

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG



LUẬN VĂN TỐT NGHIỆP

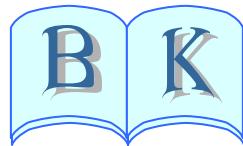
ĐỀ TÀI

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER



TP. HỒ CHÍ MINH - 1/2007

LỜI CẢM ƠN



Đầu tiên, em xin chân thành cảm ơn thầy Nguyễn Chí Ngọc đã nhiệt tình hướng dẫn và chỉ bảo em về kiến thức và tài liệu để em hoàn thành luận văn tốt nghiệp. Sau nữa là những thầy trường Đại học Bách khoa Thành phố Hồ Chí Minh cũng như các giáo sư, tiến sĩ, các kỹ sư thuộc các trường khác, đã hỗ trợ em trong lúc em làm luận văn. Sau cùng, em cũng xin chân thành cảm ơn gia đình, bạn bè, đặc biệt là bạn Trịnh Quốc Huy chuyên ngành công nghệ thông tin trường Khoa học Tự nhiên, Trịnh Ngọc Minh, Đại học Quốc gia TPHCM luôn sẵn lòng giúp đỡ và động viên cũng như cung cấp cho tôi những tư liệu trong suốt quá trình làm việc, để tôi có thể hoàn thành tốt luận văn này.

Sinh viên thực hiện:
Nguyễn Hồng Thái - Phạm Minh Trí

MỤC LỤC

Chương 1: TỔNG QUAN VỀ SERVER LOAD BALANCING	1
1.1. Tổng quan về Load Balancing	1
1.1.1. Load Balancing	1
1.1.2. Lợi ích về Load Balancing	1
1.1.3. Môi trường Load Balancing.....	1
1.1.4. Các ứng dụng của Load Balancing	2
❖ Server load balancing.....	2
❖ Global server load balancing	2
❖ Firewall load balancing.....	2
❖ Transparent cache switching.....	2
1.1.5. Các sản phẩm load-balancing.....	3
1.2. Cách tiếp cận cơ bản về Server Load Balancing	3
1.2.1. Tổng quan Web Server	3
1.2.2. Server Farm với một bộ Load Balancer	4
❖ Scalability (tính cân đối).....	4
❖ Availability (tính sẵn sàng).....	5
❖ Manageability (tính quản lý)	5
❖ Security (tính bảo mật)	6
❖ Quality of Service (viết tắt là QoS, tạm dịch là chất lượng dịch vụ)	7
1.2.3. Luồng packet cơ bản trong load balancing	7
1.2.4. Network-Address-Translation.....	8
❖ Destination NAT	9
❖ Source NAT	9
❖ Reverse NAT	11
❖ Port-Address Translation	11
1.2.5. Direct Server Return	13
1.2.6. Tóm tắt.....	15
Chương 2: HỆ ĐIỀU HÀNH UNIX	16
2.1. Giới thiệu lịch sử phát triển của Unix và Linux	16
2.1.1. Lịch sử các chuẩn UNIX.....	16
2.1.2 Hai dòng UNIX.....	16
2.1.3. Lịch sử phát triển của Linux	17
2.1.4. Vấn đề bản quyền của GNU project	18
2.1.5. Lý do sử dụng Linux.....	18
2.2. Hệ thống tiến trình của Linux, điều khiển các tiến trình	19
2.2.1. Inetd và các dịch vụ mạng.....	23
2.2.2. Hệ thống tập tin của Linux	25
a. Quyền truy cập sở hữu tập tin và thư mục của Linux.....	28
b. Sở hữu mặc định tập tin và thư mục	29

c. Quyền truy xuất tập tin	30
d. Liên kết tập tin.....	34
2.2.3. Quá trình khởi động và kết thúc của UNIX	34
2.2.4. Quản lý người sử dụng	36
a. Tạo user.....	38
b. Xóa user	39
2.2.5. Kết nối mạng thông qua TCP/IP	39
a. Hệ điều hành Linux và card mạng	40
b. Chúng ta có thể tiến hành cấu hình card mạng.....	41
Chương 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX	46
3.1. Domain Name Service	46
3.2. Dịch vụ mail, ftp.....	54
3.3. Dịch vụ proxy	59
Chương 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES	63
4.1. Giới thiệu về iptables.....	63
4.2. Cài đặt iptables	63
4.3. Cơ chế xử lý package trong iptables	63
4.4. Target và Jumps	64
4.5. Thực hiện lệnh trong iptables.....	64
4.6. Sử dụng chain tự định nghĩa	66
4.7. Lưu iptables script.....	66
4.8. Phục hồi script khi mất script file.....	67
4.9. Load kernel module cần cho iptables	68
4.10.Một số giá trị khởi tạo của iptables	68
4.11.Một số ví dụ về Firewall.....	70
4.12.Khắc phục sự cố trên iptables	75
4.13.iptables không khởi động.....	75
Chương 5: CÁC MÔ HÌNH MẠNG	76
5.1. Mô hình OSI.....	76
5.1.1. Lớp vật lý	76
5.1.2. Lớp liên kết dữ liệu	77
5.1.3. Lớp mạng.....	77
5.1.4. Lớp giao vận	77
5.1.5. Lớp phiên.....	77
5.1.6. Lớp trình bày.....	77
5.1.7. Lớp ứng dụng	77
5.2. Mô hình TCP/IP.....	78
5.2.1. Lịch sử phát triển mô hình TCP/IP	78

5.2.2. Lớp transport trong mô hình TCP/IP.....	78
5.2.3. Số port của TCP và UDP.....	79
5.2.4. Lớp internet của TCP.....	80
Chương 6: ĐỊNH TUYẾN.....	81
6.1. Làm thế nào để router có thể định tuyến packet từ nguồn đến đích.	81
6.2. Định địa chỉ network và host.....	82
6.3. Chọn đường đi và chuyển mạch packet	82
6.4. Routed Protocol và Routing Protocol.....	83
6.5. Định tuyến tĩnh và định tuyến động	85
6.6. Sử dụng định tuyến mặc định.....	86
6.7. Tại sao định tuyến động vẫn cần thiết.....	87
6.8. Các lớp routing protocol	88
6.9. Cơ bản về distance-vector protocol.....	89
a. Các vấn đề về routing host.....	89
b. Vấn đề về đếm vô định.....	91
c. Các phương pháp chống routing loop.....	91
6.10. Cơ bản về link-state routing	93
a. Trao đổi thông tin của link-state protocol.....	94
b. Truyền các thông tin về topo trong network đến các router.....	95
c. Hai vấn đề cơ bản của link-state	96
d. Một số ví dụ về định tuyến trong mạng LAN và WAN	97
Chương 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER	100
7.1. Cài đặt và cấu hình Web Server	100
7.1.1. Cài đặt Web Server	100
7.1.2. Cấu hình Web Server	101
7.2. Cài đặt và cấu hình FTP Server.....	105
7.2.1. Cài đặt FTP Server	105
7.2.2. Cấu hình FTP Server.....	107
7.3. Cấu hình để LAN có thể truy cập mạng bên ngoài.....	107
7.4. Cấu hình để mạng bên ngoài có thể truy cập được các Server.....	109
7.5. Kết quả của việc cấu hình trên.....	111
Chương 8: GIẢI PHÁP CẢI TIẾN FAULT-TOLERANT ROUTING, BROADCASTING VÀ LOAD BALANCING TRONG MẠNG HYPER-DEBRUIJN-ASTER.....	118
8.1. Giới thiệu về sự cải tiến mô hình HD*.....	118
8.2. Hyper-DeBruijn-Aster graph	119

8.2.1. Hypercube Graph H_n và De Bruijn Graph B_n	119
8.2.2. Hyper-DeBruijn-Aster Graph $HD^*(m,d,n)$	120
8.3. Shortest Routing và Diameter trong $HD^*(m,d,n)$	122
8.4. Fault Tolerance của $HD^*(m,d,n)$	126
8.5. Broadcasting và cân bằng tải trong $HD^*(m,d,n)$	127
8.6. Kết luận	127

Chương 9: MÔ HÌNH ĐỀ NGHỊ GLOBAL DYNAMIC LOAD BALANCING KẾT HỢP VỚI SHORTEST ROUTING TRONG HYPER-DEBRUIJN-ASTER MULTIPROCESSORS NETWORK 128

9.1. Giới thiệu	128
9.2. Cơ sở giải quyết vấn đề	129
9.2.1. Hyper Debruijn Aster Graph (HD[*]) :	130
9.2.1.1. Định tuyến trong Binary DeBruijn Network (BDB) :.....	130
9.2.1.2. Định tuyến trong mạng Binary Hypercube (BH)	130
9.2.1.3. Shortest Routing và Diameter trong $HD^*(m,d,n)$	131
9.2.1.4. Shortest Routing và Diameter trong Binary Hyper Debruijn Aster BHD [*] ($m,2,n$).....	132
9.3. Vấn đề Dynamic Load Balancing	133
9.4. So sánh mô hình Global và Local, Distributed và Centralized Load Balancing	134
9.5. Các hướng nghiên cứu có liên quan về Dynamic Load Balancing .. 134	
9.5.1. Dynamic Scheduling: Sắp xếp tiến trình động	135
9.5.2. Global Dynamic Load Balancing kết hợp với Shortest Routing trên BHD[*]	136
9.5.2.1. Mô hình Global Dynamic Load Balancing	136
9.5.2.2. Các đại lượng trong mô hình và một số định nghĩa	137
9.5.2.3. Giải thuật đề nghị cho mô hình Global Dynamic Load Balancing ..	138
a. Kỹ thuật thông tin, truyền dữ liệu giữa các node với Load Balancer và từ Load Balancer đến các node	138
b. Giải thuật đề nghị cho mô hình Global Dynamic Load Balancing ..	139
9.6. Đánh giá và kết luận	140

TÓM TẮT LUẬN VĂN

Cân bằng tải server là một trong những đề tài rất thú vị trong lĩnh vực mạng. Với thời đại hiện nay, mạng internet ngày càng lớn rộng ra. Do đó, việc xây dựng server để phục vụ người dùng internet không chỉ là một server mà phải cần đến cả một hệ thống server. Và đòi hỏi các server đó phải làm việc một cách thích nghi và đồng bộ với nhau. Mặt khác, với việc xây dựng một hệ thống server như vậy nếu chúng làm việc dưới dạng public toàn bộ thì vừa làm giảm không gian địa chỉ mạng, tức là làm tăng chí phí mạng, vừa khó quản lý, vừa tính bảo mật không cao. Mặt khác, nếu đứng ở vị trí người dùng thì nếu có quá nhiều địa chỉ server mà cùng phục vụ một mục đích nào đó thì rõ ràng rất phiền phức cho người dùng. Để khắc phục được những nhược điểm trên thì mô hình cân bằng tải server ra đời. Và vấn đề này được nhiều giới nghiên cứu ưu thích như các giới nghiên cứu về truyền thông, công nghệ thông tin. Và vì vậy nó sớm được phát triển, đến ngày nay nó ngày càng được các giới nghiên cứu hoàn thiện dần.

Và với vị trí là một sinh viên ngành viễn thông, chúng tôi đã nhận ra được điều này. Và với sự đam mê, thích nghiên cứu và học tập, chúng tôi đã quyết định bước vào tiếp tục nghiên cứu cải tiến nó, và vì chúng tôi chủ yếu nghiên cứu cho mô hình mạng và cụ thể là trên các server nên trong luận văn này chúng tôi chỉ dùng lại nghiên cứu “*Giải pháp cải tiến trong cân bằng tải server*”. Tuy nhiên, vì lợi ích của cân bằng tải không chỉ cho mạng internet, nó còn ứng dụng trong các lĩnh vực khác như bảo mật hệ thống, thiết kế vi mạch, Do đó, hướng phát triển đề tài là sẽ tiếp tục nghiên cứu ứng dụng cân bằng tải vào các lĩnh vực đó.

Mô hình cân bằng tải server trong thực tế là hệ thống bao gồm nhiều Server, và để giúp cho các Server hoạt động hiệu quả, giảm thiểu thời gian xử lý của hệ thống, tránh nghẽn trong mạng, chúng ta cần phải bố trí các Server theo một trật tự có tính mô hình. Ngoài ra, chúng ta cần có một giải pháp cân bằng tải trên các Server sao cho tận dụng tối đa khả năng xử lý của mỗi Server.

Trong phần nội dung, luận văn chúng tôi lẽ lần lượt trình bày các vấn đề sau đây:

Chương 1: Chúng tôi sẽ giới thiệu về cân bằng tải, cũng như chỉ ra được lợi ích và các ứng dụng của việc cân bằng tải. Đồng thời, trong chương này chúng tôi sẽ trình bày cách tiếp cận cân bằng tải server.

Chương 2, 3, 4, 5, 6: Chúng tôi sẽ trình bày về các vấn đề có liên qua đến cân bằng tải. Với mục đích làm nền tảng kiến thức trước khi bắt tay vào nghiên cứu các giải pháp cải tiến trong cân bằng tải server.

Chương 7: Chúng tôi sẽ trình bày cách thực hiện cân bằng tải với phương pháp áp dụng ipables vào web server và ftp server trên hệ thống mạng máy tính của phòng máy tính khoa Điện - Điện Tử trường Đại học Bách khoa TPHCM.

Chương 8: Chúng tôi sẽ trình bày giải pháp cải tiến cân bằng tải bằng giải thuật *Shortest Routing trên mạng Hyper-DeBruijn-Aster* do chúng tôi nghiên cứu.

Chương 9: Chúng tôi sẽ trình bày giải pháp cải tiến thứ hai trong cân bằng tải server với giải thuật *cân bằng tải Global Dynamic Load Balancing* mà chúng tôi nghiên cứu.

Vì một điều khách quan là thời gian để hoàn thành luận văn có hạn mà mô hình đề tài thì tương đối rộng, nên khó tránh khỏi những hạn chế và thiếu sót. Chúng tôi rất hoan nghênh những ý kiến đóng góp, những phản hồi từ các nhà chuyên môn các đồng nghiệp và độc giả.

Cuối cùng, chúng tôi xin chân thành cảm ơn các thầy cô Bộ môn Viễn thông, Khoa Điện - Điện tử, Trường Đại học Bách khoa Thành Phố Hồ Chí Minh cùng quý bạn bè đã giúp đỡ và đóng góp các ý kiến quý báu để chúng tôi hoàn thành tốt luận văn này.

Các ý kiến phản hồi có thể liên lạc theo địa chỉ được ghi ở phần tóm lược tiêu sử năm ở trang cuối của luận văn này.

Tác giả
Nguyễn Hồng Thái - Phạm Minh Trí

Chương 1: TỔNG QUAN VỀ SERVER LOAD BALANCING

1.1. Tổng quan về Load Balancing

1.1.1. Load Balancing:

- ❖ Load Balancing tạm dịch là cân bằng tải. Load Balancing không phải là một cách tiếp cận mới trong lĩnh vực server hoặc lĩnh vực mạng. Hầu hết, các sản phẩm thực hiện nhiều kiểu cách khác nhau về Load Balancing. Ví dụ, những router có thể phân phối lưu lượng thông qua nhiều đường đi để đến cùng một điểm đích.
- ❖ Load Balancing dựa trên những tài nguyên khác nhau. Một bộ Server Load Balancer, trên những thiết bị cầm tay khác, đã phân phối lưu lượng giữa những tài nguyên server hơn là tài nguyên mạng. Trong khi các bộ Load Balancer lại bắt đầu với việc cân bằng tải đơn giản. Các vấn đề này đã sớm được cải tiến thực hiện những chức năng khác nhau như: cân bằng tải, kỹ thuật lưu lượng và chuyển mạch lưu lượng thông minh.
- ❖ Các bộ Load Balancer có thể thực hiện các cuộc kiểm tra chất lượng và khả năng quản lý. Bởi vì, các bộ Load Balancer được triển khai như những thiết bị đầu cuối của một server farm. Nó cũng bảo vệ các server từ các user và đồng thời cải tiến việc bảo mật.
- ❖ Việc cân bằng tải được dựa trên thông tin của những gói IP hoặc nội dung trong những request ứng dụng. Các bộ Load Balancer tạo ra sự xác về lưu lượng tới các trung tâm dữ liệu, tới các server, các tường lửa, tới caches hoặc các ứng dụng.

1.1.2. Lợi ích về Load Balancing:

- ❖ Nhu cầu cần về Load Balancing thường có ở hai mảng lớn là server và mạng. Với việc xây dựng mạng Internet và Interanet, những mạng kết nối từ những server đến các máy tính nhân viên, khách hàng hoặc những nhà cung cấp.
- ❖ Để xây dựng một website cho thương mại điện tử. Ví dụ như hầu hết các thành phần là: edge routers (là những router lân cận), các switch, các firewall, các cache, các web server và các server cơ sở dữ liệu.

1.1.3. Môi trường Load Balancing

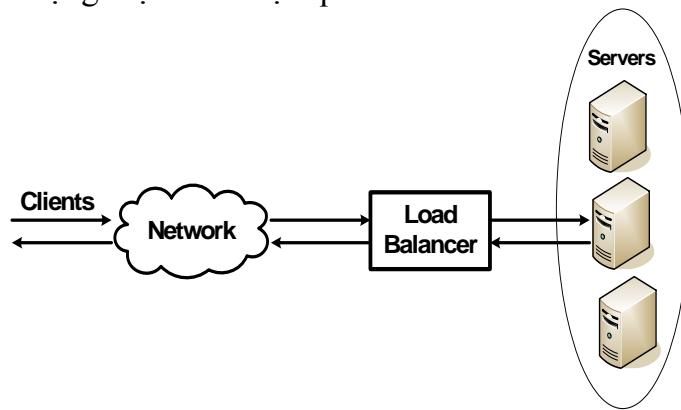
Có ít nhất hai lý do mà các công ty và các nhà cung cấp dịch vụ Internet (ISP) quan tâm đó là:

- ❖ Lý do thứ nhất là, có nhiều ứng dụng hoặc dịch vụ cần cho sự sống còn của internet như: Web, FTP (File Transfer Protocol), DNS (Domain Name Service), NFS (Network File System), email, cơ sở dữ liệu (database)...

- ❖ Lý do thứ hai là, có nhiều ứng dụng đòi hỏi có nhiều server cho mỗi ứng dụng, bởi vì một server không cung cấp đủ công suất.

1.1.4. Các ứng dụng của Load Balancing

- ❖ Với thời đại Internet hiện nay thì mạng giữa vai trò là tầng trung tâm. Như Internet kết nối cả thế giới lại với nhau còn intranet trở thành trụ cột cho các hoạt động kinh doanh. Cơ sở hạ tầng công nghệ thông tin có thể hiểu là các loại thiết bị như: máy tính với chức năng là một client hay server, và các switch/router thì kết nối các máy tính lại với nhau. Và các bộ Load Balancer là cầu nối giữa các server và mạng, được minh họa như hình dưới đây. Các bộ Load Balancer hiểu được các giao thức lớp cao hơn. Dó đó, chúng có thể giao tiếp với các server một cách thông minh. Mặt khác, các bộ Load Balancer có thể hiểu được các giao thức mạng, vì vậy chúng có thể tích hợp vào mạng một cách hiệu quả.



Hình 1.1: Server farm với một bộ Load Balancer

- ❖ Bộ Load Balancer có ít nhất 4 ứng dụng lớn sau:
 - Server load balancing: thực hiện phân đều bằng cách phân phối tải thông qua nhiều server sao cho sự phân chia là tỷ lệ với khả năng của một server và để có thể chịu đựng được việc lỗi trên một server.
 - Global server load balancing: thực hiện phân đều bằng cách chuyển hướng từ các users đến các trung tâm dữ liệu gồm những server farm.
 - Firewall load balancing: phân phối tải thông qua nhiều tường lửa sao cho tỷ lệ với khả năng của tường lửa và để có thể chịu đựng được khi một tường lửa bị lỗi.
 - Transparent cache switching: chuyển hướng lưu lượng tới cache để tăng tốc độ thời gian đáp ứng cho những client hoặc nâng cao hiệu suất của một web server bằng cách sử dụng nội dung cache tĩnh.

1.1.5. Các sản phẩm load-balancing

- ❖ Các sản phẩm load-balancing có nhiều dạng khác nhau. Chúng có thể được chia thành 3 loại: những sản phẩm phần mềm, các thiết bị và các switch, được miêu tả dưới đây:
 - Các sản phẩm load-balancing phần mềm: chạy trên những server cân bằng tải của chúng. Các sản phẩm này thực hiện những giải thuật xử lý phân phối tải giữa chúng.
 - Các thiết bị: là những sản phẩm black-box. Nó bao gồm phần cứng và phần mềm cần thiết để thực hiện chuyển mạch Web. Hộp này đơn giản là một PC hay một server, được đóng gói thành một hệ điều hành chuyên dụng cùng với phần mềm hoặc một hộp riêng với phần cứng và phần mềm tùy chọn.
 - Các switch: mở rộng những chức năng của một switch lớp 2/3 thành những lớp cao hơn bằng cách sử dụng một vài phần cứng và phần mềm.
- ❖ Load balancing đảm nhận chức năng server hay chức năng của một switch, thì điều đó không quan trọng mà quan trọng là những sản phẩm load-balancer hoặc những những kiểu sản phẩm mà chúng ta gặp nó đáp ứng yêu cầu tốt nhất của chúng ta là được. Yêu cầu đó có thể là giá thành, hiệu suất, sự tin cậy, tính có thể điều khiển và tính bảo mật.

1.2. Cách tiếp cận cơ bản về Server Load Balancing

1.2.1. Tổng quan Web Server

Khi một user gõ vào URL (Uniform Resource Locator) <http://www.nguyễnhồngthai.hcmut.edu.vn> trong trình duyệt web, thì có hầu hết mọi thứ xảy ra đằng sau nó để cho người dùng có thể xem được trang web. Thật hữu ích để hiểu được những cơ sở này, ít nhất là trong một form đơn giản, trước khi chúng ta sẽ tiến đến tìm hiểu về load balancing.

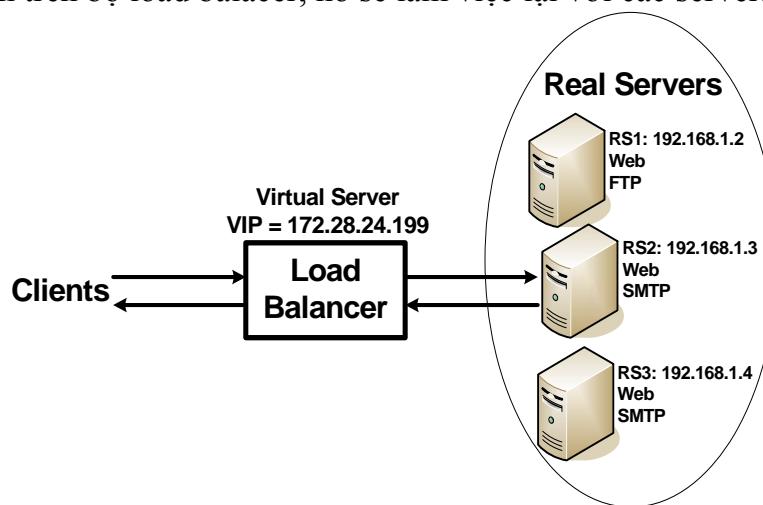
Đầu tiên, trình duyệt web sẽ phân giải tên www.nguyễnhồngthai.hcmut.edu.vn thành địa chỉ IP bằng cách hỏi DNS cục bộ (Domain Name Server). Một DNS cục bộ được cài đặt bởi nhà quản trị mạng và được cấu hình trên những máy tính của user. DNS cục bộ sử dụng giao thức DNS để tìm DNS tin cậy cho www.nguyễnhồngthai.hcmut.edu.vn mà nó đã đăng ký tại nó trong những hệ thống DNS internet cũng như sự tin cậy cho www.nguyễnhồngthai.hcmut.edu.vn. Một lần, DNS cục bộ tìm địa chỉ IP cho www.nguyễnhồngthai.hcmut.edu.vn từ DNS tin cậy, nó sẽ trả về cho trình duyệt của user. Sau đó, trình duyệt sẽ thiết lập một kết nối TCP đến host hoặc server đã chỉ xác nhận bởi địa chỉ IP tìm được ở trên, và đi theo đó với một HTTP (Hypertext Transfer Protocol) yêu cầu gửi trang web www.nguyễnhồngthai.hcmut.edu.vn. Server sẽ trả về nội dung trang web cho theo danh sách URL tới những đối tượng như hình ảnh là thành phần

của trang web. Trình duyệt sau đó sẽ lấy được các thành phần của trang web và tập hợp toàn bộ trang web để hiển thị cho user.

1.2.2. Server Farm với một bộ Load Balancer

Nhiều nhà quản trị server thích triển khai trên nhiều server với mục đích sẵn sàng và cân bằng. Để khi một server bị down thì có server sẽ thay thế và như vậy mạng vẫn còn hoạt động được trong lúc đó, server bị lỗi sẽ được sửa chữa lại.

Bây giờ, ta sẽ xét ví dụ dưới đây, một load balancer được triển khai với nhiều server, và sự kế thừa kết nối (được minh họa như Hình 1.2). Bộ load balancer được đặt đầu tiên trước của server farm. Còn tất cả các server hoặc là đi tới bộ load balancer hoặc chỉ tới switch thôi. Bộ load balancer cùng với các server, nó sẽ làm việc với client với một địa chỉ IP ảo (VIP) và VIP sẽ cấu hình trên bộ load balancer, nó sẽ làm việc lại với các server.



Hình 1.2: Servser farm với một bộ Load Balancer

Để truy cập tới bất kỳ ứng dụng nào trên server, thì địa chỉ client sẽ gửi request đến VIP. Ví dụ trong trường hợp Web site <http://www.nguyenhongthai.hcmut.edu.vn>, ban đầu nó sẽ phân giải địa chỉ <http://www.nguyenhongthai.hcmut.edu.vn> thành địa chỉ IP, mà IP này chính là VIP. Do đó, VIP sẽ nhận các request từ client và nó sẽ phân phối chúng tới các real server đã sẵn sàng. Bằng cách triển khai load balancer, chúng ta tức thời đạt được hầu như các lợi ích sau:

- ❖ *Scalability (tính cân đối):* bởi vì bộ load balancer phân phối các request của client tới tất cả các server thực sẵn sàng, làm cho khả năng xử lý trên server ảo sẽ lớn hơn khả năng của một server. Bộ load balancer dùng giả thuật phân phối tải để phân phối các request client giữa các server thực. Nếu giải thuật tốt thì khả năng của server ảo sẽ bằng toàn thể khả năng xử lý của tất cả các server thực. Tuy nhiên, ngay cả nếu

khả năng server ảo khoảng $80 \rightarrow 90\%$ khả năng của toàn bộ các server thực thì điều này cũng là tốt rồi.

- ❖ *Avaibability (tính sẵn sàng):* bộ load balancer giám sát một cách liên tục tình trạng của các server thực và các ứng dụng đang chạy trên chúng. Nếu một server thực hoặc ứng dụng nào bị lỗi thì bộ load balancer sẽ ngăn việc gửi request client tới server đó. Mặc dù, kết nối vẫn tồn tại và request đang xử lý bởi server lỗi nhưng bộ load balancer sẽ chuyển tất cả các request tới các server nào còn tốt. Ngược lại, nếu chúng ta không có bộ load balancer thì nó phải gửi lại tool giám sát mạng để kiểm tra tình trạng của server hoặc ứng dụng, rồi mới chuyển client tới một server khác. Bởi vì, bộ load balancer làm một cách thông suốt trên hệ thống đang chạy, nên việc rớt mạng (downtime) sẽ được giảm thiểu. Hơn nữa, server lỗi có thể được sửa chữa lại, bộ load balancer kiểm tra sự thay đổi tình trạng mạng và bắt đầu forward những request tới server.
- ❖ *Manageability (tính quản lý):*
 - Nếu một phần cứng của server cần phải nâng cấp, hoặc là hệ điều hành hoặc là phần mềm ứng dụng phải được nâng cấp thành phiên bản mới hơn, thì server đó phải dừng hoạt động lại. Mặc dù, nâng cấp có thể được xếp lịch trình giờ chết để giảm thiểu ảnh hưởng của thời gian chết (downtime), nhưng vẫn là có downtime. Đối với một số công việc kinh doanh thì điều này không cho phép downtime. Một vài thứ không thực sự nhận ra off-peak-hours (giờ chết), đặc biệt nếu server được truy cập bởi user trên toàn thế giới trong những miền thời gian khác nhau. Nhưng nếu triển khai với một bộ load balancer thì khác. Với load balancer, chúng ta có thể đưa server vào tình trạng offline một cách thông suốt mà vẫn duy trì tình trạng mạng tức sẽ không xảy ra tình trạng downtime. Các bộ load balancer có thể thực hiện việc shutdown các server tao nhã hơn nhờ đó bộ load balancer sẽ dừng việc gửi các request mới tới các server đó và chờ các kết nối hiện thời kết thúc. Hơn nữa, các kết nối sẽ đóng lại, server có thể offline để duy trì hoạt động của hệ thống mạng một cách an toàn. Điều này sẽ hoàn toàn liên tục đến các client. Cũng như, bộ load balancer sẽ tiếp tục phục vụ các request mà địa chỉ trỏ đến VIP bằng sự phân phối chúng trên các server thực còn lại.
 - Các bộ load balancer cũng giúp một cách dễ quản lý bằng cách tách riêng các ứng dụng từ server. Ví dụ, giả sử chúng ta có 10 server thực sẵn sàng và chúng ta cần chạy 2 ứng dụng: Web (HTTP), và File Transfer Protocol (FTP). Và giả sử chúng ta chọn 2 server làm FTP server còn lại 8 server sẽ làm Web server. Với không có bộ load balancer, chúng ta sẽ phải dùng đến DNS để thực hiện xoay vòng 2

địa chỉ IP server cho FTP và giữa 8 địa chỉ IP cho Web Server. Nếu việc yêu cầu FTP bỗng nhiên tăng lên, và chúng ta cần phải chạy trên server khác, bây giờ ta phải sửa lại DNS để thêm địa chỉ server thứ 3. Điều này mất nhiều thời gian, và không thể giải phóng sự thực hiện bằng cách này. Nếu chúng ta dùng một bộ load balancer, chúng ta chỉ cần khai báo một VIP (IP ảo). Chúng ta có thể cấu hình bộ load balancer để kết hợp VIP với các server 1 và 2 cho FTP, và các server 3 → 8 cho các ứng dụng web. Điều này cũng giống như *binding* (*binding* có nghĩa là liên kết). Tất cả các yêu cầu FTP được nhận với port FTP thông dụng là port 21. Bộ load balancer sẽ tổ chức kiểu yêu cầu dựa trên port TCP đích và chuyển nó tới server thích hợp. Nếu yêu cầu FTP tăng lên, chúng ta có thể cho phép server 3 chạy chương trình FTP và *bind* server 3 tới VIP cho ứng dụng FTP. Bây giờ, load balancer tổ chức rằng có 3 server chạy FTP và phân phối các yêu cầu giữa 3 server đó, do đó sự tăng tốc thời khắc năng xử lý toàn thể cho các FTP request. Bất kỳ sự di chuyển ứng dụng từ một server đến server khác hoặc thêm nhiều server cho một ứng dụng cho trước, với không cần ngắt server (interruption) tới các client. Nó thật sự là một công cụ mạnh cho các nhà quản trị server.

- Các bộ load balancer cũng giúp quản lý số lượng lớn nội dung, được hiểu như là *content management* (tạm dịch là quản lý nội dung). Một vài web server có thể có nhiều nội dung để phục vụ mà điều đó thì không thể làm với một server. Chúng ta có thể tổ chức các server thành nhiều nhóm khác nhau, trên đó mỗi nhóm server sẽ đáp ứng chỉ một phần nội dung, và bộ load balancer sẽ chuyển các request đó tới nhóm thích hợp dựa vào URL trong HTTP request.
- Người sử dụng load balancer không cần biết hệ điều này nó là gì bởi vì chúng làm việc dựa trên các giao thức mạng chuẩn. Các bộ load balancer có thể phân phối tải tới bất kỳ server nào trong hệ thống server. Điều này cho phép người quản trị hòa hợp (mix and match) các server khác nhau và còn mang lại lợi ích của mỗi server để cân đối toàn thể khả năng xử lý.
- ❖ *Security (tính bảo mật)*: bởi vì các bộ load balancer nằm ở trước server farm, các bộ load balancer có thể bảo vệ các user từ những user có tình làm hại. Nhiều sản phẩm load-balancing xuất phát từ những đặc trưng bảo mật, nó cho dừng các cuộc tấn công vươn tới những server. Những server thực có thể cũng được cấp những IP riêng (IP cục bộ), để khóa bất kỳ hướng truy cập bởi user bên ngoài. Những địa chỉ IP riêng không được định tuyến trên Internet. Bất kỳ người nào trên Internet đều phải đi qua một thiết bị mà nó thực hiện chuyển đổi địa chỉ mạng (Network

Address Translation hay gọi tắt là NAT), để giao tiếp với một host có địa chỉ IP riêng. Bộ load balancer có thể là các thiết bị trung gian thực hiện NAT như giữa vai trò phân phối và forwarding các request client tới các server thực khác. VIP trên load balancer có thể là một địa chỉ IP public (IP toàn cục), để người dùng Internet có thể truy cập được VIP. Nhưng các server thực ở đằng sau bộ load balancer có thể có các địa chỉ IP riêng để có thể tác động đến tất cả các giao tiếp đến từ load balancer.

- ❖ *Quality of Service* (viết tắt là QoS, tạm dịch là chất lượng dịch vụ): QoS có thể được định nghĩa với nhiều cách khác nhau. Nó có thể được định nghĩa như thời gian đáp ứng của server hoặc ứng dụng, sự sẵn sàng của một dịch vụ ứng dụng cho trước, hoặc khả năng cung cấp của những dịch vụ khác nhau dựa vào kiểu user. Ví dụ, một website nó cung cấp một thông tin của một chương trình làm việc liên tục (frequent-flier program), có thể là muốn cung cấp thời gian đáp ứng tốt nhất tới những bộ phận bạch kim của hơn là thành viên vàng hoặc thành viên bạc của nó. Load balancer có thể được dụng để phân biệt những user dựa vào một vài thông tin của packet yêu cầu, và chuyển chúng đến một server hoặc một nhóm các server hoặc đặt một số bit riêng vào gói IP để cung cấp cho lớp dịch vụ mà mình muốn.

1.2.3. Luồng packet cơ bản trong load balancing

Việc xử lý cấu hình một load balancer đơn giản thường sẽ qua các bước sau đây:

- Định nghĩa một VIP trên load balancer: VIP = 172.28.24.199.
- Xác định các ứng dụng mà chúng ta cần cân bằng tải: Web, FTP, và SMTP...
- Cho mỗi ứng dụng, bind VIP tới mỗi server thực.
- Cấu hình kiểm tra tình trạng mạng để load balancer dùng nó để xác định điều kiện của server và ứng dụng.
- Cấu hình những phương pháp phân phối tải.

Ví dụ: Luồng packet trong trường hợp Hình 1.3 dưới đây:

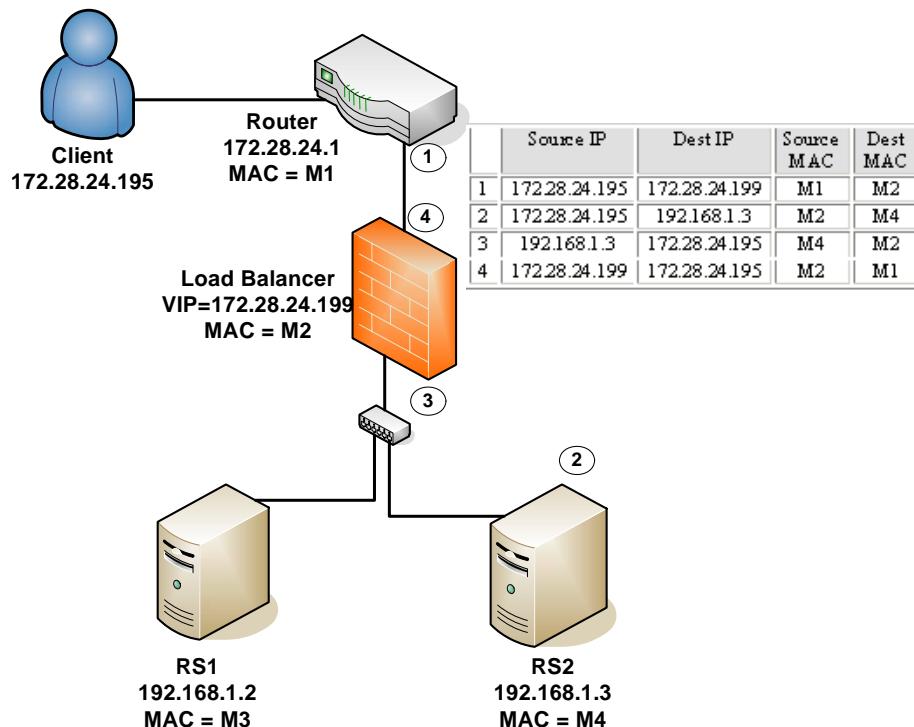
Giả thuyết: Source IP là địa chỉ IP của client, tức 172.28.24.195, còn Source port là số port dùng bởi client cho kết nối TCP. Destination IP là địa chỉ của VIP, tức 172.28.24.199, mà nó sẽ làm việc với server farm cho ứng dụng Web còn Dest port sẽ là port chuẩn(port 80). Và như vậy packet trên sẽ được định tuyến như sau:

- Bước 1: Gói tin với thông điệp request, sẽ có địa chỉ nguồn và địa chỉ đích như giả thuyết trên, còn port nguồn là M1 và port đích là M2. Như vậy gói tin đến được load balancer.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 1: TỔNG QUAN VỀ SERVER LOAD BALANCING

- Bước 2: Tại đây, load balancer sẽ làm nhiệm vụ đổi địa chỉ đích từ 172.28.24.199 → 192.168.1.3 và đồng thời đổi Source MAC = M2, Destination MAC = M4. Và gói tin sẽ đến được RS2. Đến đây, RS2 đọc request và sẽ kết quả ứng dụng (reponse).
- Bước 3: Gói tin kết quả sẽ có Source IP = 192.168.1.2, Source MAC = M4, Destination IP = 172.28.24.195, Destination MAC = M2. Như vậy, gói reponse sẽ về tới load balancer.
- Bước 4: Load balancer sẽ thực hiện đổi gói tin với địa chỉ đích giữ nguyên còn địa chỉ nguồn là 172.28.24.199 và Source MAC=M2, Destination MAC=M1. Đến đây, client nhận được kết quả.



Hình 1.3: Luồng packet trong mô hình load balancing đơn giản

1.2.4. Network-Address-Translation

NAT là khái niệm xây dựng cơ bản trong load balancing. Load balancer chủ yếu dùng NAT để chuyển request tới các server thực. Có nhiều kiểu NAT khác nhau. Khi load balancer thay đổi địa chỉ IP đích từ VIP thành IP của server thực, nó được hiểu như là *destination NAT*. Khi server thực gửi đáp ứng lại thì load balancer phải chuyển địa chỉ của server thực thành VIP. Nó ghi nhớ điều rằng việc chuyển địa chỉ IP này xảy ra ở địa chỉ nguồn của packet, khi việc gửi ngược lại từ server về client. Để giữ điều này cách đơn giản là chỉ cần load balancer thực hiện NAT ngược lại. Và client nhận biết được điều này vì địa chỉ nguồn của việc gửi ngược lại là VIP. Có 3 phần mà

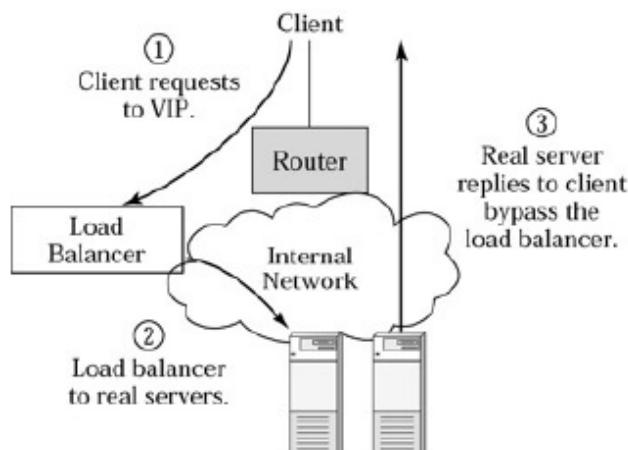
chúng ta cần chú ý để hiểu được NAT ở load balancing: địa chỉ MAC, địa chỉ IP, và số port TCP hay UDP.

❖ **Destination NAT**

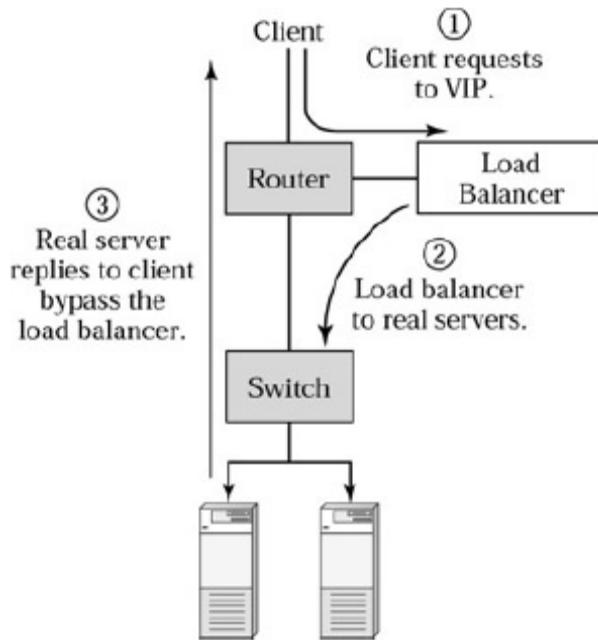
Việc xử lý chuyển đổi địa chỉ đích trong những gói tin được xem như là *destination NAT*. Hầu như, các bộ load balancer đều thực hiện destination NAT mặc định. Hình 1.3 trình bày cách làm việc destination NAT, và nó là một phần của load balancing. Mỗi gói tin đều có địa chỉ nguồn và địa chỉ đích. Do đó, destination NAT chỉ đề cập đến sự chuyển đổi địa chỉ nguồn và thỉnh thoảng nó được xem là *half-NAT*.

❖ **Source NAT**

- Nếu bộ load balancer chuyển đổi địa chỉ nguồn trong những gói tin cùng với chuyển đổi địa chỉ đích thì nó được xem như source NAT. Thỉnh thoảng nó được xem như là *full-NAT*, điều này cải tiến việc chuyển đổi cả địa chỉ nguồn lẫn địa chỉ đích. Source NAT không được sử dụng phổ biến nếu không có một topo mạng đặc biệt đòi hỏi source NAT. Nếu topo mạng sử dụng cách này để gửi packet ngược lại từ server thực thì có thể bỏ qua load balancer và như vậy source NAT phải được thực hiện. Hình 1.4 minh họa một ví dụ về sự trình bày high-level (cấp cao) của một topo mạng. Hình 1.5 trình bày việc thiết kế mạng đơn giản, mà nó đòi hỏi sử dụng source NAT. Bằng cách sử dụng source NAT trong thiết kế này, chúng ta đã tác động việc server gửi ngược lại với lưu lượng thông qua load balancer. Trong những thiết kế cụ thể, có thể có cả 2 luân phiên sử dụng source NAT. Sự luân phiên này hoặc là sử dụng *Direct Server Return* hoặc là cài đặt load balancer như gateway mặc định cho những server thực. Cả hai luân phiên đòi hỏi rằng load balancer và các server thực phải cùng broadcast domain hay domain lớp 2. Phần Direct Server Return sẽ được trình bày chi tiết trong phần sau dưới phần này.
- Khi cấu hình thực hiện source NAT, load balancer chuyển đổi địa chỉ nguồn trong tất cả các gói tin thành một địa chỉ đã định nghĩa trên load balancer, xem như *source IP*, trước khi forward các gói tin tới các server thực, xem Hình 1.6. Địa chỉ IP nguồn có thể giống như VIP hoặc khác thuộc vào sản phẩm load-balancing mà chúng ta sử dụng. Khi server nhận được các packet, nó xem request client là load balancer bởi vì sự chuyển đổi địa chỉ nguồn. Server thực bây giờ không biết địa chỉ IP nguồn của client thực sự là ai. Server thực gửi ngược lại tới load balancer, sau đó nó chuyển đổi địa chỉ đích thành địa chỉ IP của client thực sự.



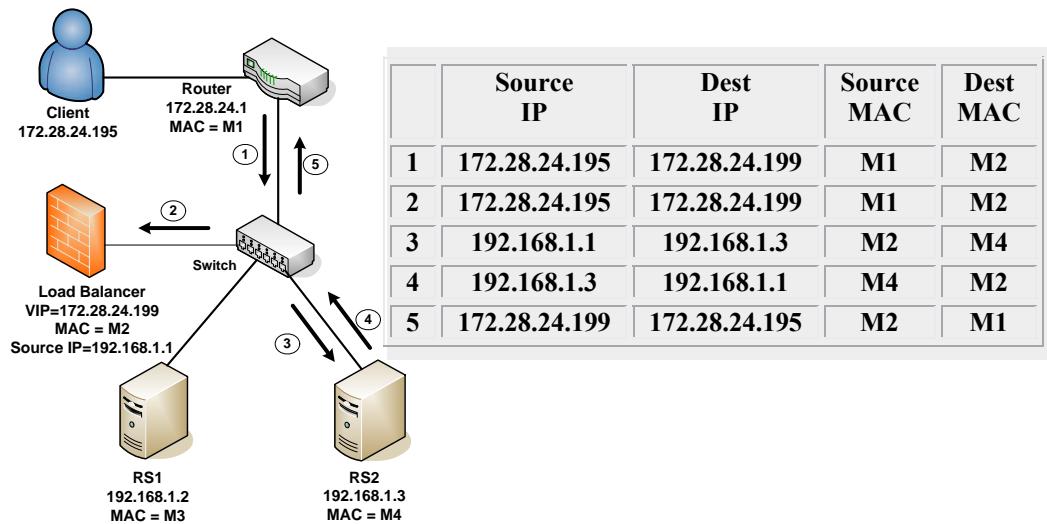
Hình 1.4: Sự trình bày cấp cao của một topo mạng đòi hỏi dùng Source NAT



Hình 1.5: Trình bày việc thiết kế mạng đơn giản đòi hỏi dùng Source NAT

- Ưu điểm của source NAT là cho phép bạn triển khai các bộ load balancer tại bất cứ nơi đâu, không bất kỳ giới hạn nào trên topo mạng.
- Nhược điểm của nó là không thấy được địa chỉ IP client gốc bởi vì load balancer đã chuyển địa chỉ IP nguồn. Một vài ứng dụng nó trả lời về sự xác nhận dựa vào địa chỉ IP nguồn mà nó sẽ bị lỗi nếu source NAT được sử dụng. Nhiều nhà quản trị web site cũng trả về những web server log để xác định sơ lược user dựa vào địa chỉ IP nguồn và do đó có thể xem như không sử dụng source NAT. Một vài sản phẩm load balancer

có thể có thể gửi thẳng mối quan tâm này bằng cách cung cấp những sự lựa chọn để ghi hoặc báo cáo địa chỉ IP nguồn của những request đi vào.



Hình 1.6: Luồng packet với source NAT

❖ Reverse NAT

Khi dùng một bộ load balancer, những server thực thường gán những địa chỉ IP riêng (IP cục bộ) để tăng tính bảo mật và bảo tồn địa chỉ IP. Load balancer thực hiện destination NAT cho tất cả các lưu lượng đã thiết lập bởi những client tới những server thực. Những server thực ở đằng sau load balancer cần thiết lập kết nối ra bên ngoài, chúng phải đi thông qua NAT bởi vì các server thực có các địa chỉ IP cục bộ. Load balancer có thể được cấu hình để thực hiện *reverse NAT* (tạm dịch là NAT ngược) nơi mà load balancer thay đổi địa chỉ IP nguồn đối với tất cả các lưu lượng đã thiết lập bởi những server thực ra bên ngoài. Load balancer thay đổi địa chỉ IP của những server thực thành địa chỉ IP toàn cục (public IP) mà nó được định nghĩa trên load balancer. Địa chỉ IP toàn cục có thể giống như địa chỉ IP ảo định nghĩa trên load balancer để thực hiện cân bằng tải, hoặc nó có thể là một địa chỉ IP cách biệt đặc biệt đã được cấu hình để sử dụng trong *reverse NAT* phụ thuộc vào sản phẩm load-balancing cụ thể.

❖ Port-Address Translation

- Việc chuyển đổi địa chỉ port (PAT) xem như chuyển đổi số port trong những gói TCP/UDP, mặc dù số port có thể được sử dụng trong những giao thức khác. PAT là sự thừa kế trong những bộ load balancer. Khi chúng ta kết hợp port 80 vào VIP thành port 1000 trên một server thực, bộ load balancer sẽ chuyển số port và forward những request tới port 1000 trên server thực. PAT chú trọng chủ yếu 3 lý do sau: bảo mật, cân bằng ứng dụng, quản lý ứng dụng.

- Bằng cách chạy những ứng dụng riêng trên những port riêng, nó có thể được sự bảo mật tốt nhất cho những server thực bằng cách đóng các port phổ biến của chúng. Ví dụ ta có thể chạy Web Server trên port 8080, và có thể bind port 80 của VIP trên load balancer thành port 8080 trên các server. Client không thấy được sự khác biệt này, và nó xem như web server vẫn tiếp tục gửi những web request tới port 80 của VIP. Load balancer sẽ chuyển số port của tất cả request và forward chúng tới port 8080 trên các server. Bây giờ, một ai đó không thể tấn công trực tiếp đến server thực bằng cách gửi những luồng độc hại tới port 80, bởi vì nó đã đóng port này rồi. Mặc dù, hacker có thể không quá khó để tìm ra những port mở, nhưng với việc thực hiện PAT thì sẽ tạo thêm một chút khó khăn cho những hacker. Thường một vài thức được làm để tăng tính bảo mật của một web site hoặc một server farm.
- Việc gán địa chỉ IP cục bộ tới các server thực, hoặc bắt buộc các danh sách điều khiển truy cập từ chối các luồng truy cập đến địa chỉ IP server thực. Như vậy, nó bắt buộc tất cả các user đều phải thông qua load balancer để có thể truy cập đến các server thực. Sau đó, load balancer có thể tác động những chính sách truy cập nào đó và cũng bảo vệ được các server chống lại những kiểu tấn công nào đó.
- PAT giúp cải thiện cân bằng bằng cách cho phép chúng ta chạy cùng ứng dụng trên nhiều port bởi những ứng dụng nào đó được thiết kế, chúng ta có thể cân chỉnh việc thực hiện ứng dụng bằng cách chạy nhiều bản copy. Phụ thuộc vào ứng dụng, việc chạy nhiều bản copy có thể dùng nhiều CPU. Ví dụ, chúng ta có thể chạy Microsoft IIS (Internet Information Server - Microsoft's Web-server software) trên nhiều port. Chúng ta có thể chạy IIS trên port 80, 81, 82 và 83 trên mỗi server thực. Như vậy, chúng ta cần bind port 80 của VIP tới mỗi port chạy IIS. Load balancer sẽ phân phối luồng không chỉ qua các server thực mà còn giữa các port trên mỗi server.
- PAT cũng có thể cải tiến tính quản lý trong những tình huống nào đó. Ví dụ, khi chúng ta làm một số web site chung cho trên các server, chúng ta có thể dùng một VIP để quảng bá tất cả web site domain. Load balancer nhận tất cả web request trên port 80 cùng VIP. Chúng ta có thể chạy ứng dụng web server trên port khác cho mỗi web site domain. Do đó, web server cho www.nguyễnhồngthai.hcmut.edu.vn chạy trên port 80 và www.nghiencuuloadbalancing.hcmut.edu.vn chạy trên port 81. Load balancer có thể được cấu hình để gửi những luồng tới những port thích hợp, phụ thuộc vào tên miền trong URL của mỗi HTTP request.

