết quả trả về, thông thường máy khách sẽ gửi lại, nhưng điều đó có thể gây nên lỗi lặp. Khi gặp lỗi kiểu này, ở phía máy chủ sẽ thực hiện theo ba chiến lược sau:

1. Đợi đến khi nào máy chủ hoạt động trở lại, tiến trình máy chủ sẽ có thực hiện yêu cầu đã nhận được trước khi sụp đổ, như thế RPC thực hiện ít nhất một lần.
2. Tiến trình máy chủ sau khi được phục hồi sẽ không thực hiện yêu cầu nhận được trước mà sẽ gửi lại thông báo thất bại xử lý, với cách này thì RPC thực hiện nhiều nhất một lần.
3. Không thực hiện lại, nếu tiến trình máy chủ sụp đổ thì máy khách không hề hay biết, như vậy RPC có thể được thực hiện nhiều lần cũng có thể không thực hiện lần nào.

Máy khách có thể thực hiện theo 4 chiến lược sau:

1. Máy khách không thực hiện gửi lại các yêu cầu vì thế không biết bao giờ yêu cầu đó mới thực hiện được hoặc có thể không bao giờ được thực hiện.
2. Máy khách liên tục gửi lại yêu cầu, như vậy có thể dẫn tới trường hợp một yêu cầu được thực hiện nhiều lần.
3. Máy khách chỉ gửi lại yêu cầu nào đó khi không nhận được thông điệp xác nhận phản hồi từ máy chủ thông báo đã nhận thành công. Trường hợp này, máy khách dùng bộ đếm thời gian, sau một khoảng thời gian xác định trước mà không nhận được xác nhận thì sẽ gửi lại yêu cầu đó.
4. Máy khách gửi lại yêu cầu nếu nhận được thông báo lỗi từ máy chủ.

**Bảng 7.2** Các trường hợp sụp đổ trên máy chủ

Thứ tự sự kiện	Mô tả
$M \rightarrow P \rightarrow C$	Sau khi đã gửi thông điệp hoàn thành và xử lý yêu cầu
$M \rightarrow C \rightarrow (P)$	Đã gửi thông điệp hoàn thành nhưng chưa bắt đầu xử lý
$P \rightarrow M \rightarrow C$	Đã xử lý và gửi thông điệp hoàn thành
$P \rightarrow C \rightarrow (M)$	Đã xử lý nhưng chưa gửi thông điệp hoàn thành
$C (\rightarrow P \rightarrow M)$	Chưa kết thúc xử lý, chưa gửi thông điệp hoàn thành
$C (\rightarrow M \rightarrow P)$	Chưa làm gì

Gọi thủ tục từ xa có thể thực hiện theo phương thức đồng bộ hoặc không đồng bộ, phương thức đồng bộ phải chờ xử lý xong yêu cầu mới phản hồi thông điệp đã hoàn thành, phương pháp không đồng bộ phản hồi thông điệp đã hoàn thành trước khi tiến hành xử lý. Để hiểu quá trình phục hồi của máy chủ, ký hiệu M là sự kiện gửi thông điệp đã hoàn thành, P là sự kiện đã hoàn thành xử lý và C là sự kiện máy chủ sụp đổ. Nếu máy chủ đang xử lý yêu cầu mà sụp đổ thì cũng coi như lỗi trước khi hoàn thành, quá trình xử lý được lưu trong nhật ký, do đó máy chủ chỉ cần tiếp tục công việc tại thời điểm đã sụp đổ. Bảng 7.2 liệt kê thứ tự các sự kiện này, đây là cơ sở để máy chủ thực hiện phục hồi sau khi sụp đổ.

**Bảng 7.3** Phối hợp các chiến lược giữa máy khách và máy chủ

Tiến trình máy khách	Tiến trình máy chủ					
	M → P			P → M		
	MPC	MC(P)	C(MP)	PMC	PC(M)	C(PM)
Không gửi lại	Tốt	Không thực hiện	Không thực hiện	Tốt	Tốt	Không thực hiện
Liên tục gửi lại	Lặp	Tốt	Tốt	Lặp	Lặp	Tốt
Gửi lại nếu không nhận được xác nhận	Lặp	Tốt	Không thực hiện	Lặp	Tốt	Không thực hiện
Gửi lại nếu nhận được thông báo lỗi	Tốt	Không thực hiện	Tốt	Tốt	Lặp	Tốt

Kết hợp các chiến lược xử lý trên máy khách, bảng 7.3 cho thấy tất cả các khả năng xảy ra. Dễ dàng nhận thấy, không có sự kết hợp nào giữa chiến lược máy khách và chiến lược máy chủ sẽ hoạt động chính xác trong tất cả các chuỗi sự kiện có thể xảy ra. Nguyên nhân là do máy khách không thể biết được thời điểm máy chủ bị sập, cần phải có sự phối hợp giữa máy khách và máy chủ, mỗi yêu cầu của máy khách phải được gán số tuần tự để tránh lỗi lặp và máy chủ phải ghi nhật ký sự kiện.

Thất bại thứ tư, mất thông điệp phản hồi từ tiến trình cung cấp dịch vụ trên máy chủ gửi kết quả cho tiến trình trên máy khách. Nếu sau khoảng thời gian qui định mà không nhận được phản hồi từ tiến trình máy chủ thì tiến trình máy khách gửi lại yêu cầu. Đây là trường hợp máy chủ đã xử lý yêu cầu, nếu tiến trình máy khách gửi lại thì sẽ bị lặp. Phát hiện yêu cầu gửi lại không khó, chỉ cần gán số tuần tự cho mỗi yêu cầu, xử lý yêu cầu gửi lại mới phức tạp, cần phải phân biệt lặp vô hại và lặp nguy hại.

Ví dụ, nếu khách hàng kiểm tra số dư tài khoản thì yêu cầu này vô hại, tiến trình máy khách gửi lại yêu cầu thì tiến trình máy chủ chỉ cần gửi thêm câu lệnh truy vấn đến cơ sở dữ liệu, đó là lặp vô hại. Nếu đó là yêu cầu chuyển tiền, nếu tiến trình máy chủ thực hiện lại thì yêu cầu chuyển tiền sẽ được thực hiện hai lần, đó là lặp nguy hại. Như vậy tiến trình máy chủ phải lưu kết quả trả về trong nhật ký, nếu nhận thấy đó là yêu cầu lặp nguy hại thì không xử lý mà chỉ đơn giản lấy kết quả trả về trong nhật ký.

Thất bại thứ năm, tiến trình máy khách bị lỗi ngay sau khi gửi yêu cầu tới tiến trình cung cấp dịch vụ trên máy chủ. Tiến trình máy khách gửi yêu cầu đến tiến trình máy chủ và bị sập trước khi nhận được kết quả trả về, công việc mà máy chủ thực hiện nhưng không có đích nào đợi để nhận kết quả. Như vậy sẽ gây lãng phí thời gian xử lý của máy chủ, trường hợp này có thể giải quyết bằng bốn phương pháp sau:

- Phương pháp 1: trước khi gửi đi yêu cầu, stub của máy khách sẽ tạo ra một bản ghi xác định công việc cần thực hiện và lưu nhật ký. Sau khi phục hồi, tiến trình máy khách sẽ lấy lại bản ghi đó và thực hiện công việc đã bị tạm dừng. Phương pháp này có nhược điểm về chi phí trong việc lưu lại mỗi bản ghi cho mỗi lời gọi thủ tục từ xa, hơn nữa cách xử lý trên tiến trình máy chủ cũng sẽ phức tạp tương tự như trường hợp mất thông điệp phản hồi.
- Phương pháp 2: chia thời gian hoạt động liên tục của máy khách thành các khoảng liên tục gọi là các thời kỳ, mỗi thời kỳ được đánh dấu bằng số tuần tự. Mỗi khi tiến trình máy khách phục hồi thì số tuần tự được tăng thêm một đơn vị, máy khách sẽ gửi thông báo đến tất cả các tiến trình có liên quan số hiệu thời kỳ mới của mình để hủy các yêu cầu của thời kỳ trước. Nếu các yêu cầu của thời kỳ trước chưa được xử lý thì việc hủy bỏ không có gì khó khăn, sẽ rất phức tạp với trường hợp yêu cầu đã được thực hiện.
- Phương pháp 3: tương tự như phương pháp 2 nhưng ít khắc nghiệt hơn, khi nhận được số hiệu thời kỳ mới, mỗi tiến trình máy chủ sẽ kiểm tra xem mình có đang thực hiện một yêu cầu từ xa hay không, nếu có sẽ cố xác định xem tiến trình máy khách nào đã gửi yêu cầu này, nếu không xác định được thì sẽ bị hủy bỏ.

- Phương pháp 4: quy định mỗi lời gọi thủ tục từ xa chỉ có một khoảng thời gian quá hạn T để thực hiện. Sau khi sụp đổ, máy khách sẽ đợi thêm một khoảng thời gian T trước khi khởi động lại, vấn đề đặt ra là phải lựa chọn giá trị khoảng thời gian T như thế nào cho hợp lý.

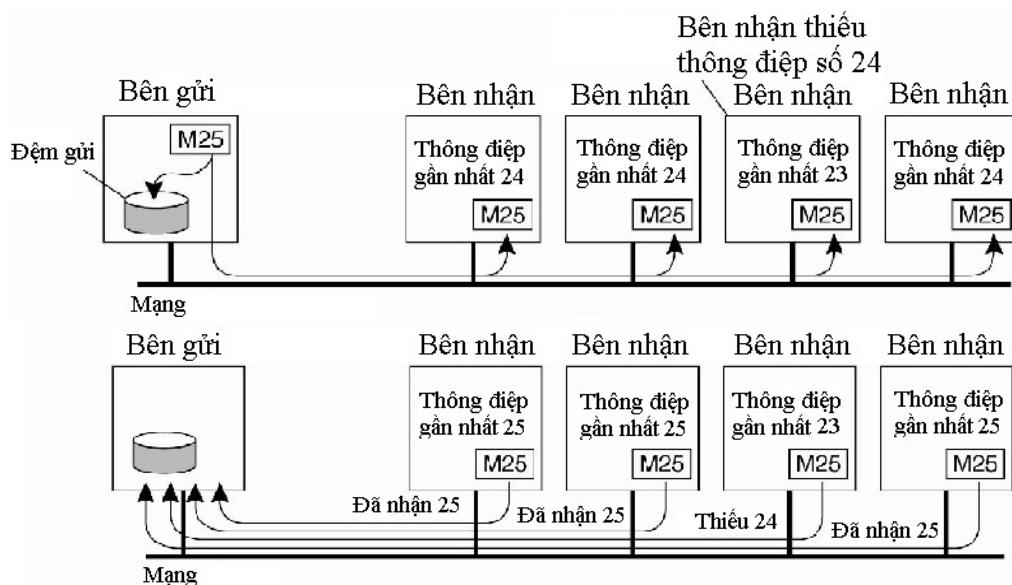
Tất cả bốn phương pháp trên đều có những nhược điểm của nó, việc loại bỏ các yêu cầu của phiên làm việc trước có thể dẫn đến hậu quả khó dự đoán. Ví dụ trường hợp máy rút tiền tự động ATM, sau khi gửi yêu cầu rút tiền thì máy ATM bị treo, nếu áp dụng phương pháp thứ nhất thì nảy sinh trường hợp cây ATM nhả tiền khi khách hàng đã rời đi. Nếu áp dụng phương pháp thứ hai, khó có thể phục hồi về giá trị cũ nếu trong thời gian đó có nhiều giao dịch khác liên quan đến tài khoản của khách hàng, tương tự như vậy nếu áp dụng các phương pháp khác.

#### 7.2.4 Truyền thông nhóm tin cậy

Truyền thông nhóm tin cậy phải có cơ chế để đảm bảo thông điệp đó đến được tất cả các thành viên trong nhóm. Nếu xảy ra lỗi thì sử dụng số tuần tự các thông điệp để khắc phục, các thông điệp được lưu tại một vùng đệm của bên gửi cho đến khi nhận được bản tin xác nhận từ tất cả các thành viên trong nhóm. Nếu bên nhận xác định là bị mất một thông điệp nào đó thì nó sẽ yêu cầu gửi lại. Thông thường, bên gửi sẽ tự động gửi lại thông điệp sau trong khoảng thời gian xác định trước nếu không nhận được thông điệp xác nhận.

##### 7.2.4.1 Truyền thông nhóm tin cậy

Truyền thông nhóm tin cậy nghĩa là thông điệp gửi đến nhóm tiến trình thì phải đến từng thành viên của nhóm, nó đóng vai trò quan trọng khi triển khai tiến trình bền bỉ và nhân bản dữ liệu, thông điệp phải được chuyển đến tất cả thành viên trong nhóm. Trong truyền thông nhóm tin cậy cơ bản, quá trình truyền tin được thực hiện theo cơ chế truyền thống, nghĩa là bên gửi sẽ chuyển thông điệp đến tất cả các thành viên trong nhóm và từng thành viên phải có trách nhiệm phản hồi kết quả đã nhận được thành công hay không, nếu có lỗi xảy ra thì áp dụng cơ chế sửa lỗi như truyền thông điểm – điểm.



Hình 7.7 Truyền thông nhóm tin cậy cơ bản

Hình 7.7 minh họa cơ chế hoạt động của truyền thông nhóm tin cậy cơ bản, khi thông điệp số 25 được chuyển đến nhóm, trong khi tất cả các tiến trình đều đã nhận được bản tin số 24 thì tiến trình thứ ba mới nhận được thông điệp số 23. Tất cả các tiến trình đều xác nhận với bên gửi đã nhận thành công thông điệp số 25, tiến trình thứ ba yêu cầu gửi lại từ thông điệp số 24. Nhận được yêu cầu từ tiến trình số ba, thông điệp số 24 và 25 đều phải gửi lại cho nhóm, nghĩa là tất cả các tiến trình đều phải nhận các thông điệp này.

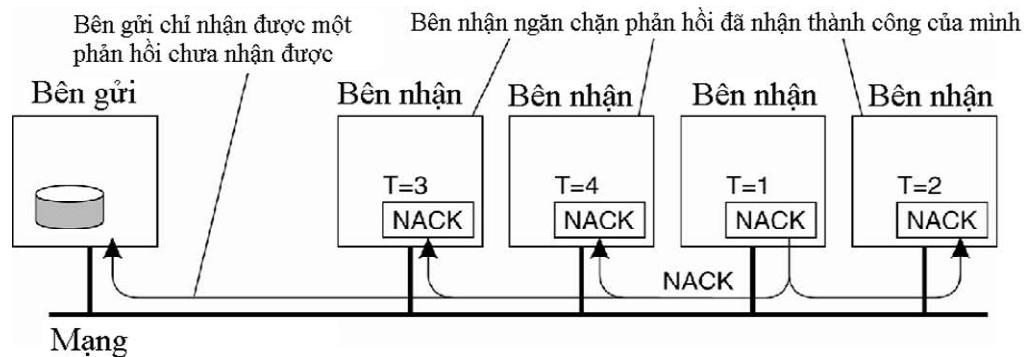
Như vậy, nếu một tiến trình nào đó trong nhóm yêu cầu gửi lại thì tất cả các tiến trình khác đều nhận lại thông điệp mặc dù trước đó đã nhận thành công. Bên gửi sẽ phải tiếp nhận phản hồi của tất cả các tiến trình trong nhóm, điều đó không những làm tăng đáng kể số lượng thông điệp mà còn có thể dẫn đến tràn vùng đệm nhận của bên gửi.

#### 7.2.4.2 Truyền thông nhóm tin cậy trong các hệ thống lớn

Truyền thông nhóm tin cậy cơ bản có hiệu năng thấp đối với các hệ thống lớn, giả sử phải gửi thông điệp đến N thành viên, khi đó bên gửi sẽ nhận được ít nhất N phản hồi từ các thành viên trong nhóm. Một giải pháp cải tiến có thể thực hiện bằng cách bên nhận không cần phải gửi phản hồi cho tất cả các thông điệp mà chỉ thông báo những thông điệp còn thiếu, do đó bên gửi phải lưu toàn bộ các thông điệp đã gửi. Để giảm thiểu số lượng phản hồi từ các thành viên nhận người ta đã áp dụng giải pháp phản hồi không phân cấp hoặc có phân cấp.

Giải pháp phản hồi không phân cấp giảm số lượng phản hồi đến bên gửi bằng cách sử dụng giao thức SRM do Floyd đề xuất năm 1997. Bên nhận không phản hồi thông điệp ACK để xác nhận đã nhận thành công, nếu phát hiện thiếu thông điệp phản hồi cho bên gửi và toàn bộ thành viên trong nhóm thông điệp NACK, nghĩa là chưa nhận được thông điệp hoặc thông điệp bị lỗi. Để giảm hơn nữa số lượng phản hồi NACK, giao thức vận dụng phương pháp tự triệt tiêu thông điệp NACK.

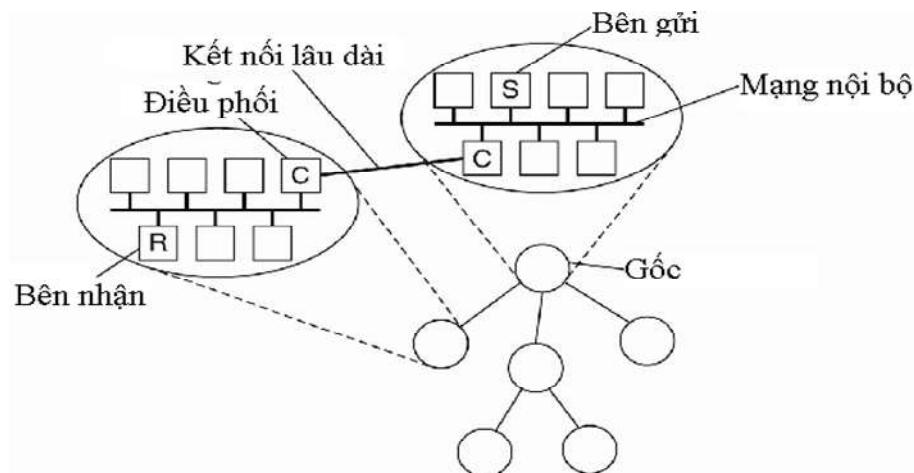
Hình 7.8 minh họa cơ chế hoạt động này, nếu một thành viên dự định phản hồi NACK nó sẽ chờ một thời gian nhất định, nếu không nhận được NACK tương ứng với thông điệp đó thì mới gửi các thành viên khác trong nhóm và bên gửi. Giao thức này đảm bảo chỉ phải gửi lại 01 thông điệp bị mất phụ thuộc vào việc lập lịch thông điệp phản hồi tại mỗi trạm nhận, nếu không cùng một thời điểm sẽ có nhiều trạm nhận gửi thông điệp NACK.



Hình 7.8 Phản hồi không phân cấp

Việc thiết lập thời gian trên toàn nhóm là điều không dễ dàng, bắt buộc mọi thành viên trong nhóm đều phải nhận thông điệp NACK ngay cả khi không cần thiết. Để khắc phục vấn đề này, có thể tạo thêm một nhóm mới phục vụ cho thông điệp NACK, điều này rất khó quản lý trong mạng qui mô lớn. Giải pháp cải tiến giao thức SRM cũng mang lại hiệu quả lớn, các thành viên trong nhóm hỗ trợ nhau phục hồi những thông điệp bị mất trước khi chuyển thông điệp NACK cho bên gửi.

Để có thể thực hiện truyền thông nhóm tin cậy cho nhóm với số lượng các tiến trình rất lớn thì tổ chức các nhóm theo kiến trúc phân cấp, hình 7.9 minh họa cách hoạt động của chúng, thông điệp chỉ cần chuyển đến nhóm các tiến trình mức cao nhất và chúng có trách nhiệm chuyển tiếp đến các tiến trình cấp thấp hơn. Việc chia thành các nhóm nhỏ hơn cho phép sử dụng các kịch bản truyền thông nhóm tin cậy cho từng nhóm, mỗi nhóm nhỏ sẽ đảm bảo một tiến trình đóng vai trò điều phối. Tiến trình điều phối có khả năng điều khiển việc gửi lại khi nhận được thông điệp thông báo lỗi. Tiến trình điều phối của mỗi nhóm sẽ có bộ đệm riêng, nếu không nhận được thông điệp thì nó sẽ gửi yêu cầu để nghị gửi lại đến tiến trình điều phối cấp cao hơn.



Hình 7.9 Điều khiển phản hồi phân cấp

Trong qui định của truyền thông tin cậy sử dụng thông điệp xác nhận, nếu tiến trình điều phối nhận thành công một thông điệp nó sẽ gửi thông điệp xác nhận tới tiến trình điều phối cấp cao hơn. Nếu tiến trình điều phối của một nhóm nhận được thông điệp xác nhận thành công việc chuyển thông điệp của tất cả các tiến trình trong nhóm gửi về thì nó sẽ xóa thông điệp khỏi bộ đệm của nó.

Phương pháp phân cấp này sinh vấn đề xây dựng cấu trúc cây, rất nhiều trường hợp yêu cầu cây phải có cấu trúc động nên phải có một cơ chế tìm đường cho cây. Khi một tiến trình muốn gửi thông điệp cho một tập các tiến trình khác theo dạng nhóm, nó sẽ không gửi thông điệp tới tất cả các tiến trình của nhóm mà chỉ gửi đến một nhóm nhỏ các tiến trình cần nhận thông điệp đó. Vấn đề đặt ra là phải đảm bảo gửi được thông điệp tới tất cả các tiến trình trong nhóm hoặc không được gửi tới bất kỳ tiến trình nào nếu một tiến trình trong nhóm sụp đổ.

### 7.3 Cam kết phân tán

Một yêu cầu quan trọng của giao tác phân tán là phải đảm bảo tính nguyên tử, nghĩa là các lệnh trong giao tác phải được thực thi thành công trên tất cả các thành viên trong nhóm, nếu có bất kỳ thành viên nào thực hiện không thành công thì các thành viên khác cũng phải hủy bỏ giao tác, đặc tính này có thể đạt được bằng cách cài đặt giao thức cam kết nguyên tử ACP. Giao thức này có thể cài đặt theo cơ chế một pha, hai pha..., số pha thực hiện càng nhiều thì tính chính xác càng cao nhưng đòi hỏi hiệu năng của hệ thống có thể bị suy giảm.

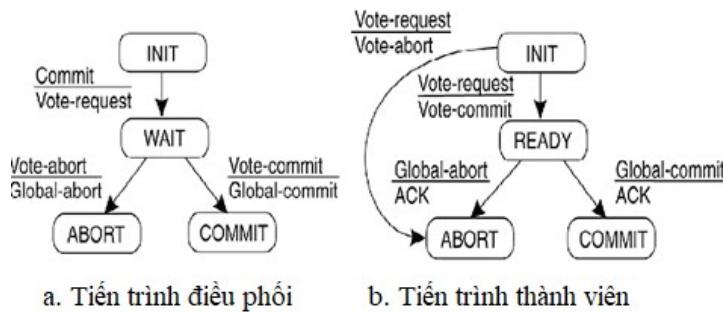
#### 7.3.1 Giao thức cam kết một pha

Giao thức cam kết một pha giảm độ phức tạp của cả thông điệp lẫn nhau ký sự kiện dựa trên giả thiết tất cả các thành viên đều thực hiện thành công giao tác. Giao thức cam kết một pha thực hiện bầu chọn không tường minh và nhật ký điều phối coi mỗi hoạt động của giao tác đều được xác nhận trong mỗi thành viên. Thao tác xác nhận trong các giao thức này không chỉ có ý nghĩa thông báo giao tác đã bảo toàn sự cách ly và các thuộc tính không phân tầng mà còn cho biết giao tác không vi phạm bất kỳ ràng buộc nhất quán nào của mỗi bên tham gia.

Mô hình thiết lập cam kết phải theo kiến trúc phân cấp và tiến trình điều phối đảm nhận nhiệm vụ cam kết. Trong cam kết một pha, tiến trình điều phối thông báo với tất cả các thành viên còn lại hoặc là thực hiện hoặc là hủy một thao tác nào đó. Nếu tiến trình thành viên nào đó không thực hiện được cũng không thể báo lại cho tiến trình điều phối biết, do đó không nên áp dụng cho các hệ thống đòi hỏi tính nhất quán cao.

#### 7.3.2 Giao thức cam kết hai pha

Xét một giao tác phân tán với các thành viên là một tập các tiến trình chạy trên các máy tính khác nhau và không có lỗi xảy ra, giao thức cam kết hai pha gồm pha biểu quyết và pha quyết định. Hình 7.10 thể hiện nguyên lý hoạt động của giao thức này, nó bao gồm tiến trình điều phối và các tiến trình thành viên. Sau khi tất cả các thành viên nhận được các câu lệnh của giao tác, trạng thái cam kết là INIT, trạng thái này sẽ biến đổi qua các bước thực hiện của giao thức.



Hình 7.10 Chuyển trạng thái trong giao thức cam kết hai pha

Pha biểu quyết thực hiện hai bước, tiến trình điều phối gửi một thông điệp cầu biểu quyết VOTE\_REQUEST tới tất cả các thành viên trong nhóm. Sau khi nhận được thông điệp VOTE\_REQUEST, nếu có thể thực hiện được thì thành viên đó sẽ gửi lại cho tiến

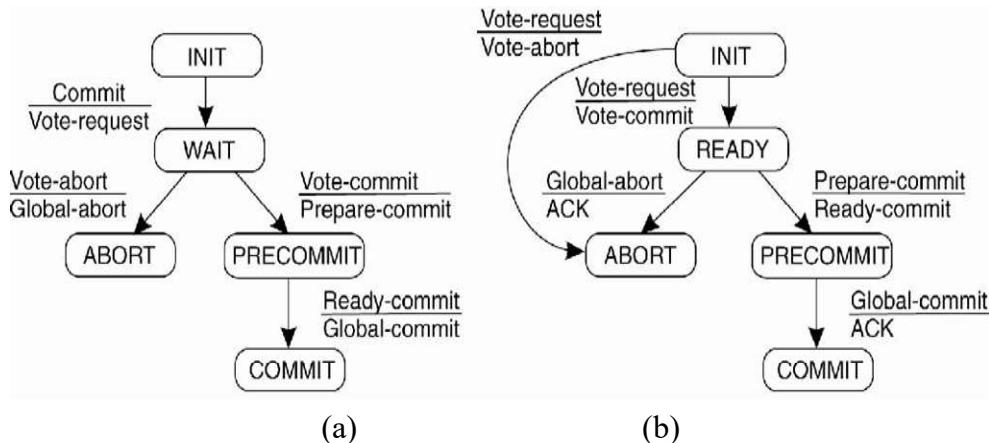
trình điều phối thông báo chấp nhận biểu quyết VOTE\_COMMIT, nếu không sẽ gửi lại thông báo từ chối VOTE\_ABORT.

Pha quyết định cũng thực hiện hai bước, tiến trình điều phối tập hợp tất cả các biểu quyết của các thành viên, nếu tất cả đều chấp nhận giao tác thì nó sẽ gửi một thông điệp GLOBAL\_COMMIT tới tất cả các thành viên, chỉ cần một thành viên gửi thông báo từ chối thì tiến trình điều phối quyết định hủy giao tác trên và sẽ gửi một thông điệp GLOBAL\_ABORT cho tất cả các thành viên trong nhóm.

Các thành viên sau khi đã gửi thông báo chấp nhận cam kết sẽ đợi phản hồi từ tiến trình điều phối, nếu nhận được thông báo GLOBAL\_COMMIT thì giao tác sẽ được chấp thuận, còn nếu nhận được GLOBAL\_ABORT thì giao tác sẽ bị hủy bỏ. Giao thức cam kết hai pha đảm bảo tính nguyên tử và phục hồi độc lập nhưng chi phí đáng kể khi thực hiện giao tác bình thường, điều đó ảnh hưởng bất lợi tới hiệu năng của hệ thống. Nguyên nhân là do chi phí phải trả vì sự phức tạp trong trao đổi thông điệp, một lượng lớn thông điệp dùng cho việc điều phối thao tác trên các nút khác nhau và sự phức tạp của nhật ký ghi lại các thao tác trên từng nút.

### 7.3.3 Giao thức cam kết ba pha

Nhược điểm chính của giao thức cam kết hai pha là tốn nhiều thời gian chờ đợi, cả tiến trình điều phối lẫn các thành viên còn lại đều phải chờ một thông điệp nào đó được gửi đến, nếu tiến trình điều phối bị lỗi thì hoạt động của cả hệ thống sẽ bị ảnh hưởng. Giao thức cam kết hai pha vận hành theo nguyên lý phong tỏa, nếu tiến trình điều phối bị lỗi thì các thành viên không thể đạt được quyết định cuối cùng.



(a) Máy trạng thái hữu hạn của thành phần điều phối  
(b) Máy trạng thái hữu hạn của thành viên

Hình 7.11 Chuyển trạng thái trong giao thức cam kết ba pha

Để khắc phục nhược điểm của cam kết hai pha trong trường hợp tiến trình điều phối bị lỗi, Skeen đã đề xuất giao thức cam kết ba pha, các trạng thái trong cam kết ba pha khá giống cam kết hai pha nhưng thêm một trạng thái tiền cam kết PRECOMMIT. Mặc dù giao thức cam kết ba pha được đề cập đến rất nhiều trong các tài liệu nghiên cứu

nhưng nó lại ít được sử dụng trong thực tế bởi vì hiện tượng phong tỏa trong giao thức cam kết hai pha rất ít khi xảy ra

Giống như giao thức cam kết hai pha, giao thức cam kết ba pha cũng gồm nhiều tiến trình trong đó có một tiến trình đóng vai trò điều phối, máy trạng thái hữu hạn của các tiến trình thể hiện trên hình 7.11. Bản chất của giao thức cam kết ba pha thể hiện ở chỗ các trạng thái của thành phần điều phối và các thành viên tham gia đáp ứng hai điều kiện sau:

1. Không có một trạng thái đơn lẻ nào mà từ đó có thể chuyển trực tiếp sang trạng thái cam kết hoặc hủy bỏ.
2. Không có trạng thái mà trong đó không thể đưa ra quyết định cuối cùng, và từ đó có thể chuyển sang trạng thái cam kết.

Hai điều kiện trên là cần thiết và đủ để đảm bảo giao thức cam kết ba pha có thể hoạt động trong chế độ không phong tỏa. Thành phần điều phối trong giao thức cam kết ba pha bắt đầu bằng việc gửi thông điệp yêu cầu bỏ phiếu Vote-Request đến tất cả các thành viên khác, sau đó sẽ chờ phản hồi từ các thành viên này. Chỉ cần một tiến trình bỏ phiếu hủy bỏ ABORT hoặc một tiến trình thành viên nào đó không phản hồi thì quyết định cuối cùng cũng sẽ là hủy bỏ, như vậy tiến trình điều phối sẽ gửi thông điệp hủy bỏ toàn cục Global\_Abort đến tất cả các tiến trình thành viên.

Nếu tất cả các tiến trình thành viên đều phản hồi phiếu bầu đồng ý cam kết, chắc chắn tất cả các tiến trình thành viên đã ở trạng thái sẵn sàng cam kết READY, nghĩa là giao tác có cơ hội được cam kết, tiến trình điều phối sẽ gửi thông điệp chuẩn bị cam kết Prepare-commit đến tất cả các thành viên. Nếu tại thời điểm này tiến trình điều phối bị lỗi thì sẽ không có một thông điệp nào được gửi đến các tiến trình thành viên, sau một thời gian nhất định từng tiến trình thành viên sẽ không được phép hủy bỏ hay cam kết giao tác.

Mỗi thành viên áp dụng điều kiện thứ nhất nên chúng phải tham khảo trạng thái của các thành viên khác, tất cả đều ở trạng thái quá hạn chờ thông điệp chuẩn bị cam kết, do đó giao tác sẽ tự động bị hủy. Nhận được thông điệp chuẩn bị cam kết từ tiến trình điều phối, các tiến trình thành viên sẽ phản hồi. Sau khi đã nhận được phản hồi từ tất cả các tiến trình thành viên, tiến trình điều phối sẽ gửi thông điệp cam kết toàn cục Global\_commit, nhận được thông điệp này các tiến trình thành viên sẽ thực hiện cam kết giao tác, như vậy giao tác phân tán đã hoàn thành trên tất cả các thành viên của hệ thống.

Xảy ra tình huống một thành viên nào đó bị lỗi nên không có phản hồi, như vậy tiến trình điều phối bị phong tỏa ở trạng thái chuẩn bị cam kết, nó biết chắc chắn tiến trình thành viên lỗi đã biết giao tác đang chờ để được cam kết, tiến trình điều phối có thể ra lệnh một cách an toàn cho những thành viên đang hoạt động bằng cách gửi thông điệp cam kết toàn cục cho nhóm các tiến trình tham gia. Ngoài ra, tiến trình điều phối dựa vào giao thức phục hồi để các tiến trình lỗi có thể thực hiện cam kết cho các giao tác khi chúng phục hồi trở lại.

Nếu tiến trình thành viên ở trạng thái READY, quá thời gian qui định mà vẫn không nhận được thông điệp chuẩn bị cam kết, nó sẽ liên lạc với tiến trình thành viên

khác, nếu trạng thái của tiến trình thành viên được hỏi là ABORT thì nó sẽ hủy bỏ giao tác, nếu là PRECOMMIT thì sẽ chuyển trạng thái của nó giống như thành viên đã hỏi, nếu là COMMIT thì sẽ thực hiện cam kết giao tác. Nếu thành viên ở trạng thái PRECOMMIT, quá thời gian qui định mà vẫn không nhận được thông điệp cam kết toàn cục, nó sẽ liên lạc với tiến trình thành viên khác, nếu trạng thái của tiến trình thành viên được hỏi là COMMIT thì nó thực hiện cam kết giao tác, nếu tất cả các tiến trình thành viên đều ở trạng thái PRECOMMIT thì nó cũng thực hiện cam kết giao tác.

## 7.4 Phục hồi

Lỗi có thể xảy ra ở bất kỳ thời điểm nào, một hệ thống đáng tin cậy phải không những có khả năng phát hiện lỗi mà còn phải biết phục hồi sau khi gặp lỗi, nói cách khác phục hồi là các biện pháp đưa hệ thống từ trạng thái bị lỗi trở về trạng thái không lỗi. Phục hồi trong các hệ thống phân tán tập trung vào việc duy trì chức năng và tính toàn vẹn của dữ liệu bất chấp lỗi xảy ra, nó bao gồm các chiến lược phát hiện lỗi, khôi phục trạng thái và đảm bảo cung cấp dịch vụ liên tục.

Phục hồi có vai trò quan trọng để đảm bảo tính đáng tin cậy, tính sẵn sàng và khả năng chịu lỗi của hệ thống. Khi một thành phần bị hỏng hoặc xảy ra lỗi, hệ thống phải khôi phục nhanh chóng và chính xác để giảm thiểu thời gian gián đoạn dịch vụ và mất dữ liệu. Các cơ chế sao lưu trạng thái, phục hồi tiến và phục hồi lùi giúp duy trì tính nhất quán, ngăn ngừa sụp đổ dây chuyền và đảm bảo hệ thống có thể tiếp tục hoạt động ngay cả khi có lỗi.

### 7.4.1 Các biện pháp phục hồi

Có hai cách tiếp cận phục hồi lỗi: phục hồi lùi và phục hồi tiến, tư tưởng của phương pháp phục hồi lùi là đưa hệ thống từ trạng thái lỗi ở thời điểm hiện tại về trạng thái không lỗi ở thời điểm trước khi xảy ra lỗi. Để làm được điều đó cần phải ghi lại trạng thái của hệ thống và khi hệ thống gặp lỗi sẽ chuyển về trạng thái đã sao lưu, thời điểm hệ thống thực hiện sao lưu dữ liệu và trạng thái gọi là điểm kiểm tra.

Khác với phục hồi lùi, phục hồi tiến không đưa hệ thống trở lại trạng thái không lỗi trước khi xảy ra lỗi mà chuyển hệ thống sang trạng thái mới không lỗi để có thể tiếp tục hoạt động. Khó khăn lớn nhất của giải pháp này là phải biết trước những lỗi nào có thể xảy ra, chỉ khi đó mới có thể sửa lỗi và chuyển sang trạng thái mới. Có thể dễ dàng giải thích sự khác biệt giữa phục hồi tiến và phục hồi lùi khi xem xét cài đặt truyền thông tin cậy. Khi phát hiện thấy một đoạn tin bị lỗi, bên nhận sẽ yêu cầu bên gửi phải gửi lại đoạn tin đó, nghĩa là hệ thống đã chuyển về trạng thái trước khi gửi đoạn tin lỗi, đó là cách thể hiện của phục hồi lùi. Một giải pháp khác là sử dụng mã sửa chữa, giả sử cần phải gửi k đoạn tin, bên gửi sẽ mã hóa thành n đoạn tin gọi là khối mã sửa chữa (n, k), chỉ cần nhận đủ k đoạn tin mã hóa sẽ có thể phục hồi lại được k đoạn tin nguyên gốc, nếu chưa đủ số lượng đoạn tin thì vẫn tiếp tục phải gửi cho đến khi dựng lại được các đoạn tin đã thất lạc.

Nói chung, kỹ thuật phục hồi lùi được sử dụng rộng rãi như một cơ chế tổng quát về phục hồi lỗi trong các hệ thống phân tán, ưu điểm của nó là tính độc lập với các tiến trình và như vậy có thể tích hợp vào tầng trung gian của hệ thống phân tán như một dịch

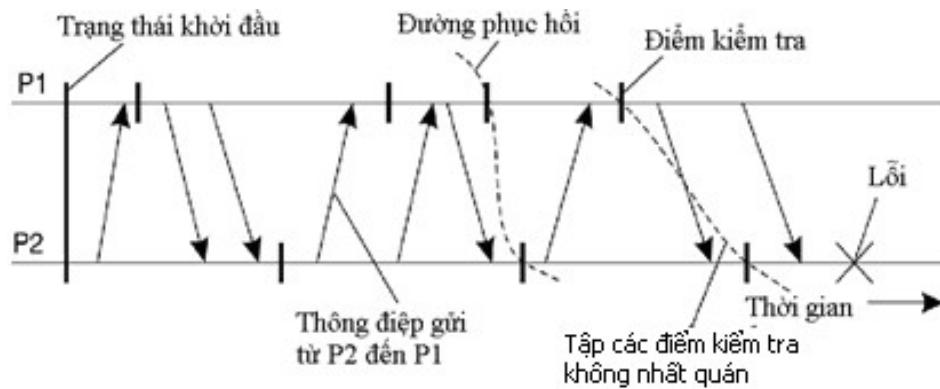
vụ đa năng. Tuy nhiên phục hồi lùi cũng bộc lộ một số vấn đề, việc phục hồi hệ thống về trạng thái trước sẽ đòi hỏi chi phí tương đối lớn về hiệu năng, cách đơn giản để thoát khỏi vấn đề này là tạo ra cơ chế ít tốn kém bằng cách khởi động lại các thành phần. Cơ chế phục hồi lùi độc lập với các ứng dụng phân tán, không có gì đảm bảo sau khi phục hồi lỗi tương tự lại không lặp lại và như vậy có thể xảy ra vòng lặp phục hồi. Cuối cùng, dựa vào cơ chế điểm kiểm tra nhưng không phải lúc nào cũng có thể phục hồi về trạng thái trước khi xảy ra lỗi, ví dụ trường hợp các máy trả tiền tự động, sau khi máy đã nhả tiền thì khó có thể rút lại được.

Hoạt động lưu trạng thái hệ thống có thể ảnh hưởng đến hiệu năng của hệ thống, do đó nhiều hệ thống phân tán có khả năng chịu lỗi kết hợp điểm kiểm tra với ghi nhật ký thông điệp, tiến trình gửi sẽ ghi nhật ký thông điệp trước khi gửi và tiến trình nhận sẽ ghi nhật ký thông điệp trước khi chuyển lên tầng ứng dụng để thực hiện. Nếu tiến trình nhận gặp lỗi, tiến trình gửi chỉ việc phục hồi về trạng thái của điểm kiểm tra sau và lấy các thông điệp trong nhật ký kể từ thời điểm thực hiện điểm kiểm tra để gửi lại, như vậy có thể giảm tần suất thực hiện các điểm kiểm tra và vì vậy sẽ giảm thiểu ảnh hưởng tới hiệu năng của hệ thống.

Để phục hồi về trạng thái không lỗi thì dữ liệu phải được lưu trữ an toàn, dữ liệu không bị hỏng hóc ngay cả khi tiến trình bị sụp đổ hoặc ngay cả khi phương tiện lưu trữ bị lỗi. Dữ liệu thường được lưu trữ ở bộ nhớ RAM hoặc ổ đĩa cứng, nếu lưu trữ trong RAM thì sẽ bị mất khi mất điện hoặc khi máy tính gặp lỗi. Dữ liệu trên ổ đĩa cũng có thể bị lỗi, vì vậy cần thiết phải sử dụng kỹ thuật RAID để nhân bản trên nhiều ổ đĩa, thậm chí cần phải xây dựng chính sách sao lưu dữ liệu sang các máy tính khác.

#### 7.4.2 Điểm kiểm tra

Điểm kiểm tra là một kỹ thuật cung cấp khả năng chịu lỗi, nó bao gồm việc trạng thái của hệ thống để có thể khởi động lại từ điểm đó trong trường hợp xảy ra lỗi. Trong hệ thống phân tán có khả năng chịu lỗi, phục hồi lùi đòi hỏi hệ thống phải đều đặn ghi lại trạng thái của nó vào vùng nhớ vĩnh cửu, đặc biệt phải ghi được trạng thái toàn cục nhất quán gọi là bản sao phân tán. Trong bản sao phân tán, nếu một tiến trình nào đó ghi nhật ký nhận được bản tin thì chắc chắn tiến trình gửi cũng sẽ phải ghi nhật ký gửi bản tin đó, nếu có lỗi xảy ra thì các tiến trình này biết được sẽ phải bắt đầu từ đâu để phục hồi hệ thống.

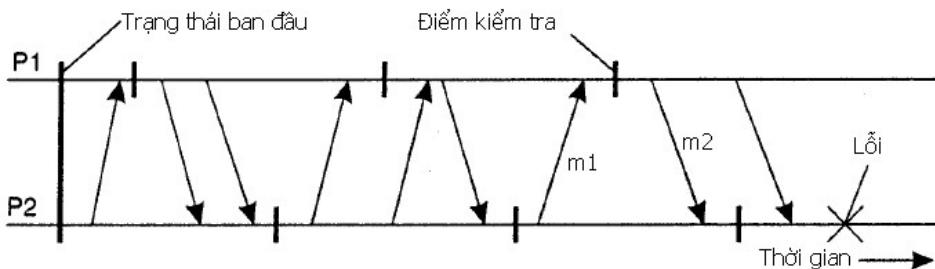


Hình 7.12 Phục hồi lùi sử dụng trạng thái toàn cục nhất quán

Hình 7.12 minh họa nguyên lý phục hồi lùi, mỗi tiến trình đều đặn ghi trạng thái của nó vào bộ nhớ cục bộ vĩnh cửu. Để phục hồi sau khi xảy ra lỗi, cần thiết phải dò tìm trạng thái nhất quán toàn cục từ những trạng thái cục bộ này, tốt nhất là phục hồi về bản sao phân tán gần nhất và gọi là đường phục hồi, nói cách khác đường phục hồi tương ứng với tập các điểm kiểm tra nhất quán gần nhất.

#### 7.4.2.1 Điểm kiểm tra độc lập

Việc thực hiện điểm kiểm tra trong các hệ thống phân tán thông thường mang tính chất cục bộ, mỗi tiến trình đều đặn ghi trạng thái cục bộ của theo cách riêng biệt, điều đó sẽ gây khó khăn cho việc xác định đường phục hồi. Để tìm ra đường phục hồi mỗi tiến trình sẽ phải quay lui về trạng thái gần nhất đã được ghi lại, nếu các trạng thái cục bộ này không tạo nên được đường phục hồi thì tiếp tục phải quay lui về trạng thái trước nữa, quá trình phục hồi phân tầng này có thể dẫn đến hiện tượng phải chạy lại từ đầu.



Hình 7.13 Điểm kiểm tra độc lập

Hình 7.13 cho thấy, khi tiến trình  $P_2$  gặp lỗi thì cần phải phục hồi trạng thái của nó về điểm kiểm tra gần nhất, tiến trình  $P_1$  cũng sẽ phải quay lui về trạng thái của điểm kiểm tra gần nhất. Rất tiếc hai trạng thái gần nhất được lưu cục bộ này không hình thành trạng thái toàn cục nhất quán, tiến trình  $P_2$  ghi nhật ký đã nhận được thông điệp  $m_2$  nhưng nhật ký của tiến trình  $P_1$  lại không ghi nhận đã gửi thông điệp  $m_2$ . Như vậy tiến trình  $P_2$  lại phải quay lui về trạng thái trước nữa, nhưng ở trạng thái này lại không nhận đã gửi thông điệp  $m_1$  trong khi nhật ký của tiến trình  $P_1$  cho thấy đã nhận được thông điệp  $m_1$ , tiến trình  $P_1$  lại quay lui về trạng thái trước nữa, cứ như vậy cả hai tiến trình phải quay lui về trạng thái ban đầu mới tìm thấy đường phục hồi.

Các tiến trình thực hiện điểm kiểm tra cục bộ độc lập với nhau gọi là phương pháp điểm kiểm tra độc lập, một giải pháp khác là phối hợp điểm kiểm tra toàn cục, nó đòi hỏi phải có sự đồng bộ trên tất cả các tiến trình và như vậy sẽ làm suy giảm hiệu năng hệ thống. Một nhược điểm khác nữa của điểm kiểm tra độc lập còn thể hiện khi dọn dẹp không gian lưu trữ cục bộ, tuy nhiên nhược điểm chính vẫn là vấn đề xác định đường phục hồi. Cài đặt điểm kiểm tra độc lập đòi hỏi phải ghi lại những phụ thuộc để các tiến trình có thể cùng nhau quay lui về trạng thái toàn cục nhất quán.

Giả sử  $CP_i(m)$  là điểm kiểm tra thứ  $m$  của tiến trình  $P_i$ ,  $INT_i(m)$  là khoảng thời gian giữa hai điểm kiểm tra  $CP_i(m-1)$  và  $CP_i(m)$ . Khi tiến trình  $P_i$  gửi thông điệp trong khoảng thời gian  $INT_i(m)$ , nó gắn kèm thông điệp cặp  $(i,m)$ . Tiến trình  $P_j$  nhận được

thông điệp với cặp đính kèm (i,m) trong khoảng thời gian INT<sub>j</sub>(n), nó ghi nhận phụ thuộc INT<sub>i</sub>(m) → INT<sub>j</sub>(n), khi thực hiện điểm kiểm tra CP<sub>j</sub>(n) nó cũng ghi nhận ký sự phụ thuộc này. Khi tiến trình P<sub>i</sub> quay lui về điểm kiểm tra CP<sub>i</sub>(m-1) thì cần phải đảm bảo rằng tất cả các tiến trình đã nhận được thông điệp từ P<sub>i</sub> trong khoảng thời gian INT<sub>i</sub>(m) cũng phải quay lui về điểm kiểm tra trước khi nhận được các thông điệp đó. Như vậy tiến trình P<sub>j</sub> sẽ phải quay lui về điểm kiểm tra CP<sub>j</sub>(n-1), nếu vẫn chưa đạt được trạng thái nhất quán toàn cục thì tiếp tục phải quay lui về điểm kiểm tra trước nữa.

Việc tính toán đường phục hồi đòi hỏi phải phân tích những phụ thuộc đã ghi nhận trong mỗi tiến trình khi thực hiện điểm kiểm tra, không cần phải đi sâu thêm về vấn đề này cũng thấy công việc này phức tạp thế nào và điều đó cho thấy sự cần thiết phải thực hiện điểm kiểm tra phối hợp. Vì vậy, kỹ thuật thực hiện điểm kiểm tra phối hợp đã được ứng dụng rộng rãi, đặc biệt đối với các hệ thống có quy mô lớn.

#### 7.4.2.2 Điểm kiểm tra phối hợp

Trong kỹ thuật điểm kiểm tra phối hợp, tất cả các tiến trình đồng bộ với nhau để cùng thực hiện việc ghi trạng thái vào bộ nhớ lưu trữ ổn định cục bộ, ưu điểm chính của nó là các trạng thái cục bộ đã tự hình thành tính nhất quán toàn cục, như vậy sẽ tránh được hiệu ứng quay lui dây chuyền. Giải thuật ảnh chụp phân tán được sử dụng để thực hiện điểm kiểm tra phối hợp, nó là ví dụ về việc phối hợp điểm kiểm tra không phong tỏa.

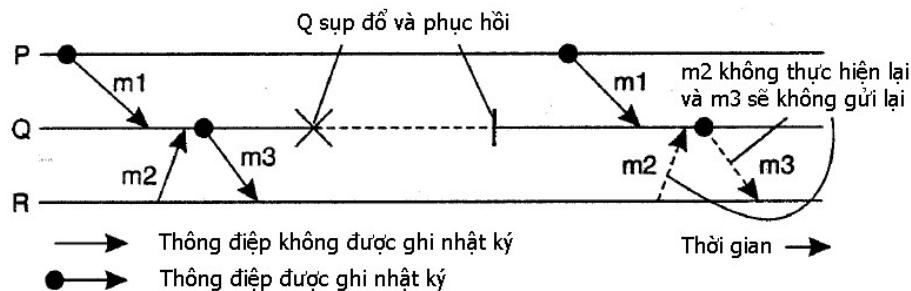
Giải pháp đơn giản hơn là sử dụng giao thức phong tỏa hai pha, tiến trình điều phối gửi thông điệp yêu cầu thực hiện điểm kiểm tra CHECKPOINT\_REQUEST đến tất cả các tiến trình, nhận được thông điệp này các tiến trình sẽ thực hiện điểm kiểm tra cục bộ và gửi thông điệp xác nhận đã hoàn thành cho tiến trình điều phối, tất cả các thông điệp nhận được sau đó hoặc những thông điệp cần gửi đều chuyển vào hàng đợi cho đến khi nhận được thông điệp hoàn thành điểm kiểm tra CHECKPOINT\_DONE từ tiến trình điều phối. Sau khi nhận được thông điệp xác nhận từ tất cả các tiến trình đã hoàn thành thực hiện điểm kiểm tra, tiến trình điều phối sẽ gửi thông điệp CHECKPOINT\_DONE đến tất cả các tiến trình thành viên để các tiến trình đó có thể tiếp tục thực hiện công việc của mình.

Có thể cài tiến giải thuật này bằng cách chỉ gửi thông điệp yêu cầu điểm kiểm tra đến các tiến trình phụ thuộc vào phục hồi của thành phần điều phối, bỏ qua các tiến trình khác. Một tiến trình được gọi là phụ thuộc khi nó nhận được thông điệp trực tiếp hoặc gián tiếp có quan hệ nhân quả với thông điệp đã nhận được từ tiến trình điều phối kể từ khi thực hiện điểm kiểm tra cuối cùng, điều này dẫn tới khái niệm ảnh lũy tiến. Để thực hiện ảnh lũy tiến, tiến trình điều phối chỉ gửi thông điệp CHECKPOINT\_REQUEST đến các tiến trình đã nhận được thông điệp của nó từ thời điểm thực hiện điểm kiểm tra cuối cùng, tiến trình P<sub>i</sub> nhận được thông điệp này sẽ làm tương tự bằng cách gửi thông điệp CHECKPOINT\_REQUEST đến các tiến trình khác có liên quan đến nó, quá trình này chỉ được phép thực hiện một lần. Sau khi các tiến trình đã hoàn thành thực hiện điểm kiểm tra cục bộ, tiến trình điều phối sẽ gửi thông điệp thứ hai CHECKPOINT\_DONE để các tiến trình có liên quan tiếp tục thực hiện công việc của mình.

### 7.4.3 Ghi nhật ký thông điệp

Thực hiện điểm kiểm tra đòi hỏi ghi trạng thái vào bộ nhớ ổn định, vì vậy cần giảm số lần thực hiện điểm kiểm tra nhưng vẫn phải đảm bảo khả năng phục hồi, một trong những kỹ thuật quan trọng được áp dụng trong hệ thống phân tán là ghi nhật ký thông điệp. Ý tưởng cơ bản của kỹ thuật này dựa trên giả thiết nếu thông điệp được gửi lại thì có thể đạt được trạng thái nhất quán toàn cục mà không cần phục hồi về trạng thái lấy từ bộ nhớ ổn định, thay vào đó trạng thái của điểm kiểm tra được lấy từ điểm khởi đầu và tất cả các thông điệp từ điểm khởi đầu đó sẽ được gửi lại và xử lý phù hợp. Cách tiếp cận này hoạt động tốt với giả thiết dựa trên mô hình xác định từng phần, việc thực thi của mỗi tiến trình là một chuỗi các khoảng sự kiện, khoảng sự kiện bắt đầu từ thời điểm xảy ra sự kiện không xác định và kết thúc tại thời điểm của sự kiện cuối cùng trước khi xảy ra sự kiện không xác định kế tiếp. Nếu thực hiện lại các khoảng sự kiện này sẽ được kết quả tương ứng với kết quả của những sự kiện không xác định, như vậy việc ghi lại tất cả các sự kiện không xác định chuyển thành có thể lặp lại toàn bộ những thực thi của tiến trình theo cách xác định.

