

ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN
THÔNG

LÊ QUANG HÙNG

KỸ THUẬT ĐIỀU KHIỂN LƯU LƯỢNG TRONG
MẠNG MÁY TÍNH CỤM

Chuyên ngành: Khoa học máy tính

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

Thái Nguyên, năm 2015

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn “Kỹ thuật điều khiển lưu lượng trong mạng máy tính cụm” là công trình nghiên cứu do tôi thực hiện dưới sự hướng dẫn của PGS.TS. NGUYỄN VĂN TAM. Các nội dung được trình bày trong luận văn là những kết quả đạt được trong thời tôi gian thực đề tài dưới sự hướng của tập thể giáo viên hướng dẫn, tôi không sao chép nguyên bản lại kết quả của các nghiên cứu đã từng được công bố và đây cũng là kết quả của quá trình nghiên cứu, học tập và làm việc nghiêm túc của tôi trong quá trình học cao học. Bên cạnh đó, trong một số nội dung luận văn là kết quả phân tích, nghiên cứu, tổng hợp từ nhiều nguồn tài liệu khác. Các thông tin tổng hợp hay các kết quả lấy từ nhiều nguồn tài liệu khác đã được tôi trích dẫn một cách đầy đủ và hợp lý. Nguồn tài liệu tham khảo có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Các số liệu và thông tin sử dụng trong luận văn này là trung thực.

Thái Nguyên, ngày 28 tháng 08 năm 2015

Người cam đoan

MỤC LỤC

LỜI CẢM ƠN	Error! Bookmark not defined.
LỜI CAM ĐOAN.....	1
MỤC LỤC	3
DANH SÁCH CÁC TỪ VIẾT TẮT	5
DANH MỤC HÌNH ẢNH.....	6
MỞ ĐẦU	8
CHƯƠNG I: TỔNG QUAN MÁY TÍNH CỤM VÀ VẤN ĐỀ ĐIỀU KHIỂN LƯU LƯỢNG.....	10
1.1 Tổng quan về hệ thống máy tính cụm.....	10
1.1.1 Khái niệm máy tính cụm.....	10
1.1.2 Các loại máy tính cụm	11
1.1.3 Kiến trúc của một Cluster	11
1.1.4 Chế độ hoạt động của Cluster	13
1.1.5 Linux Cluster.....	14
1.2 Lý thuyết lưu lượng.....	16
1.2.1 Khái niệm về lưu lượng và đơn vị Erlang.....	16
1.2.2 Hệ thống tổn thất (Loss System) và công thức Erlang B.....	20
1.2.3 Hệ thống trễ (Delay) và công thức Erlang C	22
1.3 Khái niệm điều khiển lưu lượng	23
1.4 Nhiệm vụ điều khiển lưu lượng	23
1.5 Các cơ chế điều khiển lưu lượng điển hình.....	25
1.5.1 Cơ chế cấp lại ARQ	25
1.5.2 Cơ chế cửa sổ	27
1.5.3 Cơ chế điều khiển truy cập mạng (hạn chế băng thông).....	28
CHƯƠNG 2: KỸ THUẬT ĐIỀU KHIỂN LƯU LƯỢNG TRONG MẠNG MÁY TÍNH CỤM	30
2.1 Hệ thống hàng đợi	30
2.1.1 Các đặc trưng của hệ thống hàng đợi []	30
2.1.2 Phân tích một số mô hình hàng đợi.....	31