1.2.5. Direct Server Return

- ❖ Trong phần thảo luận về load balancing trên thì tất cả những luồng trả về từ những server thực về load balancer. Nếu không, chúng ta thường sử dụng source NAT để bắt buộc việc gửi lại thông qua load balancer. Load balancer xử lý những request những reply gửi từ server thực đến client. *Direct Server Return* (DSR) cải tiến luồng reply server bỏ qua bộ load balancer. Bằng cách bỏ qua load balancer, chúng ta đã đạt được việc thực hiện tốt hơn nếu bộ load balancer bị thắt cổ chạy, bởi vì bây giờ load balancer chỉ phải xử lý request, đột ngột cắt giảm số gói xử lý. Để bỏ qua load balancer khi reply về, chúng ta cần làm một số việc bỏ NAT. Để dùng *Direct Server Return*, load balancer không cần chuyển đổi địa chỉ IP trong những request, để những luồng reply không cần phải NAT và do đó bỏ qua load balancer.
- ❖ Khi cấu hình thực hiện *Direct Server Return*, load balancer chỉ chuyển đổi địa chỉ MAC đích trong những gói request như địa chỉ IP đích vẫn còn giống như VIP. Để những request tới được các server thực dựa trên địa chỉ MAC đích, thì những server thực phải cùng domain lớp 2 với load balancer. Chỉ một lần, server thực đã nhận được gói tin, chúng ta phải làm cho các server thực nhận được nó mặc dù địa chỉ IP đích là VIP, không phải là địa chỉ của server thực. Do đó, VIP phải được cấu hình giống như địa chỉ *loopback IP* trên mỗi server thực. Địa chỉ IP loopback là một interface cục bộ có sẵn trên mỗi TCP/IP host. Nó thường được gán là 127.x.x.x, trong đó x.x.x có thể bất kỳ. Một host có thể có nhiều địa chỉ IP loopback gán như 127.0.0.1, 127.0.0.10, và 127.16.12.84. Số của địa chỉ IP loopback hỗ trợ phụ thuộc vào hệ điều hành.
- ❖ Giao thức phân giải địa chỉ (Address Resolution Protocol hay viết tắt ARP) được sử dụng trong mạng Ethernet để tìm ra địa chỉ IP host và địa chỉ MAC. Bằng cách định nghĩa, loopback interface không đáp ứng về ARP request. Do đó, không có người nào trong mạng biết được địa chỉ loopback IP trong một host, nó hoàn toàn trong host. Chúng ta có thể gán bất kỳ địa chỉ đến địa chỉ loopback; điều đó có nghĩa là địa chỉ IP không bắt đầu với 127. Trong khi một host không thể trả đáp ứng tới các ARP request mang địa chỉ loopback, thì nó có thể reply tới những ai đó gửi request tới địa chỉ. Vì vậy, không ai bên ngoài có thể biết được những gì mà địa chỉ loopback IP đã định nghĩa trên host đó. Nếu chỉ đó là không cần định nghĩa thì host có thể nhận request và reply lại cho nó. *Direct Server Return* sử dụng giả thuyết này để chống lại destination NAT trên những luồng request, bây giờ thì server thực nhận được những request bằng cách định nghĩa VIP như địa chỉ loopback trên các server.
- ❖ Hình 1.7 trình bày cách mà một luồng gói tin đi khi dùng *Direct Server Return*. Đầu tiên, load balancer bỏ qua IP đích xem như VIP trong những

gói request nhưng đổi địa chỉ MAC đích thành địa chỉ server được chọn. Vì switch nằm giữa load balancer và server thực là switch lớp 2, nó đơn giản là forward những gói tin tới server bên phải dựa vào địa chỉ MAC đích. Server thực nhận gói tin, bởi vì địa chỉ IP đích của gói tin, VIP, được định nghĩa là địa chỉ loopback IP trên server. Khi server trả lại, VIP trở thành IP nguồn và IP của client trở thành địa chỉ đích. Gói tin được forward thông qua switch lớp 2 tới router, và sau đó tới client, không cần bất kỳ NAT nào trong việc trả lại. Vì vậy, chúng ta vừa bỏ qua load balancer một cách thành công cho luồng trả lại.

- ❖ Bây giờ, chúng tôi sẽ trình bày cách định nghĩa địa chỉ loopback IP trong một cặp các hệ điều hành lớn. Ở hệ điều hành Sun Microsystem Solaris, lệnh dưới đây có thể được sử dụng để cấu hình 172.28.24.199 là địa chỉ loopback IP:

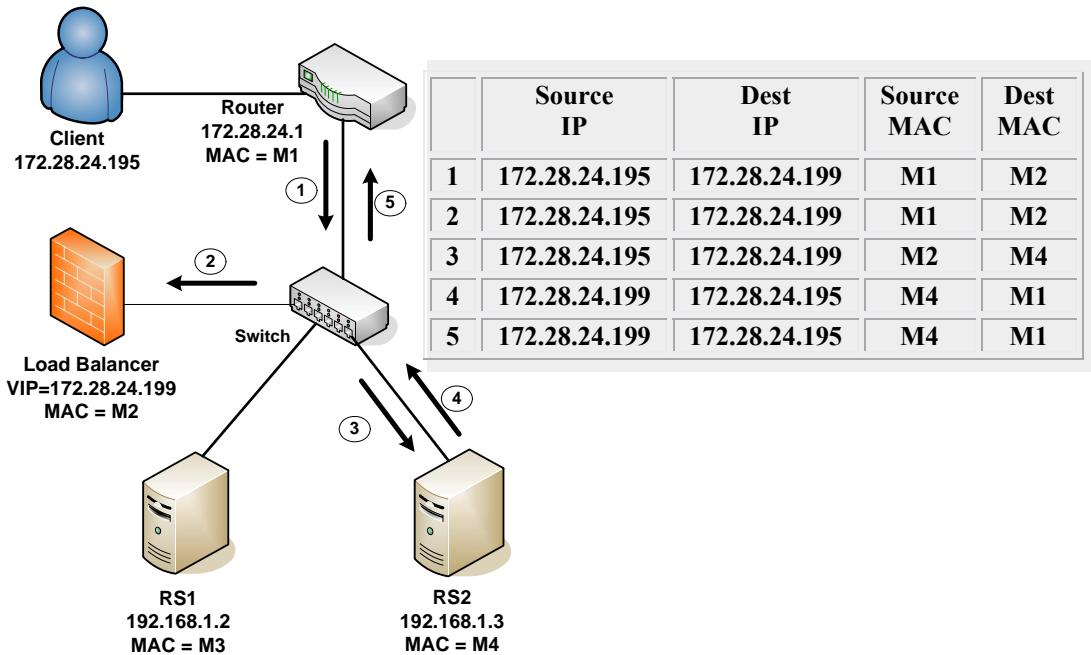
```
ifconfig lo0:1 vip addr 172.28.24.195
```

- ❖ Lệnh này áp dụng chỉ cấu hình chạy hiện hành. Để tạo địa chỉ lâu dài, thì nó phải được cấu hình sau mỗi lần reboot hoặc mở máy, bằng cách tạo ra một file “*etc/rc3.d/file_cấu_hình_loopback_IP*”, và thêm dòng lệnh trên vào file vừa tạo trên, sau đó tạo một link tới file “*/etc/init.d/file*”.
- ❖ Đối với hệ điều hành linux, lệnh dưới đây có thể dùng để cấu hình 172.28.24.199 cho địa chỉ loopback IP:

```
ifconfig lo0:0 172.28.24.199 netmask 255.255.255.0 up
```

- ❖ Lệnh này chỉ áp dụng cấu hình chạy hiện thời. Để tạo địa chỉ lâu dài thì nó phải được cấu hình sau mỗi lần reboot hoặc mở máy, bằng cách thêm dòng lệnh trên vào “*/etc/hostname.lo0:1*”.
- ❖ Direct Server Return (DSR) là một mẫu hữu dụng cho ứng dụng chuyên về thông lượng như FTP. Nếu có 20 gói tin reply cho mỗi gói request, chúng ta bỏ qua load balancer khoảng 20 gói, giảm số gói tin xử lý bởi load balancer cho mỗi request phục vụ. Điều đó có thể giúp load balancer xử lý thêm nhiều request và cung cấp cho chúng ta khả năng cao hơn.
- ❖ DSR cũng hữu ích trong cân bằng tải trường hợp đòi hỏi NAT phức tạp hoặc không hỗ trợ load balancer bởi vì DSR không cần NAT.
- ❖ DSR cũng hữu ích trong cấu hình mạng mà các gói tin không được bảo đảm đi về qua cùng load balancer.
- ❖ Bên cạnh những ưu điểm vừa trình bày trên thì load balancer có một số khuyết điểm, đó là việc xử lý trên không phải luôn luôn là có thể. Việc xử lý đòi hỏi một vài cấu hình hết sức chú ý trong phần của những server thực và những phần mềm chạy trên chúng. Việc cấu hình đặc biệt này có thể không thực hiện được với mọi hệ điều hành và mọi phần mềm server. Việc xử lý này cũng thêm phức tạp tới cấu hình, và sự phức tạp thêm vào có thể

làm cho kiến trúc mạng thêm nhiều khó khăn khi thực hiện. Cũng vậy, việc phân tích cú pháp và băm URL ở lớp 5-7 sẽ không làm việc được bởi vì việc xử lý đó đòi hỏi một đường đồng bộ dữ liệu trong và ngoài bộ load balancer. Tính bền bỉ dựa vào cookie không làm việc trong mọi tình huống mặc dù nó có thể thực hiện được.



Hình 1.7: Luồng gói tin khi sử dụng *Direct Server Return*

1.2.6. Tóm tắt:

- ❖ Load balancer mang đến nhiều lợi ích khổng lồ bằng cách cải tiến server farm về các tính chất như: tính sẵn sàng, khả năng cân bằng, tính quản lý và tính bảo mật. Cân bằng tải server là ứng dụng phổ biến nhất cho những load balancer. Các bộ load balancer có thể thực hiện những cuộc kiểm tra tình trạng mạng khác nhau để đảm bảo server, ứng dụng và nội dung phục vụ trong những điều kiện tốt. Có nhiều giải thuật cân bằng tải để cân bằng tải qua nhiều kiểu server khác nhau để được sự cân bằng lớn nhất và khả năng xử lý toàn thể.
- ❖ NAT lập ra những nguyên tắc cho việc xử lý của load balancer. Có nhiều kiểu NAT khác nhau, như destination NAT và source NAT, tạo ra sự đa dạng trong thiết kế mạng với các bộ load balancer. *Direct Server Return* giúp các ứng dụng cân bằng tải với những đòi hỏi NAT phức tạp mà không cần destination NAT.

Chương 2 : HỆ ĐIỀU HÀNH UNIX

Chương này sẽ sơ lượt giới thiệu về hệ điều hành Unix, cũng như linux. Đồng thời cũng đưa ra các lệnh cơ bản được ứng dụng để quản lý hệ thống. Mục đích của chương này là tạo kiến thức nền trước khi bước vào cài đặt và cấu hình những phần mềm cụ thể ở những chương sau như cài đặt cấu hình Web Server, FTP Server ...

2.1. Giới thiệu lịch sử phát triển của Unix và Linux:

2.1.1. Lịch sử và các chuẩn UNIX:

Giữa năm 1960, AT&T Bell Laboratories và một số trung tâm khác tham gia vào một dự án là cố gắng tạo ra một hệ điều hành mới được đặt tên là Multics. Đến năm 1969, chương trình Multics bị bãi bỏ, nhưng Ken Thompson, Dennis Ritchie, và một số đồng nghiệp của Bell Labs đã tiếp tục phát triển theo mô hình ban đầu và đặt ra một hệ thống tập tin mà sau này được phát triển thành hệ thống tập tin của UNIX. Cũng khoảng thời gian đó, ngôn ngữ lập trình C được hình thành và toàn bộ HDH UNIX được viết lại bằng C. Đây là thời điểm có thể nói là UNIX được sinh ra. UNIX là nối lái của Multics. Khoảng 1977 bản quyền của UNIX được giải phóng, cho phép HDH UNIX trở thành một thương phẩm.

2.1.2. Hai dòng UNIX :

System V của AT&T , Novell và Berkeley Software Distribution (BSD) của Đại học Berkeley.

- **System V** : Các phiên bản UNIX cuối cùng do AT&T xuất bản là System III và một vài phát hành (releases) của System V. Hai bản phát hành gần đây của System V là Release 3 (SVR3.2) và Release 4.2 (SVR4.2). Phiên bản SYR 4.2 là phổ biến nhất cho từ máy PC cho tới máy tính lớn.
- **BSD** : Từ 1970 Computer Science Research Group của University of California tại Berkeley (UCB) xuất bản nhiều phiên bản UNIX, được biết đến dưới tên Berkeley Software Distribution, hay BSD. Cải biến của PDP-11 được gọi là 1BSD và 2BSD. Trợ giúp cho các máy tính của Digital Equipment Corporation VAX

được đưa vào trong 3BSD. Phát triển của VAX được tiếp tục với 4.0BSD, 4.1BSD, 4.2BSD, và 4.3BSD. Trước 1992, UNIX là tên thuộc sở hữu của AT&T. Từ 1992, khi AT&T bán bộ phận Unix cho Novell, tên Unix thuộc sở hữu của X/Open foundation. Tất cả các hệ điều hành thỏa mãn một số yêu cầu đều có thể gọi là Unix. Ngoài ra, Institute of Electrical and Electronic Engineers (IEEE) đã thiết lập chuẩn "An Industry-Recognized Operating Systems Interface Standard based on the UNIX Operating System." Kết quả cho ra đời POSIX.1 (cho giao diện C) và POSIX.2 (cho hệ thống lệnh trên Unix)

Kết lại, vấn đề chuẩn hóa UNIX vẫn còn rất xa kết quả cuối cùng. Nhưng đây là quá trình cần thiết có lợi cho sự phát triển của ngành tin học nói chung và sự sống còn của HDH UNIX nói riêng.

2.1.3. Lịch sử phát triển của Linux và giới thiệu các phân phối (distribution) Linux ngày nay

Linux là một HDH dạng UNIX (Unix-like operating system) chạy trên máy PC với bộ điều khiển trung tâm (CPU) Intel 80386 và các thế hệ sau đó, hay các bộ vi xử lý trung tâm tương thích như AMD, Cyrix. Linux ngày nay còn có thể chạy trên các máy Macintosh hoặc SUN Sparc. Linux thỏa mãn chuẩn POSIX.1.

Linux được viết lại toàn bộ từ con số không, tức là không sử dụng một dòng lệnh nào của Unix, để tránh vấn đề bản quyền của Unix, tuy nhiên hoạt động của Linux hoàn toàn dựa trên nguyên tắc của hệ điều hành Unix. Vì vậy nếu một người nắm được Linux, thì sẽ nắm được UNIX. Nên chú ý rằng giữa các Unix sự khác nhau cũng không kém gì giữa Unix và Linux.

Năm 1991 Linus Torvalds, sinh viên của đại học tổng hợp Helsinki, Phần Lan, bắt đầu xem xét Minix, một phiên bản của Unix, làm ra với mục đích nghiên cứu cách tạo ra một hệ điều hành Unix chạy trên máy PC với bộ vi xử lý Intel 80386.

9/1991, Linus cho ra version 0.01 và thông báo trên Internet về chương trình của mình.

1/1992, Linus cho ra version 0.12 với shell và C compiler. Linus không cần Minix nữa để recompile HDH của mình. Linus đặt tên HDH của mình là Linux.

1994, phiên bản chính thức 1.0 được phát hành.

Quá trình phát triển của Linux được tăng tốc bởi sự giúp đỡ của chương trình GNU (GNU's Not Unix), đó là chương trình phát triển các Unix có khả năng chạy trên nhiều platform. Đến hôm nay, cuối 1999, phiên bản mới nhất của Linux kernel là 2.2.5-xx, có khả năng điều khiển các máy đa bộ vi xử lý.

2.1.4.. Vấn đề bản quyền của GNU project

Các chương trình tuân theo GNU Copyleft or GPL (General Public License) có bản quyền như sau :

- ❖ Tác giả vẫn là sở hữu của chương trình của mình.
- ❖ Ai cũng được quyền bán copy của chương trình với giá bất kỳ mà không phải trả cho tác giả ban đầu.
- ❖ Người sở hữu chương trình tạo điều kiện cho người khác sao chép chương trình nguồn để phát triển tiếp chương trình.

2.1.5. Lý do sử dụng Linux ?

Linux là miễn phí (free). Đối với chúng ta hôm nay không quan trọng vì ngay WindowsNT server cũng “free”. Nhưng trong tương lai, khi chúng ta muốn hòa nhập vào thế giới, khi chúng ta muốn có một thu nhập chính đáng cho người lập trình, hiện tượng sao chép trộm phần mềm cần phải chấm dứt. Khi đó, free là một thông số rất quan trọng để chọn Linux.

Linux rất ổn định. Trái với suy nghĩ “truyền thống” là “của rẻ là của ôi”, Linux từ những phiên bản đầu tiên cách đây 5-6 năm đã rất ổn định. Ngay cả server Linux của những mạng lớn (hàng trăm máy trạm) cũng hoạt động rất ổn định.

Linux đầy đủ. Tất cả những gì chúng ta thấy ở IBM, SCO, Sun ... đều có ở Linux. C compiler, perl interpreter, shell , TCP/IP, proxy, firewall, tài liệu hướng dẫn ... đều rất đầy đủ và có chất lượng. Hệ thống các chương trình tiện ích cũng rất đầy đủ .

Linux là HDH hoàn toàn 32-bit. Như các Unix khác, ngay từ đầu, Linux đã là một HDH 32 bits. Hiện nay, Linux đã có phiên bản mới dành cho 64 bit.

Linux rất mềm dẻo trong cấu hình. Linux cho người sử dụng cấu hình rất linh động, ví dụ như độ phân giải màn hình Xwindow tùy ý, dễ dàng sửa đổi ngay cả kernel ...

**Linux chạy trên nhiều máy khác nhau từ PC 386, 486 tự lắp
cho đến SUN Sparc.**

Linux được trợ giúp. Ngày nay, với các server Linux sử dụng dữ liệu quan trọng, người sử dụng hoàn toàn có thể tìm được sự trợ giúp cho Linux từ các công ty lớn. IBM đã chính thức chào bán IBM server chạy trên Linux. Tài liệu giới thiệu Linux ngày càng nhiều, không thua kém bất cứ một HDH nào khác.

Với nguồn tài liệu phong phú, chương trình từ kernel cho đến các tiện ích miễn phí và bộ mã nguồn mở, Linux là người chúng ta đồng hành lý tưởng cho những ai muốn đi vào HDH chuyên nghiệp UNIX và công cụ tốt nhất cho công tác đào tạo trong các trường đại học.

Các phiên bản của Linux. Các phiên bản của HDH Linux được xác định bởi hệ thống số dạng X.YY.ZZ. Nếu YY là số chẵn => phiên bản ổn định. YY là số lẻ => phiên bản thử nghiệm.

Các phân phối (distribution) của Linux quen biết là RedHat, Debian, SUSE, Slakware, Caldera ... Hiện nay đã có các distro mới như Ubuntu, VietkeyLinux, VNLinux, Hacao,... Thậm chí các distro với giao diện tiếng Việt.

Chú ý phân biệt số phiên bản của hệ điều hành (Linux kernel) với phiên bản của các phân phối (ví dụ RedHat 6.0 với kernel Linux 2.2.5-15).

2.2. Hệ thống tiến trình của Linux, điều khiển các tiến trình.:

Linux là một HDH đa người sử dụng, đa tiến trình. Linux thực hiện tất cả các công việc của người sử dụng cũng như của hệ thống bằng các tiến trình (process). Do đó, hiểu được cách điều khiển các tiến trình đang hoạt động trên HDH Linux rất quan trọng, nhiều khi có tính chất quyết định, cho việc quản trị hệ thống.

➤ **Dịnh nghĩa :** Tiến trình (process) là một chương trình đơn chạy trên không gian địa chỉ ảo của nó. Cần phân biệt tiến trình với lệnh vì một dòng lệnh trên shell có thể sinh ra nhiều tiến trình.

➤ **Dòng lệnh :**

\$ nroff -man ps.1 | grep kill | more

sẽ sinh ra 3 tiến trình khác nhau.

Có 3 loại tiến trình chính trên Linux :

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

- Tiến trình với đối thoại (Interactive processes) : là tiến trình khởi động và quản lý bởi shell, kể cả tiến trình foreground hoặc background.
- Tiến trình batch (Batch processes) : Tiến trình không gắn liền đến bàn điều khiển (terminal) và được nằm trong hàng đợi để lần lượt thực hiện.
- Tiến trình ẩn trên bộ nhớ (Daemon processes) : Là các tiến trình chạy dưới nền (background). Các tiến trình này thường được khởi động từ đầu. Đa số các chương trình server cho các dịch vụ chạy theo phương thức này. Đây là các chương trình sau khi được gọi lên bộ nhớ, đợi thụ động sau các cổng xác định các yêu cầu chương trình khách (client) để trả lời. Hầu hết, các dịch vụ trên Internet như mail, Web, Domain Name Service... chạy theo nguyên tắc này. Các chương trình được gọi là các chương trình daemon và tên của nó thường kết thúc bằng ký tự “d” như named, inetd ... Ký tự “d” cuối được phát âm rời ra như “đê” trong tiếng việt.

Cách đơn giản nhất để kiểm tra hệ thống tiến trình đang chạy là sử dụng lệnh *ps* (*process status*). Lệnh *ps* có nhiều tùy chọn (option) và phụ thuộc một cách mặc định vào người login vào hệ thống. Ví dụ :

```
$ ps
PID TTY STAT TIME COMMAND
41 v01 S 0:00 -bash
134 v01 R 0:00 ps
```

cho phép hiển thị các tiến trình liên quan tới một người sử dụng hệ thống.

Cột đầu tiên là PID (Process IDentification). Mỗi tiến trình của Linux đều mang một số và các thao tác liên quan đến tiến trình đều thông qua số PID này. Gạch nối – trước bash để thông báo đó là shell khởi động khi người sử dụng login.

Để hiển thị tất cả các process, ta có thể sử dụng lệnh *ps -a*. Một người sử dụng hệ thống bình thường có thể thấy tất cả các tiến trình, nhưng chỉ có thể điều khiển được các tiến trình của mình tạo ra. Chỉ có super-user mới có quyền điều khiển tất cả các tiến trình của hệ thống Linux và của người khác. Lệnh *ps -ax* cho phép hiển thị tất cả các tiến trình, ngay cả những tiến trình không gắn liền đến có bàn điều khiển (tty). Chúng ta có thể coi các tiến trình đang chạy cùng với dòng lệnh đầy đủ để khởi động tiến trình này bằng *ps -axl*. Lệnh *man ps* cho phép coi các tham số tự chọn khác của lệnh *ps*.

❖ **Dừng một tiến trình, lệnh kill :** Trong nhiều trường hợp, một tiến trình có thể bị treo, một bàn phím điều khiển không trả lời các lệnh từ bàn phím, một chương trình server cần nhận cấu hình mới, card mạng cần thay đổi địa chỉ IP ..., khi đó chúng ta phải dừng (kill) tiến trình đang có vấn đề. Linux có lệnh **kill** để thực hiện các công tác này. Trước tiên chúng ta cần phải biết PID của tiến trình cần dừng thông qua lệnh **ps**. Xin nhắc lại chỉ có super-user mới có quyền dừng tất cả các tiến trình, còn người sử dụng chỉ được dừng các tiến trình của mình. Sau đó, ta sử dụng lệnh

```
$ kill -9 PID_cua_tien_trinh
```

Tham số –9 là gửi tín hiệu dừng không điều kiện chương trình. Chú ý, nếu chúng ta logged vào hệ thống như root, nhập số PID chính xác nếu không chúng ta có thể dừng một tiến trình khác. Nếu tiến trình init, là cha của tất cả các tiến trình khác bị dừng, hệ thống sẽ bị treo. Không nên dừng các tiến trình mà mình không biết.

Một tiến trình có thể sinh ra các tiến trình con trong quá trình hoạt động của mình. Nếu chúng ta dừng tiến trình cha, các tiến trình con cũng sẽ dừng theo, nhưng không tức thì. Vì vậy phải đợi một khoảng thời gian và sau đó kiểm tra lại xem tất cả các tiến trình con có dừng đúng hay không. Trong một số hần hữu các trường hợp, tiến trình có lỗi nặng không dừng được, phương pháp cuối cùng là khởi động lại máy.

❖ **Lệnh nohup.** Trong một số trường hợp, các tiến trình mà chúng ta sử dụng chạy rất lâu và chúng ta không thể chờ đợi nó hoàn thành để logout (xin nhắc lại, tiến trình do chúng ta gọi lên sẽ là tiến trình con của tiến trình shell của chúng ta, và nếu chúng ta logout, tiến trình shell sẽ dừng và kéo theo cả tiến trình của chúng ta). Lệnh **nohup** được dành cho các trường hợp này. Cú pháp của lệnh là :

```
$ nohup tên_tien_trinh
```

Khi đó, tiến trình sau lệnh nohup sẽ không bị dừng lại khi chúng ta logout.

❖ **Lệnh at :** Bên cạnh đó, Linux có các lệnh cho phép thực hiện các tiến trình ở các thời điểm mong muốn. Lệnh **at** cho phép thực hiện một tiến trình vào thời điểm nhập trong dòng lệnh.

```
$ at 1:23<Return>
lp /usr/sales/reports/*<Return>
echo "Files printed, BossÆ| mail boss@company.com<Return>
<^D>
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

Dấu ^D có nghĩa là cần giữ phím <Ctrl>, sau đó nhấn phím D và bỏ cả 2 phím cùng một lúc.

Sau khi chúng ta kết thúc lệnh **at**, dòng thông báo giống như sau sẽ hiện ra màn hình.

```
job 756001.a at Sat Dec 21 01:23:00 2000
```

Số 756001.a cho phép tham chiếu tới công tác (job) đó, để dùng nếu chúng ta muốn xóa job đó bởi lệnh .

```
$ at -r job_number
```

Lệnh này có thể khác với các phiên bản khác nhau. Ví dụ đối với RedHat 6.2 lệnh xóa một job là atrm job_number. Trong mọi trường hợp coi manpage để biết các lệnh và tham số cụ thể.

Chúng ta có thể dùng quy tắc chuyển hướng (redirect) để lập lịch trình cho nhiều lệnh cùng một lúc.

```
$ at 10:59 <tập_lệnh
```

trong đó, **tập_lệnh** là một tập tin dạng text có các lệnh. Để kiểm tra các tiến trình mà chúng ta đã nhập vào, dùng lệnh **at -l**

☞ **Lệnh batch** : Khác với lệnh **at** là tiến trình được thực hiện vào các thời điểm do người sử dụng chọn, lệnh **batch** để cho hệ thống tự quyết định khi nào tiến trình được thực hiện dựa trên mức độ tải của hệ thống. Các tiến trình in ấn, cập nhật dữ liệu lớn ... rất thích hợp với kiểu lệnh này. Cú pháp của **batch** như sau :

```
$ batch<Return>
lp /usr/sales/reports/*<Return>
echo "Files printed, BossÆ| mail boss@company.com<Return>
<^D>
```

Các lệnh **at** và **batch** cho phép lập kế hoạch thực hiện tiến trình một lần. Linux còn cho phép lập kế hoạch có tính chất chu kỳ thông qua lệnh **cron** (viết tắt của chronograph) và các tập tin crontabs. Chương trình **cron** được khởi động ngay từ đầu với khởi động của hệ thống. Khi khởi động, **cron** xem có các tiến trình trong hàng đợi nhập vào bởi lệnh **at**, sau đó xem xét các các tập tin crontabs xem có tiến trình cần phải thực hiện hay không rồi “đi ngủ ”. **Cron** sẽ “thức dậy” mỗi phút để kiểm tra xem có phải thực hiện tiến trình nào không. Super-user và user đều có thể đặt hàng các tiến trình sẽ được cho phép thực hiện bởi cron. Để làm điều này, chúng ta cần tạo một tập tin text theo cú pháp của cron như sau.

```
Phút giờ ngày_của_tháng tháng_của_năm ngày_của_tuần lệnh
0 8 * * 1      /u/sartin/bin/status_report
```

cho phép /u/sartin/bin/status_report được thực hiện vào 8 giờ 00 phút các thứ hai.

Mỗi hàng chứa thời gian và lệnh. Lệnh sẽ được **cron** thực hiện tại thời điểm ghi ở trước trên cùng dòng đó. Năm cột đầu tiên quan trọng thời gian có thể thay thế bằng dấu sao “*” với ý nghĩa là “với mọi”. Các giá trị có thể cho các trường là :

- minute (0-59)
- hour (0-23)
- day of month (1-31)
- month of year (1-12)
- day of week (0-6, 0 is Sunday)
- Command (rest of line)

Sau đó dùng lệnh **crontab** để cài đặt tập tin lệnh vào thư mục /usr/spool/cron/crontabs. Mỗi người sử dụng sẽ có một tập tin crontab trùng tên mình (user name) để lưu tất cả các lệnh cần thực hiện theo chu kỳ trong thư mục này. Cú pháp sử dụng **crontab**:

crontab tên_tập_tin_lệnh

Sau khi hiểu rõ cấu trúc các tập tin, người sử dụng có thể tự tạo các tập tin crontab và đặt vào thư mục theo đúng quy định của **cron** mà không cần phải dùng **crontab**. Điều này còn đúng cho đại đa số các dịch vụ khác. Các chương trình của Unix thường tuân theo một quy tắc là có các tập tin cấu hình dạng text. Các tập tin này hoàn toàn có thể được tạo ra bằng các phần mềm soạn thảo văn bản. Các chương trình tiện ích chỉ là công cụ trợ giúp nếu người sử dụng muốn và không mang tính chất bắt buộc. Để có thể trở thành một người quản trị Unix thực thụ, chúng ta nên tập dần cung cách cấu hình trực tiếp không thông qua các tiện ích.

2.2.1 . Inetd và các dịch vụ mạng :

Một cách khởi động tiến trình rất phổ biến và đặc trưng của Unix là thông qua chương trình **inetd** (đọc là inét-đê). Chương trình **inetd** được sử dụng để khởi động các daemon phục vụ các dịch vụ mạng. **inetd** đợi các kết nối sau một số cổng. Khi có yêu cầu kết nối, **inetd** sẽ gọi chương trình server tương ứng để thiết lập các kết nối. Như vậy Linux có hai cách để tổ chức các service : hoặc là khởi động ngay từ đầu chương trình server , hoặc là để công tác khởi động chương trình dịch vụ theo yêu cầu (khi có yêu cầu kết nối) với sự trợ giúp của **inetd**.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

Thông thường, **inetd** được khởi động ngay từ đầu bởi các script dùng cho khởi động máy. **inetd** sẽ đọc file cấu hình */etc/inetd.conf* khi được gọi lên bộ nhớ . Sau đây là một vài dòng của tập tin */etc/inetd.conf*

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path>
<args>
# Echo, discard, daytime, and chargen are used primarily for testing.
# To re-read this file after changes, just do a 'killall -HUP inetd'
#time stream tcp nowait root internal
#time dgram udp wait root internal
#
# These are standard services.
#
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

Bên cạnh tập tin cấu hình */etc/inetd.conf*, tập tin */etc/services* cũng được **inetd** sử dụng để biết các cổng (port) của các chương trình server. Ví dụ một đoạn của tập tin */etc/services*

```
ftp-data      20/tcp
ftp          21/tcp
fsp          21/udp  fspd
ssh          22/tcp  # SSH Remote Login Protocol
ssh          22/udp  # SSH Remote Login Protocol
telnet       23/tcp  # 24 - private
smtp         25/tcp  mail # 26 - unassigned
time         37/tcp  timserver
time         37/udp  timserver
rlp          39/udp  resource # resource location
nameserver   42/tcp  name   # IEN 116
whois        43/tcp  nickname
re-mail-ck   50/tcp  # Remote Mail Checking Protocol
re-mail-ck   50/udp  # Remote Mail Checking Protocol
domain       53/tcp  nameserver # name-domain server
domain       53/udp  nameserver
```

Hai tập tin */etc/inetd.conf* và */etc/services* quan hệ mật thiết với nhau. Cột đầu tiên bao gồm tên các dịch vụ mạng và cần phải giống nhau. Một dịch vụ muốn được hoạt động nhờ **inetd** phải khai báo cổng mà nó đợi khách hàng thông qua */etc/services* và dòng lệnh khởi động nó trong */etc/inetd.conf*. Muốn tắt một dịch vụ, ta chỉ cần đặt dấu chấm # trước dòng miêu tả dịch vụ và khi đó, **inetd** sẽ không biết và không gọi dịch vụ đó nữa. Việc sử dụng **inetd** cho phép tiết kiệm tài nguyên của hệ thống vì

chỉ có những dịch vụ nào được sử dụng mới được gọi lên bộ nhớ. Như các chúng ta đọc nhận thấy nội dung của cột `<server_path> <args>` cho các dịch vụ là `/usr/sbin/tcpd in.telnetd`. Chương trình `tcpd` được `inetd` gọi lên trước để làm một số công tác kiểm tra và ghi log trước khi chương trình dịch vụ thực được gọi lên.

Cột `<flags>` cho biết chương trình `inetd` có phải đợi (wait) hay không (nowait) kết nối kết thúc trước khi “tiếp” một kết nối khác. Ví dụ trên với telnet cho thấy nhiều chương trình khách có thể được phục vụ một lúc qua cùng một cổng telnet 23. Tất nhiên, chương trình server telnet cũng phải được thiết kế thích hợp với kiểu làm việc đa khách hàng này.

Cột `<user>` quy định quyền của tiến trình khi nó được chạy trên bộ nhớ. Trong trường hợp có nghi ngờ về tính bảo mật của một dịch vụ, ta có thể giảm quyền của nó bằng cách thay đổi nội dung của cột này.

Qua ví dụ trên ta thấy dịch vụ ftp sẽ được `inetd` gọi lên thông qua dòng lệnh `/usr/sbin/tcpd in.ftpd -l -a` khi có một chương trình khách hàng dùng giao thức TCP gọi qua cổng 21.

2.2.2 Hệ thống tập tin của Linux :

Đối với hệ điều hành Linux, không có khái niệm các ổ đĩa khác nhau. Sau quá trình khởi động, toàn bộ các thư mục và tập tin được ‘gắn’ lên (mount) và tạo thành một hệ thống tập tin thống nhất, bắt đầu từ gốc ‘/’

```
/-----+
    !-----/bin
    !-----/sbin
    !-----/usr-----/usr/bin
    !
    !-----/usr/sbin
    !
    !-----/usr/local
    !
    !-----/usr/doc
    !
    !-----/etc
    !-----/lib
    !-----/var-----/var/adm
    !-----/var/log
    !-----/var/spool
```

Hình trên là cây thư mục của đa số các Unix. Với cây thư mục trên ta không thể nào biết được số lượng đĩa cứng, các phân mảnh của mỗi đĩa và sự tương ứng giữa các phân mảnh và thư mục như thế nào.

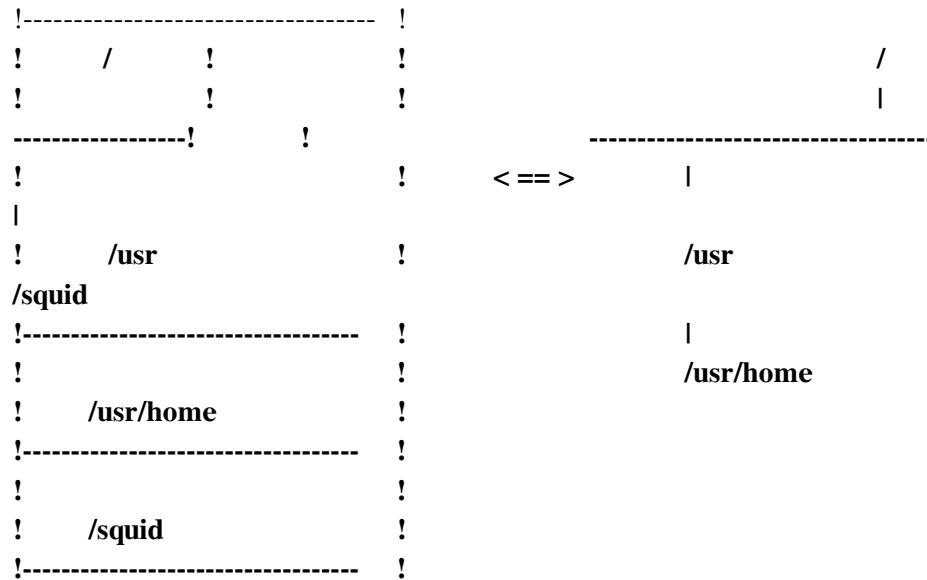
Chúng ta có thể chia đĩa cứng thành nhiều phân mảnh (partition). Mỗi partition là một hệ thống tập tin (file system) độc lập. Sau đó, các hệ

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

thống tập tin này được ‘gắn’ (mount) vào hệ thống tập tin thống nhất của toàn hệ thống. Chúng ta hoàn toàn có thể gắn thêm một đĩa cứng mới, format rồi mount vào hệ thống tập tin dưới tên một thư mục nào đó và tại một điểm (mount point) nào đó. Đối với các chương trình chạy trên Unix, không hề có khái niệm một thư mục nằm ở đĩa nào hay partition nào.

Hình sau đây cho thấy sự tương quan giữa vị trí vật lý trên đĩa và vị trí logic trong cây tập tin.



Thư mục **/usr/home** là thư mục con của **/usr** trong cây thư mục, nhưng trên đĩa vật lý, đây là hai phân mảnh (partition) cạnh nhau.

Hệ thống tập tin được OS Linux mount trong quá trình khởi động tuân theo các thông số ghi trong tập tin **/etc/fstab** (một lần nữa, nếu chúng ta nắm vững cú pháp của tập tin này, chúng ta có thể thay đổi nó thông qua một chương trình soạn thảo văn bản text bất kỳ và có một kiểu khởi động hệ thống tập tin như chúng ta muốn)

```
[tnminh@pasteur tnminh]$ more /etc/fstab
/dev/hda2      /
/dev/hda3      swap    swap    defaults    1 1
/dev/fd0       /mnt/floppy ext2    defaults    0 0
/dev/cdrom     /mnt/cdrom   iso9660  noauto,ro  0 0
none          /proc      proc    defaults    0 0
none          /dev/pts   devpts  mode=0622   0 0
```

Cột 1 (fs_spec) : các trang thiết bị (device) cần mount

- 2 (fs_file) : điểm treo (mount point)
- 3 (fs_vfstype) : Kiểu của hệ thống tập tin,

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

- 4 (fs_mntops) : các options. Default = mount khi khởi động, ro = read only, user nếu cho phép user mount hệ thống tập tin này ...
- 5 (fs_freq) : hiện thị (dumped) hay không hệ thống tập tin
- 6 (fs_passno) : có cần kiểm tra hay không bởi fsck

Tập tin **/etc/fstab** được sử dụng bởi chương trình **mount** trong quá trình khởi động của Linux. Dòng

```
/dev/cdrom /mnt/cdrom iso9660 noauto,ro 0 0
```

cho phép ổ CDROM có thể mount theo ý muốn của người dùng (không mount automatic) và gắn vào **/mnt/cdrom** với kiểu hệ thống tập tin iso9660 với mục đích chỉ đọc. Nếu không có từ khóa **user** thì chỉ có **root** mới được quyền mount ổ CDROM. Với tập tin **/etc/fstab** như trên thì lệnh mount/unmount ổ CDROM sẽ là :

```
mount /dev/cdrom hay umount /dev/cdrom
```

Mount không có thông số cho phép hiển nội dung tập tin **/etc/mtab** bằng những hệ thống tập tin đã được mounted.

```
[root@pasteur tnminh]# mount  
/dev/hda2 on / type ext2 (rw)  
none on /proc type proc (rw)  
none on /dev/pts type devpts (rw,mode=0622)  
/dev/hda1 on /home/tnminh/minh type vfat (rw)
```

So sánh với

```
[root@pasteur tnminh]# more /etc/mtab  
/dev/hda2 / ext2 rw 0 0  
none /proc proc rw 0 0  
none /dev/pts devpts rw,mode=0622 0 0  
/dev/hda1 /home/tnminh/minh vfat rw 0 0
```

Ở đây chúng ta thấy 2 dòng đặc biệt :

```
none /proc proc rw 0 0  
none /dev/pts devpts rw,mode=0622 0 0
```

/dev chứa những tập tin đặc biệt : tập tin thiết bị ngoại vi. Hệ thống Linux sử dụng các tập tin này để truy xuất dữ liệu đến các thiết bị ngoại vi. Như vậy, Linux giao tiếp đến các thiết bị ngoại vi giống như với các tập tin. Ví dụ **/dev/psaux** được dùng để giao tiếp với chuột, **/dev/hda1** để giao tiếp với phân mảng 1 của đĩa cứng master của controller số 0...

```
brw-rw---- 1 root disk 3, 1 May 6 1998 hda1  
crw-rw-r-- 1 root root 10, 1 May 6 1998 saux  
crw----- 1 root tty 4, 64 Oct 3 15:55 tyS0  
crw----- 1 root tty 4, 65 May 6 1998 tyS1  
crw----- 1 root tty 4, 66 May 6 1998 tyS2
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

crw----- 1 root tty 4, 67 May 6 1998 tyS3

Ký tự cột đầu tiên ‘**b**’ để thông báo kiểu giao tiếp block (cho thiết bị như ổ đĩa), ‘**c**’ – giao tiếp kiểu ký tự (cho thiết bị như bàn phím, chuột ...).

Tóm lại, ta nhận thấy cây thư mục của Unix cũng giống như cây thư mục của MS DOS hay Windows.

/proc là hệ thống tập tin ảo cho phép đọc các thông tin của các process trên bộ nhớ. Để thực tập, ta có thể dùng **ps** để coi các tiến trình và thấy các tập tin tương ứng trong **/proc** như ví dụ sau :

```
[oracle@appserv]$ ps ax/grep 582
582 ? S 0:17 nmbd -D
8724 pts/5 S 0:00 grep 582
[oracle@appserv]$ cd /proc/582
[oracle@appserv 582]$ ls -l
ls: exe: Permission denied
ls: root: Permission denied
ls: cwd: Permission denied
total 0
-r--r--r-- 1 root root 0 Oct 12 17:39 cmdline
lrwx----- 1 root root 0 Oct 12 17:39 cwd
-r----- 1 root root 0 Oct 12 17:39 environ
lrwx----- 1 root root 0 Oct 12 17:39 exe
dr-x----- 2 root root 0 Oct 12 17:39 fd
pr--r--r-- 1 root root 0 Oct 12 17:39 maps
-rw----- 1 root root 0 Oct 12 17:39 mem
lrwx----- 1 root root 0 Oct 12 17:39 root
-r--r--r-- 1 root root 0 Oct 12 17:39 stat
-r--r--r-- 1 root root 0 Oct 12 17:39 statm
-r--r--r-- 1 root root 0 Oct 12 17:39 status
[oracle@appserv 582]$ more cmdline
nmbd-D
[oracle@appserv 582]$
```

Các ký tự in đậm trong ví dụ trên cho phép thấy được mối liên hệ giữa **/proc** và tiến trình đang chạy. Các thao tác trên có thể thực hiện bởi một user bất kỳ.

a. *Quyền truy cập, sở hữu tập tin và thư mục của Linux (directory and file permission and ownership) :*

Do Linux là một hệ điều hành multitasking và multiuser, nhiều người cùng có thể sử dụng một máy Linux và một người có thể cho chạy nhiều chương trình khác nhau. Có hai vấn đề lớn được đặt ra : quyền sở hữu các

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

dữ liệu trên đĩa và phân chia tài nguyên hệ thống như CPU, RAM ... giữa các process. Chúng ta sẽ bàn về sở hữu các tập tin và các quyền truy xuất tập tin.

Tất cả các tập tin và thư mục của Linux đều có người sở hữu và quyền truy nhập. Chúng ta có thể đổi các tính chất này cho phép nhiều hay ít quyền truy nhập hơn đối với một tập tin hay thư mục. Quyền của tập tin còn cho phép xác định tập tin có là một chương trình (application) hay không (khác với MSDOS và MSWindows xác định tính chất này qua phần mở rộng của tên tập tin). Ví dụ với lệnh **ls -l** chúng ta có thể thấy

```
-rw-r—r— 1 fido users 163 Dec 7 14:31 myfile
```

Cột đầu chỉ ra quyền truy cập tập tin

Cột 2 chỉ số liên kết (link) đối với tập tin hay thư mục

Cột 3, 4 chỉ chủ sở hữu và nhóm sở hữu

Cột 5 chỉ độ dài của tập tin

Cột 6 chỉ thời gian thay đổi cuối cùng

Cột 7 là tên tập tin hay thư mục

Trong ví dụ trên, các ký tự **-rw-r—r—** biểu thị quyền truy cập của tập tin **myfile**. Sở hữu của **myfile** là **fido** và nhóm sở hữu **myfile** là **users**. **Fido** được quyền đọc và ghi vào **myfile**, còn những người sử dụng của nhóm **users** và những người khác chỉ được quyền đọc **myfile**.

b. *Sở hữu mặc định tập tin và thư mục.*

Khi chúng ta tạo ra một tập tin hay thư mục, chúng ta là sở hữu của những cấu trúc này. Là người sở hữu, chúng ta có quyền thay đổi quyền truy nhập và quyền sở hữu của tập tin hay thư mục. Tất nhiên, nếu chúng ta “cho” quyền sở hữu cho một người khác, chúng ta không thể lấy lại được nữa trừ khi người khác đó “cho” lại chúng ta. Các quyền truy cập mặc định đến tập tin cho chủ sở hữu, nhóm sử dụng và những người khác được xác định thông qua cấu hình môi trường của người làm việc. Chúng ta có thể xem và thay đổi cấu hình này qua lệnh **umask**. Nói một cách đơn giản, **umask** quy định các quyền sẽ được lấy đi (không cho) mỗi khi tập tin được tạo ra. Chúng ta sẽ quay lại vấn đề này.

Sở hữu của các tập tin hệ thống được xác định trong quá trình cài đặt hệ điều hành. Hệ thống tập tin của Linux thường có chủ sở hữu là root, uucp, bin ... Không được thay các sở hữu này ngoại trừ khi chúng ta hiểu rất rõ điều mình làm.

- **Lệnh chown.**

Sử dụng lệnh **chown** (change ownership) để thay đổi sở hữu một tập tin hay thư mục. Cú pháp lệnh này là **chown <owner> <filename>**. Trong ví dụ sau, chúng ta thay đổi sở hữu của **myfile** từ **fido** sang **root** :

```
darkstar:~$ ls -l myfile
-rw-r--r-- 1 fido users 114 Dec 7 14:31 myfile
darkstar:~$ chown root myfile
darkstar:~$ ls -l myfile
-rw-r--r-- 1 root users 114 Dec 7 14:31 myfile
```

Để làm tiếp các thay đổi khác, cần phải log vào hệ thống như super-user.

- **Lệnh chgrp.**

Unix tổ chức người sử dụng thành các nhóm (group) khác nhau. Đây là cách tổ chức mềm dẻo cho phép một nhóm người cùng có quyền truy nhập đến một tập tin hoặc thư mục. Lệnh **chgrp** được sử dụng để thay đổi nhóm. Cú pháp của lệnh này giống như lệnh **chown**.

c. *Quyền truy xuất tập tin (File Permissions)*

Linux cho phép người sử dụng xác định các quyền đọc (read), viết (write) và thực hiện (execute) cho từng đối tượng trong nhóm sau : sở hữu (the owner), nhóm (the group), và những người còn lại ("others" (everyone else)).

Quyền đọc cho phép chúng ta đọc nội dung của tập tin. Đối với thư mục quyền đọc cho phép chúng ta sử dụng lệnh **ls** để xem nội dung của thư mục.

Quyền viết cho phép chúng ta thay đổi nội dung hay xóa tập tin. Đối với thư mục, quyền viết cho phép chúng ta tạo ra, xóa hay thay đổi tên trong thư mục.