Việc ghi nhật ký thông điệp là cần thiết để phục hồi tiến trình sụp đổ và như vậy sẽ phục hồi được trạng thái toàn cục nhất quán, điều quan trọng là phải biết chính xác khi nào các thông điệp cần được ghi lại. Alvisi và Marzullo đã mô tả nhiều lược đồ ghi nhật ký thông điệp, mỗi thông điệp chứa thông tin điều khiển để có thể được gửi lại và xử lý chính xác. Thông tin điều khiển xác định bên gửi và bên nhận, số tuần tự để tránh trùng lặp, số thứ tự phân phát để quyết định chính xác khi nào sẽ được xử lý ở bên nhận. Một thông điệp được coi là ổn định nếu nó không bị thất lạc trên kênh truyền vì nó sẽ được ghi vào bộ nhớ ổn định, những thông điệp đó sẽ được sử dụng để gửi lại khi cần thực hiện phục hồi. Khi một tiến trình nào bị sụp đổ, tiến trình đó sẽ phải được phục hồi, ngay cả khi phục hồi thành công cũng có thể dẫn tới hiện tượng tiến trình mò côi, đó là những tiến trình sống sót sau sự sụp đổ của tiến trình khác nhưng trạng thái của nó không nhất quán với tiến trình sụp đổ và đã được phục hồi, để làm rõ vấn đề này cần phải phân tích chi tiết về quá trình ghi nhật ký thông điệp.



Hình 7.14 Không nhất quán sau khi phục hồi

Ví dụ trên hình 7.14, tiến trình Q nhận thông điệp  $m_1$  từ tiến trình P và  $m_2$  từ tiến trình R, thông điệp  $m_1$  có ghi nhật ký trong khi thông điệp  $m_2$  không được ghi nhật ký, sau đó Q lại gửi thông điệp  $m_3$  cho tiến trình R và thông điệp này được ghi vào nhật ký. Sau khi gửi thông điệp  $m_3$ , tiến trình Q sụp đổ do đó nó phải thực hiện quá trình phục hồi, tiến trình Q lấy thông điệp từ nhật ký và phục hồi thành công thông điệp  $m_1$  nhưng không

thể phục hồi thông điệp  $m_2$  vì thông điệp này không được ghi trong nhật ký. Vì thông điệp  $m_3$  phụ thuộc nhân quả vào thông điệp  $m_2$ ,  $m_2$  không được gửi lại thì Q cũng sẽ không thực hiện gửi thông điệp  $m_3$ . Tiến trình R có bản sao thông điệp  $m_3$  nhưng thông điệp này lại không có trong tiến trình Q sau khi phục hồi và như vậy trạng thái của tiến trình Q không nhất quán với tiến trình R, khi đó R được gọi là tiến trình mồ côi.

Một cách tổng quát, mỗi thông điệp  $m$  dẫn tới tập  $DEP(m)$  các tiến trình phụ thuộc vào sự phân phát thông điệp  $m$ , nếu một thông điệp  $m'$  phụ thuộc nhân quả với thông điệp  $m$  và thông điệp  $m'$  được phân phát đến bất kỳ tiến trình nào thì tiến trình đó cũng sẽ là thành viên của tập  $DEP(m)$ . Thông điệp  $m'$  phụ thuộc nhân quả vào  $m$  nếu nó được gửi bởi một tiến trình trước đó đã phân phát thông điệp  $m$ , hoặc  $m'$  thuộc tiến trình đã phân phát thông điệp khác nhưng thông điệp đó phụ thuộc nhân quả vào thông điệp  $m$ . Tập  $COPY(m)$  gồm các tiến trình nhận được bản sao thông điệp  $m$  nhưng chưa kịp ghi vào vùng nhớ ổn định. Khi một tiến trình chuyển tiếp thông điệp  $m$  thì nó cũng là thành viên của  $COPY(m)$ , nếu tất cả các tiến trình này sụp đổ thì rõ ràng việc phát lại thông điệp  $m$  sẽ không thể thực hiện được. Tiến trình mồ côi xuất hiện khi nó phụ thuộc vào thông điệp nhưng không có cách nào phát lại thông điệp đó.

Để tránh hiện tượng tiến trình mồ côi thì cần phải chắc chắn rằng mỗi tiến trình trong tập  $COPY(m)$  sụp đổ thì không ở trong tập  $DEP(m)$  của tiến trình sống sót, nghĩa là tất cả các tiến trình trong  $DEP(m)$  cũng phải bị sụp đổ. Điều kiện này có thể được thực hiện nếu đảm bảo rằng mọi tiến trình là thành viên của  $DEP(m)$  thì cũng là thành viên của  $COPY(m)$ , nói cách khác nếu một tiến trình phụ thuộc thông điệp  $m$  thì nó phải giữ bản sao của thông điệp  $m$ . Về cơ bản, có thể giải quyết vấn đề tiến trình mồ côi bằng cách sử dụng giao thức ghi nhật ký bi quan hoặc lạc quan.

Giao thức bi quan đảm bảo mỗi thông điệp chưa ổn định thì chỉ có nhiều nhất một tiến trình phụ thuộc vào nó, như vậy tiến trình sẽ chỉ được phép gửi thông điệp khi tất cả các thông điệp đã được ghi vào bộ nhớ ổn định, nghĩa là tiến trình đã thuộc tập  $COPY(m)$ . Ngược lại, giao thức ghi nhật ký lạc quan, công việc thực tế được thực hiện sau khi xảy ra sụp đổ, tiến trình mồ côi trong tập  $DEP(m)$  sẽ quay lui cho đến khi không thuộc tập này nữa, như vậy nó phải lưu vết các phụ thuộc và vì vậy sẽ làm phức tạp hóa vấn đề cài đặt. So sánh hai cách tiếp cận trên có thể thấy cách tiếp cận bi quan đơn giản hơn cách tiếp cận lạc quan rất nhiều, vì vậy trong thực tế người ta thường sử dụng cách tiếp cận này.

## THẢO LUẬN

1. Liệt kê một số ví dụ về lỗi trên các máy tính.
2. Hiện tượng khóa chết thuộc loại lỗi nào?
3. Nêu giải pháp phát hiện nhân viên quản trị hệ thống đã gian lận bằng cách cập nhật dữ liệu bằng câu lệnh SQL.
4. Trình bày giải pháp cấu hình dự phòng nóng cho các máy chủ.
5. Nêu giải pháp xử lý khi gặp lỗi gọi đối tượng từ xa.

6. Trình bày giải pháp truyền thông nhân quả.
7. Xây dựng ứng dụng minh họa giao tác phân tán.
8. Xây dựng ứng dụng sao lưu dữ liệu tự động cho các cơ sở dữ liệu phân tán.
9. Tìm hiểu các chế độ sao lưu dữ liệu trong hệ quản trị cơ sở dữ liệu Oracle.
10. Tại sao tất cả các hệ quản trị cơ sở dữ liệu đều có tập tin nhật ký?

## CHƯƠNG 8: NHẤT QUÁN VÀ NHÂN BẢN

Nhân bản dữ liệu là kỹ thuật đã được áp dụng từ lâu để để tăng tính đáng tin cậy hoặc hiệu năng hệ thống, khi dữ liệu bị lỗi hay vì một nguyên nhân nào đó mà không thể dùng được thì có thể sử dụng bản sao dữ liệu để không gián đoạn dịch vụ. Nhân bản dữ liệu cũng nhằm mục đích tăng hiệu năng của hệ thống và do đó có thể mở rộng quy mô cả về số lượng người dùng lẫn phạm vi địa lý. Nhiều tiến trình truy nhập đồng thời đến cơ sở dữ liệu, nếu dữ liệu được nhân bản trên các máy tính khác nhau thì có thể phân tải truy nhập đến các bản sao, số lượng yêu cầu đến mỗi bản sao nhỏ hơn không những giảm thời gian xử lý mà còn giảm khả năng xung đột. Tuy nhiên, sự tồn tại nhiều bản sao sẽ nảy sinh vấn đề về tính nhất quán, nghĩa là khi một bản sao được cập nhật thì các bản sao khác cũng phải cập nhật theo, nếu không sẽ vi phạm tính nhất quán. Khi nhân bản dữ liệu, Eric Brewer đã chứng minh không thể đáp ứng cả ba yêu cầu về tính nhất quán, tính sẵn sàng và khả năng chịu lỗi.

Chương này bắt đầu bằng cách thảo luận lợi ích của việc nhân bản và mối liên quan của nó tới qui mô của hệ thống phân tán, sau đó sẽ tập trung phân tích tính nhất quán. Vấn đề quan trọng trong các hệ thống phân tán là các tiến trình đồng thời truy nhập đến dữ liệu chia sẻ, tính nhất quán trong những trường hợp này thể hiện những gì tiến trình mong đợi khi đọc và cập nhật dữ liệu chia sẻ trong khi các tiến trình khác cũng đang truy nhập dữ liệu đó. Các mô hình nhất quán cho những dữ liệu chia sẻ thường khó cài đặt hiệu quả trong những hệ thống phân tán qui mô lớn, trong nhiều trường hợp, các mô hình đơn giản có thể sẽ được sử dụng để dễ dàng cài đặt hơn. Ví dụ, các mô hình lấy máy khách làm trung tâm tập trung đảm bảo tính nhất quán dưới góc độ của các máy khách.

Định nghĩa tính nhất quán mới chỉ là một nửa của vấn đề, cần phải xem xét cách cài đặt thực tế như thế nào, do đó phải giải quyết hai vấn đề cản bản trong nhân bản. Thứ nhất là vấn đề quản lý, không phải chỉ quản lý vị trí lưu trữ mà còn phải giải quyết phương pháp phân tán nội dung đến các bản sao. Vấn đề thứ hai là duy trì tính nhất quán của các bản sao, trong nhiều trường hợp các ứng dụng yêu cầu dạng nhất quán mạnh, nghĩa là các thao tác cập nhật phải được lan tỏa ngay lập tức đến các bản sao, nhiều giải pháp cũng sẽ được đề cập đến để thực hiện cài đặt thực tế các mô hình nhất quán.

### 8.1 Giới thiệu chung

Nhân bản dữ liệu là quá trình tạo và duy trì nhiều bản sao của cùng một dữ liệu ở nhiều vị trí khác nhau như một cách đảm bảo tính sẵn sàng, tính đáng tin cậy và khả năng phục hồi của dữ liệu trong hệ thống. Khi có nhiều bản sao của cùng một dữ liệu ở những vị trí khác nhau, ngay cả khi một bản sao không thể truy cập được do máy chủ sập hoặc mất kết nối mạng, bản sao khác có thể được sử dụng để giảm thiểu thời gian ngừng cung cấp dịch vụ.

Có hai lý do phải nhân bản dữ liệu, đó là tăng khả năng chịu lỗi và cải thiện hiệu năng cho hệ thống. Với lý do thứ nhất, nếu một bản sao bị hỏng thì hệ thống vẫn có thể hoạt động sau khi kết nối đến bản sao khác, bằng cách duy trì nhiều bản sao có thể bảo vệ

dữ liệu tốt hơn để phòng khi xảy ra những sự cố làm mất dữ liệu. Ví dụ, nếu có ba bản sao dữ liệu, mọi thao tác đọc và ghi đều thực hiện trên các bản sao này thì có thể tự bảo vệ khi một thao tác ghi bị lỗi bằng cách lấy giá trị của hai bản sao còn lại. Một cách tổng quát, dữ liệu được lưu trên N bản sao, khi dữ liệu của một bản sao nào đó bị lỗi thì hoàn toàn có thể tin cậy dữ liệu của ít nhất  $N/2+1$  bản sao không lỗi.

Lý do thứ hai cải thiện hiệu năng khi mở rộng qui mô là hoàn toàn dễ hiểu, các máy chủ bản sao được đặt ở nhiều vị trí địa lý khác nhau và như vậy yêu cầu của máy khách không bị tập trung về một máy chủ, thậm chí có thể áp dụng kỹ thuật cân bằng tải hoặc tính toán phân tán để chia nhỏ công việc cho nhiều máy chủ thực hiện. Việc phân bố các máy chủ bản sao tại các vị trí địa lý khác nhau sẽ tạo điều kiện cho máy khách lựa chọn máy chủ thích hợp nhất để xử lý, tiết kiệm thời gian truy nhập và đồng thời giảm lượng dữ liệu lưu chuyển trên mạng.

Nhân bản mang lại lợi ích cho tính tin cậy và hiệu năng nhưng lại phải đổi mới với vấn đề nhất quán, khi một bản sao bị thay đổi thì nội dung của nó sẽ khác với các bản sao còn lại. Như vậy, cần phải đảm bảo rằng những thay đổi đều sẽ phải thực hiện trên tất cả các bản sao, vấn đề nằm ở chỗ thành phần nào và khi nào thực hiện những thay đổi này. Để minh họa vấn đề này có thể lấy trường hợp trang web làm ví dụ, trình duyệt trên máy khách gửi yêu cầu đến máy chủ web, máy chủ web truy vấn cơ sở dữ liệu sau đó tạo dữ liệu định dạng html trả về cho trình duyệt, thời gian thực hiện hai công việc này có thể lên tới vài giây. Để giảm thời gian trễ, máy chủ web thường lưu cục bộ bản sao của trang web gọi là cache, nếu nhận được yêu cầu trang web lần nữa thì nó không cần phải truy nhập cơ sở dữ liệu, chỉ cần lấy nội dung đã được lưu trong cache, do đó thời gian xử lý chỉ vài chục nano giây, như vậy thời gian phản hồi chỉ là độ trễ của kênh truyền mạng.

Giảm thời gian chờ đợi của người sử dụng chỉ là biểu hiện bên ngoài, xét cho cùng thì người sử dụng khó cảm nhận được chất lượng dịch vụ nếu thời gian đáp ứng được cải thiện vài trăm mili giây, mục đích chính nằm ở qui mô hệ thống. Nếu giữ nguyên tiêu chuẩn về thời gian đáp ứng, giảm thời gian xử lý trên máy chủ x% đồng nghĩa với việc tăng thêm  $x/(1-x)$  số yêu cầu đồng thời truy nhập.

Tuy nhiên, nếu người sử dụng muốn phiên bản mới nhất của trang thì thật là không may mắn, vấn đề nằm ở chỗ trong khoảng thời gian giữa hai lần truy nhập trang web đã bị thay đổi nhưng những thay đổi đó vẫn chưa lan tỏa đến các bản sao đã lưu trên cache của máy chủ, kết quả là người sử dụng sẽ nhìn thấy nội dung cũ. Giải pháp khắc phục vấn đề này là đặt thời gian quá hạn cho mỗi trang web, quá thời hạn này sẽ làm tươi cache.

Kỹ thuật cache cũng có thể áp dụng cho trình duyệt trên máy khách, tuy nhiên nó chỉ có tác dụng đối với người sử dụng trên chính máy tính đó. Một giải pháp cho vấn đề bản sao cũ là cấm trình duyệt giữ các bản sao ở vị trí đầu tiên, thực sự cho phép máy chủ hoàn toàn phụ trách việc nhân bản, tuy nhiên giải pháp này vẫn có thể dẫn đến thời gian truy nhập kém nếu như không có bản sao nào gần với người sử dụng. Một giải pháp khác là cho phép máy chủ làm mát hiệu lực hoặc cập nhật các bản sao nhưng nhưng thế đòi hỏi máy chủ phải lưu vết tắt cả các bộ nhớ cache và gửi thông điệp cho chúng, kết quả là hiệu năng tổng thể của máy chủ sẽ bị suy giảm nghiêm trọng.

Nhân bản và lưu bản sao cache để nâng cao hiệu năng đã được áp dụng phổ biến như một kỹ thuật mở rộng qui mô hệ thống, việc đặt các bản sao dữ liệu gần với các tiến trình sử dụng chúng có thể cải thiện hiệu năng bằng cách giảm thời gian truy nhập và như vậy giải quyết được các vấn đề về qui mô hệ thống. Cân đối giữa việc duy trì nội dung mới nhất với yêu cầu băng thông mạng cần phải được xem xét kỹ lưỡng, ví dụ tiến trình P đọc bản sao N lần/s và bản sao được cập nhật M lần/s, nếu N nhỏ hơn M rất nhiều, nghĩa là tỉ lệ đọc/ghi rất thấp thì có thể thấy kết quả của nhiều lần cập nhật đã không được tiến trình P sử dụng, làm cho truyền thông mạng cho những phiên bản này trở nên vô ích. Trong trường hợp này nên cài đặt bản sao gần với tiến trình P hoặc áp dụng chiến lược khác để cập nhật bản sao, vấn đề này sẽ được đề cập chi tiết trong phần sau.

Tuy nhiên, nghiêm trọng hơn ở chỗ việc duy trì nhiều bản sao nhất quán cũng có thể là chủ đề cho các vấn đề về qui mô, tập các bản sao được coi là nhất quán khi tất cả các bản sao đều giống nhau, nghĩa là thao tác đọc tại bất kỳ bản sao nào cũng trả về kết quả như nhau, kết quả là thao tác cập nhật được thực hiện trên một bản sao sẽ phải được lan tỏa đến tất cả các bản sao trước khi thao tác kế tiếp thực hiện, không phân biệt bản sao khởi sướng hay thực hiện thao tác, nhất quán loại này gọi là chật chẽ và cũng được gọi là nhân bản đồng bộ. Tư tưởng chính là thao tác cập nhật được thực hiện trên các bản sao như một thao tác nguyên tử hay còn gọi là giao tác phân tán, đáng tiếc việc cài đặt tính nguyên tử liên quan đến lượng lớn các bản sao có thể bị phân tán rải rác trên các mạng qui mô lớn vốn rất khó thực hiện khi các thao tác cần phải được hoàn thành trong thời gian ngắn nhất.

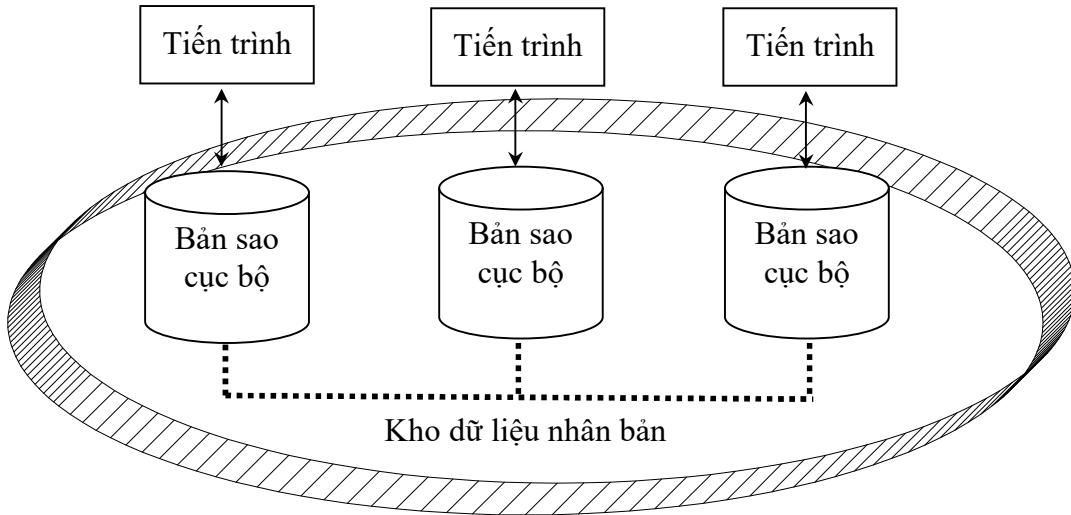
Khó khăn bắt nguồn từ thực tế cần phải đồng bộ tất cả các bản sao, về mặt nguyên tắc, các bản sao phải đạt được sự đồng thuận trên vấn đề chính xác khi nào thao tác cập nhật sẽ được thực hiện cục bộ, các bản sao có thể phải quyết định thứ tự toàn cục của các thao tác dựa trên nhãn thời gian Lamport hoặc cho phép tiến trình điều phối qui định thứ tự thực hiện. Đồng bộ toàn cục chiếm lượng lớn thời gian truyền thông, đặc biệt khi các bản sao trải dài trên mạng diện rộng. Chúng ta đổi mặt với tình thế tiến thoái lưỡng nan, một mặt có thể làm giảm bớt các vấn đề về hiệu năng khi mở rộng qui mô bằng cách áp dụng nhân bản và bộ nhớ cache, mặt khác giữ cho tất cả các bản sao nhất quán toàn cục đòi hỏi đồng bộ toàn cục mà điều này thì lại kéo theo chi phí về hiệu năng.

Trong nhiều trường hợp, chỉ có cách nói lỏng các ràng buộc nhất quán, nếu giảm bớt yêu cầu cập nhật phải được thực hiện như các thao tác nguyên tử mới có thể tránh được đồng bộ toàn cục và như thế mới tăng hiệu năng, cái giá phải trả đó là các bản sao có thể không phải lúc nào cũng giống nhau. Như vậy, nhất quán phải được nói lỏng đến mức nào sẽ phụ thuộc rất lớn vào việc truy nhập và các dạng cập nhật của dữ liệu nhân bản cũng như mục đích sử dụng những dữ liệu đó. Trong những phần sau, trước hết sẽ đề cập tới các mô hình nhất quán bằng cách định nghĩa chính xác nhất quán thực sự là gì, sau đó sẽ thảo luận những cách khác nhau để cài đặt các mô hình nhất quán.

## 8.2 Các mô hình nhất quán lấy dữ liệu làm trung tâm

Khái niệm nhất quán ở đây được hiểu trong ngữ cảnh các thao tác đọc/ghi dữ liệu, cho dù đó là dữ liệu được lưu trong bộ nhớ, trên tập tin hay trong hệ quản trị cơ sở dữ

liệu đều gọi chung là kho dữ liệu. Hình 8.1 thể hiện kho dữ liệu được nhân bản trên nhiều máy chủ, mỗi tiến trình có thể truy nhập dữ liệu trong kho được coi là có bản sao cục bộ sẵn có của toàn bộ kho dữ liệu. Thao tác làm thay đổi dữ liệu được xếp vào loại thao tác ghi và tiềm ẩn nguy cơ xung đột, nếu không thay đổi thì xếp vào loại thao tác đọc.



Hình 8.1 Tổ chức kho dữ liệu nhân bản

Mô hình nhất quán, về bản chất đó là thỏa thuận giữa tiến trình và kho dữ liệu, các tiến trình phải tuân thủ một số qui tắc để đảm bảo tính chính xác của dữ liệu. Thông thường, tiến trình thực hiện thao tác đọc mục dữ liệu và hy vọng sẽ nhận được giá trị phản ánh kết quả ghi cuối cùng đã thực hiện trên mục dữ liệu đó. Vì không tồn tại đồng hồ toàn cục nên khó có thể xác định được thao tác ghi nào là thao tác được thực hiện cuối cùng. Để thay thế, chúng ta phải định nghĩa một số mô hình nhất quán, mỗi mô hình sẽ hạn chế khả năng đọc giá trị của các mục dữ liệu, nói cách khác sẽ không có mô hình nhất quán toàn diện.

### 8.2.1 *Nhất quán liên tục*

Như đã đề cập trong trên, hiện nay chưa có một giải pháp nhân bản nào được coi là giải pháp tốt nhất, nhân bản dữ liệu đặt ra những vấn đề liên quan tới tính nhất quán và do đó chưa có một phương pháp tổng quát nào giải quyết hiệu quả vấn đề này. Chỉ khi nói lỏng tính nhất quán thì mới hy vọng đạt được giải pháp hiệu quả, đáng tiếc lại không có một qui tắc chung nào để nói lỏng tính nhất quán, điều này hoàn toàn dựa vào yêu cầu khi xây dựng hệ thống.

Thông thường, có ba cách tiếp cận dựa trên ba trực độc lập để xác định tính không nhất quán: độ lệch giá trị số giữa các bản sao, chênh lệch về thời gian cập nhật giữa các bản sao và khác biệt về thứ tự của các thao tác cập nhật, những sai lệch này là hình thành các phạm vi nhất quán liên tục. Với dữ liệu kiểu số, có thể xác định tính không nhất quán dựa trên sai số tương đối hoặc tuyệt đối, ví dụ giá trị cổ phiếu không được lệch quá 0.01 VND là sai số tuyệt đối hoặc giá trị không lệch quá 1% là sai số tương đối, nếu nằm trong phạm vi cho phép thì vẫn có thể coi các bản sao nhất quán. Độ lệch về giá trị số cũng có thể được hiểu là số lượng các yêu cầu cập nhật đã được áp dụng cho một bản sao nhưng

chưa thể hiện kết quả cho các bản sao khác, ví dụ bộ nhớ cache chưa nhìn thấy những thao tác máy chủ web xử lý một lô các câu lệnh cập nhật, độ lệch trong trường hợp này gọi là trọng số.

Chênh lệch về thời gian thực hiện liên quan đến lần cập nhật cuối cùng, một số ứng dụng chấp nhận bản sao cung cấp dữ liệu không quá cũ. Ví dụ hệ thống quan trắc thời tiết, bản tin thời tiết thường có hiệu lực trong hàng giờ, như vậy chỉ cần thường xuyên cập nhật dữ liệu quan trắc tại mỗi bản sao nhưng chúng chỉ được lan tỏa đến các bản sao khác sau mỗi 60 phút. Cuối cùng là sự khác biệt về thứ tự thực hiện các thao tác cập nhật trên các bản sao, mỗi bản sao có thể tạm thời thực hiện theo cách riêng của mình trong khi chờ sự đồng thuận của tất cả các thành viên, như vậy một số thao tác cập nhật có thể phải phục hồi lại và áp dụng theo một thứ tự khác trước khi được ghi nhận thay đổi vĩnh viễn.

Để xác định tính không nhất quán, Yu và Vahdat đưa ra khái niệm đơn vị nhất quán viết tắt là CONIT. Mỗi đơn vị nhất quán dùng để đo tính nhất quán cho một đối tượng nào đó, ví dụ giá trị cổ phiếu trên sàn giao dịch chứng khoán hay nội dung một bản báo cáo quan trắc thời tiết. Ví dụ minh họa thể hiện trên hình 8.2, mỗi bản sao duy trì một đồng hồ vector hai phần tử, phần tử thứ nhất thể hiện nhãn thời gian của bản sao A và phần tử thứ 2 thể hiện nhãn thời gian của bản sao B, sử dụng ký hiệu  $\langle t, i \rangle$  là thao tác tiến trình i thực hiện tại thời gian logic t.

Bản sao A		Bản sao B	
Đơn vị nhất quán		Đơn vị nhất quán	
x=6	y=2	x=2	y=5
Thao tác	Kết quả	Thao tác	Kết quả
$\langle 5, B \rangle$	$x := x + 2$	$\langle 5, B \rangle$	$x := x + 2$
$\langle 8, A \rangle$	$y := y + 2$	$\langle 8, A \rangle$	$y := y + 2$
$\langle 12, A \rangle$	$y := y + 1$	$\langle 12, A \rangle$	$y := y + 1$
$\langle 14, A \rangle$	$x := y * 2$	$\langle 14, A \rangle$	$x := y * 2$

Đồng hồ vector A: (15, 5)  
Độ lệch số lượng thao tác: 3  
Độ lệch giá trị số: (1, 5)

Đồng hồ vector B: (0, 11)  
Độ lệch số lượng thao tác: 2  
Độ lệch giá trị số: (3, 6)

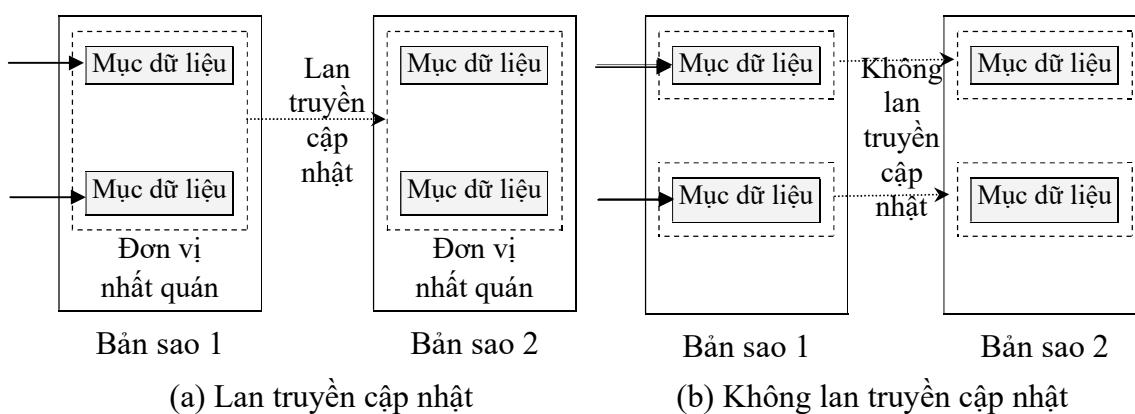
Hình 8.2 Tính nhất quán với điều kiện nối lỏng thứ tự thao tác

Trong ví dụ này, hai bản sao thao tác trên các mục dữ liệu x và y, giả thiết ban đầu chúng đều có giá trị bằng 0. Bản sao A nhận được thao tác cập nhật  $\langle 5, B \rangle$   $x := x + 2$  từ bản sao B, thao tác ghi dữ liệu vào x đã được lưu vào vùng nhớ vĩnh cửu và không thể quay lui. Bản sao A dự kiến thực hiện ba thao tác thay đổi dữ liệu  $\langle 8, A \rangle$ ,  $\langle 12, A \rangle$  và  $\langle 14, A \rangle$ , như vậy đồng hồ vector trên bản sao A sau các thao tác trên sẽ là (15,5) và độ lệch số lượng thao tác là 3. Tương tự, bản sao B dự kiến thực hiện thao tác  $\langle 10, B \rangle$ , sau các thao tác vẫn chưa nhìn thấy các thao tác từ bản sao A nên đồng hồ vector trên bản sao B có giá

trị là  $(0, 11)$ , như vậy độ lệch số lượng thao tác là 1. Từ góc độ bên ngoài nhìn vào hệ thống thì chỉ có thao tác  $<5,B> x:=x+2$  đã được thực hiện trên bản sao A, tất cả các lệnh khác mới chỉ ghi nhận tạm thời trên các bản sao cục bộ và chúng đang chờ được cam kết phân tán, do đó giá trị của  $(x, y)$  sẽ là  $(0, 0)$ .

Xét về độ lệch giá trị số trên các bản sao, biểu diễn đơn vị nhất quán R là cặp  $(u, v)$  trong đó  $u$  là số lượng thao tác đang chờ cam kết và  $v$  là tổng sai số về giá trị của các tham số, khi đó đơn vị nhất quán của bản sao B so với A sẽ là  $(3, 6)$  và đơn vị nhất quán của A so với B sẽ là  $(1, 5)$ . Như vậy giữa các bản sao cần phải thiết lập một kênh truyền thông riêng biệt để nhận biết tính không nhất quán, nếu độ lệch số lượng thao tác thực hiện trên các bản sao và sai số về giá trị không vượt quá ngưỡng cho phép thì có thể coi như đã đáp ứng yêu cầu nhân bản.

Cần phải cân nhắc khi lựa chọn đơn vị nhất quán, nếu đơn vị nhất quán đại diện cho một lượng lớn dữ liệu, ví dụ toàn bộ cơ sở dữ liệu, thì các cập nhật sẽ được tập hợp lại cho tất cả dữ liệu trong đơn vị nhất quán, kết quả là các bản sao sẽ sớm trở nên không nhất quán. Hình 8.3 minh họa đơn vị nhất quán, hai bản sao có thể khác nhau nếu còn tồn tại một yêu cầu cập nhật chưa được thực hiện.



Hình 8.3 Số lượng thao tác trong lan truyền cập nhật

Trường hợp (a) lựa chọn đơn vị nhất quán với độ lệch số lượng thao tác cập nhật là 2, khi thực hiện xong ở bản sao thứ nhất mới lan tỏa cập nhật tới bản sao thứ hai, điều này chỉ đúng trong trường hợp các mục dữ liệu liên quan chặt chẽ với nhau. Trường hợp (b) do chưa đủ các câu lệnh cập nhật theo qui định của đơn vị nhất quán nên tạm thời chưa lan truyền cập nhật, điều đó có thể dẫn đến hiện tượng không nhất quán. Vì đề chọn đơn vị nhất quán đặc biệt quan trọng khi các mục dữ liệu hoàn toàn độc lập với nhau, trường hợp đó có thể coi như đã chọn sai đơn vị nhất quán, cần phải chọn đơn vị nhất quán nhỏ hơn.

Tuy nhiên, lựa chọn đơn vị nhất quán rất nhỏ không phải là một ý tưởng tốt vì số lượng đơn vị nhất quán cần phải quản lý sẽ tăng lên đáng kể, lượng thông tin điều khiển nhiều hơn và điều đó dẫn đến suy giảm hiệu năng của hệ thống. Lý tưởng, một thao tác được thực hiện trên một hành tinh phải được thực hiện ngay lập tức trên các hành tinh khác, hoặc ít nhất sau khi thao tác cập nhật đã hoàn thành trên hành tinh thì phải được chuyển

ngay sang các bản sao khác. Tuy nhiên thực tế sẽ gặp nhiều trở ngại về mạng, nhất là những hệ thống thường xuyên cập nhật, vì vậy quyết định mỗi lần lan truyền cập nhật gồm bao nhiêu thao tác sẽ do người thiết kế hệ thống phần mềm quyết định.

Mặc dù có nhiều biện pháp hay để đạt được các yêu cầu nhất quán nhưng có hai vấn đề quan trọng cần phải giải quyết trong thực tế, đó là việc xây dựng các giao thức nhất quán liên tục và xác định các yêu cầu nhất quán cho các ứng dụng. Thực tế cho thấy, việc duy trì các yêu cầu này gặp nhiều trở ngại, lập trình viên nói chung không quen xử lý nhân bản., vì vậy cần phải cung cấp giao diện lập trình đơn giản và dễ hiểu. Có thể cài đặt tính nhất quán liên tục trong bộ công cụ dưới dạng thư viện liên kết với các ứng dụng, đơn vị nhất quán sẽ được khai báo bên cạnh các thao tác cập nhật dữ liệu.

### 8.2.2 *Nhất quán theo thứ tự thao tác*

Trong những năm qua đã có rất nhiều nghiên cứu về các mô hình nhất quán lấy dữ liệu làm trung tâm, trong đó có một loại mô hình quan trọng xuất phát từ lập trình song song. Bên cạnh tính nhất quán liên tục, các tiến trình tính toán phân tán và song song còn phải đương đầu với vấn đề chia sẻ tài nguyên nhưng vẫn đảm bảo tính tương tranh. Phần này sẽ đề cập tới mô hình đảm bảo tính nhất quán theo thứ tự các thao tác, chúng làm tăng tính nhất quán liên tục vì các bản sao cần phải đạt được sự đồng thuận về thứ tự của các thao tác cập nhật. Nhất quán theo thứ tự thao tác được phân làm bốn loại: nhất quán nghiêm ngặt, nhất quán tuần tự, nhất quán nhân quả và nhất quán hàng đợi.

#### 8.2.2.1 *Mô hình nhất quán nghiêm ngặt*

Mô hình nhất quán nghiêm ngặt là mô hình trong đó bất kỳ thao tác đọc trên một mục dữ liệu đều trả về giá trị tương ứng với kết quả của thao tác **ghi gần nhất** trên mục dữ liệu đó. Định nghĩa trên rất tự nhiên và rõ ràng, mặc dù nó ngầm thừa nhận sự tồn tại của thời gian toàn cục tuyệt đối, đó là thời gian chung cho cả hệ thống để xác định chính xác khái niệm thao tác ghi gần nhất, điều này là khó khả thi với hệ phân tán.

P<sub>1</sub>: W(x)a

P<sub>2</sub>:

(a) Nhất quán nghiêm ngặt

P<sub>1</sub>: W(x)a

P<sub>2</sub>:

(b) Không nhất quán nghiêm ngặt

Hình 8.4 Mô hình nhất quán nghiêm ngặt

Giả sử mục dữ liệu x ban đầu có giá trị là NIL, ký hiệu W<sub>i</sub>(x)a là thao tác tiến trình P<sub>i</sub> thực hiện ghi giá trị a lên mục dữ liệu x, R<sub>i</sub>(x)a là thao tác tiến trình P<sub>i</sub> thực hiện đọc mục dữ liệu x và cho kết quả a. Hình 8.4 minh họa tiến trình P<sub>1</sub> thực hiện ghi giá trị a vào biến x sau đó lan tỏa sang các bản sao khác. Tiến trình P<sub>2</sub> đọc biến x, nếu nhận được giá trị a thì đó đáp ứng yêu cầu mô hình nhất quán nghiêm ngặt, ngược lại sẽ không phải là nhất quán nghiêm ngặt.

Như vậy, để đảm bảo tính nhất quán của mô hình nhất quán nghiêm ngặt thì không thể thực hiện theo phương pháp cập nhật xong bản gốc rồi mới lan tỏa đến các bản sao vì sẽ cần một khoảng thời gian di chuyển thông điệp trên mạng. Mô hình nhất quán nghiêm ngặt đòi hỏi tính nguyên tử của các thao tác cập nhật, yêu cầu cập nhật phải đến được tất

cả các bản sao ngay lập tức, điều này chỉ có thể thực hiện được bằng các kỹ thuật cam kết phân tán. Cam kết phân tán sẽ làm tăng đáng kể số lượng thông điệp cần trao đổi giữa các bản sao, do đó thời gian cập nhật sẽ kéo dài, nếu có quá nhiều bản sao hoặc tần suất giao dịch lớn thì khó có thể đảm bảo các yêu cầu về hiệu năng, những hệ thống giao dịch trực tuyến không nên áp dụng mô hình này.

#### 8.2.2.2 Mô hình nhất quán tuần tự

Mô hình nhất quán tuần tự được Lamport đề xuất vào năm 1979 trong ngữ cảnh bộ nhớ dùng chung cho các hệ thống nhiều bộ xử lý, kho dữ liệu được coi là nhất quán tuần tự nếu kết quả của bất kỳ tiến trình nào cũng như nhau khi các thao tác được thực hiện theo thứ tự giống nhau. Mô hình nhất quán tuần tự không đề cập đến yếu tố thời gian, như vậy không cần phải tham chiếu đến thao tác ghi gần nhất lên mục dữ liệu, tiến trình phải nhìn thấy thao tác ghi từ tất cả các tiến trình khác chứ không phải chỉ thao tác ghi của mình. Tuy nhiên, thời gian vật lý trên các máy tính khác nhau, khi triển khai thực tế cần lưu ý tránh sử dụng các hàm phụ thuộc thời gian thực trong các câu lệnh cập nhật. Như vậy, nhất quán tuần tự là mô hình lỏng và yêu hơn mô hình nhất quán nghiêm ngặt, nó phải thỏa mãn các yêu cầu sau:

- Kết quả của thực hiện như nhau nếu thao tác đọc và ghi do các tiến trình thực hiện trên mục dữ liệu một cách tuần tự.
- Các thao tác của mỗi tiến trình xuất hiện trong chuỗi thao tác do chương trình qui định.
- Khi các tiến trình chạy đồng thời trên các máy khác nhau thì cho phép sự đan xen của các thao tác nhưng tất cả các tiến trình đều phải nhận biết giống nhau về sự đan xen của các thao tác.

$P_1: W(x)a$	$P_1: W(x)a$
$P_2: W(x)b$	$P_2: W(x)b$
$P_3: R(x)b$	$P_3: R(x)b$
$P_4: R(x)b$	$P_4: R(x)a$
(a) Nhất quán tuần tự	(b) Không nhất quán tuần tự

Hình 8.5 Mô hình nhất quán tuần tự

Hình 8.5 (a) minh họa nhất quán tuần tự, tiến trình  $P_1$  thực hiện ghi giá trị  $a$  vào biến  $x$  sau đó tiến trình  $P_2$  cũng thực hiện ghi giá trị  $b$  vào biến này, các tiến trình  $P_3$  và  $P_4$  đều đọc được giá trị  $b$  sau đó mới đọc được giá trị  $a$ , điều đó chứng tỏ thao tác ghi của tiến trình  $P_2$  được thực hiện trước thao tác ghi của tiến trình  $P_1$ . Ngược lại, hình 8.5 (b) cho thấy tiến trình  $P_3$  đọc được giá trị  $b$  trước giá trị  $a$ , trong khi đó tiến trình  $P_4$  lại đọc được giá trị  $a$  trước giá trị  $b$ , điều đó chứng tỏ không thỏa mãn yêu cầu nhất quán tuần tự.

Để minh họa cho nhất quán tuần tự, hãy xem xét ví dụ ba tiến trình thực hiện tương tranh trên hình 8.6, các mục dữ liệu được thể hiện bằng ba biến ( $x, y, z$ ) đều có giá trị ban đầu bằng 0, thao tác gán ( $\leftarrow$ ) thể hiện cho lệnh ghi và thao tác in (print) thể hiện cho lệnh đọc. Tuần tự xen kẽ thực hiện với 6 lệnh trên sẽ cho 720 khả năng khác nhau,

mặc dù một số tổ hợp vi phạm thứ tự của chương trình. Nếu xét trường hợp tiến trình  $P_1$  được thực hiện trước, nghĩa là trước tiên phải thực hiện lệnh gán  $x \leftarrow 1$  như vậy sẽ chỉ còn 5 lệnh do đó hình thành tối đa 120 khả năng, tương tự trong 5 lệnh này lại xét đến tiến trình số 2 lệnh gán  $y \leftarrow 1$  được thực hiện trước do đó chỉ còn 60 khả năng, cuối cùng áp dụng điều kiện lệnh gán  $z \leftarrow 1$  được thực hiện trước cho nên chỉ có 30 khả năng hợp lệ, vì có ba cách chọn tiến trình thực hiện đầu tiên cho nên sẽ có tối đa 90 khả năng hợp lệ.

Tiến trình $P_1$	Tiến trình $P_2$	Tiến trình $P_3$
$x \leftarrow 1;$ print(y,z);	$y \leftarrow 1;$ print(x,z);	$z \leftarrow 1;$ print(x,y);

Hình 8.6 Ba tiến trình thực hiện tương tranh

Hình 8.7 thể hiện 4 trong số 90 tổ hợp đã phân tích trên, (a) thể hiện thứ tự thực hiện các tiến trình  $P_1$ ,  $P_2$  và  $P_3$ , ba trường hợp còn lại minh họa các khả năng khác nhau về việc xen kẽ các lệnh nhưng vẫn đảm bảo theo đúng thứ tự đã qui định trong chương trình. Mỗi tiến trình tạo ra một xâu 2 bit, kết quả là đầu ra thực tế xuất hiện trên các thiết bị sau các lệnh in, chúng không thể hiện được bất kỳ điều gì về thứ tự thực hiện của các tiến trình.

Các khả năng thực hiện				
(a)	(b)	(c)	(d)	
$x \leftarrow 1;$ print(y,z); $y \leftarrow 1;$ print(x,z); $z \leftarrow 1;$ print(x,y); Kết quả: 001011 Chữ ký: 001011	$x \leftarrow 1;$ $y \leftarrow 1;$ print(x,z); print(y,z); $z \leftarrow 1;$ print(x,y); Kết quả: 101011 Chữ ký: 101011	$y \leftarrow 1;$ $z \leftarrow 1;$ print(x,y); print(x,z); $x \leftarrow 1;$ print(y,z); Kết quả: 010111 Chữ ký: 110101	$y \leftarrow 1;$ $x \leftarrow 1;$ $z \leftarrow 1;$ print(x,z); print(y,z); print(x,y); Kết quả: 111111 Chữ ký: 111111	....

Hình 8.7 Bốn trường hợp nhất quán tuân tự hợp lệ

Nếu đưa kết quả in của các tiến trình theo thứ tự  $P_1$ ,  $P_2$  và  $P_3$ , chúng ta sẽ nhận được xâu ký tự 6 bit thể hiện sự xen kẽ của các câu lệnh, xâu bit này gọi là chữ ký. Ví dụ trường hợp chữ ký 001011, hai bit đầu tiên 00 đại diện cho  $yz$  thể hiện khi tiến trình  $P_1$  thực hiện lệnh in thì hai biến  $y$  và  $z$  vẫn chưa được gán giá trị, nghĩa là các lệnh của tiến trình  $P_2$  và  $P_3$  chỉ được thực hiện sau tất cả các lệnh của tiến trình  $P_1$ . Hai bit tiếp theo 10 đại diện cho  $xz$  chứng tỏ khi thực hiện xong các lệnh của tiến trình  $P_2$  thì vẫn chưa thực hiện các lệnh của tiến trình  $P_3$ .

Mặc dù có 64 khả năng xảy ra nhưng không phải khả năng nào cũng hợp lệ vì cần phải đảm bảo yêu cầu theo đúng thứ tự của chương trình gán giá trị vào biến trước khi thực hiện lệnh in. Ví dụ, chữ ký 001001 không hợp lệ vì hai bit đầu tiên 00 thể hiện tiến

trình P1 được thực hiện đầu tiên, hai bit kế tiếp thể hiện tiến trình P2 thực hiện trước tiến trình P3, hai bit cuối cùng 01 thể hiện tiến trình P3 thực hiện trước tiến trình P1 nên mâu thuẫn với kết luận phân tích hai bit đầu tiên.

Như vậy trong số 90 khả năng tổ hợp được coi là hợp lệ sẽ không quá 64 khả năng được phép thực hiện theo quan điểm nhất quán tuần tự, các tiến trình phải chấp nhận các giá trị hợp lệ này. Qua phân tích nêu trên có thể thấy mô hình nhất quán tuần tự là một trong những mô hình quan trọng nhất, về bản chất nó đáp ứng cho việc phát triển các ứng dụng song song với những thao tác tương tranh.

Việc thực hiện nhất quán tuần tự không phải là điều đơn giản, để minh họa vấn đề này hãy xét ví dụ liên quan đến hai biến  $x$  và  $y$  thể hiện trên hình 8.8a. Tiến trình  $P_1$  lần lượt thực hiện lệnh ghi giá trị  $a$  vào các biến  $x$  và  $y$ , tiến trình  $P_2$  lần lượt thực hiện lệnh ghi giá trị  $b$  vào các biến  $y$  và  $x$ , khi kết hợp các lệnh của cả hai tiến trình với nhau đã vi phạm tính nhất quán tuần tự.

$P_1:$	$W(x)a$	$W(y)a$	$R(x)b$
$P_2:$		$W(y)b$	$W(x)b$
(a) Không tuyến tính hóa			
$P_1:$	$W(x)a$	$W(y)a$	$R(x)b$
$P_2:$		$W(y)b$	$W(x)b$
(b) Tuyến tính hóa			

Hình 8.8 Tuyến tính hóa trong mô hình nhất quán tuần tự

Nếu chỉ xét các thao tác ghi và đọc trên biến x thì việc  $P_1$  đọc được giá trị a là hoàn toàn phù hợp, tương tự như vậy nếu  $P_2$  đọc được giá trị của y là b. Tuy nhiên khi kết hợp với nhau, không có cách nào để sắp xếp thứ tự các thao tác ghi trên x và y sao cho  $R_1(x)a$  và  $R_2(y)b$ . Nếu chỉ cần duy trì thứ tự như tiến trình đã thực hiện, tổ hợp các trường hợp thực hiện như sau:

<b>Thứ tự thao tác</b>	<b>Kết quả</b>	
$W_1(x)a; W_1(y)a; W_2(y)b; W_2(x)b;$	$R_1(x)b$	$R_2(y)b$
$W_1(x)a; W_2(y)b; W_1(y)a; W_2(x)b;$	$R_1(x)b$	$R_2(y)a$
$W_1(x)a; W_2(y)b; W_2(x)b; W_1(y)a;$	$R_1(x)b$	$R_2(y)a$
$W_2(y)b; W_1(x)a; W_1(y)a; W_2(x)b;$	$R_1(x)b$	$R_2(y)a$
$W_2(y)b; W_1(x)a; W_2(x)b; W_1(y)a;$	$R_1(x)b$	$R_2(y)a$
$W_2(y)b; W_2(x)b; W_1(x)a; W_1(y)a;$	$R_1(x)a$	$R_2(y)a$

Về giao tác, các thao tác được thực hiện bởi các tiến trình  $P_1$  và  $P_2$  không thể tuân tự hóa, nhất quán tuần tự không phải là thành phần, nghĩa là giữ tuần tự cho các mục dữ liệu nhưng không cần tập hợp thao tác của tiến trình. Vấn đề về tính nhất quán không thành phần có thể được giải quyết bằng cách giả định tuyến tính, điều này được giải thích tốt nhất bằng cách phân biệt giữa bắt đầu và hoàn thành thao tác trong một khoảng thời gian, mỗi thao tác sẽ có hiệu lực ngay lập tức tại một thời điểm nào đó từ khi bắt đầu đến khi hoàn thành. Tác động của một thao tác sẽ diễn ra trong khoảng thời gian nhất định, khi hoàn thành thao tác cập nhật thì kết quả cũng sẽ được lan truyền đến tất cả các bản sao khác.

Hình 8.8b thể hiện cùng một tập hợp các hoạt động như ví dụ trên vùng tô xám biểu thị thời điểm thực hiện của thao tác, hiệu ứng của một thao tác xuất hiện ngay trong khoảng thời gian này. Điều này có nghĩa là tại thời điểm thao tác ghi hoàn thành thì kết quả phải được lan tỏa đến các bản sao khác, tuyển tính hóa nhất quán tuần tự nhằm mục đích các thao tác cập nhật cho kết quả duy nhất đối với các mục dữ liệu. Thao tác  $W_2(y)b$  hoàn thành trước khi  $W_1(y)a$  bắt đầu, do đó  $y$  sẽ có giá trị  $a$ , tương tự như vậy, thao tác  $W_1(x)a$  hoàn thành trước khi  $W_2(x)b$  bắt đầu, do đó  $x$  sẽ có giá trị  $b$ , kết quả sẽ có 4 khả năng sau:

Thứ tự thao tác	Kết quả	
$W_1(x)a; W_2(y)b; W_1(y)a; W_2(x)b;$	$R_1(x)b$	$R_2(y)a$
$W_1(x)a; W_2(y)b; W_2(x)b; W_1(y)a;$	$R_1(x)b$	$R_2(y)a$
$W_2(y)b; W_1(x)a; W_1(y)a; W_2(x)b;$	$R_1(x)b$	$R_2(y)a$
$W_2(y)b; W_1(x)a; W_2(x)b; W_1(y)a;$	$R_1(x)b$	$R_2(y)a$

Việc tuyển tính hóa nhất quán tuần tự không phải dễ dàng thực hiện, có thể dễ dàng xác định tính nhân quả của các sự kiện trên một tiến trình nhưng khó có thể kết luận tính nhân quả giữa các sự kiện thuộc các tiến trình khác nhau. Các tiến trình phải trao đổi nhiều thông điệp để đạt được sự đồng thuận về thứ tự các thao tác ghi, điều đó làm suy giảm hiệu năng, nhưng đó là điều tất yếu để đảm bảo mục tiêu duy trì tính nhất quán tuần tự.

#### 8.2.2.3 Mô hình nhất quán nhân quả

Mô hình nhất quán nhân quả do Hutto và Ahamad đề xuất vào năm 1990 thể hiện sự nối lồng nhất quán tuần tự, nó phân biệt các sự kiện có quan hệ nhân quả và các sự kiện không có quan hệ nhân quả. Xét ví dụ về nhân bản cơ sở dữ liệu phân tán trên hình 8.9, tiến trình  $P_1$  thực hiện ghi các giá trị  $a$  và  $c$  vào biến  $x$  bằng các lệnh  $W_1(x)a$  và  $W_1(x)c$ , tiến trình  $P_2$  đọc giá trị biến  $x$   $R_2(x)a$  và thực hiện thao tác  $W_2(x)b$  để ghi giá trị  $b$  vào biến  $x$ , dễ dàng nhận thấy quan hệ nhân quả  $W_1(x)a \rightarrow W_1(x)c$  vì các thao tác ghi đều được thực hiện trên tiến trình  $P_1$ ,  $R_2(x)a \rightarrow W_2(x)b$  vì thao tác đọc được thực hiện trước thao tác ghi trong tiến trình  $P_2$ , trong khi đó hai sự kiện  $W_2(x)b$  và  $W_1(x)c$  tương tranh với nhau nên chúng không có mối quan hệ nhân quả. Như vậy các tiến trình chỉ cần thực hiện theo thứ tự  $W_1(x)a$  còn thứ tự thực hiện các lệnh  $W_2(x)b$  và  $W_1(x)c$  không quan trọng.

$P_1:$	$W(x)a$	$W(x)c$	
$P_2:$	$R(x)a$	$W(x)b$	
$P_3:$	$R(x)a$	$R(x)c$	$R(x)b$
$P_4:$	$R(x)a$	$R(x)b$	$R(x)c$

Hình 8.9 Mô hình nhất quán nhân quả

Nếu sự kiện  $b$  được gây ra hoặc bị tác động bởi một sự kiện  $a$  xảy ra sớm hơn thì tính nhân quả đòi hỏi mọi thực thể khác phải nhìn thấy  $a$  trước rồi mới thấy  $b$ . Mô hình

nhất quán nhân quả thỏa mãn các điều kiện: các thao tác ghi có quan hệ nhân quả tiềm năng phải được nhận biết bởi tất cả các tiến trình khác trong cùng một thứ tự. Các thao tác ghi không có quan hệ nhân quả thì có thể thực hiện theo thứ tự khác nhau trên các tiến trình khác nhau.

$P_1: W(x)a$	$P_1: W(x)a$
$P_2: R(x)a \quad W(x)b$	$P_2: \quad \quad W(x)b$
$P_3: \quad \quad \quad R(x)b \quad R(x)a$	$P_3: \quad \quad \quad R(x)b \quad R(x)a$
$P_4: \quad \quad \quad R(x)a \quad R(x)b$	$P_4: \quad \quad \quad R(x)a \quad R(x)b$
(a) Không nhất quán nhân quả	(b) Không nhất nhân quả

Hình 8.10 Nhất quán nhân quả và vi phạm tính nhất quán nhân quả

Một ví dụ khác trên hình 8.10, tiến trình  $P_1$  thực hiện lệnh  $W_1(x)a$  để ghi giá trị  $a$  vào biến  $x$ , tiến trình  $P_2$  thực hiện lệnh  $W_2(x)b$  để ghi giá trị  $b$  vào biến  $x$ , tuy nhiên trường hợp (a) tiến trình  $P_2$  thực hiện lệnh đọc  $R_2(x)a$  trước thi thực hiện lệnh ghi, do đó có thể xác định quan hệ nhân quả  $W_1(x)a \rightarrow W_2(x)b$ , vì vậy trường hợp này không đáp ứng yêu cầu nhất quán nhân quả. Với trường hợp (b), thao tác  $R(x)a$  trên tiến trình  $P_2$  đã bị loại bỏ do đó  $W_1(x)a \parallel W_2(x)b$ . Như vậy trường hợp (b) có thể đáp ứng yêu cầu nhất quán nhân quả nhưng chắc chắn không thể chấp nhận được đối với yêu cầu nhất quán tuần tự.