Quyền thực hiện cho phép chúng ta gọi chương trình lên bộ nhớ bằng cách nhập từ bàn phím tên của tập tin. Đối với thư mục, chúng ta chỉ có thể vào thư mục bởi lệnh **cd** nếu chúng ta có quyền thực hiện với thư mục

Xem xét lại ví dụ trên :

```
-rw-r--r-- 1 fido users 163 Dec 7 14:31 myfile
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

Ký tự đầu tiên của quyền là ký tự – ám chỉ rằng đó là một tập tin bình thường. Nếu **myfile** là một thư mục, ta sẽ thấy vào đó ký tự **d**. Ngoài ra còn có **c** cho thiết bị ngoại vi dạng ký tự (như bàn phím), **b** cho thiết bị ngoại vi dạng block (như ổ đĩa cứng).

Chín ký tự tiếp theo chia thành 3 nhóm, cho phép xác định quyền của ba nhóm sở hữu (owner), nhóm (group) và còn lại (other). Mỗi cặp ba này cho phép xác định quyền đọc, viết và thực hiện theo thứ tự kể trên. Quyền đọc viết tắt là “**r**” ở vị trí đầu tiên, quyền viết viết tắt bằng “**w**” ở vị trí thứ hai và vị trí thứ ba là quyền thực hiện ký hiệu bằng chữ “**x**” . Nếu một quyền không được cho, tại vị trí đó sẽ có ký tự “-“.

Trong trường hợp của tập tin **myfile**, sở hữu có quyền **rw** tức là đọc và viết. **Myfile** không phải là một chương trình. **Nhóm** cùng với **còn lại** chỉ có quyền đọc tập tin (read-only).

Song song với cách ký hiệu miêu tả ở trên, quyền thao tác tập tin còn có thể cho dưới dạng 3 số . Đối với **myfile**, quyền đó là 644. Điều quan trọng là phải hiểu cách ký hiệu bằng số vì nó liên quan đến việc thay đổi các quyền sau này. Các số có thể nhận tất cả các giá trị từ 0 đến 7. Số đầu tiên miêu tả quyền của sở hữu, số thứ hai cho nhóm và số thứ ba cho còn lại.

Mỗi số là tổng của các quyền theo quy tắc sau :

Read permission	4
Write permission	2
Execute permission	1

Bảng 1: Các giá trị xác lập quyền

Vì vậy, một tập tin với quyền 751 có nghĩa là sở hữu có quyền read, write, và execute bằng $4+2+1=7$, Nhóm có quyền read và execute bằng $4+1=5$, và còn lại có quyền execute bằng 1.

Nếu chúng ta xem kỹ, chúng ta sẽ thấy mọi số từ 0 đến 7 đều tương ứng với một tổ hợp duy nhất các quyền truy nhập tập tin.

- 0 or ---: No permissions at all*
- 4 or r--: read-only*
- 2 or -w-: write-only (rare)*
- 1 or --x: execute*
- 6 or rw-: read and write*
- 5 or r-x: read and execute*

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

3 or -wx: write and execute (rare)

7 or rwx: read, write, and execute

Nếu chúng ta quen với hệ nhị phân, hãy suy nghĩ bằng hệ thống nhị phân. Khi đó, rwx sẽ như số nhị phân 3 bits. Nếu quyền được cho, số nhị phân tương ứng sẽ bằng 1, ngược lại, nó sẽ bằng 0. Ví dụ r-x sẽ là số nhị phân 101, và theo hệ thập phân sẽ là 4+0+1, hay 5. —x sẽ tương ứng 001, hay 0+0+1 = 1.

Chú ý, người sử dụng có quyền đọc thì có quyền copy tập tin và tập tin sao chép sẽ thuộc sở hữu người làm copy.

Các quyền mặc định khi tạo tập tin khi một tập tin hay thư mục được tạo ra, permission mặc định sẽ được xác định bởi các quyền trừ bớt bởi các quyền hiển thị bằng umask

```
[tnminh@pasteur tnminh]$ umask  
002  
[tnminh@pasteur tnminh]$ echo tao mot file > tmp  
[tnminh@pasteur tnminh]$ ls -l  
total 5472  
-rw-rw-r-- 1 tnminh tnminh 13 Oct 3 21:55 tmp  
[tnminh@pasteur /etc]$ umask 022  
[tnminh@pasteur tnminh]$ echo tao mot file khac >tmp1  
[tnminh@pasteur tnminh]$ ls -l  
-rw-rw-r-- 1 tnminh tnminh 13 Oct 3 21:55 tmp  
-rw-r--r-- 1 tnminh tnminh 18 Oct 3 21:59 tmp1
```

Trong ví dụ trên, quyền mặc định lúc đầu xác định bởi umask=002. Khi đó, tập tin tmp tạo ra sẽ có quyền là 664 và đó chính là bù đweeney 6 của umask. Quyền thực hiện chương trình cần được gán cố ý bởi người sử dụng hay các chương trình biên dịch (Unix khác có thể thực hiện khác quy tắc này). Sau đó ta đổi giá trị của umask thành 022 và tập tin tạo ra có quyền 644. Giá trị mặc định của các quyền thường được gán mỗi khi người sử dụng login vào hệ thống thông qua các tập tin khởi tạo biến môi trường như **.profile**, **.bashrc**. Để trên quan điểm bảo mật hệ thống, giá trị 024 là tốt nhất, nó cho người cùng nhóm có quyền đọc và không cho quyền nào với những người khác.

- Lệnh chown, chgrp và chmod :

Đây là nhóm lệnh được sử dụng rất phổ biến, cho phép thay quyền truy cập của tập tin hay thư mục. Chỉ có chủ sở hữu và superuser mới có quyền thực hiện các lệnh này.

Cách dùng lệnh : **chmod quyền_truy_cập_mới tên_file**.

darkstar:~\$ ls -l myfile

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

```
-rw-r--r-- 1 fido users 114 Dec 7 14:31 myfile  
darkstar:~$ chmod 345 myfile  
darkstar:~$ ls -l myfile  
-wxr--r-x 1 fido users 114 Dec 7 14:31 myfile  
darkstar:~$ chmod 701 myfile  
darkstar:~$ ls -l myfile  
-rwx---x 1 root users 114 Dec 7 14:31 myfile
```

Ví dụ thay đổi và hiện thị cho thấy sự thay đổi quyền truy cập tập tin **myfile**. Chú ý là ta có quyền cấp phát quyền thực hiện (execute) mà không cần biết là tập tin có phải là một chương trình hay không.

Phương pháp thay đổi tuyệt đối này có một số ưu điểm vì nó là cách định quyền tuyệt đối, kết quả cuối cùng không phụ thuộc vào quyền truy cập trước đó của tập tin. Đồng thời, dễ nói “thay quyền tập tin thành bảy-năm-năm” thì dễ hơn là “thay quyền tập tin thành đọc-viết-thực hiện, đọc-thực hiện, đọc-thực hiện”

Chúng ta cũng có thể thay đổi quyền truy nhập một cách tương đối và dễ nhớ. Để chỉ ra nhóm quyền nào cần thay đổi, chúng ta có thể sử dụng **u (user)**, **g (group)**, **o (other)**, **hay a (all)**. Tiếp theo đó là dấu + để thêm quyền và – để bớt quyền. Cuối cùng là bản thân các quyền viết tắt bởi r,w,x. Ví dụ như để bổ sung quyền thực hiện cho nhóm và còn lại, ta nhập vào dòng lệnh

```
darkstar:~$ chmod go+x myfile
```

Đây là cách thay đổi tương đối vì kết quả cuối cùng phụ thuộc vào quyền đã có trước đó mà lệnh này không liên quan đến. Trên quan điểm bảo mật hệ thống, cách thay đổi tuyệt đối dẫn đến ít sai sót hơn. Thay đổi quyền truy cập của một thư mục cũng được thực hiện giống như đối với một tập tin. Chú ý là nếu chúng ta không có quyền thực hiện (execute) đối với một thư mục, chúng ta không thể thay đổi thư mục **cd** vào thư mục đó. Mọi người sử dụng có quyền viết vào thư mục đều có quyền xóa tập tin trong thư mục đó, không phụ thuộc vào quyền của người đó đối với tập tin. Vì vậy, đa số các thư mục có quyền **drwxr-xr-x**. Như vậy chỉ có người sở hữu của thư mục mới có quyền tạo và xóa tập tin trong thư mục. Ngoài ra, thư mục còn có một quyền đặc biệt, đó là cho phép mọi người đều có quyền tạo tập tin trong thư mục, mọi người đều có quyền thay đổi nội dung tập tin trong thư mục, nhưng chỉ có người tạo ra mới có quyền xóa tập tin. Đó là sticky bit cho thư mục. Thư mục /tmp thường có sticky bit bật lên

```
drwxrwxrwt 7 root root 16384 Oct 21 15:33 tmp
```

Ta thấy chữ **t** cuối cùng trong nhóm các quyền, thể hiện cho sticky bit của **/tmp**

d. Liên kết (link) tập tin: Trong Unix có 2 hình thức liên kết hoàn toàn khác nhau, đó là hard link và soft link hay symbolic link. Hard link cho phép tạo một tên mới cho tập tin. Các tên này có vai trò hoàn toàn như nhau và tập tin chỉ bị hoàn toàn xóa bỏ khi hard link cuối cùng của nó bị xóa. Lệnh **ls -l** cho phép hiển thị số hard link đến tập tin. Symbolic link có chức năng giống như **shortcut** của MS Windows. Khi ta đọc/ghi soft link, ta đọc/ghi tập tin; khi ta xóa symbolic link, ta chỉ xóa symbolic link và tập tin được giữ nguyên. Link được tạo bởi lệnh **ln**. Tự chọn **ln -s** cho phép tạo symbolic link. Ví dụ

```
[tnminh@pascal tnminh]$ls -l  
-rw----- 1 tnminh pkt 517 Oct 27 12:00 mbox  
drwxr-xr-x 2 tnminh pkt 4096 Aug 31 17:50 security  
[tnminh@pascal tnminh]$ln -s mbox mybox  
[tnminh@pascal tnminh]$ln -s security securproj  
[tnminh@pascal tnminh]$ln -l  
-rw----- 1 tnminh pkt 517 Oct 27 12:00 mbox  
lrwxrwxrwx 1 tnminh pkt 4 Oct 27 17:57 mymail -> mbox  
lrwxrwxrwx 1 tnminh pkt 8 Oct 27 17:57 secrproj -> security  
drwxr-xr-x 2 tnminh pkt 4096 Aug 31 17:50 security  
[tnminh@pascal tnminh]$
```

Chúng ta đọc có thể thấy khá rõ kết quả của symbolic link qua thí dụ trên.

Symbolic link rất có nhiều ứng dụng. Ví dụ như một tập tin XXX của một chương trình YYY nằm trong thư mục /var/ZZZ. Nếu phân mảnh của /var/ZZZ bị quá đầy, ta có thể sơ tán XXX qua một thư mục khác thuộc phân mảnh khác và tạo một link thế vào đó mà chương trình YYY vẫn không hề “hay biết” vì nó vẫn truy cập đến /var/ZZZ/XXX như thường lệ.

2.2.3. Quá trình khởi động và kết thúc của UNIX :

Như thông lệ, khi một máy tính được khởi động, sau khi kiểm tra các thiết bị phần cứng gắn trên máy tính qua các chương trình kiểm tra ghi trong ROM, hệ điều hành được tải lên bộ nhớ. Công tác đầu tiên của hệ điều hành là kiểm tra các thiết bị ngoại vi và tải các chương trình điều khiển (driver) cần thiết lên bộ nhớ. Sau các công tác này, bắt đầu giai đoạn định hình hệ thống và mỗi hệ điều hành, thậm chí mỗi phiên bản của

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

một hệ điều hành thực hiện một khác. Chúng tôi xin giới thiệu cách thức khởi động và cấu hình hệ thống của Linux RedHat 6.x . Các Unix khác như SUN OS 6.x, 7.x cũng có hệ thống các tập tin khởi động và cơ chế hoạt động gần giống như Linux RedHat 6.x.

Tập tin đầu tiên mà hệ điều hành xem xét đến là **/etc/inittab**

```
[tnminh@proxy tnminh]$ ls -l /etc/inittab
-rw-r--r-- 1 root root 1756 May 30 15:51 inittab
```

```
[tnminh@proxy tnminh]$ more /etc/inittab
#
# inittab      This file describes how the INIT process should set up
#                 the system in a certain run-level.
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have
networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
```

Mức (level) làm việc mặc định được quy định trong tập tin này. Ví dụ trên cho thấy mức mặc định là mức 3 ở dòng cuối cùng. Unix nói chung có 7 mức hoạt động khác nhau từ 0 đến 6. Mức 0 là để shutdown hệ thống. Mức 1 là đơn người sử dụng (single user) và thường được dùng để sửa chữa lỗi hệ thống tập tin, mức 2, 3 là hai mức cho đa người sử dụng, mức 6 dùng để reboot hệ thống, mức 4,5 do người sử dụng tự thiết kế cho mình. Tương ứng với các mức trên, trong thư mục /etc/rc.d có các thư mục rc0.d – rc6.d, chứa các tập tin khởi động trong từng mức (rc là viết tắt của run command). RedHat 6.x có thư mục **/etc/rc.d/init.d** chứa tất cả các tập tin khởi động. Thường các tập tin này là các shell script (tập hợp lệnh shell) hoặc perl script (như Debian Linux chẳng hạn). Trong các thư mục rc?.d chỉ có các liên kết hình thức (symbolic link) đến các tập tin khởi động trong **/etc/rc.d/init.d**. SUN OS 7.0 lại đặt thực sự các script khởi động vào các thư mục **rc?.d** thay vì symbolic link.

```
[tnminh@proxy /etc/rc.d]$ ls -l
```

```
total 22
drwxr-xr-x 2 root root 1024 May 10 09:44 init.d
-rw xr-xr-x 1 root root 2722 Apr 15 1999 rc
-rw xr-xr-x 1 root root 693 Aug 17 1998 rc.local
-rw xr-xr-x 1 root root 9822 Apr 14 1999 rc.sysinit
drwxr-xr-x 2 root root 1024 May 3 01:44 rc0.d
drwxr-xr-x 2 root root 1024 May 3 01:44 rc1.d
drwxr-xr-x 2 root root 1024 May 3 01:44 rc2.d
drwxr-xr-x 2 root root 1024 May 10 09:47 rc3.d
drwxr-xr-x 2 root root 1024 May 3 01:44 rc4.d
drwxr-xr-x 2 root root 1024 May 3 01:44 rc5.d
drwxr-xr-x 2 root root 1024 May 3 01:44 rc6.d
```

Trong các thư mục **rc?.d**, các script bắt đầu bằng **S** (start) được sử dụng khi khởi động, còn các script bắt đầu từ **K** (kill) dùng để dừng các tiến trình trước khi qua một mức hoạt động khác.

Toàn bộ các tập tin này quyết định cấu hình làm việc của một máy Unix sau khi hoàn thành quá trình khởi động. Việc khởi động hệ thống các dịch vụ cũng thực hiện thông qua cơ chế như đã miêu tả trên.

Lệnh **init số_mức** cho phép chuyển giữa các mức của hệ thống. Ví dụ

```
[root@proxy /etc/rc.d]# init 1
```

cho phép chuyển hệ thống từ mức hiện hành qua mức 1 để sửa chữa. Sau đó **init 3** cho phép quay về mức 3 đa người dùng.

Khi chúng ta đánh lệnh **shutdown**, toàn bộ hệ thống chuyển về mức 0 và chúng ta dừng hệ thống.

2.2.4. Quản lý người sử dụng :

Trong quá trình cài đặt Linux chúng ta khởi tạo người sử dụng **root** cho hệ thống. Đây là superuser, tức là người sử dụng đặc biệt và không có giới hạn nào về quyền hạn đối với root. Sử dụng quyền root chúng ta rất thấy thoải mái vì chúng ta có thể làm được thao tác mà không phải lo lắng gì đến xét quyền truy cập này hay khác. Tuy nhiên, khi hệ thống bị sự cố do một lỗi lầm nào đó, chúng ta mới thấy sự nguy hiểm khi làm việc như root. Chúng ta thử hình dung toàn bộ các Email của một mail server của toàn công ty bị xóa do đánh một lệnh sai thì tác hại sẽ lớn đến mức nào. Vì vậy, hãy chỉ dùng quyền root khi chúng ta không có cách nào khác.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

Cần phân biệt chúng ta đang login như root hay người sử dụng thường thông qua dấu nhắc của shell.

```
login: tnminh
Password:
Last login: Sat Oct 28 14:30:15 from 172.16.10.199
[tnminh@pascal tnminh]$ su -
Password:
[root@pascal /root]#
```

Dòng 4 với dấu \$ cho thấy ta đang kết nối như một người sử dụng thường (**tnminh**). Dòng cuối cùng với dấu # cho thấy chúng ta đang thực hiện các lệnh như **root**. Lệnh **su** cho phép chúng ta thay đổi **login** dưới một user khác mà không phải **logout** rồi **login** lại.

Chúng ta cần tạo các tài khoản (**account**) cho người sử dụng thường sớm nhất có thể được (đầu tiên là cho bản thân chúng ta). Với những server quan trọng và có nhiều dịch vụ khác nhau, thậm chí chúng ta có thể tạo ra các superuser thích hợp cho từng dịch vụ để tránh dùng root cho các công tác này. Ví dụ như superuser cho công tác **backup** chỉ cần chức năng đọc (read-only) mà không cần chức năng ghi.

Tập tin /etc/passwd. Tập tin **/etc/passwd** đóng một vai trò sống còn đối với một hệ thống Unix. Mọi người đều có thể đọc được tập tin này nhưng chỉ có **root** mới có quyền thay đổi nó. Tập tin **/etc/passwd** được lưu dưới dạng text hiển như đại đa số các tập tin cấu hình của Unix.

```
[oracle@appserv oracle]$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

```
xfs:x:43:43:X Font Server:/etc/X11/fs:/bin/false  
named:x:25:25:Named:/var/named:/bin/false.  
piranha:x:60:60::/home/httpd/html/piranha:/dev/null  
pvm:x:24:24::/usr/share/pvm3:/bin/bash  
squid:x:23:23::/var/spool/squid:/dev/null  
guest:x:500:500:guest:/home/guest:/dev/null  
tnminh:x:501:501:TNMinh:/home/tnminh:/bin/bash
```

Mỗi user được lưu trong một dòng gồm 7 cột.

Cột 1 : tên người sử dụng

Cột 2 : mã liên quan đến passwd cho Unix chuẩn và ‘x’ đối với Linux. Linux lưu mã này trong một tập tin khác **/etc/shadow** mà chỉ có root mới có quyền đọc.

Cột 3:4 : user ID:group ID

Cột 5: Tên đầy đủ của người sử dụng. Một số phần mềm phá password sử dụng dữ liệu của cột này để thử đoán password.

Cột 6: thư mục cá nhân

Cột 7: chương trình shell cho user

Tập tin mở đầu bởi superuser **root**. Chú ý là tất cả những user có user ID = 0 đều là root. Tiếp theo là các user hệ thống. Đây là các user không có thật và không thể login vào hệ thống. Cuối cùng là các user bình thường.

a. **Tạo user (account) mới :** Để tạo một account, chúng ta có thể sử dụng lệnh **adduser** (hoặc useradd tùy vào phiên bản). Tất nhiên là chúng ta phải làm thao tác này dưới quyền root (dấu nhắc #)

```
[root@appserv oracle]# /usr/sbin/adduser foo  
[root@appserv oracle]# passwd foo  
Changing password for user foo  
New UNIX password:  
Retype new UNIX password:  
passwd: all authentication tokens updated successfully  
[root@appserv oracle]#
```

Sau khi chúng ta tạo xong user bởi dòng đầu tiên của ví dụ trên, user **foo** vẫn chưa kết nối được vì thiếu password. Chúng ta phải khởi tạo password cho **foo** bởi lệnh **passwd foo** như thấy ở trên.

Vì vấn đề an ninh của máy Unix này và kéo theo sự an toàn của toàn hệ thống mạng của chúng ta, rất quan trọng chọn đúng password. Một password gọi là đúng nếu :

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

- Có độ dài tối thiểu 8 ký tự. Linux hỗ trợ cho password dài hơn 8 ký tự, còn SUN OS chỉ tính đến 8 ký tự
- Phối hợp giữa chữ thường, chữ hoa, số và các ký tự đặc biệt
- Không liên quan đến tên tuổi, ngày sinh ... của chúng ta và người thân
- Không có trong từ điển

Trong ví dụ trên, chúng ta khởi tạo người dùng và không quan tâm gì đến nhóm (group) của người dùng. Rất tiện lợi nếu chúng ta tập hợp nhiều người dùng vào chung một nhóm có cùng một chức năng và cùng chia sẻ nhau dữ liệu. Khi chúng ta tạo người sử dụng như trên, Linux sẽ tạo cho mỗi người một nhóm. Đọc tập tin **/etc/passwd** ta thấy

```
[root@appserv oracle]# more /etc/passwd/grep foo  
foo:x:1012:1013::/home/foo:/bin/bash  
[root@appserv oracle]#
```

foo là user số 1012 và thuộc nhóm 1013.

Xem tập tin **/etc/group** ta thấy

```
[root@appserv oracle]# more /etc/group  
root:x:0:root  
.....  
users:x:100:  
.....  
foo:x:1013:
```

và ta có thể kết nạp **foo** vào nhóm **users** bằng cách thay số 1013 bằng 100, là group ID của users.

- b. Xóa user (account) :** Lệnh **userdel** dùng để xóa một user. Chúng ta cũng có thể xóa một user bằng cách xóa đi dòng dữ liệu tương ứng trong tập tin **/etc/passwd**.

2.2.5. Kết nối mạng thông qua TCP/IP:

Chúng ta sẽ xem xét quá trình nối một máy Linux vào mạng Ethernet để trao đổi thông tin bằng giao thức TCP/IP trên Ethernet.

a. HDH Linux và card mạng:

Để nối một máy Linux vào một mạng Ethernet, chúng ta cần phải có đầu tiên là một card mạng mà Linux đã có chương trình driver. Sau đây là một số mạng mà Linux có trợ giúp (danh sách sau không đầy đủ và các phiên bản mới của Linux hỗ trợ rất nhiều các card mạng khác nhau) :

3Com 3C509
3Com 3C503/16
Novell NE1000
Novell NE2000
Western Digital WD8003
Western Digital WD8013
Hewlett-Packard HP27245
Hewlett-Packard HP27247
Hewlett-Packard HP27250

Giả sử các chúng ta muốn gắn máy của mình vào một mạng LAN Ethernet và chúng ta đã có một card mạng. Vấn đề đầu tiên là sự nhận biết của Linux đối với card này. Nếu card của chúng ta là một card khá phổ biến như 3c509 của 3COM hay NE2000 của Novell, HDH Linux sẽ nhận biết sự hiện diện của card trong quá trình boot. Để biết xem kết quả nhận biết card mạng, ta có thể xem xét các thông báo của kernel Linux trong quá trình boot của hệ thống qua lệnh **dmesg**

```
VFS: Mounted root (ext2 filesystem) readonly.  
Freeing unused kernel memory: 60k freed  
Adding Swap: 72572k swap-space (priority -1)  
eth0: 3c509 at 0x300 tag 1, BNC port, address 00 a0 24 4f 3d dc, IRQ 10.  
3c509.c: 1.16 (2.2) 2/3/98 becker@cesdis.gsfc.nasa.gov.  
eth0: Setting Rx mode to 1 addresses.  
arpwatch uses obsolete (PF_INET,SOCK_PACKET)
```

Hai dòng in đậm báo rằng card mạng 3c509 đã được kernel nhận biết. Trong trường hợp kernel không nhận biết card \otimes , chúng ta phải làm lại kernel Linux và đặt module điều khiển (driver) của card vào trong kernel hay cấu hình ở chế độ module.

Để cấu hình tiếp nối mạng qua TCP/IP chúng ta phải xác định rõ các thông tin liên quan đến địa chỉ IP của máy. Các thông tin cần biết là :

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

-
- Địa chỉ IP của máy
 - Netmask
 - Địa chỉ của mạng
 - Broadcast
 - Địa chỉ IP của gateway

Chúng ta sẽ lần lượt đi qua các khái niệm cơ bản trên và sẽ học sâu hơn trong phần TCP/IP của khóa học.

Địa chỉ IP của máy là một dãy 4 số viết dưới dạng A.B.C.D, trong đó mỗi số nhận giá trị từ 0-255. Nếu máy của chúng ta kết nối một mạng nhỏ tại nhà do chúng ta thiết lập thì địa chỉ kiểu 192.168.1.D là một địa chỉ nên đặt, với D là các số khác nhau cho từng máy. Nếu máy của chúng ta sẽ hòa nhập với một mạng LAN đã có trước đó và chúng ta muốn kết nối với các máy khác thì hỏi người quản trị mạng về địa chỉ IP chúng ta có thể gán cho máy của mình cùng với tất cả các thông số tiếp theo.

- **Netmask.** Tương tự như trên, nếu chúng ta tự quản, netmask sẽ là 255.255.255.0
- **Địa chỉ mạng.** Nếu chúng ta tự quản, địa chỉ của mạng sẽ là 192.168.1.0
- **Broadcast.** Nếu chúng ta tự quản, broadcast là 192.168.1.255
- **Địa chỉ gateway.** Đây là địa chỉ của máy cho phép chúng ta kết nối với mạng LAN khác, tức là các máy tính với 3 số đầu của địa chỉ không giống chúng ta là 192.168.1. Chúng ta bỏ trống nếu chúng ta chỉ liên lạc với các máy cùng mạng 192.168.1.XXX. Chú ý là địa chỉ mạng của máy gateway bắt buộc phải trùng với địa chỉ mạng của chúng ta.

Sau khi đã xác định các thông số, ví dụ như

IP address = 192.168.1.15

Netmask = 255.255.255.0

suy ra network address = 192.168.1.0 và broadcast = 192.168.1.255

Gateway = 192.168.1.1

b. Chúng ta có thể tiến hành cấu hình card mạng:

- **Lệnh ifconfig**

Sau khi làm cho kernel nhận biết sự hiện diện của card mạng, công tác tiếp theo là cấu hình TCP/IP cho card. Trong quá trình cài đặt Linux

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

Redhat 6.X, bình thường chúng ta đã được chương trình cài đặt hỏi và cấu hình hộ . Trong trường hợp khi chúng ta bổ sung card mạng sau khi Linux đã được cài đặt, chúng ta có thể sử dụng tiện ích **netconf** cho mục đích này hoặc chúng ta sử dụng lệnh **ifconfig** để tự cài đặt.

Lệnh **ifconfig** được sử dụng trong quá trình boot hệ thống để cấu hình các trang thiết bị mạng. Sau đó, trong quá trình vận hành, **ifconfig** được sử dụng cho debug, hoặc để cho người quản trị hệ thống thay đổi cấu hình khi cần thiết .

Lệnh **ifconfig** không có tùy chọn dùng để hiển thị cấu hình hiện tại của máy.

```
[root@pasteur tnminh]# /sbin/ifconfig
eth0      Link encap:Ethernet HWaddr 00:A0:24:4F:3D:DC
          inet addr:192.168.2.20 Bcast:192.168.2.255 Mask:255.255.255.0
                  UP BROADCAST RUNNING PROMISC MULTICAST MTU:1500 Metric:1
                  RX packets:531 errors:4 dropped:0 overruns:0 frame:4
                  TX packets:1854 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:100
                  Interrupt:10 Base address:0x300

lo      Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
                  UP LOOPBACK RUNNING MTU:3924 Metric:1
                  RX packets:1179 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:1179 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
```

Để gán địa chỉ IP 193.105.106.10 cho card mạng Ethernet đầu tiên ta dùng lệnh

```
ifconfig eth0 193.105.106.10 netmask 255.255.255.0 broadcast 192.105.106.255
```

Linux cho phép chúng ta sử dụng bí danh (alias) cho card mạng, tức là cho phép chúng ta có nhiều địa chỉ IP cho cùng một card vật lý . Kết quả nhận được giống như chúng ta có gắn nhiều card vật lý lên máy. Do đó, chúng ta có thể dùng một card để nối với nhiều mạng logic khác nhau. Cú pháp của lệnh này là :

```
ifconfig eth0:0 208.148.45.58 netmask 255.255.255.248 broadcast
208.148.45.255 up
```

➤ **Lệnh route.**

Lệnh Route cho phép làm các thao tác đến bảng dẫn đường (forwarding table) của kernel. Nó được sử dụng để xác định đường dẫn cố định (static) đến những máy hoặc mạng qua các card mạng ethernet đã được cấu hình trước đó bởi ifconfig.

Lệnh **route** không có tùy chọn (option) cho phép hiển thị bảng dẫn đường hiện tại của kernel

```
[root@pasteur tnminh]# /sbin/route
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.2.20	*	255.255.255.255	UH	0	0	0	eth0
192.168.2.0	*	255.255.255.0	U	0	0	0	eth0
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	192.168.2.10	0.0.0.0	UG	0	0	0	eth0

Để chỉ ra rằng card mạng eth0 được nối với một mạng 208.148.45.56 ta dùng lệnh route như sau :

```
route add -net 208.148.45.56 eth0
```

Còn nếu chúng ta muốn sử dụng bí danh của card mạng để nối vào một mạng logic khác, ta có thể sử dụng lệnh

```
route add -net 193.105.106.0 eth0:0
```

Công tác cuối cùng là phải chỉ ra các địa chỉ của gateway mặc định.

```
route add default gw 193.105.106.1 metric 1
```

Biết sử dụng thành thạo cú pháp của 2 lệnh **ifconfig** và **route** rất quan trọng, nó cho phép các cán bộ quản trị thay đổi cấu hình kết nối mạng của một server một cách nhanh chóng và không phải khởi động lại máy. Vì vậy, server luôn sẵn sàng.

➤ **Lệnh ping.** Lệnh rất quan trọng vì nó cho phép chúng ta thử xem 2 máy có kết nối được với nhau chưa. Cú pháp cơ bản của lệnh rất đơn giản là *ping địa_chỉ_IP_máy_dịch*. Ví dụ như

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

```
[tnminh@proxy tnminh]$ ping sun
PING sun.vnuhcm.edu.vn (172.16.1.4): 56 data bytes
64 bytes from 172.16.1.4: icmp_seq=0 ttl=255 time=0.1 ms
64 bytes from 172.16.1.4: icmp_seq=1 ttl=255 time=0.2 ms
64 bytes from 172.16.1.4: icmp_seq=2 ttl=255 time=0.1 ms
64 bytes from 172.16.1.4: icmp_seq=3 ttl=255 time=0.1 ms

--- sun.vnuhcm.edu.vn ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.2 ms
[tnminh@proxy tnminh]$
```

Nếu 2 máy có thể liên lạc được với nhau, đồng thời chúng ta sẽ có trả lời cùng với thời gian trả lời để cho biết sự thông thoáng về mạng giữa 2 máy. Có thể nói, **ping** phải chạy trước tiên trước tất cả các hoạt động mạng khác.

➤ **Lệnh Traceroute.** Đây cũng là lệnh cho phép chẩn đoán hoạt động của mạng. Cú pháp của lệnh giống như lệnh **ping** nhưng kết quả không chỉ dừng ở sự trả lời mà còn chỉ ra các thiết bị trung gian nằm giữa 2 máy.

```
# tnminh@nefertiti ~ > traceroute 203.162.44.33
traceroute to 203.162.44.33 (203.162.44.33): 1-30 hops, 38 byte packets
 1 makeda.pasteur.fr (157.99.64.3), 1.66 ms, 1.66 ms, 1.66 ms
 2 418.ATM4-0.GW21.Defense.OLEANE.NET (195.25.28.149), 5.0 ms, 4.17 ms, 4.17 ms
 3 FastEth0-0.GW16.Defense.OLEANE.NET (195.25.25.208), 4.17 ms, 4.17 ms, 4.17s
 4 100.ATM6-1.GW2.Telhouse.OLEANE.NET (194.2.3.245), 5.0 ms, 5.0 ms, 5.0 ms
 5 Teleglobe.parix.net (198.32.246.26), 5.84 ms, 6.65 ms, 4.99 ms
 6 195.219.14.161 (195.219.14.161), 5.82 ms, 7.50 ms, 17.5 ms
 7 if-0-1.core1.London.Teleglobe.net (195.219.96.89), 20.8 ms, 18.3 ms, 18.3 ms
 8 if-10-3.core2.NewYork.Teleglobe.net (207.45.222.5), 98.4 ms (ttl=246!), 84. ms (ttl=246!), 85.1 ms (ttl=246!)
 9 if-3-0.core1.NewYork.Teleglobe.net (207.45.223.177), 84.2 ms (ttl=245!), 89.8 ms (ttl=245!), 86.5 ms (ttl=245!)
10 if-5-1.core1.LosAngeles.Teleglobe.net (207.45.223.62), 166 ms (ttl=244!), 13 ms (ttl=244!), 175 ms (ttl=244!)
11 Teleglobe.net (207.45.193.66), 285 ms, 287 ms, 288 ms
12 gsr-ote2.kddnet.ad.jp (203.181.96.129), 283 ms (ttl=246!), 284 ms (ttl=246!, 287 ms (ttl=246!)
13 cm-ote6.kddnet.ad.jp (203.181.96.182), 290 ms (ttl=245!), 283 ms (ttl=245!), 285 ms (ttl=245!)
14 210.132.93.210 (210.132.93.210), 849 ms (ttl=241!), 807 ms (ttl=241!), 970 s (ttl=241!)
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 2: HỆ ĐIỀU HÀNH UNIX

*15 202.167.121.195 (202.167.121.195), 905 ms !H 203.162.3.42 (203.162.3.42), 1
88 ms (ttl=242!)*

Chú ý là khi chúng ta thử kết nối với một máy ở xa trong Internet, do nhiều mạng áp dụng các bức tường lửa (firewall), nhiều khi lệnh ping và traceroute không chạy nhưng trên thực chất là mạng vẫn thông.

Chương 3 : CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

3.1. Domain Name Service (DNS):

Đây là dịch vụ cơ bản đầu tiên và quan trọng nhất của Internet. DNS là quan trọng vì nếu DNS hoạt động sai hoặc không hoạt động, toàn bộ phần mạng Internet liên quan sẽ bị tê liệt hoàn toàn. Hiểu rõ DNS rất quan trọng với quản trị viên máy chủ có kết nối Internet. Nó cho phép quản trị viên tìm ra nhanh chóng các nguyên nhân của các trục trặc trên mạng.

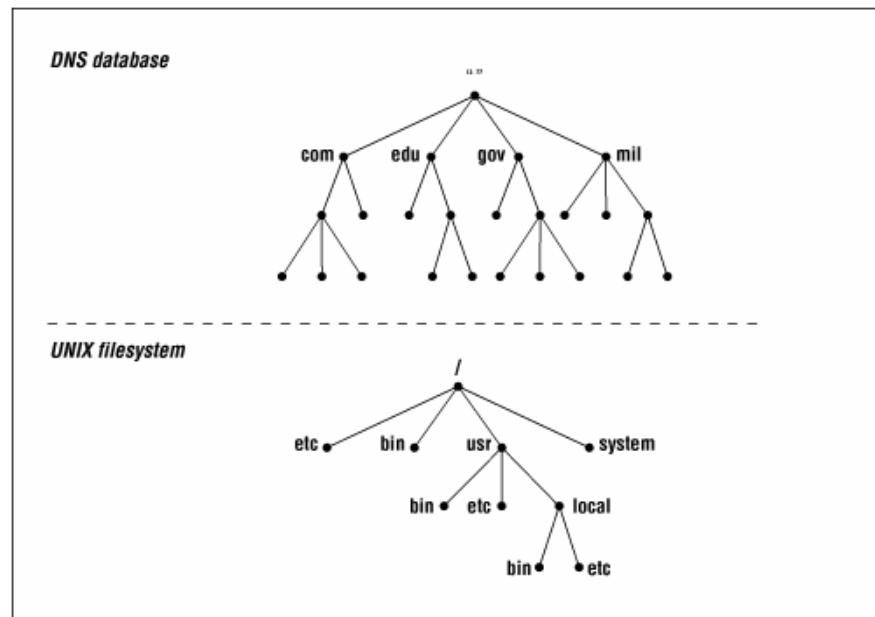
DNS nói một cách đơn giản là dịch vụ cho phép ánh xạ , chuyển đổi tên của một hệ thống nối Internet ra địa chỉ IP của nó. Nguyên nhân của sự tồn tại DNS là do con người có thói quen đặt tên cho các trang thiết bị mà các trang thiết bị thì lại chỉ có thể dùng số để liên lạc với nhau. Vào những thời kỳ đầu tiên của Internet, người ta lập bảng về mối liên hệ giữa tên và địa chỉ IP và cài đặt trên một máy tính để tất cả cùng tham khảo. Nhưng với sự phát triển quá nhanh của Internet, bảng này phát triển nhanh chóng và không một máy nào có thể hoàn thành nổi nhiệm vụ tuy đơn giản nhưng lại rất quan trọng này. Hơn nữa, mỗi thay đổi dù ở đâu cũng phải thông qua server trung tâm. Điều này trở nên không thể chấp nhận được vì luôn có thay đổi trên Internet. Một giải pháp được cộng đồng Internet chấp nhận là chia toàn bộ không gian các địa chỉ IP và tên ra thành các nhóm logic nhỏ hơn . Mỗi nhóm có quyền tổ chức thông tin của các máy của mình.

Các khái niệm cơ bản của DNS là :

- **Domain.** Tên logic của một mạng hay một tổ chức. Ví dụ **unc.edu** là tên miền gốc của Đại học Bắc Carolina, **edu.vn** là tên miền của các tổ chức giáo dục của Việt nam.
- **Domain name.** Phần của tên của máy biểu thị miền chứa máy đó. Ví dụ trong **sunsite.unc.edu** phần domain name là **unc.edu**.
- **Host, Node .** Máy tính nối mạng
- **Name server hay DNS server.** Máy tính có dịch vụ DNS chạy trên đó cho phép chuyển đổi giữa tên và địa chỉ IP. Người ta còn thường dùng các tên này để chỉ tiến trình làm chức năng phân giải tên.
- **Resolve.** Hành động chuyển đổi giữa tên và địa chỉ IP
- **Resolver.** Chương trình hay hàm thư viện cho phép lấy ra các thông tin DNS từ các DNS server.

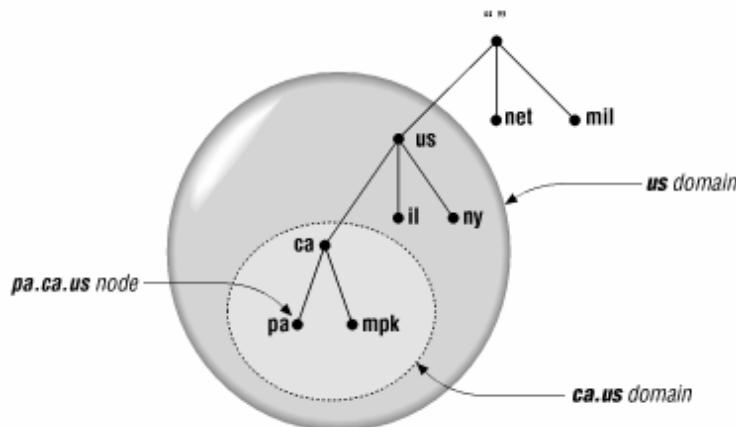
- **Reverse Resolution.** Quá trình chuyển đổi từ địa chỉ IP ra tên.
- **DNS client.** Chương trình đưa ra nhu cầu cần chuyển đổi tên thành địa chỉ số IP, ví dụ Internet Explorer, chương trình tạo và gửi Email ...
- **DNS server.** Chương trình trả lời các câu hỏi của DNS Client nhằm giúp client biết địa chỉ đối với một tên tương ứng hoặc ngược lại.

Hình sau đây minh họa sự phân chia thành các miền con và so sánh sự tương tự với cấu trúc tập tin của HDH Unix



Hình 3.1: CSDL liệu của hệ thống DNS và cấu trúc thư mục của HDH Unix

Với sự phân chia thành các miền như vậy, Internet cũng phân chia trách nhiệm và quyền hạn quản lý đến các vùng nhỏ hơn.

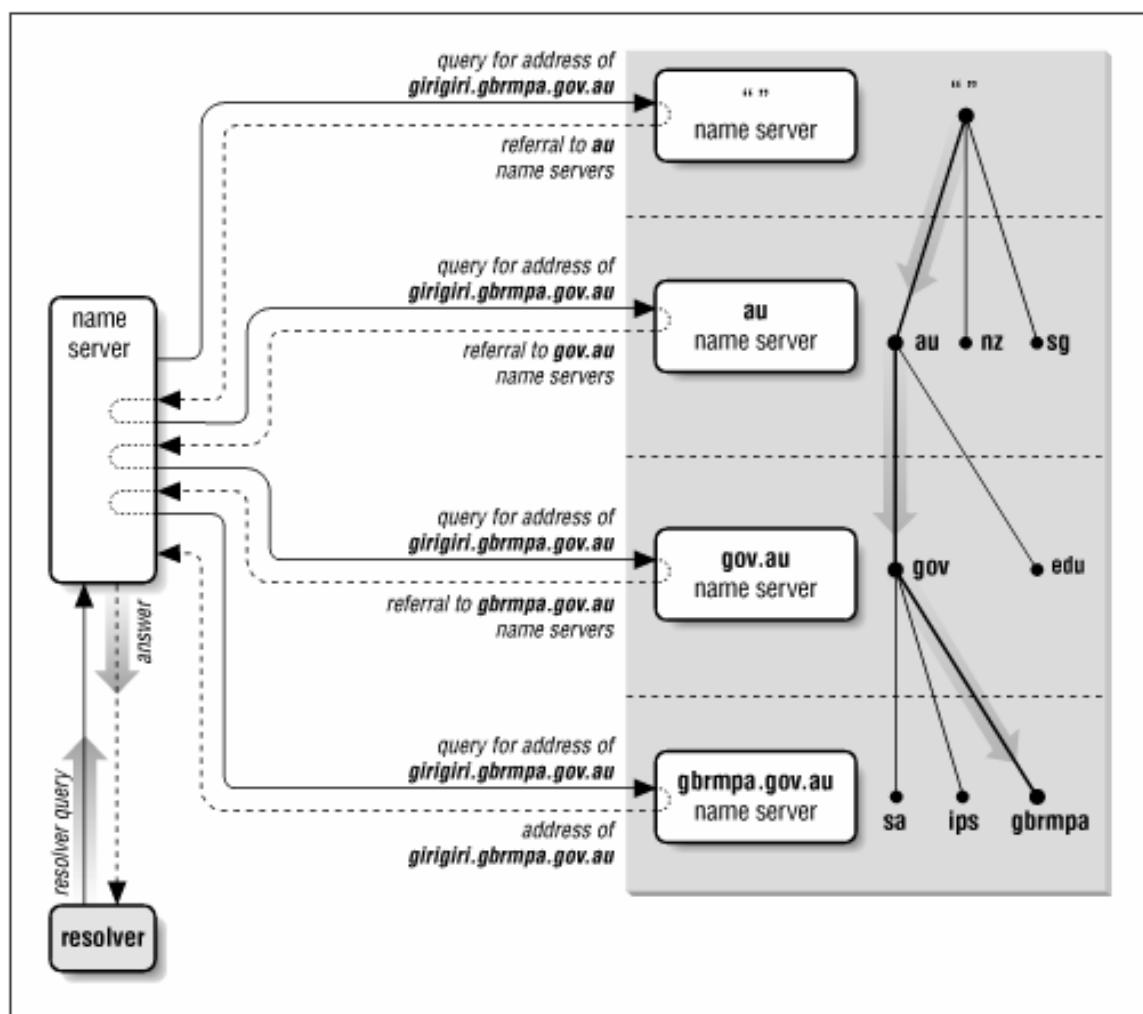


Hình 3.2: Cây phân chia sự ủy quyền các miền con

Các miền con :

Hình trên cho thấy dưới miền gốc, ký hiệu bằng dấu chấm “.” Có các miền con **us**, **net**, **mil**. Dưới **us** có **ca**, **il**, **ny**. Dưới **ca** có **pa** và **mpk**. Khi đó, miền **ca.us** sẽ chịu trách nhiệm về các miền con **pa** và **mpk** của mình. **Ca** có quyền tạo ra các miền con mới nếu cần thiết mà không phải khai báo gì với **us**, là miền trên của nó. Đối với các miền con **pa** và **mpk** của mình, **ca** cũng không “bao cấp”. Chúng ta sẽ thấy **ca** chỉ lưu các thông tin cơ bản nhất về các miền con của nó, còn lại nói chung nó để cho các miền con “tự trị”.

Sau đây, chúng ta minh họa quá trình phân giải địa chỉ của một DNS client đối với DNS server của mình về địa chỉ máy **girigiri.gbrmpa.gov.au**



Hình 3.3: Quá trình phân giải grigiri.gbrmpa.gov.au trên mạng Internet

Mô hình trên cho thấy rất rõ các đặc điểm sau đây của tổ chức miền và phân giải địa chỉ trên Internet:

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

- Một máy bất kỳ (DNS client) đều phải phân giải địa chỉ thông qua một DNS server
- DNS server chịu trách nhiệm đưa lại câu trả lời cuối cùng đối với DNS Client
- Quá trình phân giải tên được thực hiện từ phải qua trái, tức là tên miền cuối cùng (cao nhất) dần xuống các miền bên trái hơn của tên.
- Các DNS server chỉ trả lời về các thông tin tối thiểu là địa chỉ của các DNS server của miền con mà nó phải quản trị vì nó chỉ có dữ liệu như vậy về miền con của nó.
- Cuối cùng phải có một DNS server “biết” về máy mà chúng ta muốn truy vấn địa chỉ

Như vậy bước đầu tiên, một máy nối vào Internet, không phụ thuộc vào việc nó có chạy hay không DNS server, phải được cấu hình resolver, tức là chỉ ra cách thức hành động khi có yêu cầu phân giải địa chỉ. Resolver được cấu hình qua tập tin **/etc/host**

```
[root@pasteur tnminh]# more /etc/host.conf
order hosts,bind
multi on
```

Dòng thứ nhất của /etc/host.conf cho biết khi có yêu cầu phân giải tên, resolver sẽ xem xét đầu tiên tập tin /etc/hosts sau đó đến sử dụng DNS server (bind).

Dòng thứ hai cho phép một host có nhiều địa chỉ IP trong tập tin /etc/hosts.

Tập tin **/etc/hosts** chính là tiền thân của dịch vụ DNS. Hiện nay, /etc/hosts chỉ còn thường lưu các địa chỉ của mạng nội bộ hay dùng tối đa đối với một máy. Khi yêu cầu phân giải vượt qua khả năng trả lời của **/etc/hosts** từ khóa **bind** chỉ ra cần phải sử dụng dịch vụ DNS. BIND là viết tắt của Berkeley Internet Name Domain và một triển khai rộng rãi nhất của dịch vụ DNS hiện nay.

Khi đó, resolver cần thông tin tiếp theo về DNS server. Thông tin này lưu trữ trong tập tin **/etc/resolv.conf**. Tập tin này kiểm tra cách resolver sử dụng DNS để phân giải địa chỉ. Nó quyết định DNS server cụ thể cần phải truy vấn và cách bổ sung phần domain cho phần tên của máy. Ví dụ một tập tin /etc/resolv.conf

```
[root@pasteur tnminh]# more /etc/resolv.conf
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

```
search bvt.hcm
nameserver 192.168.2.10
[root@pasteur tnminh]#
```

Dòng đầu tiên cho phép resolver không chỉ phân giải tên như chương trình client yêu cầu, mà trong trường hợp phân giải không thành công, tiếp tục thử phân giải tên với phần domain tiếp nối sau. Ví dụ chúng ta muốn tìm địa chỉ máy **foo**. Nếu quá trình phân giải foo không thành công, resolver sẽ thử phân giải **foo.bvt.hvm**. Dòng tiếp theo là địa chỉ của name server cần phải truy vấn. Nhớ rằng địa chỉ của name server là số IP chứ không phải là tên, vì nếu ngược lại, ai sẽ là người phân giải tên cho máy làm nhiệm vụ phân giải tên?

Bây giờ chúng ta sẽ chuyển qua xem xét đến cấu hình của bản thân name server. Chương trình server của DNS name server là một chương trình daemon named (đọc là nêm đê). Named thường được khởi động ngay từ đầu cùng với khởi động của hệ thống. Thường thì named được chạy thông qua một script trong /etc/rc.d/rc3.d/SXXnamed . Trong quá trình khởi động named đọc các tập tin dữ liệu rồi chờ các yêu cầu phân giải qua cổng xác định trong tập tin /etc/service (thông thường là cổng 53). Named dùng đầu tiên là giao thức UDP để phân giải tên, nếu phân giải bằng UDP không có kết quả, named sẽ dùng TCP sau đó.

Tập tin đầu tiên được named tham chiếu là /etc/named.conf. Nội dung tập tin này của Linux Redhat 6.0 được cài mặc định là

```
options {
    directory "/var/named";
};

zone "." {
    type hint;
    file "root.hints";
};

zone "0.0.127.in-addr.arpa" {
    type master;
    file "pz/127.0.0";
};
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

Mở đầu là từ khóa **options** cho phép nhập các tùy chọn (options) toàn cục. *directory "/var/named"*; cho biết là các tập tin sau đây sẽ là tương đối đối với thư mục này.

Ta có thể bổ sung thêm trong phần option dòng lệnh
forwaders {205.15.2.10 ; 193.214.2.12;};

Khi đó, DNS server của chúng ta sẽ tham chiếu các name server **205.15.2.10 ; 193.214.2.12** mỗi khi nó không tìm thấy câu trả lời trong dữ liệu mà nó có .

Sau phần tham số toàn cục options, ta thấy các khối

```
zone "tên_zone" {  
    type master (hoặc slave hoặc hint);  
    file "tên_tập_tin";  
};
```

liên tiếp nhau.

Đối với mỗi domain, chúng ta cần 2 tập tin dữ liệu. Tập tin thứ nhất lưu trữ các dữ liệu liên quan đến phân giải “xuôi” từ name sang IP và tập tin thứ hai để phân giải “ngược” từ IP ra name. Trừ miền “.” có tính chất giúp đỡ là có tập tin cache đặc biệt

```
;  
; There might be opening comments here if you already have this file.  
; If not don't worry.  
;
```

```
.          6D IN NS      G.ROOT-SERVERS.NET.  
.          6D IN NS      J.ROOT-SERVERS.NET.  
.          6D IN NS      K.ROOT-SERVERS.NET.  
.          6D IN NS      L.ROOT-SERVERS.NET.  
.          6D IN NS      M.ROOT-SERVERS.NET.  
.          6D IN NS      A.ROOT-SERVERS.NET.  
.          6D IN NS      H.ROOT-SERVERS.NET.  
.          6D IN NS      B.ROOT-SERVERS.NET.  
.          6D IN NS      C.ROOT-SERVERS.NET.  
.          6D IN NS      D.ROOT-SERVERS.NET.  
.          6D IN NS      E.ROOT-SERVERS.NET.  
.          6D IN NS      I.ROOT-SERVERS.NET.  
.          6D IN NS      F.ROOT-SERVERS.NET.
```

G.ROOT-SERVERS.NET.	5w6d16h IN A	192.112.36.4
J.ROOT-SERVERS.NET.	5w6d16h IN A	198.41.0.10
K.ROOT-SERVERS.NET.	5w6d16h IN A	193.0.14.129
L.ROOT-SERVERS.NET.	5w6d16h IN A	198.32.64.12

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

M.ROOT-SERVERS.NET.	5w6d16h IN A	202.12.27.33
A.ROOT-SERVERS.NET.	5w6d16h IN A	198.41.0.4
H.ROOT-SERVERS.NET.	5w6d16h IN A	128.63.2.53
B.ROOT-SERVERS.NET.	5w6d16h IN A	128.9.0.107
C.ROOT-SERVERS.NET.	5w6d16h IN A	192.33.4.12
D.ROOT-SERVERS.NET.	5w6d16h IN A	128.8.10.90
E.ROOT-SERVERS.NET.	5w6d16h IN A	192.203.230.10
I.ROOT-SERVERS.NET.	5w6d16h IN A	192.36.148.17
F.ROOT-SERVERS.NET.	5w6d16h IN A	192.5.5.241

Đây thực chất là địa chỉ IP của các name server gốc (root) của Internet.
Ví dụ như đối với miền **land-5.com** ta cần có

```
zone "land-5.com" {
    type master;
    file "zone/land-5.com";
};

zone "177.6.206.in-addr.arpa" {
    type master;
    file "zone/206.6.177";
```

Chú ý các viết cú pháp *177.6.206.in-addr.arpa* cho tên của miền phân giải ngược IP ra name.

Sau đây ta sẽ xem xét đến cấu trúc tập tin */var/named/zone/land-5.com*

```
@   IN  SOA  land-5.com. root.land-5.com. (
        199609206      ; serial, todays date + todays serial #
        8H            ; refresh, seconds
        2H            ; retry, seconds
        1W            ; expire, seconds
        1D )          ; minimum, seconds
    NS  land-5.com.
    NS  ns2.psi.net.
    MX  10 land-5.com. ; Primary Mail Exchanger
    TXT "LAND-5 Corporation"
localhost  A  127.0.0.1
router     A  206.6.177.1
land-5.com. A  206.6.177.2
ns          A  206.6.177.3
www         A  207.159.141.192
ftp          CNAME land-5.com.
mail         CNAME land-5.com.
news         CNAME land-5.com.
funn         A  206.6.177.2
;
```

```
; Workstations
;
ws-177200    A    206.6.177.200
              MX   10 land-5.com. ; Primary Mail Host
ws-177201    A    206.6.177.201
              MX   10 land-5.com. ; Primary Mail Host
ws-177202    A    206.6.177.202
```

ký tự “@” đầu tiên thay cho miền *land-5.com*; IN là Internet ; SOA là Start Of Authority; tiếp nối bởi tên DNS server và địa chỉ người chịu trách nhiệm. Chú ý là trong địa chỉ email của người chịu trách nhiệm, dấu @ quen thuộc được thay bằng dấu chấm “.”. Sau các tên miền có dấu chấm “.” ở cuối. Trong tất cả các tập tin dữ liệu của DNS, những tên không kết thúc bởi dấu chấm sẽ được DNS server thêm vào bởi tên miền tương ứng của tập tin đó . Ví dụ đây là tập tin ứng với miền **land-5.com**, **funn** sẽ được bổ sung thêm thành **funn.land-5.com**.

Sau phần ngoặc đơn với 5 số miêu tả số serie và các thông số thời gian của thông tin, bắt đầu các dòng (record) dữ liệu. **Khoảng trắng** ở đầu dòng tương đương với tên miền (như dấu @), **NS** ám chỉ record dạng nameserver. **MX** là mail exchange, dùng để chỉ ra máy chịu trách nhiệm nhận thư điện tử cho domain này. Số 10 là mức độ ưu tiên cho mail server này. Độ ưu tiên sẽ càng cao nếu số càng nhỏ . **A** là viết tắt của Address, sẽ tiếp theo bởi một địa chỉ IP. **CNAME** là canonical name . Với CNAME ta có thể gán cho máy biệt danh tùy ý tiện cho việc sử dụng. Các dòng bắt đầu bởi ; là các chú thích.

Ví dụ tập tin dùng cho phân giải ngược */var/named/zone/206.6.177*

```
@           IN   SOA  land-5.com. root.land-5.com. (
                      199609206      ; Serial
                      28800       ; Refresh
                      7200        ; Retry
                      604800     ; Expire
                      86400)     ; Minimum TTL
                      NS   land-5.com.
                      NS   ns2.psi.net.

;
; Servers
;

1   PTR   router.land-5.com.
2   PTR   land-5.com.
2   PTR   funn.land-5.com.

;
; Workstations
;

200  PTR   ws-177200.land-5.com.
```

201	PTR	ws-177201.land-5.com.
202	PTR	ws-177202.land-5.com.

Cấu trúc tập tin `/var/named/zone/206.6.177` có phần đầu giống hệt như tập tin phân giải xuôi. Chỉ có từ khóa PTR = Pointer là khác.

Việc cấu hình các dữ liệu của name server cần rất thận trọng vì nhiều khi lỗi của nó rất khó tìm. Mỗi khi chúng ta thay đổi dữ liệu, cần phải khởi động lại named bằng cách sử dụng kill -9 named_PID để dừng named rồi khởi động lại bằng cách nhập dòng lệnh named. Tập tin `/var/log/messages` có thể giúp đỡ nhiều để tìm ra lỗi nếu named không hoạt động theo ý chúng ta muốn. Để thử hoạt động của quá trình phân giải tên, Linux có lệnh **nslookup** với nhiều tính năng rất mạnh. Xem manpage của nslookup để biết cách sử dụng.

3.2. Dịch vụ mail, ftp :

Đây là 2 dịch vụ cơ bản của Internet. Khi chúng ta cài đặt Linux server cho một mạng vừa và nhỏ, chương trình cài đặt thực hiện tất cả các cấu hình cho chúng ta và cấu hình mặc định hoạt động khá tốt.

Redhat sử dụng **wu-ftp**, một phiên bản của Wasinton University, thay cho ftp cổ điển. Với phiên bản này anonymous ftp được cài đặt luôn cho chúng ta. Lợi thế của việc sử dụng **wu-ftp** là chúng ta không phải tự cấu hình dịch vụ này, nhất là dịch vụ ftp cấu hình không chuẩn là một sơ hở đáng kể ảnh hưởng đến an toàn của server.

➤ Dịch vụ Email :

Thư điện tử, Electronic mail, Email, là dịch vụ có thể nói là quan trọng nhất đối với người sử dụng của Internet. Do tính phổ cập của Email, việc cấu hình tốt Mail server, tạo điều kiện cho người sử dụng có thể trao đổi Email là công việc đầu tiên và quan trọng nhất của người quản trị. Một cấu hình sai Email có thể dẫn đến tình trạng không gửi/nhận được thư, hoặc tệ hơn là mất thư mà không có phản hồi. Hoạt động của dịch vụ mail gắn rất chặt chẽ với cấu hình của DNS.

Chúng ta thử hình dung quá trình gửi mail để hiểu về cơ chế hoạt động của hệ thống Email.

Đầu tiên, chúng ta phải có một chương trình cho phép chúng ta soạn thảo mail. Có rất nhiều chương trình thực hiện nhiệm vụ này : Internet Explorer, Eudora, Netscape cho Windows; eml, netscape, mail cho Unix, mới nhất hiện nay là Mozilla Firefox ... Các chương trình đầu tiên cho phép chúng ta đánh địa

chỉ Email của người nhận. Địa chỉ đó ngày nay có dạng recipient_name@domain_name.top_domain , ví dụ như foo@cisco.com. Sau đó chúng ta soạn thảo nội dung thư và gửi đi bằng một lệnh hay một nhấp chuột. Khi đó, chương trình mail client sẽ theo cấu hình mà chúng ta đã làm, tìm một SMTP server, outgoing server. SMTP là viết tắt của Simple Mail Transfer Protocol và server sử dụng giao thức SMTP được gọi là SMTP server. Người ta còn thường quen dùng là mail server. Khi chúng ta khai báo SMTP server chúng ta thường dùng tên và như vậy chúng ta phải sử dụng DNS server mà máy chúng ta phải khai báo từ trước để nhờ phân giải và tìm địa chỉ IP tương ứng. Sau khi tìm ra địa chỉ IP của SMTP server, chương trình mail của chúng ta sẽ thực hiện một kết nối TCP/IP với SMTP server vào cổng 25, là cổng quy định cho SMTP server. Hai tiến trình mail client và mail server sẽ trao đổi thông tin với nhau thông qua SMTP protocol. Nếu mọi việc thông suốt, email của chúng ta sẽ được chấp nhận lưu trữ trên SMTP server và chương trình mail client của chúng ta kết thúc phiên làm việc. Công việc tiếp theo là SMTP server của chúng ta tìm cách gửi mail của chúng ta tới người nhận. Để làm việc này, SMTP server của chúng ta thực hiện 2 thao tác :

- Tìm mail server của người nhận của email của chúng ta
- Gửi email của chúng ta đến mail server của người nhận trong email của chúng ta

Thao tác đầu tiên hoàn toàn dựa vào DNS servers. Cụ thể là SMTP của chúng ta sẽ đóng vai trò một DNS client để hỏi DNS server của miền của chúng ta xem “ai là mail server của miền cisco.com ?” Quá trình tra hỏi này đưa đến việc tìm ra một record có dạng

cisco.com. IN MX 10 mailserver.cisco.com.

nằm trong CSDL của một DNS server nào đó, thường là DNS server của miền cisco.com. Nếu quá trình này không thành công, thư của chúng ta sẽ không gửi đi được và chúng ta sẽ nhận được một thông báo trả lời rằng email của chúng ta không được vì “host unknown”. Nếu ngược lại, SMTP của chúng ta sẽ mở một kết nối TCP/IP đến mailserver.cisco.com vào cổng 25 để gửi email của chúng ta. Lúc này SMTP của chúng ta đóng vai trò một mail client. Giao thức SMTP lại được sử dụng để chuyển thư trong khâu này. Nếu mọi thứ thành công, email của chúng ta sẽ được lưu trữ trên mailserver.cisco.com và người nhận foo sẽ phải kết nối với mailserver.cisco.com để lấy thư về máy của mình và đọc thư.

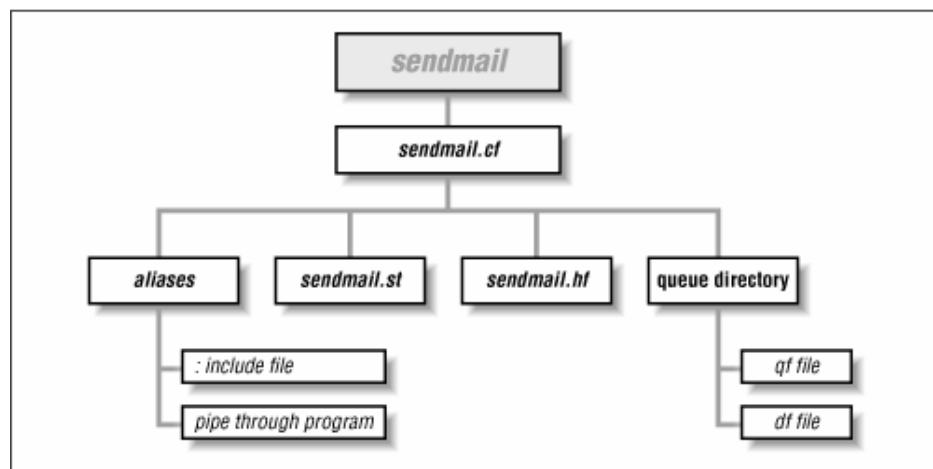
GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

Trên đây là miêu tả một quá trình gửi mail điển hình trên Internet. Trên thực tế, quá trình này có thể phức tạp và thay đổi khá nhiều phụ thuộc vào cấu hình của từng mạng. Đó chính là yếu tố làm phức tạp hóa rất nhiều hệ thống Email và khó khăn đối với công tác quản trị dịch vụ Email. Các chúng ta cũng nhận thấy có ít nhất 4 máy tính tham gia vào quá trình chuyển mail, nhiều lần DNS server tham gia vào và nếu hệ thống DNS server không chạy hoàn hảo, chúng ta không thể gửi Email được.

Có nhiều chương trình SMTP server, nhưng Sendmail có lẽ là chương trình SMTP server nổi tiếng nhất trên Unix từ lâu nay bởi tính năng mạnh và cũng bởi tính phức tạp của nó. Chương trình Sendmail được viết bởi Eric Allman khi ông là một sinh viên của University of California at Berkeley vào năm 1979. RedHat Linux có 2 chương trình mail server là smail và sendmail. Nhìn chung smail thích hợp cho một mạng đơn giản, còn sendmail thì có thể dùng cho cả hai. Trong khuôn khổ bài giảng này, chúng ta sẽ nghiên cứu chương trình sendmail.

Chương trình sendmail có thể được gọi lên bộ nhớ bởi 2 cách. Cách thứ nhất là sendmail được gọi lên bởi chương trình mail client, ví dụ như chương trình cùng tên mail. Khi đó sendmail sẽ mở một kết nối để gửi mail đi. Đây là cấu hình sendmail nếu máy của chúng ta không phải là SMTP server. Cách thứ hai là sendmail được hoạt động theo kiểu daemon, tức là thường trú trên bộ nhớ. Khi đó, daemon sendmail “nghe” sau cổng 25 các kết nối đến. Mỗi khi có kết nối đến cổng 25, sendmail daemon sinh ra một tiến trình sendmail con để tiếp nhận kết nối này, còn bản thân mình thì tiếp tục chờ đợi các kết nối khác. Với lệnh netstat -n ta có thể hiển thị các kết nối đang trong thực hiện. Sendmail sử dụng các tập tin cấu hình và thư mục như sau



Hình 3.4: Các kết nối khi thực hiện Sendmail

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

Đầu tiên, Sendmail sử dụng tập tin cấu hình /etc/sendmail.cf mỗi khi được gọi lên bộ nhớ. Tập tin này rất thích hợp cho các công tác của sendmail nhưng cực kỳ khó hiểu đối với người đọc. Ví dụ như đoạn sau đây của sendmail.cf

```
R$- $@ ${HUB} user -> user@hub  
R$-@$w ${HUB} user@local -> user@hub
```

Đây là một nhược điểm đồng thời là một ưu điểm của sendmail vì nó cho phép cấu hình sendmail cực kỳ uyển chuyển và thỏa mãn các yêu cầu dù éo le nhất của một mail server. Nếu chúng ta chưa một lần phải “võ đầu” bởi những ký tự á rập này thì chúng ta chưa phải là quản trị viên thực thụ.

Trong tập tin sendmail.cf có một số trường quan trọng là

```
# Alias for this host  
Cw citd.edu.vn. training.vnedu.net  
Cw localhost pascal.citd.edu.vn.
```

Dòng thứ 2 xác định rằng tất cả các email với địa chỉ *user@ citd.edu.vn, training.vnedu.net* là thuộc về máy mà chương trình sendmail đang chạy, cần phải đưa về cho chương trình chuyển mail trên máy local và phải thử xem *user* là có tồn tại trên máy này không. Tất cả những mai với phần domain ngoài **Cw** đều được coi là cho miền ngoài và phải chuyển đi qua mạng bằng sendmail.

```
# Smart host  
DSvnuserv.vnuhcm.edu.vn  
# Use this mailer to reach the Smart host  
DNsmtp
```

Dòng thứ 2 của ví dụ trên chỉ ra rằng với tất cả các mail không local, chỉ cần chuyển đến trạm mail trung chuyển (mail relay) và tên của mail relay là chuỗi ký tự nằm sau DS.

Để thử xem sendmail có phân giải địa chỉ và chuyển thư đúng theo ý định của mình hay không, chúng ta có thể dùng lệnh **sendmail -bt** hoặc **mail -v địa chỉ**

```
[tnminh@pascal tnminh]$ /usr/sbin/sendmail -bt  
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)  
Enter <ruleset> <address>  
> 3,0 a@citd.edu.vn  
rewrite: ruleset 3 input: a @ citd . edu . vn  
rewrite: ruleset 96 input: a < @ citd . edu . vn >
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

```
rewrite: ruleset 96 returns: a < @ citd . edu . vn . >
rewrite: ruleset 3 returns: a < @ citd . edu . vn . >
rewrite: ruleset 0 input: a < @ citd . edu . vn . >
rewrite: ruleset 196 input: a < @ citd . edu . vn . >
rewrite: ruleset 196 returns: a < @ citd . edu . vn . >
rewrite: ruleset 98 input: a < @ citd . edu . vn . >
rewrite: ruleset 98 returns: a < @ citd . edu . vn . >
rewrite: ruleset 195 input: a < @ citd . edu . vn . >
rewrite: ruleset 195 returns: $# local $: a
rewrite: ruleset 0 returns: $# local $: a
>
> 3,0 a@yahoo.com
rewrite: ruleset 3 input: a @ yahoo . com
rewrite: ruleset 96 input: a < @ yahoo . com >
rewrite: ruleset 96 returns: a < @ yahoo . com . >
rewrite: ruleset 3 returns: a < @ yahoo . com . >
rewrite: ruleset 0 input: a < @ yahoo . com . >
rewrite: ruleset 196 input: a < @ yahoo . com . >
rewrite: ruleset 196 returns: a < @ yahoo . com . >
rewrite: ruleset 98 input: a < @ yahoo . com . >
rewrite: ruleset 98 returns: a < @ yahoo . com . >
rewrite: ruleset 195 input: a < @ yahoo . com . >
rewrite: ruleset 90 input: < yahoo . com > a < @ yahoo . com . >
rewrite: ruleset 90 input: yahoo . < com > a < @ yahoo . com . >
rewrite: ruleset 90 returns: a < @ yahoo . com . >
rewrite: ruleset 90 returns: a < @ yahoo . com . >
rewrite: ruleset 95 input: < vnuserv . vnuhcm . edu . vn > a < @ yahoo . com .
>
rewrite: ruleset 95 returns: $# smtp $@ vnuserv . vnuhcm . edu . vn $: a < @
yahoo . com . >
rewrite: ruleset 195 returns: $# smtp $@ vnuserv . vnuhcm . edu . vn $: a < @
yahoo . com . >
rewrite: ruleset 0 returns: $# smtp $@ vnuserv . vnuhcm . edu . vn $: a < @
yahoo . com . >
>
```

```
[tnminh@pascal tnminh]$ mail -v tnminh@vnuserv.vnuhcm.edu.vn
Subject: test
hello
```

```
.
Cc:
tnminh@vnuserv.vnuhcm.edu.vn... Connecting to vnuserv.vnuhcm.edu.vn. via
smtp...
220 vnuserv.vnuhcm.edu.vn ESMTP Sendmail 8.9.3/8.8.7;Tue, 7 Nov 2000
11:37:53 G
MT
>>> EHLO pascal.citd.edu.vn
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

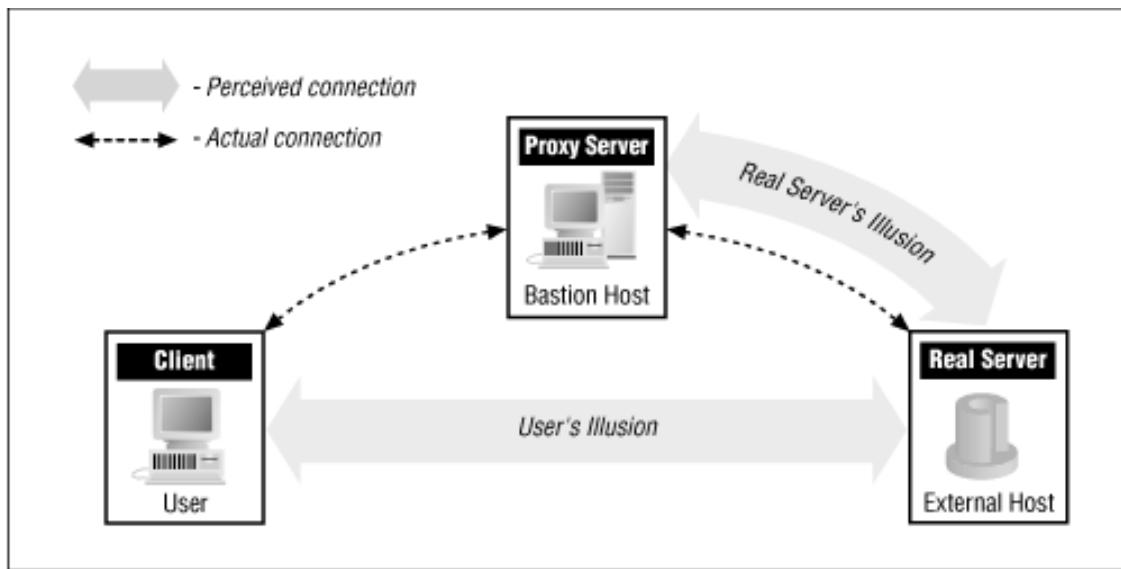
```
250-vnuserv.vnuhcm.edu.vn Hello IDENT:root@mailhost-CITD-16 [172.16.1.10],  
pleas  
ed to meet you  
250-EXPN  
250-VERB  
250-8BITMIME  
250-SIZE 2000000  
250-DSN  
250-ONEX  
250-ETRN  
250-XUSR  
250 HELP  
>>> MAIL From:<tnminh@pascal.citd.edu.vn> SIZE=54  
250 <tnminh@pascal.citd.edu.vn>... Sender ok  
>>> RCPT To:<tnminh@vnuserv.vnuhcm.edu.vn>  
250 <tnminh@vnuserv.vnuhcm.edu.vn>... Recipient ok  
>>> DATA  
354 Enter mail, end with "." on a line by itself  
>>> .  
250 LAA16041 Message accepted for delivery  
tnminh@vnuserv.vnuhcm.edu.vn... Sent (LAA16041 Message accepted for  
delivery)  
Closing connection to vnuserv.vnuhcm.edu.vn.  
>>> QUIT  
221 vnuserv.vnuhcm.edu.vn closing connection  
[tnminh@pascal tnminh]$
```

3.3. Dịch vụ ủy quyền (proxy) :

Proxy cho phép chúng ta chỉ cần một máy hoặc một nhóm nhỏ các máy để trợ giúp cho truy cập Internet cho tất cả các máy của chúng ta. Sử dụng proxy có hai lợi thế quan trọng. Thứ nhất là chúng ta chỉ cần ít, hay một địa chỉ IP chính thức mà lại có thể cho nhiều máy cùng được truy cập Internet. Thứ hai là nếu một trang Web đã được proxy lấy về, nó sẽ được lưu trên đĩa của máy proxy và khi có một yêu cầu khác lấy đúng trang Web đó, proxy không cần phải ra Internet lấy dữ liệu nữa mà lấy thẳng từ trong đĩa cứng của mình và như vậy sẽ tiết kiệm được đường kết nối ra Internet thường rất mắc tiền và bận bịu. Mô hình kết nối sử dụng Proxy có thể được minh họa qua hình sau

Trong mô hình này, máy client phải khai báo trong các chương trình truy cập Internet là cần sử dụng proxy để đi lấy dữ liệu, còn máy server thì không hề hay biết hoặc quan tâm đến việc mình đang nói chuyện với client hay một proxy server.

Trong đĩa CDROM của Linux RedHat luôn có kèm theo chương trình proxy **squid**. Sau đây chúng ta sẽ đề cập đến các thao tác cơ bản để cài đặt và cấu hình một Squid proxy trên Linux.



Hình 3.5: Mô hình kết nối sử dụng Proxy

Cài đặt: Đầu tiên, nên có một số khái niệm về đòi hỏi phần cứng đối với một proxy server

- Tốc độ truy cập đĩa cứng : rất quan trọng vì Squid thường xuyên phải đọc và ghi dữ liệu trên ổ đĩa cứng. Một đĩa SCSI với tốc độ truyền dữ liệu lớn là một ứng cử viên tốt cho nhiệm vụ này.
- Dung lượng đĩa dành cho cache phụ thuộc vào kích cỡ của mạng mà Squid phục vụ. Từ 1 đến 2 Gb cho một mạng trung bình khoảng 100 máy. Tuy nhiên đây chỉ là con số có tính chất vì dụ vì nhu cầu truy cập Internet mới là yếu tố quyết định sự cần thiết của độ lớn của đĩa cứng
- RAM: Rất quan trọng, ít RAM thì Squid sẽ chậm hơn một cách rõ ràng
- CPU : không cần mạnh lắm, khoảng 133MHz là cũng có thể chạy tốt với tải 7 requests/second

Compilers: gcc

Cài đặt Squid với RedHat Linux rất đơn giản. Squid sẽ được cài nếu chúng ta chọn nó trong quá trình cài đặt ngay từ đầu. Hoặc nếu chúng ta đã cài Linux không Squid, chúng ta có thể cài sau qua tiện ích rpm với lệnh `rpm -i tên_gói_Squid`

Khi đó Squid sẽ được cài và chúng ta có thể bước qua phần cấu hình Squid.

Chọn OS:



Hình 3.5: Các distro Linux

Cấu hình squid. Chúng ta cần xác định các thông số sau trước tiên để có thể trả lời các câu hỏi trong quá trình cài đặt:

- http_port 8080. Nếu chúng ta muốn dùng port <1024 để client kết nối với Squid, thì Squid phải có quyền root. Nhìn chung không nên cấu hình như vậy vì vấn đề an ninh hệ thống. Cổng 8080 là một ví dụ phổ biến.
- icp_port 8082. Đây là port cho các Squid nói chuyện với nhau
- cache_mem 128 MB
- Log files, pid file, error messages
- cache_dir /usr/local/squid/cache/ 100 16 256
- cache_mgr tên_admin@địa_chỉ_admin
- cache_effective_user squid
- cache_effective_group squid

Access Control list, ACL. ACL cho phép chúng ta kiểm tra truy cập Web thông qua Squid. Ví dụ như chúng ta có thể ngăn không cho một subnet sử dụng Squid của chúng ta hoặc không cho phép kết nối đến một Web site có nội dung xấu nào đó.

ACL cho nhiều ip

- http_access deny 10.0.1.0/255.255.255.0
- http_access allow 10.0.0.0/255.0.0.0
- icp_access allow 10.0.0.0/255.0.0.0

Đối với các đơn vị lớn ta phân lớp ACL để tiện quản lý

- acl CITD src 172.16.10.1-172.16.10.10/255.255.255.0
- acl DHKT src 172.25.1.1-172.25.1.10/255.255.255.0
- http_access allow CITD
- http_access allow DHKT
- http_access deny all

Ngăn chặn các địa chỉ có nội dung xấu:

- acl forbidden dstdomain xxx.com, convit.org

Đối với một danh sách các địa chỉ bị cấm để tiện quản lý ta cho vào một tập tin text. Lưu ý mỗi địa chỉ cần cấm phải nằm trên một dòng.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 3: CÁC DỊCH VỤ INTERNET CƠ BẢN TRÊN MỘT MÁY CHỦ LINUX

- acl badurl dstdomain “/squid/etc/badurl“
- acl polictic url_regex “/squid/etc/polictic“
- Dstdomain: Destination Domain
- url_regex: url regular expression
- *Time outs*
- dead_peer_timeout 10 seconds
- connect_timeout 120 seconds
- request_timeout 30 seconds
- client_lifetime 1 day
- shutdown_lifetime 30 seconds
- *reference_age 1 year: thời gian mà một object không được tham khảo tới sẽ bị xoá khỏi cache*

Khởi động và dừng Squid

Chạy lần đầu tiên

```
$SQUID_HOME/bin/squid -z -f $SQUID_HOME/etc/squid.conf
```

Chạy squid

```
$SQUID_HOME/bin/squid -D -f $SQUID_HOME/etc/squid.conf
```

Stop squid

```
$SQUID_HOME/bin/squid -k shutdown -f $SQUID_HOME/etc/squid.conf
```

Cấu hình lại squid

```
$SQUID_HOME/bin/squid -k reconfigure-f $SQUID_HOME/etc/squid.conf
```

Chương 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

4.1. Giới thiệu về iptables

Iptables do Netfilter Organization viết ra để tăng tính năng bảo mật trên hệ thống Linux. Iptables cung cấp các tính năng sau:

- Tích hợp tốt với kernel của Linux.
- Có khả năng phân tích package hiệu quả.
- Lọc package dựa vào MAC và một số cờ hiệu trong TCP Header
- Cung cấp chi tiết các tùy chọn để ghi nhận sự kiện hệ thống
- Cung cấp kỹ thuật NAT
- Có khả năng ngăn chặn một số cơ chế tấn công theo kiểu DoS

4.2. Cài đặt iptables

Iptables được cài đặt mặc định trong hệ thống Linux, package của iptables là iptables-version.rpm hoặc iptables-version.tgz ..., ta có thể dùng lệnh để cài đặt package này:

```
$ rpm -ivh iptables-version.rpm đối Red Hat  
$ apt-get install iptables đối với Debian  
- Khởi động iptables: service iptables start  
- Tắt iptables: service iptables stop  
- Tái khởi động iptables: service iptables restart  
- Xác định trạng thái iptables: service iptables status
```

4.3. Cơ chế xử lý package trong iptables

Iptables sẽ kiểm tra tất cả các package khi nó đi qua iptables host, qua trình kiểm tra này được thực hiện một cách tuần tự entry đầu tiên đến entry cuối cùng.

Có ba loại bảng trong iptables:

Mangle table: chịu trách nhiệm biến đổi quality of service bits trong TCP header. Thông thường loại table này được ứng dụng trong SOHO (Small Office/Home Office).

Filter queue: chịu trách nhiệm thiết lập bộ lọc packet (packet filtering), có ba loại built-in chains được mô tả để thực hiện các chính sách về firewall (firewall policy rules).

- Forward chain: Cho phép packet nguồn chuyển qua firewall.
- Input chain: Cho phép những gói tin đi vào từ firewall.
- Output chain: Cho phép những gói tin đi ra từ firewall.

NAT queue: thực thi chức năng NAT (Network Address Translation), cung cấp hai loại built-in chains sau đây:

- Pre-routing chain: NAT từ ngoài vào trong nội bộ. Quá trình NAT sẽ thực hiện trước khi thực thi cơ chế routing. Điều này thuận lợi cho

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

- việc đổi địa chỉ đích để địa chỉ tương thích với bảng định tuyến của firewall, khi cấu hình ta có thể dùng khóa DNAT để mô tả kỹ thuật này.
- **Post-routing chain:** NAT từ trong ra ngoài. Quá trình NAT sẽ thực hiện sau khi thực hiện cơ chế định tuyến. Quá trình này nhằm thay đổi địa chỉ nguồn của gói tin. Kỹ thuật này được gọi là NAT one-to-one hoặc many-to-one, được gọi là Source NAT hay SNAT.
 - **OUTPUT:** Trong loại này firewall thực hiện quá trình NAT.

4.4. Target và Jumps

- **Jump** là cơ chế chuyển một packet đến một target nào đó để xử lý thêm một số thao tác khác.
- **Target** là cơ chế hoạt động trong iptables, dùng để nhận diện và kiểm tra packet. Các target được xây dựng sẵn trong iptables như:
 - **ACCEPT:** iptables chấp nhận chuyển data đến đích.
 - **DROP:** iptables khóa những packet.
 - **LOG:** thông tin của packet sẽ gửi vào syslog daemon iptables tiếp tục xử lý luật tiếp theo trong bảng mô tả luật. Nếu luật cuối cùng không match thì sẽ drop packet. Với tùy chọn thông dụng là `--log-prefix="string"`, tức iptables sẽ ghi nhận lại những message bắt đầu bằng chuỗi “`string`”.
 - **REJECT:** ngăn chặn packet và gửi thông báo cho sender. Với tùy chọn thông dụng là `--reject-with qualifier`, tức qualifier chỉ định loại reject message sẽ được gửi lại cho người gửi. Các loại qualifer sau: `icmp-port-unreachable (default)`, `icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-proto-unreachable`, ...
 - **DNAT:** thay đổi địa chỉ đích của packet. Tùy chọn là `--to-destination ipaddress`.
 - **SNAT:** thay đổi địa chỉ nguồn của packet. Tùy chọn là `--to-source <address>[-address][:<port>-<port>]`
 - **MASQUERADING:** được sử dụng để thực hiện kỹ thuật NAT (giả mạo địa chỉ nguồn với địa chỉ của interface của firewall). Tùy chọn là `[--to-ports <port>[-<port>]]`, chỉ định dãy port nguồn sẽ ánh xạ với dãy port ban đầu.

4.5. Thực hiện lệnh trong iptables

Iptables command Switch	Mô tả
<code>-t <table></code>	Chỉ định bảng cho iptables bao gồm: filter, nat, mangle tables.
<code>-j <target></code>	Nhảy đến một target chain khi packet thỏa luật hiện tại.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

-A	Thêm luật vào cuối iptables chain.
-F	Xóa tất cả các luật trong bảng lựa chọn.
-p <protocol-type>	Mô tả các giao thức bao gồm: icmp, tcp, udp và all
-s <ip-address>	Chỉ định địa chỉ nguồn
-d <ip-address>	Chỉ định địa chỉ đích
-i <interface-name>	Chỉ định “input” interface nhận packet
-o <interface-name>	Chỉ định “output” interface chuyển packet ra ngoài

Bảng 4.1: Bảng mô tả về iptables command Switch

Ví dụ 1: Firewall chấp nhận cho bất kỳ TCP packet đi vào interface eth0 đến địa chỉ 172.28.24.199

```
# iptables -A INPUT -s 0/0 -i eth0 -d 172.28.24.199 -p tcp -j ACCEPT
```

Ví dụ 2: Firewall chấp nhận TCP packet được định tuyến khi nó đi vào interface eth0 và đi ra interface eth1 để đến đích 172.28.2.2 với port nguồn bắt đầu 1024→65535 và port đích 8080

```
# iptables -A FORWARD -s 0/0 -i eth0 -o eth1 -d 172.28.2.2 -p tcp \
--sport 1024:65535 --dport 8080 -j ACCEPT
```

Ví dụ 3: Firewall cho phép gửi icmp echo-request và icmp echo-reply

```
# iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
# iptables -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
```

Ví dụ 4: Chỉ định số lượng yêu cầu phù hợp cho một đơn vị thời gian theo dạng(/second, /minute, /hour, /day)

```
# iptables -A INPUT -p icmp -icmp-type echo-request -m limit --limit 1/s \
-i eth0 -j ACCEPT
```

Ưu điểm của nó là giới hạn được số lượng kết nối, giúp cho ta chống được các cơ chế tấn công như DoS (Denial of Service attack).

Khóa chuyển (Switch)	Mô tả
-m multiport –sport<port,port>	Mô tả nhiều dãy sport, phải cách nhau bằng dấu “,” và dùng tùy chọn –m
-m multiport –dport<port,port>	Mô tả nhiều dãy dport, phải cách nhau bằng dấu “,” và dùng tùy chọn –m
-m multiport –ports<port,port>	Mô tả nhiều dãy port, phải cách nhau

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

	bằng dấu “,” và dùng tùy chọn -m
-m -state<state>	Kiểm tra trạng thái: ESTABLISHED: đã thiết lập connection NEW: bắt đầu thiết lập connection RELATED: thiết lập connection thứ 2(FTP data transfer hoặc ICMP error)

Bảng 4.2: Mô tả một số thông số mở rộng

Ví dụ 5: Firewall chấp nhận TCP packet từ bất kỳ địa chỉ nào đi vào interface eth0 đến địa chỉ 172.28.24.195 qua interface eth1, source port từ 1024→65535 và destination port là 8080 và 443 (dòng lệnh thứ 1). Packet trả về cũng được chấp nhận từ 172.28.2.2 (dòng lệnh thứ 2).

```
# iptables -A FORWARD -s 0/0 -i eth0 -d 172.28.24.195 -o eth1 -p tcp \
--sport 1024:65535 -m multiport --dport 8080,443 -j ACCEPT
# iptables -A FORWARD -d 0/0 -i eth0 -s 172.28.2.2 -o eth1 -p tcp \
-m state --state ESTABLISHED -j ACCEPT
```

4.6. Sử dụng chain tự định nghĩa

Thay vì sử dụng các chain đã được xây dựng trong iptables, ta có thể sử dụng User Defined chains để định nghĩa một chain name mô tả cho tất cả protocol-type cho packet. Ta có thể dùng User Defined chains thay thế chain dài dòng bằng cách sử dụng chain chính chỉ đến nhiều chain con.

Ví dụ 6:

```
# iptables -A INPUT -i eth0 -d 172.28.24.198 -j fast-input-queue
# iptables -A OUTPUT -o eth0 -s 172.28.2.2 -j fast-output-queue
# iptables -A fast-input-queue -p icmp -j icmp-queue-in
# iptables -A fast-output-queue -p icmp -j icmp-queue-out
# iptables -A icmp-queue-out -p icmp --icmp-type echo-request \
-m state --state NEW -j ACCEPT
# iptables -A icmp-queue-in-p icmp --icmp-type echo-reply \
-m state --state NEW -j ACCEPT
```

4.7. Lưu iptables script

Lệnh service iptables save để lưu trữ cấu hình iptables trong file */etc/sysconfig/iptables*. Khi ta khởi động lại thì chương trình iptables-restore sẽ đọc lại file script này và kích hoạt lại thông tin cấu hình. Định dạng của file như sau:

```
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006
*nat
:PREROUTING ACCEPT [4169:438355]
:POSTROUTING ACCEPT [106:6312]
:OUTPUT ACCEPT [22:1332]
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

```
-A PREROUTING -d 172.28.24.199 -i eth0 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080
-A PREROUTING -d 172.28.24.199 -i eth0 -p tcp -m tcp --dport 8888 -j DNAT --to-destination 192.168.1.3:80
-A PREROUTING -i eth0 -p tcp -m tcp --dport 20:21 -j DNAT --to-destination 192.168.1.2:21
-A PREROUTING -i eth0 -p tcp -m tcp --dport 2020:2121 -j DNAT --to-destination 192.168.1.3:21
-A POSTROUTING -o eth0 -j SNAT --to-source 172.28.24.199
COMMIT
# Completed on Thu Nov 9 15:47:54 2006
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006
*filter
:INPUT DROP [4011:414080]
:FORWARD ACCEPT [552:57100]
:OUTPUT ACCEPT [393:43195]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i ! eth0 -m state --state NEW -j ACCEPT
-A FORWARD -d 192.168.1.3 -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
COMMIT
# Completed on Thu Nov 9 15:47:54 2006
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006
*mangle
:PREROUTING ACCEPT [5114:853418]
:INPUT ACCEPT [4416:773589]
:FORWARD ACCEPT [552:57100]
:OUTPUT ACCEPT [393:43195]
:POSTROUTING ACCEPT [945:100295]
COMMIT
# Completed on Thu Nov 9 15:47:54 2006
```

4.8. Phục hồi script khi mất script file

Để có thể phục hồi script khi mất script file. Đầu tiên, ta phải lưu script lại dùng lệnh: *iptables-save > script_du_phong*. Sau đó, ta có thể xem lại *script_du_phong* vừa lưu, dùng lệnh cat *script_du_phong*. Kết quả như sau:

```
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006
*nat
:PREROUTING ACCEPT [4169:438355]
:POSTROUTING ACCEPT [106:6312]
:OUTPUT ACCEPT [22:1332]
-A PREROUTING -d 172.28.24.199 -i eth0 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080
-A PREROUTING -d 172.28.24.199 -i eth0 -p tcp -m tcp --dport 8888 -j DNAT --to-destination 192.168.1.3:80
-A PREROUTING -i eth0 -p tcp -m tcp --dport 20:21 -j DNAT --to-destination 192.168.1.2:21
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

```
-A PREROUTING -i eth0 -p tcp -m tcp --dport 2020:2121 -j DNAT --to-destination  
192.168.1.3:21  
-A POSTROUTING -o eth0 -j SNAT --to-source 172.28.24.199  
COMMIT  
# Completed on Thu Nov 9 15:47:54 2006  
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006  
*filter  
:INPUT DROP [4011:414080]  
:FORWARD ACCEPT [552:57100]  
:OUTPUT ACCEPT [393:43195]  
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT  
-A INPUT -i ! eth0 -m state --state NEW -j ACCEPT  
-A FORWARD -d 192.168.1.3 -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT  
COMMIT  
# Completed on Thu Nov 9 15:47:54 2006  
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006  
*mangle  
:PREROUTING ACCEPT [5114:853418]  
:INPUT ACCEPT [4416:773589]  
:FORWARD ACCEPT [552:57100]  
:OUTPUT ACCEPT [393:43195]  
:POSTROUTING ACCEPT [945:100295]  
COMMIT  
# Completed on Thu Nov 9 15:47:54 2006
```

Sau đó, sửa file *script_du_phong* và nạp lại iptables thông qua lệnh *iptables-restore*

```
# iptables-restore < script_du_phong
```

Cuối cùng, ta dùng lệnh để lưu trữ lại các luật vào file cấu hình:

```
# service iptables save
```

4.9. Load kernel module cần cho iptables

Úng dụng iptables yêu cầu load một số module sau:

- *iptable_nat* module cho NAT.
- *ip_conntrack_ftp* module cần cho FTP support
- *ip_conntrack* module để theo dõi trạng thái của TCP connect.
- *ip_nat_ftp* module cần cho việc load FTP servers sau NAT firewall.

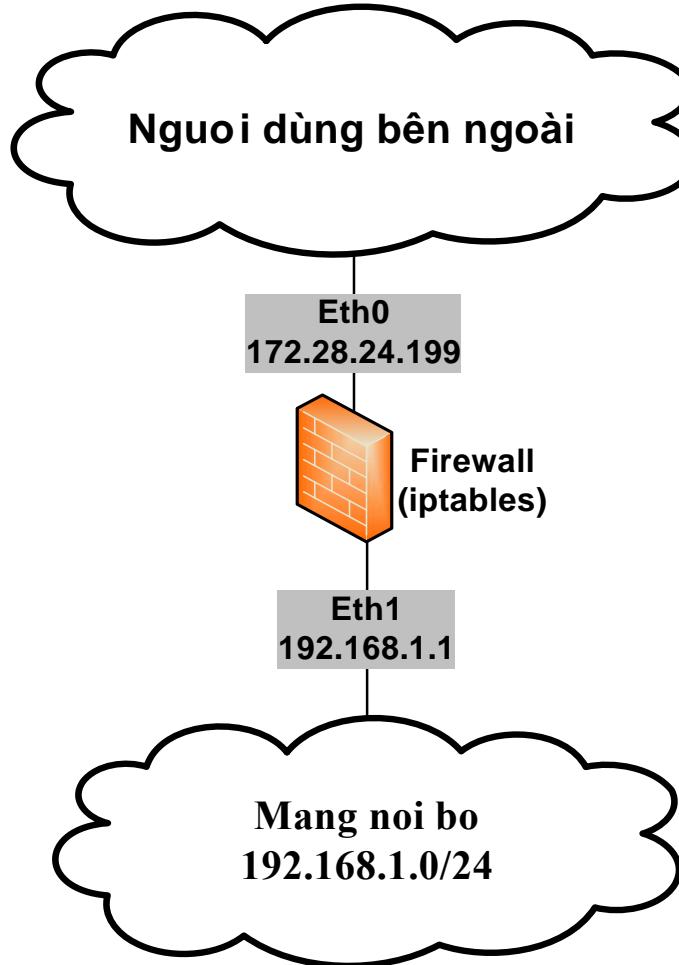
4.10. Một số giá trị khởi tạo của iptables

```
##### Internal-Firewall.sh script
##### Cho phép tự chạy script bằng shell
#!/bin/sh
#### Gán lệnh vào biến
IPTABLES=/sbin/iptables
##### Các giá trị khởi tạo
INTERNAL_LAN="192.168.1.0/24"      # Địa chỉ mạng LAN
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

```
INTERNAL_LAN_INTERFACE="eth1"      # Interface nối đến mạng LAN
INTERNAL_LAN_INTERFACE_ADDR="192.168.1.1" ##Địa chỉ int eth1
EXTERNAL_INTERFACE="eth0"          ## Interface public
EXTERNAL_INTERFACE_ADDR="172.28.24.199"  ## Địa chỉ eth0
$IPTABLES -F FORWARD           ## Xóa các luật của FORWARD chain
$IPTABLES -F INPUT              ## Xóa các luật của INPUT chain
$IPTABLES -F OUTPUT             ## Xóa các luật của OUTPUT chain
$IPTABLES -P FORWARD DROP      ## Mặc định FORWARD chain là DROP
$IPTABLES -P OUTPUT ACCEPT     ## Mặc định OUTPUT chain là ACCEPT
$IPTABLES -P INPUT DROP        ## Mặc định INPUT chain là DROP
#####
## Cho phép tất cả các packet đi vào loopback với tất cả các protocol
$IPTABLES -A INPUT -i lo -p all -j ACCEPT
## Cho phép các gói tin đi vào firewall chỉ với icmp protocol
$IPTABLES -A INPUT -p icmp -j ACCEPT
## Cho phép các packet đi vào eth1 có địa chỉ nguồn là địa chỉ của LAN
$IPTABLES -A INPUT -i $INTERNAL_LAN_INTERFACE -s $INTERNAL_LAN -j ACCEPT
# Cho phép các packet ra từ eth1 có địa chỉ đích là địa chỉ của LAN
$IPTABLES -A OUTPUT -o $INTERNAL_LAN_INTERFACE \
-d $INTERNAL_LAN -j ACCEPT
# Thực hiện NAT bằng cách đổi địa chỉ nguồn của gói tin trước khi định tuyến,
#####đi ra từ eth0 với bất kỳ địa chỉ nào khác địa chỉ của LAN
$IPTABLES -A -t nat -A POSTROUTING -o $EXTERNAL_LAN_INTERFACE \
-d ! $INTERNAL_LAN -j MASQUERADE
## Cho phép các gói tin đi qua firewall có địa chỉ nguồn hoặc địa chỉ đích
#####là địa chỉ của LAN
$IPTABLES -A FORWARD -s $INTERNAL_LAN -j ACCEPT
$IPTABLES -A FORWARD -d $INTERNAL_LAN -j ACCEPT
```



Hình 4.1: Mô hình mạng mô tả cho script internal-firewall.sh

4.11. Một số ví dụ về Firewall

Ví dụ 7: Cho phép truy xuất DNS đến Firewall

```
# iptables -A OUTPUT -p udp -o eth0 --dport 53 --sport 1024:65535 -j ACCEPT  
# iptables -A INPUT -p udp -i eth0 --dport 53 --sport 1024:65535 -j ACCEPT
```

Ví dụ 8: Cho phép www và ssh truy xuất tới Firewall

```
# iptables -A OUTPUT -o eth0 -m state --state ESTABLISHED, RELATED -j ACCEPT  
# iptables -A INPUT -p tcp -i eth0 --dport 22 --sport 1024:65535 -m state \  
--state NEW -j ACCEPT  
# iptables -A INPUT -p tcp -i eth0 --dport 80 --sport 1024:65535 -m state \  
--state NEW -j ACCEPT
```

Ví dụ 9: Masquerading (many to One NAT) là kỹ thuật NAT Many to One để cho phép nhiều máy cục bộ có thể sử dụng địa chỉ IP chính thức (được cung cấp từ ISP) để truy cập internet.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

```
##### Cho phép script tự khởi động với shell
#!/bin/sh
##### Nạp module iptable_nat
modprobe iptable_nat
##### Bật chức năng định tuyến
echo 1 > /proc/sys/net/ipv4/ip_forward
##### Cho phép sử dụng NAT giả mạo trong đó
##### - Interface eth0 là interface liên kết mạng internet
##### - Interface eth1 liên kết đến mạng nội bộ
iptables -A POSTROUTING -t nat -o eth0 -s 192.168.1.0/24 -d 0/0 -j
MASQUERADE
# Cho phép đi qua firewall trong trường hợp các trường hợp các kết nối là mới,
###đã thiết lập hoặc có liên hệ
iptables -A FORWARD -t filter -o eth0 -m state \
--state NEW, ESTABLISHED, RELATED -j ACCEPT
iptables -A FORWARD -t filter -i eth0 -m state \
--state NEW, ESTABLISHED, RELATED -j ACCEPT
```

Ví dụ 10: Thực hiện Port Forwarding với DHCP DSL. Trong trường hợp ta nhận 1 địa chỉ IP động từ ISP và ta muốn sử dụng đại chỉ này để cung cấp cho tất cả địa chỉ trong mạng nội bộ và public các server nội bộ ra bên ngoài internet. Tất cả các yêu cầu trên có thể giải quyết bằng cách sử dụng kỹ thuật Port Forwarding.

```
##### Cho script chạy với shell
#!/bin/sh
##### Nạp module iptable_nat
modprobe iptable_nat
##### Gán eth0 lên biến external_int
external_int = "eth0"
##### Thực hiện lấy ip mà DHCP cấp cho máy này
external_ip = `ifconfig $external_int | grep 'inet addr' | awk '{print $2}' | \
sed -e 's/.*://'`
##### Cho phép các interface forward với nhau
echo 1 > /proc/sys/net/ipv4/ip_forward
##### Thực hiện đổi địa chỉ đích trước khi thực hiện routing
iptables -t nat - PREROUTING - tcp -i eth0 - $external_ip --dport 80 \
--sport 1024:65535 - DNAT -to 192.168.1.2:8080
# Cho phép các packet FORWARD qua firewall trong các trường hợp dưới đây
iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.2 -dport 8080 \
-sport 1024:65535 -m state --state NEW -j ACCEPT
iptables -A FORWARD -t filter -o eth0 -m state \
--state NEW, ESTABLISHED, RELATED -j ACCEPT
iptables -A FORWARD -t filter -i eth0 -m state \
--state NEW, ESTABLISHED, RELATED -j ACCEPT
```

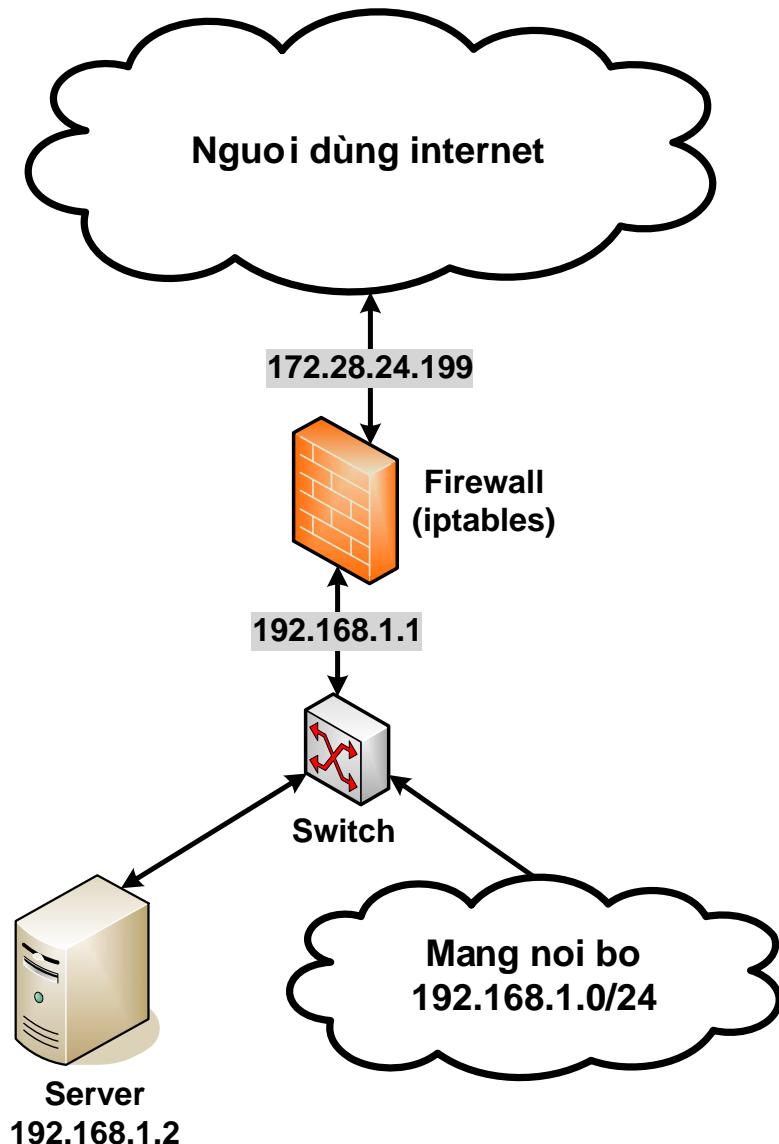
Ví dụ 11: Thực hiện NAT với ip tĩnh.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

- Sử dụng one to one NAT để cho phép server có địa chỉ 192.168.1.2 trên mạng nội bộ truy xuất ra ngoài internet thông qua địa chỉ 172.28.24.199.
- Tạo many to one NAT để cho mạng 192.168.1.0 có thể truy xuất đến tất cả các server trên internet thông qua địa chỉ 172.28.24.199.