$P_1: W(x)a$		
$P_2: \quad \quad \quad R(x)a \quad W(y)b$		
$P_3: \quad \quad \quad \quad \quad R(y)b \quad R(x)?$		
$P_4: \quad \quad \quad \quad \quad R(x)a \quad R(y)?$		

Hình 8.11 Nhất quán nhân quả trên các biến khác nhau

Việc cài đặt nhất quán nhân quả đòi hỏi phải lưu vết những tiến trình đã thực hiện lệnh ghi, tuy nhiên cũng còn nhiều vấn đề cần phải giải quyết. Ví dụ trên hình 8.11 thay thế lệnh ghi  $W_2(x)b$  bằng lệnh  $W_2(y)b$ , tiến trình  $P_3$  thực hiện lệnh  $R_3(y)$  cho kết quả  $b$  vì quan hệ nhân quả  $W(x)a \rightarrow R(x)a \rightarrow W(y)b$ , như vậy lệnh  $R_3(x)$  phải trả về kết quả là  $a$  thì mới đáp ứng yêu cầu nhất quán nhân quả. Thao tác  $R_4(y)$  không nhất thiết phải cho kết quả là  $b$  mặc dù tồn tại quan hệ nhân quả  $W(x)a \rightarrow W(y)b$  nhưng thao tác ghi giá trị  $a$  vào biến  $x$  hoàn toàn độc lập với thao tác ghi giá trị  $b$  vào biến  $y$ . Cần thiết phải xây dựng đồ thị phụ thuộc, điều này có thể thực hiện bằng cách gán nhãn thời gian vector cho các thao tác.

#### 8.2.2.4 Mô hình nhất quán hàng đợi

Nhất quán hàng đợi thỏa mãn điều kiện các thao tác ghi bởi một tiến trình đơn phải được tất cả các tiến trình khác nhìn thấy theo cùng một trật tự mà chúng đề ra nhưng thao tác ghi bởi nhiều tiến trình khác nhau có thể được thấy theo những trật tự khác nhau. Trong ví dụ trên hình 8.12, mặc dù các tiến trình  $P_3$  và  $P_4$  đọc được các giá trị theo thứ tự

khác nhau nhưng chúng vẫn đảm bảo thực hiện các thao tác  $W(x)b$  và  $W(x)c$  theo đúng thứ tự qui định trong tiến trình  $P_2$ .

$P_1:$	$W(x)a$		
$P_2:$	$R(x)a$	$W(x)b$	$W(x)c$
$P_3:$		$R(y)b$	$R(x)a$
$P_4:$		$R(x)a$	$R(x)b$

Hình 8.12 Mô hình nhất quán hàng đợi

Triển khai nhất quán hàng đợi khá dễ dàng, nó chỉ cần đảm bảo thực thi các thao tác cập nhật theo đúng thứ tự mỗi tiến trình đã qui định mà không cần để ý đến các thao tác cập nhật của các tiến trình khác. Điều này thực hiện đơn giản bằng cách gán nhãn thời gian Lamport cho mỗi thao tác cập nhật, tuy nhiên cần lưu ý về tính tương tranh giữa các thao tác cập nhật của các tiến trình khác nhau.

### 8.2.3 *Nhất quán theo nhóm các thao tác*

Nhất quán tuần tự và nhất quán nhân quả được định nghĩa ở mức độ các thao tác đọc và ghi, cấp độ nhỏ nhất này xuất phát từ lịch sử, ban đầu các mô hình được phát triển cho các hệ thống nhiều bộ xử lý chia sẻ bộ nhớ, chúng thực sự được cài đặt trong phần cứng. Mặc dù mô hình nhất quán hàng đợi đã có hiệu năng tốt hơn các mô hình trước nhưng nó vẫn có một số hạn chế không cần thiết đối với các ứng dụng bởi vì nó đòi hỏi các thao tác ghi phát sinh từ một tiến trình phải được các tiến trình khác nhìn thấy theo đúng trình tự. Ví dụ một tiến trình trong vùng tới hạn đang đọc và ghi một số biến trong vòng lặp, một số tiến trình không cho rằng có thể tiếp cận các biến này cho đến khi tiến trình thứ nhất rời khỏi vùng tới hạn, bộ nhớ không biết chắc khi nào tiến trình ở trong vùng tới hạn và do đó nó sẽ lan tỏa thao tác ghi đến các bộ nhớ theo cách thông thường.

#### 8.2.3.1 *Mô hình nhất quán yếu*

Mô hình nhất quán yếu không tập trung vào các thao tác trên dữ liệu như các mô hình nhất quán theo thứ tự các thao tác mà chúng quan tâm đến trật tự các nhóm lệnh bằng việc sử dụng các biến đồng bộ. Sau khi các thao tác cập nhật đã hoàn thành trên một bản sao thì nó yêu cầu chiếm giữ biến đồng bộ để lan tỏa các cập nhật sang các bản sao khác. Năm 1986 Dubois đã phát biểu về ba đặc điểm của mô hình nhất quán yếu như sau:

- Việc truy nhập đến một biến đồng bộ là nhất quán tuần tự.
- Không có thao tác nào lên các biến đồng bộ được phép thực hiện cho đến khi tất cả các thao tác ghi trước đó được hoàn thành ở mọi nơi.
- Không có thao tác đọc hay ghi dữ liệu lên các mục dữ liệu nào được phép thực hiện cho đến khi tất cả các thao tác trước đó lên các biến đồng bộ được thực hiện.

Đặc điểm thứ nhất cho thấy tất cả các tiến trình nhìn nhận việc truy nhập biến đồng bộ theo cùng thứ tự, khi một tiến trình chiếm được quyền sở hữu biến đồng bộ thì nó quảng bá thông điệp đến tất cả các tiến trình khác. Không một biến đồng bộ nào khác có thể được truy nhập bởi bất kỳ tiến trình nào cho đến khi tất cả các thao tác cập nhật của nó đã hoàn thành ở mọi nơi.

Đặc điểm thứ hai nghĩa là việc truy nhập biến đồng bộ được thực hiện theo cơ chế tuần tự, nó buộc những thao tác cập nhật đang thực hiện, thực hiện một phần hoặc đã thực hiện xong trên một số bản sao nào đó thì đều phải hoàn thành ở mọi nơi. Khi thực hiện đồng bộ thì phải đảm bảo tất cả các thao tác cập nhật trước đó đều đã hoàn thành, với việc thực hiện đồng bộ sau khi cập nhật dữ liệu thì tiến trình có thể buộc các giá trị mới được phải được cập nhật ở các bản sao khác.

Đặc điểm thứ ba nghĩa là khi truy nhập đến biến đồng bộ bình thường để đọc hoặc ghi dữ liệu thì tất cả những đồng bộ trước đó đều đã phải hoàn thành. Với việc thực hiện đồng bộ trước khi đọc dữ liệu thì chắc chắn tiến trình sẽ nhận được những giá trị cập nhật gần nhất. Cũng cần phải lưu ý tính chất phức tạp ẩn sau khái niệm hoàn thành thao tác cập nhật ở mọi nơi, thao tác đọc được coi là hoàn thành nếu không có một thao tác ghi kế tiếp nào ảnh hưởng đến giá trị trả về của nó, thao tác ghi được coi là hoàn thành nếu các thao tác đọc kế tiếp nhận được kết quả mới được cập nhật.

$P_1: W(x)a \quad W(x)b \quad S$	$P_1: W(x)a \quad W(x)b \quad S$
$P_2:$	$R(x)a \quad R(x)b \quad S$
$P_3:$	$R(x)b \quad R(x)a \quad S$
(a) Đáp ứng yêu cầu nhất quán yếu	(a) Không đáp ứng yêu cầu nhất quán yếu

Hình 8.13 Mô hình nhất quán yếu

Xét ví dụ trên hình 8.13, tiến trình  $P_1$  thực hiện hai lệnh ghi liên tiếp  $W_1(x)a$  và  $W_1(x)b$  sau đó thực hiện đồng bộ, trên hình vẽ ký hiệu bằng chữ  $S$ . Trường hợp (a), hai tiến trình  $P_2$  và  $P_3$  thực hiện các lệnh đọc sau đó mới đồng bộ nên thứ tự của chúng khác nhau nhưng vẫn được coi là đáp ứng yêu cầu của mô hình nhất quán yếu. Trường hợp (b), tiến trình  $P_2$  thực hiện đồng bộ sau đó mới đọc giá trị của biến  $x$ , như vậy có thể coi tiến trình  $P_2$  đã được đồng bộ nên giá trị đọc được của biến  $x$  phải là  $b$  chứ không phải  $a$ , do đó có thể coi như không đáp ứng yêu cầu mô hình nhất quán yếu.

#### 8.2.3.2 Mô hình nhất quán phát hành

Trong mô hình nhất quán yếu, khi đã truy nhập được biến đồng bộ thì không biết được khi nào tiến trình mới hoàn thành việc đọc hoặc ghi dữ liệu, kết quả là cần phải thực hiện động tác để đảm bảo chắc chắn các cập nhật cục bộ đều đã hoàn thành. Nếu các tiến trình có thể cho biết việc thâm nhập và rời khỏi vùng tới hạn thì việc cài đặt sẽ hiệu quả hơn có thể thực hiện được, cần thiết phải có hai biến đồng bộ chứ không phải là một biến.

Cấp độ rất nhỏ của các mô hình nhất quán theo thứ tự các thao tác không phù hợp với mức độ của các ứng dụng, tính tương tranh giữa dữ liệu chia sẻ được điều khiển bằng các cơ chế đồng bộ để loại trừ lẫn nhau và các giao tác, các lệnh đọc và ghi được thực hiện giữa hai lệnh chiếm vùng tới hạn ENTER\_CS và rời khỏi vùng tới hạn LEAVE\_CS, khi đã vào được vùng tới hạn thì tiến trình có thể an tâm thực hiện chuỗi các thao tác đọc và ghi trên kho dữ liệu.

Về bản chất, điều gì xảy ra bên trong chương trình mà dữ liệu chịu tác động bởi chuỗi các thao tác đọc ghi được bảo vệ chống lại những truy nhập tương tranh sẽ dẫn đến

việc nhìn thấy những giá trị khác với kết quả của việc thực hiện toàn bộ chuỗi. Nói cách khác, việc đưa chuỗi các thao tác đọc/ghi thành một đơn vị nguyên tử thì sẽ nâng cao mức độ chi tiết, để đạt được điều này thì cần phải có ngữ cảnh chính xác liên quan tới các thao tác xâm nhập và thoát khỏi vùng tới hạn, ngữ cảnh này được hình thành dựa trên khái niệm các biến đồng bộ chia sẻ hoặc chỉ đơn giản là khóa. Mỗi khóa liên kết với các mục dữ liệu chia sẻ nhưng mỗi mục dữ liệu chia sẻ chỉ được liên kết với một khóa. Một tiến trình muôn vào vùng tới hạn thì phải yêu cầu biến đồng bộ, nghĩa là phải thiết lập các khóa liên quan và khi thoát khỏi vùng tới hạn thì phải giải phóng các khóa này.

Mỗi khóa có một chủ sở hữu hiện hành, cụ thể đó là tiến trình cuối cùng chiếm hữu được khóa. Một tiến trình muôn có được khóa thì phải gửi thông điệp đến chủ sở hữu hiện hành để yêu cầu quyền sở hữu và các giá trị hiện tại của dữ liệu liên quan đến khóa. Nếu tiến trình sở hữu khóa ở chế độ độc quyền thì có thể thực hiện các thao tác đọc/ghi lên các mục dữ liệu liên quan của khóa, nếu ở chế độ không độc quyền thì chỉ có thể đọc nhưng không được phép thực hiện các thao tác ghi. Tất nhiên sở hữu không độc quyền chỉ có thể được cấp khi chưa có tiến trình nào sở hữu khóa ở chế độ độc quyền. Bershad năm 1993 đã chỉ ra ba yêu cầu cần phải đáp ứng:

1. Yêu cầu chiếm hữu khóa chỉ có thể thành công khi đã hoàn thành tất cả các thao tác cập nhật dữ liệu chi sẻ liên quan tới khóa.
2. Yêu cầu chiếm hữu khóa ở chế độ độc quyền chỉ có thể thành công nếu chưa có tiến trình nào sở hữu khóa ở chế độ độc quyền hay không độc quyền.
3. Yêu cầu chiếm hữu khóa ở chế độ không độc quyền chỉ có thể thành công nếu việc chiếm hữu độc quyền trước đó đã kết thúc và tất cả các thao tác cập nhật dữ liệu liên quan đến khóa này đều đã hoàn thành.

Điều kiện thứ nhất nói rằng, khi một tiến trình chiếm hữu khóa thì việc chiếm hữu đó có thể chưa hoàn thành cho đến khi tất cả các dữ liệu được bảo vệ đã hoàn thành cập nhật. Điều kiện thứ hai ngũ ý trước khi cập nhật mục dữ liệu chia sẻ thì tiến trình phải vào vùng tới hạn ở chế độ độc quyền để đảm bảo rằng không một tiến trình nào khác có thể thay đổi mục dữ liệu chia sẻ trong cùng thời gian đó. Điều kiện thứ ba nói rằng nếu một tiến trình muôn vào vùng tới hạn trong chế độ không độc quyền thì đầu tiên phải kiểm tra với chủ sở hữu của biến đồng bộ bảo vệ vùng tới hạn để lấy bản sao gần nhất của dữ liệu chia sẻ được bảo vệ.

$P_1:$	Acq(L)	W(x)a	W(x)b	Rel(L)
$P_2:$			Acq(L)	R(x)b
$P_3:$				Rel(L)

Hình 8.14 Mô hình nhất quán phát hành

Mô hình nhất quán phát hành sử dụng lệnh chiếm giữ Acq để có thể vào vùng tới hạn và lệnh giải phóng Rel để ra khỏi vùng tới hạn, hai lệnh này được thực hiện bằng các thao tác theo thứ tự trên các biến hoặc bằng các thao tác đặc biệt, chúng chỉ thực hiện với

các dữ liệu dùng chung chứ không áp dụng cho tất cả các dữ liệu. Việc truy nhập vào các biến đồng bộ hóa là nhất quán hàng đợi, không yêu cầu nhất quán tuần tự.

Tiến trình chỉ được phép đọc/ghi lên dữ liệu khi đã hoàn thành thao tác chiếm giữ Acq, tất cả các thao tác đọc/ghi phải được hoàn tất trước khi thực hiện lệnh giải phóng Rel. Hình 8.14 minh họa mô hình nhất quán phát hành, tiến trình P<sub>1</sub> gửi yêu cầu chiếm giữ và thực hiện cập nhật các mục dữ liệu, khi cập nhật xong dữ liệu trên bản sao cục bộ mới lan tỏa cập nhật tới các bản sao khác.

#### 8.2.3.3 Mô hình nhất quán mục dữ liệu

Mô hình nhất quán mục dữ liệu cũng sử dụng hai lệnh chiếm giữ Acq và lệnh giải phóng Rel khi muốn vào vùng thời hạn, thay cho việc khóa toàn bộ dữ liệu chia sẻ thì chỉ kết hợp khóa với mỗi mục dữ liệu, các lệnh này thao tác trên từng mục dữ liệu của vùng dữ liệu chia sẻ. Tiến trình nào muốn sử dụng mục dữ liệu thì phải đợi cho tất cả các tiến trình khác giải phóng mục dữ liệu đó.

Hình 8.15 minh họa nhất quán mục dữ liệu, tiến trình P<sub>1</sub> yêu cầu chiếm giữ và ghi giá trị a vào biến x sau đó tiếp tục chiếm giữ và thay đổi giá trị b vào biến y, cuối cùng giải phóng hai biến này. Tiến trình P<sub>2</sub> gửi yêu cầu chiếm giữ để đọc giá trị của biến x sau khi tiến trình P<sub>1</sub> đã giải phóng biến này nhưng chưa giải phóng biến y, do đó P<sub>2</sub> đọc được giá trị a và nếu đọc biến y thì có thể sẽ nhận được giá trị null. Tiến trình P<sub>3</sub> gửi yêu cầu chiếm giữ để đọc biến y sau khi tiến trình P<sub>1</sub> giải phóng biến y và vì vậy nó nhận được giá trị b.

P <sub>1</sub> :	Acq(Lx)	W(x)a	Acq(Ly)	W(y)b	Rel(Lx)	Rel(Ly)
P <sub>2</sub> :					Acq(Lx)	R(x)a
P <sub>3</sub> :					R(y) Nil	Acq(Ly) R(y)b

Hình 8.15 Mô hình nhất quán mục dữ liệu

Vấn đề của nhất quán mục dữ liệu là phải kết hợp chính xác dữ liệu với biến đồng bộ, trong cách tiếp cận dựa trên đối tượng chúng ta có thể kết hợp biến đồng bộ với đối tượng và thực hiện tuần tự hóa khi gọi các đối tượng này. Nếu không sử dụng cách tiếp cận trên thì có thể thông báo tường minh những dữ liệu nào sẽ bị ảnh hưởng khi thực hiện giao tác.

Để ghi lên một mục dữ liệu, máy khách phải chiếm được biến đồng bộ của mục đó trong chế độ dành riêng, điều này có nghĩa là không máy khách nào khác có thể sử dụng biến đó, khi máy khách cập nhật xong mục dữ liệu thì nó giải phóng biến đó. Khi máy khách muốn đọc một mục dữ liệu nào đó, nó phải có được biến đồng bộ hóa kết hợp ở chế độ không dành riêng, nhiều máy khách có thể giữ biến đồng bộ hóa ở chế độ không dành riêng.

Khi thực hiện một thao tác chiếm giữ Acq, máy khách lấy về phiên bản mới nhất của mục dữ liệu từ tiến trình cuối cùng thực hiện thao tác chiếm giữ trên biến đó. Thao tác chiếm giữ Acq để truy nhập vào một biến đồng bộ hóa không được phép thực hiện trong một tiến trình cho đến khi tất cả các cập nhật lên mục dữ liệu trong tiến trình đó

được hoàn thành. Trước khi một truy nhập trong chế độ dành riêng của một tiến trình tới một biến đồng bộ được phép thực hiện thì không tiến trình nào khác còn được giữ các biến đồng bộ, trong chế độ không dành riêng thì không cần yêu cầu như vậy. Sau khi một truy nhập trong chế độ dành riêng lên một biến đồng bộ hóa được thực hiện thì bất kì sự truy nhập của tiến trình nào khác trong chế độ không dành riêng lên biến đó cũng không được thực hiện cho đến khi chủ nhân của biến đồng bộ thực hiện xong việc truy nhập của mình.

#### **8.2.4 Tính nhất quán và gắn kết**

Tại thời điểm này cần thiết phải làm rõ sự khác biệt giữa hai khái niệm liên quan mật thiết với nhau, các mô hình nhất quán đề cập trong các phần trước thực tế đã giải quyết vấn đề các tiến trình thực hiện thao tác đọc và ghi trên tập các mục dữ liệu, chúng mô tả kết quả mong đợi khi nhiều tiến trình đồng thời thực hiện các thao tác trên dữ liệu, sẽ nhất quán nếu tuân thủ các qui tắc đã đề ra cho mô hình.

Các mô hình gắn kết mô tả kết quả mong đợi đối với từng mục dữ liệu, trong trường hợp này dữ liệu được nhân bản ở nhiều nơi, chúng được coi là gắn kết khi các bản sao tuân thủ qui tắc đã định nghĩa trong mô hình gắn kết. Mô hình phổ biến về gắn kết là mô hình nhất quán tuần tự nhưng chỉ áp dụng cho một mục dữ liệu, nghĩa là trong trường hợp các yêu cầu cập nhật xảy ra đồng thời thì tất cả các tiến trình sẽ nhìn thấy cùng một thứ tự thực hiện các yêu cầu cập nhật.

### **8.3 Nhất quán lấy máy khách làm trung tâm**

Các mô hình nhất quán lấy dữ liệu làm trung tâm nhằm mục đích cung cấp cách nhìn nhất quán dữ liệu trên toàn hệ thống, cần phải đảm bảo tính nhất quán khi các tiến trình tương tranh có thể đồng thời cập nhật dữ liệu. Ví dụ trường hợp nhất quán mục dữ liệu, khi gọi một đối tượng thì nó sẽ được cung cấp bản sao của đối tượng đó phản ánh tất cả những thay đổi trên đối tượng cho tới thời điểm hiện hành, quá trình thực hiện không bị tác động của các tiến trình khác.

Khả năng xử lý các thao tác tương tranh trên dữ liệu dùng chung trong khi duy trì tính nhất quán mạnh là yếu tố cơ bản đối với các hệ thống phân tán. Tính nhất quán mạnh có thể được đảm bảo khi các tiến trình sử dụng các cơ chế như giao tác hoặc các biến đồng bộ, nhưng vì lý do hiệu năng nên phải chấp nhận những mô hình nhất quán yếu hơn, chẳng hạn kết hợp mô hình nhất quán nhân quả với nhất quán sau cùng.

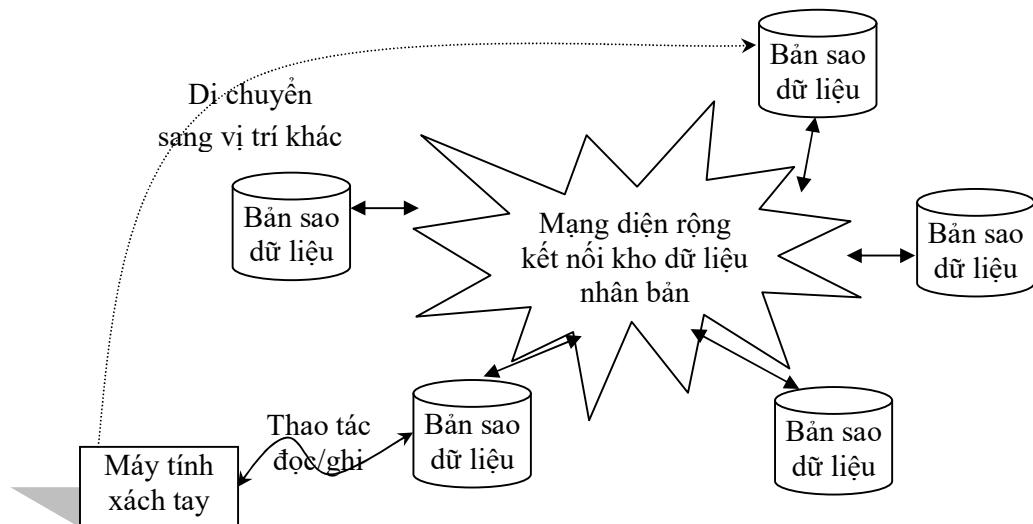
Phần này sẽ đề cập tới một số trường hợp đặc biệt của các dữ liệu phân tán, giả thiết không có các thao tác cập nhật đồng thời hoặc nếu có thì cũng dễ dàng giải quyết, các thao tác đọc chiếm đa số trong các hoạt động của hệ thống. Các bản sao dữ liệu này cung cấp mô hình nhất quán yếu, ví dụ mô hình nhất quán sau cùng. Các mô hình nhất quán lấy máy khách làm trung tâm cho thấy có thể che giấu tính không nhất quán với chi phí tương đối thấp.

#### **8.3.1 Khái niệm nhất quán sau cùng**

Trong nhiều hệ thống cơ sở dữ liệu, hầu hết các tiến trình thực hiện đọc chứ ít khi thực hiện các thao tác ghi, như vậy xung đột ghi/ghi hiếm khi xảy ra mà chủ yếu phải giải

quyết vấn đề xung đột đọc/ghi. Nếu một tiến trình cập nhật dữ liệu và nhiều tiến trình đang cần đọc thì phải giải quyết là làm sao các thao tác ghi được thực hiện nhanh nhất để cung cấp kết quả cho các tiến trình chỉ đọc, vậy tính nhất quán cần hoàn toàn có thể bị vi phạm. Nếu trong một khoảng thời gian dài mà không xuất hiện lệnh cập nhật thì các bản sao sẽ dần dần trở nên nhất quán, loại nhất quán này gọi là nhất quán sau cùng.

Nhất quán sau cùng sẽ hoạt động tốt nếu máy khách chỉ truy nhập đến một bản sao, vấn đề sẽ nảy sinh khi máy khách thực hiện cập nhật tại một bản sao và trong thời gian ngắn chuyển sang bản sao khác. Trường hợp máy khách là thiết bị di động, việc thực hiện yêu cầu trên gấp khó khăn hơn, phải luôn đảm bảo rằng ngay cả khi máy khách thay đổi về vị trí vật lý thì việc sử dụng các bản sao cũng phải chính xác, tức là các bản sao luôn nhất quán.



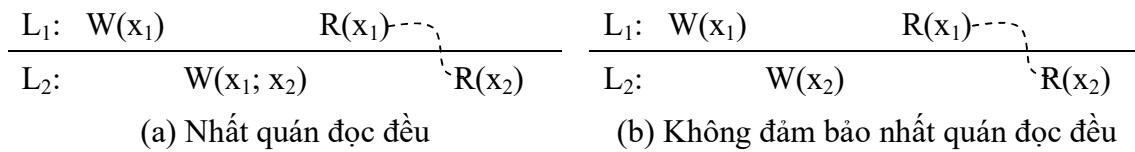
Hình 8.16 Người sử dụng truy nhập các bản sao dữ liệu nhân bản

Ví dụ trên hình 8.16, người dùng sử dụng máy tính kết nối mạng không dây, việc kết nối đến bản sao nào hoàn toàn trong suốt đối với người sử dụng, sau khi cập nhật tại một bản sao người dùng di chuyển sang vị trí khác và hoàn toàn có khả năng phần mềm ứng dụng có thể kết nối đến bản sao chưa kịp cập nhật. Sau các thao tác cập nhật thì tất cả các bản sao đều phải giống nhau, yêu cầu này sẽ được thực hiện tốt nếu mỗi máy khách luôn chịu khó cập nhật cho các bản sao. Việc cập nhật các bản sao ngay sau khi cập nhật bản chính có thể kéo dài thời gian thực hiện, do đó lập trình viên cần dự đoán thời gian thực hiện mỗi yêu cầu và lựa chọn phương án thích hợp.

Vấn đề trên có thể giảm bớt bằng cách sử dụng nhất quán lấy máy khách làm trung tâm, nó đảm bảo tính nhất quán cho một máy khách chứ không đảm bảo nhất quán cho các máy khách khác nhau. Về cơ bản, máy khách sẽ chỉ kết nối đến một bản sao, bản sao đó có thể cài đặt cục bộ, mọi thao tác đều được thực hiện trên bản sao này, các thao tác cập nhật sẽ được lan tỏa đến các bản sao khác trong hệ thống.

### 8.3.2 Mô hình nhất quán đọc đều

Mô hình nhất quán đọc đều phải đảm bảo điều kiện tiên trình thực hiện thao tác đọc trên một mục dữ liệu thì những thao tác đọc tiếp theo sẽ cho kết quả không cũ hơn kết quả lần trước. Máy khách sẽ không bao giờ phải nhìn thấy những dữ liệu cũ hơn những gì mà mình đã đọc trước đó, điều này có nghĩa là khi máy khách thực hiện một thao tác đọc trên một bản sao rồi tiếp theo lại đọc trên một bản sao khác thì bản sao thứ hai ít nhất cũng phải được ghi giống với bản sao đầu tiên.

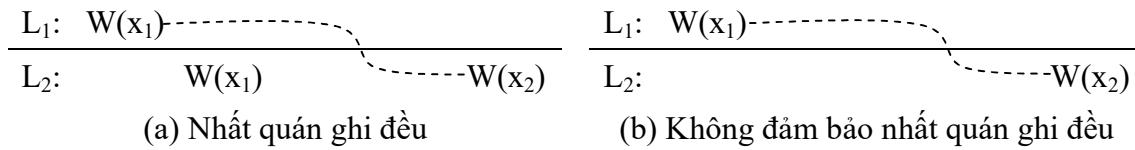


Hình 8.17 Mô hình nhất quán đọc đều

Hình 8.17 minh họa ví dụ về hệ thống lưu trữ gồm hai bản sao  $L_1$  và  $L_2$ , ký hiệu  $WS(x_i)$  là thao tác ghi  $x_i$  vào mục dữ liệu,  $WS(x_i, x_{i+1})$  là thao tác bao gồm cả thao tác ghi  $x_i$  vào mục dữ liệu. Trường hợp (a), tiên trình cập nhật dữ liệu lên bản sao  $L_1$  sau đó lan tỏa đến bản sao  $L_2$ , tại bản sao  $L_2$ , thao tác ghi  $x_1$  là phần đầu của thao tác  $WS(x_1, x_2)$  nên máy khách đọc bản sao trên  $L_2$  chắc chắn sẽ nhận được giá trị  $x_2$ , như vậy thỏa mãn yêu cầu nhất quán đọc đều. Trường hợp (b) bản sao  $L_2$  chỉ thực hiện thao tác ghi  $WS(x_2)$  sau đó đọc được giá trị  $x_2$ , tuy nhiên thao tác  $WS(x_1)$  chưa được thực hiện trên  $L_2$  nên không có gì đảm bảo sau này không đọc được giá trị  $x_1$ , vì vậy trường hợp này không đáp ứng yêu cầu nhất quán đọc đều.

### 8.3.3 Mô hình nhất quán ghi đều

Trong nhiều trường hợp, thứ tự thực hiện các thao tác cập nhật của một tiên trình phải được lan tỏa đến các bản sao theo đúng thứ tự. Mô hình nhất quán ghi đều phải đảm bảo điều kiện thao tác ghi lên mục dữ liệu  $x$  của một tiên trình phải được hoàn thành trước bất kỳ một thao tác ghi nào khác trên  $x$  bởi cùng tiên trình đó, nói cách khác các thao tác ghi của một tiên trình lên một mục dữ liệu sẽ được sắp xếp theo thứ tự đã đề ra trong tiên trình.



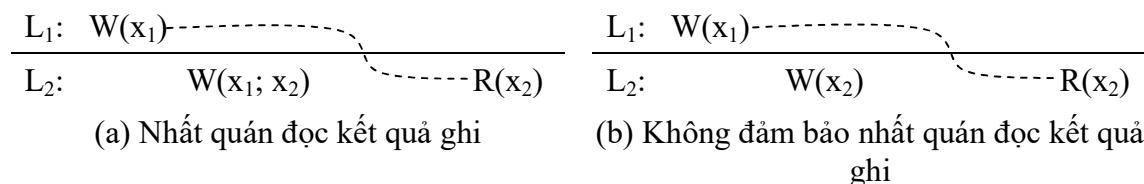
Hình 8.18 Mô hình nhất quán ghi đều

Như vậy, nhất quán ghi đều giống như nhất quán hàng đợi trong trường hợp áp dụng cho một tiên trình. Ví dụ trên hình 8.18 tiên trình thực hiện thao tác cập nhật  $W(x_1)$  và  $WS(x_2)$ , trường hợp (a) bản sao  $L_3$  nhận thực hiện cả hai lệnh này theo thứ tự nên nó đáp ứng yêu cầu nhất quán ghi đều. Trường hợp (b) bản sao  $L_2$  chỉ thực hiện lệnh cập

nhật  $W(x_2)$ , như vậy lệnh cập nhật  $W(x_1)$  có thể sẽ đến sau và do đó không đảm bảo yêu cầu nhất quán ghi đè.

#### **8.3.4 Nhất quán đọc kết quả ghi**

Trong mô hình nhất quán này, người dùng được đảm bảo rằng sẽ luôn được nhìn thấy những kết quả ghi mới nhất. Mô hình nhất quán đọc kết quả ghi phải thỏa mãn điều kiện kết quả thao tác ghi của một tiến trình lên mục dữ liệu x sẽ luôn được nhìn thấy bởi một thao tác đọc kế tiếp tiến trình đó trên x, nói cách khác thao tác ghi phải được hoàn thành trước bất kỳ thao tác đọc kế tiếp nào của cùng tiến trình ở bất cứ bản sao nào. Mô hình nhất quán đọc kết quả ghi gần giống với mô hình nhất quán đọc đều, điểm khác biệt chỉ ở chỗ tính nhất quán bây giờ được quyết định bằng thao tác ghi của tiến trình chứ không phải thao tác đọc.

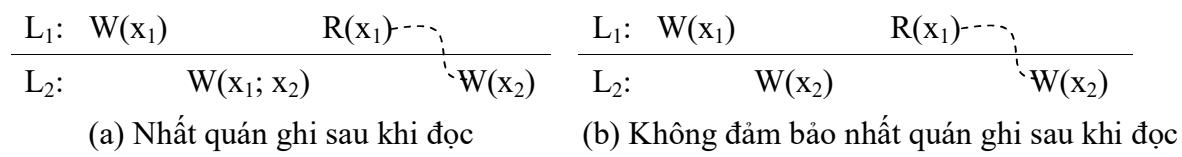


Hình 8.19 Mô hình nhất quán đọc kết quả ghi

Ví dụ trường hợp máy chủ web sử dụng cơ chế cache, khi biên tập viên đã hoàn thành cập nhật bài viết nhưng người dùng vẫn không nhìn thấy nội dung mới cập nhật, nguyên nhân là do máy chủ web trả về giá trị lưu trong cache. Nhất quán đọc kết quả ghi đảm bảo rằng khi biên tập viên cập nhật nội dung của mục dữ liệu thì cache cũng được cập nhật. Hình 8.19 minh họa tiến trình thực hiện thao tác ghi  $W(x_i)$  sau đó thực hiện thao tác đọc  $R(x_i)$  trên bản sao khác, trường hợp (a) thao tác  $W(x_1; x_2)$  thể hiện thao tác ghi  $W(x_1)$  là một phần của thao tác ghi  $W(x_2)$ , trường hợp (b) thể hiện thao tác ghi  $W(x_1)$  chưa được lan tỏa đến bản sao  $L_2$  do đó không đảm bảo tính nhất quán giữa hai bản sao này.

### **8.3.5 Nhất quán ghi sau khi đọc**

Mô hình nhất quán ghi sau khi đọc đảm bảo rằng người dùng sẽ luôn thực hiện thao tác ghi lên một mục dữ liệu mà ít nhất cũng phải mới hơn phiên bản cuối cùng của mục dữ liệu đó, lan truyền cập nhật phải dựa trên kết quả đọc.



Hình 8.20 Mô hình nhất quán ghi sau khi đọc

Hình 8.20 minh họa tiến trình thực hiện thao tác ghi  $W(x_i)$  sau đó thực hiện thao tác đọc  $R(x_i)$  trên bản sao khác, trường hợp (a) thao tác  $W(x_1;x_2)$  thể hiện thao tác ghi  $W(x_1)$  đã được lan tỏa đến bản sao  $L_2$ , tiến trình đọc được kết quả thao tác ghi  $W(x_1)$  sau đó mới thực hiện thao tác ghi  $W(x_2)$ . Trường hợp (b) thể hiện thao tác ghi  $W(x_1)$  chưa

được lan tỏa đến bản sao L<sub>2</sub> do đó không đảm bảo tính nhất quán giữa hai bản sao này. Có thể lấy ví dụ về thảo luận nhóm, các thành viên chỉ có thể đưa ra góp ý sau khi đã nhìn thấy chủ đề thảo luận.

## 8.4 Quản lý các bản sao

Trước khi nhân bản trong các hệ thống phân tán cần phải trả lời các câu hỏi: ở đâu, khi nào và cho ai và sau đó là các cơ chế dùng để duy trì tính nhất quán của các bản sao. Câu hỏi đặt bản sao ở đâu sẽ gồm hai phần: vị trí máy chủ của bản sao và nội dung được lưu trữ trên bản sao đó, sự khác biệt không dễ mô tả nhưng quan trọng và chúng không tách biệt nhau một cách rõ ràng. Vị trí đặt máy chủ bản sao liên quan tới việc tìm kiếm vị trí tốt nhất để đặt máy chủ lưu trữ dữ liệu, nội dung được lưu trữ liên quan đến vấn đề tìm kiếm máy chủ tốt nhất để lưu trữ nội dung, thông thường đó là tìm kiếm vị trí tối ưu cho mục dữ liệu.

### 8.4.1 Vị trí máy chủ bản sao

Lựa chọn vị trí đặt máy chủ bản sao dựa trên kết quả phân tích các thuộc tính về mạng và máy khách sao cho độ trễ truyền thông giữa máy chủ và máy khách có thể đạt giá trị thấp nhất. Tuy nhiên trong thực tế còn phải tính đến các vấn đề thương mại chứ không phải chỉ đơn thuần vấn đề tối ưu truyền thông khách/chủ. Có nhiều cách khác nhau để tính toán vị trí tốt nhất đặt các máy chủ bản sao nhưng tựu chung lại đều là vấn đề tối ưu tìm ra K vị trí tốt nhất trong số N vị trí, những vấn đề tính toán này phức tạp và chỉ có thể được giải quyết thông qua kinh nghiệm.

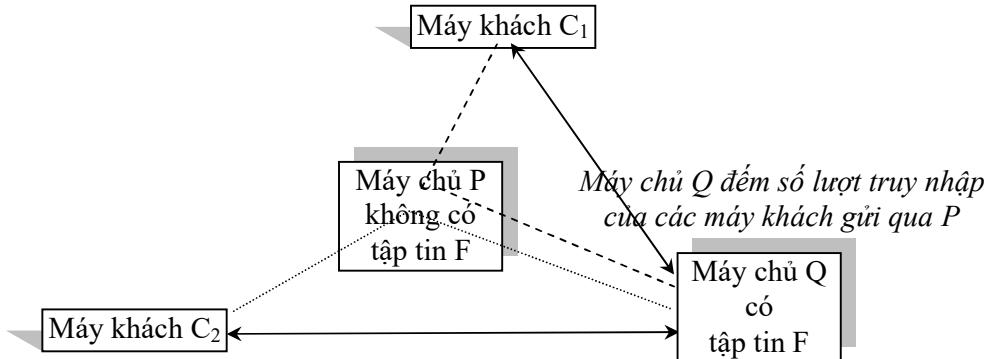
Ví dụ, có thể lấy khoảng cách giữa máy khách và máy chủ để làm tiêu chí nhưng cũng có thể lấy tiêu chí tối ưu là vùng tự trị, các giải thuật này đều có chi phí tính toán cao. Năm 2006 Szymaniak đề xuất giải pháp bản sao theo khu vực, một khu vực được xác định bằng tập các nút truy nhập cùng nội dung, tuy nhiên cần phải tính toán sao cho hạn chế sự phân mảnh nội dung.

### 8.4.2 Nhân bản nội dung và vị trí

Khi nói về nhân bản nội dung và vị trí, các bản sao nội dung có thể chia thành ba loại: bản sao thường trực, bản sao khởi đầu từ máy chủ và các bản sao khởi đầu từ máy khách. Các bản sao thường trực có thể coi như tập khởi đầu của các bản sao, chúng tạo nên kho dữ liệu phân tán. Số lượng các bản sao thường trực thường nhỏ, ví dụ khi cài đặt trang web có thể tổ chức thành nhóm các máy chủ web tại một vị trí hoặc các máy chủ phân tán. Cách thứ nhất sao chép các tập tin của trang web lên các nhóm các máy chủ, khi có yêu cầu gửi đến thì sẽ sử dụng chiến lược round-robin để chuyển tới một trong các máy chủ xử lý. Cách thứ hai là dùng cơ chế phản chiếu, trang web sẽ được nhân bản lên một số máy chủ phân bố dựa theo vị trí địa lý trên toàn mạng Internet, máy khách lựa chọn một trong các máy chủ để truy nhập trang web.

Bản sao khởi nguồn từ máy chủ được sử dụng để làm tăng hiệu năng và chúng được tạo ra xuất phát từ yêu cầu của kho dữ liệu, các bản sao này được xếp đặt động dựa vào yêu cầu của máy chủ khác. Ví dụ máy chủ web đặt tại một vị trí, bình thường vẫn xử lý các yêu cầu nhưng quản trị viên nhận thấy có sự tăng trưởng đột ngột từ một khu vực nào đó, như vậy có thể nhân bản trang web tạm thời lên các máy chủ ở những khu vực

có nhiều người truy nhập, quyết định đặt nội dung ở đâu sẽ dễ dàng hơn quyết định đặt máy chủ.



Hình 8.21 Đếm số lượng yêu cầu từ máy khách

Giải thuật nhân bản động giải quyết vấn đề giảm tải cho máy chủ và đưa thông tin cần thiết gần với vị trí của người sử dụng, mỗi máy chủ lưu vết truy nhập của người sử dụng, dựa trên thông tin về nội dung và vị trí truy nhập có thể quyết định máy chủ nào gần với người sử dụng nhất. Trên hình 8.21, tập tin F đặt tại máy chủ Q nhưng nhiều người sử dụng truy nhập đến tập tin này phải đi qua máy chủ P, nếu số lượng truy nhập qua P lớn hơn ngưỡng nhân bản thì sẽ tạo bản sao tập tin F trên máy chủ P. Nếu số lượng truy nhập đến tập tin F trên máy chủ Q không vượt qua ngưỡng xóa thì hoàn toàn có thể loại bỏ tập tin này trên máy chủ Q, như vậy sẽ giảm số lượng các bản sao.

Ngưỡng nhân bản luôn có giá trị lớn hơn ngưỡng xóa, số lượng yêu cầu lớn hơn ngưỡng nhân bản nghĩa là sẽ tốt hơn nếu đặt tài nguyên trên máy chủ khác. Nếu số lượng yêu cầu nằm trong khoảng ngưỡng xóa và ngưỡng nhân bản thì nên tạo thêm bản sao cho tài nguyên. Tuy nhiên việc di chuyển tài nguyên có kích thước lớn từ máy chủ này sang máy chủ khác không phải lúc nào cũng thực hiện được, trong trường hợp này phải thực hiện cơ chế nhân bản. Bản sao thường trực vẫn thường được sử dụng với mục đích dự phòng hoặc với mục đích đảm bảo tính nhất quán, các bản sao khởi nguồn từ máy chủ dùng để thay thế các bản sao chỉ đọc cho máy khách.

Bản sao khởi nguồn từ máy khách được tạo ra từ yêu cầu của những ứng dụng nằm trên chính máy khách, chẳng hạn như việc ghi nhớ dữ liệu cache, bản chất đó là vùng nhớ để máy khách tạm thời lưu dữ liệu vừa yêu cầu. Về nguyên tắc, việc quản lý cache hoàn toàn dành cho máy khách, kho dữ liệu không có việc gì phải duy trì tính nhất quán, tuy nhiên trong nhiều trường hợp máy khách có thể dựa vào sự tham gia của kho dữ liệu để thông báo cho nó khi nào dữ liệu cache đã cũ. Dữ liệu cache của máy khách dùng để cải thiện thời gian truy nhập dữ liệu, chúng được xếp đặt động tùy thuộc vào yêu cầu của người sử dụng các ứng dụng trên máy khách.

Bình thường khi máy khách truy nhập dữ liệu nào đó, nó kết nối đến bản sao gần nhất của kho dữ liệu đã lấy ra và muốn đọc hoặc nơi nó đã thay đổi. Nếu là hệ thống khai phá dữ liệu, hiệu năng có thể cải thiện bằng cách cho phép máy khách lưu dữ liệu đã yêu cầu trong cache ở vị trí gần, như vậy có thể lưu trên chính máy khách hoặc trên máy tính trong mạng cục bộ cùng với máy khách, những lần kế tiếp dữ liệu sẽ được lấy từ cache

cục bộ, phương pháp này sẽ hoạt động tốt nếu không có bất kỳ sự thay đổi dữ liệu nào trong khoảng thời gian đó. Nói chung dữ liệu trong cahe được giữ một thời gian nhất định để tránh dữ liệu quá cũ hoặc để dành bộ nhớ cho các máy khác khác.

### **8.4.3 Phân phát nội dung**

Quản lý bản sao cũng phải giải quyết vấn đề lan truyền cập nhật nội dung đến các máy chủ nhân bản tương ứng. Thay đổi dữ liệu trên một bản sao thì phải lan tỏa đến các bản sao khác sao cho ít ảnh hưởng nhất đến hiệu năng của hệ thống cung cấp dịch vụ cho người sử dụng. Để giải quyết vấn đề này cần trả lời cho câu hỏi lan tỏa trạng thái hay thao tác, cách thức chuyển dữ liệu giữa các bản sao.

#### *8.4.3.1 Trạng thái hay thao tác*

Thao tác cập nhật làm thay đổi dữ liệu, nghĩa là trạng thái của bản sao đã thay đổi, như vậy có thể lan tỏa ba loại thông tin: trạng thái, dữ liệu mới hoặc câu lệnh cập nhật. Lan truyền trạng thái là hình thức thông báo mục dữ liệu hết hiệu lực, các bản sao khác cần phải thực hiện thao tác đồng bộ, hình thức phụ thuộc vào mô hình nhất quán cụ thể được hỗ trợ. Ưu điểm chính của phương pháp này là lượng thông tin nhỏ nên sử dụng rất ít băng thông, chỉ có thông tin xác định những dữ liệu nào không còn hiệu lực được truyền đi.

Hình thức thức lan tỏa trạng thái hoạt động tốt trong các hệ thống giao dịch trực tuyến, số lượng các thao tác cập nhật lớn hơn rất nhiều so với các thao thác đọc, nghĩa là tỉ lệ thao tác đọc so với thao tác ghi tương đối nhỏ. Hình thức này cũng phù hợp với những trường hợp dung lượng dữ liệu cập nhật lớn, chuyển dữ liệu đến các bản sao sẽ phải chi phí lượng lớn thời gian và như vậy sẽ ảnh hưởng đến hiệu năng của hệ thống. Nếu hai lệnh cập nhật liên tiếp xảy ra trên cùng một mục dữ liệu, kết quả thao tác cập nhật thứ hai sẽ ghi đè lên kết quả cập nhật thứ nhất, do đó việc thông báo cập nhật sẽ hiệu quả hơn nhiều so với hai hình thức còn lại.

Chuyển dữ liệu đã bị thay đổi giữa các bản sao là hình thức thứ hai, nó thực hiện tốt trong các hệ thống khai phá dữ liệu, nghĩa là khi tỉ lệ thao tác đọc so với thao tác ghi tương đối cao. Có thể cải tiến bằng cách ghi lại những thay đổi trên mục dữ liệu và chỉ truyền những thay đổi đó để tiết kiệm băng thông, hơn nữa có thể tập hợp chúng thành một thông điệp và chuyển đi, như vậy cũng sẽ tiết kiệm thông tin điều khiển.

Hình thức thứ ba, thông báo cho mỗi bản sao biết cần phải thực hiện thao tác cập nhật nào, chỉ truyền đi những giá trị của tham số cần thiết cho những thao tác đó. Hình thức này còn gọi là nhân bản chủ động, với giả thiết mỗi bản sao đại diện cho tiến trình có khả năng đảm bảo dữ liệu có liên quan sẽ được cập nhật bằng cách thực hiện các thao tác. Ưu điểm của phương pháp này là việc lan truyền cập nhật có thể được thực hiện với chi phí băng thông tối thiểu, dung lượng dữ tham số liên quan tới thao tác tương đối nhỏ. Hơn nữa các thao tác có thể phức tạp tùy ý, điều này cho phép cải thiện hơn nữa trong việc giữ tính nhất quán của các bản sao. Mặt khác mỗi bản sao có thể đáp ứng năng lực xử lý lớn hơn, đặc biệt trong những trường hợp các thao tác tương đối phức tạp.

#### 8.4.3.2 Hình thức đẩy hay kéo

Một vấn đề khác khi thiết kế cơ chế nhân bản nội dung là lựa chọn hình thức đẩy hay kéo, hình thức đẩy nghĩa là một bản sao chủ động thông báo cho các bản sao khi xuất hiện thao tác cập nhật, hình thức kéo nghĩa là mỗi bản sao chủ động kiểm tra những thay đổi trên các bản sao khác. Hình thức đẩy phù hợp với các hệ thống đòi hỏi tính nhất quán rất cao, thay đổi trên mỗi bản sao sẽ lan tỏa ngay lập tức đến các bản sao khác, điều đó cho thấy nó phù hợp cho các hệ thống số lượng bản sao ít. Hình thức này thường được sử dụng cho các bản sao thường trực và các bản sao khởi nguồn từ máy chủ, đôi khi cũng được sử dụng cho các bản sao khởi nguồn từ máy khách.

Ví dụ trang tin điện tử được nhân bản trên nhiều máy chủ, số lượng máy chủ nhân bản thường không nhiều, thay đổi nội dung của một trang trên một máy chủ thì phải đồng bộ ngay lập tức đến các máy chủ khác. Kỹ thuật cache có thể áp dụng trên cả máy chủ lẫn máy khách, số lượng máy chủ web ít hơn số lượng máy khách rất nhiều, có thể sử dụng hình thức đẩy sẽ phù hợp với cache trên máy chủ nhưng không phù hợp cho cache trên máy khách.

Hình thức kéo nghĩa là một bản sao gửi yêu cầu đến bản sao khác để thực hiện đồng bộ, nếu có thay đổi thì tải về những dữ liệu đã thay đổi hoặc các thao tác cập nhật đã lưu trong nhật ký. Tính nhất quán của hình thức này rất thấp, nó phụ thuộc chu kỳ hoặc sự kiện kiểm tra, hình thức này thường dùng cho các bản sao khởi nguồn từ máy khách.

Ví dụ cơ chế cache trên máy khách, khi người sử dụng truy nhập lần thứ nhất, dữ liệu từ máy chủ được chuyển về máy khách và được lưu trong cache. Người sử dụng truy nhập lần thứ hai, nếu cache máy khách chưa quá hạn thì lấy ngay nội dung trên máy tính đó, nếu quá hạn thì mới gửi yêu cầu lên máy chủ. Nếu nội dung trên cache máy chủ không mới, máy khách sẽ lấy dữ liệu từ cache cục bộ, nói cách khác máy khách đã thăm dò máy chủ để biết có cần phải cập nhật hay không.

Cách tiếp cận kéo sẽ hiệu quả khi tỉ lệ giữa thao tác đọc và thao tác ghi tương đối thấp, đó thường là trường hợp cache riêng của máy khách hoặc cache chung của nhiều máy khách nhưng ít chia sẻ các mục dữ liệu. Bảng 8.1 so sánh hình thức đẩy và kéo, nhược điểm cơ bản của hình thức kéo là thời gian phản hồi sẽ tăng lên trong trường hợp mất cache.

**Bảng 8.1** So sánh hình thức đẩy và kéo

Mục so sánh	Đẩy	Kéo
Trạng thái bản sao	Danh sách trạng thái các bản sao	Không cần
Thông điệp trao đổi	Cập nhật	Thăm dò và cập nhật
Thời gian phản hồi	Ngay lập tức	Thời gian tải dữ liệu

Hình thức đẩy đòi hỏi mỗi bản sao phải lưu trữ trạng thái của tất cả các bản sao khác, điều này làm tăng đáng kể lượng dữ liệu chứa thông tin điều khiển. Ví dụ máy chủ web, khi cập nhật trang thì máy chủ sẽ phải thực hiện cập nhật toàn bộ cache của máy khách,

điều này làm tăng đáng kể lưu lượng truyền thông. Thông điệp trao đổi giữa máy khách và máy chủ cũng khác nhau, trong cách tiếp cận đẩy thì chỉ có máy chủ gửi các thao tác cập nhật cho máy khách, chỉ khi nào đó là những yêu cầu cập nhật thì máy khách mới phải lấy những dữ liệu đã thay đổi.

Trong hình thức kéo, máy khách phải thăm dò máy chủ xem có cần thiết lấy dữ liệu đã bị thay đổi hay không. Cuối cùng, thời gian trả lời tại máy khách cũng khác nhau, nếu máy chủ đẩy dữ liệu thay đổi về máy khách thì thời gian đáp ứng tại phía máy khách gần như bằng 0, nếu dữ liệu trên máy khách không còn hiệu lực thì thời gian đáp ứng sẽ giống như cách tiếp cận kéo, đó là thời gian tải dữ liệu từ máy chủ về máy khách.

Việc cân bằng hai cách tiếp cận này dẫn đến một dạng lai ghép lan truyền cập nhật dựa trên khái niệm thời gian quá hạn cho nội dung, máy chủ sẽ đẩy cập nhật đến các máy khách sau một thời gian nhất định, khi quá thời gian đó thì máy khách sẽ buộc hỏi máy chủ về cập nhật và nếu có dữ liệu thay đổi thì kéo về hoặc máy khách sẽ gia hạn thêm thời gian đẩy cập nhật. Việc tính toán thời gian quá hạn dựa trên các tiêu chí khác nhau, ví dụ có thể dựa trên thời gian cập nhật cuối cùng, nếu một mục dữ liệu nào đó không thay đổi trong một khoảng thời gian dài thì có thể tăng thời gian quá hạn, như vậy sẽ giảm đáng kể các thông điệp cập nhật.

Cũng có thể dựa vào số lượng yêu cầu của máy khách để quyết định thời gian quá hạn hoặc thậm chí quyết định có lưu trạng thái của máy khách hay không, tần suất yêu cầu càng nhiều chứng tỏ máy khách càng cần phải được đảm bảo tính nhất quán. Những máy khách đã được máy chủ lưu trạng thái nếu càng gửi nhiều yêu cầu thì càng tăng thời gian quá hạn, yêu cầu cập nhật vẫn sẽ được chuyển đến các máy khách này khi có sự thay đổi nội dung mặc dù chưa hết thời gian quá hạn. Tiêu chí cuối cùng là không gian trạng thái tại máy chủ, khi máy chủ quá tải thì phải hạ thấp thời gian quá hạn để máy chủ giảm số lượng máy khách cần phải lưu trạng thái, thậm chí máy chủ có thể chuyển sang dạng không trạng thái, nghĩa là sẽ không lưu trạng thái của bất kỳ máy khách nào.