```
##### Cho script chạy với shell
#!/bin/sh
## Load module và cho phép forward giữa các card mạng
modprobe iptable_nat
echo 1 > /proc/sys/net/ipv4/ip_forward
# Thực hiện DNAT để đổi địa chỉ đích thành địa chỉ của server
##### mạng nội bộ (192.168.1.2) khi truy cập đến 172.28.24.199
iptables -t nat -A PREROUTING -d 172.28.24.199 -i eth0 \
-j DNAT to-destination 192.168.1.2
## Thực hiện SNAT để đổi địa chỉ nguồn từ 192.168.1.2
##### ##### ##### ##### → 172.28.24.199
iptables -t nat -A POSTROUTING -s 192.168.1.2 -o eth0 \
-j SNAT --to-source 172.28.24.199
## Tương tự như trên, cho phép máy từ LAN truy cập đến các server
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 \
-j SNAT --to-source 172.28.24.199
## Cho phép bên ngoài truy xuất vào server (192.168.1.2)
##### thông qua các port 80, 443, 22
iptables -A FORWARD -p tcp -i eth0 -o eth1 -d 192.168.1.2 \
-m multiport --dport 80,443,22 -m state --state NEW -j ACCEPT
# Cho phép chuyển tất cả các NEW, ESTABLISHED SNAT connections
##### bắt đầu từ homework và thực sự đã thiết lập trước đó với DNAT
connections
iptables -A FORWARD -t filter -o eth0 -m state \
--state NEW, ESTABLISHED, RELATED -j ACCEPT
# Cho phép chuyển tất cả các connections bắt đầu từ internet đã được thiết lập
##### thông qua từ khóa NEW
iptables -A FORWARD -t filter -i eth0 -m state \
--state ESTABLISHED, RELATED -j ACCEPT
```



Hình 4.2: Mô hình mạng LAN với server

Ví dụ 12: Tạo một proxy

```
##### Cho phép script chạy với sh
#!/bin/sh
INTIF="eth1"      ## Gán chuỗi "eth1" vào INTIF
EXTIF="eth0"      ## Gán chuỗi "eth0" vào EXTIF
##### Thực hiện lấy địa chỉ ip mà DHCP cấp
EXTIP=`/sbin/ifconfig eth0 | grep 'inet addr' | awk '{print $2}' | sed -e
's/.*/\`'
##### Load module cần thiết
/sbin/depmod -a
/sbin/modprobe ip_tables
/sbin/modprobe ip_conntrack
/sbin/modprobe ip_conntrack_ftp
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

```
/sbin/modprobe ip_conntrack_irc  
/sbin/modprobe iptable_nat  
/sbin/modprobe ip_nat_ftp  
## Cho phép các card mạng có thể forward được với nhau  
echo "1" > /proc/sys/net/ipv4/ip_forward  
##### Cho phép thực hiện với ip động  
echo "1" > /proc/sys/net/ipv4/ip_dynaddr  
iptables -P INPUT ACCEPT      ## Mặc định INPUT chain là ACCEPT  
iptables -F INPUT              ## Xóa các luật trong INPUT chain  
iptables -P OUTPUT ACCEPT     ## Mặc định OUTPUT chain là ACCEPT  
iptables -F OUTPUT             ## Xóa các luật trong OUTPUT chain  
iptables -P FORWARD DROP      ## Mặc định FORWARD chain là DROP  
iptables -F FORWARD            ## Xóa các luật trong FORWARD chain  
iptables -t nat -F             ## Xóa tất cả các luật của bảng nat  
## Cho phép FORWARD đi vào eth0 đi ra eth1 trong trường hợp  
##### các connection là ESTABLISHED, RELATED  
iptables -A FORWARD -i $EXTIF -o $INTIF -m state \  
--state ESTABLISHED,RELATED -j ACCEPT  
##### Và ngược lại  
iptables -A FORWARD -i $INTIF -o $EXTIF -j ACCEPT  
## Thực hiện đổi địa chỉ nguồn trong trường hợp đi ra từ eth0  
iptables -t nat -A POSTROUTING -o $EXTIF -j MASQUERADE
```

Kết quả của việc cấu hình proxy trên, như sau:

```
# Generated by iptables-save v1.2.8 on Thu Nov 9 10:02:42 2006  
*nat  
:PREROUTING ACCEPT [536:76253]  
:POSTROUTING ACCEPT [2:119]  
:OUTPUT ACCEPT [15:909]  
-A POSTROUTING -o eth0 -j MASQUERADE  
COMMIT  
# Completed on Thu Nov 9 10:02:42 2006  
# Generated by iptables-save v1.2.8 on Thu Nov 9 10:02:42 2006  
*filter  
:INPUT ACCEPT [132:12857]  
:FORWARD DROP [0:0]  
:OUTPUT ACCEPT [0:0]  
:RH-Firewall-1-INPUT - [0:0]  
-A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT  
-A FORWARD -i eth1 -o eth0 -j ACCEPT  
-A RH-Firewall-1-INPUT -i lo -j ACCEPT  
-A RH-Firewall-1-INPUT -i eth0 -j ACCEPT  
-A RH-Firewall-1-INPUT -p icmp -m icmp any -j ACCEPT  
-A RH-Firewall-1-INPUT -p esp -j ACCEPT  
-A RH-Firewall-1-INPUT -p ah -j ACCEPT  
-A RH-Firewall-1-INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT  
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 4: CÀI ĐẶT VÀ CẤU HÌNH IPTABLES

```
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 21 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 23 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m state --state NEW -m tcp --dport 25 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Thu Nov 9 10:02:42 2006
```

4.12. Khắc phục sự cố trên iptables

- Với phần trình bày về iptables ở trên là khá đầy đủ. Với kiến thức trên, chúng ta có thể thực hiện những yêu cầu về lọc gói tin một cách khá tốt. Nhưng phần trên chỉ trình bày cách thực hiện với iptables mà không nêu ra cách khắc phục sự cố trên iptables. Trong phần này, chúng tôi sẽ trình bày cách khắc phục sự cố về iptables nói riêng, những phần mềm trên hệ điều hành mã nguồn mở nói chung.
- Cách vận hành và bảo trì những phần mềm trên linux thường sẽ qua những bước sau đây: cài đặt, cấu hình, vận hành và khắc phục sự cố khi có lỗi. Trong những phần trên, chúng tôi đã trình bày cách cài đặt, cấu hình và vận hành. Còn phần khắc phục sự cố về những phần mềm trên linux, thường thì người quản trị sẽ đọc file Log, cụ thể với iptables thì chúng ta cần kiểm tra Firewall Logs.
- Firewall logs được ghi nhận vào file `/var/log/message`. Để cho phép iptables ghi vào `/var/log/message`, chúng ta phải cấu hình như sau:

```
iptables -A OUTPUT -j LOG
iptables -A INPUT -j LOG
iptables -A FORWARD -j LOG
iptables -A OUTPUT -j DROP
iptables -A INPUT -j DROP
iptables -A FORWARD -j DROP
```

4.13. iptables không khởi động

- Khi ta khởi động iptables thì ta dùng lệnh `/etc/init.d/iptables start`. Lúc này, iptables gọi script trong file `/etc/sysconfig/iptables`. Do đó, nếu file này không tồn tại hoặc bị lỗi thì iptables sẽ không thực hiện được.
- Khi ta thay đổi cấu hình trên iptables thì ta phải dùng lệnh `service iptables save` để lưu lại các thông tin cấu hình. Sau đó, mới tiến hành restart lại iptables.

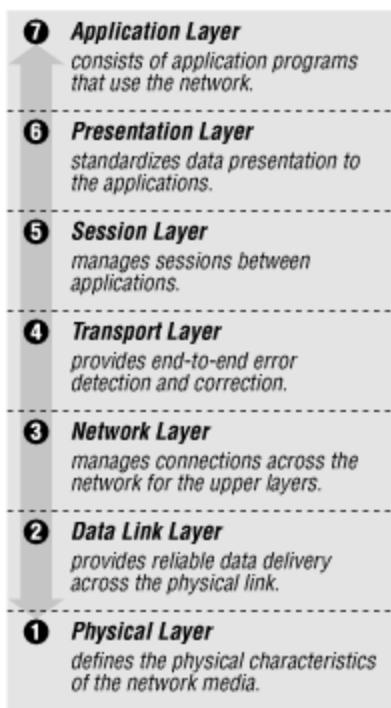
Ví dụ 13:

```
# service iptables start      ## Khởi động iptables
# touch /etc/sysconfig/iptables ## Tạo file iptables trống
## Thiết lập quyền cho file này
# chmod 600 /etc/sysconfig/iptables
# service iptables start
Applying iptables firewall rules: [OK]
```

Chương 5 : CÁC MÔ HÌNH MẠNG

5.1. Mô hình OSI :

Mô hình OSI cho các giao thức nhằm trao đổi dữ liệu (Data Communication Protocols). Mô hình được thành lập với mục đích làm chuẩn tham chiếu cho các giao thức truyền dữ liệu khác nhau trên thực tế.



Hình 5.1: Mô hình OSI

5.1.1. Lớp vật lý (Physical) :

Đây là lớp dưới cùng của mô hình OSI. Lớp vật lý xác định các đặc trưng của phần cứng cần thiết cho việc truyền các tín hiệu. Những đặc tả như hiệu điện thế để truyền tín hiệu, các loại cables, tốc độ truyền ... được xác định ở lớp này. Phương thức truyền thông đồng bộ và bất đồng bộ là hai phương thức cơ bản dùng cho giao tiếp mạng.

*F*o*u**r*o*u**g* *t*hu*ý* *t*ruy*ê*n *b*át *d*óng *b*ộ : Không có quy định nào về sự đồng bộ giữa máy gửi và máy nhận. Các bits đặc biệt START và STOP được dùng để tách dòng các bits được chuyển đi.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 5: CÁC MÔ HÌNH MẠNG

Phương thức truyền đồng bộ : Phương thức này đòi hỏi sự đồng bộ giữa máy gửi và máy nhận. Hệ thống “nhịp” được sử dụng cho phương thức này.

5.1.2. Lớp liên kết dữ liệu (Data Link) :

Lớp liên kết dữ liệu làm nhiệm vụ phân phát dữ liệu bảo đảm thông qua lớp vật lý. Lớp liên kết quy định dạng thức, kích thước địa chỉ của máy gửi và máy nhận. Lớp liên kết cũng chịu trách nhiệm sao cho các gói thông tin được chuyển đến đúng địa chỉ nhận với nội dung như khi gửi đi.

5.1.3. Lớp mạng (Network) :

Nhiệm vụ của lớp này là kết nối các mạng khác nhau đảm bảo việc truyền các gói thông tin từ mạng này đến một mạng khác. Hai chức năng chủ yếu của lớp này là chọn đường (routing) và chuyển tiếp (relaying). Đây là hai chức năng mà mọi nút trung gian trên đường truyền phải có để đảm bảo cho bó tin đến đích. Lớp này phải có khả năng đáp ứng với nhiều kiểu mạng khác nhau vì các nút có thể được nối với hai mạng hoàn toàn khác nhau như Ethernet và Token ring.

5.1.4. Lớp giao vận (Transport) :

Có trách nhiệm phân phát nguồn - đích (đầu cuối - đầu cuối) của toàn bộ thông điệp. Mỗi khi lớp mạng giám sát phân phát đầu cuối - đầu cuối các gói riêng lẻ, nó không nhận ra bất kỳ quan hệ giữa các gói đó. Nó ứng xử một cách độc lập như thể những mảnh nhỏ phụ thuộc vào thông điệp riêng biệt. Nhưng ở lớp giao vận, nó đảm bảo cả kiểm soát luồng và kiểm soát lỗi ở mức độ nguồn đích.

5.1.5. Lớp phiên (Session) :

Lớp phiên là một bộ điều khiển đàm thoại mạng. Nó thiết lập, duy trì và đồng bộ những tương tác giữa các hệ thống thông tin.

5.1.6. Lớp trình bày (Presentation) :

Quan tâm đến cú pháp và ngữ nghĩa của thông tin trao đổi giữa hai hệ thống. Có trách nhiệm: diễn dịch, mật hóa và nén.

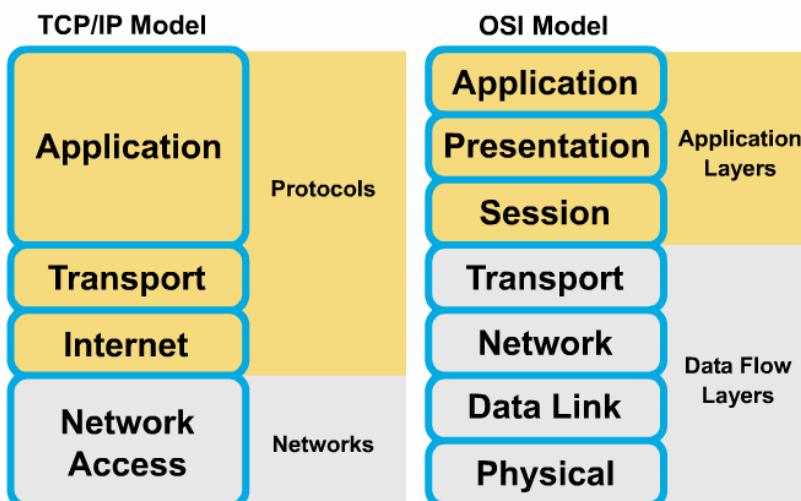
5.1.7. Lớp ứng dụng (Applicaton) :

Cho phép người sử dụng có thể là con người hay phần mềm truy cập mạng.

5.2. Mô hình TCP/IP

5.2.1. Lịch sử phát triển mô hình TCP/IP

- TCP/IP được phát triển trong một dự án được nghiên cứu bởi DARPA. Ban đầu giao thức này chỉ được dùng bên trong các hệ thống của DARPA. Sau đó TCP/IP được tích hợp vào các phần mềm chạy trên nền UNIX. Ngày nay TCP/IP là một chuẩn truyền thông mạng, cho phép hàng triệu máy tính truyền thông với nhau trên toàn cầu.
- Mô hình TCP/IP gần giống mô hình OSI 7 lớp ngoại trừ lớp Network của OSI tương đương với lớp Internet của TCP/IP. Hai lớp cuối là lớp physical và lớp data-link được ghép chung để trở thành lớp network interface của TCP/IP. Lớp 5, 6, 7 của OSI được gọi chung là lớp application của TCP/IP.
- Lớp ứng dụng của TCP/IP hỗ trợ rất nhiều giao thức truyền thông như: DNS, WINS, HOSTS, POP3, SMTP, FTP, TFTP, HTTP, ...



Hình 5.2: Mô hình TCP/IP và OSI

5.2.2. Lớp transport trong mô hình TCP/IP: cung cấp 2 protocol:

- TCP (Transmission Control Protocol) là một protocol thiên hướng kết nối, tin cậy cung cấp cơ chế điều khiển luồng bằng cách đưa ra các cửa sổ trượt, số thứ tự hay ACK để giám sát quá trình truyền dữ liệu. TCP sẽ gửi lại bất kỳ packet nào mà chưa có ACK trả về và cung cấp một mạch ảo giữa các ứng dụng đầu cuối. Ưu điểm của TCP là truyền dữ liệu rất tin cậy.

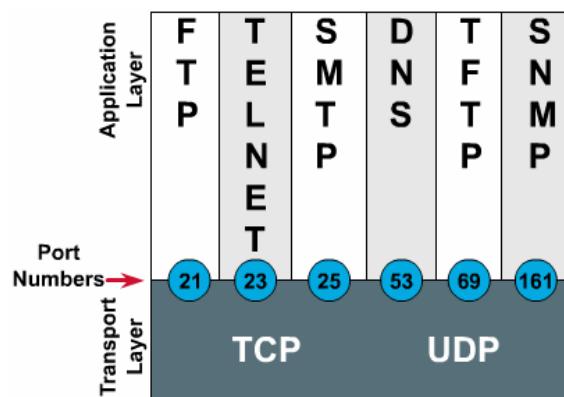
GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 5: CÁC MÔ HÌNH MẠNG

- UDP (User Datagram Protocol) là một protocol không kết nối và không tin cậy được xây dựng để truyền các message và không có cơ chế kiểm tra việc truyền các gói tin. Ưu điểm của UDP là tốc độ, bởi vì UDP không cung cấp cơ chế yêu cầu ACK, không điều khiển lưu lượng đi qua mạng nên nó truyền dữ liệu nhanh hơn.

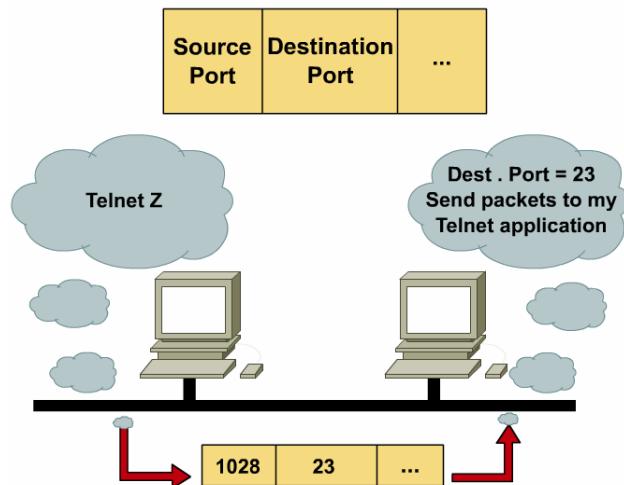
5.2.3. Số port của TCP và UDP

- Cả TCP và UDP đều sử dụng số port để truyền dữ liệu lên lớp trên. Số port được sử dụng để theo dõi các cuộc đối thoại khác nhau đi qua Network cùng một thời điểm



Hình 5.3: Các port thông dụng của TCP và UDP

- Các nhà phát triển phần mềm ứng dụng đã đồng ý sử dụng một số port nổi tiếng (được định nghĩa trong chuẩn RFC 1700). Ví dụ Telnet sử dụng port 23, FTP dùng port 21, DNS dùng port 53...
- Các ứng dụng khác sẽ được gán một port khác với các port nổi tiếng trên để thiết lập đường truyền dữ liệu giữa các thiết bị đầu cuối. Các số port này phải nằm trong khoảng sau:
 - Nhỏ hơn 255 dùng cho các ứng dụng phổ biến
 - Từ 255 đến 1023 được dùng cho các ứng dụng thương mại
 - Lớn hơn 1023 chưa được dùng

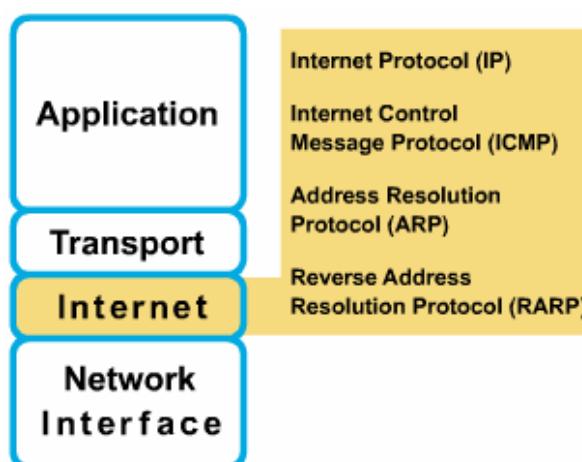


Hình 1.4: Sử dụng số port của TCP/UDP

5.2.4. Lớp Internet của TCP/IP :

Lớp Internet của TCP/IP tương ứng với lớp Network trong mô hình OSI, rất nhiều protocol hoạt động trên lớp Internet:

- IP (Internet Protocol): cung cấp việc truyền data không kết nối, hiệu suất cao
- ICMP (Internet Control Message Protocol): dùng để truyền error và các message điều khiển
- ARP (Address Resolution protocol): xác định MAC khi biết IP
- RARP (Reverse Address Resolution Protocol): xác định địa chỉ network khi biết MAC



Hình 5.5: Các protocol hoạt động trên lớp Internet

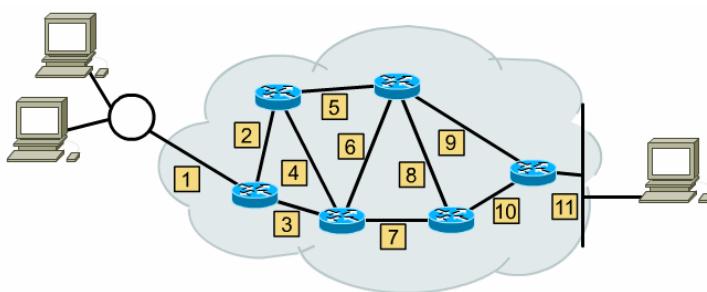
Chương 6: ĐỊNH TUYẾN

Xác định đường đi cho một packet đi qua một network được thực hiện ở lớp network. Chức năng xác định đường đi cho phép một router ước lượng các đường đi có thể đến đích được và xác định một cơ chế quản lý thích hợp. Các dịch vụ định tuyến sẽ dùng các thông tin về topo mạng. Các thông tin này có thể được cấu hình bởi người quản trị mạng hay được thu nhập thông qua các tiến trình chạy tự động trên network.

Lớp network cung cấp cơ chế truyền packet qua network một cách hiệu quả nhất. Lớp network sẽ sử dụng bảng định tuyến IP để gửi packet từ network nguồn đến network đích. Sau khi xác định được đường đi, router sẽ forward packet đó. Điều đó có nghĩa là nó nhận packet ở một interface và forward packet qua một interface khác mà đường đi đến đích là tốt nhất.

6.1. Làm thế nào để router có thể định tuyến cho packet từ nguồn đến đích

Về mặt thực tế, một network là các đường liên kết giữa các router. Mỗi line giữa các router có một số xác định được gọi là địa chỉ mạng.

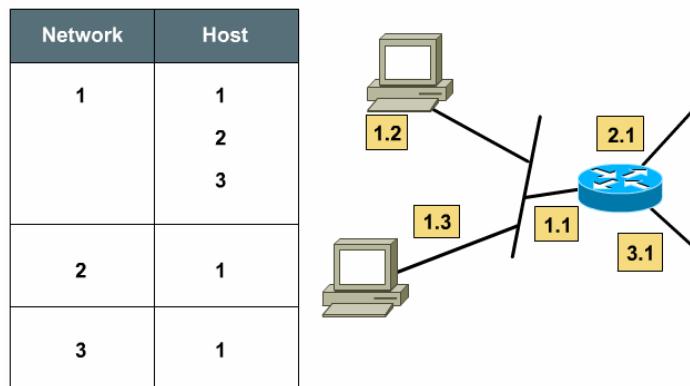


Hình 6.1: Định địa chỉ đặc trưng cho các đường đi
kết nối môi trường truyền

- Các địa chỉ này phải chứa các thông tin mà có thể được dùng để định tuyến cho các packet. Sử dụng các địa chỉ này, lớp network có thể cung cấp một kết nối chuyển tiếp mà liên kết với các network độc lập khác. Có thể cải tiến hiệu quả sử dụng băng thông bằng cách chặn các thông tin được broadcast không cần thiết. Bằng cách sử dụng cách đánh địa chỉ end-to-end đặc trưng cho các đường kết nối vật lý, lớp network có thể tìm ra đường đi đến đích mà không làm tăng băng thông của các đường liên kết network.

6.2. Định địa chỉ network và host

- Router sử dụng địa chỉ network để xác định network đích (LAN) của một packet bên trong một internetwork. Hình vẽ cho thấy 3 network xác định 3 segment liên kết với router.

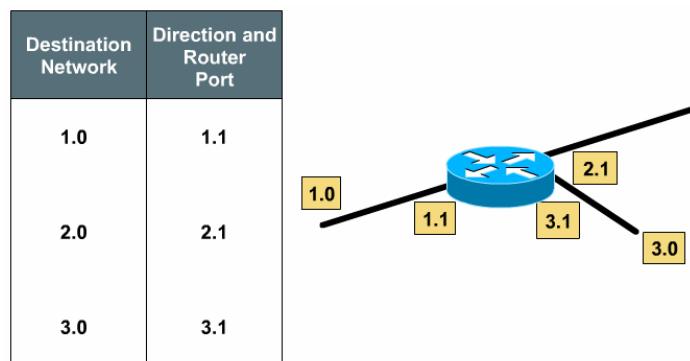


Hình 6.2: Một địa chỉ network bao gồm 2 phần: phần xác định network và phần xác định host

- Với một số protocol lớp network, có thể xác định địa chỉ host dựa trên địa chỉ segment mà host đó thuộc về.

6.3. Chọn đường đi và chuyển mạch packet

- Hai nhiệm vụ chính của router là chọn đường đi tốt nhất và chuyển mạch packet. Hình vẽ cho thấy router sử dụng địa chỉ để định tuyến và chuyển mạch packet như thế nào. Router sẽ sử dụng phần địa chỉ mạng của địa chỉ packet để định tuyến và truyền packet đến router kế tiếp dọc theo đường đi.

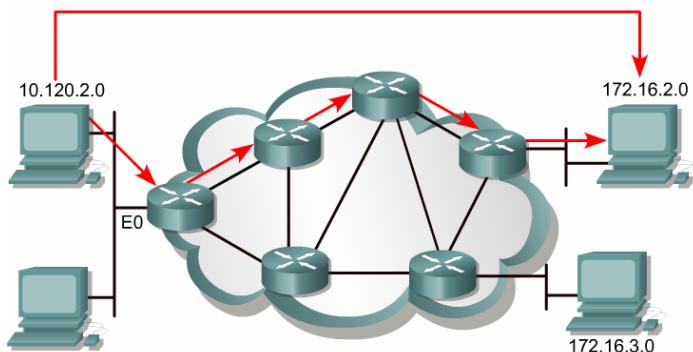


Hình 6.3: Phần địa chỉ network dùng để chọn đường đi cho packet

- Phần địa chỉ host của packet sẽ được sử dụng bởi router cuối cùng (router nối trực tiếp với network đích) truyền packet đó đến đúng host xác định.

6.4. Routed protocol và routing protocol

- *Routed protocol*: là bất kỳ một protocol network nào cung cấp đủ thông tin trong địa chỉ lớp network để có thể cho phép một packet được forward từ một host đến một host khác dựa trên một cách đánh địa chỉ. Routed protocol định nghĩa các field bên trong packet. Một routed protocol sẽ sử dụng bảng định tuyến để forward packet. IP hay IPX là các ví dụ điển hình của routed protocol. Nói chung, routed protocol là các protolcol hỗ trợ lớp network .



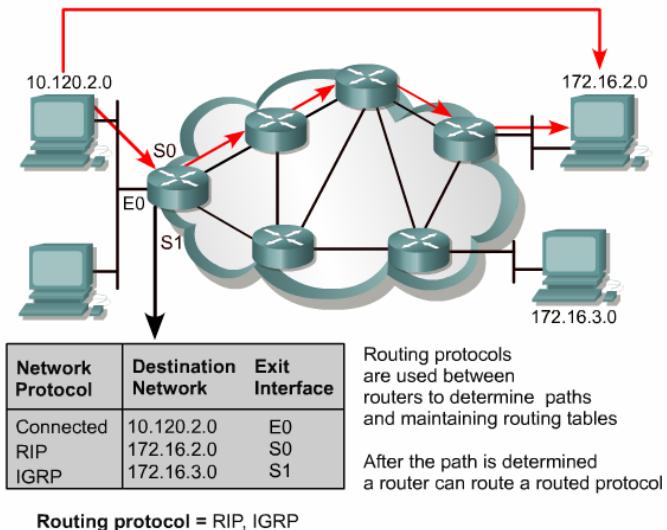
Routed protocol transport data from one end-station to another.

Hình 6.4: Routed protocol

- *Routing protocol*: hỗ trợ cho routed protocol bằng cách cung cấp các cơ chế để chia sẻ các thông tin định tuyến. Các message của routing protocol được truyền đi giữa các router. Một routing protocol cho phép các router giao tiếp với nhau để cập nhật và duy trì bảng định tuyến.

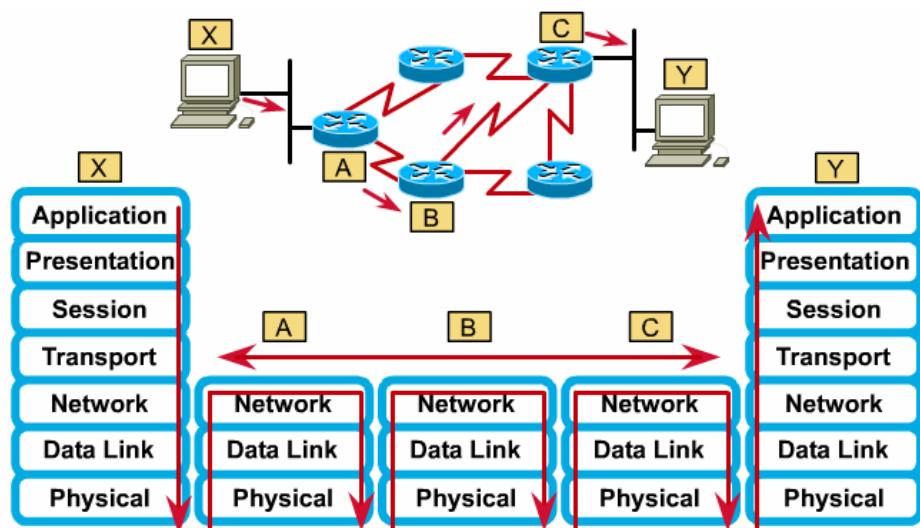
Các ví dụ về routing protocol:

- Routing Information Protocol (RIP)
 - Interior Gateway Routing Protocol (IGRP)
 - Enhanced Interior Gateway Routing Protocol (EIGRP)
 - Open Shortest Path First (OSPF).



Hình 6.5: Routing protocol

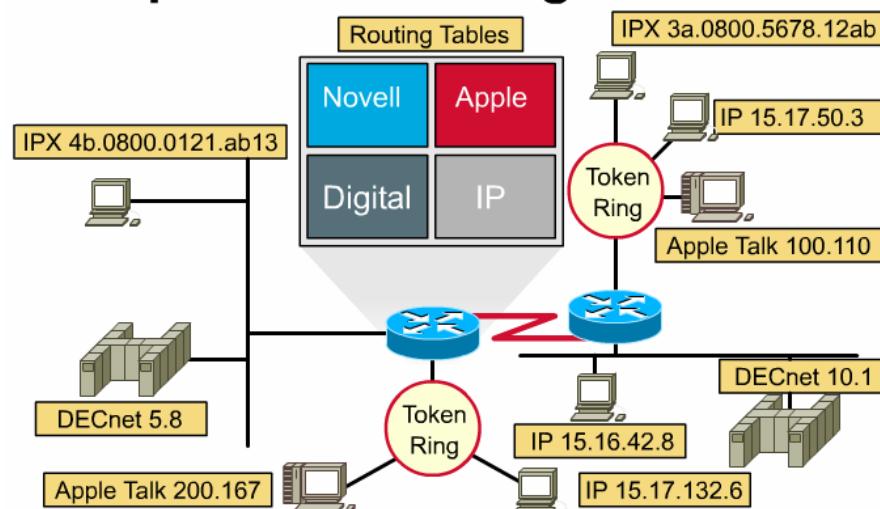
- Khi một ứng dụng host cần gửi một packet đến một host đích ở một network khác, host sẽ đánh địa chỉ của frame data-link đến router, dùng một trong các địa chỉ MAC của các interface của router. Các quá trình xử lý lớp network của router sẽ kiểm tra header của packet tới và xác định network đích, sau đó nó sẽ tham chiếu bảng định tuyến để biết là cần gửi packet đó ra cho interface nào của router. Packet sẽ được đóng gói lại theo định dạng của interface đó và được forward đi đến router kế tiếp.



Hình 6.6: Hoạt động của protocol network

- Tại router nối trực tiếp với network đích, packet sẽ được đóng gói theo định dạng tương ứng và được truyền đến host đích.
- Router có khả năng hỗ trợ nhiều routing protocol hoạt động độc lập và duy trì các bảng routing hỗ trợ cho nhiều routed protocol. Khả năng này cho phép router truyền các packet từ nhiều routed protocol khác nhau trên một đường kết nối data giống nhau.

Multiprotocol Routing



- ◆ Routers pass traffic from all routed protocols over the internetwork

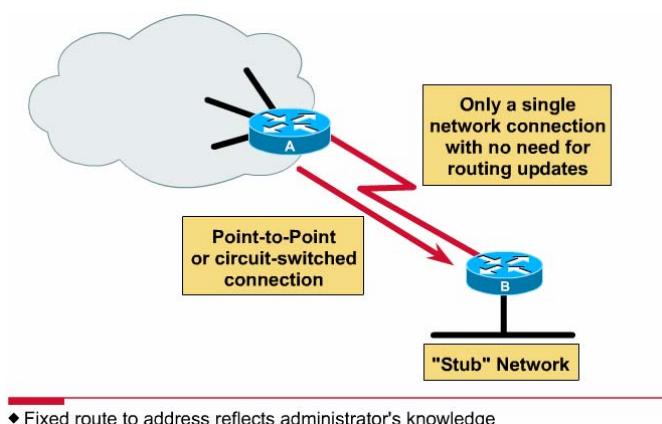
Hình 6.7: Khả năng hỗ trợ đa protocol của router

6.5. Định tuyến tĩnh và định tuyến động (static and dynamic routing)

- *Định tuyến tĩnh* được thiết lập bằng tay bởi người quản trị network (người cấu hình cho các router) khi có bất kỳ thay đổi nào về topo network trong internetwork.
- *Định tuyến động* thì hoạt động khác. Sau khi người quản trị network cấu hình cho router để nó định tuyến động thì các thông tin để định tuyến sẽ tự động được cập nhật bằng một quá trình định tuyến bất kỳ khi nào có một thông tin mới router nhận được từ internetwork. Việc thay đổi động về các thông tin định tuyến đó là một phần của quá trình cập nhật của router.
- Mục đích của định tuyến tĩnh: định tuyến tĩnh có rất nhiều ứng dụng

hữu ích. Định tuyến động có khuynh hướng bộc lộ mọi thứ về internetwork (nghĩa là có thể biết được topo network khi đọc bảng định tuyến). Điều này không tốt lắm trong việc bảo mật khi bạn muốn che dấu một phần của network mình quản lý. Trong khi định tuyến tĩnh cho phép bạn xác định thông tin mà bạn muốn thể hiện về network.

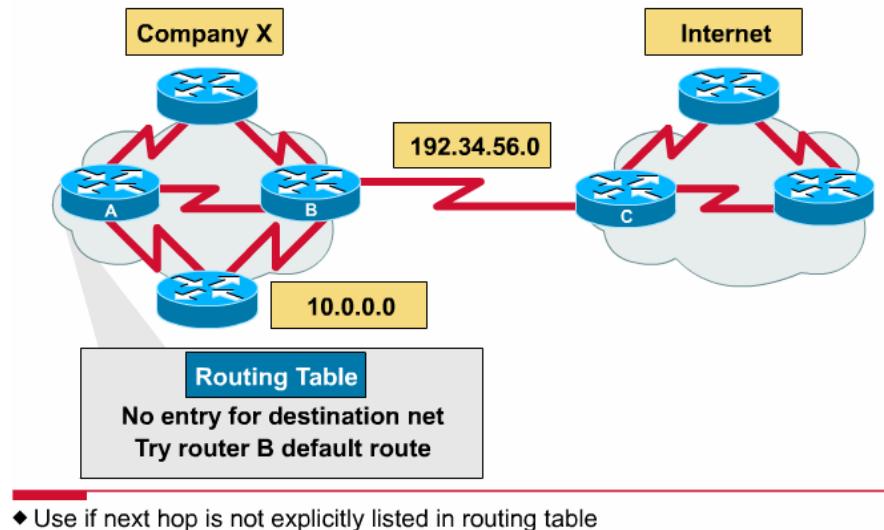
- Khi một network chỉ có một đường truy cập (ví dụ đi ra Internet chẵng hạn) thì việc sử dụng định tuyến tĩnh rất hữu ích. Cấu hình cho router hoạt động với định tuyến tĩnh sẽ làm tăng tốc độ truyền dữ liệu (vì không phải truyền thông tin về routing giữa các router nhiều).



Hình 6.8: Định tuyến tĩnh có khả năng giới hạn nhu cầu cho phép các router cập nhật thông qua các đường kết nối

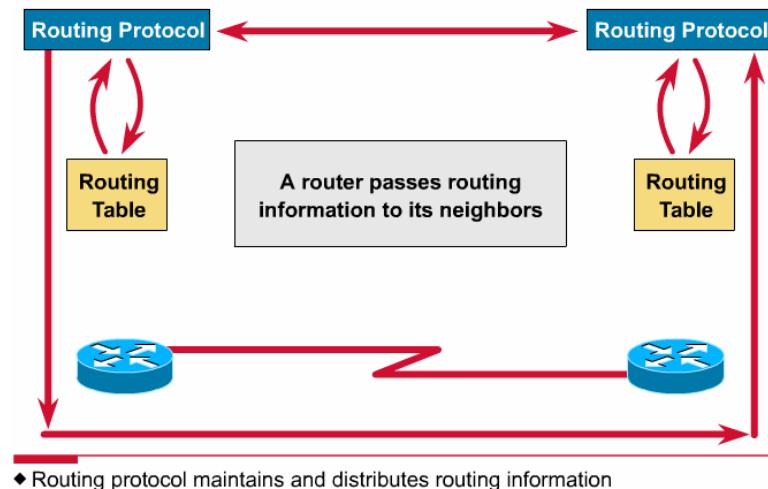
6.6. Sử dụng định tuyến mặc định (default route)

- *Định tuyến mặc định* được sử dụng khi một router không biết đường đi của packet. Lúc này nó sẽ forward packet ra interface được xác định bởi định tuyến mặc định.
- Trong ví dụ sau, các router của company X chỉ cần quản lý định tuyến trong phạm vi topo mình quản lý, không cần biết đến topo của các company khác. Việc duy trì thông tin của mọi network là không cần thiết và quá lãng phí băng thông. Thay vì vậy mỗi router của company X sẽ đọc cấu hình một định tuyến mặc định là đi ra Internet (IP = 192.34.56.0) khi nó không tìm được đường đi của packet trong bảng định tuyến.



Hình 6.9: Định tuyến mặc định được sử dụng khi không tìm thấy đường đi trong bảng định tuyến

6.7. Tại sao định tuyến động vẫn cần thiết



Hình 6.10: Định tuyến động duy trì và phân bố các thông tin định tuyến

- Định tuyến động có khả năng tự phát hiện lỗi trên đường truyền và tự động tìm đường đi khác. Đó là điều mà định tuyến tĩnh không làm được. Bất kỳ một thay đổi nào về topo network cũng được các router cập nhật.Thêm vào đó định tuyến động có khả năng forward các packet đi các đường khác nhau để cân bằng lưu lượng tải.
- Hoạt động của định tuyến động dựa trên 2 chức năng cơ bản của router:
 - Duy trì bảng routing

- Định thời gian quá trình cập nhật thông tin giữa các router.
- Định tuyến dựa trên một routing protocol để chia sẻ thông tin giữa các router. Một routing protocol sẽ định nghĩa một tập hợp các nguyên tắc được router sử dụng khi nó giao tiếp với các router kế bên. Ví dụ, một routing protocol sẽ mô tả các thông tin sau:
 - Khi nào thì gửi các thông tin cập nhật
 - Cần gửi thông tin gì trong các packet cập nhật này
 - Gửi các thông tin cập nhật như thế nào
 - Xác định các thông tin trong các packet cập nhật này ra sao.
- Khi một routing protocol cập nhật bảng routing, nó phải làm sao xác định được thông tin nào tốt nhất để ghi vào bảng routing. Mỗi một routing protocol có một cách xác định riêng đường đi tốt nhất. Nhưng cách thức chung là sẽ có một giải thuật phát sinh ra một số gọi là trọng số (metric) cho mỗi đường đi trong network. Trọng số càng bé thì đường đi càng tốt. Một số trọng số được router sử dụng để tính toán đường đi:
 - *Băng thông* (bandwidth): tốc độ dữ liệu của một đường liên kết
 - *Thời gian trễ* (delay): thời gian cần thiết để một packet được truyền từ nguồn đến đích.
 - *Tải lượng* (load): lượng thông tin truyền trên một đường liên kết dữ liệu.
 - *Độ tin cậy* (reliability): liên quan đến tỉ lệ lỗi bit của đường truyền.
 - *Số lượng hop* (hop-count): chính là số router mà packet phải đi qua.
 - *Chi phí* (cost): liên quan đến chất lượng, băng thông hay bất kỳ đại lượng nào khác, thông số này được xác lập bởi người quản trị network.
 - *Tick*: thời gian trễ khi dùng clock tick (xấp xỉ 55 ms).

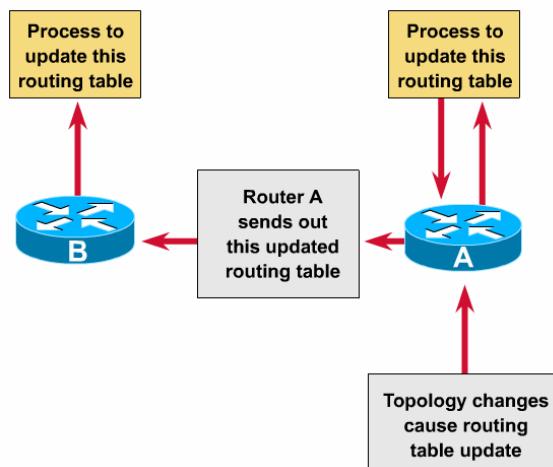
6.8. Các lớp routing protocol

- Hầu hết các giải thuật routing được chia làm 3 giải thuật cơ bản:
 - *Giải thuật distance-vector*: xác định được hướng đi và khoảng cách đến bất kỳ node nào trên network.
 - *Giải thuật Link-State*: tạo ra chính xác topo network của toàn bộ internetwork, từ đó tìm ra đường đi tốt nhất cho packet.
 - *Giải thuật hỗn hợp* (ghép 2 giải thuật trên).
- Thời gian hội tụ: giải thuật routing dựa trên cơ sở định tuyến động. Bất kỳ khi nào topo network thay đổi do sự phát triển, do cấu hình

hay sự cố thì cơ sở thông tin về network cũng thay đổi theo. Thông tin phải phản ánh được chính xác trạng thái mới của topo network. Khi tất cả các router trong một internetwork hoạt động với các thông tin về topo network giống nhau thì internetwork đó được gọi là hội tụ. Độ hội tụ nhanh là rất cần thiết bởi vì nó sẽ làm giảm thời gian quyết định định tuyến sai sau khi topo network thay đổi.

6.9. Cơ bản về distance-vector routing

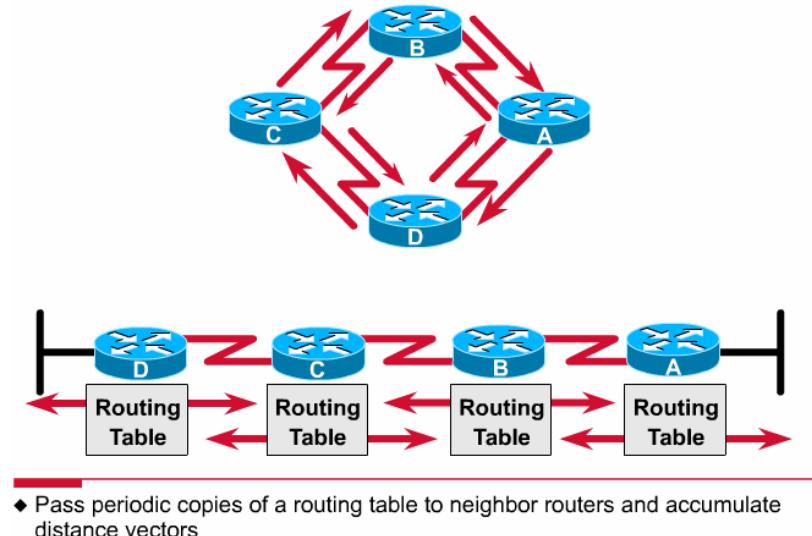
- Distance-vector routing dựa trên ý tưởng là sẽ gửi một bản copy của bảng định tuyến từ router này đến router khác sau một khoảng thời gian xác định. Mỗi router sẽ nhận được một bảng định tuyến từ router kế cận, nó sẽ tính toán lại các tham số trong bảng định tuyến hiện hành sao cho các đường đi là tốt nhất đối với các routing protocol đang sử dụng.



Hình 6.11: Khái niệm về định tuyến distance-vector

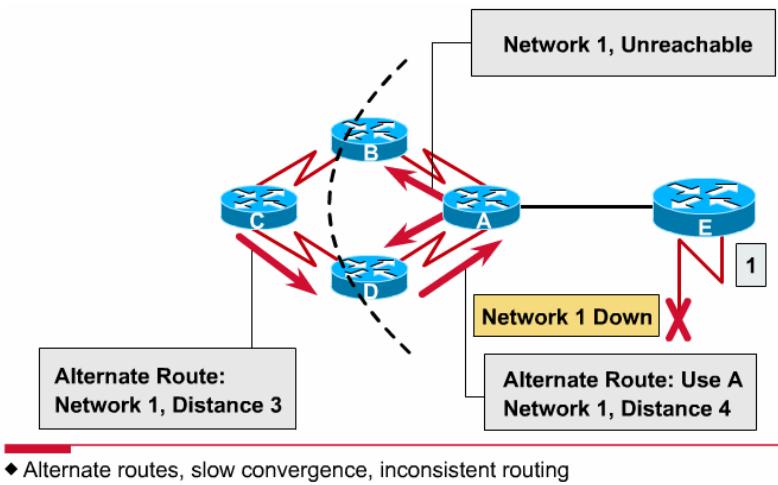
a)- Các vấn đề về routing loop

- Routing loop xảy ra với các network có độ hội tụ chậm so với các thay đổi về topo network, gây nên những quyết định sai về định tuyến. Hình vẽ cho thấy Routing loop xảy ra.
- Trước khi network 1 down, tất cả các router đều có bảng định tuyến đúng. network giả sử là hội tụ, và đường đi từ C đến network 1 là 3.



Hình 6.12: Hoạt động của distance-vector routing

- Khi network 1 down, E gửi update cho A. A ngừng xử lý định tuyến cho packet đến network 1 và B, C, D vẫn tiếp tục định tuyến bởi vì chúng chưa nhận được thông tin về việc network 1 bị down. Khi A gửi cập nhật thì B và D nhận được nên ngừng định tuyến đến network 1 nhưng C vẫn tiếp tục. Đối với C thì vẫn có thể đến network 1 được bằng cách đi qua B.



Hình 6.13: Router A cập nhật bảng định tuyến để phản ánh đúng những thay đổi mới của topo network nhưng những thông tin ấy lại sai

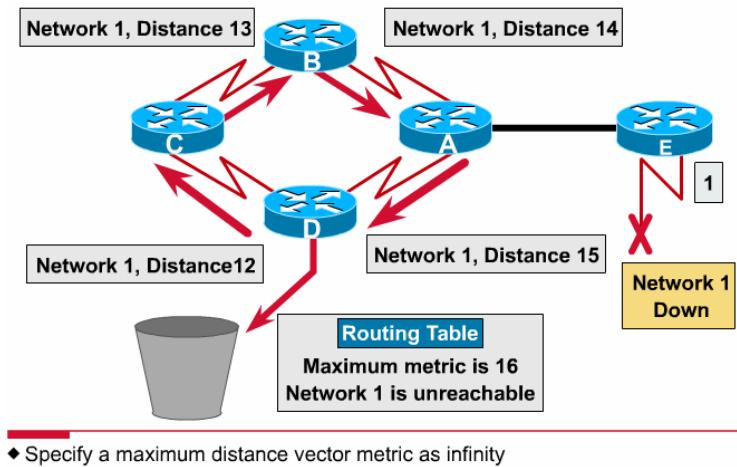
- Bây giờ C gửi update cho D xác nhận rằng có một đường đi đến network 1 bằng cách đi qua B. D thay đổi bảng định tuyến của nó để phản ánh đúng thông tin nó nhận được. Nhưng đây lại là thông tin sai, D lại chuyển thông tin sai này đến A. A lại chuyển đến B và E. Như vậy một packet muốn đến network 1 bị loop từ C đến B qua A lại đến D rồi quay lại C.

b)- Vấn đề về đếm vô định (count infinity)

- Trở lại ví dụ vừa rồi, ta thấy thông tin cập nhật sai về network một tiếp tục bị loop cho đến khi nào có một tiến trình làm ngừng vấn đề loop này. Điều kiện này được gọi là đếm vô định, các packet bị loop chạy vòng trong mạng trong khi sự thật là network 1 đã bị down. Trong khi các router đang đếm vô định thì các thông tin sai lại cho phép routing loop tồn tại.
- Nếu không có việc đếm để ngừng loop, trọng số về hop-count sẽ tăng mỗi khi packet đi qua một router.

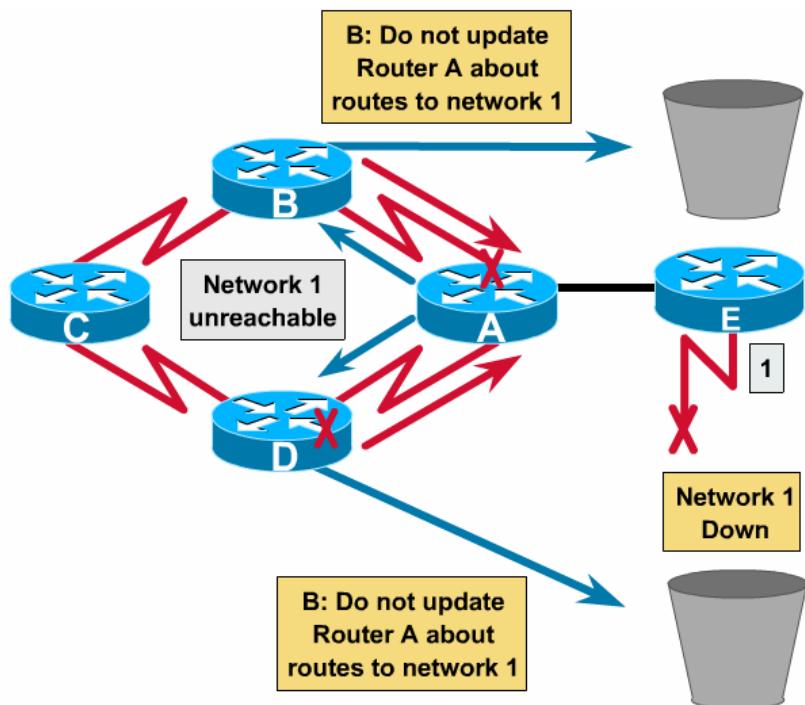
c)- Các phương pháp chống routing loop

- Xác định số hop-count tối đa: khi packet bị loop, qua một router trọng số hop-count sẽ được tăng lên một đơn vị. Khi hop-count của một packet bằng 16, packet đó sẽ bị hủy bỏ không truyền nữa.
- Split horizon*: cách này dựa trên ý tưởng là router sẽ không gửi cập nhật về phía interface nơi mà nó đã học được thông tin định tuyến mới (cập nhật bảng định tuyến). Split horizon làm giảm các thông tin định tuyến sai và làm giảm các overhead định tuyến.



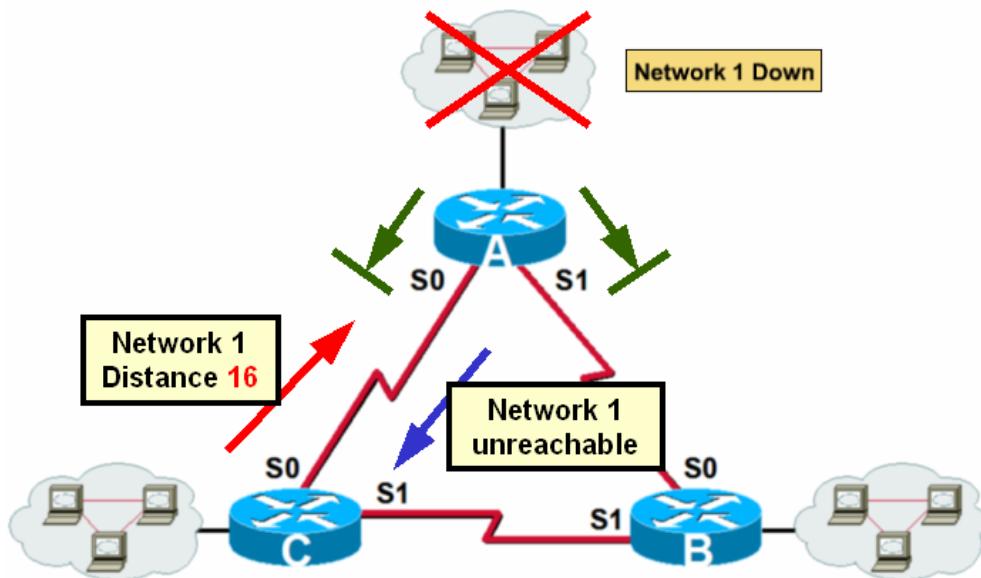
Hình 6.14: Xác định số hop-count tối đa tránh tình trạng loop trong network

- Poisson Reserve*: router sẽ gửi bảng routing đến mọi interface, nhưng những interface nào mà nó đã học được đường đi thì nó sẽ gửi thông tin với trọng số khoảng cách là lớn nhất, để khi router kế cận nhận được thông tin này sẽ hủy bỏ mà không cập nhật bảng routing.
- Route Poisioning*: khi một router xác định là một đường đi không tồn tại, nó sẽ gửi thông tin cập nhật đến mọi router với thông tin về đường đi này có trọng số khoảng cách là lớn nhất để các router khác không cập nhật đường đi này vào bảng định tuyến.

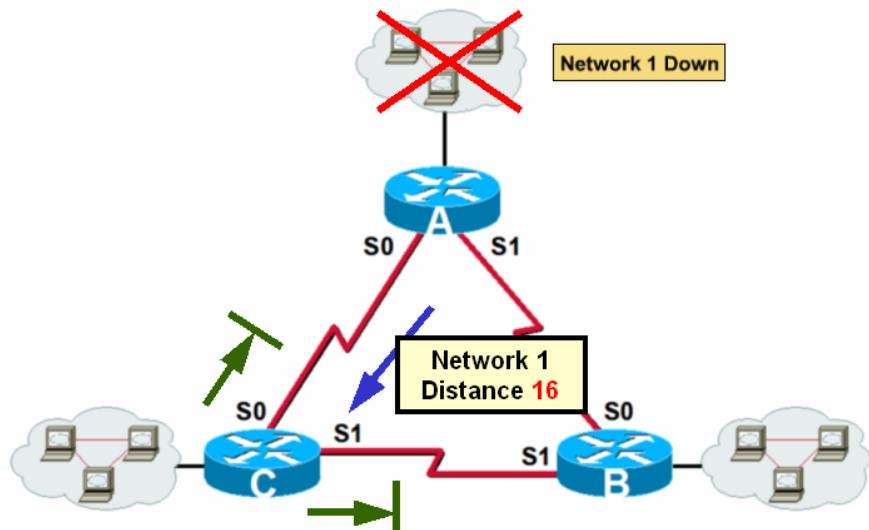


Hình 6.15: Sử dụng split horizon để tránh tình trạng loop trong network

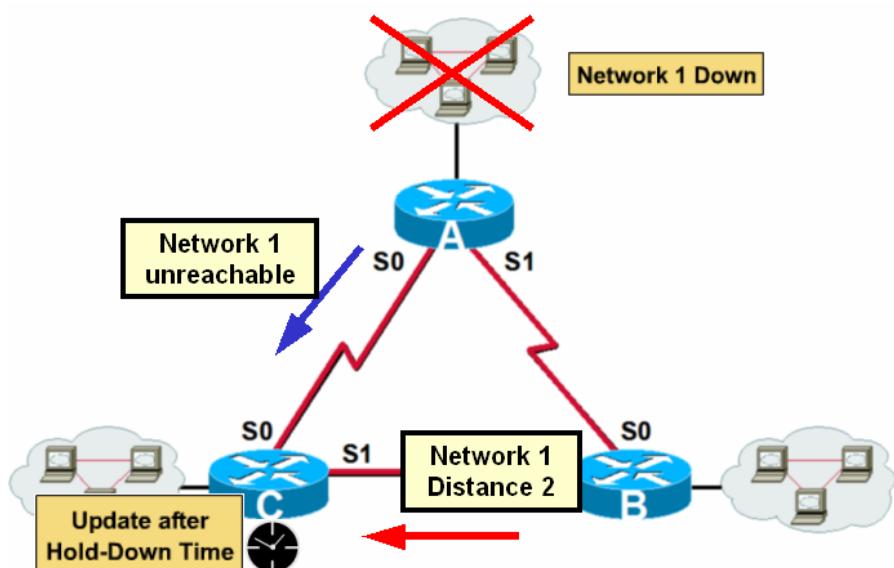
- *Sử dụng hold-down timer:* sau khi biết một network bị down, router sẽ chờ trong một khoảng thời gian nhất định để chắc chắn rằng không còn có một thông tin cập nhật nào nói rằng network ấy còn tồn tại. Khi đó nó sẽ xóa network bị down đó trong bảng routing.



Hình 6.16: Sử dụng poisson reserve để tránh tình trạng loop trong network



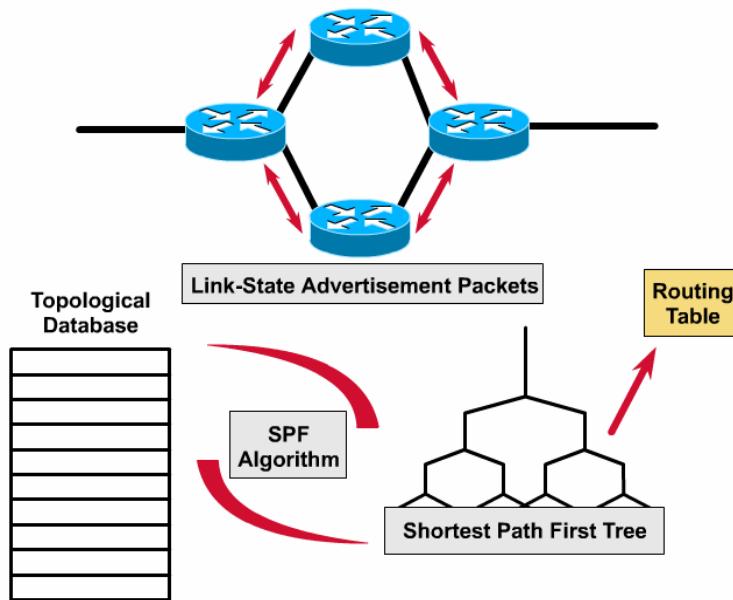
Hình 6.17: Sử dụng route poisioning để tránh tình trạng loop trong network



Hình 6.18: Sử dụng hold-down timer để tránh tình trạng loop trong network

6.10. Cơ bản về link-state routing

- Giải thuật định tuyến cơ bản thứ 2 là giải thuật link-state còn được gọi là giải thuật SPF (Shortest Path First). Giải thuật này duy trì một cơ sở dữ liệu phức tạp về thông tin của topo network. Ở những nơi mà giải thuật distance-vector không có đầy đủ các thông tin về các khoảng cách trong network và các khoảng cách giữa các router, một giải thuật định tuyến link-state sẽ duy trì đầy đủ thông tin về khoảng cách giữa các router và chúng kết nối với nhau như thế nào.

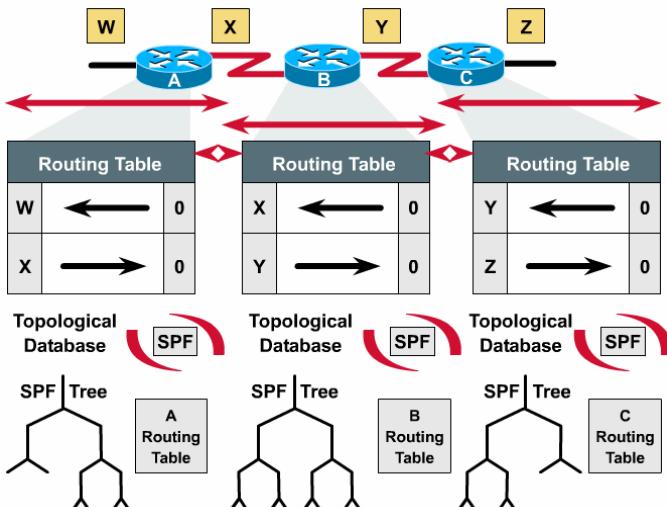


Hình 6.19: Khái niệm về định tuyến link-state

- Link-state routing sử dụng các thông tin sau:
 - Quảng bá link-state (LSA)
 - Cơ sở dữ liệu về topo network.
 - Giải thuật SPF và kết quả của cây SPF
 - Bảng định tuyến của các đường đi và port đến mỗi network.

a)- Trao đổi thông tin của Link-state protocol

- Cơ chế hoạt động của link-state network là tạo ra một bức tranh tổng thể chung về toàn bộ network. Mọi link-state router chia sẻ các thông tin về toàn bộ network này. Điều này tương tự như có nhiều bản đồ khác nhau của một thành phố. Trong hình vẽ, 4 network (W, X, Y & Z) được nối với nhau bởi 3 link-state router.
- Việc giao tiếp giữa các router sử dụng các quá trình sau:
 - Router sẽ trao đổi LSA với router khác. Mỗi router sẽ bắt đầu với network nối trực tiếp với nó, những network mà nó có thể lấy được thông tin đầu tiên và trực tiếp.
 - Mỗi router song song với các router khác sẽ xây dựng một cơ sở dữ liệu về topo network chứa tất cả LSA từ internetwork.
 - Giải thuật SPF sẽ tính toán khả năng tìm được đường đi trong internetwork. Router sẽ xây dựng topo logic này thành một cấu trúc cây mà nó là gốc, chứa mọi đường đi có thể đến một network trong internetwork. Sau đó nó sẽ sắp xếp các đường đi này theo thứ tự ưu tiên đường đi ngắn nhất trước tiên (SPF).



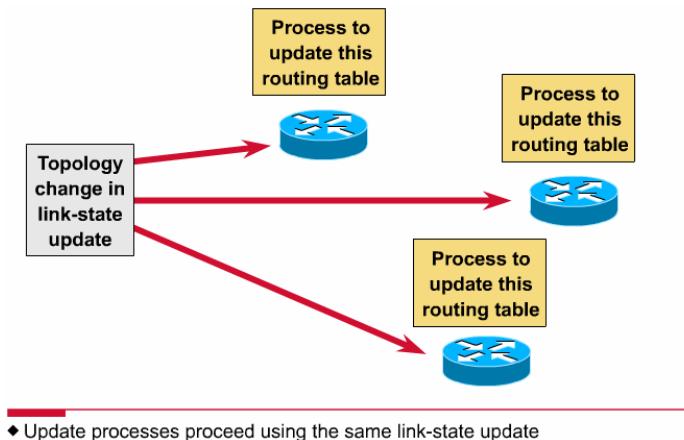
Hình 6.20: Trong định tuyến link-state các router tính toán đường đi dựa trên các cây SPF

- Router sẽ liệt kê những đường đi tốt nhất và các port đến những network đích trong bảng định tuyến. Nó cũng duy trì các cơ sở dữ liệu khác về chi tiết trạng thái và thành phần của topo network.

b)- Truyền các thông tin về topo trong network đến các router

- Giải thuật link-state dựa trên việc sử dụng các cập nhật link-state. Như hình vẽ, bất kỳ khi nào có sự thay đổi về topo link-state, các router sẽ bắt đầu gửi một packet LSA mới đến các router khác hay đến một router xác định để các router khác có thể sử dụng thông tin này cập nhật bảng định tuyến. Packet LSA này sẽ được broadcast đến mọi router trong internetwork. Để đạt được độ hội tụ, mỗi router tiến hành như sau:
 - Theo dõi các router nối liền với nó, bao gồm tên, khi nào thì các router bên cạnh nó (nối trực tiếp) up hay down và phí tổn (cost) của đường liên kết đến các router kế cận.
 - Xây dựng một packet LSA liệt kê tên các router kế cận nó và phí tổn đường liên kết, bao gồm các router kế cận mới, các thay đổi về phí tổn đường liên kết và các liên kết đến các router kế cận đã bị down.
 - Gửi LSA packet này sao cho mọi router có thể nhận được.
 - Khi router nhận được packet LSA, nó sẽ lưu lại packet này trong cơ sở dữ liệu của nó để nó có thể sử dụng được hầu hết các packet LSA mới nhất mà nó nhận được từ các router khác.
 - Tính toán lại toàn bộ internetwork dựa trên các LSA packet đã tích lũy được và tính toán các đường đi đến mọi network khác

bằng cách sử dụng giải thuật SPF.



Hình 6.21: Quá trình cập nhật sử dụng các thông tin giống nhau

- Mỗi khi một packet LSA tạo nên một thay đổi đến cơ sở dữ liệu , giải thuật SPF sẽ tính toán lại các đường đi tốt nhất và cập nhật bảng routing. Sau đó mọi router sẽ xác định đường đi ngắn nhất cho mỗi packet.

c)- Hai vấn đề cơ bản của định tuyến link-state

Định tuyến link-state yêu cầu về xử lý bộ nhớ và về băng thông đủ mạnh để có thể thực hiện được các năng lực tính toán phức tạp.

c.1. Yêu cầu về xử lý và bộ nhớ:

- Khi dùng link-state protocol, các router phải thực hiện xử lý nhiều hơn khi dùng distance-vector protocol. Người quản trị network phải đảm bảo rằng các router mà họ chọn phải có khả năng cung cấp những tài nguyên cần thiết.
- Router theo dõi mọi router khác trong nhóm và các network mà nó có thể đi đến trực tiếp. Với định tuyến link-state, bộ nhớ phải có khả năng lưu giữ thông tin từ các cơ sở dữ liệu khác, cây topo và bảng định tuyến khi sử dụng giải thuật Dijkstra để tính toán SPF yêu cầu một quá trình xử lý phức tạp các đường liên kết trong internetwork, nhân với số lượng các router trong internetwork.

c.2. Yêu cầu về băng thông

- Yêu cầu về băng thông lớn nhất khi broadcast các packet link-state khởi tạo. Trong suốt quá trình khởi tạo, mọi router sử dụng các protocol định tuyến link-state sẽ gửi các packet LSA đến mọi router khác. Điều này sẽ tăng tải lượng trên internetwork rất lớn. Tuy nhiên

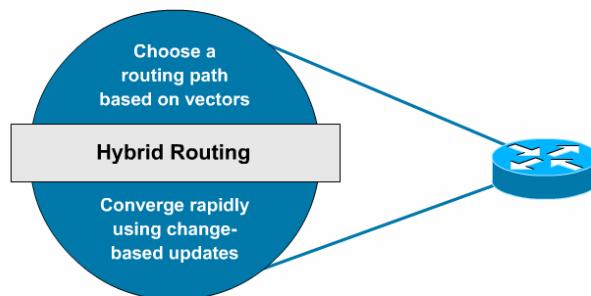
sau khi quá trình này kết thúc thì protocol định tuyến link-state cần rất ít băng thông để gửi các packet LSA phản ánh tình trạng thay đổi topo network.

Bảng so sánh Distance - vector và link - state protocol

Distance-vector	Link-state
• Thu thập topo mạng từ các router kế cận	• Có topo Network tổng thể
• Thêm các vector khoảng cách từ router đến router	• Tính toán đường đi ngắn nhất đến các router
• Cập nhật thường xuyên theo chu kỳ, hội tụ chậm	• Cập nhật khi topo mạng thay đổi, hội tụ nhanh
• Gửi một bản copy của bảng định tuyến đến router kế cận	• Gửi cập nhật link-state đến các router khác

Một loại protocol định tuyến thứ 3 là kết hợp giữa 2 loại trên. Nó có tên gọi là protocol định tuyến kết hợp cân bằng (*balanced hybrid routing protocol*). Protocol này hội tụ nhanh, dùng ít tài nguyên về băng thông, bộ nhớ và các overhead hơn. Diễn hình của loại này là EIGRP.

Hybrid Routing



◆ Share attributes of both distance-vector and link-state routing

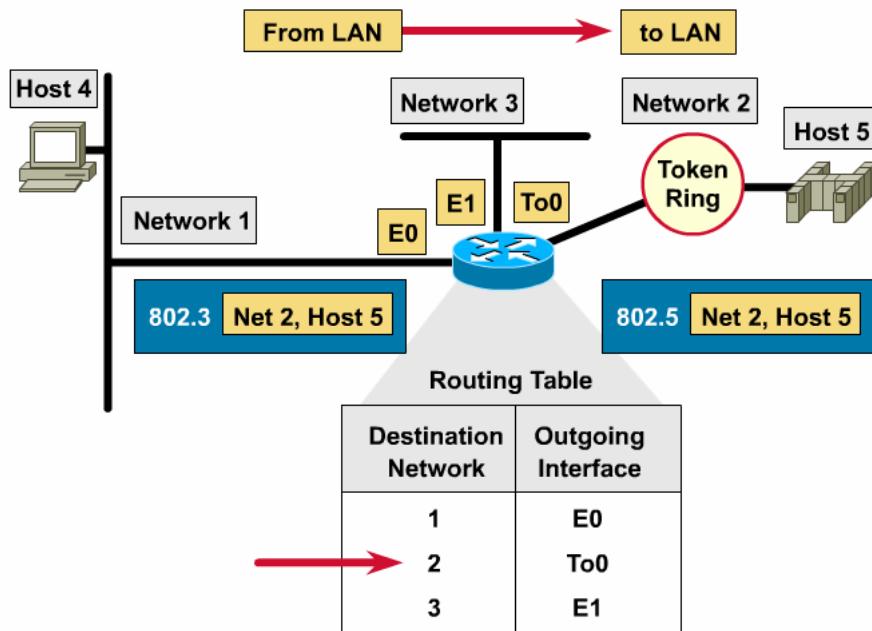
Hình 6.22: Protocol kết hợp cân bằng tận dụng được cả hai ưu điểm của hai loại protocol vừa khảo sát

d)- Một số ví dụ về định tuyến trong mạng LAN và WAN

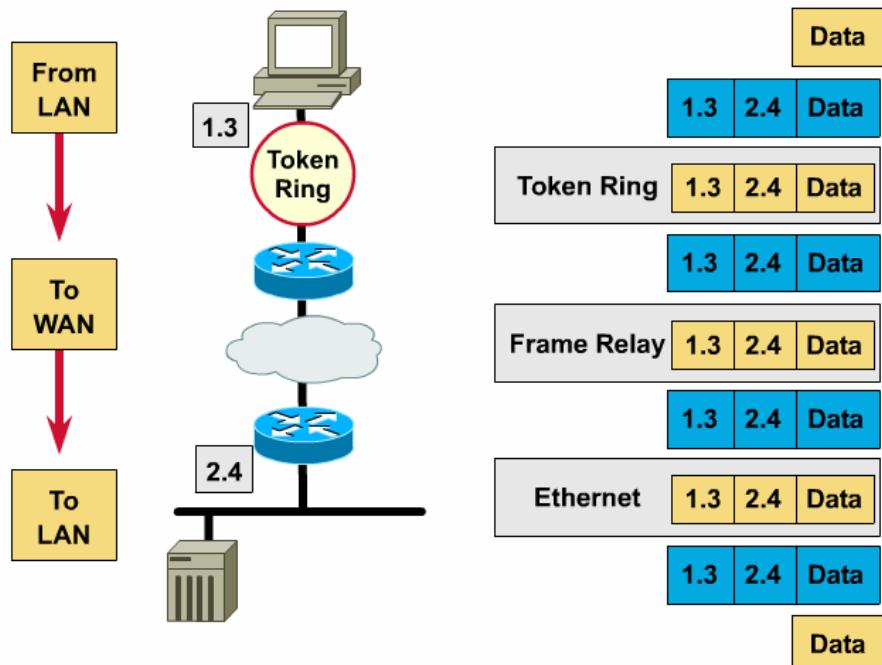
Các router của Cisco có khả năng hỗ trợ đa protocol định tuyến (RIP, IGRP, ...) cũng như đa mô hình truyề (Token Ring, Frame Relay, FDDI, ...). Điều này cung cấp cho các người dùng độ linh hoạt cao nhất khi sử dụng các thiết bị này.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

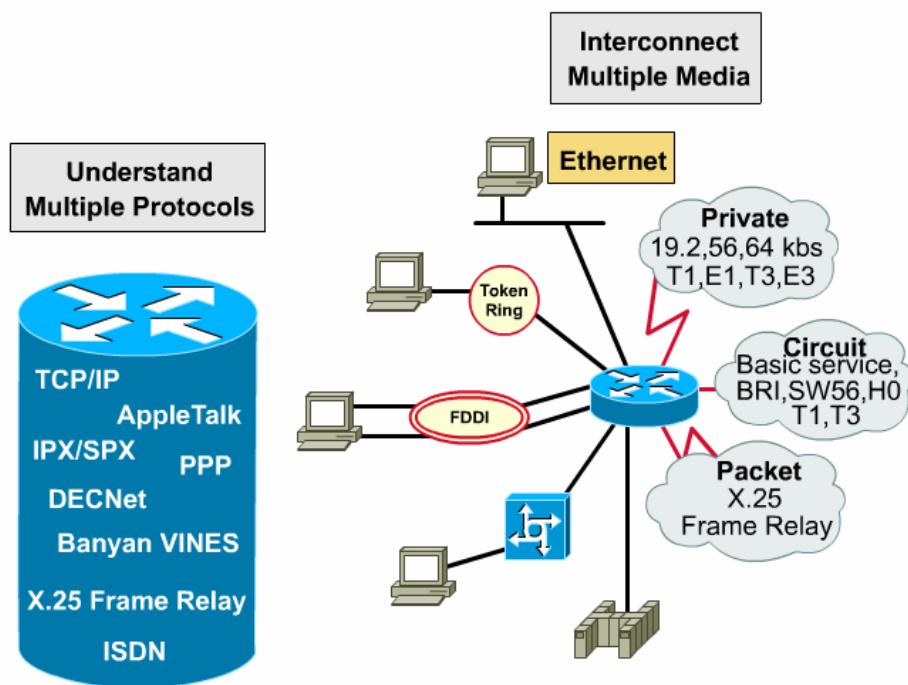
CHƯƠNG 6: ĐỊNH TUYẾN



Hình 6.23: Định tuyến LAN-to-LAN



Hình 6.24: Định tuyến LAN-to-WAN



Hình 6.25: Đa protocol và đa môi trường truyền

Chương 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

7.1. Cài đặt và cấu hình Web Server

7.1.1. Cài đặt Web Server

- ❖ Web là một nhu cầu không thể thiếu hiện nay. Nó là một trong những phương tiện để mọi người trên thế giới có thể trao đổi thông tin. Đứng về phương diện nào đó thì Web có thể xem như một tờ báo điện tử, nó chứa đựng các thông tin để mọi người có thể nắm bắt một cách dễ dàng. Nó có ưu điểm hơn báo chí bình thường thông tin chứa đựng trên đó nhiều hơn, hình ảnh đặc sắc hơn... Nó còn cho phép người xem có thể tương tác phản hồi... và đặc biệt nó tiện lợi rất nhiều trong việc tìm kiếm thông tin. Nó thật sự là một công cụ không thể thiếu đối với chúng ta. Nhưng làm sao để có một trang Web? Ta cần phải có một Web Server. Web Server là nơi chứa những trang web. Web Server còn một nhiệm vụ là quản lý, bảo vệ các trang web. Và để có một Web Server thì chúng ta sẽ từng bước làm như phần trình bày dưới đây.
- ❖ Để cài đặt Web Server thì chúng ta cần một phần mềm hỗ trợ làm điều này. Chúng ta có thể chọn *Apache*. Đây là phần mềm có nhiều tính năng mạnh và linh hoạt dùng để cài Web Server. Nó hỗ trợ đầy đủ những giao thức HTTP trước đây là HTTP/1.1. Có thể cấu hình và mở rộng với những module của công ty thứ ba. Cung cấp source code đầy đủ với license không hạn chế. Chạy trên nhiều hệ điều hành như Windows NT/9x, Netware 5.x, Ó/2 và trên hầu hết các hệ điều hành Unix.
- ❖ Đối với phiên bản Apache trên Windows, ta chỉ cần download gói về (như *apache_2.2.3-win32-x86-no-ssl.msi*) và cài đặt nó. Như vậy, chúng ta có thể sử dụng nó ngay bây giờ nếu chúng ta muốn.
- ❖ Đối với phiên bản phiên bản trên Linux, thường thì chúng ta sẽ cài đặt ngay từ đầu lúc mà chúng ta cài đặt hệ điều hành. Còn nếu chưa cài đặt thì chúng ta có thể cài đặt nó như sau. Chúng ta có thể cài đặt từ các gói đã tạo sẵn với đuôi file thường là deb hoặc rpm, deb là các gói của Debian, dành cho các distro như: Debian, SuSe, Ubuntu ... Còn rpm, đây là các gói cài đặt dành cho Red Hat, viết tắt từ cụm từ RedHat Package Management. Tuy có đuôi file là như vậy nhưng chúng ta có thể cài đặt trên những distro khác ngoài nó ví dụ như các gói đuôi deb vẫn có thể cài đặt trên Red Hat hoặc các gói rpm vẫn có thể cài đặt trên Debian hay Ubuntu ..., chỉ cần có trình quản lý nó. Ví dụ như với các gói đuôi rpm thì ta có trình quản lý nó là rpm còn các gói deb thì có apt-get quản lý nó. Các gói này có thể xem tương tự như trên Windows, các gói cài đặt có đuôi msi hay exe. Tuy nhiên, trên linux còn cho phép ta cài từ mã nguồn. Điều này, rất có ích cho ta chẳng hạn như có thể sửa

lại mã nguồn nếu chúng ta muốn. Lợi ích thứ 2 là chúng ta sẽ có thể quản lý được phần mềm của chúng ta. Vì trên Windows, các gói có mã nguồn là đóng vì vậy chúng ta không thể làm được điều này. Với Linux, chúng ta có thể chọn gói mã nguồn như *httpd-2.2.3.tar.gz*. Đây là gói miễn phí, chúng ta hoàn toàn có thể download được trên mạng. Và để cài đặt gói này, chúng ta sẽ làm như sau:

- Giải nén mã nguồn dùng lệnh: *tar xvzf httpd-2.2.3.tar.gz*
- Di chuyển vào thư mục chứa mã nguồn: *cd httpd-2.2.3*
- Sau đó, chúng ta sẽ lần lượt cài đặt nó:
./configure && make && make install.
- Nếu cài đặt trên Debian hay Ubuntu thì gõ lệnh: *apt-get install apache*
- Còn nếu cài đặt từ những gói rpm thì gõ lệnh: *rpm -ivh httpd-2.2.3.rpm*
- Bây giờ, chúng ta có thể chạy Web Server nếu chúng ta muốn. Tuy nhiên, vẫn có khi gặp trường hợp không thể khởi động được như: lỗi vì đã có phần mềm nào đó chạy trên port mà Web Server ta sẽ chạy. Điều này có thể khắc phục được một cách dễ dàng, bằng cách tắt chương trình chạy trên port đó đi. Và bây giờ khởi động lại là có thể chạy được.
- Để khởi động hay tạm dừng hay tái khởi động apache ta script sau:
/etc/init.d/httpd start/stop/restart

Hoặc dùng lệnh:

```
#chkconfig httpd on  
#service httpd restart
```

- Tuy nhiên, để có thể hiểu cũng như có thể vận hành theo đúng ý muốn của chúng ta thì ta cần phải hiểu và cũng như phải tận tay cấu hình nó. Và việc cấu hình đó, chúng tôi sẽ trình bày trong mục 2 phần cấu hình Web Server dưới đây.

7.1.2. Cấu hình Web Server

Các tập tin và thư mục cấu của Apache:

- **/etc/httpd/conf**: thư mục lưu giữ các tập tin cấu hình như *httpd.conf*.
- **/etc/httpd/modules**: lưu giữ các module của Web Server.
- **/etc/httpd/logs**: lưu các tập tin log của Apache.
- **/var/www/html**: lưu các trang web.
- **/var/www/cgi-bin**: lưu các script sử dụng cho các trang web.

Tập tin cấu hình Apache được tạo thành từ nhiều chỉ dẫn (directive) khác nhau. Mỗi dòng hoặc mỗi một directive và phục vụ cho một cấu hình riêng biệt. Có những directive có ảnh hưởng với nhau. Những dòng bắt đầu bằng dấu # là những dòng chú thích. Sau đây là những directive quan trọng khi cấu hình Web Server.

a). ServerName:

Cú pháp: *ServerName <hostname>:port*

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER
CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

Trong đó, hostname là tên máy tính của Server. Nó được dùng trong việc tạo ra những URL chuyển tiếp (direction URL). Nếu không chỉ ra, server sẽ cố gắng suy luận từ địa chỉ IP của nó. Tuy nhiên, điều này có thể không tin cậy hoặc không trả ra tên máy tính đúng.

Ví dụ: ServerName www.nguyenhongthai.hcmut.edu.vn

b). ServerAdmin: địa chỉ email của người quản trị hệ thống

Cú pháp: ServerAdmin <địa chỉ email>

Ví dụ: ServerAdmin webmaster@hcmut.edu.vn

c). ServerType: quy định cách nạp chương trình. Có 2 cách:

inetd: chạy từ các init level.

standalone: chạy từ hệ thống.

Cú pháp: ServerType <inetd/standalone>

Ví dụ: ServerType standalone

d). DocumentRoot: cấu hình thư mục gốc lưu trữ nội dung của Website.

Web Server sẽ lấy những tập tin trong thư mục này phục vụ cho yêu cầu của client

Cú pháp: DocumentRoot <đường dẫn thư mục>

Ví dụ: DocumentRoot /usr/web

e). ServerRoot: chỉ dẫn vị trí cài đặt chương trình Apache.

Cú pháp: ServerRoot <vị trí cài đặt Apache>

Ví dụ: ServerRoot /user/local/apache

f). ErrorLog: chỉ ra tập tin để server ghi vào bất kỳ những lỗi nào mà nó gặp phải.

Cú pháp: ErrorLog <vị trí tập tin log>

Ví dụ: ErrorLog logs/error_log

g). DirectoryIndex: các tập tin mặc định được truy vấn khi truy cập trang Web.

Cú pháp: DirectoryIndex <danh sách các tập tin>

Ví dụ: DirectoryIndex index.html

h). MaxClients: quy định số yêu cầu tối đa từ các client có thể gửi đồng thời đến server.

Cú pháp: MaxClients <số kết nối tối đa cho phép>

Ví dụ: MaxClients 256

i). Listen: quy định địa chỉ IP hoặc cổng mà Apache nhận kết nối từ Client.

Cú pháp: Listen <Port/IP>

Ví dụ: Listen 80

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

j). **BindAddress:** quy định địa chỉ card mạng để chạy Apache trên Server.

Cú pháp: BindAddress <IP/*>

Sử dụng dấu "*" để có thể sử dụng tất cả các địa chỉ trên máy.

Ví dụ: BindAddress 172.28.24.199

k). **TimeOut:** quy định thời gian sống của một kết nối (được tính bằng giây).

Cú pháp: TimeOut <thời gian tối đa cho một kết nối>

Ví dụ: TimeOut 300

l). **KeepAlive:** cho phép hoặc không cho phép client gửi được nhiều yêu cầu dựa trên một kết nối đến với Web Server.

Cú pháp: KeepAlive <On/Off>

Ví dụ: KeepAlive On

m). **MaxKeepAliveRequests:** số Request tối đa trên một kết nối (nếu cho phép nhiều Request trên một kết nối).

Cú pháp: MaxKeepAliveRequests <số Request>

Ví dụ: MaxKeepAliveRequests 100

n). **KeepAliveTimeout:** quy định thời gian để chờ một Request kế tiếp từ cùng một client trên cùng một kết nối (được tính bằng giây).

Cú pháp: KeepAliveTimeout <thời gian>

Ví dụ: KeepAliveTimeout 15

o). **Alias:** ánh xạ đường dẫn cục bộ (không nằm trong DocumentRoot) thành tên đường dẫn địa chỉ URL.

Cú pháp: Alias <đường dẫn http><đường dẫn cục bộ>

Ví dụ: Alias /doc /usr/share/doc

Khi truy cập `http://www.nguyenhongthai.hcmut.edu.vn/doc`, nó sẽ vào /usr/share/doc. Để giới hạn việc truy cập của người dùng ta có thể kết hợp với Directory directive.

Ví dụ:

```
Alias /doc /usr/share/doc
<Directory /usr/share/doc>
    AuthType Basic      # kiểu authentication sẽ sử dụng là Basic
    AuthName intranet    # đặt tên cho sự chứng thực là intranet
    AuthUserFile /etc/httpd/passwd    # vị trí của tập tin password
    Require user hongthai minhtri #user cho phép truy cập tài nguyên
    Allow from internal.hcmut.edu.vn # cho phép truy cập từ đchi này
</Directory>
```

p). **UserDir:** cho phép người dùng tạo Home page của user trên Web Server.

Cú pháp:

```
<IfModule mod_userdir.c>
    #UserDir Disables    ## để thực thi cơ chế enable UserDir
    UserDir www    ## Khai báo thư mục chứa Website của user
</IfModule>
<Directory /home/*/www>
    ...
</Directory>
```

Trong thư mục Home Directory của người dùng tạo thư mục www. Ví dụ /home/nhthai/www. Khi đó, cú pháp truy cập từ Web Browser có dạng: <http://www.nguyenhongthai.hcmut.edu.vn/~<tênUser>>, tức trong trường hợp này là <http://www.nguyenhongthai.hcmut.edu.vn/~nhthai>. Khi người dùng có gắng truy cập đến thư mục của mình, có thể gặp một message lỗi “Forbidden”. Điều này có thể là quyền truy cập đến home directory của người dùng bị giới hạn. Như vậy để khắc phục lỗi trên, chúng ta cần giới hạn lại quyền truy cập home directory của người dùng với những câu lệnh như sau:

```
chown nhthai /home/nhthai /home/nhthai/www
chmod 750 /home/nhthai /home/nhthai/www
```

q). **VirtualHost:** là tính năng của Apache, giúp ta duy trì nhiều hơn một web server trên một máy tính. Nhiều tên cùng chia sẻ một địa chỉ IP gọi là named-based virtual hosting và sử dụng những địa chỉ IP khác nhau cho từng domain gọi là IP-based virtual hosting.

➤ **IP-based Virtual Host:** Virtual Host dựa trên IP yêu cầu những server phải có một địa chỉ IP khác nhau cho mỗi virtual host dựa trên IP. Như vậy, một máy tính phải có nhiều interface hay sử dụng cơ chế virtual interface mà những hệ điều hành sau hỗ trợ. Nếu máy của chúng ta có một địa chỉ IP, 172.28.24.199, chúng ta có thể cấu hình một địa chỉ IP khác trên cùng một card mạng như sau:

```
ifconfig eth0:1 172.28.24.198 netmask 255.255.255.0 up
```

Sau đó, chúng ta mô tả thông tin cấu hình trong file httpd.conf

```
<VirtualHost *> ; VirtualHost default
```

```
...
DocumentRoot      /tmp
ServerName        www.domain
```

```
...
</VirtualHost>
<VirtualHost 172.28.24.199>;VirtualHost cho site 1
```

```
...
DocumentRoot      /home/www/site1
ServerName        www1.domain
```

```
</VirtualHost>
<VirtualHost 172.28.24.198>;VirtualHost cho site 2
...
DocumentRoot      /home/www/site2
ServerName        www2.domain
...
</VirtualHost>
```

➤ **Name-based Virtual Host:** IP-based Virtual Hosts dựa vào địa chỉ IP để quyết định *Virtual Host* nào đúng để truy cập. Vì thế, chúng ta cần phải có địa chỉ khác nhau cho mỗi *Virtual Host*. Với Named-based Virtual Host, server dựa vào HTTP header của client để biết được hostname. Sử dụng kỹ thuật này, một địa chỉ IP có thể có nhiều tên máy tính khác nhau. Named-based Virtual Host rất đơn giản, chúng ta chỉ cần cấu hình DNS sao cho nó phân giải mỗi tên máy đúng với một địa chỉ IP và sau đó cấu hình Apache để tổ chức những web server cho những miền khác nhau.

7.2. Cài đặt và cấu hình FTP Server

7.2.1. Cài đặt FTP Server

- ❖ Cũng như Web, FTP cũng là một công cụ không thể thiếu trong lĩnh vực mạng. FTP là chữ viết tắt của File Transfer Protocol. Giao thức này được xây dựng dựa trên chuẩn TCP. FTP cung cấp cơ chế truyền tin dưới dạng file thông qua mạng TCP/IP. FTP là dịch vụ đặc biệt vì nó dùng đến 2 cổng: cổng 20 dùng để truyền dữ liệu (data port) và cổng 21 dùng để truyền lệnh (command port). FTP hoạt động ở một trong 2 cơ chế: cơ chế chủ động (active) và cơ chế bị động (passive).
- ❖ Khi FTP Server hoạt động ở cơ chế chủ động, client không chủ động tạo kết nối thật sự vào cổng dữ liệu của FTP Server, mà chỉ đơn giản là thông báo cho server biết rằng nó đang lắng nghe trên cổng nào và server phải kết nối ngược về client vào cổng đó. Trên quan điểm firewall đối với máy client điều này giống như một hệ thống bên ngoài khởi tạo kết nối vào hệ thống bên trong và điều này thường bị ngăn chặn trên hầu hết hệ thống firewall.
- ❖ Để giải quyết vấn đề server phải tạo kết nối đến client, một phương thức kết nối FTP khác đã được phát triển. Phương thức này gọi là FTP thụ động hoặc PASV (là lệnh mà client gửi cho server để báo cho biết nó đang ở chế độ passive). Trong khi FTP ở chế độ thụ động giải quyết được vấn đề phía client thì nó gây ra nhiều vấn đề khác về phía server. Thứ nhất là cho phép máy ở xa kết nối vào cổng bất kỳ lớn hơn 1024 của server. Điều này khá nguy hiểm trừ khi FTP cho phép mô tả dãy các cổng lớn hơn hoặc bằng 1024 mà FTP sẽ dùng. Vấn đề thứ hai là, một số FTP client lại không hỗ trợ chế độ thụ động. Ví dụ tiện ích FTP mà

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

- Solaris cung cấp không hỗ trợ FTP thụ động. Khi đó, cần phải dùng thêm trình FTP client. Một lưu ý khác là hầu hết các trình duyệt Web chỉ hỗ trợ FTP thụ động khi truy cập FTP server theo đường URL `ftp://`.
- ❖ Chương trình FTP Server: FTP Server là một máy chủ lưu giữ những tài nguyên và hỗ trợ giao thức FTP để giao tiếp với những máy tính khác. Nó cho phép truyền dữ liệu trên Internet. Một số chương trình FTP Server sử dụng trên Linux như: vsftpd, Wu-ftp, PureFTPD, ProFTPD. Trên Windows, ta có thể sử dụng phiên bản hỗ trợ của MicroSoft hoặc có thể sử dụng phiên bản của Golden như: *Golden-FTP-server-PRO-setup.exe* (*bản đòi hỏi license*) hoặc có thể dùng bản miễn phí *GoldenFTPServer-setup.exe*.
 - ❖ Về phần cài đặt, nếu cài trên Windows sử dụng phiên bản hỗ trợ của MicroSoft, ta vào *Control Panel* → *Add/Remove Program* → *Add/Remove Windows Components* → *Chọn IIS* → *Chọn install*. Còn dùng phiên bản của Golden thì ta chỉ cài gói cài đặt duy nhất.
 - ❖ Bây giờ, chúng tôi sẽ trình bày phần cài đặt từ source cho linux. Chọn gói cài đặt là *vsftpd-2.0.5.tar.gz*. Các bước sẽ tiến hành như sau:

```
# tar xvzf vsftpd-2.0.5.tar.gz      ## Giải nén mã nguồn
# cd vsftpd-2.0.5                  ## Di chuyển đến thư mục chứa mã nguồn
# make                                ## Tạo binary file
# make /var/ftp                      ## Tạo thư mục chứa các file để truy cập FTP
# useradd -d /var/ftp ftp             ## Tạo tài khoản người dùng vào thư mục chỉ định
# chown root.root /var/ftp            ## Chuyển quyền sở hữu sang root
# chmod go-w /var/ftp                ## Không cho phép ghi đổi với người dùng khác
# make install                       ## Cài đặt FTP Server
```

Nếu không thực hiện được lệnh ‘make install’ thì ta có thể làm như sau:

```
# cp vsftpd /usr/local/sbin/vsftpd
# cp vsftpd.conf.5 /usr/local/man/man5
# cp vsftpd.8 /usr/local/man/man8
```

Tiếp theo, là chép file cấu hình vào thư mục /etc:

```
# cp vsftpd.conf /etc
```

Cuối cùng, ta cần chỉnh sửa một chút để cho phép làm việc theo kiểu nào. Nếu cho chạy theo kiểu *standalone* thì thêm dòng *listen=YES* vào cuối file */etc/vsftpd.conf*. Còn nếu muốn cho chạy với inetd thì thêm dòng *ftp stream tcp nowait root /usr/sbin/tcpd /usr/local/sbin/vsftpd* vào file */etc/inetd.conf*.
 - ❖ Nếu không quen với việc cài đặt từ mã nguồn, ta có thể chọn các cài đặt đã làm sẵn như những gói có đuôi deb hoặc rpm. Và việc cài đặt các gói này tương tự như cài đặt Web Server.
 - ❖ Vsftpd là một package mới. Nó được phát triển xoay quanh tính năng nhanh, ổn định và an toàn. Vsftpd có khả năng quản lý số lượng kết nối lớn một cách hiệu quả và an toàn.
Để khởi động và dừng vsftpd:
`# service vsftpd start/stop/restart`

Hoặc sử dụng lệnh:
/etc/init.d/vsftpd start/stop/restart

7.2.2. Cấu hình FTP Server

Những tập tin và thư mục thường được qua tay khi cấu hình vsftpd server:

- **/etc/pam.d/vsftpd**: tập tin cấu hình PAM cho vsftpd. Tập tin này định nghĩa những yêu cầu mà người dùng phải cung cấp khi đăng nhập vào ftp server.
-  PAM là chữ viết tắt từ Pluggable Authentication Modules, tạm dịch là các mô-đun kiểm tra có thể cắm được. PAM được phát triển cho hệ thống Solaris từ Sun Microsystems. Dự án Linux-PAM làm cho PAM có sẵn đối với hệ điều hành Linux. PAM là bộ thư viện dùng chung để cấp phát các đặc quyền cho ứng dụng liên quan đến PAM.
- **/etc/vsftpd/vsftpd.conf**: tập tin cấu hình vsftpd server.
- **/etc/vsftpd.ftpusers**: liệt kê những người dùng không được login vào vsftpd. Mặc định, danh sách những người dùng này gồm root, bin, deamon và những người dùng khác.
- **/etc/vsftpd.user_list**: tập tin này được cấu hình để cấm hay cho phép những dùng được liệt kê truy cập ftp server. Điều này phụ thuộc vào tùy chọn userlist_deny được xét YES hay NO trong tập tin *vsftpd.conf*. Nếu những người dùng đã liệt kê trong tập tin này thì không được xuất hiện trong *vsftpd.ftpusers*.
- **/var/ftp**: thư mục chứa các tập tin đáp ứng cho vsftpd. Nó cũng chứa thư mục pub cho người dùng *anonymous* (có thể hiểu là người dùng ẩn danh). Thư mục này chỉ có thể đọc, chỉ có root mới có khả năng ghi.

7.3. Cấu hình để LAN có thể truy cập mạng bên ngoài

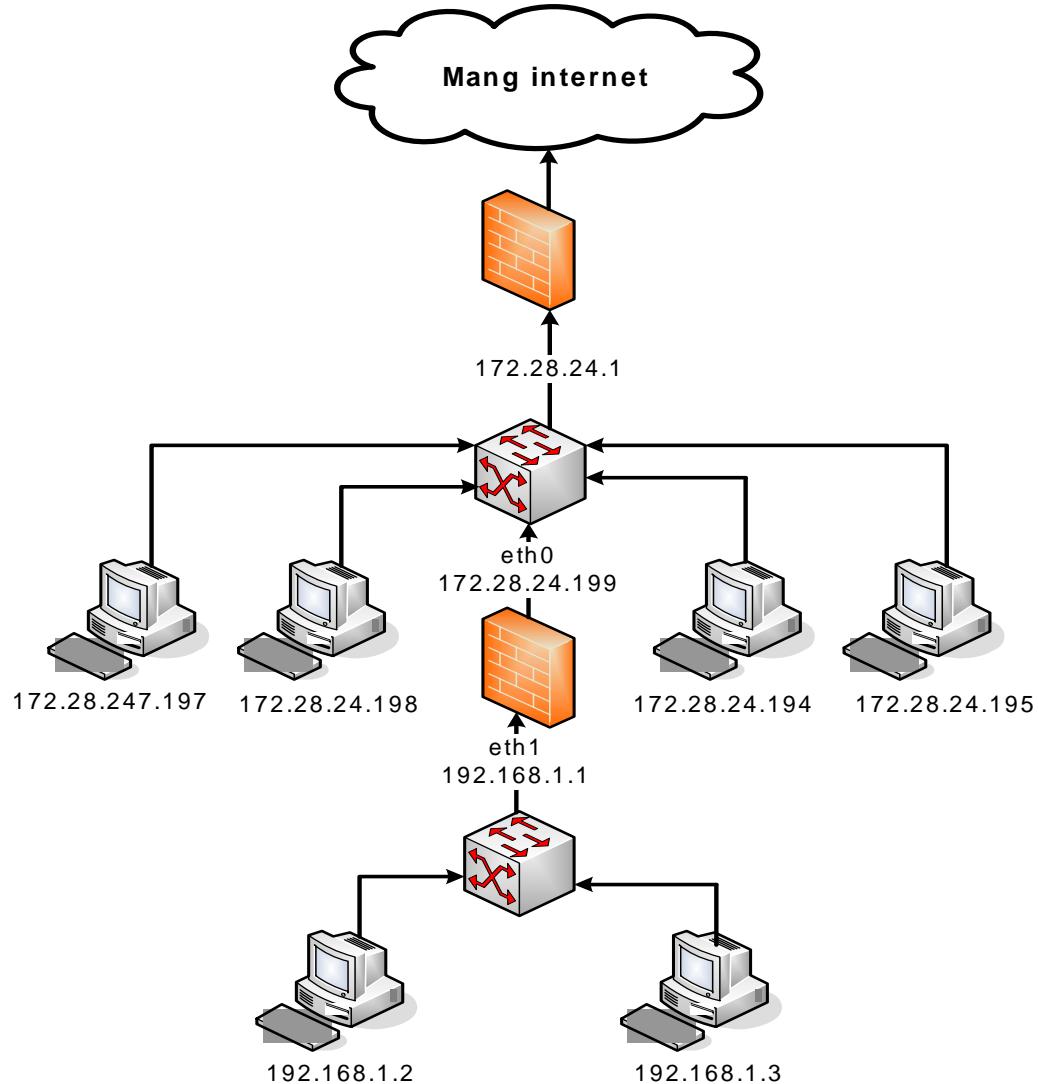
Việc cấu hình để các máy từ LAN có thể truy cập ra bên ngoài internet. Đây là mô hình cho phép nhiều máy cùng chia sẻ một IP public. Để có thể làm điều này trên hệ điều hành Linux, ta có thể chọn lựa tool chạy rất ổn định, đó là iptables để cấu hình. Ngoài mục đích trên, iptables còn có thể dùng để lọc gói tin rất hiệu quả. Chúng ta có thể cho phép những gói tin nào đó hay chặn những gói tin nào đó mà ta muốn. Để thực hiện một cách cụ thể, chúng tôi đưa ra một mô hình cụ thể tự chúng tôi thiết lập và đã cho chạy thực tế. Sử dụng mạng máy tính cụ thể, đó là mạng máy tính của phòng máy tính khoa điện - điện tử. Với mô hình thiết lập như hình dưới đây.

Việc cấu hình có thể được giải thích như sau. Để một gói tin đi từ một mạng LAN bên trong ra mạng bên ngoài thì ta cần phải thay đổi địa chỉ nguồn của gói tin để khi ra khỏi mạng LAN mà muốn định tuyến được thì mạng đó đòi hỏi phải cùng subnet và đồng thời đòi hỏi địa chỉ nguồn phải được đổi trước khi nó thực hiện định tuyến ra ngoài. Do đó, ta thực hiện Source NAT. Và cứ như thế nó sẽ có thể đi ra ngoài mạng

internet. Và việc cấu hình Source ta có thể chọn iptables. Chúng tôi sẽ trình bày việc cấu hình SNAT tại máy dùng làm gateway của mạng 192.168.1.0/24. Trình tự các bước sẽ làm như sau:

```
# modprobe ipt_MASQUERADE ## Load mô-đun ip_MASQUERADE
# iptables -F ## Xóa các luật trong bảng filter
# iptables -t nat -F ## Xóa các luật trong bảng nat
# iptables -t mangle -F ## Xóa các luật trong bảng mangle
## Nếu gói tin đi từ 192.168.1.0/24 ra mạng ngoài thì thực hiện đổi địa chỉ
##### nguồn thành 172.28.24.199
# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 172.28.24.199
## Cho phép các interface có thể forward được với nhau
# echo 1 > /proc/sys/net/ipv4/ip_forward
## Cho phép các gói tin từ các kết nối đã thiết lập hoặc có mối liên hệ với
#### kết nối hiện tại. Lệnh này có ý nghĩa trong trường hợp kết nối FTP
# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
## Cho phép các gói tin đến từ những interface không phải eth0
# iptables -A INPUT -m state --state NEW -i ! eth0 -j ACCEPT
## Mặc định là DROP (cấm)
# iptables -P INPUT DROP
## Nếu gói tin forward từ eth0 đến eth0 thì ngăn lại và trả thông báo về
#### cho người gửi biết
# iptables -A FORWARD -i eth0 -o eth0 -j REJECT
Trong trường hợp đường nối ra mạng ngoài không phải là card ethernet
mà là dial up thì ta sẽ đổi eth0 thành ppp0
```

Với việc cấu hình iptables trên trong trường hợp mô hình mạng như hình 7.1.Giả máy 192.168.1.2 muốn gửi Request đến máy 172.28.2.2. Suy ra, gói tin sẽ có địa chỉ nguồn là 192.168.1.2 và địa chỉ đích là 172.28.2.2. Nó sẽ định tuyến đến gateway vì địa chỉ đích không cùng subnet của địa chỉ nguồn, tại đây iptables sẽ thiết lập lại gói tin, tức sẽ đổi địa chỉ nguồn thành 172.28.24.199 còn địa chỉ đích giữ nguyên. Tiếp theo, nó mới thực hiện định tuyến. Và việc định tuyến sẽ giống như trên, nó xem lại gói tin rõ ràng địa chỉ đích không cùng subnet của địa chỉ nguồn, nó sẽ định tuyến đến gateway và sẽ thực hiện đổi địa chỉ nguồn tại đây. Việc định tuyến cứ tiếp tục như thế. Đến khi nó thấy rằng gói tin có địa chỉ đích có cùng subnet với địa chỉ nguồn thì nó xác định được máy cần đến nằm tại mạng này. Và như thế, nó sẽ không cần đến gateway mà chỉ cần đến switch và chuyển gói tin thẳng đến đích. Quá trình trình gửi Response từ máy 172.28.2.2 về máy 192.168.1.2, nó sẽ xem header mà định tuyến về đích.



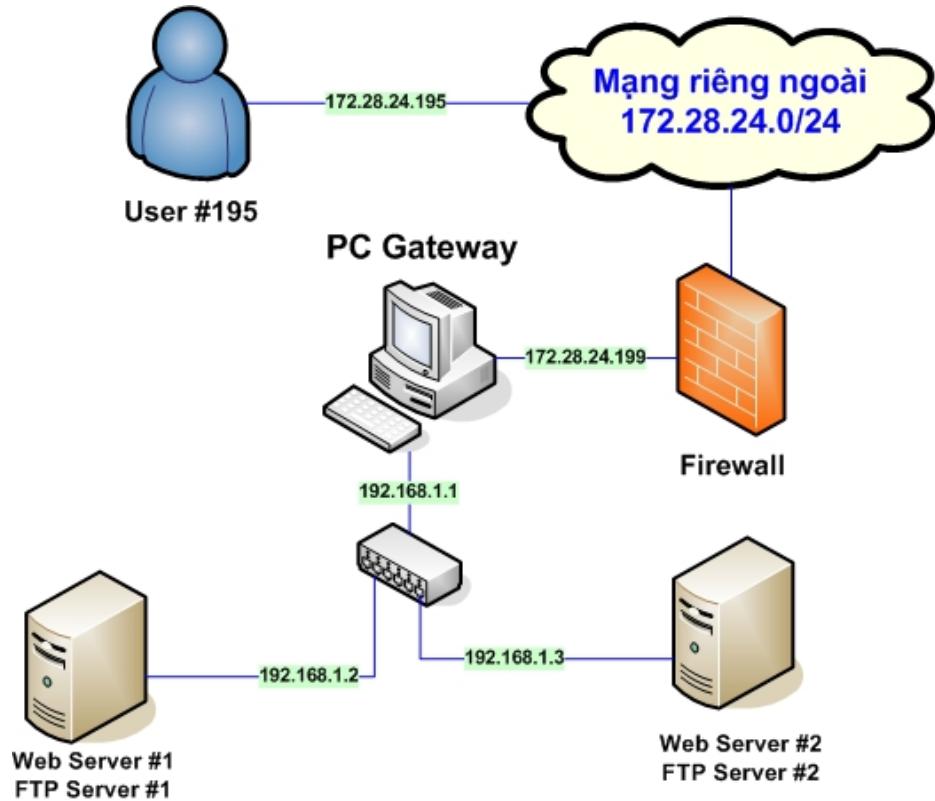
Hình 7.1: Mô hình mạng LAN tự thiết lập

7.4. Cấu hình để mạng bên ngoài có thể truy cập được các Server

Việc cấu hình để mạng bên ngoài có thể truy cập được các Server từ một LAN nội bộ. Đây cũng là mô hình rất phổ biến. Nó có thể làm công việc cân bằng tải vừa tạo tính an toàn cho mạng nội bộ. Phương pháp thực hiện điều này có thể lý giải ngắn gọn như sau: người dùng internet muốn truy cập đến một trang web nào đó thì trên URL họ chỉ gõ địa chỉ của Server ảo (hay còn gọi là VIP, viết tắt từ cụm từ Virtual IP). Và Server ảo này cũng là gateway, tại đây ta cũng thiết lập tường lửa. Tại đây, nó sẽ xem xét địa chỉ cũng như port, sau đó nó sẽ forward đến server cần thiết

Mô hình cấu hình server do chúng tôi tự thiết lập được minh họa ở hình dưới đây. Và trình tự cấu hình sẽ lần lượt như sau:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward ## Cho phép IP forwarding
## Load các modules
# modprobe ip_conntrack_ftp
# modprobe ip_nat_ftp
## Thiết lập các chính sách mặc định và giải phóng các bảng của iptables
# iptables -t nat -F
# iptables -P INPUT ACCEPT
# iptables -F INPUT
# iptables -P OUTPUT ACCEPT
# iptables -F OUTPUT
# iptables -P FORWARD ACCEPT
# iptables -F FORWARD
## Cấu hình Web Server trên máy 192.168.1.2
## Đổi địa chỉ đích của gói tin khi gói tin có địa chỉ đích là 172.28.24.199
## port 80, đi vào eth0, dùng giao thức tcp thành 192.168.1.2 port 8080
# iptables -t nat -A PREROUTING -d 172.28.24.199 -i eth0 -p tcp \
--dport 80 -j DNAT --to-destination 192.168.1.2:8080
# Cho phép các gói tin trên có thể forward
# iptables -A FORWARD -p tcp -i eth0 -d 192.168.1.2 --dport 8080 \
-j ACCEPT
## Tương tự, ta cấu hình Web Server trên máy 192.168.1.3
# iptables -t nat -A PREROUTING -d 172.28.24.199 -i eth0 -p tcp \
--dport 8888 -j DNAT --to-destination 192.168.1.3:80
## Cấu hình FTP Server trên máy 192.168.1.3
# iptables -A FORWARD -p tcp -i eth0 -d 192.168.1.3 --dport 80 \
-j ACCEPT
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 20:21 \
-j DNAT --to-destination 192.168.1.2:21
# iptables -A FORWARD -p tcp -i eth0 -d 192.168.1.2 --dport 21 \
-j ACCEPT
## Tương tự, ta cấu hình cho máy 192.168.1.3
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 2020:2121 \
-j DNAT --to-destination 192.168.1.3:21
# iptables -A FORWARD -p tcp -i eth0 -d 192.168.1.3 --dport 21 \
-j ACCEPT
```



MÔ HÌNH MẠNG LAN TỰ THIẾT LẬP

Hình 7.2: Mô hình mạng LAN cùng với các server

7.5. Kết quả của việc cấu hình trên

Kết quả của việc cấu hình iptables sẽ được lưu trong file /etc/sysconfig/iptables như sau:

```
# Generated by iptables-save v1.2.8 on Thu Nov 9 15:47:54 2006
*nat
:PREROUTING ACCEPT [4169:438355]
:POSTROUTING ACCEPT [106:6312]
:OUTPUT ACCEPT [22:1332]
-A PREROUTING -d 172.28.24.199 -i eth0 -p tcp -m tcp --dport 80 -j DNAT --to-
destination 192.168.1.2:8080
-A PREROUTING -d 172.28.24.199 -i eth0 -p tcp -m tcp --dport 8888 -j DNAT --
to-destination 192.168.1.3:80
-A PREROUTING -i eth0 -p tcp -m tcp --dport 20:21 -j DNAT --to-destination
192.168.1.2:21
-A PREROUTING -i eth0 -p tcp -m tcp --dport 2020:2121 -j DNAT --to-destination
192.168.1.3:21
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

```
-A POSTROUTING -o eth0 -j SNAT --to-source 172.28.24.199
COMMIT
# Completed on Thu Nov  9 15:47:54 2006
# Generated by iptables-save v1.2.8 on Thu Nov  9 15:47:54 2006
*filter
:INPUT DROP [4011:414080]
:FORWARD ACCEPT [552:57100]
:OUTPUT ACCEPT [393:43195]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i ! eth0 -m state --state NEW -j ACCEPT
-A FORWARD -d 192.168.1.3 -i eth0 -p tcp -m tcp --dport 80 -j ACCEPT
COMMIT
# Completed on Thu Nov  9 15:47:54 2006
# Generated by iptables-save v1.2.8 on Thu Nov  9 15:47:54 2006
*mangle
:PREROUTING ACCEPT [5114:853418]
:INPUT ACCEPT [4416:773589]
:FORWARD ACCEPT [552:57100]
:OUTPUT ACCEPT [393:43195]
:POSTROUTING ACCEPT [945:100295]
COMMIT
# Completed on Thu Nov  9 15:47:54 2006
```

Kết quả khi thực hiện *traceroute* từ máy 192.168.1.2 đến máy khác như sau:

```
sysadmin@debian:~$ traceroute 172.28.24.195
traceroute to 172.28.24.195 (172.28.24.195), 30 hops max, 38 byte packets
1 192.168.1.1 (192.168.1.1) 2.541 ms 3.409 ms 0.142 ms
2 172.28.24.195 (172.28.24.195) 0.298 ms 3.125 ms 0.256 ms
```

```
sysadmin@debian:~$ traceroute 172.28.2.2
traceroute to 172.28.2.2 (172.28.2.2), 30 hops max, 38 byte packets
1 192.168.1.1 (192.168.1.1) 0.259 ms 4.546 ms 0.185 ms
2 172.28.24.1 (172.28.24.1) 1.182 ms 2.777 ms 0.820 ms
3 hcmut-server.hcmut.edu.vn (172.28.2.2) 0.988 ms 4.159 ms 5.069 ms
sysadmin@debian:~$
```

Kết quả khi thực hiện *mtr* từ máy 192.168.1.2 đến máy khác như sau:

```
My traceroute [v0.67]
debian (0.0.0.0)(tos=0x0 psize=64 bitpattern=0x00) Wed Nov 15 11:11:31
2006
Keys: Help  Display mode  Restart statistics  Order of fields  quit
      Packets          Pings
      Host           Loss%  Snt  Last  Avg  Best  Wrst StDev
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER
CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

1. 192.168.1.1	0.0%	70	0.3	1.4	0.2	59.0	7.2
2. 172.28.24.195	0.0%	70	0.4	6.8	0.3	292.3	35.7

```
My traceroute [v0.67]
debian (0.0.0.0)(tos=0x0 psize=64 bitpattern=0x00) Wed Nov 15 11:13:13
2006
Keys: Help Display mode Restart statistics Order of fields quit
Packets Pings
Host Loss% Snt Last Avg Best Wrst StDev
1. 192.168.1.1 0.0% 12 0.2 0.3 0.2 1.2 0.3
2. 172.28.24.1 0.0% 12 0.9 1.5 0.8 6.0 1.6
3. hcmut-server.hcmut.edu.vn 0.0% 12 0.4 0.9 0.4 4.7 1.3
```

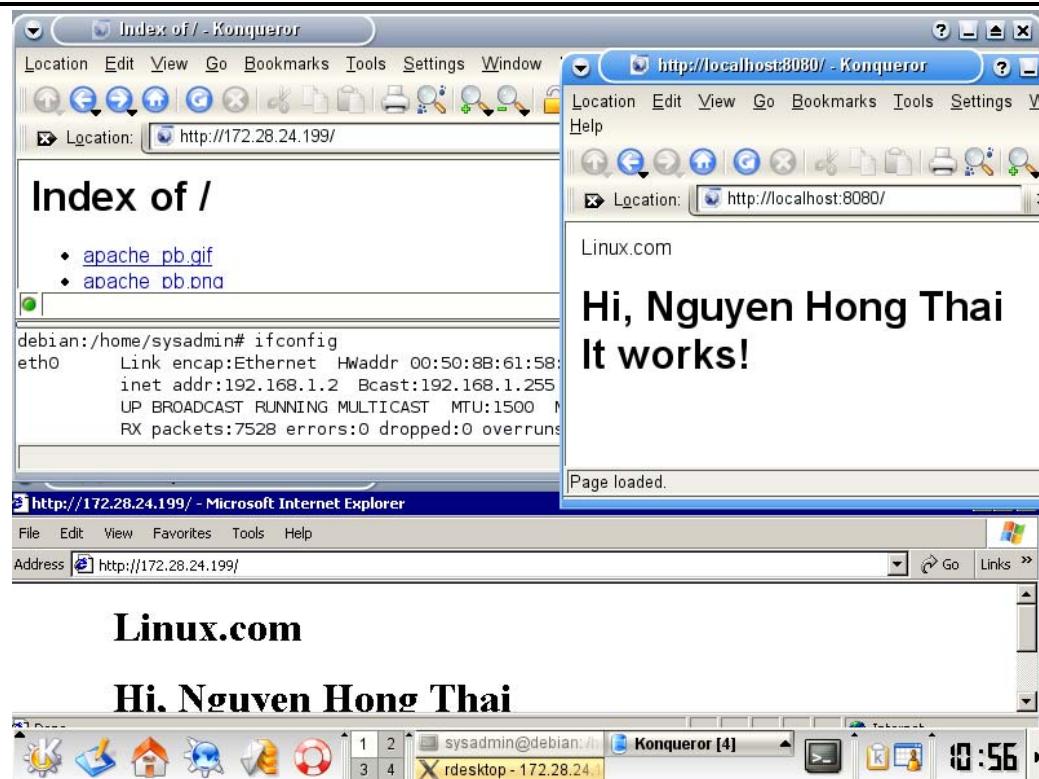
Kết quả từ máy 192.168.1.2, dùng Konqueror để truy cập Web server. Nếu ở URL gõ `http://localhost:8080/` máy tính sẽ hiểu là truy cập Web Server trên máy máy 192.168.1.2 trên port 8080. Điều này có thể dễ dàng nhận ra vì dùng lệnh `ifconfig` thì thấy rằng địa chỉ 192.168.1.2 chính là địa chỉ của interface eth0 của máy 192.168.1.2. Còn nếu ở URL gõ `http://172.28.24.199` thì nó sẽ hiểu địa chỉ này không phải địa chỉ trong mạng của nó. Do đó, nó gởi đến gateway và trên gateway sẽ định tuyến gói theo quy luật mà iptables đã cài ở trên (ở phần *cài đặt để LAN có thể truy cập ra mạng bên ngoài*). Nó ánh xạ địa chỉ 172.28.24.199 port 80 → 192.168.1.2 port 8080. Vì vậy, mà tuy gõ hai địa chỉ ở URL khác nhau nhưng kết quả trả về từ web server là giống nhau. Đồng thời, trên máy 192.168.1.2 ta đăng nhập từ xa đến một máy khác ngoài mạng thử dùng Internet Explorer để truy cập `http://172.28.24.199` thì ta vẫn nhận được kết quả từ web server hoàn toàn giống với 2 kết quả trên.

Còn nếu trên máy dùng làm gateway ta dùng Mozilla Firefox để truy cập `http://localhost/` thì nó sẽ hiểu là truy cập Web Server trên máy này mặc dù máy này có địa chỉ 172.28.24.199. Ta rõ ràng thấy sự khác biệt trong điều này, mặc dù cùng địa chỉ 172.28.24.199 và cùng port 80 nhưng ở những vị trí truy cập khác nhau thì cho kết quả khác nhau.

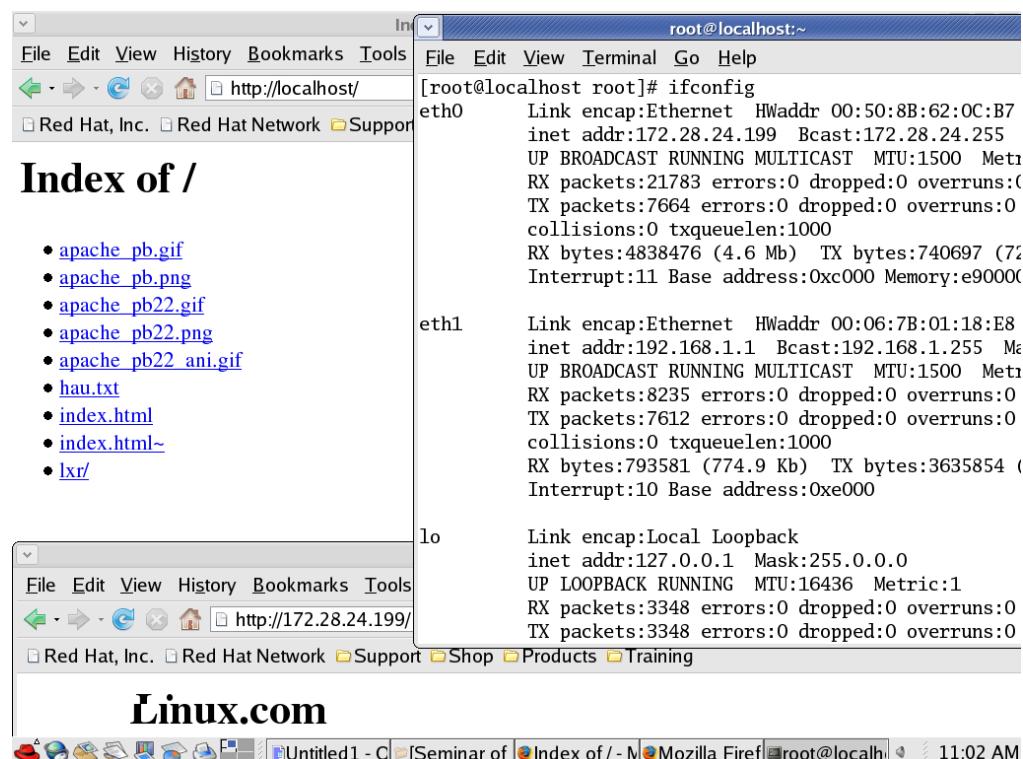
Tương tự như trên, trên máy 192.168.1.2 ta truy cập `ftp://localhost` và đăng nhập từ xa đến máy ở mạng khác và dùng Internet Explorer để truy cập `ftp://172.28.24.199`. Cả 2 điều này cùng có nghĩa là truy cập đến FTP Server trên máy 192.168.1.2 port 21. Do đó, ta nhận được hai kết quả giống nhau.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER



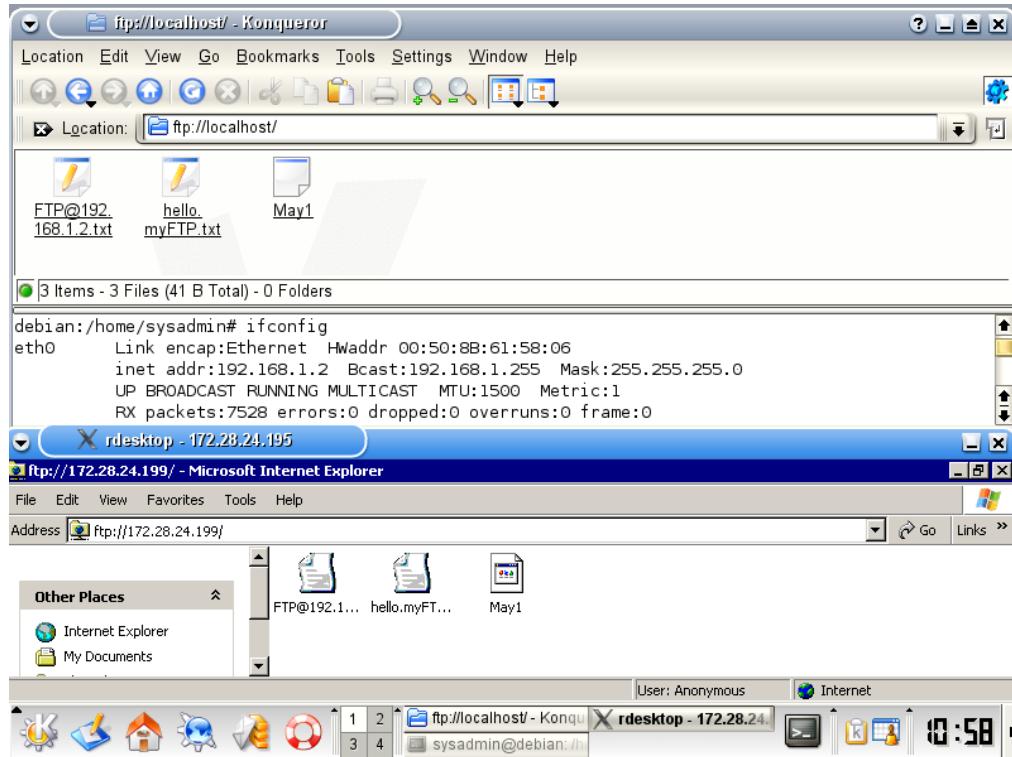
Hình 7.3: Kết quả truy cập Web Server trên 2 máy khác nhau



Hình 7.4: Kết quả truy cập Web Server trên máy dùng làm gateway

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER



Hình 7.5: Kết quả truy cập ftp đồng thời trên 2 máy

Kết quả trên máy dùng gateway, chúng tôi dùng chương trình tcpdump giám sát việc định tuyến qua gateway. Kết quả thu được như sau:

```
11:01:09.614831 172.28.24.199.1065 > 172.28.24.195.3389: . ack 309 win  
53576 <nop,nop,timestamp 589252 23626> (DF)  
11:01:09.908869 172.28.24.199.1026 > www.hcmut.edu.vn.domain: 59879+  
PTR? 164.24.28.172.in-addr.arpa. (44) (DF)  
11:01:09.909556 www.hcmut.edu.vn.domain > 172.28.24.199.1026: 59879  
NXDomain* 0/1/0 (112) (DF)  
11:01:09.925041 172.28.24.195.3389 > 172.28.24.199.1065: P 309:326(17) ack  
1 win 64376 <nop,nop,timestamp 23629 589252> (DF)  
11:01:09.925202 172.28.24.199.1065 > 172.28.24.195.3389: . ack 326 win  
53576 <nop,nop,timestamp 589283 23629> (DF)  
11:01:10.455809 172.28.24.195.3389 > 172.28.24.199.1065: P 326:342(16) ack  
1 win 64376 <nop,nop,timestamp 23634 589283> (DF)  
11:01:10.455995 172.28.24.199.1065 > 172.28.24.195.3389: . ack 342 win  
53576 <nop,nop,timestamp 589336 23634> (DF)  
11:01:10.555978 172.28.24.195.3389 > 172.28.24.199.1065: P 342:371(29) ack  
1 win 64376 <nop,nop,timestamp 23635 589336> (DF)  
11:01:10.556143 172.28.24.199.1065 > 172.28.24.195.3389: . ack 371 win  
53576 <nop,nop,timestamp 589346 23635> (DF)  
11:01:10.986546 172.28.24.195.3389 > 172.28.24.199.1065: P 371:388(17) ack  
1 win 64376 <nop,nop,timestamp 23640 589346> (DF)
```

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER

CHƯƠNG 7: ÁP DỤNG IPTABLES VÀO WEB SERVER VÀ FTP SERVER

```
11:01:10.986722 172.28.24.199.1065 > 172.28.24.195.3389: . ack 388 win  
53576 <nop,nop,timestamp 589389 23640> (DF)  
11:01:11.517327 172.28.24.195.3389 > 172.28.24.199.1065: P 388:404(16) ack  
1 win 64376 <nop,nop,timestamp 23646 589389> (DF)  
11:01:11.517490 172.28.24.199.1065 > 172.28.24.195.3389: . ack 404 win  
53576 <nop,nop,timestamp 589442 23646> (DF),nop,timestamp 23626  
589230> (DF)  
11:01:09.614831 172.28.24.199.1065 > 172.28.24.195.3389: . ack 309 win  
53576 <nop,nop,timestamp 589252 23626> (DF)  
11:01:09.908869 172.28.24.199.1026 > www.hcmut.edu.vn.domain: 59879+  
PTR? 164.24.28.172.in-addr.arpa. (44) (DF)  
11:01:09.909556 www.hcmut.edu.vn.domain > 172.28.24.199.1026: 59879  
NXDomain* 0/1/0 (112) (DF)  
11:01:09.925041 172.28.24.195.3389 > 172.28.24.199.1065: P 309:326(17) ack  
1 win 64376 <nop,nop,timestamp 23629 589252> (DF)  
11:01:09.925202 172.28.24.199.1065 > 172.28.24.195.3389: . ack 326 win  
53576 <nop,nop,timestamp 589283 23629> (DF)  
11:01:10.455809 172.28.24.195.3389 > 172.28.24.199.1065: P 326:342(16) ack  
1 win 64376 <nop,nop,timestamp 23634 589283> (DF)  
11:01:10.455995 172.28.24.199.1065 > 172.28.24.195.3389: . ack 342 win  
53576 <nop,nop,timestamp 589336 23634> (DF)  
11:01:10.555978 172.28.24.195.3389 > 172.28.24.199.1065: P 342:371(29) ack  
1 win 64376 <nop,nop,timestamp 23635 589336> (DF)  
11:01:10.556143 172.28.24.199.1065 > 172.28.24.195.3389: . ack 371 win  
53576 <nop,nop,timestamp 589346 23635> (DF)  
11:01:10.986546 172.28.24.195.3389 > 172.28.24.199.1065: P 371:388(17) ack  
1 win 64376 <nop,nop,timestamp 23640 589346> (DF)  
11:01:10.986722 172.28.24.199.1065 > 172.28.24.195.3389: . ack 388 win  
53576 <nop,nop,timestamp 589389 23640> (DF)  
11:01:11.517327 172.28.24.195.3389 > 172.28.24.199.1065: P 388:404(16) ack  
1 win 64376 <nop,nop,timestamp 23646 589389> (DF)  
11:01:11.517490 172.28.24.199.1065 > 172.28.24.195.3389: . ack 404 win  
53576 <nop,nop,timestamp 589442 23646> (DF)
```

Kết quả trên cho thấy ta hoàn toàn không thấy được những máy trên mạng 192.168.1.0/21.

Tóm lại, dùng iptables để cấu hình việc NAT từ trong ra ngoài để cho phép từ những máy trong mạng LAN có thể truy cập đến các Server bên ngoài. Và việc NAT từ ngoài vào trong là để cho phép các máy có thể ở ngoài mạng có thể truy cập đến những Server bên trong mạng LAN. Kết quả cho thấy, ta đã thực hiện được cân bằng tải server, tức là cùng địa chỉ IP nhưng khác port, ta có thể truy cập đến 2 server khác nhau. Thứ hai, là nếu với những cách truy cập khác nhau và ở những vị trí khác nhau thì máy tính cũng sẽ hiểu khác nhau. Và thứ ba là, từ kết quả của chương trình tcptrace cho thấy với iptables ta ngoài việc thực hiện lọc gói tin, nó còn

thực hiện được NAT và đồng thời vẫn đảm bảo tính bảo mật cho mạng bên trong. Tuy nhiên, việc cấu hình cân bằng tải server và bảo mật cho mạng người ta không làm trên phần mềm mà làm trực tiếp trên các phần cứng.

Chương 8: GIẢI PHÁP CẢI TIẾN FAULT-TOLERANT ROUTING, BROADCASTING VÀ LOAD BALANCING TRONG MẠNG HYPER-DEBRUIJN-ASTER

8.1. Giới thiệu về sự cải tiến mô hình HD*:

Trong bài viết này, đầu tiên chúng tôi trình bày một họ của mạng gọi là Hyper-DeBruijn-Aster network và sau đó là những vấn đề truyền thông được nghiên cứu. Các tính chất của mạng De-Bruijn trình bày rằng mạng này là điển hình tốt cho thế hệ công nghệ tương lai, sau hypercube [4]. Nó hỗ trợ nhiều ứng dụng quan trọng từ việc tạo ra mạng De-Bruijn. Điểm mạnh của Hypercube là khá lớn. Còn điểm mạnh trong k-cube là degree và diameter là không có mối liên hệ với nhau [4]. Mà Hypercube thừa hưởng từ k-cube nên tận dụng được những điểm mạnh đó. Và một điểm mạnh nữa mà chúng tôi tìm thấy đó là nếu kết hợp giữa De-Bruijn và Hypercube trong shortest routing thì rất là hiệu quả. Vấn đề này chúng tôi sẽ trình bày trong phần Hyper-DeBruijn-Aster graph.

Việc định tuyến trong De-Bruijn Graph được điều tra bởi Mao và Yang, tuy nhiên những giải thuật của họ không thể đạt được với shortest path nếu như có node bị hỏng trên đường đi [4]. Những vấn đề về broadcasting trong De-Bruijn graph cũng vừa được điều tra bởi Esfahanian, Ganesan, Ohring, tuy nhiên, những giải thuật của họ chỉ làm việc ở một mạng binary de Bruijn [4]. Và điều đó được tác giả Nguyễn Chí Ngọc trong bài viết [4], cải thiện khá tốt. Tuy nhiên, tác giả Ngọc vẫn chưa tối ưu trong vấn đề *fault tolerant routing* và *broadcasting*. Việc tối ưu đó, chúng tôi làm bằng cách kết hợp Hypercube và De-Bruijn graph tạo ra một graph gọi là Hyper-DeBruijn-Aster graph (nó có thể hiểu là Hypercube-DeBruijn-Asterisk và chúng tôi ký hiệu nó là *HD**). *HD** là sự kết hợp những ưu điểm của Hypercube, De-Bruijn với miền bit mở rộng thay vì là binary de Bruijn graph. Và *HD** còn có một ưu điểm khác đó là có thể thực hiện được *shortest routing*. Còn đối với bài viết [1] của Elango Ganesan, Dhiraj K Pradhan, các tác giả đã ứng dụng được Hyper-DeBruijn graph. Tuy nhiên, các tác giả chỉ dừng lại ở Binary de Bruijn graph và giải thuật DeadLock-Free routing. Với giải thuật của chúng tôi, đọc giả sẽ dễ hiểu hơn vì nó dựa trên kiến thức và nền tảng của hypercube, deBruijn và giải thuật *shorest rounting*. Đây là những mảng mà người đọc phần lớn đã biết. Còn trong bài viết [3] của Wei Shi và Pradip K Srimani, tác giả đã đưa ra một giải thuật định tuyến khá hay là kết hợp giữa Hypercube và Butterfly để tạo ra một graph mang tên là Hyper-Butterfly và dùng giải thuật định tuyến theo đường đi ngắn nhất. Trong bài viết tác giả cho là Hyper-DeBruijn graph là không có quy tắc, chưa tối ưu trong vấn đề *fault tolerant routing* và nó chưa tối ưu mà còn lại phức tạp. Tuy nhiên, những khuyết điểm đó chúng tôi sẽ khắc phục trong bài viết này. Mặt khác, với Hyper-DeBruijn-Aster và với giải thuật *Shortest Routing* mà chúng tôi sẽ trình

bày dưới đây, thì mạng Hyper-DeBruijn-Aster còn có thể làm nhiệm vụ *cân bằng tải* và *xử lý động bộ*, điều này đóng góp rất nhiều vào những giải thuật *xử lý song song* và *phân tán dữ liệu*. Do đó có thể nói rằng, nó cải tiến rất nhiều trong việc tăng hiệu suất mạng truyền thông đa xử lý.

8.2. Hyper-DeBruijn-Aster graph

8.2.1. Hypercube Graph H_n và De Bruijn Graph B_n

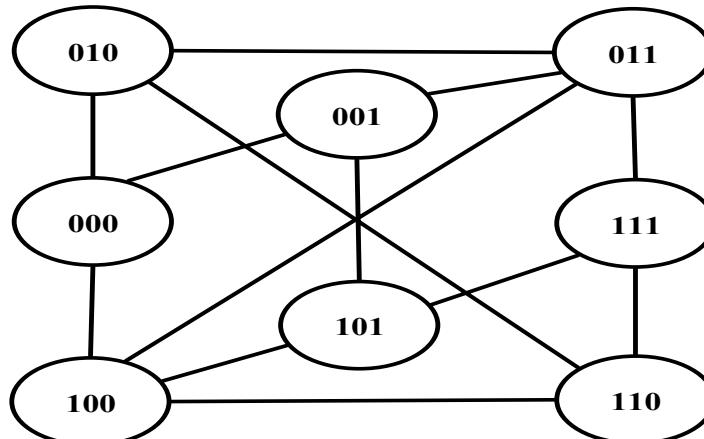
Hypercube Graph bậc m , $H(m)$ bao gồm tập hợp các node Z_2^m . Có một cạnh nằm giữa 2 đỉnh nếu và chỉ nếu các nhãn node của chúng khác nhau chính xác một bit.

Binary de Bruijn Graph bậc n , $B(n)$ bao gồm các tập đỉnh Z_2^n .

Cho $\alpha, \beta \in Z_2$ và $x \in Z_2^{n-2}$, mỗi node có dạng là $ax\beta$, được kết nối tới:

- $x\beta\alpha$ thông qua shuffle arc
- $x\beta\bar{\alpha}$ thông qua shuffle-exchange arc
- $\beta\alpha x$ thông qua inverse-shuffle arc
- $\bar{\beta}\alpha x$ thông qua inverse-shuffle-exchange arc

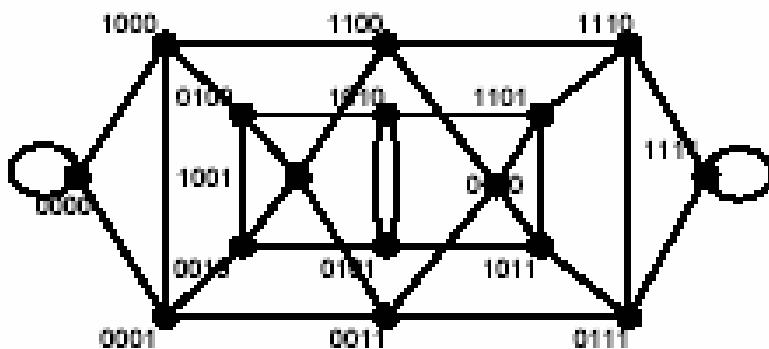
Tương tự vậy, nếu ta thay đổi cho $\alpha, \beta \in Z_d$ và $x \in Z_d^{n-2}$ (với $d \geq 2$), thì rõ ràng ta có thể mở rộng số liên kết trên mỗi node cũng như mở rộng số node lân cận. Và điều này có thể cải thiện vấn đề *Fault Tolerant Routing* và *Broadcasting* trong mạng De-Bruijn.



Hình 8.1: Hypercube graph $H(3)$

Nhận xét 1: $D(d,n)$ là một graph vô hướng có quy tắc đối xứng, có degree là $2d$, diameter là n và có số node là $N(D_{dn}) = d^n$ [4].

Nhận xét 2: $H(m)$ là một graph vô hướng có quy tắc đối xứng, có degree là m , diameter là m và có số node là $N(D_m) = 2^m$ [3].



Hình 8.2: De Bruijn graph $D(2,2)$

8.2.2. Hyper-DeBruijn-Aster Graph $HD^*(m,d,n)$

Với sự mở rộng DeBruijn graph ở trên kết hợp với mạng hypercube, chúng tôi đã tìm ra một họ mạng mới, đó là *Hyper-DeBruijn-Aster bậc (m,n)* và *hệ số mở rộng DeBruijn* là d , ký hiệu là $HD^*(m,d,n)$. $HD^*(m,d,n)$ là một graph tích $H(m)xD(d,n)$.

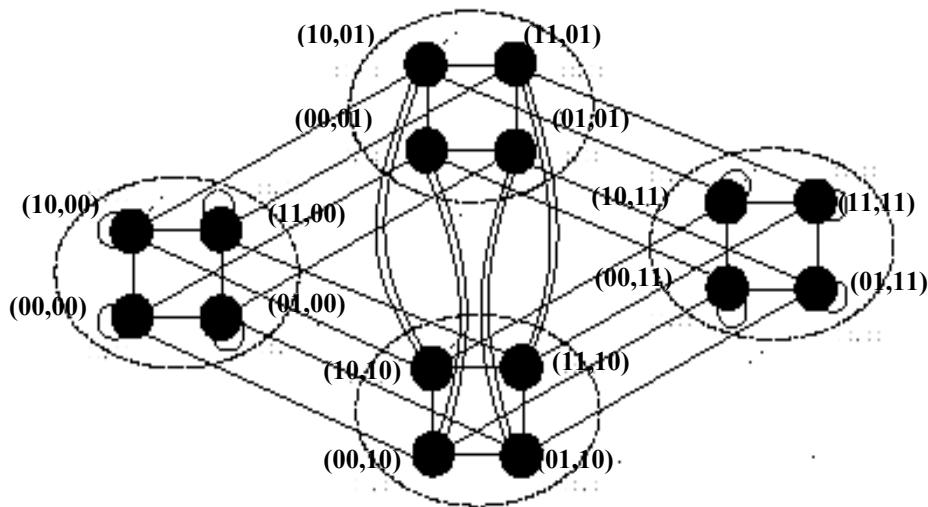
Ta dễ dàng thấy rằng một node $\langle x_{m-1}x_{m-2}\dots x_0y_{n-1}y_{n-2}\dots y_0 \rangle$ nối đến những node dưới đây đối với phần lân cận deBruijn:

$$\langle x_{m-1}x_{m-2}\dots x_0, \alpha y_{n-1}y_{n-2}\dots y_1 \rangle$$

$$\langle x_{m-1}x_{m-2}\dots x_0, y_{n-1}y_{n-2}\dots y_1\alpha \rangle, \text{ với } \alpha \in Z_d$$

Và những node dưới đây đối với phần lân cận Hypercube

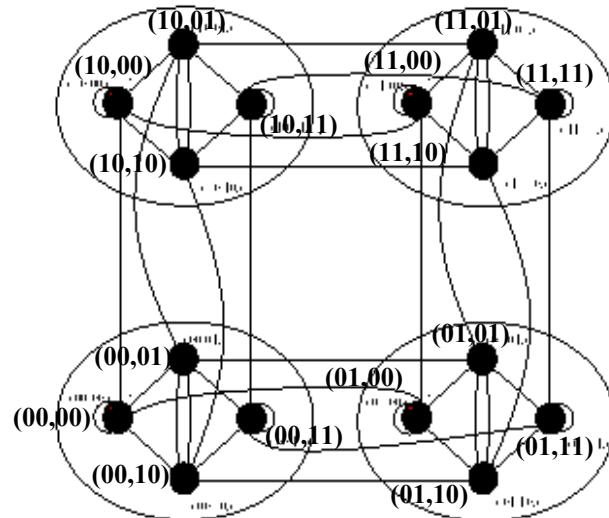
$$\langle x_{m-1}x_{m-2}\dots x_i x_i x_{i-1}\dots x_0, y_{n-1}y_{n-2}\dots y_0 \rangle \text{ với } 0 \leq i \leq m-1$$



Hình 8.3: Trình bày $HD^*(2,2,2)$

Nhận xét 3: $HD^*(m,d,n)$ là một graph vô hướng có quy tắc đối xứng, có degree là $m+2d$, diameter là $m + n$ và có số node là $N(HD^*) = 2^m x d^n$.

Nhận xét 4: Địa chỉ HD^* gồm 2 phần: phần đầu của địa chỉ HD^* , đó chính là phần địa chỉ của Hypercube graph còn phần sau của địa chỉ HD^* , đó chính là phần địa chỉ của DeBruijn graph.



Hình 8.4: Một dạng trình bày khác của $HD^*(2,2,2)$

Chứng minh 1: Vì Hypercube graph và De-Bruijn graph là 2 loại graph đối xứng (nhận xét 1 và 2). Một graph $HD^*(m,d,n)$, ta có thể phân ra 2 phần, đó là Hypercube graph và De-Bruijn graph. Và mỗi phần của Hypercube bao gồm một DeBruijn graph hay với cách phân khác thì mỗi phần của DeBruijn graph, nó bao gồm một Hypercube graph. Do đó, HD^* graph cũng là graph đối xứng (hình 8.3, 8.4).

Chứng minh 2: Từ phần chứng minh 1. Ta thấy nếu một graph $HD^*(m,d,n)$, thì với cách phân chia theo Hình 8.4, ta có m khối, đó chính là m hypercube graph. Trong mỗi khối đó có d^n node. Do đó, số node của $HD^*(m,d,n)$ là $2^m x d^n$ hay có thể hiểu một cách dàng là $N(HD^*) = N(H_m) x N(D_{nd})$.

Chứng minh 3: degree của HD^* là $m + 2d$.

Định nghĩa 1: degree $d_G(v)$ của một đỉnh v trong graph G là số cạnh của graph G hướng tới v , mỗi lần tính vòng tính là 2 cạnh [2] trang 10, như ở node 0000 và 1111 ở Hình 8.2.

Degree $d(H_m) = m$ (nhận xét 1) và $d(D(d,n)) = 2xd$ (nhận xét 2). Và trên mỗi node sẽ có những liên kết đến một số node của phần DeBruijn graph cũng như phần Hypercube vừa được trình bày ở trên.

Nên $d(HD^*) = d(H_m) + d(D(d,n))$ hay $d(HD^*) = m + 2d$.

Chứng minh 4: diameter của HD^* là $m + n$.

Định nghĩa 2: diameter của graph $G D_G$ là khoảng cách lớn nhất giữa 2 đỉnh của graph G [2] trang 14.

Với đặc tính của HD^* graph thì một node $v(h,d)$ đi đến node $v(h',d')$.
 Thì tiến trình định tuyến sẽ từ $v(h,d) \rightarrow v(h',d) \rightarrow v(h',d')$ hoặc từ $v(h,d) \rightarrow v(h,d') \rightarrow v(h',d')$.

Do đó, $D_G(HD^*) = D_G(Hm) + D_G(D(d,n))$ hay $D_G(HD^*) = m + n$.

Thông số	Node	Degree	Diameter	Fault-Tolerance
Hypercube $H(m+n)$	2^{m+n}	$m+n$	$m+n$	$m+n$
Butterfly $B(m+n)$	$(m+n) \times 2^{m+n}$	4	$3n/2$	4
De Bruijn $D(d,m+n)$	d^{m+n}	$2xd$	$m+n$	$2xd-2$
HyperButterfly $HB(m,n)$	$nx2^{m+n}$	$m+4$	$m+3n/2$	$m+4$
Hyper- DeBruijn-Aster $HD^*(m,d,n)$	$2^m \times d^n$	$m+2xd$	$m+n$	$m+2xd-2$

8.3. Shortest Routing và Diameter trong $HD^*(m,d,n)$

Việc định tuyến theo phương pháp gọi là *Shortest Routing* từ điểm đến điểm trong $HD^*(m,d,n)$ là cực kỳ đơn giản và dễ hiểu. Như ở phần *nhận xét 4* và phần *chứng minh 4* vừa thảo luận ở trên thì việc định tuyến từ 2 node bất kỳ $v(h,d)$ đến $v(h',d')$ dùng *Shortest Routing* có thể được thiết lập như sau:

- Đi từ $v(h,d) \rightarrow v(h',d)$ dùng giàn đồ *Shortest Routing* trong một hypercube graph.
- Đi từ $v(h',d) \rightarrow v(h',d')$ dùng giàn đồ *Shortest Routing* trong một De Bruijn graph.
- Hoặc ta có thể đảo ngược trình tự trên.

Nhận xét 5: Từ việc thảo luận trên ta có thể rút ra một công thức, đó là:

$$\text{ShortestRouting}(HD^*) = \text{ShortestRouting}(H) + \text{ShortestRouting}(D) (*)$$

Và như thế với việc đảo ngược trình tự trên thì vấn đề *Shortest Routing* trong mạng HD^* là không đổi.

Từ nhận xét 5, cho thấy vấn đề *Shortest Routing* trong mạng HD^* là rất hiệu quả. Và điều này làm cho HD^* graph trở thành ưu điểm hơn hẳn so sánh với các mô hình graph khác như hypercube, DeBruijn, Butterfly Tuy nhiên, với Hyper-DeBruijn-Aster graph thì HD^* đã tăng số cạnh trên một node và điều này

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER
CHƯƠNG 8: GIẢI PHÁP CẢI TIẾN FAULT-TOLERANT ROUTING, BROADCASTING
VÀ LOAD BALANCING TRONG MẠNG HYPER-DEBRUIJN-ASTER

cho phép ta có thể tăng số đường *Shortest Routing* trong mỗi graph con DeBruijn lên, thêm vào đó ta kết hợp giữa Hypercube và DeBruijn graph. Điều này cho phép ta tăng số đường *Shortest Routing* lên rất nhiều. Và với đặc điểm này ta so sánh lại với Hypercube, DeBruijn hay Butterfly thì đây là một lợi thế trong vấn đề *Shortest Routing*. Tuy nhiên, vấn đề *Shortest Routing* trong bài [3] của Wei Shi and Pradip K Srimani, các tác giả đề cập có phần giống với cách thức định tuyến của chúng tôi nhưng các tác giả chỉ thực hiện *Shortest Routing* trong mạng Hyper-Butterfly, không đưa ra quy luật cho mạng Hyper-DeBruijn và chỉ nói rằng mạng Hyper-DeBruijn là không có quy tắc và bằng so sánh của tác giả chỉ dừng lại ở mạng Hyper-DeBruijn trong đó phần DeBruijn chỉ dừng lại ở Binary DeBruijn graph. Chúng tôi đã cải tiến nó, làm cho nó trở nên hiệu quả hơn, đó là đưa ra Hyper-DeBruijn-Aster (kiến trúc mạng vừa được đề cập ở những phần trên) và chúng tôi đã áp dụng *Shortest Routing* trên mạng này một cách hiệu quả với việc tìm ra quy luật (*). Tiếp theo, chúng tôi sẽ trình bày một giải thuật định tuyến mà chúng nghiên cứu trong mạng Hyper-DeBruijn-Aster. Với giải thuật này, các node trong mạng sẽ làm việc nhẹ nhàng hơn, nhanh hơn và đặc biệt là hiệu quả trong vấn đề *Fault Tolerant Routing, Broadcasting* và *cân bằng tải*. Vì với cách trình bày trên cùng với các nhận xét, chúng minh và bằng so sánh trên ta thấy Hyper-DeBruijn-Aster graph mà chúng tôi đưa ra là hơn hẳn so với những graph khác như Hypercube, DeBruijn, Butterfly và HyperButterfly. Và trong bài viết [3] của Wei Shi and Pradip K Srimani, các tác giả đã so sánh graph của mình, đó là HyperButterfly graph có những ưu điểm hơn hẳn so với Hypercube, DeBruijn, Butterfly và Hyper-DeBruijn. Dưới đây, chúng tôi sẽ so sánh ưu điểm giữa Hyper-Butterfly graph và graph mà chúng tôi đề nghị.

Giả sử Hyper-DeBruijn-Aster là $HD^*(m_1, d_1, n_1)$, trong đó m_1 là bậc của hypercube, n_1 là bậc của DeBruijn, và d_1 là chỉ số mở rộng bit trên graph ($d_1 \geq 2$).

Và Hyper-Butterfly là $HB(m_2, n_2)$, trong đó m_2 là bậc của hypercube còn n_2 là bậc của Butterfly.

Và giả sử việc thiết kế mạng là như nhau tức cùng số node:

$$\text{Ta có, } 2^{m_1} d_1^{n_1} = n_2 2^{m_2 + n_2} \Leftrightarrow m_1 + n_1 \log_2 d_1 = m_2 + n_2 + \log_2 n_2 \quad (1)$$

Do đó, degree của HD^* : $d_{HD^*} = m_1 + 2x d_1 \geq m_1 + 4$ thì rõ ràng nếu d_1 càng lớn thì d_{HD^*} càng lớn và nếu như ta cho $m_1 = m_2 \rightarrow d_{HD^*} \geq d_{HB}$. Điều này làm mạng HD^* hiệu quả hơn HB trong vấn đề *Fault-Tolerant Routing* và *Broadcasting*.

Bên cạnh đó, Diameter của HD^* : $D_{HD^*} = m_1 + n_1 = m_2 + n_2 + \log_2 n_2 - n_1 \log_2 d_1 + n_1$.

Rõ ràng khi d_1 tăng lên thì D_{HD^*} giảm và nếu $d_1 \geq 2n_2$ thì $D_{HD^*} \leq D_{HB}$. Điều này cho ta thêm một minh chứng nữa là mạng HD^* hiệu quả hơn HB trong vấn đề *Shortest Routing*.

Và cuối cùng là $\text{Fault-Tolerance}(HD^*) = m_1 + 2d_1 - 2(2)$. Rõ ràng nếu d_1 tăng thì $\text{Fault-Tolerance}(HD^*)$ tăng lên. Trong khi, $\text{Fault-Tolerance}(HB)$ chỉ bằng $m_2 + 4$.

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER
CHƯƠNG 8: GIẢI PHÁP CẢI TIẾN FAULT-TOLERANT ROUTING, BROADCASTING
VÀ LOAD BALANCING TRONG MẠNG HYPER-DEBRUIJN-ASTER

Với những gì vừa trình bày trên, cho thấy Hyper-DeBruijn-Aster tối ưu hơn so với Hyper-Buttfly trong vấn đề *Fault-Tolerant Routing*, *Shortest Routing* và *Broadcasting*.

Sau đây, chúng tôi sẽ trình bày giải thuật *Shortest Routing* trong mạng Hyper-DeBruijn-Aster mà chúng tôi nghiên cứu.

Bước 1: Dùng giải thuật *Shortest Routing* trong mạng DeBruijn để tìm ra những đường đi ngắn nhất đi từ $d_S \rightarrow d_D$. Đây là bước tính lý thuyết dựa vào cơ sở hạ tầng của mạng mà không phụ thuộc và trạng thái của mạng lúc định tuyến.

Bước 2: Thực hiện định tuyến trong mạng HD^* , việc định tuyến của mạng ở bước này sẽ phụ thuộc vào tình trạng của mạng. Cách thức định tuyến sẽ tiến hành như sau:

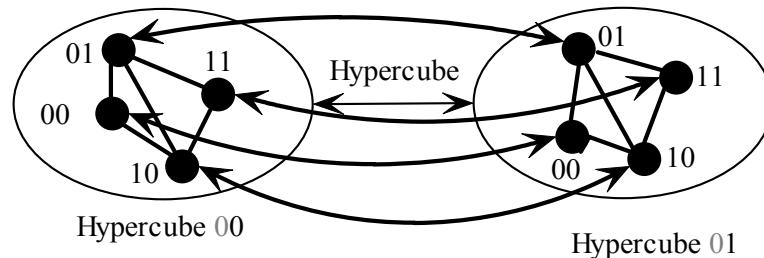
- Ban đầu, nó sẽ xét đến đường đi ngắn nhất thứ nhất (chọn bất kỳ một trong số đường đi ngắn nhất vừa tìm được ở bước 1). Và nó sẽ định tuyến đến node thứ nhất trên đường đi thứ nhất đó. Nó sẽ kiểm tra kết nối nếu thành công thì nó sẽ định tuyến đến đó. Nếu không thành công thì nó sẽ kiểm tra đến node thứ 1 của đường ngắn nhất thứ 2, và nếu vẫn không thành công thì nó sẽ kiểm tra đến node thứ 1 của đường tiếp theo. Cứ như thế mà đến đường cuối cùng. Còn nếu đến node thứ 1 của đường cuối cùng mà vẫn không thành công thì nó sẽ thực hiện việc định tuyến đến một hypercube thứ 1 (lân cận hypercube của nó). Trong đó, hypercube thứ nhất được xác định bằng cách đảo ngược 1 bit khác nhau giữa phần địa chỉ hypercube của địa chỉ nguồn và đích (sự so sánh khác biệt này có thể từ trái sang phải hoặc từ phải sang trái). Nếu không thành công với hypercube lân cận thứ 1 thì tiếp tục xét đến hypercube lân cận thứ 2. Nếu đến hypercube lân cận cuối cùng mà không thành công thì chúng ta sẽ quay lại bước 1 với giả sử là các node lân cận của địa chỉ nguồn vừa xét ở trên là không tham gia vào việc tìm ở bước 1. Sau đó, ta sẽ tiếp tụ lại từ đầu bước 2 và ở bước này việc tìm nếu xay tra lỗi đến tất cả các node lân cận của phần DeBruijn mà gặp lỗi thì nó bỏ qua việc thử đến các hypercube lân cận và tiếp tục lại bước 1 như trên. Vì tính chất của HD^* là để đi từ node $v(h,d) \rightarrow v(h',d')$ thì nó sẽ lần lượt $v(h,d) \rightarrow v(h,d') \rightarrow v(h',d')$ nên việc thực hiện giải thuật *Shortest Routing* như trên là tối ưu.
- Khi nó đã định tuyến được đến node tiếp theo thì ta tiếp tục định tuyến đến node tiếp theo với giải thuật tương tự như trên. Và bây giờ địa chỉ nguồn cho việc định tuyến tiếp theo, nó là địa chỉ của node tiếp theo này. Trường hợp, việc định tuyến từ node này đến tất cả các node lân cận (trừ node vừa định tuyến trước đó) bị lỗi thì nó gửi lại thông báo cho node định tuyến trước đó biết. Để node ngay trước đó, nó định tuyến lại đến node khác. Hoặc thời gian của node ngay trước đó bị timeout thì node ngay trước đó cũng sẽ định tuyến lại với hypercube lân cận khác.