#### 8.4.3.3 Phương pháp lan truyền cập nhật

Liên quan tới hình thức đẩy hay kéo cập nhật thì sẽ quyết định sử dụng phương pháp truyền thông điểm-điểm hay truyền theo nhóm. Nếu sử dụng phương pháp truyền thông điểm-điểm nhất thì sẽ phải gửi thông điệp cập nhật riêng rẽ cho từng bản sao, nếu sử dụng phương pháp truyền thông theo nhóm thì các thông điệp sẽ dựa trên hạ tầng mạng để gửi đến các bản sao một cách hiệu quả.

Đa số các trường hợp sử dụng các phương tiện truyền theo nhóm sẵn có, nếu các bản sao nằm trong mạng cục bộ thì sự khác biệt giữa các phương pháp truyền không đáng kể. Phương pháp truyền theo nhóm thường có hiệu quả khi sử dụng kèm với hình thức đẩy để lan truyền cập nhật, nếu tích hợp cẩn thận thì chỉ cần sử dụng một nhóm. Ngược lại, cách tiếp cận kéo thông thường do một máy khách hoặc máy chủ yêu cầu bản sao được cập nhật, trong trường hợp này giải pháp truyền điểm-điểm sẽ hiệu quả nhất.

### 8.5 Các giao thức nhất quán

Các mô hình nhất quán và những vấn đề thiết kế giao thức nhất quán đã được giới thiệu trong những phần trên, phần này sẽ tập trung cho việc cài đặt thực tế các mô hình

nhất quán bằng cách triển khai một số giao thức nhất quán. Giao thức nhất quán mô tả cách cài đặt mô hình nhất quán riêng, trước hết sẽ xem xét dưới khía cạnh mô hình nhất quán lấy dữ liệu làm trung tâm sau đó sẽ đề cập tới các giao thức cho mô hình lấy máy khách làm trung tâm.

### 8.5.1 Nhất quán liên tục

Các giao thức nhất quán liên tục dựa trên các giải pháp của mô hình nhất quán liên tục, Yu và Vahdat trong các công trình nghiên cứu về tính nhất quán liên tục đã phát triển một số giao thức để giải quyết ba tiêu chí nhất quán, đó là những tiêu chí về sai số giữa các bản sao, chênh lệch trạng thái và thứ tự các thao tác cập nhật.

#### 8.5.1.1 Giới hạn sai số

Giả sử cần cập nhật mục dữ liệu  $x$ , mỗi thao tác cập nhật  $W(x)$  được gán trọng số  $\text{weight}(W)$ , để đơn giản giả thiết  $\text{weight}(W) > 0$ . Mỗi thao tác ghi ban đầu được chuyển đến một trong số  $N$  máy chủ bản sao và máy chủ này trở thành gốc của thao tác ghi, ký hiệu  $\text{origin}(W)$ . Nếu xét hệ thống tại một thời điểm sẽ thấy một số cập nhật cần phải được lan tỏa đến tất cả các máy chủ. Mỗi máy chủ  $S_i$  sẽ lưu vết nhật ký  $L_i$  của các thao tác cập nhật được thực hiện trên bản sao cục bộ của mục dữ liệu  $x$ . Giả sử  $TW[i,j]$  là các thao tác cập nhật được thực hiện trên máy chủ  $S_i$  với yêu cầu xuất phát từ máy chủ  $S_j$ :

$$TW[i,j] = \sum \{\text{weight}(W) \mid \text{origin}(W) = S_j \& W \in \log(S_i)\}$$

Mục đích cần đạt là với bất kỳ thời gian t nào thì giá trị  $V_i$  tại máy chủ  $S_i$  xê dịch trong khoảng giá trị thực  $v(t)$  của mục dữ liệu  $x$ , giá trị thực này được quyết định bởi tất cả các thao tác cập nhật, nghĩa là nếu  $v(0)$  là giá trị ban đầu của mục dữ liệu  $x$  thì giá trị thực  $v(t)$  của mục dữ liệu  $x$  được tính theo công thức:

$$v(t) = v(0) + \sum_{k=1}^N TW[k,k]$$

và giá trị  $V_i$  của bản sao i:

$$v(i) = v(0) + \sum_{k=1}^N TW[i,k]$$

Tập trung vào độ lệch tuyệt đối, kết hợp cận trên  $\delta_i$  cho mỗi máy chủ  $S_i$ , như vậy cần phải đảm bảo  $v(t) - v_i \leq \delta_i$ . Các thao tác cập nhật gửi đến máy chủ  $S_i$  sẽ phải lan truyền đến các máy chủ khác, có nhiều cách thực hiện nhưng thông thường sử dụng giao thức bệnh dịch sẽ cho phép phổ biến các cập nhật nhanh nhất, ý tưởng cơ bản của thuật toán lan truyền như sau:

- Giả thiết không xảy ra xung đột giữa các thao tác ghi-ghi.
- Các thao tác cập nhật ban đầu được thực hiện chỉ trên một số bản sao.
- Một bản sao chỉ gửi các cập nhật của nó tới tập hữu hạn hàng xóm.
- Việc lan truyền các cập nhật xảy ra chậm chạp và không phải ngay lập tức.
- Cuối cùng thì mỗi cập nhật cũng đến được từng bản sao.

Dựa trên phương pháp lan truyền bệnh dịch, điều đáng lưu ý trong mô hình này là các cập nhật được lan truyền tới các bản sao với số lượng thông điệp càng ít càng tốt và càng nhiều bản sao nhận được các lan truyền càng nhanh càng tốt, cuối cùng nếu bản sao

nào không lan truyền được cập nhật của mình thì nó sẽ bị loại bỏ. Một trong những mô hình lan truyền cập nhật gọi là anti entropy, mỗi bản sao cứ định kì lại chọn ngẫu nhiên một bản sao khác và trao đổi các trạng thái khác nhau của mình, sau một thời gian thì cả hai bên sẽ có những trạng thái giống hệt nhau. Một mô hình khác là gossiping, trong mô hình này mỗi bản sao thực hiện các thao tác cập nhật sẽ gửi cho các bản sao khác những cập nhật đó.

Trong mọi trường hợp, khi máy chủ  $S_i$  lan truyền cập nhật khởi nguồn từ máy chủ  $S_j$  đến máy chủ  $S_k$  thì máy chủ sau sẽ có thể học được giá trị  $TW[i,j]$  tại thời điểm gửi thao tác cập nhật, nói cách khác  $S_k$  có thể duy trì cách nhìn nhận  $TW_k[i,j]$  mà nó tin cậy  $S_j$ , sẽ có giá trị  $TW[i,j]$ , rõ ràng  $0 \leq TW_k[i,j] \leq TW[i,j] \leq TW[j,j]$ . Khi máy chủ  $S_k$  nhận thấy máy chủ  $S_j$  không giữ được nhịp độ cập nhật đã chuyển đến  $S_k$ , nghĩa là  $TW_k[i,k]$  khác xa  $TW[k,k]$ , đặc biệt nếu  $TW[k,k]-TW_k[i,k] > \delta_j/(N-1)$ , máy chủ  $S_k$  sẽ chuyển tiếp các thao tác cập nhật đã lưu trong nhật ký của nó đến  $S_j$ . Việc chuyển tiếp này sẽ cải thiện đáng để cách nhìn  $TW_k[i,k]$  mà  $S_k$  đã có của  $TW[i,k]$ , làm cho độ lệch  $TW[i,k] - TW_k[i,k]$  nhỏ hơn. Đặc biệt,  $S_k$  cải thiện cách nhìn của nó trên  $TW[i,k]$  khi ứng dụng đệ trình thao tác cập nhật mới làm tăng  $TW[k,k]-TW_k[i,k]$  vượt quá  $\delta_j/(N-1)$ , việc cải thiện luôn đảm bảo  $v(t) - v_i \leq \delta_i$ .

#### *8.5.1.2 Giới hạn chênh lệch trạng thái*

Có nhiều cách để giữ trạng thái của các bản sao trong giới hạn xác định, một trong những cách đơn giản là cho phép máy chủ  $S_k$  giữ đồng hồ vector theo thời gian thực  $RVC_k$  trong đó  $RVC_k[i] = T(i)$ , nghĩa là máy chủ  $S_k$  đã nhìn thấy tất cả các thao tác cập nhật đã gửi cho nó đến thời điểm  $T(i)$ , với giả thiết các thao tác cập nhật được gán nhãn thời gian và  $T(i)$  là nhãn thời gian cục bộ của máy chủ  $S_i$ . Khi máy chủ  $S_k$  nhận thấy giá trị  $T(i)-RVC_k[i]$  vượt quá giới hạn xác định thì nó chỉ cần kéo các thao tác cập nhật xuất phát từ  $S_i$  với nhãn thời gian lớn hơn  $RVC_k[i]$ . Trong trường hợp này, máy chủ bản sao có trách nhiệm giữ cho mục dữ liệu được cập nhật mới nhất bất chấp các thao tác cập nhật xuất phát từ đâu, ngược lại với giải pháp giới hạn sai số cho phép máy chủ gốc giữ các bản sao luôn được cập nhật mới nhất bằng cách chuyển tiếp các thao tác cập nhật. Trường hợp giới hạn chênh lệch trạng thái, thao tác đầy cập nhật không đảm bảo tính nhất quán vì không biết trước thời gian lan truyền cập nhật tối đa sẽ là bao nhiêu, vì vậy thao tác kéo cập nhật sẽ cải thiện tình huống này do nhiều máy chủ cùng giúp giữ cho bản sao mục dữ liệu được cập nhật mới nhất.

#### *8.5.1.3 Giới hạn độ lệch thứ tự*

Nguyên nhân độ lệch thứ tự trong nhất quán liên tục là do máy chủ bản sao đang tạm thời thực hiện những thao tác cập nhật đã gửi cho nó, nghĩa là các thao tác này chưa được cam kết, chúng đang ở trong hàng đợi cục bộ và thứ tự thực sự của chúng đang phải chờ để được quyết định, độ lệch thứ tự được giới hạn bằng cách xác định chiều dài tối đa của hàng đợi các thao tác cập nhật chưa được cam kết. Như vậy việc phát hiện vi phạm nhất quán thứ tự xảy ra khi chiều dài hàng đợi này vượt quá chiều dài tối đa xác định trước, tại thời điểm đó máy chủ sẽ không tiếp nhận thêm các thao tác cập nhật mới gửi đến, máy chủ sẽ cố gắng thực hiện cam kết bằng cách đàm phán với các máy chủ khác về

thứ tự của các thao tác đang chờ cam kết. Nói cách khác, cần phải bắt buộc các máy chủ nhất quán toàn cục thứ tự các thao tác đang chờ cam kết, điều này trong thực tế thực hiện bằng các giao thức dựa trên bản chính hoặc giao thức dựa trên việc biểu quyết.

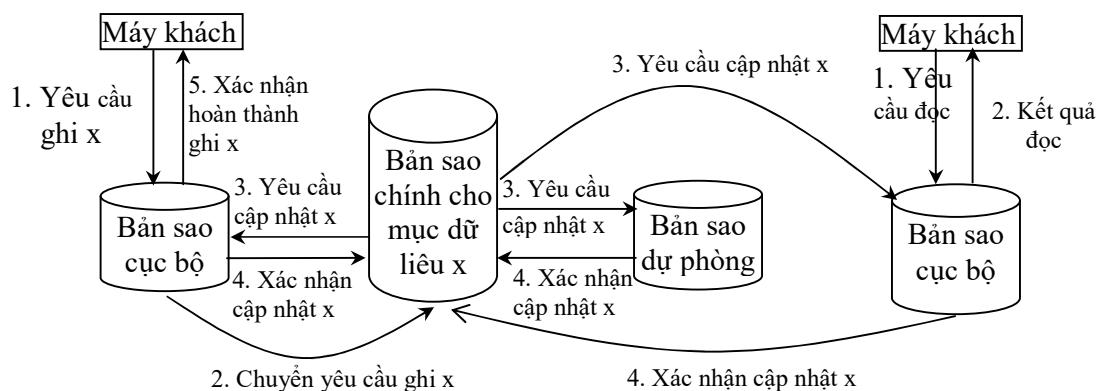
### 8.5.2 Các giao thức dựa trên bản chính

Trong thực tế, tính nhất quán trong các ứng dụng phân tán nói chung được xây dựng theo mô hình giới hạn độ lệch trạng thái hoặc độ lệch sai số bởi vì đó là những mô hình tương đối dễ hiểu. Khi nói đến các mô hình xử lý nhất quán theo thứ tự của các thao tác thì thông thường chúng được gộp thành nhóm thông qua khóa hoặc giao tác. Nếu mô hình nhất quán hơi khó hiểu, dù cho mô hình đó có cải thiện hiệu năng thì những người phát triển phần mềm cũng không muốn áp dụng. Điểm mấu chốt nằm ở chỗ nếu ngữ nghĩa của mô hình nhất quán không trực quan rõ ràng thì cũng sẽ tốn nhiều thời gian để xây dựng các ứng dụng chính xác, tính đơn giản cần phải được đề cao.

Trường hợp nhất quán tuần tự cho thấy giao thức dựa trên bản chính chiếm ưu thế, mỗi mục dữ liệu được gắn với một bản chính và bản chính đó có trách nhiệm điều phối thao tác ghi trên tất cả các bản sao của mục dữ liệu. Nhân bản dựa trên bản chính thuộc về nhóm nhất quán tuần tự, một bản sao dữ liệu được chỉ định đóng vai trò chủ đạo cập nhật dữ liệu, gọi là bản chính để phân biệt với các bản sao khác. Cần phân biệt trường hợp bản chính cố định trên một máy chủ với trường hợp các thao tác cập nhật có thể được thực hiện trên máy chủ cục bộ sau khi chuyển mục dữ liệu từ bản chính về tiến trình phát sinh thao tác cập nhật, trường hợp thứ nhất gọi là giao thức ghi từ xa và trường hợp thứ hai gọi là giao thức ghi cục bộ.

#### 8.5.2.1 Giao thức ghi từ xa

Giao thức dựa trên bản chính hỗ trợ nhân bản thực hiện theo nguyên lý tắt cả các thao tác cập nhật đều được chuyển về một máy chủ cố định, các thao tác đọc sẽ được thực hiện cục bộ, những giao thức này còn gọi là các giao thức sao lưu chính. Hình 8.22 minh họa trường hợp cần phải cập nhật mục dữ liệu x, tiến trình sẽ chuyển thao tác cập nhật cho tiến trình trên máy chủ chính, sau khi hoàn thành cập nhật cục bộ, tiến trình trên máy chủ chính sẽ trả về kết quả cho tiến trình khởi đầu thao tác cập nhật và đồng thời lan tỏa cập nhật này tới các máy chủ của các bản sao khác, các bản sao sau khi hoàn thành cập nhật cục bộ cũng sẽ xác nhận với máy chủ chính.



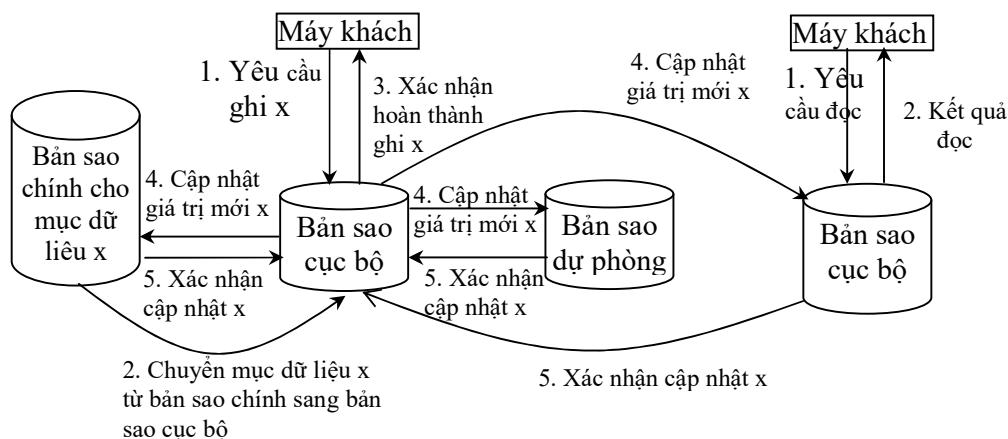
Hình 8.22 Giao thức ghi từ xa

Giao thức này thường được kết hợp với các hệ thống khách/chủ truyền thông, nhược điểm cơ bản của nằm ở vấn đề hiệu năng, thao tác cập nhật có thể thực hiện tương đối lâu, tiến trình khởi sướng yêu cầu cập nhật sẽ bị phong tỏa trong thời gian đó, có thể khắc phục bằng cách sử dụng cơ chế không phong tỏa. Tuy nhiên nếu sử dụng cơ chế không phong tỏa sẽ phải giải quyết các vấn đề liên quan đến khả năng chịu lỗi, nếu hoạt động dựa trên cơ chế phong tỏa thì tiến trình máy khách sẽ tin chắc thao tác cập nhật đã được hoàn thành trên tất cả các bản sao.

Giao thức dựa trên bản chính cung cấp khả năng cài đặt trực tiếp mô hình nhất quán tuần tự, máy chủ chính có thể sắp xếp tất cả các thao tác cập nhật theo thứ tự thời gian duy nhất trong toàn bộ hệ thống, kết quả là các tiến trình sẽ nhìn thấy các thao tác ghi theo cùng một thứ tự và sẽ không ảnh hưởng đến kết quả các thao tác đọc trên bất kỳ bản sao nào. Nếu sử dụng cơ chế phong tỏa thì các tiến trình luôn nhìn thấy kết quả của thao tác ghi gần nhất, điều này sẽ không được đảm bảo nếu sử dụng cơ chế không phong tỏa.

#### 8.5.2.2 Giao thức ghi cục bộ

Giao thức dựa trên bản chính có thể thực hiện theo cách khác, bản sao chính di chuyển giữa các tiến trình để thực hiện thao tác ghi. Hình 8.23 thể hiện hoạt động của giao thức này, khi tiến trình muốn cập nhật mục dữ liệu, nó lấy bản chính của mục dữ liệu và chuyển về máy cục bộ để xử lý, sau khi hoàn thành trên máy cục bộ sẽ lan tỏa thao tác cập nhật đến tất cả các bản sao khác. Ưu điểm của cách tiếp cận này là nhiều thao tác cập nhật liên tiếp có thể thực hiện cục bộ trong khi các tiến trình khác vẫn có thể đọc bản sao cục bộ của nó, có thể cải thiện hiệu năng bằng phương thức không phong tỏa. Giao thức này phù hợp cho các thiết bị di động hay những nơi chất lượng mạng kém, chúng có thể vận hành ngay cả khi không có kết nối mạng.



Hình 8.23 Giao thức ghi cục bộ

Trước khi mất kết nối, thiết bị di động trở thành máy chủ chính cho các mục dữ liệu cần cập nhật, khi không được nối vào mạng các thao tác cập nhật sẽ được thực hiện cục bộ trong khi tiến trình trên các máy khác chỉ được phép thực hiện thao tác đọc, sau khi kết nối lại các thao tác cập nhật sẽ được lan truyền đến các bản sao để đưa dữ

liệu về trạng thái nhất quán. Nên tổ chức theo cách chọn cố định một máy chủ trung tâm như trường hợp giao thức ghi từ xa, máy chủ tạm thời có thể cho phép một bản sao được phép thực hiện các thao tác cập nhật cục bộ, thực hiện xong thì chuyển về máy chủ trung tâm để lan tỏa đến các bản sao khác.

### **8.5.3 Các giao thức nhân bản cập nhật**

Trong các giao thức này, thay cho việc cập nhật trên một bản sao sau đó mới lan tỏa đến các bản sao khác, thao tác cập nhật có thể được tiến hành đồng thời trên nhiều bản sao. Một tiến trình sẽ gán định danh cho mỗi thao tác cập nhật và chuyển đến các bản sao, có thể thực hiện bằng theo phương pháp nhân bản tích cực hoặc biểu quyết theo đa số.

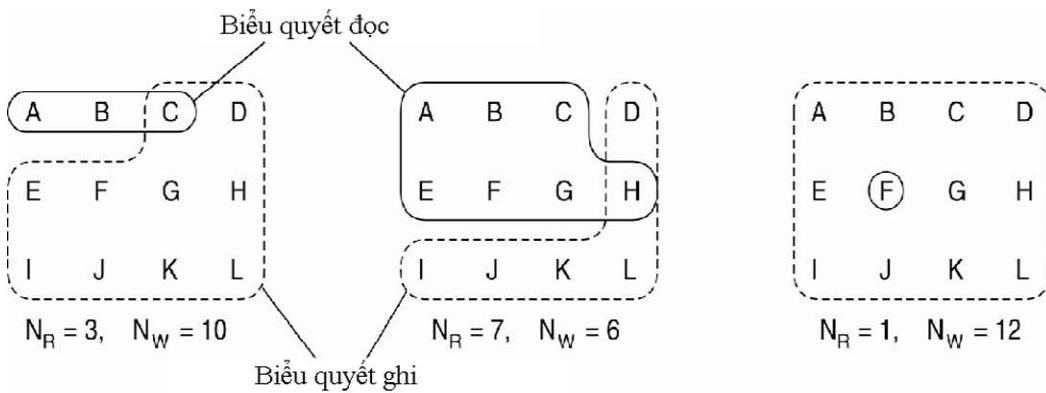
#### **8.5.3.1 Nhân bản tích cực**

Trong nhân bản tích cực, mỗi bản sao kết hợp với tiến trình thực hiện các thao tác cập nhật, các thao tác cập nhật được gửi đến đến tất cả các bản sao và được thực hiện theo cùng một thứ tự, các thao tác đọc được thực hiện cục bộ trên mỗi bản sao. Ưu điểm của phương pháp này là tất cả các bản sao đều đồng thời nhận được các thao tác cùng thứ tự và cũng không cần đánh dấu một bản chính hay phải gửi tất cả các thao tác tới một máy chủ. Tuy nhiên giao thức này lại đòi hỏi phải truyền theo kiểu nhóm động hoặc phải có một bộ sáp xếp dây tập trung mà cả hai phương pháp này đều khó có thể tiếp cận một cách linh hoạt.

Như vậy cần thiết phải sử dụng cơ chế truyền theo nhóm có thứ tự, điều này có thể thực hiện bằng cách gán nhãn thời gian logic, tuy nhiên cơ chế này không phù hợp với những hệ thống qui mô lớn, do đó có thể thay thế bằng cách sử dụng tiến trình điều phối và đồng thời đóng vai trò như một thành phần sản sinh số tuần tự. Các thao tác cập nhật sẽ được gửi đến tiến trình điều phối, tại đây chúng được gán số tuần tự và sau đó chuyển cho các bản sao, các bản sao sẽ thực hiện thao tác theo số tuần tự đã cấp. Việc sử dụng tiến trình điều phối cũng không giải quyết được vấn đề quy mô, trong thực tế cần phải kết hợp việc truyền theo nhóm với gán nhãn thời gian logic.

#### **8.5.3.2 Giao thức dựa trên đại diện**

Cách tiếp cận khác sử dụng kết quả bỏ phiếu để hỗ trợ thực hiện nhân bản thao tác ghi, máy khách yêu cầu các máy chủ cấp quyền được cấp nhập trước khi thao tác với mục dữ liệu, giao thức dựa trên đại diện thực hiện các thao tác ghi trên một tập nhỏ nhất các bản sao. Ví dụ để được quyền cập nhật mục dữ liệu, máy khách phải nhận được sự chấp thuận của đa số các máy chủ bản sao, khi đã được cấp quyền thì tiến hành cập nhật và đồng thời đánh dấu phiên bản cho mục dữ liệu để thao tác đọc có thể nhận biết được phiên bản mới nhất. Trong giao thức này, tất cả các mục dữ liệu được kết hợp với một số phiên bản, mỗi lần một mục bị sửa đổi thì số phiên bản của nó cũng được tăng lên. Khi thực hiện một thao tác đọc, máy khách cũng phải liên hệ với các bản sao để tìm ra phiên bản mới nhất của dữ liệu.



Hình 8.24 Giao thức dựa trên đại biểu

Gifford đã tổng quát hóa cách tiếp cận dựa trên đại diện, giao thức này định nghĩa ra số đại biểu đọc và số đại biểu ghi, hai đại biểu này sẽ xác định số bản sao phải được liên hệ trước khi thực hiện thao tác đọc và ghi. Số đại biểu đọc phải lớn hơn  $\frac{1}{2}$  tổng số bản sao, vì thế tổng của số đại biểu đọc và ghi phải lớn hơn tổng số bản sao. Bằng cách này, máy khách muốn thực hiện một thao tác đọc thì phải đảm bảo việc liên hệ với ít nhất một bản sao có chứa phiên bản mới nhất của mục dữ liệu. Việc lựa chọn số lượng đại biểu dựa vào tỉ lệ giữa thao tác đọc và ghi cùng với chi phí khi thực hiện phương pháp giao tiếp giữa các nhóm. Giả sử hệ thống gồm  $N$  bản sao, ký hiệu  $N_r$  là tập các máy chủ bản sao tùy ý để đọc và  $N_w$  là tập các máy chủ bản sao chấp thuận thao tác cập nhật, khi đó phải thỏa mãn hai điều kiện:

1.  $N_r + N_w > N$
2.  $N_w > N/2$

Điều kiện thứ nhất tránh xung đột đọc-ghi, điều kiện thứ hai tránh xung đột ghi-ghi, chỉ khi nào thỏa mãn các điều kiện này thì mới được phép đọc hoặc ghi. Ví dụ trên hình 8.24, trường hợp (a)  $N_r=3$  và  $N_w = 10$ , như vậy phiên bản mới nhất xuất hiện trên 10 bản sao, sẽ có ít nhất một thành viên trong tập  $N_r$  cũng sẽ là thành viên của tập  $N_w$  máy khách sẽ tìm được phiên bản mới nhất và tìm ra máy chủ bản sao nào chứa phiên bản mới nhất. Trường hợp (b) có thể xảy ra xung đột ghi-ghi vì  $N_w = N/2$ , trường hợp (c) thể hiện đọc một và ghi tất cả, có thể đọc bất kỳ máy chủ nào cũng được bản sao mới nhất.

#### 8.5.4 Giao thức gắn kết bộ nhớ cache

Bộ nhớ cache là trường hợp đặc biệt trong nhân bản, theo ngữ nghĩa, nói chung chúng chịu sự kiểm soát của máy khách chứ không phải máy chủ. Tuy nhiên, các giao thức gắn kết bộ nhớ cache đảm bảo chúng nhất quán với các bản sao khởi nguồn từ máy chủ, về nguyên lý các giao thức này không khác biệt nhiều so với các giao thức nhất quán. Đã có nhiều công trình nghiên cứu về việc thiết kế và cài đặt bộ nhớ cache, đặc biệt trong ngữ cảnh các hệ thống nhiều bộ xử lý chia sẻ bộ nhớ, nhiều giải pháp dựa vào sự hỗ trợ của phần cứng.

Trong ngữ cảnh các hệ thống phân tán dựa vào tầng gian xây dựng trên các hệ điều hành đa năng, các giải pháp dựa trên phần mềm có nhiều điểm thú vị hơn, trong trường hợp này hai tiêu chí riêng biệt thường được duy trì để phân loại các giao thức lưu

trữ bộ nhớ cache. Thứ nhất, các giải pháp lưu giữ bộ nhớ cache có thể khác nhau trong chiến lược phát hiện tính gắn kết của chúng, nghĩa là phát hiện xác định khi nào thực sự phát hiện ra tính không nhất quán, có hai cách thực hiện như sau:

- Phát hiện tĩnh: Chương trình biên dịch thực hiện các phân tích cần thiết trước khi thực hiện và quyết định những mục dữ liệu nào thực sự không nhất quán, như vậy phần mềm biên dịch chỉ cần thêm vào một số chỉ thị lệnh đảm bảo tính nhất quán.
- Phát hiện động: Thường áp dụng trong các hệ thống phân tán, sự không nhất quán sẽ bị phát hiện trong thời gian chạy, ví dụ kiểm tra dữ liệu trên máy chủ để biết mục dữ liệu có bị thay đổi kể từ thời điểm thực hiện lưu giữ cache.

Trong cơ sở dữ liệu phân tán, các giao thức phát dựa trên phát hiện động có thể tiếp tục phân loại bằng cách xem xét chính xác khi nào phát hiện ra sự không nhất quán trong thời gian thực hiện giao tác, chúng gồm ba loại sau:

- Khi đang thực hiện giao tác, máy khách truy nhập vào mục dữ liệu bộ nhớ cache và cần phải kiểm tra tính nhất quán với phiên bản đang lưu trên máy chủ, giao tác sẽ không thể tiếp tục sử dụng phiên bản đang lưu giữ cho đến khi hoàn thành việc kiểm tra tính nhất quán.
- Giao tác tiếp tục thực hiện trong khi đang thực hiện kiểm tra, đây được coi là cách tiếp cận lạc quan, nó giả thiết dữ liệu trong cache đã được cập nhật mới nhất trước khi giao tác bắt đầu, nếu sau đó chứng minh được giả thiết trên là sai thì sẽ hủy bỏ giao tác.
- Thực hiện kiểm tra dữ liệu đã lưu trong cache đã được cập nhật mới nhất trước khi thực hiện cam kết giao tác, đây cũng được coi là cách tiếp cận lạc quan, giao tác bắt đầu bằng cách thao tác trên dữ liệu hiện hành của bộ nhớ cache với niềm tin đó là dữ liệu mới nhất, sau khi hoàn thành các bước chuẩn bị thì kiểm tra tính nhất quán của các mục dữ liệu đã sử dụng, nếu đó là dữ liệu cũ thì hủy bỏ giao tác.

Một vấn đề khác của giao thức gắn kết bộ nhớ cache là chiến lược ép buộc gắn kết, nó quyết định phương pháp duy trì tính nhất quán giữa bộ nhớ cache với dữ liệu lưu trên các máy chủ. Cách đơn giản nhất là cấm tất cả dữ liệu chia sẻ lưu giữ trên bộ nhớ cache, dữ liệu chia sẻ chỉ được lưu tại các máy chủ, tính nhất quán sẽ được đảm bảo bằng các giao thức dựa trên bản chính hoặc các giao thức nhân bản cập nhật, máy khách chỉ được phép lưu bộ nhớ cache cho những dữ liệu riêng, rõ ràng giải pháp này chỉ cải thiện một phần hiệu năng. Nếu dữ liệu chia sẻ được lưu trong bộ nhớ cache, có hai cách tiếp cận để ép buộc tính gắn kết, cách thứ nhất máy chủ gửi thông điệp hết hiệu lực đến tất cả các cache khi cập nhật mục dữ liệu, cách thứ hai đơn giản chỉ cần lan truyền cập nhật. Nhiều hệ thống cache sử dụng cả hai cách này, lựa chọn động giữa việc gửi thông báo hết hiệu lực hay lan truyền cập nhật đôi khi được sử dụng trong các cơ sở dữ liệu khách/chủ.

Cuối cùng, cần phải xem xét điều gì sẽ xảy ra khi tiến trình thay đổi dữ liệu cache, nếu sử dụng cache chỉ đọc thì các thao tác cập nhật chỉ cần thực hiện trên máy chủ sau đó sẽ sử dụng các giao thức phân phát để đảm bảo các cập nhật lan truyền đến cache. Trong nhiều trường hợp cách tiếp cận kéo cập nhật sẽ được sử dụng, máy khách phát hiện cache đã lỗi thời và gửi yêu cầu đến máy chủ để nhận được dữ liệu mới nhất. Một giải pháp

khác thường được sử dụng trong các hệ thống tập tin phân tán, cho phép các máy khách trực tiếp thay đổi dữ liệu đã lưu giữ trong cache sau đó chuyển cập nhật đến máy chủ. Giải pháp này tương tự như giao thức cập nhật cục bộ dựa trên bản chính trong đó bộ nhớ cache của máy khách trở thành bản chính tạm thời. Để đảm bảo tính nhất quán, cần thiết phải cấp quyền cập nhật riêng cho máy khách, nếu không có thể xảy ra xung đột ghi-ghi. Thay đổi trực tiếp trên cache cải thiện đáng kể hiệu năng vì các thao tác được thực hiện cục bộ, có thể cải tiến hơn nữa bằng cách gộp một số thao tác cập nhật sau đó mới gửi đến máy chủ.

#### **8.5.5 Cài đặt nhất quán lấy máy khách làm trung tâm**

Nếu bỏ qua yếu tố hiệu năng thì việc cài đặt nhất quán lấy máy khách làm trung tương đối dễ hiểu, phần này sẽ giới thiệu một số cách cài đặt có tính chất thực tiễn. Trong cài đặt đơn thuần của nhất quán lấy máy khách làm trung tâm, mỗi thao tác cập nhật W được gán một định danh toàn cục duy nhất do máy chủ đưa ra yêu cầu cập nhật qui định, trong nhất quán liên tục máy chủ này gọi là nguồn Origin(W). Máy chủ phải duy trì hai tập dữ liệu lưu vết thao tác đọc và thao tác ghi cho mỗi máy khách.

Đối với mô hình nhất quán đọc đều, khi máy khách thực hiện thao tác đọc trên máy chủ, máy chủ sẽ chuyển đến tập thao tác đọc để kiểm tra xem tất cả các định danh trên tập đọc đã được cập nhật cục bộ hay chưa, kích thước tập dữ liệu này có thể ảnh hưởng tới vấn đề hiệu năng. Nếu chưa được thực hiện, máy chủ có thể liên hệ với các máy chủ khác để cập nhật dữ liệu mới nhất trước khi thực hiện thao tác đọc hoặc chuyển tiếp yêu cầu cho máy chủ khác đã hoàn thành các thao tác cập nhật. Sau khi hoàn thành thao tác đọc, máy chủ sẽ thêm vào tập dữ liệu lưu vết thao tác đọc một bản ghi tương ứng với thao tác đọc mà máy khách đã yêu cầu.

Như vậy, cần phải quyết định chính xác các thao tác ghi đã xác định trong tập thao tác đọc đã được thực hiện ở đâu, ví dụ định danh thao tác ghi có thể gồm định danh của máy chủ đã đệ trình thao tác cập nhật. Máy chủ đó cần phải ghi nhật ký các thao tác cập nhật sao cho chó thể thực hiện lại trên máy chủ khác, hơn nữa các thao tác cập nhật cần được thực hiện theo thứ tự đã đệ trình. Việc sắp xếp thứ tự có thể thực hiện bằng cách để máy khách tạo ra số tuần tự duy nhất trong hệ thống và gắn vào định danh của thao tác cập nhật, nếu mỗi mục dữ liệu chỉ có thể thay đổi bởi chủ của nó thì sau này có thể cung cấp số tuần tự.

Mô hình nhất quán ghi đều cũng được cài đặt tương tự như mô hình nhất quán đọc đều, khi máy khách khởi sướng một thao tác cập nhật trên máy chủ thì máy chủ sẽ chuyển đến tập dữ liệu thao tác cập nhật để kiểm tra xem thao tác cập nhật đó đã từng được thực hiện hay chưa, kích thước của tập này có thể rất lớn ảnh hưởng đến các yêu cầu về hiệu năng, nếu đó là yêu cầu mới thì máy chủ sẽ thực hiện thao tác cập nhật thao đúng thứ tự. Sau khi hoàn thành thao tác cập nhật mới, định danh của thao tác sẽ được thêm vào tập thao tác ghi, việc duy trì tập dữ liệu các thao tác cập nhật sẽ làm tăng đáng kể thời gian trả lời cho máy khách vì máy khách sẽ chờ cho đến khi hoàn toàn thực hiện xong thao tác.

Nhất quán đọc kết quả ghi đòi hỏi máy chủ thực hiện thao tác đọc phải nhìn thấy tất cả các thao tác ghi trong tập thao tác ghi cho máy khách, các thao tác ghi có thể đơn giản lấy từ các máy chủ khác trước khi thực hiện thao tác đọc, mặc dù điều này có thể dẫn tới thời gian đáp ứng kém. Cách thực hiện khác, phần mềm trên máy khách có thể tìm kiếm máy chủ mà ở đó đã hoàn thành các thao tác trong tập thao tác cập nhật của máy khách. Nhất quán ghi theo sau đọc có thể được cài đặt bằng cách trước hết phải đảm bảo tất cả máy chủ đã hoàn thành các thao tác cập nhật có trong danh sách tập thao tác đọc, sau đó mới thêm định danh của thao tác vào tập dữ liệu các thao tác cập nhật.

Dễ dàng nhận thấy tập thao tác đọc và ghi cho mỗi máy khách có thể sẽ rất lớn, để duy trì việc quản lý những tập này, các thao tác đọc và ghi của máy khách sẽ nhóm lại thành các phiên. Mỗi phiên thường kết hợp với một ứng dụng, mỗi phiên sẽ được mở khi ứng dụng bắt đầu và đóng lại khi thoát khỏi ứng dụng, mỗi phiên cũng có thể kết hợp với tạm thời kết thúc. Khi máy khách đóng phiên thì tập dữ liệu này cũng sẽ bị xóa, tất nhiên nếu máy khách không bao giờ đóng phiên thì tập dữ liệu này có thể vẫn sẽ rất lớn.

Vấn đề chính của cài đặt đơn thuần nằm ở chỗ thể hiện tập các thao tác đọc và ghi, mỗi tập gồm một số định danh cho các thao tác ghi, bất cứ khi nào máy khách chuyển yêu cầu đọc hoặc ghi đến máy chủ, tập các cũng được chuyển cho máy chủ để biết các thao tác cập nhật tương ứng với yêu cầu đọc đã được máy chủ đó thực hiện hay chưa, thông tin này có thể được thể hiện hiệu quả hơn bằng cách sử dụng nhãn thời gian vector.

Khi máy chủ tiếp nhận một thao tác cập nhật mới W, nó gán cho thao tác đó một định danh và kèm theo nhãn thời gian ts(W), các thao tác cập nhật sau đệ trình lên máy chủ đó sẽ có giá trị cao hơn. Mỗi máy chủ  $S_i$  duy trì nhãn thời gian  $WVC_i$  trong đó  $WVC_i[j]$  bằng với nhãn thời gian của thao tác cập nhật gần nhất xuất phát từ máy chủ  $S_j$  và đã được máy chủ  $S_j$  xử lý, giả thiết các thao tác cập nhật từ máy chủ  $S_j$  đều được xử lý theo thứ tự đã đệ trình. Khi máy khách gửi yêu cầu thực hiện thao tác đọc hoặc ghi tại một máy chủ, máy chủ sẽ trả về kết quả kèm theo nhãn thời gian của nó. Ví dụ cho mỗi phiên A sẽ xây dựng nhãn thời gian vector  $SVC_A$  với  $SVC_A[i]$  được đặt giá trị bằng nhãn thời gian lớn nhất của tất cả các thao tác cập nhật trong A xuất phát từ máy chủ  $S_j$ :

$$SVC_A[j] = \max\{ts(W)|W \sqsubseteq A \text{ & } origin(W) = S_j\}$$

Nói cách khác, nhãn thời gian của phiên luôn luôn thể hiện các thao tác cập nhật mới nhất, chúng được các ứng dụng đang thực hiện nhìn thấy như một phần của phiên. Tính cô đọng đạt được bằng cách thể hiện tất cả các thao tác cập nhật quan sát được xuất phát từ cùng máy chủ thông qua một nhãn thời gian.

Ví dụ máy khách là một phần của phiên A truy nhập vào máy chủ  $S_i$  và chuyển  $SVC_A$  cho  $S_i$ , giả thiết  $SVC_A[j] > SVC_i[j]$ , nghĩa là  $S_i$  chưa nhìn thấy tất cả các thao tác cập nhật xuất phát từ  $S_j$  mà máy khách đã nhìn thấy. Tùy thuộc vào tính nhất quán được yêu cầu, máy chủ  $S_i$  có thể lấy các thao tác cập nhật này trước khi thông báo nhất quán cho máy khách. Khi hoàn thành thao tác, máy chủ  $S_i$  sẽ trả về nhãn thời gian của nó  $WVC_i$ , ở thời điểm này  $SVC_A$  được điều chỉnh  $SVC_A[j] = \max\{SVC_A[j], WVC_i[j]\}$ . Một lần nữa chúng ta lại thấy nhãn thời gian vector cung cấp cách đơn giản và nhỏ gọn thể hiện lịch sử sự kiện trong các tiến trình của hệ thống phân tán.

## THẢO LUẬN

1. Trình bày vai trò của nhân bản dữ liệu và ảnh hưởng của nó đối với việc đảm bảo tính nhất quán của dữ liệu trong hệ phân tán?
2. So sánh cách tiếp cận nhân bản lấy dữ liệu làm trung tâm với nhân bản lấy máy khách làm trung tâm.
3. Cài đặt SnapShot trong hệ quản trị cơ sở dữ liệu SQL Server.
4. Cam kết phân tán được sử dụng trong mô hình nhân bản nào?
5. Dữ liệu của hệ thống phân giải tên miền DNS được nhân bản theo mô hình nào?
6. Giao thức nhân bản nào phù hợp với hệ thống thanh khoản trực tuyến?
7. Hệ thống bán hàng trong các siêu thị nên sử dụng hình thức nhân bản nào?
8. Nêu một số ví dụ về nhân bản lấy máy khách làm trung tâm.
9. Nêu một số ví dụ về nhân bản yếu.
10. Nêu những giải pháp nhân bản để dữ liệu gần với người sử dụng nhất.

## CHƯƠNG 9: BẢO MẬT

Bảo mật không phải là yếu tố quan trọng nhất nhưng có thể thấy đó là một trong những vấn đề phức tạp nhất vì nó bao trùm lên toàn bộ hệ thống, một lỗ hổng trong thiết kế bảo mật có thể làm cho tất cả các biện pháp bảo mật trở nên vô dụng. Xây dựng tất cả các loại cơ chế bảo mật áp dụng cho hệ thống sẽ không có ý nghĩa gì nếu không biết cách sử dụng để chống lại đối tượng nào, điều này đòi hỏi phải biết về chính sách bảo mật phải được thực hiện. Chương này sẽ giới thiệu những vấn đề cơ bản về bảo mật, các chính sách bảo mật, vấn đề thiết kế cơ chế trợ giúp thực thi chính sách bảo mật, mã hóa dữ liệu và cuối cùng sẽ giới thiệu giải pháp tích hợp vào hệ thống phân tán để hỗ trợ bảo mật.

Bảo mật trong hệ thống phân tán có thể chia thành hai nhóm, nhóm thứ nhất liên quan đến truyền thông giữa người sử dụng với người sử dụng hoặc giữa các tiến trình với nhau, nhóm thứ hai liên quan đến vấn đề lưu trữ trên các máy tính khác nhau. Nhóm thứ nhất liên quan tới các cơ chế căn bản để đảm bảo bí mật truyền thông là bí mật kênh truyền, trong khi nhóm thứ hai liên quan tới các vấn đề về ủy quyền và kiểm soát truy nhập. Bí mật kênh truyền và kiểm soát truy nhập đòi hỏi các cơ chế phân phát các khóa mã hóa và đồng thời cũng cần phải có những cơ chế về việc quản lý người sử dụng, những vấn đề này sẽ được đề cập tới trong mục quản lý bảo mật. Ngoài ra, cũng sẽ thảo luận các vấn đề giải quyết liên quan tới quản lý các khóa mã hóa, quản lý nhóm bảo mật và xử lý chứng chỉ bảo mật.

### 9.1 Giới thiệu chung

Bảo mật trong một hệ thống máy tính liên quan chặt chẽ đến khái niệm về tính đáng tin cậy, có thể hiểu một hệ thống máy tính đáng tin cậy là hệ thống mà người sử dụng hoàn toàn tin tưởng vào các dịch vụ mà hệ thống đó cung cấp. Tính chịu lỗi của một hệ thống bao gồm khả năng sẵn sàng, mức độ đáng tin cậy, mức độ an toàn và khả năng bảo trì. Khả năng sẵn sàng liên quan tới công tác đảm bảo hệ thống hoạt động liên tục, đáp ứng ngay những yêu cầu sử dụng của người sử dụng, điều này cũng liên quan tới vấn đề dự phòng và khắc phục khi xảy ra sự cố.

Tuy nhiên, nếu chúng ta tin tưởng vào hệ thống máy tính thì cũng nên tính đến tính bí mật và toàn vẹn thông tin, tính bí mật ngụ ý thông tin chỉ được phép tiết lộ cho những đối tượng có thẩm quyền. Tính bí mật thể hiện sự bảo vệ dữ liệu của hệ thống thông tin, chỉ những người dùng hợp pháp mới được phép truy nhập thông tin và thông tin đó được trao đổi theo các cơ chế đảm bảo an toàn, tránh truy nhập của người dùng bất hợp pháp. Tính toàn vẹn là đặc tính chỉ có thể thay đổi tài nguyên hệ thống theo cách đã được ủy quyền, đảm bảo dữ liệu không bị sửa đổi hoặc phá hủy bởi người dùng bất hợp pháp. Tài nguyên chính của hệ thống máy tính bao gồm phần cứng, phần mềm và dữ liệu, hệ thống phải có khả năng phát hiện và phục hồi cho những tài nguyên này khi những thay đổi không hợp lệ.

#### 9.1.1 Các hình thức xâm phạm hệ thống thông tin

Một cách nhìn khác về bảo mật trong các hệ thống máy tính là chúng ta cố gắng bảo vệ các dịch vụ và dữ liệu chống lại các mối đe dọa về bảo mật. Hiểm họa đối với

thông tin bao gồm những vấn đề liên quan tới điều kiện vật lý và các cuộc tấn công vào hệ thống. Hiểm họa vật lý có thể kể tới vấn đề nguồn điện, môi trường..., các cuộc tấn công từ bên ngoài có thể xuất hiện dưới dạng tấn công thăm dò, truy nhập trái phép, tấn công từ chối dịch vụ, tấn công bằng phần mềm chứa mã độc... Pfleeger đã chỉ ra bốn loại hiểm họa bảo mật gồm chặn thông điệp, làm gián đoạn hệ thống, sửa đổi nội dung và bịa đặt nội dung.

Khái niệm chặn thông điệp đề cập đến tính huống đối tượng không có thẩm quyền đã giành được quyền truy nhập vào dịch vụ hoặc dữ liệu, ví dụ thông tin đã bị nghe trộm trên đường truyền. Hiểm họa gián đoạn cũng xảy ra khi dữ liệu bị sao chép trái phép, ví dụ sau khi đột nhập vào thư mục cá nhân sẽ làm hỏng hoặc xóa các tập tin trong hệ thống, như vậy hệ thống sẽ không còn có khả năng cung cấp dịch vụ liên quan tới các tập tin đó. Tổng quát hơn, hiểm họa gián đoạn đề cập tới các tình huống dịch vụ hoặc dữ liệu không thể sẵn sàng, không thể sử dụng được hoặc bị phá hủy hoàn toàn, các cuộc tấn công từ chối dịch vụ có gắng làm cho người sử dụng không thể tiếp cận được với các dịch vụ cũng được xếp vào loại này.

Khái niệm sửa đổi bao gồm thay đổi trái phép dữ liệu hoặc giả mạo dịch vụ để nó không còn giữ được những đặc điểm kỹ thuật ban đầu, ví dụ chặn và sau đó thay đổi dữ liệu trên đường truyền, giả mạo các thực thể cơ sở dữ liệu và thay đổi chương trình để bí mật ghi lại các hoạt động của người sử dụng. Khái niệm bịa đặt đề cập đến tình huống tạo thêm dữ liệu hoặc hoạt động bình thường không tồn tại, ví dụ kẻ đột nhập có thể có gắng thêm thông tin vào tập tin mật khẩu hoặc cơ sở dữ liệu, đột nhập vào hệ thống và gửi lại các thông điệp do hệ thống đã gửi trước đó. Lưu ý rằng các hình thức gián đoạn, sửa đổi và bịa đặt đều có thể được xem như những hình thức giả mạo dữ liệu.

Chỉ đơn giản nói rằng hệ thống nên có khả năng tự chống lại tất cả các hiểm họa bảo mật thì không phải là cách thực sự xây dựng một hệ thống an toàn, trước hết cần phải mô tả các yêu cầu bảo mật hay còn gọi là các chính sách bảo mật. Chính sách bảo mật mô tả chính xác các hành động nào các thực thể trong hệ thống được phép thực hiện và những hành động nào bị cấm. Các thực thể bao gồm người sử dụng, các định vụ, dữ liệu, máy móc... Khi đã thiết lập chính sách bảo mật thì có thể tập trung vào những cơ chế bảo mật để đảm bảo thực thi chính sách. Các cơ chế bảo mật quan trọng bao gồm mã hóa dữ liệu, xác thực, ủy quyền và lưu vết.

Mã hóa là nền tảng của bảo mật trong các hệ thống thông tin, nó chuyển dữ liệu thành dạng để kẻ tấn công không thể hiểu được, nghĩa là cung cấp các phương tiện để thực hiện bí mật dữ liệu. Hơn nữa, mã hóa còn cho phép chúng ta kiểm tra xem dữ liệu có bị sửa đổi hay không, như vậy nó cũng có thể hỗ trợ kiểm tra tính toàn vẹn. Xác thực dùng để kiểm tra tính chính danh của đối tượng tham gia, đối tượng có thể là người sử dụng hoặc máy tính và các thực thể khác, như vậy việc xác thực sẽ loại bỏ được những đối tượng giả mạo. Thông thường, người sử dụng được xác thực bằng mật khẩu, các thiết bị có thể được xác thực bằng địa chỉ logic hoặc địa chỉ vật lý và cũng có thể được xác thực bằng mật khẩu.

Sau khi đã xác thực, cần thiết phải kiểm tra xem đối tượng có được ủy quyền thực hiện các hành động đã yêu cầu hay không, thực chất đó là những qui định về phân quyền thực hiện. Các công cụ lưu vết ghi lại những đối tượng nào đã truy nhập và truy nhập bằng cách nào, mặc dù nó không bảo vệ chống lại các hiểm họa bảo mật nhưng nhật ký lưu vết có thể đặc biệt hữu ích cho việc phân tích lỗ hổng bảo mật và tìm ra các biện pháp chống lại kẻ tấn công. Vì lý do này kẻ tấn công moi chung rất ranh ma không để lại dấu vết nào để có thể dẫn đến lộ danh tính của mình, trong trường hợp này ghi nhật ký truy nhập đôi khi sẽ làm cho cuộc tấn công mạo hiểm hơn. Các điểm yếu của hệ thống phân tán nằm ở kỹ thuật truyền thông và chính sách bảo mật thông tin. Điểm yếu về kỹ thuật truyền tin trên mạng có nguyên nhân từ đặc tính mở của các giao thức như HTTP, FTP, ICMP, SMNP, SMTP.... Điểm yếu về chính sách bảo mật hệ thống thông tin thể hiện ở việc không xây dựng chính sách khắc phục khi tấn công xảy ra và cũng không có qui trình giám sát và kiểm tra các hoạt động trong hệ thống thông tin.

#### 9.1.1.1 Tấn công thăm dò

Đối tượng truy nhập trái phép sử dụng hình thức tấn công này để khám phá hệ thống thông tin, ví dụ thông tin về hình trạng mạng, hệ điều hành, các dịch vụ sử dụng và các điểm yếu có thể lợi dụng để tấn công nhằm mục đích cuối cùng là lấy cắp dữ liệu, bắt đầu cho một cuộc tấn công nguy hiểm hoặc tấn công từ chối dịch vụ. Tấn công thăm dò thường được thực hiện theo các bước sau:

1. Ping đến địa chỉ đích xem địa chỉ IP đó có hoạt động không.
2. Quét cổng để xác định cổng hay dịch vụ nào đang hoạt động trên địa chỉ IP đó.
3. Truy vấn đến các cổng, từ đó có thể xác định kiểu và phiên bản của ứng dụng, hệ điều hành.
4. Từ các bước trên suy ra các điểm yếu có thể lợi dụng để tấn công.

Hệ điều hành trên các máy tính hiện nay đều cung cấp dịch vụ tường lửa, để hạn chế hình thức tấn công này, nên cấm dịch vụ Ping và chỉ mở những cổng cần thiết cho dịch vụ của hệ thống.

#### 9.1.1.2 Truy nhập trái phép

Để truy nhập được vào hệ thống thông tin phải có tài khoản đang nhập, việc lộ bí mật tài khoản dưới bất cứ hình thức nào cũng có thể gây nên tổn hại làm mất thông tin. Đối tượng truy nhập trái phép trước hết phải đánh cắp thông tin tài khoản, công việc này có thể được thực hiện bằng các hình thức sau:

1. Thăm dò mật khẩu: nhằm mục đích lấy cắp thông tin đăng nhập của người dùng bao gồm tên đăng nhập và mật khẩu, có thể sử dụng phương pháp dự đoán mật khẩu theo ký tự từ điển, đoán mật khẩu
2. Đóng vai trung gian: Sử dụng các công cụ và phần mềm bắt gói tin, phân tích sửa đổi và là người trung gian chuyển dữ liệu cho máy trạm và máy chủ.
3. Sử dụng tính năng tin cậy có sẵn trong một số hệ thống truyền tin: Một số thiết bị và máy tính có cài đặt sẵn tính năng này nhằm nâng cao hiệu suất hoạt động.

Để chống lại thăm dò mật khẩu, mật khẩu phải thuộc loại mạnh, nghĩa là phải dài tối thiểu 8 ký tự bao gồm các chữ in hoa và in thường, chữ số và ký tự đặc biệt, không

nên chứa chuỗi ký tự liên tiếp của tên truy nhập. Trong cơ sở dữ liệu, chỉ lưu trữ mật khẩu đã mã hóa một chiều, sẽ tốt hơn nếu mã hóa cả tên truy nhập, tên truy nhập không nên chứa dấu cách, cách làm này có thể ngăn chặn tấn công chèn đoạn mã. Nếu kiểm tra thông tin đăng nhập không đúng, không nên phản hồi nguyên nhân sai tên truy nhập hay mật khẩu, nên hạn chế số lần đăng nhập sai. Chức năng đăng nhập nên bổ sung phần nhập mã kiểm tra, yêu cầu người sử dụng phải nhập xâu ký tự ngẫu nhiên, qui định khoảng thời gian tối thiểu giữa lần hiển thị mã kiểm tra và lần người sử dụng gửi thông tin tài khoản để đăng nhập. Để ngăn chặn hiện tượng bắt gói tin trên kênh truyền, mật khẩu cũng như một số dữ liệu khác phải được mã hóa trước khi máy khách gửi đến máy chủ.

#### 9.1.1.3 Tấn công từ chối dịch vụ

Đối tượng tấn công vô hiệu hóa hoặc phá hỏng hệ thống thông tin, làm chậm hệ thống và các dịch vụ ứng dụng với mục đích từ chối yêu cầu sử dụng dịch vụ cho những người sử dụng hợp pháp. Hình thức tấn công này dựa trên nguyên lý hoạt động của giao thức TCP, phải thiết lập liên kết trước khi truyền số liệu, như vậy máy chủ sẽ luôn phải tiếp nhận các yêu cầu liên kết từ địa chỉ của kẻ tấn công. Tấn công từ chối dịch vụ có thể xuất phát từ một máy tính, phát hiện và xử lý trường hợp này khá đơn giản, chỉ cần cấm địa chỉ IP. Vấn đề sẽ phức tạp hơn nếu đó là tấn công từ chối dịch vụ dạng phân tán, trường hợp này cần có sự giúp đỡ của các chuyên gia bảo mật.

#### 9.1.1.4 Phần mềm độc hại

Phần mềm độc hại (Worms, Virus, Trojan Horses) là các chương trình chứa mã độc được chèn vào máy tính để phá hủy hệ thống, tự nhân bản và phát tán, từ chối dịch vụ hoặc truy nhập tới mạng, hệ thống và các dịch vụ. Virus xâm nhập vào máy tính theo khi sao chép tập tin từ ổ đĩa trên máy tính, thông qua thư điện tử.... Các dấu hiệu khi hệ thống bị nhiễm virus bao gồm:

- Chương trình khởi động chậm hơn, điều này là do virus đang phát tán tới các tập tin khác trên hệ thống hoặc đang chiếm tài nguyên.
- Xuất hiện các tập tin lạ trên ổ cứng hoặc mất tập tin, một số virus xóa các tập tin quan trọng của hệ thống.
- Kích thước chương trình thay đổi so với phiên bản cài đặt ban đầu, điều này xảy ra vì virus gắn chính nó vào chương trình.
- Trình duyệt, chương trình xử lý văn bản, màn hình, thực đơn và cách ứng dụng khác bị thay đổi.
- Hệ thống bị tắt hoặc khởi động một cách bất thường.
- Mất quyền truy nhập đến ổ cứng hoặc các tài nguyên hệ thống khác, virus có thể thay đổi các thiết lập trên thiết bị làm cho nó không hoạt động chính xác.
- Hệ thống không khởi động hoặc đưa ra các thông báo lỗi trong suốt quá trình khởi động.

Trojan horse không tự nhân bản, khi xâm nhập vào máy tính chúng thường tạo ra một lỗ hổng bảo mật gọi là cửa sau, kẻ đột nhập có thể thông qua cửa sau để xâm nhập và nắm quyền kiểm soát máy tính. Worm là một dạng mã độc có khả năng tự nhân bản, lây

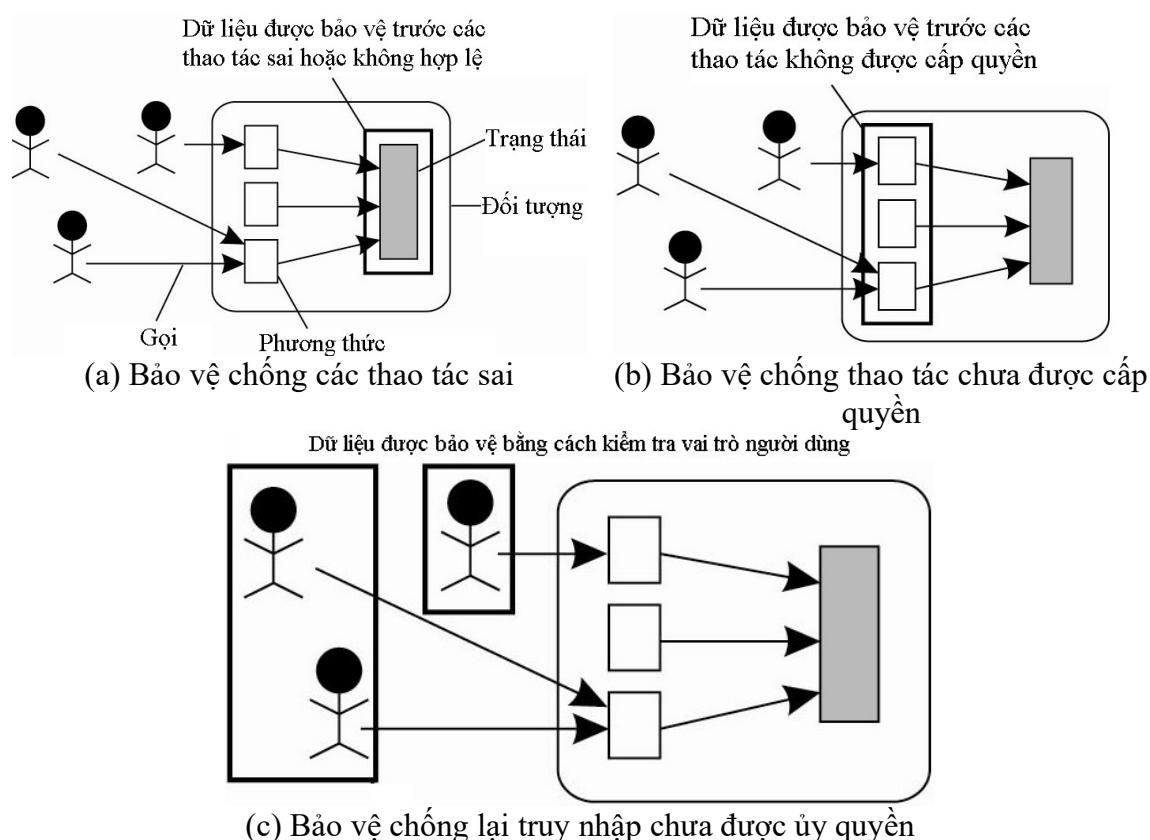
lan và không cần vật chủ, chúng có kích thước lớn hơn virus và thường nấp dưới dạng một tập tin đính kèm thư điện tử. Worms không phải là virus nhưng hay được sử dụng như một công cụ để chứa và phát tán virus, khi máy tính bị nhiễm Worms nó sẽ nhanh chóng chiếm bộ nhớ và tự nhân bản.

### 9.1.2 Các vấn đề thiết kế

Hệ thống phân tán phải cung cấp các dịch vụ bảo mật để có thể cài đặt các chính sách bảo mật, nhiều vấn đề thiết kế quan trọng cần phải lưu ý khi cài đặt các dịch vụ bảo mật đa năng. Phần này sẽ giới thiệu những vấn đề cơ bản trong thiết kế bảo mật, đó là xác định trọng tâm kiểm soát, phân tầng cơ chế bảo mật, phân bố cơ chế bảo mật và tính đơn giản của cơ chế bảo mật.

#### 9.1.2.1 Xác định trọng tâm bảo mật

Khi xem xét bảo vệ hệ thống phân tán, hình 9.1 thể hiện ba cách tiếp cận cơ bản: bảo vệ dữ liệu, phân quyền thao tác và phân quyền cho người sử dụng. Cách tiếp cận thứ nhất tập trung vào việc bảo vệ trực tiếp dữ liệu, không quan tâm đến thao tác thực hiện mà mục đích chính là đảm bảo tính toàn vẹn dữ liệu. Ví dụ, các ràng buộc trong các hệ quản trị cơ sở dữ liệu tự động kiểm tra sự thay đổi của mục dữ liệu, có thể đặt thuộc tính hoặc sử dụng đặc tính tự nhiên của giao tác để ngăn ngừa thay đổi cho những mục dữ liệu chỉ được phép đọc.



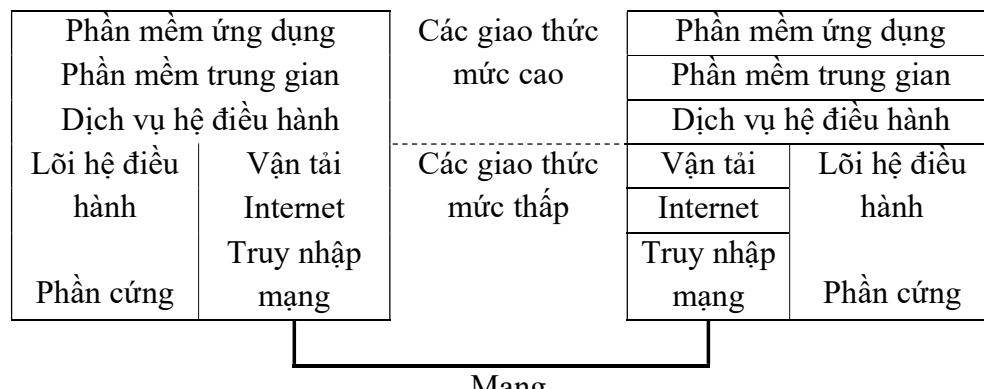
Hình 9.1 Các cách tiếp cận bảo mật

Cách tiếp cận thứ hai tập trung vào các thao tác, xác định đối tượng được phép gọi thao tác và thậm chí xác định thời gian được phép truy nhập tài nguyên, trọng tâm kiểm soát liên quan mạnh mẽ với các cơ chế kiểm soát truy nhập. Cách tiếp cận thứ ba tập trung trực tiếp vào người sử dụng bằng cách tạo ra các phương tiện và phân quyền cho họ để có thể truy nhập vào hệ thống. Như vậy vấn đề kiểm soát sẽ tập trung vào việc định nghĩa các vai trò của người sử dụng, hệ thống cần phải được thiết kế sao cho có thể định nghĩa các quyền, cung cấp cơ chế hỗ trợ kiểm soát truy nhập dựa trên việc phân quyền.

#### 9.1.2.2 Phân tầng cơ chế bảo mật

Một điểm quan trọng trong thiết kế các hệ thống an toàn là quyết định các cơ chế bảo mật nên đặt ở tầng nào, khái niệm tầng ở đây được mạnh dạn sử dụng vì tất cả các hệ thống phân tán đều được xây dựng trên nền tảng mạng máy tính và các mạng máy tính đều được tổ chức phân tầng theo mô hình tham chiếu.

Hình 9.2 thể hiện mô hình phân tầng của hệ thống phân tán, thông thường gồm tầng ứng dụng, tầng phần mềm trung gian, tầng dịch vụ hệ điều hành và tầng lõi của hệ điều hành. Việc tách biệt các dịch vụ đa dụng với các dịch vụ truyền thông để hiểu về cơ chế phân tầng bảo mật trong các hệ thống phân tán. Bảo mật là câu chuyện của kỹ thuật còn tin tưởng lại là cảm xúc, cơ chế bảo mật sẽ được đặt ở tầng nào phụ thuộc vào việc các dịch vụ sẽ được bảo mật như thế nào trong các tầng cụ thể để máy khách có thể tin tưởng.

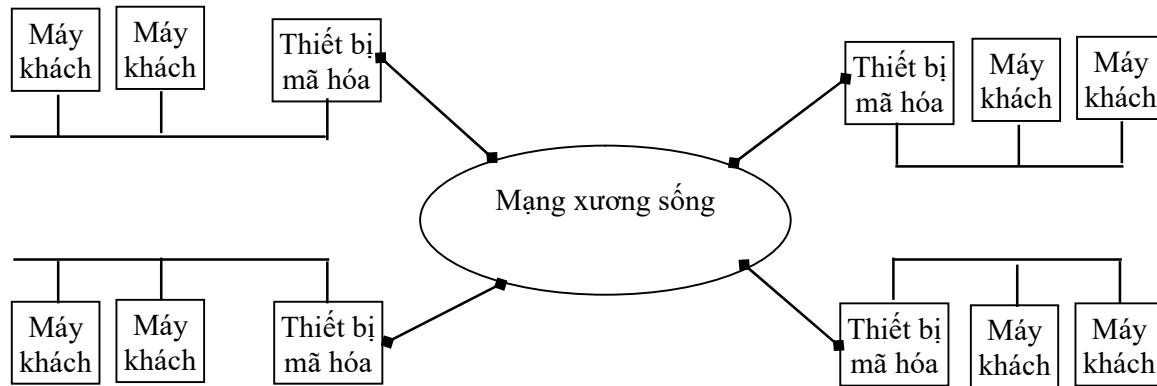


Hình 9.2 Tổ chức phân tầng logic hệ thống phân tán

Ví dụ, xem xét tổ chức được đặt tại các địa điểm khác nhau và chúng được nối với nhau qua dịch vụ chuyển mạch tốc độ cao SMDS, mạng SMDS có thể coi như xương sống nối các mạng cục bộ phân tán về mặt địa lý. Có thể thực hiện bảo mật bằng cách đặt thiết bị mã hóa tại mỗi thiết bị định tuyến của SMDS, những thiết bị này tự động mã hóa và giải mã các gói tin được gửi giữa các địa điểm nhưng không bảo vệ truyền thông giữa các máy cùng địa điểm. Người sử dụng thuộc các địa điểm khác nhau gửi thông điệp cho nhau, họ đều lo sợ thông điệp sẽ bị lộ khi gửi trên đường truyền, nhưng ít nhất họ phải tin tưởng rằng thông điệp đều sẽ được mã hóa khi ra khỏi mạng cục bộ, như vậy hệ thống phân tán đã cung cấp các biện pháp chống lại việc giả mạo thiết bị.

Giả sử người sử dụng không tin tưởng vào bảo mật liên mạng, họ có thể sử dụng các dịch vụ bảo mật của tầng vận tải như SSL, các đoạn tin của tầng vận tải vẫn được mã

hóa trên thiết bị mã hóa của mạng SMDS nhưng trong trường hợp này người sử dụng đặt niềm tin vào SSL. Trong các hệ thống phân tán, các cơ chế bảo mật thường đặt tại tầng phần mềm trung gian, nếu người sử dụng không tin tưởng SSL thì vẫn có thể sử dụng dịch vụ gọi thủ tục từ xa an toàn, người sử dụng một lần nữa lại tin tưởng vào dịch vụ gọi thủ tục từ xa an toàn như đã hứa hẹn, như thế sẽ không bị rò rỉ thông tin hoặc xác thực chính xác máy khách và máy chủ.



Hình 9.3 Bảo mật liên mạng

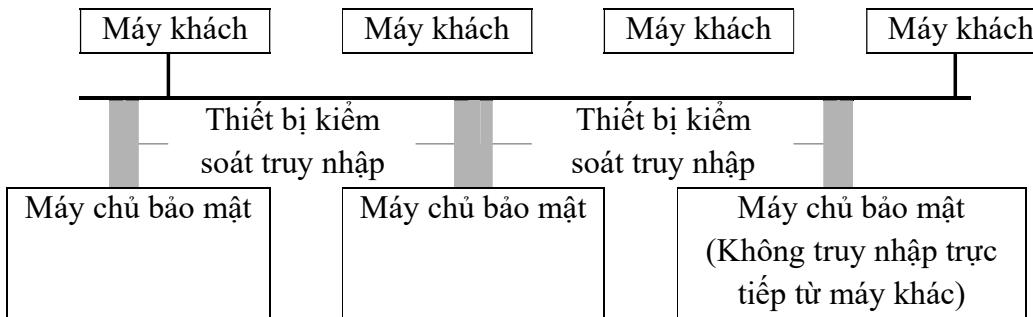
Các dịch vụ bảo mật đặt ở tầng phần mềm trung gian của hệ thống phân tán chỉ có thể đáng tin cậy nếu chúng dựa trên các dịch vụ thực sự an toàn. Ví dụ, dịch vụ gọi thủ tục từ xa an toàn được cài đặt bằng SSL thì sự tin cậy của thủ tục gọi từ xa hoàn toàn phụ thuộc vào mức độ tin cậy SSL, nếu nó không đáng tin cậy thì dịch vụ gọi thủ tục từ xa cũng không đáng tin cậy.

#### 9.1.2.3 Phân bố cơ chế bảo mật

Sự phụ thuộc giữa các dịch vụ đối với sự tin cậy dẫn đến khái niệm cơ sở tính toán tin cậy, đó là tập tất cả các cơ chế bảo mật trong hệ thống phân tán cần thiết để đảm bảo chính sách bảo mật, như vậy nó phải được tin cậy, cơ sở tính toán tin cậy càng nhỏ càng tốt. Nếu hệ thống phân tán được xây dựng như phần mềm trung gian trên hệ điều hành thì tính bảo mật của nó có thể phụ thuộc vào tính bảo mật của các hệ điều hành cục bộ đang sử dụng, nghĩa là cơ sở tính toán tin cậy trong hệ thống phân tán có thể bao gồm cả các hệ điều hành cục bộ trên các máy khác nhau. Ví dụ các máy chủ tập tin trong hệ thống tập tin phân tán có thể phải dựa vào các cơ chế bảo vệ khác nhau do hệ điều hành cục bộ cung cấp, những cơ chế này không chỉ gồm việc bảo vệ các tập tin chống lại sự truy nhập của các tiến trình khác ngoài máy chủ tập tin mà còn các cơ chế bảo vệ máy chủ tập tin tránh được tấn công mạng.

Các hệ thống phân tán dựa trên phần mềm trung gian như vậy đòi hỏi tin tưởng vào các hệ điều hành cục bộ mà chúng phụ thuộc, nếu không tin tưởng thì một phần chức năng của hệ điều hành cục bộ có thể sẽ phải được tích hợp vào hệ thống phân tán. Hầu hết các dịch vụ của hệ điều hành đều chạy như những tiến trình của người sử dụng, trong trường hợp này hệ thống tập tin có thể phải thay thế hoàn toàn để đáp ứng các nhu cầu riêng của hệ thống phân tán, bao gồm cả các biện pháp bảo mật. Cách tiếp cận phù hợp sẽ là tách riêng các dịch vụ bảo mật với các loại dịch vụ khác bằng cách phân tán các dịch

vụ trên các máy khác nhau tùy theo yêu cầu bảo mật. Ví dụ để bảo mật hệ thống tập tin phân tán có thể cách ly máy chủ tập tin với các máy khách bằng cách đặt máy chủ trên máy với hệ điều hành tin cậy, có thể chạy hệ thống tập tin an toàn chuyên dụng, các máy khách đặt trên các máy không tin cậy.



Hình 9.4 Nguyên lý RISSC áp dụng cho hệ thống phân tán

Việc tách riêng này làm giảm đáng kể cơ sở tính toán tin cậy xuống còn một lượng nhỏ các máy và các thành phần phần mềm, kết quả là bảo vệ các máy này tránh được những tấn công bảo mật từ bên ngoài, sự tin cậy tổng thể trong bảo mật hệ thống phân tán có thể sẽ tăng lên. Hình 9.4 thể hiện cách lấp đặt giao diện thu gọn để bảo mật các thành phần của hệ thống RISSC để ngăn chặn các máy khách các ứng dụng trực tiếp truy nhập vào các dịch vụ quan trọng, bất kỳ máy chủ bảo mật quan trọng nào đều được đặt trên máy riêng biệt cách ly với các hệ thống đầu cuối người sử dụng và sử dụng các giao diện mạng an toàn mức thấp, máy khách và các ứng dụng của họ có thể truy nhập đến máy chủ bảo mật thông qua các giao diện mạng này.

#### 9.1.2.4 Tính đơn giản trong thiết kế bảo mật

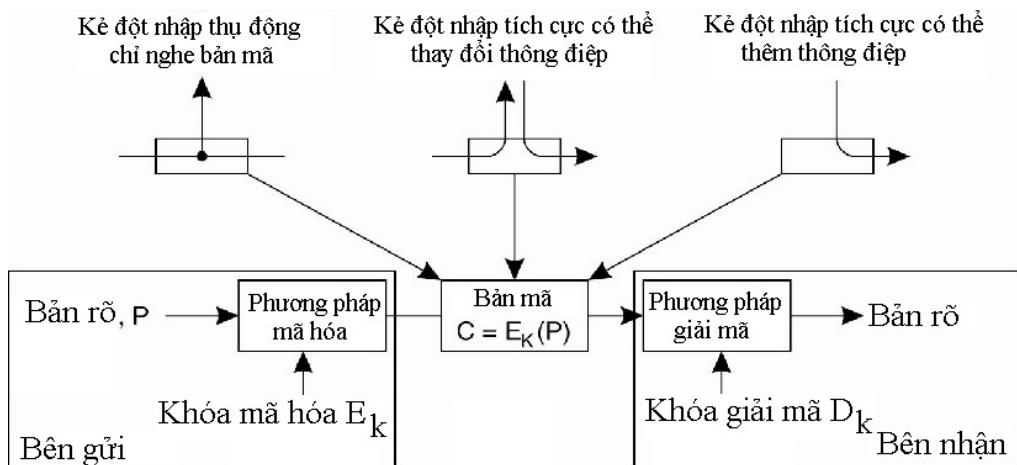
Quyết định tầng nào sẽ đặt các cơ chế bảo mật phải đảm bảo tính đơn giản, nói chung việc thiết kế an toàn cho hệ thống máy tính là nhiệm vụ khó khăn, thiết kế sao cho cơ chế vận hành đơn giản và dễ hiểu nhưng vẫn phải đảm bảo tính tin cậy. Rất tiếc những cơ chế đơn giản không phải lúc nào cũng đủ để thực thi chính sách bảo mật, ví dụ về trường hợp kết nối liên mạng trên hình 9.3, mã hóa tầng truy nhập mạng là cơ chế đơn giản và dễ hiểu để bảo vệ chống lại nguy cơ đánh cắp khi trao đổi thông tin liên mạng, tuy nhiên người sử dụng lại muốn đảm bảo chính xác đối tượng gửi thông điệp cho mình. Trong trường hợp này cần đến các dịch vụ và xác thực mức người sử dụng, họ có thể muốn biết các dịch vụ đó làm việc như thế nào để đặt niềm tin vào những dịch vụ đó. Do đó, xác thực người dùng có thể yêu cầu ít nhất khái niệm về các khóa bí mật và hiểu biết về chứng chỉ, mặc dù nhiều dịch vụ bảo mật được tự động hóa ở mức cao và hoàn toàn trong suốt đối với người sử dụng.

Trong các trường hợp khác, bản thân ứng dụng vốn đã phức tạp và việc giới thiệu bảo mật chỉ làm cho vấn đề tồi tệ hơn. Ví dụ điển hình liên quan đến các giao thức bảo mật phức tạp là các hệ thống thanh toán, sự phức tạp của giao thức thanh toán thường do nhiều bên liên quan tham gia. Trong những trường hợp này, điều quan trọng là các cơ chế nền tảng được sử dụng để cài đặt các giao thức phải tương đối đơn giản và dễ hiểu.

Tính đơn giản sẽ góp phần tạo ra sự tin cậy mà người sử dụng đầu cuối sẽ đưa vào ứng dụng và quan trọng hơn sẽ góp phần thuyết phục các nhà thiết kế rằng hệ thống không có lỗ hổng bảo mật.

### 9.1.3 Mã hóa

Nền tảng cho bảo mật trong các hệ thống phân tán là sử dụng các kỹ thuật mã hóa, đó là các giải pháp khác nhau bảo vệ thông tin khi lưu chuyển trên mạng cũng như truy nhập vào kho dữ liệu. Tư tưởng cơ bản trong việc mã hóa thông tin là che giấu nội dung dữ liệu đối với những đối tượng không có thẩm quyền, vì vậy bên gửi sẽ mã hóa dữ liệu trước khi gửi, bên nhận sẽ giải mã dữ liệu nhận được để phục hồi lại nội dung ban đầu. Tuy nhiên, việc mã hóa thông tin sẽ làm tăng công việc xử lý, điều đó làm ảnh hưởng đến hiệu suất hoạt động của hệ thống. Do đó, việc mã hóa thông tin cần phải thực hiện cho từng loại thông tin nhằm đảm bảo tính sẵn sàng phục vụ của hệ thống phân tán, cần phải biết cân đối hài hòa giữa mục tiêu bảo mật và khả năng đáp ứng yêu cầu dịch vụ của hệ thống.



Hình 9.5 Xâm nhập và nghe trộm trong truyền thông

Mã hóa và giải mã kết hợp với nhau bằng cách sử dụng các phương thức mã hóa khóa bí mật, dạng ban đầu của thông điệp gọi là bản rõ và dạng đã được mã hóa gọi là bản mã. Gọi bản tin rõ là  $P$ , khóa mã hóa là  $K$ , thông điệp được mã hóa theo khóa  $E_K$  là  $C=E_K(P)$ , thông điệp được giải mã theo khóa giải mã  $P=D_K(C)$ . Hình 9.5 minh họa ba dạng tấn công trên đường truyền, kỹ thuật mã hóa sẽ trợ giúp để chống lại những dạng tấn công này. Loại thứ nhất, kẻ đột nhập có thể chặn thông điệp mà bên gửi và bên nhận đều không biết đang bị nghe trộm, do bản tin đã được mã hóa nên khó có thể giải mã nếu không biết chính xác khóa bí mật, vì vậy việc chặn bản tin trở nên vô nghĩa, kẻ đột nhập không thể đọc được nội dung bản tin. Tuy nhiên cũng cần phải lưu ý, đôi khi chỉ cần một thông điệp kẻ tấn công cũng đủ để đưa ra nhận định nào đó, ví dụ lượng thông tin gửi đến một địa điểm đã qui định giảm đáng kể so với thông thường, trong khi lượng tương ứng lại tăng lên đáng kể chuyển đến một địa điểm khác.

Loại tấn công thứ hai là sửa đổi thông điệp, sửa đổi thông điệp bản rõ rất dễ nhưng sửa thông điệp bản mã một cách chính xác sẽ phức tạp hơn rất nhiều, đầu tiên kẻ đột nhập phải giải mã được thông điệp trước khi thực hiện sửa đổi, sau đó lại phải mã hóa

chính xác, nếu không bên nhận có thể được cảnh báo thông điệp đã bị giả mạo. Loại tấn công thứ ba, kẻ đột nhập thêm các bản tin mã hóa vào hệ thống truyền thông để bên nhận B tin rằng đó là thông điệp xuất phát từ bên gửi A, kẻ đột nhập đã thay đổi được nội dung thông điệp thì việc thêm một thông điệp giả mạo là chuyện bình thường, để thực hiện điều này tất nhiên kẻ tấn công phải biết được khóa bí mật trao đổi giữa bên gửi A và bên nhận B. Về cơ bản, dựa trên khóa mã hóa và khóa giải mã, có hai loại hệ thống mã hóa khác nhau, mã hóa đối xứng sử dụng khóa bí mật chung cho cả bên gửi lẫn bên nhận, trong mã hóa bất đối xứng khóa mã hóa bên gửi khác với khóa giải mã bên nhận.

Mã hóa đối xứng dùng khóa bí mật, khóa mã hóa và khóa giải mã giống nhau, bên nhận và bên gửi đều phải có khóa trên và khóa phải được giữ bí mật nên phương pháp này còn gọi là mã hóa bí mật hoặc mã hóa chia sẻ. Sử dụng ký pháp  $K_{A,B}$  ngụ ý khóa K được chia sẻ giữa A và B, ta có:

$$P = D_{K_{A,B}}(E_{K_{A,B}}(P))$$

Mã hóa bất đối xứng dùng khóa công khai, khóa mã hóa và khóa giải mã khác nhau nhưng chúng tạo thành một cặp duy nhất giữa bên gửi và bên nhận, một khóa sẽ được giữ bí mật còn một khóa sẽ được công khai nên phương pháp này còn gọi là mã hóa công khai. Ký hiệu  $K_E$  là khóa mã hóa và  $K_D$  là khóa giải mã, ta có:

$$P = D_{K_D}(E_{K_E}(P))$$

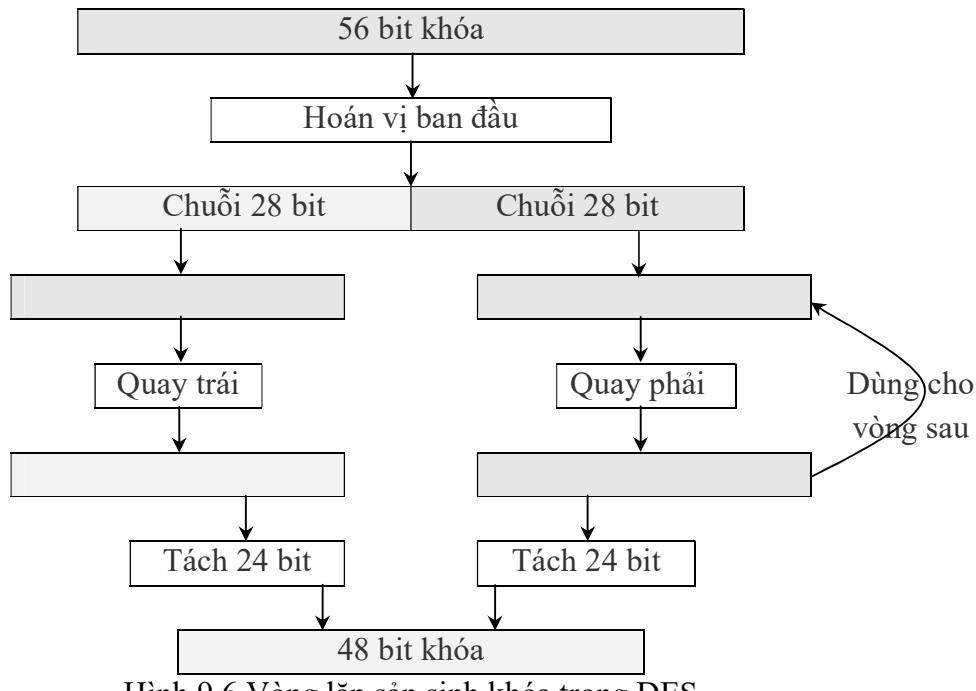
Khóa nào được giữ bí mật sẽ tùy thuộc vào việc sử dụng khóa, ví dụ nếu A muốn gửi thông điệp bí mật cho B thì A sẽ sử dụng mã khóa công khai do B cung cấp, B nắm giữ khóa bí mật riêng nên chỉ có B mới có thể giải mã được thông điệp. Ngược lại nếu B muốn đảm bảo chắc chắn thông điệp đến từ A thì A phải sử dụng khóa bí mật riêng của mình để mã hóa thông điệp, B sử dụng khóa công khai của A để giải mã, nếu giải mã thành công thì chắc chắn thông điệp là của A.

Ứng dụng cuối cùng của mã hóa trong hệ thống phân tán là việc sử dụng các hàm băm, đầu vào của hàm băm là thông điệp m có độ dài tùy ý nhưng đầu ra là chuỗi các bit có độ dài cố định  $h=H(m)$ . Hàm băm là hàm một chiều, việc tính toán hàm băm đơn giản nhưng không thể tìm thấy thông điệp m tương ứng với giá trị trả về của hàm băm. Nếu hai thông điệp khác nhau thì hàm băm sẽ trả về những giá trị khác nhau, ngược lại không thể tìm thấy hai thông điệp khác nhau mà có cùng giá trị trả về của hàm băm. Các đặc tính trên đã được áp dụng cho việc sản sinh cặp khóa bí mật và khóa công khai.

#### 9.1.3.1 Giải thuật mã hóa DES

Giải thuật DES dùng cho các hệ thống mã hóa đối xứng, giải thuật thực hiện trên các khối dữ liệu 64 bit, mỗi khối được mã hóa qua 16 vòng lặp, mỗi vòng có một khóa mã hóa 48 bit riêng, 16 khóa này được sinh ra từ 56 bit khóa chính. Đầu vào của vòng lặp mã hóa thứ i là dữ liệu đã được mã hóa của vòng lặp thứ i-1, 64 bit dữ liệu qua mỗi vòng lặp được chia thành hai phần bằng nhau  $L_{i-1}$  và  $R_{i-1}$ , cùng bằng 32 bit. Phần dữ liệu bên phải  $R_{i-1}$  được lấy làm phần bên trái của dữ liệu cho vòng sau:  $R_{i-1} = L_i$ . Hàm f với đầu vào là  $R_{i-1}$  và khóa  $K_i$  sinh ra khối 32 bit được XOR với  $L_{i-1}$  để sinh ra  $R_i$ . Mỗi khóa 48 bit cho mỗi vòng lặp được sinh ra từ khóa chính 56 bit như sau: hoán vị khóa chính, chia đôi

thành hai phần 28 bit. Tại mỗi vòng, mỗi một nửa đó sẽ quay trái một hoặc hai bit, sau đó lấy ra 24 bit kết hợp với 24 bit của nửa còn lại tạo ra khóa.



Hình 9.6 Vòng lặp sản sinh khóa trong DES

Nguyên lý của DES hoàn toàn đơn giản nhưng rất khó bẻ khóa nếu sử dụng các phương pháp phân tích, lý do vì sao vẫn chưa được giải thích công khai. Sử dụng tấn công thăm dò bằng cách tìm kiếm sẽ làm cho công việc bẻ khóa trở nên dễ dàng hơn, vì vậy Triple DES cải tiến giải thuật bằng cách sử dụng DES ba lần với các khóa khác nhau sẽ làm tăng độ an toàn lên rất nhiều. Giải thuật DES đã được sử dụng làm kỹ thuật mã hóa chuẩn trong nhiều năm, tuy nhiên gần đây đang được thay thế bằng giải thuật Rijndael với khối 128 bit hoặc lớn hơn. Giải thuật Rijndael được thiết kế để chạy nhanh hơn và như vậy có thể được sử dụng trong các thẻ thông minh.

#### 9.1.3.2 Giải thuật mã hóa RSA

Giải thuật RSA được sử dụng rộng rãi trong các hệ thống khóa công khai, tính bảo mật của giao thức xuất phát từ thực tế chưa có phương pháp nào hiệu quả để tìm ra số nguyên tố rất lớn. Như chúng ta đã biết, mỗi số nguyên dương đều có thể viết dưới dạng tích của các số nguyên tố, trong RSA các khóa riêng và khóa công khai được tạo ra từ các số nguyên tố rất lớn với hàng trăm chữ số, việc bẻ khóa sẽ phải tốn rất nhiều thời gian. Cách sinh khóa của giải thuật RSA thực hiện theo bốn bước:

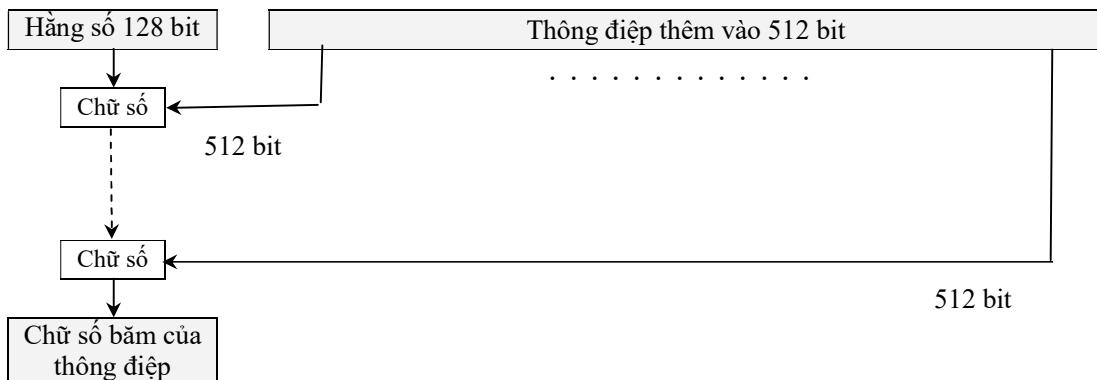
1. Chọn 2 số nguyên tố lớn khác nhau  $p$  và  $q$
2. Tính  $n = p \cdot q$  và  $\phi = (p-1) \cdot (q-1)$
3. Chọn một số tự nhiên  $d$  sao cho  $1 < d < \phi$  và nguyên tố cùng nhau với  $\phi$
4. Tính toán  $e$  sao cho  $e \cdot d \equiv 1 \pmod{\phi}$ , nghĩa là tìm số tự nhiên  $x$  sao cho  $e \cdot (x \cdot \phi + 1) \equiv 1 \pmod{d}$  cũng là số tự nhiên.

Như vậy d và e tạo thành một cặp duy nhất, nếu dùng d để giải mã thì e sẽ dùng để mã hóa, có thể công khai một trong hai số này. Giải thuật RSA coi thông điệp được truyền đi là một dãy các số nhị phân, chia thông điệp m thành các khối có kích thước cố định  $m_i$  sao cho  $0 \leq m_i \leq n$ . Ở bên gửi, mỗi khối  $m_i$  sẽ tính giá trị  $c_i = m_i^e \pmod{n}$  rồi gửi đi. Bên nhận sẽ giải mã bằng cách tính  $m_i = c_i^d \pmod{n}$ , như vậy để mã hóa cần biết e và n còn để giải mã thì cần biết d và n. So với giải thuật mã hóa đối xứng, giải thuật RSA tính toán phức tạp hơn do đó thời gian thực hiện sẽ gấp hàng trăm lần. Vì vậy giải thuật RSA chỉ nên áp dụng để trao đổi các khóa bí mật chứ không dùng cho việc mã hóa các dữ liệu thông thường.

#### 9.1.3.3 Hàm băm MD5

Hàm băm MD5 được sử dụng rộng rãi trong các giải thuật mã hóa, nó tính toán xâu ký tự nhị phân bất kỳ để cho về giá trị băm 128 bit và thường thể hiện dưới dạng chữ số hệ hexa. Mật khẩu của người sử dụng nên được mã hóa bằng giải thuật này, thời gian thực hiện nhanh và người quản trị hệ thống cũng không thể biết bản rõ của mật khẩu.

Dữ liệu đầu vào trước hết được thêm vào sao cho số dư của chiều dài sau khi thêm là 448 bits, sau đó thêm 64 bit thể hiện chiều dài của dữ liệu ban đầu, như vậy toàn bộ dữ liệu sẽ được chia thành các đoạn 512 bit. Giải thuật MD5 bắt đầu bằng hằng số 128 bit và thực hiện trong k pha, trong đó k là số lượng khối 512 bit, trong mỗi pha chữ số 128 bit được tính từ khối 512 bit của dữ liệu đầu vào và 128 bit của pha trước.



Hình 9.7 Qui trình mã hóa MD5

Mỗi pha trong MD5 gồm 4 vòng tính toán, mỗi vòng sử dụng một trong bốn hàm, mỗi hàm thao tác với các biến số 32 bit x, y và z:

$$\begin{aligned}
 F(x.y.z) &= (x \text{ AND } y) \text{ OR } (\text{NOT } x) \text{ AND } z \\
 G(r.y.z) &= (x \text{ AND } z) \text{ OR } (y \text{ AND } (\text{NOT } z)) \\
 H(x.y.z) &= x \text{ XOR } y \text{ XOR } z \\
 I(z.y.z) &= y \text{ XOR } (x \text{ OR } (\text{NOT } z))
 \end{aligned}$$

Ví dụ, khối b được chia thành 16 khối con dài 32 bit ký hiệu  $b_0$  đến  $b_{15}$ , mỗi khối con sẽ ký hiệu là (p.q.r.s). Tại vòng thứ nhất, hàm f dùng để thay đổi bốn biến số (p.q.r.s) trong 16 thao tác lặp như trên hình 9.8, có tổng cộng 64 hằng số định nghĩa trước  $C_i$ , ký hiệu  $x << n$  nghĩa là quay sang trái, các bit của biến x sẽ được dịch sang bên trái n vị trí, bit bên trái nhất sẽ chuyển về vị trí của bit bên phải nhất.

<b>Thao tác 1-8</b>	<b>Thao tác 9-15</b>
$p \leftarrow (p+F(q,r,s)+b_0+C_1) <<< 7$	$p \leftarrow (p+F(q,r,s)+b_8+C_9) <<< 7$
$s \leftarrow (s+F(p,q,r)+b_1+C_2) <<< 12$	$s \leftarrow (s+F(p,q,r)+b_9+C_{10}) <<< 12$
$r \leftarrow (r+F(s,p,q)+b_2+C_3) <<< 17$	$r \leftarrow (r+F(s,p,q)+b_{10}+C_{11}) <<< 17$
$q \leftarrow (q+F(r,s,p)+b_3+C_4) <<< 22$	$q \leftarrow (q+F(r,s,p)+b_{11}+C_{12}) <<< 22$
$p \leftarrow (p+F(q,r,s)+b_4+C_5) <<< 7$	$p \leftarrow (p+F(q,r,s)+b_{12}+C_{13}) <<< 7$
$s \leftarrow (s+F(p,q,r)+b_5+C_6) <<< 12$	$s \leftarrow (s+F(p,q,r)+b_{13}+C_{14}) <<< 12$
$r \leftarrow (r+F(s,p,q)+b_6+C_7) <<< 17$	$r \leftarrow (r+F(s,p,q)+b_{14}+C_{15}) <<< 17$
$q \leftarrow (q+F(r,s,p)+b_7+C_8) <<< 22$	$q \leftarrow (q+F(r,s,p)+b_{15}+C_{16}) <<< 22$

Hình 9.8 Thao tác lặp vòng đầu của pha trong MD5

Các biến này được thực hiện cho mỗi vòng tiếp theo và sau khi một pha kết thúc được chuyển sang pha kế tiếp. Vòng thứ hai sử dụng hàm G theo cách tương tự, hàm H sử dụng cho vòng thứ 3 và I sử dụng cho vòng thứ 4, cứ lặp lại như vậy cho đến khi hoàn thành. Như vậy mỗi vòng sẽ gồm 64 thao tác lặp lại sau đó lại bắt đầu pha mới với các giá trị (p.q.r.s) ở tại điểm này.

## 9.2 Các kênh bảo mật

Mô hình khách chủ thường được sử dụng để xây dựng hệ thống phân tán, máy chủ cũng có thể đóng vai trò máy khách đối với các máy chủ khác. Bảo mật cho hệ thống phân tán phải bảo đảm thông tin được an toàn trên các máy chủ và trên đường truyền mạng. Bảo mật dữ liệu trên mỗi máy chủ liên quan tới vấn đề kiểm soát quyền truy nhập, bảo mật truyền thông đòi hỏi xác thực các bên tham gia.

Trong nhiều trường hợp, bảo mật truyền thông yêu cầu tính toàn vẹn và bí mật dữ liệu, đặc biệt truyền thông nội bộ giữa các máy chủ. Vấn đề bảo mật truyền thông có thể thực hiện bằng cách thiết lập kênh bảo mật giữa các bên tham gia, nó sẽ bảo vệ chống lại các hiểm họa bảo mật. Chống lại hiểm họa chặn thông điệp bằng các phương tiện mã hóa, chống lại hiểm họa sửa đổi và bịa đặt bằng cách sử dụng các giao thức xác thực lẫn nhau, phản tiếp theo sẽ đề cập tới cách sử dụng các giao thức này.

### 9.2.1 Xác thực

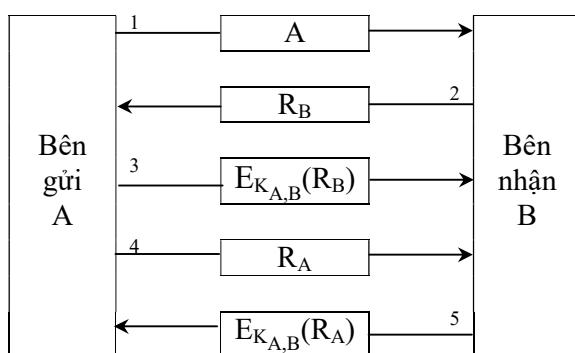
Xác thực và đảm bảo toàn vẹn dữ liệu luôn phải đi kèm với nhau, một hệ thống đảm bảo tính xác thực nhưng không đảm bảo toàn vẹn thì cũng không có ý nghĩa trong bảo mật dữ liệu, thông điệp hoàn toàn có thể bị sửa đổi khi di chuyển trên đường truyền. Tương tự như vậy một hệ thống bảo đảm tính toàn vẹn nhưng không bảo đảm tính xác thực thì cũng không có ý nghĩa, kẻ tấn công hoàn toàn có thể giả mạo bên gửi hoặc bên nhận. Trong nhiều giao thức hai công việc này được cài đặt bằng cách thực hiện xác thực khi thiết lập kênh liên kết các bên tham gia, sau đó bảo đảm tính toàn vẹn dữ liệu bằng cách sử dụng mã hóa khóa bí mật cho mỗi phiên giao dịch, khóa bí mật của mỗi phiên sẽ bị hủy bỏ khi kết thúc kênh liên kết.

#### 9.2.1.1 Xác thực dựa trên khóa bí mật

Xác thực dựa trên khóa bí mật giả thiết tất cả các bên tham gia hai đều phải biết khóa bí mật, làm thế nào để trao đổi khóa này sẽ đề cập trong các phần sau. Giao thức

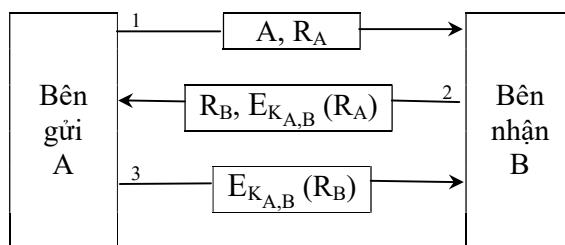
được thực hiện theo cách thông thường, các bên tham gia sẽ không nghi ngờ nhau nếu các bên đều biết khóa bí mật.

Hình 9.9 minh họa trao đổi thông tin giữa A và B để thực hiện xác thực dựa trên khóa bí mật, cả A và B đều phải biết khóa bí mật  $K_{A,B}$ . Đầu tiên, A gửi cho B định danh của mình để yêu cầu thiết lập kênh truyền, B sẽ gửi lại A một số ngẫu nhiên  $R_B$ , A sử dụng khóa bí mật  $K_{A,B}$  để trả về cho B giá trị đã mã hóa  $E_{K_{A,B}}(R_B)$ . Nhận được giá trị từ A, B sẽ giải mã  $D_{K_{A,B}}(E_{K_{A,B}}(R_B))$ , nếu kết quả trùng khớp với  $R_B$  thì B có thể tin cậy đối tượng đang thiết lập liên kết chính xác là A. Tuy nhiên A vẫn chưa biết đối tượng đang thiết lập liên kết có chính xác là B hay không, vì vậy A sẽ gửi tiếp một số ngẫu nhiên  $R_A$  cho B, B trả về giá trị mã hóa  $E_{K_{A,B}}(R_A)$ , nhận được giá trị này A sẽ giải mã  $D_{K_{A,B}}(E_{K_{A,B}}(R_A))$ , nếu kết quả trùng khớp với  $R_A$  thì A có thể tin cậy đối tượng đang thiết lập liên kết chính xác là B.



Hình 9.9 Xác thực dựa trên khóa bí mật thực hiện 5 bước

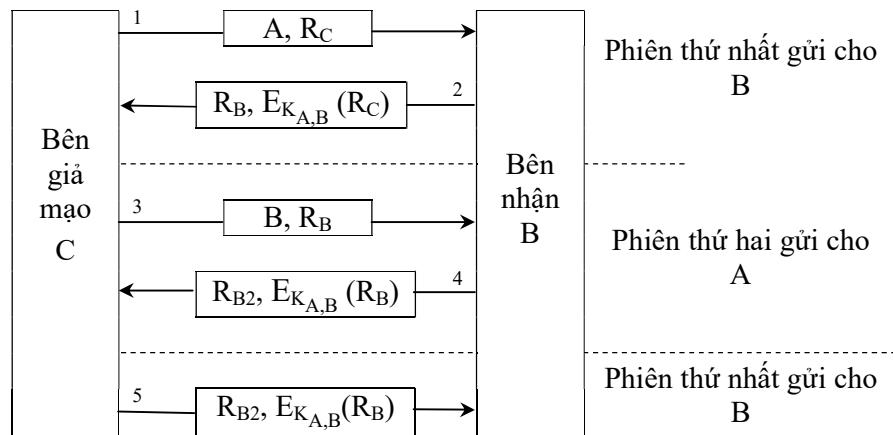
Qui trình thực hiện như trên đòi hỏi phải trao đổi năm thông điệp, hình 9.10 thể hiện phiên bản rút gọn gộp thông điệp thứ tư vào thông điệp thứ nhất và thông điệp thứ năm vào thông điệp thứ hai, như vậy sẽ chỉ còn trao đổi ba thông điệp và sẽ tiết kiệm thời gian xác thực giữa các bên tham gia. Đầu tiên, A gửi cho B định danh của mình và một số ngẫu nhiên  $R_A$  để yêu cầu thiết lập kênh truyền, B sẽ gửi lại A một số ngẫu nhiên  $R_B$  kèm theo giá trị mã hóa  $E_{K_{A,B}}(R_A)$ , A sử dụng khóa bí mật  $K_{A,B}$  để trả về cho B giá trị đã mã hóa  $E_{K_{A,B}}(R_B)$ . Nhận được giá trị từ A, B sẽ giải mã  $D_{K_{A,B}}(E_{K_{A,B}}(R_B))$ , nếu kết quả trùng khớp với  $R_B$  thì B có thể tin cậy đối tượng đang thiết lập liên kết chính xác là A.



Hình 9.10 Xác thực dựa trên khóa bí mật thực hiện 3 bước

Rất tiếc giao thức rút gọn này rất dễ bị tấn công phản xạ, giả sử tình huống C muốn mạo danh A để xác thực với B mà không cần biết khóa bí mật  $K_{A,B}$ . Quá trình giả mạo

tiến hành bằng cách C thiết lập phiên làm việc thứ nhất để gửi cho B thông điệp chúa định danh của A và số ngẫu nhiên  $R_C$ , B sẽ trả về  $R_B, E_{K_{A,B}}(R_C)$ , tuy nhiên C không biết khóa bí mật nên nó thiết lập một phiên khác và gửi cho A thông điệp C,  $R_B$ , A sẽ mã hóa  $E_{K_{A,B}}(R_B)$  và gửi cho C, tất nhiên C sẽ lấy giá trị này và trả về phiên làm việc thứ nhất để gửi cho B, như vậy B tin rằng C chính là A. Nói chung, cho phép các bên tham gia thiết lập kênh an toàn thực hiện một số công việc như nhau không phải là ý tưởng tốt.

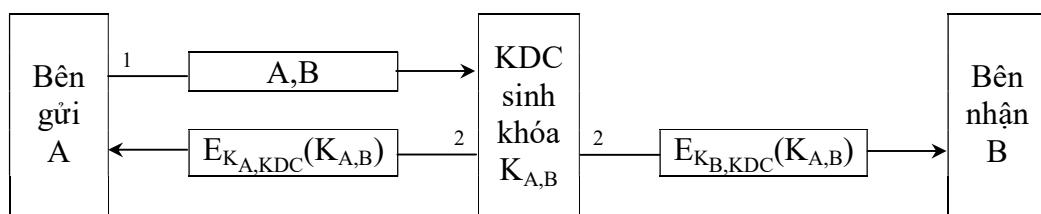


Hình 9.11 Tấn công phản xạ

Trong ví dụ trên, B đã vi phạm qui tắc an toàn khi cung cấp thông tin hữu ích  $R_B, E_{K_{A,B}}(R_C)$  trong khi chưa biết C là ai, nhưng điều này lại không vi phạm qui tắc của giao thức rút gọn. Bài học ở đây là vấn đề thiết kế các giao thức bảo mật thực tế khó khăn hơn rất nhiều so với những hình dung ban đầu, việc tinh chỉnh giao thức để tăng hiệu năng có thể ảnh hưởng tới tính chính xác của giao thức như đã minh chứng trên.

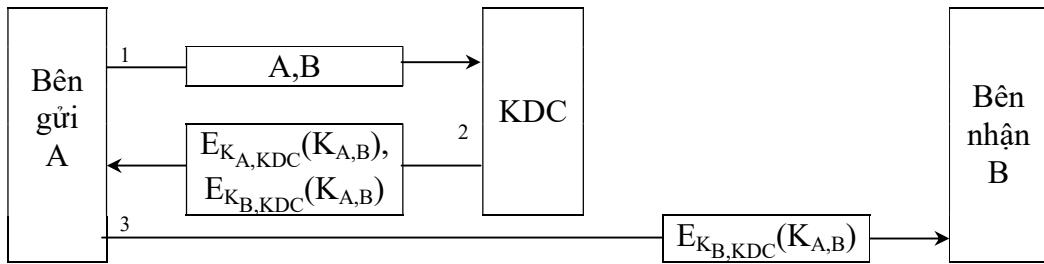
#### 9.2.1.2 Xác thực sử dụng trung tâm phân phối khóa

Xác thực dựa trên khóa bí mật sẽ không phù hợp cho hệ thống qui mô lớn, giả sử có N máy thì mỗi máy sẽ phải giữ N-1 khóa và cả hệ thống sẽ cần tới  $N(N-1)/2$  khóa. Để khắc phục vấn đề này có thể sử dụng trung tâm phân phối khóa KDC, trung tâm chia sẻ khóa bí mật với mỗi máy nhưng không có cặp máy nào phải có khóa bí mật, trung tâm chỉ cần giữ N khóa. Hình 9.12 thể hiện nguyên lý đơn giản trong cấp phát khóa, nếu A muốn thiết lập kênh an toàn với B thì A sẽ yêu cầu KDC trợ giúp bằng cách gửi khóa bí mật cho cả A và B.