Nhận xét 6: Với Hyper-DeBruijn với việc mở rộng bit, cho phép ta tăng số đường đi ngắn nhất giữa 2 node trong mạng Hyper-DeBruijn. Hơn nữa, mạng Hypercube cũng cho ta nhiều đường đi ngắn nhất giữa 2 điểm. Kết hợp với công thức (*), cho ta một graph Hyper-DeBruijn-Aster với số đường đi ngắn nhất tăng lên rất nhiều. Nếu giả sử đường đi ngắn nhất từ 2 điểm trong phần DeBruijn mà không có sự tham dự của Hypercube đi từ $d_S \rightarrow d_D$, nó đi qua là N node (kể cả node nguồn và node đích). Và giả sử số bit khác nhau giữa địa chỉ nguồn và đích của phần là M thì số đường đi ngắn nhất giữa 2 điểm bất kỳ trong mạng Hyper-DeBruijn-Aster là $N \times M!$ đường.

Nhận xét 7: Ưu điểm của giải thuật trên là chỉ cần tính đường ngắn nhất của phần De-Bruijn graph mà không cần phải tính đường đi ngắn nhất của toàn mạng. Điều này cho phép ta tăng tốc độ xử lý và giảm gánh nặng cho mạng.

Nhận xét 8: Việc định tuyến trong phần Hypercube, ta chỉ cần xét số bit khác nhau giữa địa chỉ nguồn và địa chỉ đích, đó chính là số node hypercube lân cận có thể gói tối đa trong giải thuật *Shortest Routing*. Do đó, có thể tính được số đường đi ngắn nhất của Hypercube một cách dễ dàng bằng số bit khác nhau giữa địa chỉ nguồn và địa chỉ đích lấy gai thừa. Đây là đặc điểm giúp ta tìm ra đường đi ngắn nhất trong hypercube một cách dễ dàng và nhanh chóng mà không cần phải tính toán.

Nhận xét 9: Ta thấy rằng mạng Hyper-DeBruijn-Aster có đặc tính ánh xạ theo cạnh của hypercube, được minh họa ở Hình 8.5. Do đó, nếu việc định tuyến từ một node đến một node kế tiếp trong cùng một Hypercube bị lỗi thì nó có thể chuyển sang một Hypercube kế tiếp. Tuy nhiên, ta vẫn giữ được phần địa chỉ De-Bruijn, tức là có thể định tuyến theo hướng ban đầu, thay vì trên Hypercube ban đầu thì nó là một Hypercube khác.



Hình 8.5: Từng phần tử của Hypercube 00 sẽ ánh xạ sang Hypercube 01

Ví dụ: để định tuyến từ node $(00,01) \rightarrow (01,11)$. Thì ban đầu nó dùng giải thuật định tuyến *Shortest Routing De-Bruijn* trên cùng một Hypercube. Tức sẽ đi từ $(00,01) \rightarrow (00,11) \rightarrow (01,11)$. Nếu node $(00,11)$ bị lỗi thì nó sẽ thực hiện đi từ $(00,01) \rightarrow (01,01)$. Ta thấy địa chỉ của phần De-Bruijn vẫn không đổi. Và tiếp tục định tuyến theo hướng từ $01 \rightarrow 11$ của phần De-Bruijn, tức là đi từ $(01,01) \rightarrow (01,11)$. Với cách định tuyến như trên, mạng cho phép node định tuyến tiếp theo bị lỗi mà vẫn đảm bảo vẫn đè *Shortest Routing*. Và như giải thuật vừa trình bày trên

và ví dụ minh chứng vừa trình bày thì nếu các node lân cận của node tiếp theo bị lỗi hết thì node tiếp theo đó sẽ gửi một thông điệp báo lỗi không thể định tuyến được về lại node trên nó. Để node trên nó sẽ đi theo đường đi ngắn nhất thứ 2 hoặc là gửi sang Hypercube lân cận. Và với giải thuật như vậy, ta vẫn đảm bảo vấn đề *Shortest Routing*.

Nhận xét 10: Việc định tuyến theo giải thuật Shortest Routing như trên, ta thấy rằng có sự tỷ lệ giữa độ dài đường đi và số đường đi ngắn nhất. Và điều này phụ thuộc vào số bit khác nhau trong phần hypercube giữa địa chỉ nguồn và đích. Nếu số bit khác nhau tăng thì đường đi sẽ dài hơn và như thế số đường đi ngắn nhất sẽ tăng lên. Và điều này phù hợp thực tế, vì đường đi càng dài thì xác suất lỗi cũng tăng lên và ngược lại.

Với giải thuật trên, trong trường hợp không lỗi và giả sử số node trên đường đi ngắn nhất của phần De-Brujin là N_D và số bit khác nhau giữa địa chỉ nguồn và đích là M_H thì số node mà nó định tuyến qua sẽ bằng $N_D + M_H \leq D_{HD^*}$, D_{HD^*} là Diameter của HD^* .

8.4. Fault Tolerance của $HD^*(m,d,n)$

Định nghĩa 3: *Fault Tolerance của một mạng là khả năng của mạng có thể tiếp tục vận hành đúng đắn ngay cả khi một hay nhiều node bị lỗi.*

Việc áp dụng giải pháp *Shortest Routing* vào một graph nào đó thì cơ bản ta sẽ tìm ra một đường ngắn nhất và cho định tuyến theo đường đó. Điều này rất tốt đối với việc tăng tốc định tuyến hay giảm thiểu chi phí hay số node đi qua ... nhưng nó sẽ không thể định tuyến thành công nếu một node nào đó trên đường đi bị lỗi. Tuy nhiên, với graph HD^* , nó cho phép chúng ta tăng số đường đi ngắn nhất lên. Điều này, cho phép ta thừa hưởng được ưu điểm của việc *Shortest Routing* và đồng thời cải tiến được vấn đề *Fault Tolerance*. Hơn nữa, việc áp dụng giải thuật *Shortest Routing* mà chúng tôi nghiên cứu vào HD^* , điều này làm cải tiến được vấn đề *Shortest Routing*, tức nếu vấn đề lỗi tại một node lân cận nào đó thì nó sẽ giải quyết ngay tại node đó hay node trên nó mà không cần phải yêu cầu gửi lại từ địa chỉ nguồn nguyên thủy.

Trong một mạng bất kỳ và yêu cầu là một giải thuật định tuyến tốt nhất thì khi định tuyến từ 1 điểm \rightarrow 1 điểm thì điều kiện tối thiểu là một trong những node lân cận của địa chỉ nguồn và một trong những node lân cận của địa chỉ đích và các Hypercube trung gian phải hoàn toàn tốt. Ngược lại, mạng không thể hoạt động được. Vài giải thuật *Shortest Routing* chỉ không thực hiện thành công trong HD^* chỉ xẩy ra trong trường hợp như trên. Điều này cho thấy, giải thuật *Shortest Routing* của chúng tôi trong HD^* là tối ưu trong vấn đề *Fault Tolerance*.

Giả sử, node bất kỳ trong mạng có địa chỉ $<x_0x_1x_2..x_i...x_{n-1}>$, $0 \leq i \leq n-1$

Với mạng $HD^*(m,d,n)$ thì số node lân cận của một node bất kỳ là N_{HD^*} bằng:

- $N_{HD^*} = m + 2d \leftrightarrow \exists x_i \mid x_i \neq x_{i+2}, 0 \leq i \leq n - 3$ (3)
- $N_{HD^*} = m + 2d - 1 \leftrightarrow \left\{ \begin{array}{l} \forall x_i \mid x_i \neq x_{i+2}, 0 \leq i \leq n - 3 \\ x_0 \neq x_1 \end{array} \right\}$ (4)
- $N_{HD^*} = m + 2d - 2 \leftrightarrow \forall x_i \mid x_i = x_{i+1}, 0 \leq i \leq n - 2$ (5)

Do đó, *Fault Tolerance* của mạng HD^* khi áp dụng giả thuật *Shortest Routing* bằng $N_{HD^*} - 1$. Hay nói cách khác, khả năng chịu lỗi trên đường định tuyến giữa 2 node bất kỳ sử dụng giải thuật *Shortest Routing* nó phụ thuộc vào vị trí node bị lỗi. Do đó, độ phức tạp của giải thuật cũng phụ thuộc vào vị trí lỗi của node trong mạng.

8.5. Broadcasting và cân bằng tải trong $HD^*(m, d, n)$

Việc mở rộng bit trong HD^* , cho phép ta tăng degree và giảm Diameter (chứng minh ở mục 2) cũng như tăng số node lân cận và số cạnh lân cận. Điều này làm cho HD^* tăng khả năng broadcasting.

Cùng với việc tăng *Fault Tolerance*. Như vậy, việc định tuyến node → node sẽ thành công cao hơn. Như vậy khả năng broadcasting đến tất cả các node lân cận trong mạng sẽ được bảo đảm hơn.

Do đặc tính của HD^* là cho nhiều đường đi ngắn nhất. Điều này làm cho mạng có vừa có khả năng định tuyến đường đi ngắn nhất vừa cân bằng tải vừa có khả năng đồng bộ hóa trong những vấn đề xử lý song song và phân tán dữ liệu. Và có thể nói, nó cải tiến rất nhiều trong việc tăng hiệu suất truyền thông đa xử lý.

8.6. Kết luận

Chúng tôi vừa đề nghị một cách tiếp cận mới trong vấn đề định tuyến giữa 2 node trong mạng. Đó là đưa ra một mô hình graph mới là Hyper-DeBruijn-Aster graph. Và đồng thời chúng tôi cũng đề nghị một giải thuật định tuyến *Shortest Routing cải tiến* áp dụng trên mạng HD^* . Với cách tiếp cận trên, chúng tôi đã cải thiện rất nhiều trong vấn đề *Shortest Routing*, *Fault Tolerance* và *Broadcasting*. Bên cạnh những ưu điểm đó, HD^* còn có thể góp phần không nhỏ trong vấn đề *cân bằng tải*, *xử lý đồng bộ* trong những *giải thuật xử lý song song* và *phân tán dữ liệu* trong mạng truyền thông đa xử lý.

Chương 9: MÔ HÌNH ĐỀ NGHỊ GLOBAL DYNAMIC LOAD BALANCING KẾT HỢP VỚI SHORTEST ROUTING TRONG HYPER-DEBRUIJN-ASTER MULTIPROCESSORS NETWORK

Các chương trước đã trình bày những kỹ thuật giúp cân bằng tải trong mạng như: phân luồng trong mạng dùng kỹ thuật NAT Routing, hay sử dụng chức năng firewall thông qua bảng IP Table.v.v..

Mô hình Server Load Balancing trong thực tế là hệ thống bao gồm nhiều Server, và để giúp cho các Server hoạt động hiệu quả, giảm thiểu thời gian xử lý của hệ thống, tránh nghẽn trong mạng, chúng ta cần phải bố trí các Server theo một trật tự có tính mô hình, ngoài ra chúng ta cần có một giải pháp cân bằng tải trên các Server sao cho tận dụng tối đa khả năng xử lý của mỗi Server. Đối với các ứng dụng thực tế, các Server chính là các processor (ví dụ như các chip vi mạch sử dụng công nghệ Nano tích hợp nhiều processor), và các processor này được sắp xếp theo mô hình Multiprocessors Network, được liên kết với nhau theo dạng Interconnection.

Trong chương này, chúng tôi giới thiệu một mô hình tối ưu Multiprocessors Network được thiết kế dựa vào các đặc trưng của mô hình Hyper-De Bruijn- Aster Graph trong chương trước. Song song đó, chúng tôi đề nghị một giải thuật cân bằng tải Global Dynamic Load Balancing, giải thuật này tỏ ra rất hiệu quả và phù hợp với Hyper-De Bruijn-Aster Multiprocessors Network

9.1. Giới thiệu

Multiprocessors Network (MNK) là mô hình mạng đang rất phổ biến cho các ứng dụng về lưu trữ, xử lý và truyền tải dữ liệu. Việc sắp xếp, lập trình.v.v.. các processors theo mô hình mạng đã được nhiều nhà nghiên cứu tìm hiểu, phát triển hết sức hiệu quả, đại diện như [5], [6], [7] , nhưng chúng tôi cảm thấy trong các mô hình trên thì mô hình graph Hyper Debruijn Aster(HD*) ở chương trước phù hợp với các yêu cầu “ Cân bằng tải động ” cho MNK.

HD* tỏ ra hữu hiệu trong vấn đề Fault Tolerant và Broadcasting trong MNK, và một trong những điều quan trọng trong chương này quan tâm chính là các qui tắc “cứng”(như là routing rules, firing rules) về Shortest Routing trong mạng HD*, cũng như dựa vào HD*, chúng ta có thể tính toán được số node trung gian mà một node A muốn đi qua node B.

Với mạng MNK thì vấn đề “ cân bằng tải “ trên mạng là hết sức cần thiết. Việc cân bằng tải đòi hỏi phải chỉ định công việc cho mỗi processors (hay là các nodes trong mô hình graph) phải thật phù hợp với khả năng xử lý của processor, điều này giúp làm giảm tối thiểu thời gian (hay chi phí) xử lý của chương trình. Vấn đề cân bằng tải “ tinh” là rất tốt cho môi trường mạng lý tưởng (các processors có tốc độ xử lý cao và giống nhau trong suốt quá trình xử lý, mạng không có sự cố.v.v..) và môi trường mạng thông tin, giao tiếp theo phương pháp định tuyến cố định (hay định tuyến được lập trình trước), nhưng thực tế trong MNK, luôn tồn tại các điều kiện không “ lý tưởng”, như là tốc độ xử lý, khả năng “ buffer” của các processors rất khác nhau tại thời điểm runtime, thậm chí là tại thời điểm compile time, việc định tuyến trong mạng vì thế luôn phải thay đổi, cập nhật để phù hợp với điều kiện của mạng. Để giải quyết vấn đề tính toán, lập trình và định tuyến MNK trong điều kiện luôn thay đổi tình trạng của MNK, cân bằng tải động (Dynamic Load Balancing, DLB) là vấn đề xử lý chủ yếu. Vấn đề DLB đã được các tác giả [12], [13] xử lý khá hiệu quả, nhưng do luôn phải cập nhật việc định tuyến trong mạng tai các thời điểm run time tốn nhiều thời gian và chi phí trong việc tính toán so với cân bằng tải tĩnh (việc định tuyến đã được lập trình trước). Như vậy nếu kết hợp định tuyến tĩnh và cân bằng động trong tính toán phân phối tải cho các node sẽ giảm được chi phí và thời gian rất nhiều, nhưng sự kết hợp này có thể thực hiện được không?

Dựa vào các qui tắc định tuyến “cứng” rất hiệu quả trong HD*, trong chương này chúng tôi sẽ kết hợp việc định tuyến tĩnh và cân bằng động trong tính toán trong mô hình MNK, hầu như việc kết hợp này chưa được tìm hiểu và chưa có sự nghiên cứu rõ ràng.

Phần còn lại của chương này sẽ được sắp xếp theo nội dung chính như sau: Đầu tiên chúng tôi sẽ giới thiệu các nghiên cứu của các tác giả khác về vấn đề định tuyến trong mô hình HD*, cũng như một số nghiên cứu về Dynamic Load Balancing (mục 9.2). Mục 9.3 chúng tôi sẽ giới thiệu các nghiên cứu Dynamic Load Balancing của một số tác giả. Mô hình Dynamic Load Balancing kết hợp với Shortest Routing trong HD* của chúng tôi sẽ được giới thiệu một cách cụ thể trong mục 9.4. Mục 9.5 là đánh giá và kết luận .

9.2. Cơ sở giải quyết vấn đề

Trong chương này chúng tôi đề cập đến giải thuật Shortest Routing trong mô hình Hyper Debruijn Aster Network và vấn đề cơ sở của Dynamic Load Balancing.

9.2.1. Hyper Debruijn Aster Graph (HD*):

9.2.1.1. Định tuyến trong Binary DeBruijn Network (BDB) :

Đối với mạng BDB, việc định tuyến cũng tương đối đơn giản, và giải thuật Shortest Routing trong mạng De Bruijn đã được [6] khai thác rất kỹ, ở đây chúng tôi xin giới thiệu sơ lược một kiểu định tuyến đơn giản trong mạng De Bruijn.

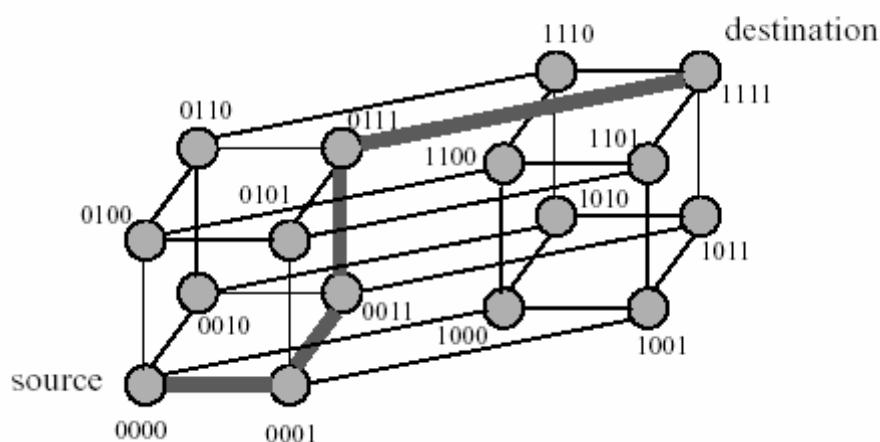
Cho $V = v_m v_{m-1} \dots v_1$ là node nguồn, cho $W = w_m w_{m-1} \dots w_1$ là node đích, cách thức định tuyến sẽ là $v_m v_{m-1} \dots v_1 \rightarrow v_{m-1} \dots v_1 w_m \rightarrow v_{m-2} \dots v_1 w_m w_{m-1} \rightarrow \dots \rightarrow w_m w_{m-1} \dots w_1$.

Ví dụ như node $V= 1001$ muốn đi đến node $V= 1111$ thì tiến trình định tuyến sẽ như sau: $1001 \rightarrow 0011 \rightarrow 0111 \rightarrow 1111$, tiến trình định tuyến này gọi là định tuyến dịch bit trái, trong đó quá trình dịch sẽ bắt đầu với bit w_m . Và dĩ nhiên là chúng ta cũng có tương tự cách định tuyến theo phương pháp dịch bit phải.

Điều chúng tôi muốn đề cập ở đây đó là nếu địa chỉ nguồn và đích xác định, dựa vào những cách thức định tuyến và kỹ thuật Shortest Routing trong mạng De Bruijn mà chúng ta đã biết, chúng ta có thể hoàn toàn nắm rõ đường đi của gói dữ liệu và số node mà gói đó đi qua, đây chính là kiểu định tuyến có hướng trong mạng, đồng nghĩa với việc ta hoàn toàn có thể lập trình trước quá trình định tuyến trong mạng De Bruijn.

9.2.1.2. Định tuyến trong mạng Binary Hypercube (BH)

Trong mạng Binary Hypercube, việc định tuyến hoàn toàn có qui tắc, trong đó hai nút kế cận nhau luôn khác nhau một bit.



Việc định tuyến trong mạng BH đi từ một node đến một node có nhiều cách, trường hợp định tuyến ví dụ khi node (0000) muốn đến node (1111) như sau:

- (0000) → (0001) → (0011) → (0111) → (1111)

Do tính chất hai nút kế cận nhau khác nhau một bit, cho nên nếu node nguồn và node đích khác nhau M bit thì số node trung gian là $(M-1)$ node (luôn đúng trong các cách định tuyến khác nhau khi đi từ node nguồn đến node đích trong BH), trong trường hợp ví dụ trên thì node (0000) khác node (1111) là 4 bit cho nên số node mà (0000) đi qua để đến được (1111) là $(4-1) = 3$ node trung gian.

Với nhận định trên trong mạng BH, việc tính toán số node mà gói dữ liệu đi từ node nguồn (có địa chỉ xác định) và cách thức định tuyến gói dữ liệu đó đến node đích (có địa chỉ xác định) là hoàn toàn có thể lập trình trước được, đây cũng chính là khả năng định tuyến có hướng của mạng Hypercube.

Khả năng định tuyến có hướng của mạng Hyper De Bruijn Aster cũng được giới thiệu trong phần tiếp theo.

9.2.1.3. Shortest Routing và Diameter trong $HD^*(m,d,n)$

Việc định tuyến theo phương pháp gọi là *Shortest Routing* từ điểm đến điểm trong $HD^*(m,d,n)$ là cực kỳ đơn giản. Như ở phần nhận xét ở chương trước việc định tuyến từ 2 node bất kỳ $v(h,d)$ đến $v(h',d')$ dùng Shortest Routing có thể được thiết lập như sau:

+ Đi từ $v(h,d) \rightarrow v(h',d)$ dùng giản đồ Shortest Routing trong một hypercube graph.

+ Đi từ $v(h',d) \rightarrow v(h',d')$ dùng giản đồ Shortest Routing trong một De Bruijn graph.

Hoặc bạn có thể đảo ngược trình tự trên.

Nhận xét: Từ việc thảo luận trên ta có thể rút ra một công thức, đó là:

$$\text{ShortestRouting}(HD^*) = \text{ShortestRouting}(H) + \text{ShortestRouting}(D)$$

Và như thế với việc đảo ngược trình tự trên thì vẫn đề Shortest Routing trong mạng HD^* là không đổi.

9.2.1.4. Shortest Routing và Diameter trong Binary Hyper Debruijn Aster BHD* ($m,2,n$)

Với các tính chất của HD* vừa nêu ở trên, bây giờ chúng tôi xét đến Shortest Routing trên BHD* ($m,2,n$).

Số node trên BHD* ($m,2,n$) là: $2^m \times 2^n$.

Dựa vào kỹ thuật định tuyến theo Shortest Routing, chúng tôi đề nghị phương pháp định tuyến “cứng” như sau:

$$\text{ShortestRouting(BHD*)} = \text{ShortestRouting(BH)} + \text{ShortestRouting(BD)}$$

Điều này có nghĩa là khi một node nguồn A (H_n, D_n) muốn định tuyến đến một node đích B (H_d, D_d) trong mô hình BHD* thì phải theo các bước:

Bước 1: Chỉ xét Shortest Routing trong phần địa chỉ Hypercube mà không xét đến ảnh hưởng của De Bruijn: $H_n \rightarrow H_d$. Theo mục 9.2.1.A ở trên, ở tại bước này, việc định tuyến phải đi qua $(M-1)$ node.

Bước 2: Chỉ xét Shortest Routing trong phần địa chỉ De Bruijn mà không xét đến ảnh hưởng của Hypercube: $D_n \rightarrow D_d$. Theo mục 9.2.1.B, ở tại bước này, việc định tuyến phải đi qua giả sử là N node.

Bước 3: Chỉ xét việc định tuyến từ: $D_d \rightarrow H_n$, số node đi qua là 1 node.

Do tính độc lập về định tuyến trong mạng BHD* cho nên tổng số node trung gian mà node A phải đi qua để đến được node B là:

$$(M-1+N+1)=M+N=K \text{ node.}$$

Do M node là xác định về số lượng lẫn vị trí các node thành phần, điều đó cũng tương tự đối với N node cũng xác định. Suy ra, tổng K node trong BHD* cũng hoàn toàn xác định về số lượng lẫn vị trí các node thành phần. Các nhận xét trên cũng hoàn toàn đúng với mô hình HD*(m,d,n) với hệ số (d) có thể mở rộng, do chương trước đã chứng minh tính chất Shortest Routing trên mạng HD* đúng với mọi trường hợp có hệ số (d) bất kỳ. Với các nhận định vừa trình bày thì việc định tuyến với điều kiện node nguồn và node đích xác định trong HD* là hoàn toàn biết trước, điều này giúp cho việc lập trình định tuyến tĩnh xác định (Oriented Routing) đối với Global Dynamic Load Balancing trở nên đơn giản đồng nghĩa với việc tính toán chi phí cho quá trình định tuyến sẽ dễ dàng hơn và giúp giảm thiểu bộ nhớ hay chi phí của hệ thống so với việc định tuyến được xác lập chỉ tại thời điểm mà mạng cập nhật tình trạng hoạt động của mạng.

9.3. Vấn đề Dynamic Load Balancing

Có 4 khuynh hướng lựa chọn Dynamic Load Balancing theo 2 kiểu .Các kĩ thuật Dynamic Load Balancing theo khuynh hướng *Global* hoặc *Local* (việc lựa chọn tùy thuộc vào thông tin mà chúng ta muốn ra quyết định cân bằng tải), và hay theo khuynh hướng *Centralized* hay *Distributed* (việc chọn lựa phụ thuộc vào việc ta đặt Master processor tại trung tâm hay đặt tại mỗi processor). Trong chương này chúng tôi tập trung nghiên cứu về Global Dynamic Load Balancing(GDLB).

Kiểu khuynh hướng Global Load Balancing: Quyết định cân bằng tải dựa vào thông tin chung của tất cả các processor trong hệ thống, trong kiểu cân bằng này lại chia nhỏ thành hai khuynh hướng khác nhau.

- + **Global Centralized DLB (GCDLB):** Khuynh hướng này thể hiện qua việc bộ cân bằng tải (Load Balancer) chính là Master processor trung tâm, thông tin của các processor tại thời điểm cập nhật sẽ được gửi đến Load Balancer. Sau khi tính toán việc phân phối tải lần kế tiếp, và tính khả thi của việc gửi đi công việc cho processor, bộ cân bằng tải sẽ gửi các hướng dẫn cho về khối lượng công việc phải chuyển đi, và địa chỉ đích nhận công việc cho mỗi processor. Các processor đích chỉ cần đợi cho đến khi nhận được khối lượng công việc mà processor đó cần phải xử lý.
- + **Global Distributed DLB (GDDLB):** Kiểu cân bằng này được đặc trưng bởi các bộ cân bằng tải giống nhau được đặt tại tất cả các processor. Không giống như GCDLB, trong hệ thống GDDLB, processor sẽ gửi tất cả các thông tin của processor đó tại thời điểm cập nhật tình trạng mạng cho tất cả các processor còn lại. Khi đó, processor nhận chỉ cần chờ đợi công việc đến, trong khi processor phát chỉ cần truyền đi dữ liệu.

Kiểu khuynh hướng Local Load Balancing: Trong mô hình cân bằng mạng Local, các processors được chia thành nhiều nhóm, mỗi nhóm có kích cỡ là K. Trong mỗi nhóm processors, các procesors có tính chất vật lý gần giống nhau (như tốc độ xử lý, bộ nhớ.v.v), trong kiểu cân bằng này bao gồm 2 khuynh hướng cân bằng khác.

- + **Local Centralized DLB (LCDLB) :** Mô hình này chỉ có 1 bộ Load Balancer trung tâm, ban đầu Load Balancer nhận được thông tin của các processors trong 1 nhóm, sau đó sẽ gửi các hướng dẫn về việc phân phối lại tải chỉ dành cho nhóm ấy, việc này sẽ được thực hiện trước khi xử lý các nhóm khác.

- + **Local Distributed DLB (LDDLB):** Các bộ Load Balancer giống nhau được đặt đồng thời tại các processor, nhưng thông tin của các processor chỉ được phổ biến cho các processor thành viên trong cùng 1 nhóm.

9.4. So sánh mô hình Global và Local, Distributed và Centralized Load Balancing

Global và Local: Điểm mạnh của mô hình Global là việc phân phối lại tải rất tối ưu. Hơn nữa, nếu ta xét các processor tại cùng một thời điểm thì tốc độ hội tụ của cân bằng tải trong mô hình Global nhanh hơn so với mô hình Local. Tuy nhiên trong Global, tương lai của việc phân phối thì chưa dự đoán trước được nên sẽ không tối ưu trong toàn quá trình thực hiện, ngoài ra chi phí cho thông tin rộng khắp là khá tốn kém. Và ngược lại, mô hình Local thì không tối ưu trong việc phân phối lại tải, tốc độ hội tụ chậm. Tuy nhiên, chi phí cho việc thông tin trong mạng ít tốn kém hơn. Ngoài ra, các group trong mô hình Local Load Balancing có thông số chất lượng và khả năng chịu tải không giống nhau, về tổng quan thì mạng Local cân bằng tái không hiệu quả, ví dụ như, nếu 1 group chứa các processors có thông số chất lượng thấp (phải chịu tải nhiều), do chỉ cân bằng tải trong group đó nên tổng quát group đó vẫn phải chịu tải nhiều, còn group chứa các processor có thông số chất lượng cao (chịu tải ít hay không phải chịu tải) sẽ xử lý công việc nhanh và thường xuyên trong trạng thái rảnh.

Centralized và Distributed: Trong mô hình Centralized, bộ cân bằng trung tâm phải đảm nhiệm việc cân xứng và tinh chỉnh lượng lớn các processor, điều này dẫn đến việc phải cần dùng nhiều tài nguyên và tốn chi phí cho bộ cân bằng trung tâm, nếu bộ cân bằng trung tâm bị hỏng thì toàn hệ thống sẽ gặp sự cố. Vấn đề đó sẽ được mô hình Distributed giải quyết hiệu quả, mỗi processor sẽ có 1 bộ cân bằng riêng, nhưng nếu việc cân bằng tải gồm nhiều bước và hình thức cân bằng trong mạng tại các processor là tương tự nhau thì điều này sẽ gây lãng phí cho việc thiết kế khi nhúng bộ cân bằng vào các processor.

9.5 Các hướng nghiên cứu có liên quan về Dynamic Load Balancing

Trong mục này chúng tôi xin đề cập đến một số hướng nghiên cứu về Dynamic Load Balancing và các nhận định của chúng tôi về các hướng nghiên cứu ấy.

9.5.1. Dynamic Scheduling: Sắp xếp tiến trình động

Dự đoán tương lai: Một số tác giả nghiên cứu việc cân bằng tải động mạng NOWS (Network of Workstations) bằng cách dự đoán tương

lai chất lượng của mạng dựa vào các thông tin thu thập trong quá khứ. Ví dụ như, ở [10], kiểu mạng Global Distributed đã được giới thiệu, việc cân bằng diễn ra với việc trao đổi thông tin có tính chu kỳ. CHARM [11] đã vận dụng mô hình Local Distributed Receiver-Initiated, trong đó thông tin trao đổi theo dạng Forecasted Finish Time (FFT), FFT trong trường hợp này chính là thời gian cần thiết mà processor hoàn thành công việc còn lại trên processor. Nếu như thông số FFT của một node có giá trị nhỏ hơn mức ngưỡng đã định trước thì node đó sẽ yêu cầu các node lân cận có giá trị FFT cao hơn gửi thêm công việc cho node ấy.

Mô hình Task Queue: Mô hình này có một bộ Task Queue trung tâm thực hiện chức năng cân bằng Queue trong mạng. Khi processor hoàn thành công việc được chỉ định, processor đó sẽ được đề nghị thêm nhiều công việc từ quyết định của bộ cân bằng Queue trung tâm.

Trong đó, tất cả các nghiên cứu hầu như chỉ xoay quanh vấn đề tính toán cân bằng động tổng quát mà không chỉ ra mô hình ứng dụng cụ thể. Trong bài [12], các tác giả cung cấp cho chúng ta mô hình tính toán rõ ràng cho mạng NOWS nhưng gặp phải một số vấn đề về giải thuật cân bằng, tác giả đưa ra bốn bước cho quá trình cân bằng:

Bước 1: Kiểm soát chất lượng processor.

Bước 2: Trao đổi thông tin giữa các processor.

Bước 3: Tính toán việc phân phối mới về tải cho mạng và đưa ra các quyết định về di chuyển công việc cho processor.

Bước 4: Dữ liệu thật được chuyển cho các processor.

Tác giả dùng mô hình Interrupt- Based Receiver – Initiated, tại bước 1 và bước 2 là hai bước của quá trình đồng bộ, sự đồng bộ sẽ được kích khởi khi trong mạng có processor nào xử lý xong phần việc trước nhất sẽ gửi 1 interrupt (1 yêu cầu về gián đoạn) đến tất cả các processor đang hoạt động, và tất cả processor sau khi nhận được yêu cầu interrupt liền gửi thông tin về chất lượng của processor đó cho Load Balancer.

Vấn đề đồng bộ của tác giả không rõ ràng ở chỗ: làm sao biết chính xác được processor nào là xử lý xong công việc trước các processor khác, như vậy cần phải có giá trị thời gian “ngưỡng” xử lý xong công việc của các processor để các processor dựa vào đó so sánh, thì giá trị thời gian này sẽ được xác định dựa trên cơ sở như thế nào?. Ngoài ra, mỗi processor đều phải có khả năng gửi các interrupt cho các processor khác, điều này tăng tính phức tạp cho việc thiết kế tính năng cho mỗi processor, nếu số lượng processor lớn thì vấn đề này sẽ chiếm chi phí rất lớn.

Đối với phạm vi nghiên cứu của chúng tôi, chủ yếu tập trung vào mô hình Global Dynamic Load Balancing với việc định tuyến, thông tin

trong mạng và đường đi phân bố tải đều có thể lập trình trước (Oriented Routing), điều này giảm thiểu tối đa chi phí cho việc định tuyến trong mạng so với trường hợp định tuyến chỉ xác lập trong thời điểm cập nhật tình trạng chất lượng của mạng, điểm mạnh của bài báo sẽ được đề cập rõ ràng trong các mục kế tiếp.

9.5.2. Global Dynamic Load Balancing kết hợp với Shortest Routing trên BHD*

Do mô hình BHD* ($m,2,n$) với hệ số ($d = 2$) có cùng tính chất Shortest Routing như mô hình HD* (m,d,n) có hệ số (d) bất kỳ, nên trong chương này chúng tôi đề nghị mô hình Global Dynamic Load Balancing trong mạng BHD* với mục tiêu là việc tính toán chi phí và tiếp cận giải thuật cân bằng trở nên dễ dàng hơn nhưng việc áp dụng giải thuật cân bằng vào mô hình BHD* vẫn không mất tính tổng quát và luôn đúng cho mô hình mạng HD*.

Trong mục này chúng tôi sẽ định nghĩa một số đại lượng và thông số của mạng, và đồng thời đưa ra giải thuật cho mô hình Global Dynamic Load Balancing kết hợp với giải thuật Shortest Routing trên mạng Binary Hyper-De Bruijn-Aster Multiprocessors Network.

9.5.2.1. Mô hình Global Dynamic Load Balancing

Mạng Multiprocessors được sắp xếp theo Graph Binary Hyper-De Bruijn-Aster.

Mạng Multiprocessors BHD* $(m,2,n)$ có số node là: $2^m \times 2^n = P$ nodes.

Lúc ban đầu khi mạng chưa nhận dữ liệu từ ngoài vào thì mạng BHD* có tính đồng nhất (Homogeneous), tức là các node (hay còn gọi là các đỉnh của Graph) chính là các processor, và các processor này được thiết kế lúc ban đầu theo cùng một chuẩn vật lý: cùng tốc độ xử lý, cùng khả năng lưu trữ dữ liệu trong memory v.v., ngoài ra các liên kết giữa các node (hay còn gọi là các cạnh liên kết các node) có cùng khả năng truyền (cùng tốc độ truyền tín hiệu điện, hầu như không nghẽn trên đường truyền).

Mạng gồm ($P-1$) node processor, và 1 node chính là Load Balancer cũng thuộc mạng BHD*. Load Balancer có các nhiệm vụ như sau:

- a) Là thiết bị trung gian giao tiếp giữa môi trường mạng bên ngoài với môi trường mạng BHD* bên trong.
- b) Nhận dữ liệu từ bên ngoài đưa đến, và chia nhỏ dữ liệu ấy thành các công việc nhỏ hơn.
- c) Tính toán việc phân phối tải cho lần xử lý kế tiếp và đưa ra quyết định dịch chuyển công việc cho từng node
- d) Gửi các dữ liệu thật cần xử lý đến từng node.
- Tất cả các processor trong mạng có cùng chức năng và nhiệm vụ như sau:
- a) Giao tiếp với các node lân cận.
- b) Gửi các thông tin về chất lượng mỗi processor cho Load Balancer tại thời điểm cập nhật.
- c) Nhận công việc từ Load Balancer gửi đến, xử lý rồi trả kết quả về cho Load Balancer.

9.5.2.2. Các đại lượng trong mô hình và một số định nghĩa

Đại lượng của processor: 1) Số processor là ($P-1$), 2) Tốc độ xử lý của các processor là như nhau (V_0).

Đơn vị tải (Load Unit, lu): là khối tải, trong đó có chứa dữ liệu với độ lớn lựa chọn (phải phù hợp với memory và tốc độ xử lý của processor) được lưu trữ tại mỗi processor và cần processor xử lý.

Đơn vị công việc dịch chuyển (Migrating Unit, mu): Dữ liệu đầu vào (công việc do bên ngoài gửi đến Load Balancer) được Load Balancer tổng hợp và phân thành nhiều đơn vị công việc, sau đó dịch chuyển đến các processor chờ xử lý, mỗi processor (i) có khả năng xử lý $W_i \times mu = W_i(mu)$.

Workload (W): Trọng số của mỗi processor thể hiện khả năng chịu tải của processor, cũng chính là tài nguyên hữu dụng còn trống trên mỗi processor (i) có khả năng xử lý $W_i \times khối tải = W_i(lu)$. Lúc chưa chạy chương trình, mỗi processor có cùng khả năng xử lý tối đa (W_0) đơn vị tải.

Để thực hiện Dynamic Load Balancing thì tổng các công việc được chuyển tới mỗi processor phải tỉ lệ với khả năng chịu tải của processors hay trọng số của processor.

Định nghĩa: Độ lớn của đơn vị dữ liệu (lu)= Độ lớn đơn vị dịch chuyển (mu)

Hệ quả: Tổng số đơn vị dịch chuyển được đề nghị tại mỗi node (W_{mu})= Tổng đơn vị tải mà processor đó có thể xử lý (W_{lu})

Thời gian xử lý công việc của 1 processor i: $T_{comp} = W_i / V_o$

Chi phí tính toán tại 1 node (C_{comp}) tỉ lệ thuận với thời gian xử lý công việc tại 1 node.

Chi phí truyền của 1 gói dữ liệu từ Load Balancer đến 1 node và từ Load Balancer đến các node: $C_{comm} = Số node trung gian = K_i = constant$ (do mục 9.2.1.D).

9.5.2.3. Giải thuật đề nghị cho mô hình Global Dynamic Load Balancing

Trong phần này chúng tôi sẽ trình bày kỹ thuật định tuyến từ Load Balancer đến các node và từ các node đến Load Balancer, cũng như đưa ra giải thuật đề nghị cho mô hình Global Dynamic Load Balancing trong BHD*.

a) Kỹ thuật thông tin, truyền dữ liệu giữa các node với Load Balancer và từ Load Balancer đến các node

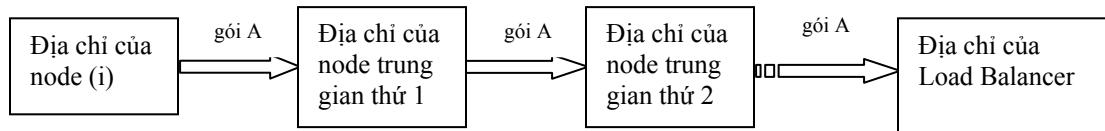
Các node (i): chỉ cần lưu lại địa chỉ các node trung gian biết trước (chính là địa chỉ dạng nhị phân) khi 1 gói đi từ node (i) đến LB (dựa vào phần chứng minh ở mục 9.2.1.D), ngoài ra node i chỉ cập nhật liên kết vật lý với các node lân cận.

Đối với bộ Load Balancer: phải lưu lại địa chỉ của tất cả các node trong mạng, cũng như phải tính toán các số node trung gian phải truyền qua khi LB muốn thông tin, truyền dữ liệu từ LB đến các node (i) trong mạng.

Kỹ thuật thông tin hai chiều giữa node và Load Balancer chủ yếu thực hiện thông qua việc tra địa chỉ đích dựa vào *bảng định tuyến* tại mỗi node, trong đó *bảng định tuyến* chỉ cập nhật các liên kết vật lý đến các node lân cận.

Một gói dữ liệu được truyền sẽ gắn 1 header trong đó có chứa địa chỉ nguồn ,địa chỉ các node trung gian và địa chỉ đích.

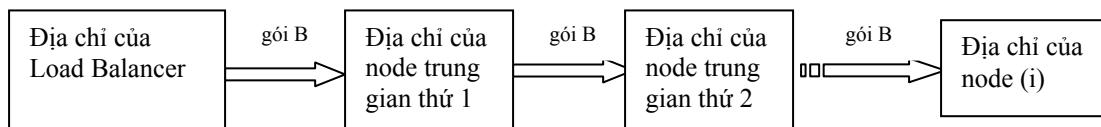
Header của 1 gói A được gửi từ node (i) bất kỳ đến Load Balancer:



Phương thức truyền dữ liệu từ node (i) bất kỳ đến Load Balancer:

Gói cần truyền ở tại node (i) được gắn header, node (i) dựa vào bảng địa chỉ định tuyến thực hiện liên kết vật lý (liên kết điện) với các node bên cạnh và chọn node bên cạnh nào có địa chỉ trùng với “Địa chỉ node trung gian thứ 1” để truyền gói đến, sau khi gói đến, tại đây thực hiện công việc tương tự là tìm node bên cạnh nào có địa chỉ trùng với địa chỉ node trung gian thứ 2 và gửi đến node đó. Công việc truyền cứ tiếp tục và chỉ ngừng cho đến khi gói đến được Load Balancer.

Header của 1 gói B được gửi từ Load Balancer đến 1 node (i):



Việc thông tin từ Load Balancer đến node (i) cũng thông qua các bước như trên.

b) Giải thuật đề nghị cho mô hình Global Dynamic Load Balancing

Tiến trình được xét khi Load Balancer nhận dữ liệu từ ngoài vào lần đầu tiên và được lưu tại bộ đệm của Load Balancer chờ xử lý, tiếp đó sẽ thực hiện cân bằng thông qua các bước sau:

- Lúc ban đầu do (P-1) node ở trạng thái tĩnh (chưa hoạt động) nên có Workload giống nhau W_0
 $Tổng tài nguyên hữu ích còn trong mạng = (P-1).W_0 \text{ (mu)}$ { có giá trị lớn nhất }.
- 1) Load Balancer phân công việc thành (P-1) phần với mỗi phần có giá trị là $W_0 \text{ (mu)}$, sau đó gửi cho từng node.
- Sau khi các node nhận được $W_0 \text{ (mu)}$ liền xử lý rồi trả về dữ liệu yêu cầu cho Load Balancer.
- Khi Load Balancer nhận được dữ liệu trả về sớm nhất từ node (i) thì Load Balancer liền gửi 1 yêu cầu hỏi về thông tin W của mỗi node đến tất cả các node.

- Các node sau khi nhận được gói yêu cầu liền xử lý và trả về Load Balancer các gói mang thông tin W của từng node.
- Load Balancer tổng hợp thông tin W từng node, tính tổng tài nguyên hữu ích còn trống trong mạng tại thời điểm cập nhật thông tin.

Tổng tài nguyên hữu ích còn trống trong mạng = W_0 (node i) + W_1 (node 1)+ W_2 (node 2)++ W_{i-1} (node {i-1})+ W_{i+1} (node {i+1})+..+ W_{P-1} (node {P-1})

Trong đó W_i (node i)= W_0 do lúc này node (i) đã xử lý xong phần việc được phân ban đầu nên tài nguyên còn trống trên node là lớn nhất.

2) Load Balancer phân công việc thành (P-1) phần với mỗi phần có workload (W) tương ứng với từng node.

- Các node nhận từng công việc có độ lớn W tương ứng, xử lý rồi trả kết quả về cho LB.
 - LB tổng hợp các kết quả do các node gửi đến, sau đó chuyển kết quả xử lý về cho môi trường mạng bên ngoài.
- 3) Tiến trình xử lý được lặp lại khi Load Balancer nhận được kết quả trả về sớm nhất từ 1 node (k), quá trình phân phối và cân bằng tải được thiết lập lại.

Nhận xét 1: Quá trình xử lý công việc tại mỗi node vẫn luôn tiếp tục thực hiện kể cả thời điểm Load Balancer gửi gói yêu cầu thông tin Workload của mỗi node, điều này tận dụng tối đa khả năng xử lý của processor, không xảy ra trường hợp node này xử lý ít công việc, hoặc đang “rảnh” trong khi node khác lại xử lý nhiều công việc.

Nhận xét 2: Trong thời điểm (T_1) từ lúc node A gửi cho Load Balancer thông tin về Workload của mỗi node, và thời điểm (T_2) lúc node A nhận được công việc phù hợp với khả năng tải của node đó thì Workload tại hai thời điểm không giống nhau, nhưng điều này không những không ảnh hưởng đến chất lượng mạng mà còn tăng khả năng cân bằng của mạng do $W(T_2) > W(T_1)$, và hiển nhiên là khả năng tải của mỗi node tăng thì chất lượng mạng cũng tăng theo.

9.6. Đánh giá và kết luận

Mô hình Global Dynamic Load Balancing kết hợp với định tuyến tĩnh Shortest Routing trong mạng HD* đã thể hiện tính tối ưu so với các mô hình [12], [13] thông qua giải thuật cân bằng đơn giản và hiệu quả. Mục tiêu của chương này chính là xây dựng một cấu trúc mạng cân bằng mà khả năng xử lý

GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG TẢI SERVER
CHƯƠNG 9: MÔ HÌNH ĐỀ NGHỊ GLOBAL DYNAMIC LOAD BALANCING
KẾT HỢP VỚI SHORTEST ROUTING TRONG HYPER-DEBRUIJN-ASTER
MULTIPROCESSORS NETWORK

của các processor được tận dụng tối đa, giảm thiểu thời gian xử lý, tiết kiệm chi phí và dễ dàng lập trình chức năng cho processor và Load Balancer.

Mô hình này có thể mở rộng cho hệ thống Multicomputers Network, hoặc dùng cho việc thiết kế các loại vi mạch siêu nhỏ tích hợp nhiều processor . Ngoài ra, mô hình này còn có thể ứng dụng trong Heterogeneous System (hệ thống bao gồm các thiết bị có các thuộc tính khác nhau, không đồng nhất một chuẩn thiết kế), ví dụ như là hệ thống Network of Workstation (NOWS).

Tài liệu tham khảo

- [1] Elango Ganesan, Dhiraj K Pradhan, *Wormhole Routing In De Bruijn Networks And Hyper-Debruijn Networks*, Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on Volume: 3 , 25-28 May 2003.
- [2] J. A. Bondy and U. S. R. Murty, *Graph Theory and Application*.
- [3] Wei Shi and Pradip K Srimani, *Hyper-Butterfly Network: A Scalable Optimally Fault Tolerant Architecture*.
- [4] Ngoc Chi Nguyen, Nhat Minh Dinh Vo and Sungyoung Lee, *Fault Tolerant Routing and Broadcasting in de Bruijn networks*.
- [5] Tony Bourke, *Server Load Balancing*, Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.
- [6] Chandra Kopparapu, *Load Balancing Servers, Firewall and Caches*, Wiley Computer Publishing by John Wiley & Sons, Inc, 2002.
- [7] Trịnh Ngọc Minh, *Nhập môn Linux và Mạng TCP/IP*, Đại học Quốc Gia Thành phố Hồ Chí Minh 2001.
- [8] Nguyễn Thị Địệp và Tiêu Đông Nhơn, *Giáo trình Dịch vụ mạng Linux*, Đại học Quốc Gia Thành phố Hồ Chí Minh 12/2005
- [9] Trần Văn Sư, *Truyền số liệu và Mạng thông tin số*, NXB Đại học Quốc Gia Thành phố Hồ Chí Minh 2005.
- [10] N. Nedeljkovic and M. J. Quinn. Data-parallel programming on a network of heterogeneous workstations. *1st HPDC*, Sep 1992.
- [11] V. Saletore et al. Parallel computations on the charm heterogeneous workstn. cluster. *3rd HPDC*, Apr 1994.
- [12] M. Zaki, W. Li, and S. Parthasarathy. Customized dynamic load balancing for a NOW. In *5th IEEE Intl. Symp.High-Performance Distributed Computing, also TR 602, U. Rochester*, Aug. 1996.
- [13] hahzad Malik (219762) Dynamic Load Balancing in a Network of Workstations95.515F Research Report November 29, 2000.
- [14] Lee, B. *Dynamic Load Balancing in a Message Passing Virtual Parallel Machine*. Technical Report, Division of Computer Engineering, School of Applied Science, Nanyang Technological University, Singapore, 1995.

TÓM LƯỢC TIỂU SỬ TÁC GIẢ

NGUYỄN HỒNG THÁI

Sinh viên khoa Điện - Điện tử

Trường Đại học Bách khoa TPHCM

Chuyên ngành Viễn Thông

Đồ án môn học 1: TÌM HIỂU VỀ SMART CARD

Đồ án môn học 2: TÌM HIỂU CẤU TRÚC ẢNH BITMAP VÀ
NHẬN DẠNG BIÊN SỐ XE

Luận văn tốt nghiệp: GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG
TẢI SERVER

Lĩnh vực nghiên cứu yêu thích:

- ❖ Optimize Fault Tolerant Routing, Broadcasting and Load Balancing in Network Models
- ❖ Applications basing on Open Sources

Địa chỉ liên lạc: 241A, Lý Thường Kiệt, P.15, Q.11, TPHCM

Email: nhthai2005@gmail.com

PHẠM MINH TRÍ

Sinh viên khoa Điện - Điện tử

Trường Đại học Bách khoa TPHCM

Chuyên ngành Viễn Thông

Đồ án môn học 1: TÌM HIỂU VHDL

Đồ án môn học 2: TÌM HIỂU CẤU TRÚC ẢNH BITMAP VÀ
NHẬN DẠNG BIÊN SỐ XE

Luận văn tốt nghiệp: GIẢI PHÁP CẢI TIẾN TRONG CÂN BẰNG
TẢI SERVER

Lĩnh vực nghiên cứu yêu thích:

- ❖ Static and Dynamic Load Balancing
- ❖ Parallel and Distributed System
- ❖ Applications of Graph Models in Multiprocessor Network

Địa chỉ liên lạc: 31D1, Tân Quy Đông, Nguyễn Thị Thập, P. Tân Phong, Q.7, TPHCM

Email: minhtriphamlth@yahoo.com