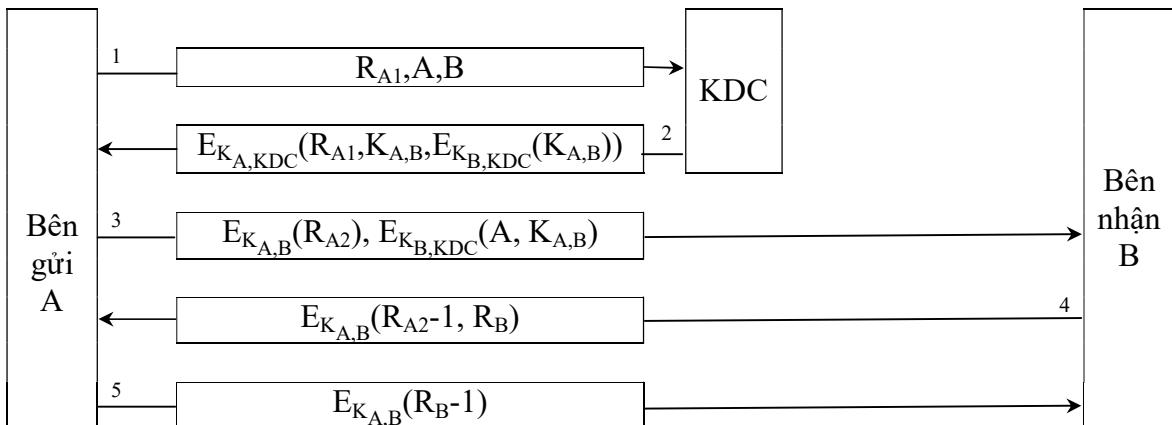
Hình 9.12 Nguyên lý sử dụng trung tâm phân phối khóa

Đầu tiên A gửi thông điệp cho KDC để đạt nguyện vọng muốn kết nối với B, KDC sẽ sinh khóa  $K_{A,B}$  sau đó mã hóa  $E_{K_{A,KDC}}(K_{A,B})$  gửi cho A và  $E_{K_{B,KDC}}(K_{A,B})$  gửi cho B. Nhược điểm chính của giải pháp này là A có thể bắt đầu thiết lập kênh an toàn với B trong khi B chưa nhận được khóa bí mật từ KDC, điều này có thể khắc phục bằng cách KDC sẽ chuyển thẻ  $E_{K_{B,KDC}}(K_{A,B})$  cho A để A tự kết nối với B như minh họa trên hình 9.13, B vẫn là đối tượng duy nhất biết cách giải mã thẻ  $E_{K_{B,KDC}}(K_{A,B})$ , các phiên bản của giải pháp này là các giao thức Needham-Schroeder và Kerberos.



Hình 9.13 KDC cho phép hai bên tự kết nối

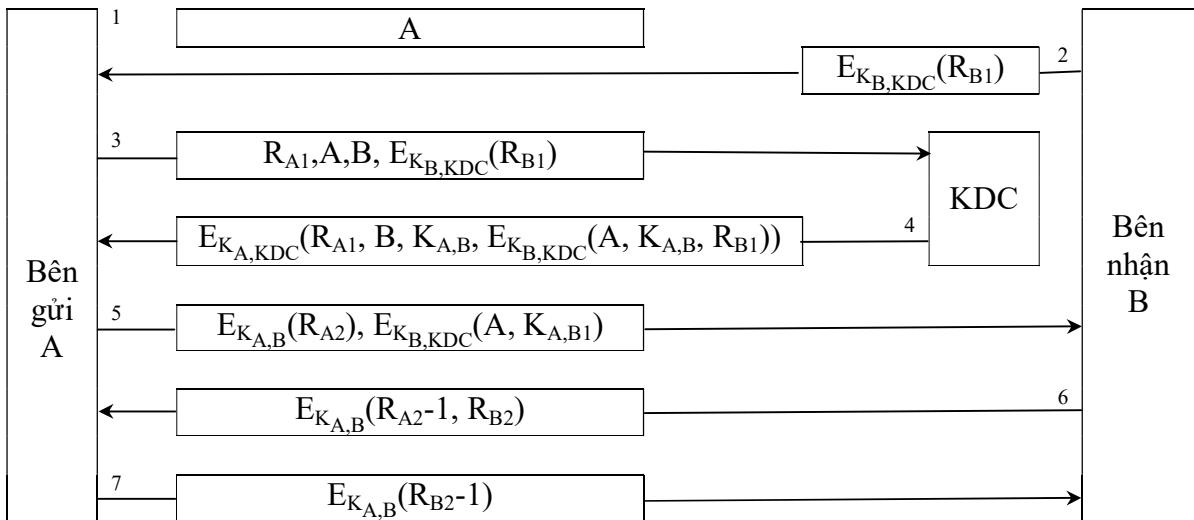
Nếu sử dụng giao thức Needham-Schroeder, A sẽ gửi yêu cầu đến KDC chứa thẻ mời B bao gồm định danh của A và số ngẫu nhiên  $R_{A1}$ . KDC sẽ trả lời bằng cách trả về giá trị mã hóa  $E_{K_{A,KDC}}(R_{A1}, B, K_{A,B}, E_{K_{B,KDC}}(A, K_{A,B}))$ , các giá trị  $R_{A1}$  và B trong tham số hàm mã hóa để A kiểm chứng với giá trị nó đã gửi cho KDC nhằm mục đích tránh tấn công phản xạ, thẻ  $E_{K_{B,KDC}}(A, K_{A,B})$  để A có thể thiết lập liên kết với B.



Hình 9.14 Giao thức Needham-Schroeder

Sau khi nhận được kết quả từ KDC, A sẽ sinh số ngẫu nhiên  $R_{A2}$  và thực hiện mã hóa  $E_{K_{A,B}}(R_{A2})$ ,  $E_{K_{B,KDC}}(A, K_{A,B})$  để gửi cho B, nhận được bản tin này B sẽ giải mã  $D_{K_{B,KDC}}(E_{K_{B,KDC}}(A, K_{A,B}))$  và hiểu rằng A muốn thiết lập liên kết với khóa bí mật là  $K_{A,B}$ , dựa vào khóa này sẽ giải mã  $D_{K_{A,B}}(E_{K_{A,B}}(R_{A2}))$  và biết được giá trị  $R_{A2}$ . B sẽ sinh số ngẫu nhiên  $R_B$  và gửi cho A giá trị  $E_{K_{A,B}}(R_{A2}-1, R_B)$ ,  $R_{A2}-1$  chứ không phải  $R_{A2}$  ngũ ý B đã biết khóa bí mật và sử dụng khóa đó giải mã thành công, nhận được bản tin A sẽ gửi

cho B giá trị  $E_{K_{A,B}}(R_B-1)$ . Như vậy giao thức đã chống được tấn công phản xạ, tuy nhiên nếu kẻ tấn công C biết được khóa cũ  $K_{A,B}$  thì sẽ gửi lại thông điệp thứ 3 để giả mạo A, trong trường hợp này cần phải tạo mối ràng buộc giữa thông điệp thứ ba với thông điệp đầu tiên, giải pháp được thể hiện trên hình 9.15.

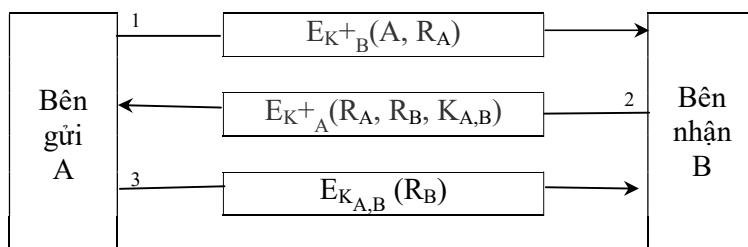


Hình 9.15 Giao thức Needham-Schroeder chống lại tái sử dụng khóa phiên trước

Bí quyết nằm ở chỗ trước khi kết nối với KDC, A sẽ gửi thông điệp yêu cầu kết nối với B, khi đó B sinh ra số ngẫu nhiên  $R_{B1}$  và trả về cho A giá trị mã hóa  $E_{K_{B,KDC}}(R_{B1})$ , tất nhiên A sẽ không giải mã được giá trị này mà chỉ gửi kèm theo thông điệp gửi đến KDC. Nhận được yêu cầu từ A, KDC sẽ giải mã và biết được giá trị  $R_{B1}$  nên sẽ gửi cho A thông điệp trả về giá trị  $E_{K_{A,KDC}}(R_{A1}, B, K_{A,B}, E_{K_{B,KDC}}(A, K_{A,B}, R_{B1}))$ , các bước tiếp theo tương tự như giao thức cơ bản đã trình bày trên.

#### 9.2.1.3 Xác thực dựa trên mã hóa khóa công khai

Hình 9.16 thể hiện qui trình xác thực dựa trên mã hóa khóa công khai đòi hỏi mỗi bên cần phải biết khóa công khai của nhau, ký hiệu  $K^+$  là khóa công khai và  $K^-$  là khóa riêng bí mật. A bắt đầu phiên bằng cách sinh số ngẫu nhiên  $R_A$  và sử dụng khóa công khai  $K_B^+$  để mã hóa bản tin  $E_{K_B^+}(A, R_A)$  và gửi cho B, nhận được thông điệp này B sẽ sử dụng khóa riêng bí mật  $K_A^-$  để giải mã, nếu thành công sẽ sinh số ngẫu nhiên  $R_B$  và khóa phiên  $K_{A,B}$  sau đó sử dụng khóa công khai  $K_A^+$  để mã hóa bản tin  $E_{K_A^+}(R_A, R_B, K_{A,B})$  và gửi cho A.



Hình 9.16 Xác thực lẩn nhau dựa trên mã hóa khóa công khai

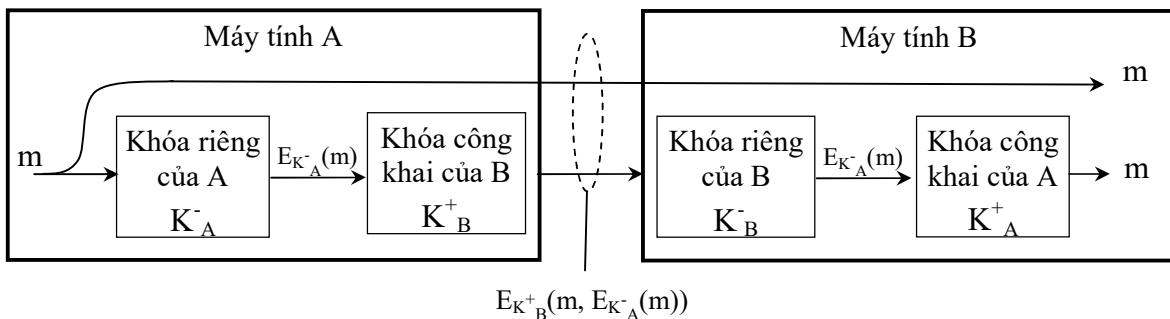
Nhận được bản tin A sẽ sử dụng khóa riêng  $K_A^-$  để giải mã, nếu thành công sẽ sử dụng khóa phiên  $K_{A,B}$  để mã hóa bản tin  $E_{K_{A,B}}(R_B)$ . B nhận được bản tin sẽ sử dụng khóa bí mật của phiên để giải mã, nếu giá trị giải mã bằng  $R_B$  thì B tin rằng đối tượng thiết lập kết nối chắc chắn là A.

### 9.2.2 Toàn vẹn và bí mật thông điệp

Bên cạnh việc xác thực, kênh an toàn còn phải đảm bảo tính toàn vẹn và bí mật cho thông điệp, tính toàn vẹn bảo vệ thông điệp chống lại những sửa đổi gian lận trong khi đó tính bí mật đảm bảo thông điệp không bị đọc trộm. Tính bí mật có thể thực hiện được bằng cách sử dụng khóa bí mật của các bên hoặc mã khóa công khai của bên nhận để mã hóa thông điệp trước khi gửi, tuy nhiên bảo vệ thông điệp không bị sửa đổi là vấn đề phức tạp hơn nhiều.

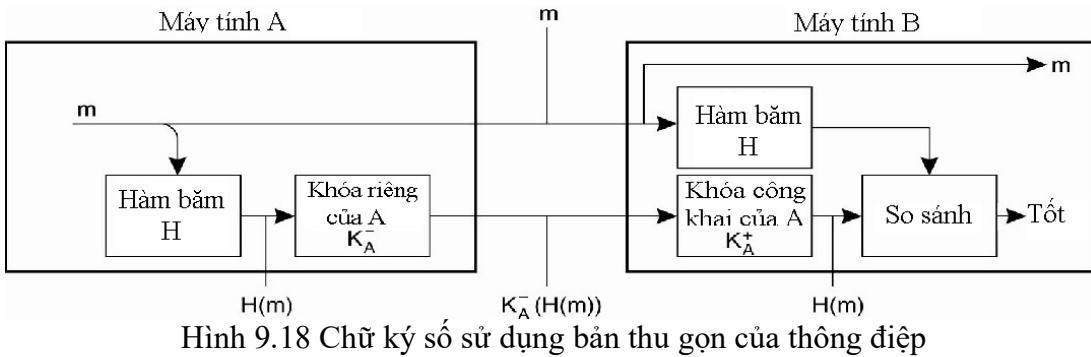
#### 9.2.2.1 Chữ ký số

Xét một giao dịch thương mại điện tử, A gửi thông điệp cho B đồng ý mua hàng, có thể xảy ra hai tình huống gian lận: A phủ nhận thanh toán vì cho rằng đó là thông điệp giả mạo, B nâng tổng số tiền mà A đã mặc cả. Hai tình huống này có thể giải quyết nếu A sử dụng chữ ký số cho thông điệp đã gửi, sự kết hợp duy nhất giữa thông điệp và chữ ký số bảo vệ không bị sửa đổi, A cũng không thể phủ nhận nếu chứng minh được đó là chữ ký của A. Một trong những cách phổ biến là sử dụng mã hóa khóa công khai như RSA như minh họa trên hình 9.17, A sử dụng khóa riêng  $K_A^-$  để mã hóa gửi thông điệp gửi cho B, nếu muốn bí mật nội dung thì có thể sử dụng khóa công khai  $K_B^+$ , như vậy thông điệp gửi lên kênh truyền sẽ là  $E_{K_B^+}(m, E_{K_A^-}(m))$ .



Hình 9.17 Chữ ký số sử dụng khóa mã hóa công khai

Nhận được thông điệp, B sẽ sử dụng khóa riêng của mình  $K_B^-$  để giải mã, như vậy sẽ nhận được bản rõ và bản mã của thông điệp. Bên gửi sẽ không thể nghi ngờ bên nhận đã sửa đổi thông điệp vì chỉ sử dụng khóa công khai  $K_A^+$  để giải mã  $E_{K_A^-}(m)$ , nếu kết quả giải mã trùng khớp với bản rõ  $m$  chứng tỏ bản tin đã không bị sửa đổi và đó là bản tin mà A đã ký trước khi gửi đi, bên gửi cũng không thể chối bỏ trách nhiệm của mình. Tuy nhiên, tính hợp lệ của chữ ký có thể duy trì khi khóa riêng của bên gửi không bị lộ, vì vậy bên A vẫn có thể chối bỏ với lý do khóa riêng đã bị đánh cắp trước khi gửi thông điệp.



Hình 9.18 Chữ ký số sử dụng bản thu gọn của thông điệp

Bên gửi hoàn toàn có quyền thay đổi khóa riêng của mình, đây là việc bình thường vì thường xuyên thay đổi khóa sẽ giúp chống lại xâm nhập trái phép. Khi bên A thay đổi khóa thì những thông điệp đã gửi cho B sẽ không có giá trị, trường hợp này đòi hỏi phải có một trung tâm quản lý lưu vết khóa để ghi nhận thời điểm thay đổi và sử dụng thêm nhãn thời gian cho chữ ký điện tử. Ngoài ra có thể thấy việc mã hóa toàn bộ thông điệp là công việc khá tốn kém về mặt kỹ thuật, một giải pháp khác đơn giản hơn là chỉ cần sử dụng chữ ký cho bản thu gọn của thông điệp. Bản thu gọn thông điệp là xâu bit có chiều dài cố định và được tính toán từ thông điệp có chiều dài bất kỳ bằng cách sử dụng hàm băm, như vậy thay cho việc mã hóa cho toàn bộ thông điệp  $m$  thì chỉ cần mã hóa cho xâu bit có chiều dài cố định  $H(m)$ , hình 9.18 minh họa cho quá trình này. Bên B sẽ chỉ cần giải mã  $H(m)$  bằng khóa công khai của bên A sau đó so sánh với kết quả thực hiện hàm băm trên bản rõ nhận được, nếu trùng khớp thì có thể cam kết đó là thông điệp do bên A đã ký trước khi gửi.

#### 9.2.2.2 Khóa phiên

Trong truyền thông qua kênh an toàn, sau khi hoàn thành xác thực nói chung các bên sẽ sử dụng khóa phiên chia sẻ duy nhất để đảm bảo bí mật, khóa phiên sẽ bị loại bỏ một cách an toàn khi kênh không còn được sử dụng nữa. Tất nhiên hoàn toàn có thể sử dụng lại các khóa này, tuy nhiên việc sử dụng lại có thể dễ dàng bị lộ, kẻ tấn công có thể chặn những thông điệp sử dụng cùng một khóa và tìm ra những đặc tính của khóa. Vì vậy, sẽ an toàn hơn nhiều nếu sử dụng các khóa xác thực càng ít càng tốt, nên sử dụng các cơ chế bằng ngoài và giữ ở mức tối thiểu việc trao đổi các khóa. Một lý do quan trọng khác, với việc sử dụng khóa phiên thì các bên tham gia có thể được bảo vệ trước những cuộc tấn công lặp lại, các thông điệp cần phải được gắn số tuần tự và nhãn thời gian.

Giả sử tính toàn vẹn và bảo mật thông điệp đã đạt được bằng cách sử dụng cùng một khoá khi thiết lập phiên, nếu bị lộ khóa thì kẻ đột nhập có thể giải mã các thông điệp đã trao đổi trong những phiên trước, như vậy sử dụng khóa phiên sẽ an toàn hơn rất nhiều, nếu bị lộ khóa thì kẻ tấn công chỉ có thể đọc được những thông điệp trong phiên đó. Ngoài ra, ngay cả những đối tượng nhận thông điệp cũng không đáng tin cậy, do đó chỉ cần khóa phiên là đủ để bảo đảm cho tính toàn vẹn và bảo mật thông điệp cho những đối tượng này, kết hợp sử dụng khóa lâu dài với khóa phiên để triển khai các kênh an toàn cũng là giải pháp tốt cần được áp dụng.

### **9.2.3 Truyền thông nhóm bảo mật**

Kênh truyền bảo mật không chỉ giới hạn cho hai thành viên tham gia, trong một số trường hợp có thể nhiều hơn hai thành viên, ví dụ trao đổi thông tin giữa các bản sao khi thực hiện nhân bản cũng phải đảm bảo các yêu cầu về bảo mật.

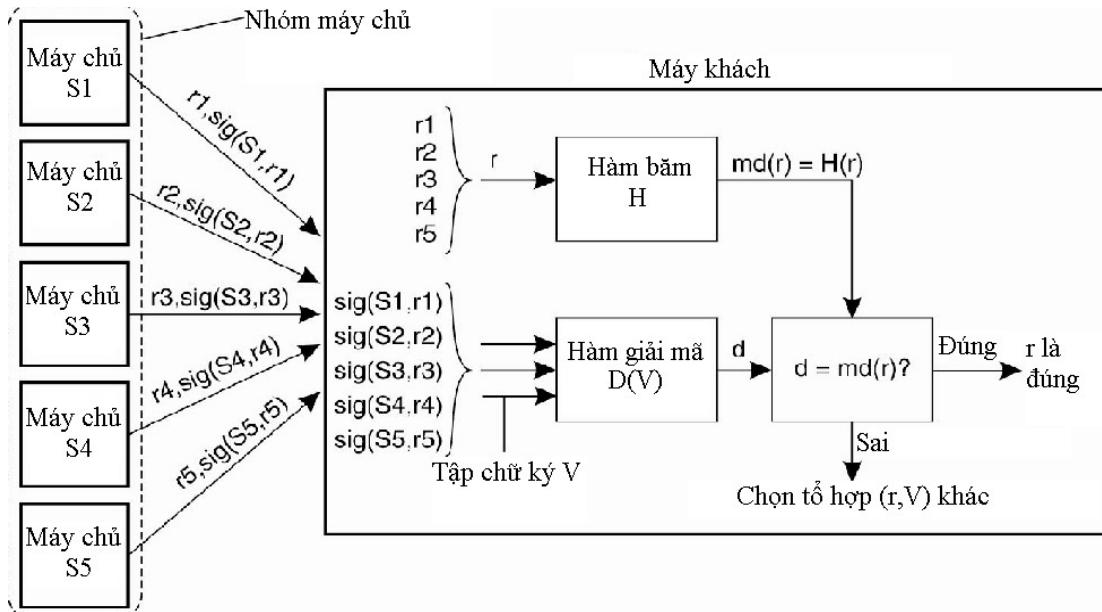
#### **9.2.3.1 Bí mật truyền thông trong nhóm**

Tính bí mật truyền thông trong nhóm được thực hiện đơn giản bằng cách yêu cầu tất cả các thành viên sử dụng khóa bí mật của nhóm để mã hóa và giải mã các thông điệp lưu chuyển giữa các thành viên trong nhóm, tất nhiên mỗi thành viên sẽ phải nghiêm túc thực hiện giữ bí mật cho khóa này. Giải pháp thứ hai phức tạp hơn nhưng lại đảm bảo bí mật hơn, từng cặp thành viên sẽ sử dụng một khóa bí mật, như vậy nếu nhóm có N thành viên sẽ cần phải có  $N(N - 1)/2$  khóa. Giải pháp thứ ba dung hòa hai giải pháp trên bằng cách sử dụng hệ thống mã hóa khóa công khai, mỗi thành viên sở hữu cặp khóa công khai và khóa riêng, khóa công khai sẽ chia sẻ cho tất cả các thành viên để gửi các thông điệp bí mật. Như vậy, chỉ cần quản lý N khóa, nếu một thành viên không còn đáng tin cậy thì chỉ cần loại bỏ khóa của thành viên đó ra khỏi nhóm mà không ảnh hưởng đến các khóa khác.

#### **9.2.3.2 Bảo mật các máy chủ nhân bản**

Xét trường hợp khá phức tạp, máy khách gửi yêu cầu đến nhóm các máy chủ nhân bản, các máy chủ có thể được nhân bản vì lý do đảm bảo tính chịu lỗi và hiệu năng, vì bất kỳ lý do nào thì máy khách chỉ mong đợi kết quả trả về đáng tin cậy. Xét trên góc độ bảo mật thông tin, máy khách không muốn trở thành nạn nhân của các vụ tấn công bảo mật, dù cho một trong số các máy chủ đã bị hư hỏng vì những cuộc tấn công đó. Máy khách có thể tập hợp kết quả trả về từ các máy chủ và xác thực chúng, nếu đa số đều được xác thực thì có thể tin cậy vào kết quả, tuy nhiên giải pháp này đã vi phạm tính trong suốt nhân bản.

Năm 1994 Reiter đã đề xuất giải pháp bảo mật máy chủ nhân bản nhưng vẫn duy trì được tính trong suốt nhân bản, máy khách không biết các bản sao thực sự, do đó có thể thêm hoặc bớt theo cách thức bí mật. Bản chất của bảo mật và các máy chủ nhân bản trong suốt nằm ở cách chia sẻ bí mật, tất cả đều biết bí mật nhưng không ai biết toàn bộ bí mật trừ khi họ tập trung lại với nhau. Trong trường hợp bảo mật, k trong số N máy chủ có thể trả về kết quả không chính xác và trong đó nhiều nhất là  $c \leq k$  thực sự bị ảnh hưởng bởi tấn công bảo mật, cần thiết phải phân biệt giữa lỗi máy chủ với lỗi do tấn công bảo mật. Xét trường hợp máy khách gửi yêu cầu đồng thời đến N máy chủ nhân bản, để bảo đảm bí mật nhóm các máy chủ nhân bản thì mỗi kết quả trả về phải đính kèm chữ ký số. Ký hiệu  $r_i$  là kết quả trả về của máy chủ  $S_i$ ,  $md(r_i)$  là bản thu gọn bằng cách sử dụng khóa riêng  $K_i$  của máy chủ  $S_i$ , hệ thống phải có khả năng chịu đựng được c máy chủ bị hủy hoại khi xảy ra tấn công bảo mật, như vậy vấn đề sẽ được giải quyết nếu có ít nhất  $c+1$  chữ ký để đảm bảo chữ ký hợp lệ cho kết quả trả về.



Hình 9.19 Chia sẻ chữ ký bí mật trong nhóm các máy chủ nhân bản

Ví dụ trên hình 9.19, nhóm gồm 5 máy chủ nhân bản như vậy có thể chịu được 2 máy chủ sụp đổ khi xảy ra tấn công bảo mật. Mỗi máy chủ  $S_i$  trả về kết quả  $r_i$  kèm theo bản tóm tắt  $md(r_i)$  trong chữ ký cho máy khách  $\text{sig}(S_i, r_i) = K_i^-(md(r_i))$ , như vậy máy khách sẽ nhận được 5 cặp dữ liệu này. Máy khách tính toán bản tóm tắt cho mỗi kết quả trả về  $r_i$ , nếu kết quả trả về  $r_i$  không chính xác thì sẽ bị phát hiện khi thực hiện giải mã  $D_{K_i^+}(md(r_i))$ , tuy nhiên phương pháp này không còn được sử dụng nữa vì không một máy chủ riêng nào được tin cậy. Thay vào đó, máy khách sử dụng hàm giải mã  $D$  với đầu vào gồm ba tập chữ ký và cho ra kết quả một bản tóm tắt:

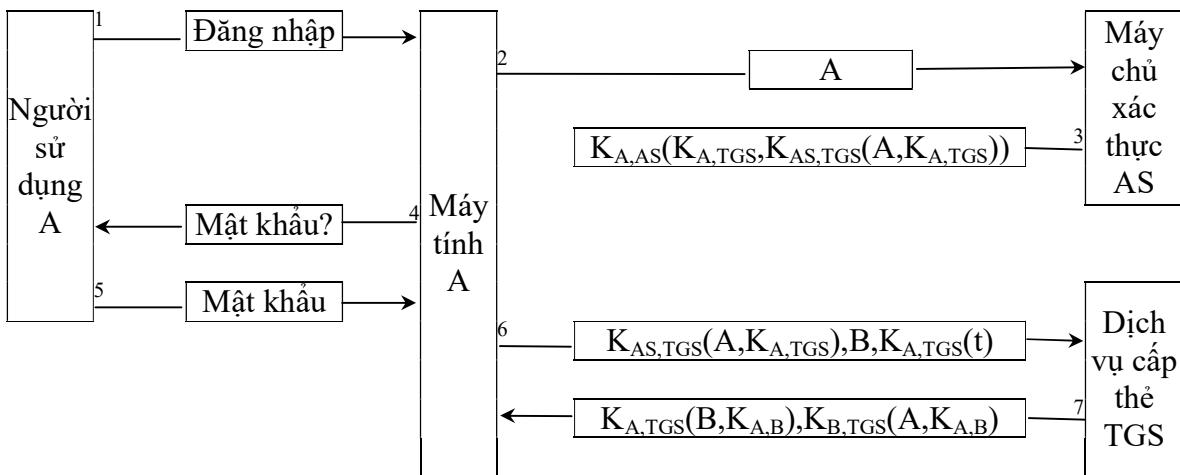
$$d = D(\text{sig}(S, r), \text{sig}(S', r'), \text{sig}(S'', r''))$$

Như vậy sẽ có tổng cộng  $5!/(3!2!) = 10$  khả năng xảy ra, chỉ cần một trường hợp cho kết quả chính xác bản tóm tắt  $md(r_i)$  đối với kết quả  $r_i$  nào đó thì có thể coi  $r_i$  là kết quả trả về chính xác, máy khách có thể tin cậy kết quả trả về nếu có ít nhất 3 máy chủ cho kết quả xác thực chính xác. Ví dụ trên có thể được tổng quát hóa trên trường hợp  $n=N$  máy chủ và ngưỡng chịu đựng  $m = c+1$  là ngưỡng chịu lỗi do bị tấn công bảo mật và gọi là lược đồ ngưỡng  $(m,n)$ , áp dụng thực tế một bản tin sẽ được chia thành  $n$  phần và chỉ có thể dựng lại bản tin khi biết chính xác ít nhất  $m$  phần.

#### 9.2.4 Xác thực bằng Kerberos

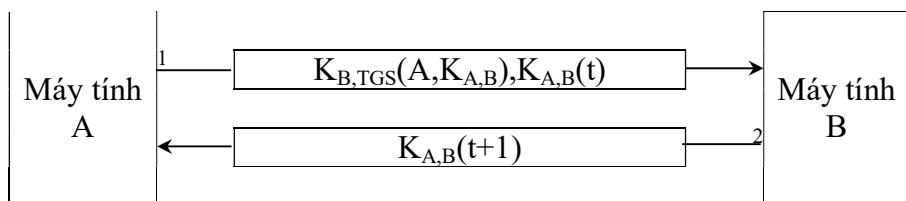
Có thể thấy việc kết hợp bảo mật vào trong hệ thống phân tán không phải đơn giản, chỉ cần một bộ phận không an toàn thì có thể tồn tại cho cả hệ thống. Để trợ giúp cho việc xây dựng các hệ thống phân tán có thể thực thi vô số các chính sách bảo mật, một số hệ thống hỗ trợ đã được phát triển và có thể sử dụng làm cơ sở cho việc phát triển tiếp, một trong những hệ thống đang được sử dụng rộng rãi là Kerberos. Kerberos dựa trên giao thức Needham-Schroeder và có thể coi như hệ thống bảo mật trợ giúp máy khách thiết lập kênh bí mật với máy chủ của hệ thống phân tán, tính bảo mật dựa trên các khóa bí mật chia sẻ. Hệ thống gồm hai thành phần, máy chủ xác thực AS có nhiệm vụ xử

lý các yêu cầu đăng nhập của người sử dụng, nếu xác thực người sử dụng thành công sẽ cung cấp khóa để thiết lập các kênh bảo mật với máy chủ, thiết lập kênh bảo mật với máy chủ do dịch vụ cấp thẻ TGS thực hiện.



Hình 9.20 Xác thực trong Kerberos

Khi A đăng nhập vào hệ thống phân tán sử dụng Kerberos để thiết lập kênh bảo mật với B, máy khách sẽ gửi tên đăng nhập đến máy chủ AS và nhận được khóa  $K_{A,TGS}$  và thẻ cần thiết để chuyển cho thành phần cung cấp dịch vụ TGS. Thẻ chứa định danh của A và chỉ có thành phần TGS mới đọc được thông tin này, vì vậy nó được mã hóa bằng khóa bí mật  $K_{AS,TGS}$ . Máy khách của A sẽ mã hóa mật khẩu bằng khóa  $K_{A,AS}$ , như vậy mật khẩu sẽ được mã hóa khi truyền trên mạng và máy khách cũng không lưu được bản rõ của mật khẩu, hơn nữa ngay sau khi khóa chia sẻ  $K_{AS}$  được sinh ra thì máy khách sẽ tìm thấy khóa phiên  $K_{AS,TGS}$ , có thể quên đi mật khẩu của A, chỉ sử dụng khóa bí mật chia sẻ  $K_{A,AS}$ .



Hình 9.21 Thiết lập kênh bảo mật trong Kerberos

Sau khi hoàn thành xác thực, A có thể coi như đã thâm nhập vào hệ thống và thẻ sẽ được lưu tạm thời trong AS khoảng 8 đến 24 giờ và dùng để truy nhập các dịch vụ từ xa, tất nhiên A phải xóa dữ liệu đã lưu trên máy khách. Nếu muốn nói chuyện với B, A phải yêu cầu khóa phiên cho B như trên hình 9.20, TGS sẽ trả về khóa phiên  $K_{A,B}$  trong thẻ để sau này A sẽ chuyển cho B. Thông điệp thứ 6 chưa nhãn thời gian đã được mã hóa để chống lại tấn công lặp, TGS sẽ kiểm tra nhãn thời gian nếu quá hạn thì sẽ bị từ chối. Nếu A muốn làm việc với máy chủ khác của hệ thống mà không thay đổi máy khách thì

không cần phải xác thực lại, về nguyên tắc các máy chủ đã ủy quyền xác thực cho máy chủ AS.

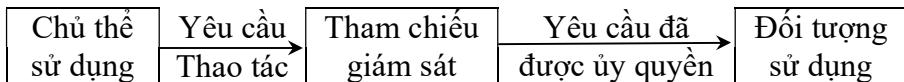
Việc thiết lập liên kết với B giờ đây đã trở nên đơn giản như minh họa trên hình 9.21, máy tính của A gửi cho máy tính của B thông điệp chứa thẻ đã nhận được cấp từ dịch vụ TGS kèm theo nhãn thời gian đã được mã hóa, máy tính của B nhận được sẽ giải mã và biết rằng A muốn nói chuyện với B vì chỉ có TGS mới xây dựng được thẻ này. Máy tính của B cũng nhận được khóa bí mật  $K_{A,B}$  để kiểm tra nhãn thời gian, bằng thông điệp trả về  $E_{K_{A,B}}(t+1)$  sẽ chứng minh cho A tính chính danh của B.

### 9.3 Kiểm soát truy nhập

Trong mô hình khách/chủ, mỗi khi máy khách và máy chủ thiết lập kênh bảo mật, máy khách có thể gửi yêu cầu để máy chủ thực hiện, những yêu cầu đó đòi hỏi thực hiện các thao tác trên tài nguyên do máy chủ kiểm soát. Tài nguyên trên máy chủ gồm nhiều đối tượng và máy khách chỉ được phép thao tác trên các đối tượng đó nếu được cấp quyền truy nhập, việc kiểm tra quyền truy nhập còn gọi là kiểm soát truy nhập, hai khái niệm này liên quan mật thiết với nhau. Để bảo mật hệ thống cần thiết phải xây dựng các chính sách kiểm soát các thao tác của người sử dụng trên máy chủ, có thể xây dựng bức tường lửa hoặc một hệ thống kiểm soát truy nhập.

#### 9.3.1 Nguyên lý kiểm soát truy nhập

Kiểm soát truy nhập đối tượng bao gồm tất cả những biện pháp bảo vệ đối tượng chống lại những thao tác bất hợp pháp trên đối tượng, đó có thể là những vấn đề liên quan tới quản lý hay những vấn đề liên quan tới thao tác trên đối tượng.

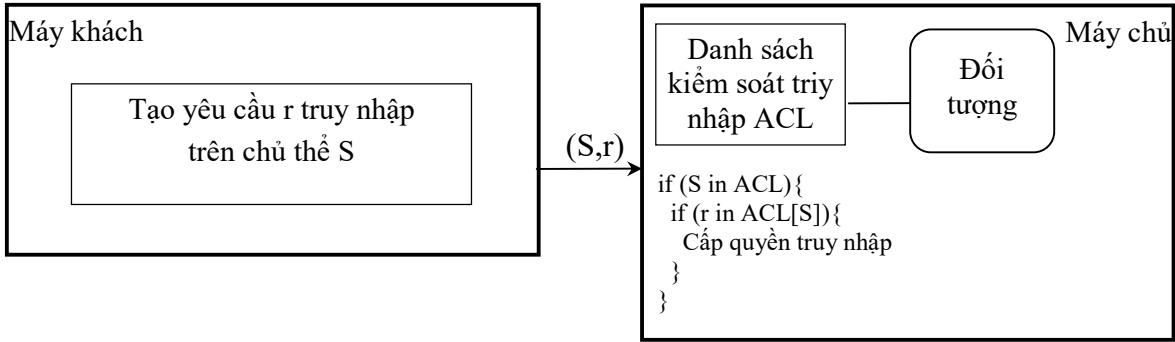


Hình 9.22 Nguyên lý kiểm soát truy nhập đối tượng

Nguyên lý kiểm soát truy nhập đối tượng được thể hiện trên hình 9.22, việc bảo vệ do thành phần giám sát tham chiếu đảm nhiệm, thành phần này chứa danh sách những chủ thẻ được phép thực hiện thao tác, nó sẽ được sử dụng mỗi khi gọi đối tượng, như vậy thành phần này đóng vai trò rất quan trọng và không thể bị kẻ tấn công đánh lừa.

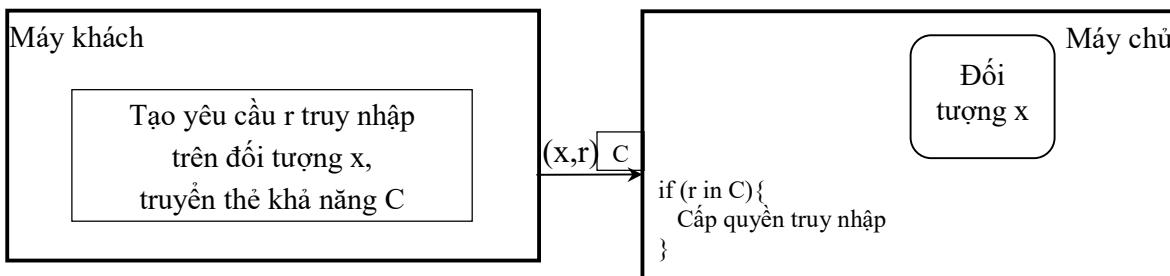
##### 9.3.1.1 Ma trận kiểm soát truy nhập

Cách đơn giản nhất thường được áp dụng là xây dựng ma trận kiểm soát truy nhập, đó là một ma trận gồm hàng biểu diễn cho một chủ thẻ truy nhập và cột biểu diễn cho tài nguyên, giá trị chứa trong mỗi phần tử của ma trận thể hiện quyền người sử dụng được thao tác trên tài nguyên đó. Ký hiệu ma trận là M, S là chủ thẻ truy nhập và O là đối tượng truy nhập, khi đó  $M[s,o]$  gồm danh sách những thao tác mà chủ thẻ S có thể thực hiện trên đối tượng O. Nếu chủ thẻ S thực hiện gọi thao tác m trên đối tượng O, thành phần giám sát tham chiếu sẽ kiểm tra xem thao tác m có xuất hiện trong danh sách  $M[s,o]$ , nếu có thì cho phép thực hiện, ngược lại sẽ không cho phép tiếp tục thực hiện và như vậy lời gọi thao tác coi như thất bại.



Hình 9.23 Danh sách kiểm soát truy nhập

Thông thường hệ thống sẽ có rất nhiều người sử dụng và có thể gồm hàng ngàn đối tượng cần được bảo vệ, do đó xây dựng một ma trận thực như trên là không hợp lý, nhiều mục trong ma trận sẽ rỗng. Giải pháp thứ nhất, hình 9.23 thể hiện danh sách kiểm soát truy nhập ACL, đó là ma trận được phân tán theo cột và những mục rỗng sẽ bị bỏ qua, mỗi đối tượng sẽ duy trì danh sách các quyền truy nhập của các chủ thẻ muốn truy nhập đối tượng, như vậy mỗi đối tượng sẽ có một danh sách kiểm soát truy nhập riêng.



Hình 9.24 Kiểm soát theo khả năng

Giải pháp thứ hai là phân tán theo hàng, hình 9.24 thể hiện nguyên lý hoạt động của nó, mỗi chủ thẻ sẽ được cấp những khả năng được phép thao tác với mỗi đối tượng tương đương với mỗi mục trong ma trận kiểm soát truy nhập. Khả năng được phép thao tác có thể coi như một tấm thẻ trên đó ghi những quyền được phép thực hiện, như vậy cần thiết phải có cơ chế để chủ thẻ sở hữu không được phép thay đổi các quyền trên tấm thẻ này. Hai giải pháp trên khác nhau ở cơ chế vận hành kiểm soát, danh sách kiểm soát truy nhập cần phải biết chủ thẻ trong khi đó giải pháp sử dụng khả năng chỉ quan tâm tới thao tác yêu cầu thực hiện có xuất hiện trong danh sách khả năng hay không.

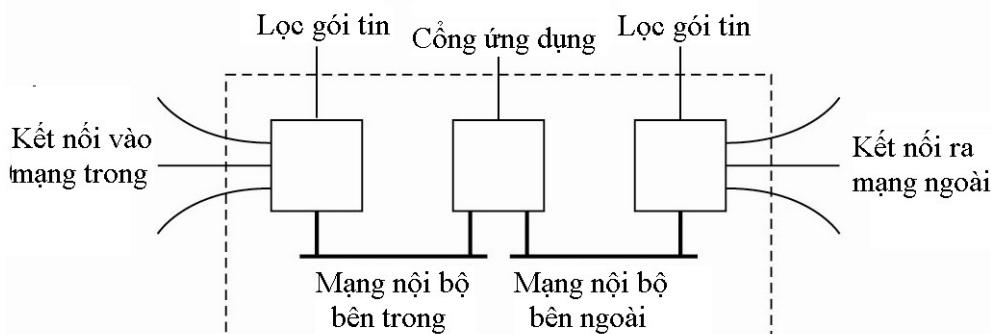
### 9.3.1.2 Miền bảo vệ

Danh sách kiểm soát truy nhập tuy đã khắc phục được nhược điểm của ma trận kiểm soát truy nhập bằng cách loại bỏ những mục rỗng, tuy nhiên kích thước của chúng vẫn còn rất lớn, để giảm kích thước của danh sách kiểm soát truy nhập thì cần phải đưa ra khái niệm miền bảo vệ. Miền bảo vệ là một tập các cặp đối tượng và quyền truy nhập, mỗi cặp xác định chính xác một đối tượng và các thao tác hợp lệ trên nó, các yêu cầu thực hiện thao tác đều thuộc một miền bảo vệ. Khi một chủ thẻ yêu cầu thao tác trên đối tượng, tiến trình giám sát sẽ tìm trong miền bảo vệ liên quan với yêu cầu này để biết có

được phép thực hiện hay không. Để đạt hiệu quả cao hơn, người ta dùng kết hợp miễn bảo vệ với việc phân nhóm các đối tượng, việc phân nhóm các đối tượng có thể hiểu tương tự như cách quản lý vai trò của người sử dụng.

### 9.3.2 Tường lửa

Các giải pháp mã hóa kết hợp và cài đặt kiểm soát truy nhập sẽ hoạt động tốt khi các bên tham gia đều tuân thủ những qui tắc xác định, điều này chỉ đúng khi hệ thống phân tán cách ly với thế giới bên ngoài. Vấn đề trở nên phức tạp hơn khi hệ thống cung cấp các dịch vụ công cộng, để bảo vệ hệ thống chắc chắn sẽ cần thêm những giải pháp khác. Trong thực tế, truy nhập từ bên ngoài đến hệ thống phân tán phải được kiểm soát bằng một công cụ đặc biệt gọi là tường lửa, hình 9.25 minh họa cách cài đặt tường lửa, nó dùng để ngăn chặn các luồng không được phép, chúng có thể lọc thông tin từ mạng đến tầng ứng dụng, về nguyên tắc nó phải có khả năng bảo vệ chống lại các hiểm họa bảo mật và không bao giờ bị lỗi.



Hình 9.25 Cài đặt tường lửa

Về cơ bản, tường lửa gồm hai loại, tường lửa lọc gói tin và tường lửa lọc nội dung. Tường lửa lọc gói tin hoạt động như một thiết bị định tuyến cho phép hoặc không cho phép gói tin chuyển qua mạng dựa trên địa chỉ nguồn và địa chỉ đích của gói tin, thường dùng để ngăn chặn các gói tin từ ngoài đi vào trong mạng. Trường hợp hệ thống phân tán cài đặt trên nhiều mạng cục bộ thì có thể sử dụng mạng riêng ảo để kết nối các mạng này nhưng vẫn duy trì được tường lửa. Mạng riêng ảo là một kênh truyền bảo mật trên môi trường mạng công cộng nhằm tiết kiệm về chi phí so với việc sử dụng các kênh thuê riêng, nó sử dụng các giao thức đường hầm cho phép đảm bảo tính toàn vẹn, bí mật thông tin, xác thực và mã hóa dữ liệu trên kênh truyền. Tường lửa mức ứng dụng không những kiểm tra thông tin điều khiển của gói tin mà còn kiểm tra nội dung của gói tin đó.

### 9.3.3 Bảo mật mã di động

Di trú mã là một trong những kỹ thuật nhằm nâng cao hiệu năng xử lý của hệ thống phân tán, tuy nhiên đây cũng là một trong những hiểm họa bảo mật. Các đoạn mã của tác tử phải được bảo mật để chống lại những sửa đổi và đồng thời các máy tính cũng phải có khả năng chống lại những tác tử độc hại.

### 9.3.3.1 Bảo vệ tác từ

Tác tử di động lang thang trên hệ thống phân tán để tìm kiếm thông tin, tất nhiên phải có cơ chế bảo mật để sao cho khi đến một máy tính nào đó bản thân tác tử sẽ không

bị đánh cắp hoặc sửa đổi thông tin. Rất tiếc cho đến nay chưa có một giải pháp đầy đủ nào để có thể thực hiện yêu cầu này, vì vậy cần tổ chức sao cho ít nhất cũng có thể phát hiện được những sửa đổi các tác tử.

Chủ sở hữu có thể phát hiện tác tử đã bị giả mạo dựa trên ba cơ chế: trạng thái chỉ đọc, chỉ thêm nhặt ký và tiết lộ trạng thái cho một số máy chủ nhất định. Trạng thái chỉ đọc của tác tử gồm tập các mục dữ liệu đã được chủ sở hữu ký trước khi gửi đến máy tính khác, khi tác tử trở về chỉ cần kiểm tra chữ ký có trùng khớp với nguyên gốc hay không.

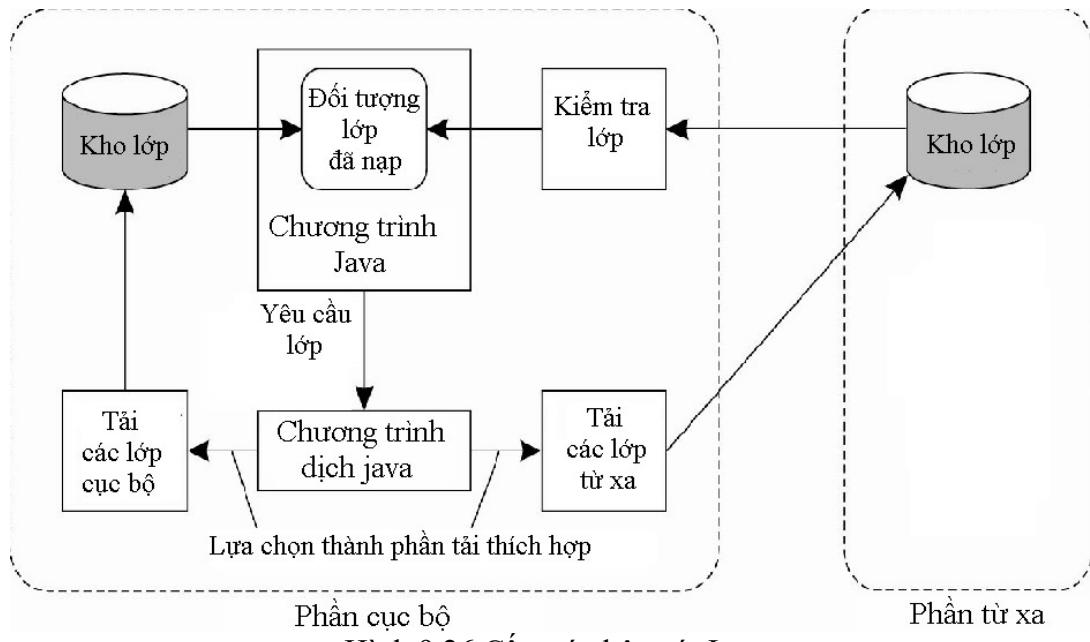
Cơ chế chỉ thêm nhặt ký nghĩa là chỉ được phép thêm mà không được phép sửa hoặc xóa những bản ghi trước, ban đầu nhặt ký rỗng và chỉ chứa giá trị kiểm tra  $C_{init}=E_{K^+_{Owner}}(N)$ , trong đó  $K^+_{Owner}$  là khóa công khai của chủ sở hữu tác tử và  $N$  là giá trị chứa định danh của chủ sở hữu. Khi tác tử chuyển đến máy S, nó thêm dữ liệu X,S vào nhặt ký kèm theo chữ ký  $sig(S,X)$  và tính giá trị kiểm tra mới  $C_{new}=E_{K^+_{Owner}}(C_{old}, sig(S,X),S)$  trong đó  $C_{old}$  là giá trị kiểm tra cũ. Lúc tác tử trở về, chủ sở hữu dễ dàng phát hiện nhặt ký có bị giả mạo hay không bằng cách đọc từ cuối nhặt ký, giải mã  $D_{K^-_{Owner}}(C)$ , mỗi bước sẽ trả về giá trị kiểm tra của bước kế tiếp và chữ ký  $sig(S,X)$  cho máy S, nếu không bị giả mạo thì xử lý và thực hiện bước tiếp theo. Vòng lặp sẽ dừng lại khi tiến đến giá trị kiểm tra ban đầu hoặc khi phát hiện mục dữ liệu bị giả mạo.

Cơ chế thứ ba được thực hiện bằng cách cung cấp một mảng mục dữ liệu, mỗi mục là một máy chủ xác định và được mã hóa bằng khóa công khai của máy chủ đó, toàn bộ mảng được chủ sở hữu Agent ký để đảm bảo tính toàn vẹn. Nếu một mục dữ liệu bị thay đổi giả mạo thì các máy chủ sẽ thông báo và có thể thực hiện các biện pháp thích hợp.

#### 9.3.3.2 Bảo vệ đích

Bảo vệ mã di động chống lại những máy phá hoại là quan trọng nhưng bảo vệ các máy tính trước những mã di động phá hoại còn quan trọng hơn. Nếu coi việc gửi tác tử ra thế giới bên ngoài là nguy hiểm thì nhìn chung người sử dụng có nhiều lựa chọn để lấy kết quả từ những tác tử này, tuy nhiên để cho phép tác tử thâm nhập vào hệ thống thì chỉ có duy nhất một cách là kiểm tra toàn bộ tác tử, người sử dụng cần phải kiểm soát toàn bộ những gì tác tử có thể thực hiện.

Không có cách nào bảo vệ được tác tử trước những thay đổi mang tính chất phá hoại nhưng lại có thể phát hiện sửa đổi trên nó, trong tình huống xấu nhất thì từ chối tiếp nhận khi nó trở về. Có thể sẽ quá muộn nếu như phát hiện mã độc hại sau đã khi xâm nhập vào hệ thống, vì vậy cần phải bảo vệ tất cả các tài nguyên chống lại các đoạn mã tải xuống truy nhập trái phép. Một trong những cách bảo vệ là xây dựng hộp cát, cấu trúc của nó trên hình 9.26, đó là kỹ thuật kiểm soát toàn bộ tất cả các chỉ thị lệnh, nếu phát hiện thấy lệnh nào bị cấm thì dừng ngay chương trình. Tương tự như vậy, việc thực thi đoạn mã tải về sẽ bị tạm dừng nếu phát hiện thâm nhập vào thanh ghi hoặc những vùng nhớ máy tính không cho phép.



Hình 9.26 Cấu trúc hộp cát Java

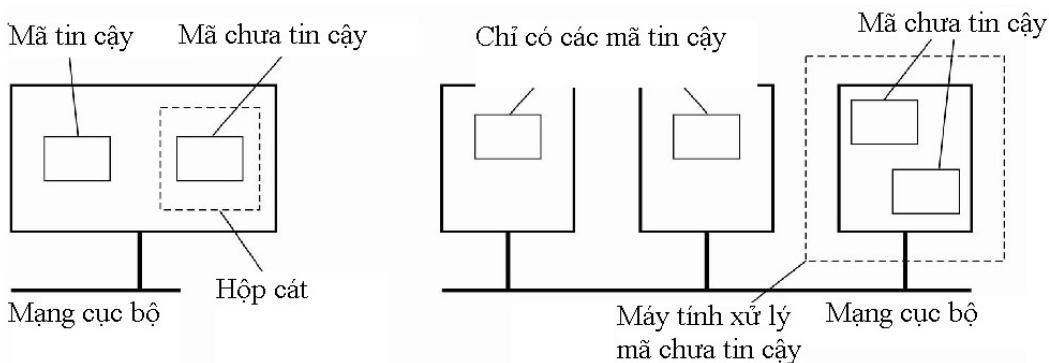
Cài đặt hộp cát khá phức tạp, có thể kiểm tra mã thực hiện khi tải xuống và chèn thêm các chỉ thị lệnh cho những trường hợp chỉ có thể kiểm tra khi chạy, điều này khá đơn giản đối với những đoạn mã thông dịch. Mỗi chương trình viết bằng Java bao gồm các lớp và từ đó tạo ra các đối tượng, không có các biến toàn cục và các hàm, tất cả đều nằm được khai báo trong các lớp, việc thực thi chương trình bắt đầu từ phương thức gọi hàm chính. Chương trình viết bằng Java được biên dịch thành các chỉ thị lệnh để từ đó sẽ được thông dịch bằng máy ảo Java, vì vậy máy ảo Java sẽ lần lượt xử lý từng chỉ thị lệnh của chương trình đã tải về bằng cách biên dịch thành các chỉ thị lệnh của nó, bắt đầu từ chương trình chính.

Trong hộp cát Java, việc bảo vệ bắt đầu bằng cách đảm bảo thành phần xử lý chuyển chương trình đến máy khách có thể tin cậy, tải xuống trong Java do tiến trình nạp lớp đảm nhiệm. Mỗi thành phần nạp lớp có trách nhiệm lấy lớp xác định từ máy chủ và cài đặt trên không gian địa chỉ của máy khách, như vậy máy ảo Java có thể tạo các đối tượng từ những lớp đó. Vì thành phần nạp lớp chỉ là một lớp Java khác, chương trình đã tải về có thể chứa thành phần nạp lớp của riêng nó, hộp cát đầu tiên sẽ xử lý những thành phần nạp lớp đáng tin cậy sẽ được sử dụng, chương trình Java không cho phép tạo các thành phần nạp lớp riêng nếu chúng phá vỡ phương thức nạp lớp thông thường vẫn xử lý.

Thành phần thứ hai của hộp cát là trình kiểm tra mã byte, nó kiểm tra xem lớp đã tải về có tuân thủ các qui tắc bảo mật của hộp cát hay không, đặc biệt nó kiểm tra xem lớp đó có chứa các chỉ thị lệnh bất hợp pháp hoặc những lệnh làm hỏng ngăn xếp hoặc bộ nhớ hay không. Không phải kiểm tra tất cả các lớp mà chỉ tập trung vào những lớp được tải từ máy chủ bên ngoài về máy khách, nói chung những lớp nằm trên máy khách đều đáng tin cậy. Khi tải xuống một lớp và đã kiểm tra an toàn, máy ảo Java có thể khởi tạo và thực hiện các phương thức của đối tượng, để ngăn chặn các đối tượng truy nhập trái

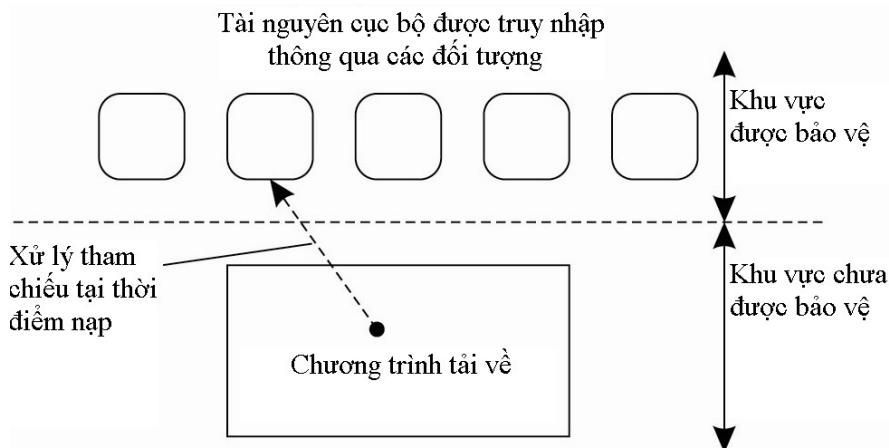
phép vào tài nguyên của máy khách, trình quản lý bảo mật thực hiện những kiểm tra khác nhau trong khi chạy.

Các chương trình tải về buộc phải sử dụng trình quản lý bảo mật để không có cách nào gây tổn hại, nghĩa là mọi thao tác đều phải tuân thủ yêu cầu của trình bảo mật. Trình bảo mật sẽ cấm nhiều thao tác như từ chối truy nhập vào các tập tin cục bộ và chỉ cho phép thiết lập kết nối đến các máy chủ nguồn gốc của các chương trình đã tải về, tuy nhiên chúng có thể tuy nhập đến thư viện đồ họa để phục vụ cho mục đích hiển thị hoặc bắt các sự kiện như di chuyển chuột hoặc bấm chuột vào các nút điều khiển. Trình bảo mật Java ban đầu được cài đặt rất khắt khe, chúng không phân biệt các chương trình đã tải từ những máy chủ khác nhau, trong nhiều trường hợp chúng đã giới hạn quá mức và cần thiết phải linh hoạt hơn.



Hình 9.27 Hộp cát và sân chơi

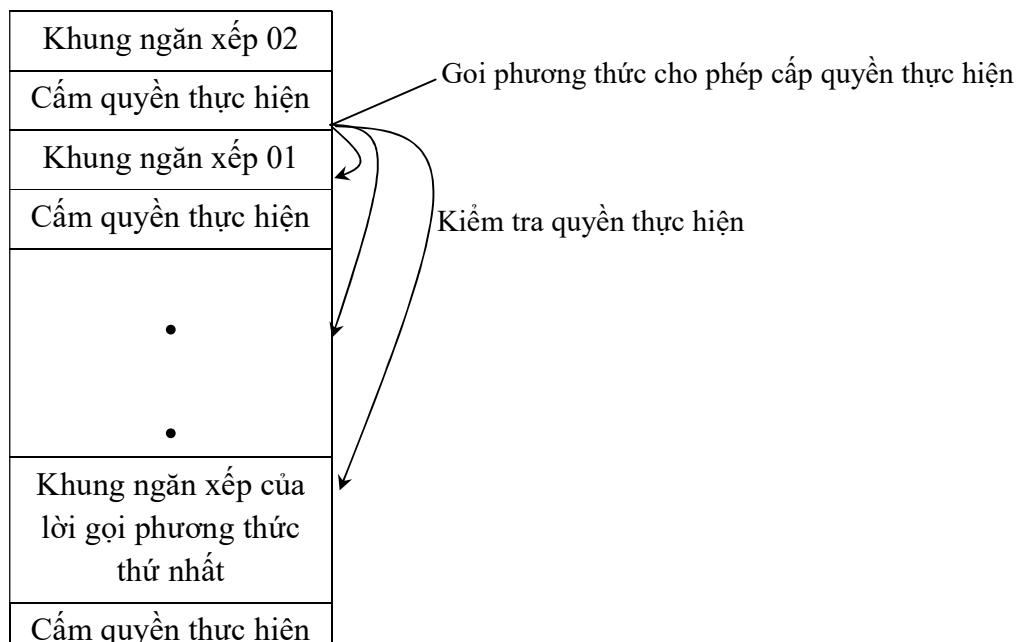
Một cách tiếp cận khác tương tự như hộp cát nhưng linh hoạt hơn là tạo ra một sân chơi cho mã di động tải về, hình 9.27 thể hiện sân chơi là một máy riêng biệt chỉ dành cho mã di động đang chạy. Tài nguyên cục bộ như các tập tin và các kết nối đến máy chủ bên ngoài đều có sẵn cho các chương trình chạy tại sân chơi với các cơ chế bảo mật thông thường. Tuy nhiên các tài nguyên cục bộ đến các máy khác bị ngắt vĩnh viễn và mã tải về không thể truy nhập được, người sử dụng trên các máy khác có thể truy nhập đến sân chơi như cách thông thường, ví dụ gọi thủ tục từ xa, nhưng không một mã di động nào được tải về các máy cục bộ.



Hình 9.28 Nguyên lý sử dụng tham chiếu đối tượng

Hướng tới mục tiêu tăng độ linh hoạt, mỗi chương trình tài về phải được xác thực và sau đó thực thi chính sách bảo mật riêng dựa trên nguồn gốc của chương trình. Mục tiêu thứ nhất có thể thực hiện bằng cách sử dụng chữ ký cho mã di động như những văn bản thông thường, có thể áp dụng như một phương án dự phòng cho hộp cát, như vậy chỉ những mã máy chủ tin cậy mới được tiếp nhận. Hình 9.28 minh họa cách giải quyết vấn đề thứ hai, có thể dựa trên việc sử dụng tham chiếu đối tượng như những khả năng được phép thực hiện, để truy nhập vào tài nguyên cục bộ thì phải tham chiếu đến đối tượng riêng xử lý các thao tác khi tải chương trình, nếu không có tham chiếu nào thì không được phép thực hiện.

Ban đầu, chương trình không nhìn thấy tất cả các giao diện của đối tượng thao tác với tập tin hệ thống bằng cách không chuyển tham chiếu cho những giao diện này, như vậy đảm bảo không thể xây dựng bất kỳ tham chiếu nào tới một trong những giao diện đó trong thời gian chạy. Hơn nữa có thể sử dụng thuộc tính của Java để giữ cho biên và các phương thức hoàn toàn thuộc về bên trong của lớp, chương trình có thể được bảo vệ chống lại việc khởi tạo một đối tượng riêng xử lý tập tin bằng cách che giấu thao tác tạo các đối tượng mới từ một lớp đã cho.



Hình 9.29 Nguyên lý kiểm tra bên trong ngăn xếp

Cơ chế thứ hai thực thi chính sách bảo mật là kiểm tra bên trong ngăn xếp được minh họa trong hình 9.29, về bản chất nghĩa là trước khi gọi bất kỳ phương thức truy nhập tài nguyên cục bộ thì phải gọi một thủ tục kiểm tra xem thủ tục đó có được phép gọi thủ tục thao tác với tài nguyên hay không. Nếu được ủy quyền thì mở quyền tạm thời trong thời gian gọi thủ tục đó, trước khi trả quyền điều khiển cho phương thức gọi sẽ gọi thủ tục để đóng quyền truy nhập. Để gọi thủ tục mở/đóng quyền truy nhập, có thể yêu cầu lập trình viên giao diện đến tài nguyên cục bộ chèn những lời gọi này vào những vị trí thích hợp, tuy nhiên sẽ tốt hơn nếu để trình thông dịch của Java tự động xử lý những lời gọi này. Khi gọi đến tài nguyên cục bộ, trình thông dịch Java tự động mở quyền thực

hiện và kiểm tra xem lời gọi có được phép thực hiện hay không, nếu được thì đẩy lời gọi đóng quyền thực hiện vào ngăn xếp để đảm bảo chắc chắn quyền thực hiện sẽ bị cấm khi phương thức gọi trở về.

Phương pháp này cho phép kiểm tra tốt hơn quyền thực thi, giả sử chương trình gọi đối tượng cục bộ 01 và đối tượng đó lại gọi đối tượng 02, mặc dù đối tượng 01 có thể được cấp quyền gọi đối tượng 02, nếu chủ thể gọi đối tượng 01 không được tin cậy để gọi phương thức thuộc về đối tượng 02 thì kiểu gọi dây chuyền đó cũng không được phép. Kiểm tra bên trong ngăn xếp dễ dàng kiểm tra những chuỗi như vậy, trình thông dịch chỉ cần kiểm tra mỗi khung ngăn xếp bắt đầu từ đỉnh để biết khung nào đã mở quyền thực thi hoặc khung nào cấm truy nhập đến tài nguyên hiện hành. Về cơ bản có thể nói kiểm tra bên trong ngăn xếp cho phép đính kèm các quyền vào lớp hoặc phương thức và kiểm tra những quyền này cho mỗi lần gọi riêng biệt, bằng cách này có thể thực hiện miền bảo vệ dựa trên lớp.

Cách tiếp cận thứ ba để thực thi chính sách bảo mật là quản lý không gian tên, để truy nhập đến tài nguyên cục bộ thì chương trình phải tích hợp những tập tin thích hợp chứa các lớp thực hiện những tài nguyên đó. Việc tích hợp đòi hỏi phải cung cấp tên cho trình thông dịch, sau đó phân giải thành lớp và sau đó được nạp trong thời gian chạy. Để thực thi chính sách bảo mật cho một chương trình tải xuống, cùng một tên có thể phân giải thành các lớp khác nhau tùy thuộc vào việc chương trình được tải từ đâu. Thông thường việc phân giải tên do thành phần nạp lớp xử lý, nó cần phải được điều chỉnh để thực hiện cách tiếp cận này.

Những cách tiếp cận trên liên kết quyền thực hiện với các lớp và các phương thức dựa trên địa điểm tải chương trình xuống, với ưu điểm của trình thông dịch Java có thể thực thi chính sách bảo mật thông qua các cơ chế đã trình bày trên đây. Theo nghĩa này, kiến trúc bảo mật phụ thuộc vào ngôn ngữ bậc cao và cần phải phát triển ngôn ngữ mới. Các giải pháp độc lập ngôn ngữ đòi hỏi cách tiếp cận tổng quát hơn để thực thi bảo mật và cũng khó cài đặt hơn, trong những trường hợp này cần có sự trợ giúp bảo mật của bệ điều hành nhận biết mã di động tải xuống và thực thi tất cả những lời gọi đến tài nguyên cục bộ để chạy qua phần lõi, nơi sẽ tiếp tục thực hiện kiểm tra.

#### **9.3.4 Từ chối dịch vụ**

Nói chung, kiểm soát truy nhập đảm bảo cẩn thận chỉ những tiến trình ủy quyền mới được truy nhập tài nguyên, một loại tấn công quấy nhiễu liên quan tới kiểm soát truy nhập bằng cách ròc tâm ngăn cản những tiến trình đã được ủy quyền truy nhập tài nguyên. Việc phòng thủ chống lại những cuộc tấn công từ chối dịch vụ DoS ngày càng trở nên quan trọng khi hệ thống phân tán đã mở trên toàn mạng Internet, nếu những cuộc tấn công này chỉ xuất phát từ vài điểm thì có thể dễ dàng xử lý, vẫn đề sẽ phức tạp hơn nhiều khi phải chống lại những cuộc tấn công từ chối dịch vụ phân tán DDoS.

Tấn công DDoS tập hợp một lượng rất lớn tiến trình để cố gắng triệt hạ dịch vụ mạng, kẻ tấn công đã thành công trong việc cướp một lượng lớn máy tính vô tình tham gia tấn công, chúng có thể chiếm hết băng thông hoặc tê liệt tài nguyên. Việc chiếm hết băng thông được thực hiện đơn giản bằng cách gửi rất nhiều thông điệp đến một máy làm

cho những thông điệp bình thường khó có thể tiếp cận đến máy đó, tấn công tê liệt tài nguyên làm cho bên cung cấp dịch vụ không còn khả năng vận hành bình thường, ví dụ làm cho máy tính ngập lụt trong việc xử lý các yêu cầu thiết lập liên kết của giao thức TCP.

Không có biện pháp đơn lẻ nào chống lại tấn công DDoS, kể tấn công cù dụng những nạn nhân vô tội bằng cách bí mật cài đặt phần mềm lên máy tính của họ. Trong những trường hợp này giải pháp duy nhất là phải có máy liên tục giám sát trạng thái bằng cách kiểm tra các tập tin bị nhiễm, xem xét bằng cách nào virus có thể phát tán trên mạng Internet. Chỉ dựa vào biện pháp đối phó này thì chưa đủ, cần thiết phải liên tục giám sát lưu lượng mạng, ví dụ bắt đầu từ những thiết bị định tuyến cửa ngõ của mạng. Kinh nghiệm cho thấy loại bỏ những gói tin có nguồn gốc không thuộc mạng của cơ quan có thể ngăn chặn rất nhiều những cuộc tàn phá, nói chung lọc được nhiều gói tin càng gần nguồn càng tốt. Hai trong số các nguyên tắc cơ bản về danh sách kiểm soát truy nhập, danh sách kiểm soát truy nhập chuẩn thì đặt càng gần đích càng tốt trong khi danh sách kiểm soát truy nhập mở rộng thì đặt càng gần nguồn càng tốt.

Có thể tập trung vào những thiết bị định tuyến cửa ngõ vào mạng của cơ quan, tuy nhiên điều này có thể đã quá muộn vì có thể sẽ chặn cả những người sử dụng thông thường. Tốt hơn hết là phải xử lý từ những thiết bị định tuyến của nhà cung cấp dịch vụ Internet bằng cách lọc những gói tin nghi là tấn công, thiết bị định tuyến sẽ loại bỏ những gói tin nếu phát hiện thấy tỉ lệ không cân đối giữa lưu lượng nhận và gửi từ một nút mạng nào đó. Nói chung, cần thiết phải triển khai rất nhiều biện pháp để chống lại tấn công từ chối dịch vụ, hơn nữa những hình thức tấn công mới càng ngày càng đa dạng và hiểm độc hơn.

## 9.4 Quản lý bảo mật

Những phần trên đã xem xét các kênh bảo mật và kiểm soát truy nhập nhưng hầu như chưa đề cập đến vấn đề lấy các khóa như thế nào, phần này sẽ trình bày kỹ hơn về quản lý bảo mật. Thứ nhất cần xem xét quản lý chung các khóa mã hóa, cách phân phát các khóa này, từ đó thấy được tầm quan trọng của các chứng chỉ. Thứ hai là thảo luận quản lý bảo mật nhóm các máy chủ bằng cách tập trung vào vấn đề thêm thành viên mới đã được thành viên hiện hành tin cậy, đối mặt với dịch vụ phân tán và nhân bản thì điều quan trọng là vấn đề bảo mật sẽ không bị ảnh hưởng khi tiếp nhận tiến trình độc hại vào nhóm. Vấn đề thứ ba chú trọng vào việc quản lý ủy quyền bằng cách xem xét những khả năng và những thứ được gọi là chứng chỉ thuộc tính, vấn đề quan trọng trong các hệ thống phân tán liên quan đến quản lý ủy quyền là một tiến trình có thể ủy thác quyền truy nhập của nó cho một tiến trình khác.

### 9.4.1 Quản lý khóa

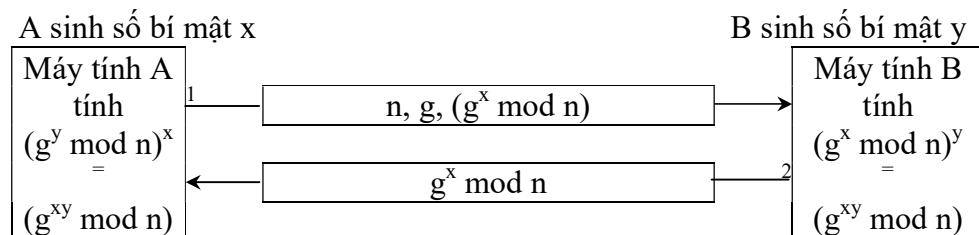
Khi giới thiệu các giao thức mã hóa chúng ta đã ngầm hiểu các khóa đã sẵn sàng, ví dụ trường hợp các hệ thống mã khóa công khai, bên gửi coi như đã biết khóa công khai của bên nhận để tùy ý sử dụng và như vậy mới có thể mã hóa thông điệp nhằm đảm bảo tính bí mật. Tương tự như vậy, trường hợp xác thực sử dụng trung tâm phân phối khóa KDC, mỗi bên tham gia cũng đã chia sẻ khóa bí mật với KDC. Tuy nhiên, việc thiết

lập và phân phát khóa không phải vấn đề tầm thường, trong nhiều trường hợp phân phát khóa qua kênh không bảo mật có thể nảy sinh vấn đề và vì vậy cần thiết phải sử dụng các biện pháp kênh ngoài. Tương tự như vậy cũng cần phải có biện pháp thu hồi khóa để ngăn chặn việc sử dụng khóa sau khi đã bị lộ hoặc không còn hiệu lực.

#### 9.4.1.1 Thiết lập khóa

Khi thiết lập kênh bảo mật, bên gửi sẽ sử dụng khóa công khai của bên nhận để khởi tạo kênh truyền, nếu chấp thuận thì bên nhận sinh khóa phiên sau đó mã hóa bằng khóa công khai của bên gửi và để trả về cho bên gửi, bằng cách mã hóa khóa phiên trước khi gửi lên đường truyền thì có thể yêu tâm khóa đó sẽ được chuyển an toàn qua mạng. Dựa vào cách thức trên có thể sinh và phân phát khóa phiên khi hai bên đã chia sẻ khóa bí mật, tuy nhiên các bên tham gia đều phải có sẵn phương tiện để thiết lập kênh truyền bí mật, nghĩa là phải có một hình thức thiết lập và phân phối khóa đã được thực hiện, tương tự như vậy khi thiết lập với bên tin cậy thứ ba, ví dụ như trung tâm phân phối khóa KDC.

Hình 9.30 thể hiện nguyên lý trao đổi khóa Diffie-Hellman, giao thức này đã được ứng dụng rộng rãi để thiết lập khóa chia sẻ qua kênh không an toàn, giả sử A và B muốn thiết lập khóa bí mật chia sẻ, cả A lẫn B đều tạo ra hai số lớn  $n$  và  $g$ , hai số này có thể được công khai. Bên A sẽ tạo ra một số lớn khác là  $x$ , bên B tạo ra số lớn  $y$  và giữ bí mật chúng, bên A sẽ gửi cho bên B các số  $n$ ,  $g$  và  $(g^x \bmod n)$ , đây đều là những bản rõ ràng như không thể tính được  $x$  khi chỉ biết giá trị  $(g^x \bmod n)$ .



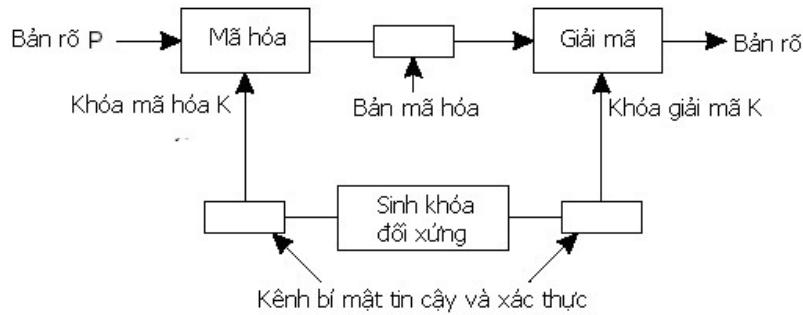
Hình 9.30 Nguyên lý trao đổi khóa Diffie – Hellman

Bên B thực hiện tính  $(g^x \bmod n)^y$  và về mặt toán học sẽ bằng  $(g^{xy} \bmod n)$ , đồng thời bên B cũng gửi cho bên A giá trị  $(g^y \bmod n)$ . Bên A thực hiện tính toán  $(g^y \bmod n)^x$  về mặt toán học cũng bằng  $(g^{xy} \bmod n)$ , như vậy chỉ duy nhất A và B mới biết khóa chia sẻ bí mật  $(g^{xy} \bmod n)$ , bản thân A và B cũng không cần biết giá trị  $x$  và  $y$  của nhau. Giao thức Diffie-Hellman có thể xem như một hệ thống mã hóa khóa công khai trong đó khóa riêng của A là  $x$  và khóa công khai của A là  $(g^x \bmod n)$ .

#### 9.4.1.2 Phân phát khóa

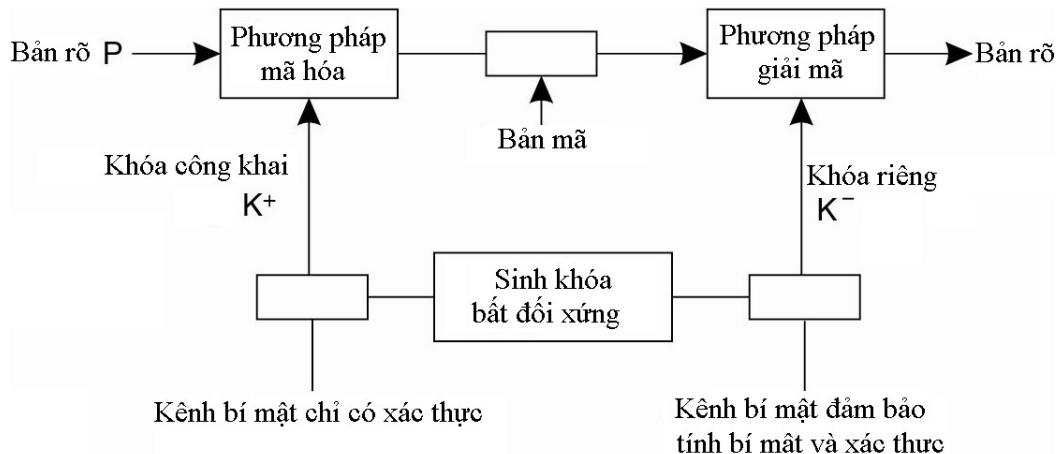
Một trong những khâu phức tạp trong quản lý khóa là phân phát thực tế các khóa ban đầu, hình 9.31 thể hiện cách phân phối khóa bí mật trong hệ thống mã hóa đối xứng. Khóa bí mật chia sẻ ban đầu phải được trao đổi theo kênh bảo mật đảm bảo xác thực cũng như tính bí mật, nếu không sẵn có khóa nào cho các bên để thiết lập kênh bảo mật

nhiều vậy thì cần phải phân phát khóa bằng ngoài, ví dụ gọi điện thoại hoặc gửi qua bưu điện.



Hình 9.31 Phân phát khóa bí mật

Trường hợp hệ thống mã hóa khóa công khai thì cần phân phát khóa công khai sao cho bên nhận có thể chắc chắn khóa đó là một cặp với khóa riêng đã yêu cầu, mặc dù không cần phải mã hóa khóa công khai nhưng nó phải được chuyển qua kênh xác thực. Trong thực tế, việc phân phát khóa công khai thực hiện bằng các phương tiện của chứng chỉ khóa công khai, chúng gồm khóa công khai kèm theo xâu ký tự xác định thực thể liên quan tới khóa đó, thực thể có thể là người sử dụng hoặc thiết bị mạng. Cả khóa công khai lẫn định danh đều được ký bởi cơ quan chứng nhận và chữ ký đó là đúng, định danh của cơ quan xác nhận đương nhiên là một phần của chứng chỉ. Việc ký thực hiện bằng khóa riêng  $K_{CA}^-$  của cơ quan chứng thực, khóa công khai  $K_{CA}^+$  coi như đã biết, ví dụ có thể đăng tải trên các trang web.



Hình 9.32 Phân phát khóa công khai

Hình 9.32 thể hiện phân phát khóa công khai, giả sử khách hàng muốn xác minh khóa công khai trong chứng chỉ có thực sự thuộc về một thực thể xác định hay không, họ sẽ sử dụng khóa công khai của cơ quan xác thực có liên quan để kiểm tra chữ ký của chứng chỉ, nếu chữ ký trên chứng chỉ trùng với cặp (khóa công khai, định danh thực thể) thì có thể chấp nhận khóa công khai thực sự thuộc về thực thể xác định và có thể tin tưởng chứng chỉ không bị giả mạo. Như vậy, khách hàng đã coi khóa công khai  $K_{CA}^+$  là

của cơ quan chứng thực có liên quan, nếu nghi ngờ thì có thể kiểm tra tính hợp lệ của  $K_{CA}^+$  thông qua chứng chỉ khác từ một cơ quan khác, ví dụ từ cơ quan chứng thực đáng tin cậy hơn.

Mô hình tin cậy phân cấp, trong đó cơ quan chứng thực cấp cao nhất được mọi người tin tưởng khá phổ biến, ví dụ thư được tăng cường tính riêng tư PEM sử dụng mô hình tin cậy ba mức, mức thấp nhất do cơ quan chứng nhận chính sách PCA xác thực, cơ quan này lại được tổ chức đăng ký chính sách Internet IPRA xác thực. Nếu người sử dụng không tin tưởng IPRA thì cũng không hy vọng người đó sẽ tin tưởng những thông điệp thư điện tử gửi qua kênh tăng cường tính riêng tư.

#### 9.4.1.3 Thời gian sống của chứng chỉ

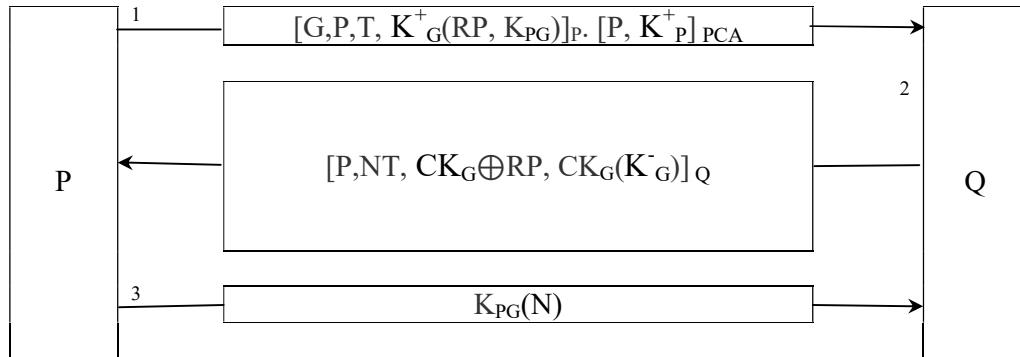
Một trong những vấn đề quan trọng là tuổi thọ của chứng chỉ, khóa công khai của chứng chỉ sẽ luôn hợp lệ cho thực thể xác định bởi chứng chỉ đó, nếu khóa riêng của chứng chỉ bị lộ thì không nên sử dụng khóa công khai đó nữa, như vậy cần phải có cơ chế công khai tuyên bố chứng chỉ không còn hiệu lực. Giải pháp phổ biến là công bố danh sách thu hồi chứng chỉ CRL, khách hàng phải kiểm tra xem chứng chỉ đã bị thu hồi hay chưa. Nếu danh sách này công bố hàng ngày thì chứng chỉ bị lộ có thể được sử dụng cho đến thời điểm công bố kế tiếp, như vậy thời gian giữa những lần công bố không được phép quá dài. Giải pháp thứ hai là giới hạn thời gian cho mỗi chứng chỉ, quá thời hạn thì chứng chỉ không còn hiệu lực, trong thời gian còn hiệu lực thì vẫn có thể đưa chứng chỉ vào danh sách thu hồi. Giải pháp thứ ba là giảm thời gian tồn tại có hiệu lực của một chứng chỉ xuống gần bằng 0, khi đó máy khách luôn luôn phải kiểm tra chứng chỉ để xác định thời gian có hiệu lực của khóa công khai.

#### 9.4.2 Quản lý bảo mật nhóm

Nhiều hệ thống bảo mật sử dụng các dịch vụ như trung tâm phân phối khóa KDC hoặc chứng chỉ CA, điều đó chứng tỏ vấn đề phức tạp trong các hệ thống phân tán. Trước hết những đơn vị cung cấp dịch vụ đó phải đáng tin cậy, muôn cải thiện tính tin cậy trong các dịch vụ bảo mật thì phải cung cấp mức độ cao chống lại tất cả các hiểm họa bảo mật. Ví dụ, khi chứng chỉ CA bị lộ thì không thể kiểm tra tính hợp lệ của khóa công khai làm cho toàn bộ hệ thống bảo mật hoàn toàn không còn giá trị. Mặt khác, các dịch vụ bảo mật phải có tính sẵn sàng cao, như vậy cần thiết phải nhân bản, nhưng nhân bản lại làm cho hệ thống dễ bị tấn công hơn. Vấn đề cần giải quyết là đảm bảo làm sao khi một tiến trình yêu cầu tham gia nhóm G thì tính toàn vẹn của nhóm không bị đe dọa. Giả sử các thành viên nhóm G sử dụng khóa bí mật  $CK_G$  để mã hóa các thông điệp trong nhóm, nhóm còn có thêm một cặp khóa công khai và khóa riêng ( $K_G^+, K_G^-$ ) để giao tiếp với các thành viên của nhóm khác.

Hình 9.33 thể hiện qui trình tiếp nhận thành viên, tiến trình P muốn tham gia vào nhóm G sẽ gửi yêu cầu tham gia JR, thời gian cục bộ T của tiến trình P, bảng phản hồi RP và khóa bí mật  $K_{P,G}$ , RP và  $K_{P,G}$  được mã hóa cùng nhau sử dụng khóa công khai  $K_G^+$  của nhóm, JR được ký bởi tiến trình P và được gửi đi cùng với chứng chỉ chứa khóa công khai của P, ký hiệu  $[M]_A$  là thông điệp M do chủ thể A ký. Khi thành viên Q của nhóm nhận được yêu cầu tham gia, trước hết nó xác thực P sau đó sẽ liên lạc với các

thành viên khác của nhóm để xem có nhận P làm thành viên của nhóm hay không. Việc xác thực P thực hiện theo cách thông thường bằng chứng chỉ, nhãn thời gian T dùng để xem chứng chỉ có còn hiệu lực tại thời điểm gửi yêu cầu hay không. Thành viên Q kiểm tra chữ ký của cơ quan chứng thực sau đó tách khóa công khai của P từ chứng chỉ để kiểm tra tính hợp lệ của JR, tại thời điểm này tiếp tục sử dụng giao thức riêng của nhóm để biết các thành viên khác có đồng ý kết nạp P hay không.



Hình 9.33 Qui trình tiếp nhận thành viên mới

Nếu tiến trình P được phép tham gia vào nhms thì Q sẽ trả về thông điệp tiếp nhận chứa định danh của P và thẻ của nhóm N, phần RP dùng để mã hóa khóa liên lạc trong nhóm  $CK_G$ , khóa riêng của nhóm  $K_G^-$  được mã hóa bằng khóa  $CK_G$ . Toàn bộ thông điệp sẽ được Q ký bằng khóa  $K_{P,G}$ , tiến trình P giờ đây có thể xác thực Q vì chỉ những thành viên của nhóm mới có thể nhận ra khóa  $K_{P,G}$ , thẻ N trong giao thức không sử dụng cho việc bảo mật mà P sẽ mã hóa bằng khóa  $K_{P,G}$  và gửi trở lại để thông báo cho Q biết P đã nhận được tất cả các khóa cần thiết và đã thực sự tham gia nhóm. Thay cho việc sử dụng RP, P và Q có thể sử dụng khóa công khai của P để mã hóa  $CK_G$ , tuy nhiên RP chỉ được sử dụng một lần để mã hóa khóa truyền thông của nhóm trong thông điệp GA, sử dụng RP sẽ an toàn hơn. Nếu khóa riêng của P bị lộ thì có thể bị lộ khóa  $CK_G$  và như vậy sẽ tồn tại đe dọa bám mật truyền thông của nhóm.

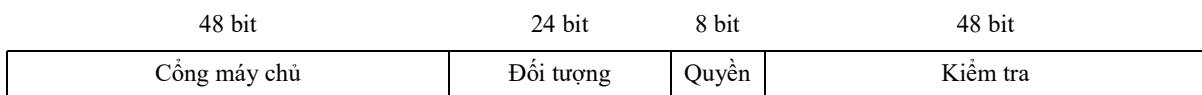
#### 9.4.3 Quản lý ủy quyền

Quản lý bảo mật quan trọng thiết với việc quản lý quyền truy nhập, vấn đề quyền truy nhập ban đầu được gán cho người sử dụng hoặc cho nhóm và cách duy trì chúng như thế nào vẫn chưa được đề cập tới. Nếu hệ thống quản lý tập trung thì việc gán quyền truy nhập khá đơn giản, mọi công việc liên quan tới tài khoản đều được người quản trị hệ thống xác định trước. Nếu hệ thống quản lý phân tán, vấn đề gán quyền truy nhập phức tạp hơn vì tài nguyên được đặt trên nhiều máy, áp dụng như hệ thống tập trung sẽ phải thiết lập tài khoản cho mỗi người sử dụng trên từng máy. Có thể giải quyết vấn đề đơn giản hơn bằng cách tạo tài khoản trên máy chủ trung tâm, mỗi lần người sử dụng truy nhập tài nguyên hệ thống sẽ tham khảo máy chủ này, về cơ bản phương pháp này giống như quản lý tài khoản trong hệ điều hành mạng.

#### 9.4.3.1 Chứng chỉ quyền truy nhập

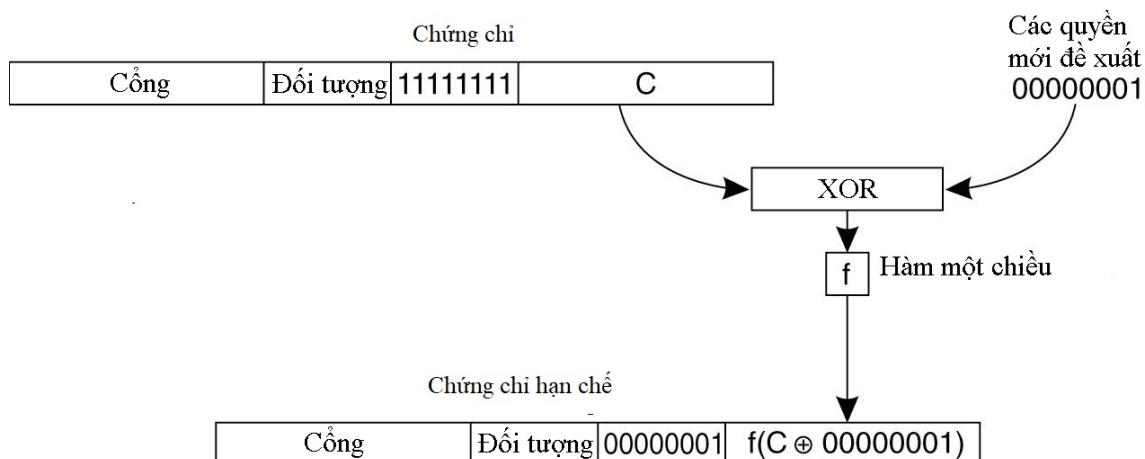
Một cách tiếp cận tốt hơn đã được áp dụng rộng rãi trong các hệ thống phân tán là sử dụng chứng chỉ năng lực, đó là một cấu trúc dữ liệu không thể giả mạo cho một tài nguyên riêng biệt, nó xác định chính xác quyền truy nhập của người nắm giữ khả năng đối với tài nguyên đó. Có thể cài đặt các khả năng theo nhiều cách khác nhau, ví dụ hệ thống coi các đối tượng phân tán như những đối tượng từ xa đặt trên máy chủ, máy khách truy nhập đến đối tượng thông qua tiến trình ủy quyền. Để thực hiện thao tác trên đối tượng, máy khách chuyển khả năng vào hệ điều hành cục bộ, hệ điều hành cục bộ sẽ xác định máy chủ chứa đối tượng cần truy nhập và sau đó thực hiện gọi thủ tục từ xa đến máy chủ đó.

Hình 9.34 mô tả cấu trúc định danh của chứng chỉ quyền truy nhập dài 128 bit, với cấu trúc bên trong gồm bốn trường thông tin, 48 bit đầu tiên gọi là cổng máy chủ do máy chủ của đối tượng khởi tạo khi sinh đối tượng và có tác dụng trong việc tạo ra một định danh máy tính độc lập cho máy chủ của đối tượng. 24 bit tiếp theo được sử dụng để xác định đối tượng trên máy chủ đã cho, như vậy kết hợp với phần cổng máy chủ sẽ tạo ra 72 bit xác định đối tượng duy nhất trong toàn bộ hệ thống. 8 bit kế tiếp xác định quyền truy nhập của chủ thẻ nắm giữ chứng chỉ, 48 bit cuối là trường kiểm tra và được dùng để bảo vệ làm cho chứng chỉ không thể bị sửa đổi.



Hình 9.34 Cấu trúc định danh của chứng chỉ

Khi tạo một đối tượng, máy chủ lấy một trường kiểm tra ngẫu nhiên và lưu trong quyền truy nhập và trong một bảng nội bộ riêng của nó, tất cả các bit của trường quyền truy nhập trong một chứng chỉ mới được thiết lập bằng 1 và đó là chứng chỉ trả về cho máy khách. Nếu máy khách yêu cầu thực hiện một thao thác thì phải gửi trả lại máy chủ chứng chỉ này, lúc đó trường kiểm tra sẽ dùng để xác minh xem chứng chỉ đã cấp có bị sửa đổi hay không.



Hình 9.35 Tạo chứng chỉ hạn chế từ chứng chỉ của chủ sở hữu

Để tạo quyền truy nhập hạn chế, máy khách có thể chuyển chứng chỉ ngược trở lại máy chủ kèm theo mặt nạ bit cho quyền mới, máy chủ lấy trường kiểm tra gốc từ bảng của nó và thực hiện phép tính XOR với các bit của quyền mới và sau đó chạy kết quả qua hàm một chiều. Máy chủ tạo quyền truy nhập mới với trường đối tượng giữ nguyên nhưng các bit quyền mới và trường kiểm tra là kết quả đầu ra của hàm một chiều, chứng chỉ mới này sẽ được trả về cho bên gọi và máy khách có thể gửi khả năng mới này cho tiến trình khác nếu muốn. Hình 9.35 minh họa phương pháp tạo các quyền truy nhập hạn chế, chủ sở hữu đã tắt tất cả các quyền ngoại trừ một quyền. Quyền truy nhập hạn chế có thể cho phép đọc đối tượng nhưng không cho phép bất kỳ quyền nào khác, ý nghĩa của trường quyền là khác nhau đối với từng loại đối tượng vì bản thân các hoạt động hợp pháp cũng khác nhau tùy theo từng loại đối tượng.

Khi quyền truy nhập hạn chế quay lại máy chủ, máy chủ nhìn thấy trường quyền thực hiện khác với quyền của chủ sở hữu vì ít nhất một bit đã bị tắt, nghĩa là giá trị của bit bằng 0, máy chủ lấy số ngẫu nhiên gốc từ bảng của nó và thực hiện phép tính XOR với các bit trong trường quyền thực hiện của khả năng và chạy kết quả qua hàm một chiều, nếu kết quả trả về trùng với trường kiểm tra chứng tỏ chứng chỉ đó là lợp lệ. Như vậy nếu người sử dụng thêm quyền chưa được cấp thì chứng chỉ sẽ không hợp lệ, đảo ngược trường kiểm tra trong khả năng giới hạn cũng không thể được vì đã sử dụng hàm một chiều. Về cơ bản hàm một chiều này cũng tương tự như tính toán chữ ký số, nếu thay đổi thông điệp ban đầu thì sẽ bị phát hiện ngay lập tức.

Một số hệ thống phân tán sử dụng chứng chỉ thuộc tính, về bản chất đó là bước tổng quát hóa các khả năng, không giống như chứng chỉ thông thường sử dụng khóa công khai để kiểm tra tính hợp lệ, chứng chỉ thuộc tính sử dụng danh sách cẩn (thuộc tính, giá trị) để áp dụng cho một thực thể xác định. Đặc biệt, chứng chỉ thuộc tính có thể dùng để liệt kê các quyền truy nhập của chủ thể nắm giữ chứng chỉ đến một tài nguyên xác định. Tương tự như chứng chỉ, các chứng chỉ thuộc tính cũng được chuyển từ những cơ quan chứng thực cụ thể, các quyền truy nhập liệt kê trong chứng chỉ sẽ do cơ quan chứng thực thuộc tính ký.

#### 9.4.3.2 Ủy nhiệm

Ủy nhiệm là một kỹ thuật quan trọng để bảo vệ máy tính trong hệ thống phân tán, nó được thực hiện bằng cách chuyển quyền truy nhập từ tiến trình này sang tiến trình khác nhằm mục đích phân tán công việc giữa các tiến trình mà không làm ảnh hưởng tới việc bảo vệ tài nguyên. Giả sử người sử dụng A có quyền đọc tập tin, người sử dụng muốn in nhưng vì kích thước tập tin quá lớn, đáng lẽ phải gửi tập tin đến máy in thì người sử dụng chỉ gửi tên tập tin để tiến trình trên máy in có thể sao chép vào vùng đệm của mình. Vấn đề nằm ở chỗ tiến trình thực hiện in không có quyền truy nhập đến tập tin trên, vì vậy máy tính của người sử dụng sẽ từ chối tiến trình in của máy chủ khi truy nhập đến tập tin, vấn đề chỉ có thể giải quyết bằng cách người sử dụng tạm thời ủy nhiệm quyền truy nhập đến tập tin cho máy in.

Trong hệ thống phân tán, các tiến trình có thể chạy trên những máy khác nhau và thậm chí trong những miền quản trị khác nhau, việc ủy nhiệm có thể tránh được chi phí

cũng như bảo vệ thông thường được xử lý trên máy cục bộ. Giải pháp chung cho vấn đề này là sử dụng đại diện, đó là thẻ cho phép chủ sở hữu vận hành với những quyền hạn ché mà chủ thẻ đã cấp. Tiến trình có thể tạo đại diện với những quyền giống như nó đang có, nếu tiến trình tạo một đại diện mới dựa trên đại diện hiện hành đang có thì đại diện suy dẫn cũng sẽ bị hạn chế trong những quyền của đại diện gốc, thậm chí có thể bị hạn chế nhiều hơn. Vấn đề này có thể giải quyết bằng hai cách, nếu A biết tất cả các thành viên và A muốn ủy nhiệm quyền đọc cho B thì chỉ cần gửi chứng chỉ  $[A,B,R]_A$ , B muốn chuyển quyền này cho C thì chỉ cần yêu cầu C liên hệ với A để nhận được chứng chỉ thích hợp, hình 9.36 thể hiện cấu trúc của đại diện dùng để ủy quyền.

Chứng chỉ			Phần riêng của bí mật
Quyền truy nhập	Phản công khai của bí mật	Chữ ký	
R	$S^+_{Proxy}$	$\text{sig}(A, \{R, S^+_{Proxy}\})$	$S^-_{Proxy}$

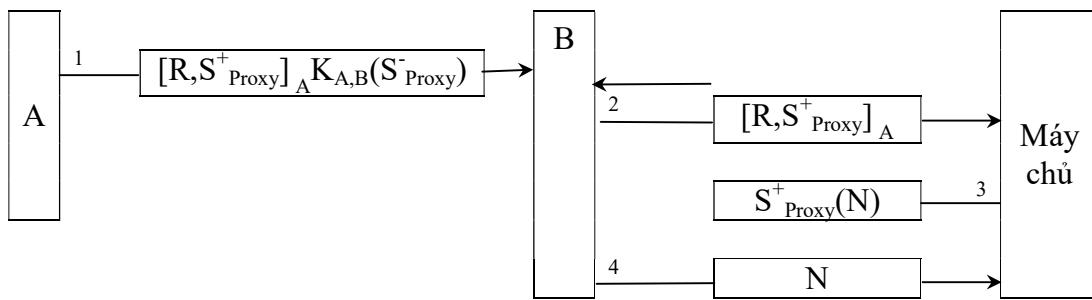
Hình 9.36 Cấu trúc chung của đại diện dùng để ủy quyền

Cách thứ hai, A đưa ra chứng chỉ và bất cứ thành viên nào nắm được chứng chỉ này sẽ có quyền thực hiện như A đã cấp, như vậy trường hợp này sẽ phải có biện pháp chống lại việc sao chép chứng chỉ bất hợp pháp. Lược đồ Neuman xử lý trường hợp này, cũng giống như việc tránh vấn đề mà A cần phải biết tất cả những thành viên sẽ được ủy nhiệm. Lược đồ Neuman gồm hai phần, giả sử A là tiến trình đã tạo ra đại diện, phần thứ nhất là  $C=[R,S^+_{Proxy}]_A$  bao gồm R là các quyền truy nhập đã cấp cho A và phản công khai của bí mật ký hiệu là  $S^+_{Proxy}$  được sử dụng để xác thực chủ nắm giữ chứng chỉ, chứng chỉ mang theo chữ ký  $\text{sig}(A, C)$  của A để bảo vệ chống lại sửa đổi.

Phần thứ hai chứa phần còn lại của bí mật ký hiệu là  $S^-_{Proxy}$  được bảo vệ để chống lại việc tiếp tục ủy nhiệm cho tiến trình khác. Nếu A muốn ủy nhiệm cho B một số quyền nào đó thì tạo danh sách R các quyền mà B có thể sử dụng, bằng cách ký vào danh sách, A sẽ ngăn chặn được B giả mạo nó. Tuy nhiên, chỉ ký thôi thì vẫn chưa đủ, nếu muốn sử dụng các quyền này thì có thể phải chứng minh B đã được sự nhận danh sách này từ A chứ không phải đánh cắp từ một chủ thẻ khác. Vì vậy, A đặt ra câu hỏi kiểm tra mà chỉ có nó mới biết câu trả lời, không một ai có thể kiểm chứng tính đúng đắn của câu trả lời đối với câu hỏi đã cho, câu hỏi này sẽ được thêm vào danh sách trước khi A thêm chữ ký.

Khi ủy nhiệm một số quyền của mình, A thêm câu hỏi vào danh sách các quyền và ký trước khi gửi cho B và đồng thời cung cấp đáp án sao cho không một ai có thể lấy cắp. Khi truy nhập đến C, C sẽ lấy câu hỏi từ danh sách để hỏi lại B, nếu B trả lời đúng chứng tỏ A đã thực sự ủy nhiệm các quyền trong danh sách cho B. Như vậy, A không cần phải biết tất cả các thành viên trong hệ thống, thực tế B có thể chuyển các quyền trong danh sách cho D, bằng cách tương tự cũng trao cho D đáp án của câu hỏi để D có thể chứng minh danh sách đã được chuyển từ đối tượng có thẩm quyền. Hình 9.37 thể hiện giao thức ủy quyền, A và B sử dụng khóa bí mật  $K_{A,B}$  để mã hóa thông điệp trao đổi với nhau. Đầu tiên A gửi cho B chứng chỉ  $C=\{R,S^+_{Proxy}\}$  ký bằng  $\text{sig}(A,C)$ , ký hiệu là  $[R,S^+_{Proxy}]_A$ ,

không cần thiết phải mã hóa thông điệp này, chỉ có phần riêng của bí mật  $S^-_{Proxy}$  mới phải mã hóa bằng khóa  $K_{A,B}$ .



Hình 9.37 Chứng minh sở hữu quyền truy nhập

Giả sử A được ủy quyền thực hiện một số thao tác trên đối tượng đặt tại máy chủ và đã ủy nhiệm cho B những thao tác này, nếu B muốn thao tác thực hiện trên đối tượng tại máy chủ thì phải chuyển ủy nhiệm cho máy chủ dưới dạng chứng chỉ đã ký  $[R, S^+_{Proxy}]_A$ . Nhận được ủy nhiệm, máy chủ sẽ kiểm tra xem danh sách quyền thực hiện C có bị làm giả hay không hoặc yêu cầu B trả lời câu hỏi kiểm hòc. Tuy nhiên máy chủ vẫn chưa biết B có sở hữu toàn quyền chứng chỉ hay không, vì vậy phải sử dụng bí mật đính kèm với C. Máy chủ sử dụng khóa  $S^+_{Proxy}$  để mã hóa chuỗi định danh N và gửi cho B, B sử dụng khoá riêng  $S^-_{Proxy}$  để giải mã và gửi lại giá trị N để chứng minh biết bí mật và nắm toàn quyền của chứng chỉ.

## 9.5 Một số vấn đề an toàn và bảo mật thông tin khác

Để đảm bảo an toàn cho thông tin trên hệ thống phân tán cần phải xây dựng các chính sách bảo mật, chính sách bảo mật cần phải được cụ thể hóa đến từng đối tượng và các thành phần trong hệ thống. Chính sách bảo mật máy chủ nhằm mục đích ngăn chặn truy nhập trái phép đến các máy chủ trong hệ thống, nội dung cơ bản của chính sách bảo mật máy chủ bao gồm chính sách bảo vệ vật lý và bảo vệ logic, chính sách bảo vệ vật lý bao gồm:

- Tránh nguy cơ ảnh hưởng của môi trường: Để tránh ảnh hưởng của môi trường đến thiết bị, trung tâm dữ liệu cần có điều hòa nhiệt độ, điều hòa độ ẩm. Các tham số về môi trường trong trung tâm dữ liệu cần được giám sát và cảnh báo từ xa. Ngoài ra, phòng đặt trung tâm dữ liệu phải không bị ảnh hưởng của nhiều từ trường.
- Đối với nguồn cung cấp điện: Để tránh những nguy cơ về điện làm ảnh hưởng đến hoạt động của hệ thống mạng, trung tâm dữ liệu cần được cung cấp một nguồn điện ưu tiên đặc biệt hơn các phòng khác. Cần xây dựng hệ thống điện của công ty đủ và liên tục để cung cấp nguồn điện cho các thiết bị. Đối với nguồn điện cần sử dụng bộ lưu điện, máy phát điện, nguồn cung cấp dự phòng và nên chú ý vấn đề cách điện nhằm ngăn chặn cháy nổ do điện gây ra. Ngoài ra, cần sử dụng hệ thống cảnh báo, giám sát theo dõi các tham số nguồn điện từ xa.
- Đối với việc bảo trì: Cáp sử dụng phải được đánh nhãn rõ ràng, cần bảo vệ tủ thiết bị để tránh đứt kết nối, lỏng đầu cáp hay cắm sai cổng, các dây phải được bó gọn.

Nên có một kho chứa các thiết bị dự phòng khi cần, thoát khỏi giao diện quản trị sau khi thực hiện xong công việc, bảo vệ truy nhập đến công kết nối trực tiếp đến thiết bị phục vụ cho quản lý, khi thay thế hay thao tác trực tiếp với các thiết bị cần chú ý tránh sôc điện làm hỏng thiết bị.

Chính sách bảo vệ logic bao gồm:

- Ghi chép thông tin cơ bản về máy chủ: Địa điểm lắp đặt, người chịu trách nhiệm quản lý trực tiếp, các thông tin về cấu hình và dịch vụ.
- Việc thay đổi cấu hình phải tuân thủ các qui định đã được phê duyệt.
- Loại bỏ hoặc tạm ngừng các dịch vụ và ứng dụng không cần thiết.
- Phải có qui trình theo dõi hoạt động của máy chủ.
- Thường xuyên cập nhật và cài đặt các bản vá lỗi cho hệ điều hành và các ứng dụng trên máy chủ.
- Hạn chế tối đa việc cấu hình tự động truy nhập hoặc cho phép truy nhập từ hệ thống khác.
- Xây dựng qui trình sao lưu dữ liệu.
- Xây dựng chính sách truy nhập từ xa đến máy chủ.
- Xây dựng qui định về bảo mật tài khoản truy nhập, qui định cách đặt tên và mật khẩu cho các tài khoản, thời gian quá hạn cho mỗi tài khoản.

Ngoài các chính sách trên, cần phải xây dựng chính sách bảo mật các máy khách nhằm đảm bảo an toàn cho thông tin được lưu trữ trên các máy chủ và thông tin mà các máy khách truy nhập. Phát hiện đột nhập là khả năng phát hiện, nhận dạng mã độc tấn công vào mạng hoặc một máy tính nào đó. Hệ thống phát hiện đột nhập có khả năng phát hiện các kiểu tấn công thăm dò, tấn công truy nhập, tấn công từ chối dịch vụ, worms, virus và có thể được cấu hình để gửi cảnh báo khi phát hiện những kiểu lưu lượng không rõ ràng. Ngăn chặn đột nhập là khả năng phát hiện và ngăn chặn việc thực thi các mã độc tấn công xâm nhập vào hệ thống, khi phát hiện có xâm nhập trái phép.

## THẢO LUẬN

1. Trình bày các nguy mất an toàn dữ liệu trong hệ phân tán?
2. Phân biệt khái niệm giải thuật mã hóa và hàm băm.
3. Lập trình giải thuật Triple DES.
4. Làm thế nào để chống nguy cơ tấn công “chèn thêm” trong chức năng đăng nhập.
5. Nêu một số ví dụ áp dụng các giải thuật mã hóa.
6. Lập trình giải thuật RSA.
7. Tìm hiểu hệ thống chứng chỉ bảo mật.
8. Trình bày cách áp dụng danh sách kiểm soát truy nhập trong các thiết bị định tuyến.
9. Tìm hiểu các tham số bảo mật trong tập tin cấu hình trang web.
10. Xây dựng giải pháp cho ứng dụng cảnh báo xâm nhập trái phép.

## CHƯƠNG 10: PHÁT TRIỂN HỆ THỐNG PHÂN TÁN

Hệ thống phân tán bao gồm nhiều tiến trình chạy trên các máy tính khác nhau, do đó phát triển hệ thống là vấn đề rất phức tạp, quan điểm xây dựng phần mềm theo cách tiếp cận đơn khôi không còn đáp ứng các yêu cầu trong xu thế phát triển. Dù sử dụng mô hình nào thì vấn đề trao đổi thông tin giữa các tiến trình vẫn là nhiệm vụ quan trọng cho tất cả các hệ thống, do đó chương này sẽ giới thiệu một số công nghệ đối tượng phân tán được phát triển dựa trên gọi thủ tục từ xa, sau đó sẽ giới thiệu kiến trúc hướng dịch vụ SOA và Microservice thường được áp dụng để phát triển các hệ thống phân tán.

### 10.1 Đối tượng phân tán

Gọi thủ tục từ xa mới chỉ đảm bảo tính trong suốt về truyền thông, nó đóng vai trò nền tảng cho những công nghệ đối tượng phân tán. Năm 1989 là mốc quan trọng trong công nghệ phần mềm, quan điểm xây dựng các ứng dụng theo hướng thủ tục chuyển sang hướng đối tượng. Đặc điểm cơ bản của lập trình hướng đối tượng là việc che giấu xử lý, do đó những thay đổi bên trong đối tượng sẽ không ảnh hưởng đến các đối tượng khác. Một đặc tính quan trọng của đối tượng là tính bao đóng các thuộc tính và phương thức xử lý, thao tác với các thuộc tính của đối tượng đều phải được thực hiện thông qua các hàm giao diện.

Để thuận tiện cho việc xây dựng hệ thống phân tán người ta đã nâng tầm phương pháp gọi thủ tục từ xa thành gọi đối tượng từ xa, ví dụ Java RMI, Microsoft DCOM, CORBA và dịch vụ web. Đối tượng phân tán kế thừa cả hai đặc điểm hướng đối tượng và gọi thủ tục từ xa, nó còn cung cấp thêm nhiều tiện ích tạo điều kiện thuận lợi dễ dàng phát triển các ứng dụng phân tán. Khi tiến trình máy khách gọi đối tượng phân tán, giao diện của đối tượng sẽ được nạp vào không gian địa chỉ của máy khách và đảm nhiệm vai trò tương tự như Stub trong phương pháp gọi thủ tục từ xa, nó đảm nhiệm chức năng chuyển đổi lời gọi các hàm giao diện thành dạng thông điệp để hệ điều hành chuyển đến máy chủ và ngược lại chuyển đổi thông điệp nhận được từ máy chủ thành kết quả trả về.

Đối tượng thực nằm ở phía máy chủ, giao diện của nó giống như tiến trình trên máy khách và đồng thời thành phần Skeleton có nhiệm vụ chuyển đổi thông điệp nhận được từ máy khách thành lời gọi thủ tục tương ứng và sau đó chuyển kết quả thực hiện thành thông điệp để hệ điều hành chuyển cho máy khách. Các đối tượng phân tán liên quan đến vấn đề biên dịch, ngôn ngữ biên dịch lại phụ thuộc vào ngôn ngữ lập trình, do đó chúng phải xây dựng một cách tinh minh, như vậy mới có thể xây dựng các ứng dụng dựa trên các ngôn ngữ lập trình khác nhau. Vấn đề cài đặt bên trong các đối tượng phân tán là công việc của các lập trình viên, điều quan trọng cần phải đảm bảo các tính thống nhất về mặt giao diện để các máy tính khác có thể gọi đối tượng. Giải pháp cho vấn đề này là xây dựng một thành phần thích nghi đối tượng để thể hiện các đối tượng phân tán trên môi trường mạng.

#### 10.1.1 Mô hình đối tượng thành phần phân tán

Đầu những năm 1990, hãng Microsoft phát triển mô hình đối tượng thành phần COM, đó là nền tảng cơ bản của liên kết và nhúng đối tượng OLE để xác định giao diện gữa các đối tượng thành phần. Mô hình đối tượng thành phần phân tán DCOM là phiên

bản mở rộng của COM nhằm hỗ trợ truyền thông giữa các đối tượng trên nhiều máy khác nhau qua mạng máy tính. Để các đối tượng có thể liên lạc với nhau trong môi trường mạng, DCOM cung cấp các dịch vụ cho ActiveX và hỗ trợ các giao diện lập trình ứng dụng. Về bản chất, các đối tượng phân tán tương tác với nhau bằng cách sử dụng cơ chế gọi thủ tục từ xa. DCOM chỉ thay thế các tương tác giữa các tiến trình máy khách và máy chủ bằng giao thức mạng, tương tự như gọi thủ tục từ xa, nó che giấu truyền thông. Ngay tên gọi của nó đã thể hiện quan điểm xây dựng ứng dụng dựa trên thành phần, các thành phần đều có thể tái sử dụng và phát triển mà không ảnh hưởng đến các thành phần khác.

Việc gọi các hàm của đối tượng DCOM tương tự như gọi hàm của đối tượng chạy trên máy tính cục bộ, điểm khác biệt cơ bản là trước khi gọi hàm của đối tượng phân tán thì phải nhúng đối tượng cục bộ vào đối tượng phân tán. Có thể thực hiện bằng cách gọi tường minh hoặc không tường minh, nếu gọi tường minh thì phải khai báo biến con trỏ trên máy tính máy khách, sau đó gọi hàm bind() để nhúng đối tượng phân tán vào biến con trỏ cục bộ.

### **10.1.2 Gọi phương thức từ xa trong Java**

Gọi phương thức từ xa Java RMI thực chất là việc áp dụng phương pháp gọi thủ tục từ xa trong ngôn ngữ lập trình Java. Java RMI cho phép một đối tượng chạy trên một máy ảo Java này có thể kích hoạt một phương thức của một đối tượng đang chạy trên một máy ảo Java khác. Đối tượng có phương thức được gọi từ xa gọi là các đối tượng ở xa, một ứng dụng RMI bao gồm chương trình máy chủ và một chương trình máy khách.

Chương trình máy chủ tạo một số các đối tượng từ xa, tạo các tham chiếu đến chúng và chờ những chương trình máy khách kích hoạt các phương thức của các đối tượng từ xa này. Chương trình máy khách lấy một tham chiếu đến một hoặc nhiều đối tượng từ xa trên máy chủ và kích hoạt các phương thức từ xa thông qua các tham chiếu. Một chương trình máy khách có thể kích hoạt các phương thức ở xa trên một hay nhiều máy chủ, nghĩa là việc thực thi của chương trình được trải rộng trên nhiều máy tính. Đây chính là đặc điểm của các ứng dụng phân tán, RMI là cơ chế để xây dựng các ứng dụng phân tán bằng ngôn ngữ Java.

Mô hình khách/chủ dựa trên RMI bao gồm máy chủ là chương trình cung cấp các đối tượng có thể được gọi từ xa và máy khách là chương trình tham chiếu đến các phương thức của các đối tượng ở xa trên máy chủ. Stub chứa các tham chiếu đến các phương thức ở xa trên máy chủ, trong khi đó Skeleton đón nhận các tham chiếu từ Stub để kích hoạt phương thức tương ứng trên máy chủ, thành phần tham chiếu từ xa là đóng vai trò truyền thông của RMI.

Cơ chế định vị đối tượng từ xa trong RMI xác định cách thức mà chương trình máy khách có thể lấy được tham chiếu đến các đối tượng từ xa. Thông thường người ta sử dụng dịch vụ đặt tên lưu giữ các tham chiếu đến các đối tượng cho phép gọi từ xa mà máy khách sau đó có thể tìm kiếm. Cơ chế giao tiếp với các đối tượng từ xa chi tiết hóa cơ chế giao tiếp với các đối tượng ở xa được cài đặt bởi RMI. Java RMI tải các lớp dạng bytecodes cho các lớp mà nó được chuyển tải qua lại giữa máy ảo, vì RMI cho phép các chương trình gọi phương thức từ xa trao đổi các đối tượng với các phương thức từ xa

dưới dạng các tham số hay giá trị trả về của phương thức nên RMI cần có cơ chế cần thiết để tải mã Bytecodes của các đối tượng từ máy ảo này sang máy ảo khác, Java hỗ trợ các lớp cần thiết để cài đặt các ứng dụng khách/chủ dựa trên RMI trong các gói java.rmi.

DCOM lại chỉ tương thích với các hệ thống được xây dựng bằng các công cụ phát triển phần mềm của hãng Microsoft, Java RMI cũng chỉ hỗ trợ cho các phần mềm phát triển bằng chính ngôn ngữ này. Những hạn chế trên hạn chế đáng kể cho việc tích hợp các hệ thống phân tán, trong trường hợp này nên sử dụng dịch vụ web hoặc CORBA.

### **10.1.3 Dịch vụ web**

Dịch vụ web là một mô hình phát triển dựa trên nền tảng công nghệ web, nó bao gồm tập các chuẩn cho phép các xây dựng các ứng dụng phân tán mà không phụ thuộc nền tảng hệ thống. Dịch vụ web sử dụng các giao thức mở và bên trong tự chứa các đặc tả dữ liệu theo chuẩn XML, đó là ngôn ngữ mô tả dữ liệu để trao đổi giữa các hệ thống. Tốc độ trao đổi thông tin dựa trên dịch vụ web thường không cao, tuy nhiên việc phát triển các ứng dụng này tương đối đơn giản, rút ngắn thời gian phát triển phần mềm.

Dịch vụ web liên lạc với máy khách thông qua các thông điệp XML được chuyển qua mạng nhờ các giao thức của tầng ứng dụng như HTTP, FTP, SMPP..., do đó các ứng dụng có thể truy nhập dịch vụ web để trao đổi thông tin một cách dễ dàng. Các ứng dụng dịch vụ web có thể được viết bằng nhiều loại ngôn ngữ lập trình khác nhau như Java, C#.... và có thể dễ dàng mở rộng hoặc nâng cấp.

Để tạo ra ứng dụng lớn với nhiều đơn vị tham gia, ứng dụng này sẽ được chia thành những thành phần nhỏ hay dịch vụ chia sẻ đặt tại các máy chủ khác nhau, chúng được đóng gói và giao dịch qua mạng sử dụng các giao thức chuẩn của tầng ứng dụng. Ưu điểm của các giao thức chuẩn là có thể được truy cập một cách dễ dàng qua môi trường Internet, khắc phục được tính đóng của các công nghệ phân tán khác.

Nền tảng của dịch vụ web là XML và HTTP, XML là ngôn ngữ mô tả dữ liệu dùng để trao đổi thông tin giữa các hệ thống có nền tảng và ngôn ngữ lập trình khác nhau, HTTP là giao thức phổ biến được sử dụng nhiều nhất trên mạng Internet. Các thành phần dịch vụ của dịch vụ web bao gồm giao thức truy nhập đối tượng đơn giản SOAP, mô tả vạn năng khám phá và tích hợp UDDI và ngôn ngữ mô tả dịch vụ web WSDL. SOAP là giao thức dựa trên ngôn ngữ XML, được dùng để cho các ứng dụng trao đổi thông tin qua giao thức HTTP. UDDI là dịch vụ thư mục, nơi các ứng dụng có thể đăng ký hay tìm kiếm dịch vụ. WSDL là ngôn ngữ dựa trên ngôn ngữ XML, dùng để xác định các mô tả trong dịch vụ web.

Dịch vụ web gồm ba thành phần chính, đó là thành phần cung cấp dịch vụ, thành phần sử dụng dịch vụ và môi trường dịch vụ. Thành phần cung cấp dịch vụ công bố sự hiện diện của dịch vụ web cho môi trường dịch vụ, bên yêu cầu dịch vụ tìm kiếm và yêu cầu thông tin dịch vụ tại môi trường này. Khi tìm thấy thông tin yêu cầu, phía máy khách sẽ nhúng dịch vụ web vào ứng dụng. Bên cung cấp dịch vụ tạo ra một dịch vụ web, nó sử dụng WSDL để mô tả, sau đó đăng ký dịch vụ. Ứng dụng khách xác định và yêu cầu dịch vụ đã được đăng ký bằng cách truy vấn UDDI, nếu tìm thấy thì nhúng dịch vụ vào ứng dụng, dữ liệu được trao đổi dưới dạng XML trên nền giao thức HTTP.

SOAP là giao thức truyền thông giữa các ứng dụng qua môi trường Internet, nó định dạng thông điệp trao đổi dựa trên ngôn ngữ XML và giao thức HTTP. XML định dạng dữ liệu dưới dạng văn bản, do đó có thể dễ dàng đi qua tường lửa, giao thức SOAP đơn giản vì có khả năng mở rộng và độc lập với ngôn ngữ lập trình cũng như nền tảng hệ thống. Giao thức SOAP mô tả rõ định dạng của thông điệp yêu cầu và trả lời dùng trong việc gửi và nhận thông tin thông qua giao thức HTTP nhờ phương thức POST. Giao thức SOAP hỗ trợ gọi thủ tục từ xa hoặc truyền văn bản, nhiều dịch vụ web sử dụng hình thức gọi thủ tục từ xa vì có thêm nhiều trợ giúp. Nếu sử dụng cách truyền văn bản thì lập trình viên phải mô tả dữ liệu cũng như phương thức sẽ gọi để thực hiện yêu cầu, tuy nhiên đây là cách linh hoạt hơn rất nhiều so với hình thức gọi thủ tục từ xa. Các định dạng thông điệp, tham số và lời gọi đến các hàm giao diện lập trình ứng dụng trong hai kiểu trên khác nhau, do đó việc quyết định chọn cái nào tùy thuộc vào thời gian và đặc điểm của dịch vụ cần xây dựng.

WSDL là ngôn ngữ mô tả giao tiếp và thực thi dựa trên thông điệp định dạng XML, WSDL dùng để mô tả và xác định dịch vụ web, nó cho biết dịch vụ web cung cấp chức năng gì và được đặt tại đâu, cách thức truy nhập và gọi phương thức cung cấp dịch vụ. UDDI sử dụng ngôn ngữ XML và các giao thức HTTP, DNS và truyền thông qua SOAP để thực hiện vai trò trung gian trong quá trình trao đổi thông tin, nó là một thư mục dùng để lưu thông tin của mọi dịch vụ web. Danh sách các dịch vụ web được mô tả bằng ngôn ngữ WSDL sẽ được gửi tới UDDI để đăng ký, tại đây sẽ ánh xạ danh sách dịch vụ thành bản ghi UDDI định dạng XML, bản ghi đăng ký UDDI bao gồm trang trắng, trang vàng và trang xanh.

Để xây dựng một ứng dụng phân tán dựa trên dịch vụ Web, trước hết cần định nghĩa và xây dựng các chức năng mà dịch vụ sẽ cung cấp, sau đó tạo WSDL cho dịch vụ, chuyển đổi thông tin giữa máy chủ và WSDL và xây dựng SOAP trên máy chủ. Sau khi đã hoàn thành viết các chức năng cung cấp dịch vụ thì phải đăng ký WSDL với UDDI để cho phép các máy khách có thể tìm thấy và truy nhập. Lập trình viên cho ứng dụng máy khách nhận tập tin định dạng WSDL và từ đó dựng SOAP cho máy khách để có thể kết nối với SOAP trên máy chủ. Cuối cùng, xây dựng ứng dụng phía khách, sau đó gọi dịch vụ thông qua việc kết nối đến máy chủ cung cấp dịch vụ web.

#### **10.1.4 Kiến trúc môi trường yêu cầu đối tượng chung**

Kiến trúc môi trường yêu cầu đối tượng chung CORBA do OMG đưa ra từ năm 1990, nhóm này cung cấp các sườn kiến trúc chung cho lập trình hướng đối tượng, trong đó mô tả rõ các kiến trúc quản lý đối tượng như môi trường yêu cầu đối tượng, dịch vụ đối tượng, giao diện và các đối tượng ứng dụng. CORBA được xây dựng và phát triển dựa trên các đặc tả trên nhằm mục đích chuẩn hóa việc xây dựng các ứng dụng phân tán. Trong CORBA, các đối tượng phân tán được định nghĩa để có thể lưu giữ và gọi từ xa, do đó nó phải đảm bảo tất cả các nhiệm vụ chung liên quan tới việc lập trình đối tượng phân tán bao gồm việc đăng ký và sử dụng đối tượng phục vụ.

CORBA bao gồm các tiêu chuẩn hỗ trợ việc phát triển các ứng dụng theo mô hình hướng đối tượng, trong đó các đối tượng có thể thực hiện trên một bộ xử lý hoặc được

phân tán trên mạng. Để đạt được mục tiêu tích hợp các hệ thống của các nhà cung cấp khác nhau, kiến trúc CORBA đảm bảo không phụ thuộc hệ điều hành và thiết bị phần cứng, không phụ thuộc ngôn ngữ lập trình và không phụ thuộc vị trí vật lý cài đặt các ứng dụng. Các đối tượng trong CORBA được thể hiện và cài đặt tương tự như các lớp trong ngôn ngữ lập trình hướng đối tượng do đó có tính tương thích cao và hoàn toàn độc lập với hệ điều hành. Nhằm mục đích chuẩn hóa kiến trúc này, nhóm OMG đưa ra các đặc tả giao diện, có thể thấy nó rất giống với mô hình đối tượng đã được đề cập khi xây dựng hệ thống phân tán.

Thành phần cơ bản trong kiến trúc CORBA là môi trường yêu cầu đối tượng ORB, nó cung cấp các giao diện, dịch vụ và các phương tiện phục vụ cho các đối tượng CORBA. Khi ORB nhận được yêu cầu của máy khách nó chuyển yêu cầu đó đến máy chủ thích hợp. Theo cách lập trình khách/chủ truyền thống, máy khách phải biết tên và vị trí của máy chủ, phương pháp thiết lập và duy trì kênh liên lạc, phương pháp gửi/nhận các tham số. Nếu sử dụng CORBA, máy khách chỉ cần biết tên của thủ tục cần gọi và các tham số cần thiết để gửi/nhận, mọi thao tác khác do các thành phần của CORBA đảm nhiệm. Yêu cầu của máy khách sẽ được chuyển đến môi trường ORB, việc xử lý trên thành phần phục vụ nào sẽ do môi trường này quyết định. Như vậy, phương pháp tiến trình bền bỉ vốn rất phức tạp đã được giải quyết, áp dụng kiến trúc CORBA có thể dễ dàng xây dựng hệ thống có khả năng chịu lỗi.

Môi trường ORB là thành phần cơ bản trong CORBA, đó là phần mềm nằm giữa tầng ứng dụng và tầng vận tải. Bản thân ORB không thuộc thành phần của hệ điều hành mà là một dạng phần mềm trung gian dưới sự kiểm soát của hệ điều hành, nó đảm nhiệm các công việc sau:

- Sẵn sàng nhận các yêu cầu của máy khách.
- Cung cấp tất cả các thủ tục cần thiết để tìm ra đối tượng thực hiện yêu cầu của máy khách.
- Trao đổi dữ liệu và gửi yêu cầu đến thành phần cung cấp dịch vụ.
- Cung cấp một số dịch vụ khi yêu cầu trong thời gian thực hiện.

Ứng dụng máy khách là đối tượng yêu cầu dịch vụ, nó chỉ cần liên hệ với giao diện giao diện hoàn toàn độc lập với vị trí cài đặt đối tượng thực hiện. Servant là các đối tượng phục vụ yêu cầu của các máy khách, nó định nghĩa các thao tác, hỗ trợ giao diện IDL của CORBA. ORB là môi trường trung gian thiết lập quan hệ khách/chủ giữa các đối tượng. bằng cách sử dụng ORB, đối tượng máy khách có thể gọi các hàm trên máy cục bộ hoặc trên mạng. Giao diện với môi trường ORB cung cấp các giao diện để máy khách và Servant kết nối với nhau qua môi trường này. Stub và Skeleton tương tự như trong gọi thủ tục từ xa Stub gồm các lệnh cho phép máy khách truy nhập tới các thành phần của máy chủ, Skeleton gồm các lệnh trên máy chủ để liên lạc với các thành phần CORBA, nó đóng vai trò cầu nối giữa ORB với đối tượng thực hiện trên máy chủ.

Máy khách có thể sử dụng Stub hoặc giao diện gọi đối tượng động, một số chức năng có thể trực tiếp tương tác với môi trường ORB. Môi trường ORB chuyển yêu cầu của máy khách đến đối tượng phục vụ thông qua Skeleton tĩnh hoặc động, có thể định

nghĩa giao diện tĩnh hoặc động. Giao diện tĩnh được định nghĩa trong ngôn ngữ IDL để qui định các loại đối tượng và các thao tác trên đối tượng, giao diện động sử dụng dịch vụ kho lưu trữ giao diện để thể hiện các thành phần của giao diện như các đối tượng phục vụ và cho phép truy nhập các thành phần này trong thời gian chạy. Đối với môi trường ORB thì phương pháp sử dụng ngôn ngữ định nghĩa giao diện hay sử dụng dịch vụ kho giao diện đều có ý nghĩa như nhau.

Ứng dụng máy khách khởi tạo yêu cầu bằng cách gọi các thủ tục trong Stub liên quan đến đối tượng hoặc bằng cách tạo yêu cầu theo phương pháp động. Môi trường ORB xác định mã thực hiện thích hợp, chuyển các tham số và chuyển quyền điều khiển cho đối tượng phục vụ thông qua Skeleton của ngôn ngữ định nghĩa giao diện hoặc qua Skeleton động. Trong quá trình thực hiện đối tượng Servant có thể sử dụng một số dịch vụ của môi trường ORB thông qua thành phần thính nghi đối tượng. Khi thực hiện xong yêu cầu, quyền điều khiển và các giá trị trả về sẽ được chuyển cho ứng dụng máy khách.

Theo kiến trúc của CORBA, ứng dụng máy khách sử dụng tham chiếu đối tượng để gọi các hàm do đối tượng cung cấp. Để gọi một đối tượng cần phải tiến hành các bước như: xác định đối tượng sẽ gọi, hàm thực hiện và các tham số cung cấp cho hàm cũng như giá trị trả về của nó. Môi trường ORB quản lý các quá trình chuyển quyền điều khiển và chuyển dữ liệu giữa đối tượng thực hiện và ứng dụng máy khách, ứng dụng máy khách gọi các hàm của đối tượng phục vụ tương tự như gọi các hàm trên máy cục bộ.

Đối tượng thực hiện là các đối tượng bao hàm cả dữ liệu và các thao tác thực hiện trên dữ liệu, theo cách hiểu trong lập trình hướng đối tượng đây là các thể hiện của đối tượng. Chức năng thực hiện của đối tượng này hoàn toàn độc lập với môi trường ORB, môi trường này chỉ đảm nhiệm liên kết giữa máy khách và đối tượng thực hiện. Tham chiếu đối tượng là thao tác nhận thông tin cần thiết để xác định một đối tượng thực hiện trong môi trường ORB, tương tự như mô hình lập trình trên một máy tính, có thể hình dung thao tác này tương tự như thao tác lấy địa chỉ của đối tượng phục vụ. Để gọi các thao tác trên đối tượng phục vụ, máy khách phải nhận được tham chiếu đối tượng do môi trường ORB cung cấp. Môi trường ORB sử dụng phương pháp ánh xạ như nhau đối với tất cả các ngôn ngữ lập trình, vì vậy ứng dụng viết trên các ngôn ngữ khác nhau hoàn toàn có thể giao tiếp với nhau trong môi trường ORB của kiến trúc CORBA. Các yêu cầu cài đặt nói chung phụ thuộc nhiều vào ba yếu tố: thiết bị phần cứng, hệ điều hành và ngôn ngữ lập trình sử dụng để viết mã lệnh.

Môi trường ORB tạo điều kiện thuận lợi cho việc trao đổi thông tin giữa các đối tượng và để các đối tượng xác định nhau. Tuy nhiên các tính năng đó chưa đủ để xây dựng các ứng dụng phân tán quy mô lớn, đặc biệt đối với các ứng dụng thích hợp cho chuyên ngành hẹp, vì vậy hãng OMG đã đưa ra một số tính năng mới thể hiện dưới dạng các dịch vụ và các phương tiện như minh họa trên hình 10.8. Nhóm OMG chỉ nêu đặc tả các dịch vụ giao diện mà các dịch vụ cung cấp mà không thể hiện cách cài đặt các dịch vụ như thế nào, các dịch vụ đó độc lập với môi trường ORB và do đó số lượng dịch vụ phụ thuộc vào sản phẩm CORBA của từng hãng.

Dịch vụ đặt là một trong các dịch vụ cơ bản nhất trong các dịch vụ của kiến trúc CORBA nói riêng và trong các ứng dụng mạng nói chung, nó cho phép các ứng dụng máy khách xác định được đối tượng phục vụ bằng cách cung cấp tên sau đó sẽ nhận được tham chiếu đến đối tượng. Dịch vụ này cho phép các đối tượng Servant được đăng ký và xác định bằng tên, nó sử dụng ngữ cảnh đặt tên để bảo đảm tập các tên là duy nhất. Sự kết hợp giữa tên với đối tượng gọi là ràng buộc tên và luôn được định nghĩa liên quan với ngữ cảnh của tên.

Dịch vụ đặt tên cho phép kết hợp một hoặc nhiều tên logic với một tham chiếu đối tượng, lưu giữ các tên trong không gian tên dưới dạng phân cấp và cho phép đăng ký tên đối tượng trong thời gian chạy. Các ứng dụng máy khách có thể tìm ra đối tượng cần sử dụng bằng cách xác định tham chiếu đối tượng trong không gian tên.

Kiến trúc CORBA tuân thủ các yêu cầu của phần mềm trung gian, điều này được thể hiện trong các khía cạnh về định nghĩa giao diện nó sử dụng ngôn ngữ IDL, xác định đối tượng và cơ chế gọi các thao tác trên đối tượng phục vụ. Để truy nhập các đối tượng từ xa, ứng dụng máy khách cần có địa chỉ của đối tượng phục vụ, thông tin này chưa trong bảng tham chiếu đối tượng.

Khi đã xác định được giao diện và địa chỉ của đối tượng phục vụ, có thể gọi các thao tác từ xa, quá trình này tuân thủ các bước thực hiện của phần mềm trung gian. Bắt đầu từ việc biên dịch tập tin mô tả giao diện viết trên ngôn ngữ IDL sẽ nhận được Máy trạm stub và Máy chủ skeleton. Yêu cầu của ứng dụng máy khách được chuyển đến đối tượng phục vụ thông qua môi trường ORB, quá trình này hoàn toàn trong suốt đối với người phát triển.

Để xây dựng phần mềm của hệ thống phân tán theo kiến trúc CORBA cần phải xác định các đối tượng cung cấp dịch vụ và đối tượng sử dụng dịch vụ. Khái niệm tiền trình máy khách và tiền trình máy chủ trong gọi thủ tục từ xa dường như không còn nữa, các thành phần trong hệ thống chỉ là những đối tượng. Áp dụng CORBA có thể dễ dàng xây dựng các ứng dụng ngang hàng, tất nhiên vẫn dễ dàng xây dựng các ứng dụng theo mô hình khách/chủ. Có thể thấy, việc xây dựng ứng dụng tương tự như gọi thủ tục từ xa, che giấu truyền thông là điểm dễ dàng nhận thấy. Các tiền trình cung cấp dịch vụ có thể nhân bản để chạy trên nhiều máy tính khác nhau, nếu một tiền trình sụp đổ thì các tiền trình khác sẵn sàng thay thế, đây là ưu điểm nổi bật khi áp dụng xây dựng các hệ thống có khả năng chịu lỗi.

## 10.2 Kiến trúc hướng dịch vụ

Sự phát triển nhanh chóng của công nghệ làm nảy sinh môi trường giao tiếp không đồng nhất giữa các thành phần trong hệ thống phân tán. Một vấn đề đặt ra đối với các tổ chức công nghệ thông tin là làm sao xây dựng được một kiến trúc phần mềm có khả năng tích hợp và sử dụng các thành phần mới nhằm giảm thiểu chi phí phát triển và bảo trì hệ thống phần mềm. Mô hình đối tượng phân tán có đặc tính ràng buộc chặt chẽ giữa các thành phần với nhau làm cho các kiến trúc này chưa thật hiệu quả, kiến trúc hướng dịch vụ ra đời gần nhằm mục đích quyết những vấn đề khó khăn trong giao tiếp giữa các thành phần của hệ thống phân tán.

### **10.2.1 Giới thiệu kiến trúc hướng dịch vụ**

Kiến trúc hướng dịch vụ SOA là cách tiếp cận tổ chức hệ thống thông tin sao cho có thể truy nhập các tài nguyên thông qua các giao diện và thông điệp. Kiến trúc này dùng trong các chuẩn mở để biểu diễn các dịch vụ thông qua các giao diện đã được chuẩn hóa, từng thành phần riêng lẻ sẽ trở thành những khôi cơ bản để có thể tái sử dụng trong các ứng dụng khác, vì vậy kiến trúc này được sử dụng để tích hợp các ứng dụng.

Kiến trúc hướng dịch vụ có thể hiểu đó là tập hợp các dịch vụ và chức năng phù hợp với doanh nghiệp mà họ muốn cung cấp cho khách hàng, đó là phong cách yêu cầu mô tả dịch vụ và cách tương tác giữa các dịch vụ. Kiến trúc này chứa đựng những nguyên tắc, mẫu và tiêu chí giải quyết các vấn đề về tính mô-đun hệ thống, vấn đề đóng gói và ghép nối cấu thành hệ thống.

Dịch vụ là các hàm chức năng thực hiện theo quy trình nghiệp vụ nào đó, chúng có thể dễ dàng tìm thấy và liên thông với nhau, tuy nhiên mức độ gắn kết không cao. Mỗi dịch vụ bao gồm nhiều thành phần và được đóng gói ở mức cáo, người sử dụng không cần biết vị trí của chúng. SOA bao gồm các dịch vụ kết nối mềm dẻo với nhau, mỗi dịch vụ có giao diện được định nghĩa rõ ràng và độc lập với nền tảng của hệ thống và có thể tái sử dụng. SOA là cấp độ cao hơn của sự phát triển ứng dụng, chú trọng đến quy trình nghiệp vụ và dùng giao diện chuẩn để che giấu sự phức tạp bên trong.

Thiết kế SOA tách riêng phần thực hiện dịch vụ với giao diện gọi dịch vụ, điều này tạo nên một giao diện nhất quán cho ứng dụng sử dụng dịch vụ mà không cần quan tâm tới công nghệ thực hiện dịch vụ. Thay vì xây dựng các ứng dụng đơn lẻ và đồ sộ, nhà phát triển sẽ xây dựng các dịch vụ tinh gọn hơn có thể triển khai và tái tạo sử dụng trong toàn bộ quy trình nghiệp vụ. Điều này cho phép tái sử dụng phần mềm tốt hơn, cũng như tăng sự mềm dẻo vì các nhà phát triển có thể cải tiến dịch vụ mà không làm ảnh hưởng đến ứng dụng sử dụng dịch vụ.

SOA không hoàn toàn mới, DCOM và CORBA cũng có kiến trúc tương tự, tuy nhiên các kiến trúc cũ ràng buộc các thành phần với nhau quá chặt ví dụ như các ứng dụng phân tán muốn làm việc với nhau phải được thoả thuận về chi tiết tập hàm giao diện lập trình ứng dụng, một thay đổi mã lệnh trong thành phần DCOM sẽ yêu cầu những thay đổi tương ứng đối với mã lệnh truy nhập thành phần này. Ưu điểm lớn nhất của SOA là khả năng kết nối mềm dẻo và tái sử dụng, các dịch vụ có thể được sử dụng trên mọi nền tảng và được viết bằng bất kỳ ngôn ngữ lập trình nào.

SOA dựa trên hai nguyên tắc thiết kế quan trọng, đó là tính mô đun và đóng gói, tính mô đun tách vấn đề lớn thành nhiều vấn đề nhỏ để thuận lợi cho việc xử lý, đóng gói là tính năng che giấu dữ liệu và nghiệp vụ bên trong đối với người sử dụng bên ngoài. Dịch vụ được thiết kế phù hợp với SOA cần phải được đóng gói cao ở mức độ cao và có thể dễ dàng tái sử dụng, việc cài đặt đảm bảo tính độc lập và trong suốt về vị trí.

### **10.2.2 Các dịch vụ**

Theo nghĩa thông thường, dịch vụ là hoạt động công việc của người này phục vụ người khác, trong kỹ thuật khái niệm dịch vụ được hiểu là chức năng tiếp nhận các yêu cầu và trả về kết quả thông qua giao diện chuẩn đã qui định. Kiến trúc hướng dịch vụ tập

trung vào qui trình nghiệp vụ trên các hệ thống khác nhau, mục tiêu cơ bản của dịch vụ là thể hiện tính năng tương ứng với hoạt động thương mại thực tế. Hình 1.17 minh họa kiến trúc hướng dịch vụ, nó phân rã các chức năng của hệ thống thành các dịch vụ, mỗi dịch vụ lại được phân rã thành các dịch vụ nhỏ hơn...

Có thể nói dịch vụ là nhân tố chủ yếu hình thành nền SOA, nói cách khác kiến trúc SOA lấy dịch vụ làm trung tâm để xây dựng các ứng dụng. Từ các quy trình, chính sách, nguyên lý hay phương pháp hiện thực trong SOA đều hướng đến khái niệm dịch vụ. Các công cụ được lựa chọn bởi SOA hướng đến việc tạo và triển khai các dịch vụ, ngay cả cơ sở hạ tầng thực thi được cung cấp bởi SOA cũng hướng đến việc thực thi và quản lý các dịch vụ.

Về mặt kỹ thuật, dịch vụ là những sản phẩm phần mềm được định nghĩa một cách rõ ràng thông qua các giao diện. Với góc nhìn của doanh nghiệp, dịch vụ được gắn với một chức năng thực tiễn mà nó đảm nhận trong hệ thống. Mỗi dịch vụ thường kèm theo những chính sách sử dụng như: quyền truy xuất, thời gian truy xuất, mức độ bảo mật, chi phí sử dụng dịch vụ... Dịch vụ kỹ thuật tái sử dụng định nghĩa các dịch vụ chỉ phục vụ cho mục đích nghiệp vụ và được tái sử dụng trong nhiều dòng dịch vụ khác nhau. Các dịch vụ thuộc loại này có thể tính đến dịch vụ truy xuất dữ liệu, đăng nhập, quản lý người dùng.

Phương châm dịch vụ kinh doanh là tập hợp các dịch vụ hỗ trợ cho nghiệp vụ nhằm mục đích phục vụ trực tiếp hay gián tiếp cho khách hàng thông qua hệ thống tự động. Các kênh dịch vụ kinh doanh thường được định nghĩa thành miền dịch vụ như tài chính, bán hàng, quảng cáo, sản xuất, vận chuyển, kỹ thuật, quản lý lợi nhuận, chăm sóc khách hàng.... Tất cả các dịch vụ trong miền dịch vụ nên có sự kết nối với nhau thông qua một từ điển dữ liệu chung để có thể dễ dàng vận hành trong hệ thống. Các dịch vụ trong những miền dịch vụ khác nhau có thể không đồng nhất về từ điển dữ liệu, do đó cần có các chính sách truyền dữ liệu khi có yêu cầu đồng bộ dữ liệu giữa các miền dịch vụ.

Hợp đồng dịch vụ là giao diện giữa dịch vụ nghiệp vụ và dịch vụ kỹ thuật nhằm che giấu những thể hiện chi tiết của dịch vụ kỹ thuật. Nền tảng dịch vụ Web gồm các chuẩn và phương tiện giúp các dịch vụ có thể giao tiếp với nhau một cách độc lập với công nghệ. Các quy trình và chính sách hướng dẫn của SOA bao gồm các chỉ dẫn cho sự phối hợp của các dịch vụ nhằm đạt mức lợi nhuận cao nhất cho doanh nghiệp. Các phương pháp và công cụ là các công cụ SOA sẽ dùng trong quá trình quản lý dự án, mô hình dịch vụ, mô hình dữ liệu, quản lý và phát triển hệ thống. Các nguyên lý và chỉ dẫn bao gồm các nguyên lý giúp cho các nhà kiến trúc và các nhà phát triển trong quá trình xác định các dịch vụ kỹ thuật và dịch vụ nghiệp vụ.

Dịch vụ là khái niệm chính trong kiến trúc hướng dịch vụ, mỗi dịch vụ được định nghĩa bởi một hợp đồng dịch vụ phân biệt rõ ràng giữa chức năng và hiện thực. Các dịch vụ chỉ nên giao tiếp với các dịch vụ khác thông qua những giao diện được định nghĩa một cách rõ ràng. Dịch vụ có thể được truy xuất thông qua những chuẩn dùng trong môi trường giao tiếp rộng như SOAP, WSDL, XML, HTTP, UDDI... và không nên có ràng buộc quá

chặt chẽ. Mỗi dịch vụ nên thực hiện những tác vụ rời rạc và cung cấp giao diện đơn giản khi truy xuất nhằm khuyến khích việc tái sử dụng chúng cho các hệ thống về sau. Các dịch vụ nên cung cấp các siêu dữ liệu định nghĩa các ràng buộc cũng như chức năng của chúng. SOA lấy dịch vụ làm trọng tâm, do đó cần cung cấp các công cụ giúp mô hình hóa, phát triển, triển khai, liên kết, quản lý và kiểm tra độ bảo mật của các dịch vụ, đó là các sản phẩm, công nghệ và tiện ích đã được phê chuẩn.

### **10.2.3 Mô hình cặp lỏng**

Khái niệm gắn kết ám chỉ đến một số điều kiện ràng buộc giữa các thành phần với nhau, chúng có thể là những điều kiện chặt hoặc lỏng, thậm chí có những ràng buộc không hề được biết trước. Hầu hết mọi kiến trúc phần mềm đều hướng đến tính ràng buộc lỏng giữa các thành phần và gọi là mô hình cặp lỏng.

Mức độ gắn kết của mỗi hệ thống ảnh hưởng trực tiếp đến khả năng chỉnh sửa hệ thống, gắn kết càng chặt thì càng ảnh hưởng đến phía sử dụng dịch vụ mỗi khi có thay đổi nào đó xảy ra. Mức độ gắn kết tăng dần khi bên sử dụng dịch vụ biết nhiều thông tin ngầm định của bên cung cấp dịch vụ. Ngược lại, nếu bên sử dụng dịch vụ biết ít thông tin chi tiết bên trong dịch vụ trước khi gọi nó thì quan hệ giữa hai bên càng lỏng.

SOA hỗ trợ gắn kết lỏng thông qua việc sử dụng hợp đồng và nhúng, người dùng truy vấn đến nơi lưu trữ và cung cấp thông tin dịch vụ để lấy thông tin, một thành phần trong hệ thống cung cấp dịch vụ truy vấn sẽ trả về tất cả những dịch vụ thỏa mãn tiêu chuẩn tìm kiếm, từ đó người dùng chỉ việc chọn dịch vụ cần thiết và thực thi phương thức trên đó theo mô tả dịch vụ. Bên sử dụng dịch vụ không cần phụ thuộc trực tiếp vào cài đặt của dịch vụ mà chỉ dựa trên hợp đồng mà dịch vụ đó hỗ trợ.

Mô hình cặp lỏng giúp gỡ bỏ những ràng buộc điều khiển giữa những thành phần đầu cuối, mỗi thành phần trong hệ thống phân tán có thể tự quản lý độc lập nhằm tăng hiệu suất, khả năng mở rộng và khả năng đáp ứng cao, những thay đổi trong cài đặt cũng được che giấu. Gắn kết lỏng đảm bảo tính độc lập giữa bên cung cấp và bên sử dụng nhưng nó đòi hỏi các giao diện phải theo chuẩn và cần một thành phần trung gian quản lý, trung chuyển yêu cầu giữa các thành phần đầu cuối.

### **10.2.4 Chu kỳ sống dịch vụ**

Dịch vụ là một phần của phần mềm trong hệ thống phân tán, do đó chu kỳ sống của dịch vụ cũng gần tương tự như chu kỳ sống của phần mềm, dịch vụ có thể ở giai đoạn phát triển hoặc đã đưa vào hoạt động sản xuất kinh doanh. Nếu ở giai đoạn phát triển cần phải chỉ ra được các tương tác dịch vụ và chỉ ra những dịch vụ cần thiết phải xây dựng, do đó dịch vụ ở giai đoạn này được hiểu là một phần xây dựng qui trình kinh doanh.

Phần mềm đã được đưa vào hoạt động kinh doanh sẽ phát sinh ra nhiều vấn đề và yêu cầu mới cần phải chỉnh sửa hoặc nâng cấp, có một số dịch vụ sẽ được sửa trực tiếp trong khi hệ thống vẫn đang vận hành. Tuy nhiên, nhiều trường hợp phức tạp sẽ đòi hỏi phải tạm thời ngừng dịch vụ để tập trung cho việc sửa đổi và nâng cấp.

### **10.2.5 Phân loại dịch vụ**

Kiến trúc hướng dịch vụ chia các dịch vụ thành dịch vụ cơ bản, dịch vụ tích hợp và dịch vụ qui trình, ba loại này liên quan mật thiết đến quá trình cung cấp dịch vụ. Dịch vụ cơ bản cung cấp các tính năng kinh doanh cơ bản nhất, chúng chưa được phân cho các dịch vụ khác, thời gian chạy tương đối ngắn và thuộc loại không trạng thái, do đó các dịch vụ này rất phù hợp với phương thức gọi đồng bộ.

Thực tế, các dịch vụ cơ bản thường được cài đặt để truy nhập dữ liệu và một số nghiệp vụ cơ bản như tạo một người dùng mới hoặc thay đổi mật khẩu... Các dịch vụ tích hợp được cấu thành từ một số dịch vụ cơ bản, nhìn chung thời gian thực hiện của các dịch vụ này cũng tương đối ngắn và thuộc loại không trạng thái, sự tích hợp ở đây có thể thuộc về một hoặc nhiều nền tảng. Dịch vụ qui trình khác với hai loại trên, nó phản ánh một qui trình kinh doanh, do đó thời gian thực hiện khá dài và thuộc loại có trạng thái.

### **10.2.6 Trục dịch vụ doanh nghiệp**

Trục dịch vụ doanh nghiệp là hạ tầng kiến trúc cho phép sử dụng các dịch vụ của hệ thống sản xuất, thường triển khai các ứng dụng, các nền tảng và các quy trình nghiệp vụ. Các dịch vụ này được liên kết và trao đổi thông tin với nhau nhưng không sử dụng một loại định dạng dữ liệu chung và cũng không có một chuẩn giao tiếp chung. Nếu cần giao tiếp với các hệ thống bên ngoài, vấn đề tích hợp sẽ mở rộng ra khỏi phạm vi của doanh nghiệp, nó bao chùm lên các hệ thống và quy trình nghiệp vụ của các doanh nghiệp khác nhau.

Những năm gần đây, một số giải pháp như tích hợp ứng dụng doanh nghiệp, doanh nghiệp với doanh nghiệp, kiến trúc hướng dịch vụ và dịch vụ Web đã tập trung giải quyết những vấn đề liên quan tới tích hợp hệ thống thông tin của các doanh nghiệp. Những giải pháp trên tập trung vào một vài vấn đề về tích hợp, nhưng chúng thường là sản phẩm của một công ty nào đó, giá thành đắt và tốn thời gian triển khai.

Trục dịch vụ doanh nghiệp theo tiêu chuẩn sẽ giải quyết các vấn đề liên quan đến việc tích hợp mà không cần phải loại bỏ những giải pháp đang sử dụng. Mục đích của trục dịch vụ doanh nghiệp là làm cho việc tích hợp các ứng dụng và quy trình trở nên thuận tiện hơn bằng cách cung cấp một quy trình phân tán, điều hướng thông minh, bảo mật và có thể tự động chuyển đổi dữ liệu. Trong trục dịch vụ doanh nghiệp, những dịch vụ trên là những dịch vụ nền tảng do đó các ứng dụng không cần phải thi hành riêng biệt những yêu cầu trên theo một cách thức riêng biệt của chúng.

Trục dịch vụ doanh nghiệp giải quyết những điểm yếu của những giải pháp có sẵn bằng cách tạo ra một nền tảng chuẩn cho việc tích hợp. Giải pháp điểm–điểm yêu cầu cứ N thành phần tham gia hệ thống thì phải có N-1 giao diện để có thể giao tiếp được với các thành phần còn lại được thay thế bằng giải pháp trực, mỗi thành phần chỉ cần một giao diện để giao tiếp với trực và như vậy sẽ giao tiếp với các thành phần còn lại. Trục dịch vụ doanh nghiệp đảm bảo giao tiếp phân tán, chuyển hướng, xử lý nghiệp vụ, ổn định và bảo mật, đồng thời cũng cung cấp các dịch vụ có khả năng cắm và chạy.

Xây dựng hệ thống dựa trên SOA thoát nhìn khá phức tạp, nó tùy thuộc vào góc nhìn đối với hệ thống, có thể đó là góc độ kinh doanh hay kỹ thuật. Mô hình logic phân

chia hệ thống thành các miền, mỗi miền đảm nhiệm vai trò và trách nhiệm riêng, khái niệm miền ở đây phản ánh một thực thể nào đó, ví dụ đó là công ty, phòng/ban... Đứng trên góc độ kỹ thuật có thể thấy trực dịch vụ doanh nghiệp đóng vai trò trung tâm, các vùng chỉ cung cấp các dịch vụ cơ bản và dịch vụ tích hợp, các dịch vụ qui trình được tách biệt riêng rẽ.

So với kiến trúc dựa trên thành phần, điểm khác biệt chính của SOA là cung cấp khả năng giao tiếp giữa các dịch vụ sử dụng thông điệp dựa trên các giao thức phổ biến như HTTP, FTP, SMTP, ... và vì vậy kiến trúc SOA mới có khả năng độc lập với nền tảng. Các dịch vụ hoạt động trên nền tảng khác nhau vẫn có thể giao tiếp với nhau nhờ vào các giao diện đã được chuẩn hóa để cộng tác xử lý một tác vụ nào đó.

Phương thức trao đổi thông điệp đã được tất cả các nền tảng và ngôn ngữ lập trình hỗ trợ, do đó các dịch vụ trên các nền tảng nào sẽ hoạt động với cấu trúc dữ liệu đặc thù của nền tảng đó. Trao đổi thông điệp có thể thực hiện theo cơ chế không đồng bộ, bên gửi và và bên nhận không cần phải chờ nhau, điều này giúp cho mỗi bên tiếp tục xử lý công việc sau khi gửi thông điệp mà không cần dừng thực thi để chờ thông điệp trả lời.

### 10.3 Kiến trúc MicroService

Các ứng dụng phần mềm phức tạp lớn được tạo thành từ nhiều dịch vụ nhỏ nhất có thể gọi là Microservices, chúng có thể được triển khai độc lập và kết hợp với nhau theo mối quan hệ không chặt chẽ. Mỗi Microservice tập trung hoàn thành tốt nhất một nhiệm vụ, nó được xây dựng bằng ngôn ngữ lập trình và công nghệ phù hợp nhất với nhiệm vụ đang thực hiện. Có thể viết phần mềm cho các Microservice bằng những ngôn ngữ lập trình khác nhau, tương tác giữa chúng được thực hiện thông qua giao diện lập trình ứng dụng API, việc xây dựng hệ thống chuyển sang chuẩn hóa về cách tích hợp chứ không phải về nền tảng xây dựng mỗi dịch vụ.

Microservice thường tập trung giải quyết một công việc cụ thể, do đó chúng thường có kích thước nhỏ, tất nhiên khái niệm nhỏ hay lớn cũng chỉ là tương đối, điều quan trọng phải giữ cho giao diện với các dịch vụ khác nhau nhất có thể. Mã nguồn của các dịch vụ này có thể được tái sử dụng cho các dịch vụ khác nhưng điều quan trọng nhất vẫn là vấn đề tối ưu sao cho có thể đạt được hiệu năng cao nhất. Mức độ chi tiết của dựa trên nhu cầu kinh doanh, ví dụ nhận dạng đối tượng dựa trên hình ảnh hoặc âm thanh... đều có thể xây dựng thành các Microservice và cung cấp cho các bên có nhu cầu sử dụng.

Sự kết hợp lỏng lẻo là đặc điểm của Microservice để có thể triển khai một cách độc lập, nó cho phép cài đặt nhanh chóng phục vụ nhu cầu kinh doanh, tuy nhiên điều này lại nảy sinh một số vấn đề. Việc tạo ra các dịch vụ quá chi tiết hoặc phụ thuộc quá nhiều vào các dịch vụ khác có thể làm độ trễ tăng lên, hơn nữa rủi ro tiềm ẩn lại nằm ở vấn đề nhát quản.

Về cơ bản, Microservice chỉ là một dạng chuyên môn hóa của SOA, mục tiêu và vấn đề cần giải quyết của nó thường ở mức độ nhỏ hơn. Trong khi SOA cố gắng đưa các dịch vụ này đến với bất kỳ ai muốn sử dụng thì Microservice được tạo ra với mục tiêu chỉ là một phần của một hệ thống phân tán duy nhất. Hệ thống phân tán này thường được xây

dụng bằng cách chia nhỏ một ứng dụng lớn thành các Microservice và tập hợp chúng như một ứng dụng duy nhất, nó không có tham vọng phục vụ nhiều hệ thống cùng một lúc.

Khác với SOA, Microservice thường tồn tại ngầm định, nó không yêu cầu thành phần trung gian như trục dịch vụ doanh nghiệp, chúng có thể nhận nhau thông qua tập tin cấu hình. Kiến trúc Microservice chủ yếu tập trung vào việc phát triển dần dần một ứng dụng lớn, đơn khôi thành một hệ thống microservice phân tán dễ quản lý, phát triển và triển khai hơn bằng cách sử dụng các phương pháp tích hợp liên tục. Việc tái sử dụng không diễn ra ở cấp độ dịch vụ, tất cả các Microservice trong hệ thống chủ yếu được điều khiển bởi một ứng dụng chính. SOA được thiết kế với tham vọng giải quyết các vấn đề kiến trúc doanh nghiệp rất phức tạp, với mục tiêu tạo điều kiện cho khả năng tái sử dụng ở mức độ cao. Tóm lại, cả kiến trúc SOA và kiến trúc microservice đều là các loại kiến trúc dịch vụ, vì cả hai đều xử lý các hệ thống dịch vụ phân tán giao tiếp qua mạng. Tuy nhiên, kết quả thực tế khá khác nhau, vì trọng tâm của SOA là khả năng tái sử dụng và khám phá, trong khi trọng tâm của Microservice lại là thay thế một ứng dụng đơn khôi bằng một hệ thống dễ quản lý và phát triển theo từng bước.

Microservice đang là xu hướng đang được nhiều doanh nghiệp hướng tới, nó giúp hạn chế phát sinh chi phí hạ tầng khi mở rộng qui mô hệ thống, tuy nhiên không nên cường điệu vai trò của loại kiến trúc này. Nếu xây dựng hệ thống mới thì đừng bao giờ sử dụng Microservice, ứng dụng ban đầu sẽ có qui mô tương đối nhỏ, đủ để các thành viên trong nhóm phát triển biết rõ về mỗi thành phần của hệ thống. Không nên cố gắng cài đặt Microservice mà không có quá trình triển khai và giám sát chặt chẽ, điều này đòi hỏi phải có đội ngũ phát triển và vận hành. Không nên tạo ra quá nhiều Microservice, mỗi dịch vụ đều tiêu tốn một lượng tài nguyên nhất định, sự tích tụ này đến một lúc nào đó sẽ vượt tầm kiểm soát, việc chia nhỏ sẽ đơn giản hóa dịch vụ nhưng lại làm phức tạp vấn đề tích hợp. Việc chia các dịch vụ quá nhỏ cũng sẽ làm tăng độ trễ, mỗi lời gọi dịch vụ đều phải được gửi qua mạng, do đó hiệu năng tổng thể của hệ thống sẽ bị suy giảm, thậm chí gây nên hiện tượng nghẽn mạng.

## THẢO LUẬN

1. Phân tích ưu điểm và nhược điểm của DCOM.
2. Phân tích ưu điểm và nhược điểm của Java RMI.
3. Mô hình DCOM khác với kiến trúc CORBA ở điểm nào?
4. Tìm hiểu một số phần mềm phát triển hệ thống theo kiến trúc CORBA.
5. Tại sao sử dụng kiến trúc CORBA có thể dễ dàng xây dựng hệ thống dự phòng nóng?
6. So sánh Java RMI, dịch vụ Web và CORBA trong khía cạnh về khả năng sẵn sàng của hệ thống.
7. Xây dựng ứng dụng khách/chủ dựa trên dịch vụ web.
8. Kiến trúc hướng dịch vụ SOA là gì? Nêu ưu và nhược điểm của kiến trúc này.
9. Phân tích ưu điểm và nhược điểm của Microservice
10. So sánh cách phát triển hệ thống theo mô hình đơn khôi, SOA và Microservice.

## TÀI LIỆU THAM KHẢO

- [1] A. S. Tanenbaum, M. V. Steen, "Distributed Systems: Principles and Paradigms", 3<sup>nd</sup> Edition, Pearson Education, 2020.
- [2] A. S. Tanenbaum, M. V. Steen, "Distributed Systems: Principles and Paradigms", 2<sup>nd</sup> Edition, Prentice-Hall, 2007.
- [3] G. Coulouris, J. Dollimore, T. Kinberg, G. Blair, "Distributed systems: Concept and Design", 5<sup>th</sup> Edition, Addison-Wesley, 2012.
- [4] N.M. Josuttis, "SOA in Practice – The Art of Distributed System Design", O'Reilly, 2007
- [5] <http://www.differencebetween.net/technology/difference-between-grid-computing-and-cloud-computing/>
- [6] <https://www.linux.com/training-tutorials/building-beowulf-cluster-just-13-steps>
- [7] <https://docs.microsoft.com/en-us/dotnet/framework/network-programming/socket-code-examples>
- [8] Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach.  
Shahir Daya, Shishir Narain.... IBM Redbook, First Edition August 2015, SG24