2.1.3 Kỹ thuật hàng đợi.....	36
2.2 Điều khiển lưu lượng theo thuật toán gáo rò (leaky bucket)	43
2.2.1 Nguyên lý của thuật toán gáo rò	43
2.2.2 Mô hình giải tích	44
2.2.3 Thuật toán gáo rò (Leaky bucket)	45
2.2.4 Thuật toán gáo rò trong điều khiển lưu lượng	47
CHƯƠNG 3: CHƯƠNG TRÌNH THỰC NGHIỆM	51
3.1 Nhiệm vụ của luận văn.....	51
3.1.1 Bài toán đặt ra	51
3.1.2 Mô hình hệ thống	52
3.2 Xây dựng một Web Cluster.....	52
3.2.1 Mô hình	52
3.2.2 Cài đặt hai server chạy hệ điều hành Linux trên Vmware Workstation.....	54
3.2.3 Cài đặt Apache và PHP trên Linux CentOS	55
3.2.4 Cài đặt và cấu hình heartbeat trên các Node	56
3.3 Chương trình thuật toán gáo rò	58
3.4 Một số kịch bản thử nghiệm.....	61
3.5 Đánh giá kết quả.....	63
KẾT LUẬN.....	64
TÀI LIỆU THAM KHẢO.....	65
PHỤ LỤC : KẾT QUẢ HIỂN THỊ SAU KHI CÀI ĐẶT XONG HEARTBEAT ...	66

DANH SÁCH CÁC TỪ VIẾT TẮT

Từ viết tắt	Viết đầy đủ	Ý nghĩa
ARQ	Automatic Repeat Request	yêu cầu lặp lại tự động
TCP	Transmission Control Protocol	Giao thức điều khiển truyền vận
FBA-TCP	Fair Bandwidth Allocation for TCP	Phân bổ băng thông công bằng cho TCP
LCC	Lost Calls Cleared	Mô hình LCC (Mô hình tổn thất)
PCT -I	Pure Chance Traffic Type I	
PASTA	Poisson Arrival See Time Average	
FEC	Forward Error Correction	Sửa lỗi trực tiếp bên thu
WFQ	Weighted Fair Queue	Hàng đợi cân bằng có trọng số
HA Cluster	High-availability clusters/HA	Hệ thống máy tính cụm có độ sẵn sàng cao

DANH MỤC HÌNH ẢNH

Hình 1.1 Hệ thống máy tính cụm của trung tâm vật lý lý thuyết, viện Vật lý – Viện Hàn Lâm Khoa học và công nghệ Việt Nam	11
Hình 1.2 Kiến trúc một Hadoop Cluster	13
Hình 1.3 Nguyên lý hoạt động của một Cluster.....	13
Hình 1.4 Sơ đồ nguyên lý của một Linux cluster lớn	15
Hình 1.5: Lưu lượng mạng (mật độ)(bằng số thiết bị bận) là một hàm thời gian (đường cong C). Lưu lượng trung bình trong khoảng thời gian T (đường cong D)....	17
Hình 1.6 Hoạt động của mạng khi không có sự kiểm soát	23
Hình 1.7 Phát lại theo cơ chế dừng và đợi	25
Hình 1.8 Nguyên tắc hoạt động của cơ chế cửa sổ trượt	27
Hình 1.9: (a) thuật toán gáo rò với nước, (b) thuật toán gáo rò với các gói tin ...	28
Hình 2.1 Mô hình chung của hệ thống hàng đợi.....	30
Hình 2.2 Chuỗi Markov của một quá trình sinh-tử.....	33
Hình 2.3 Chuỗi Markov của hàng đợi M/M/1	34
Hình 2.4 Chuỗi Markov của hàng đợi M/M/1	35
Hình 2.5 Leaky bucket	37
Hình 2.6 Token Bucket	38
Hình 2.8 Hàng đợi ưu tiên.....	41
Hình 2.9 Custom Queue.....	42
Hình 2.10 Weighted Fair Queue	43
Hình 2.11 Mô hình gáo rò	43
Hình 2.12 Mô hình gáo rò bằng kí hiệu toán học	44
Hình 2.13 Mô hình chuyển đổi sang hàng đợi	45
Hình 2.14 Lưu đồ thuật toán gáo rò	46
Hình 2.15 Thuật toán gáo rò	47
Hình 2.16 Điều khiển lưu lượng đưa vào mạng bằng thuật toán cái gáo rò	47
Hình 2.17 Sử dụng thuật toán cái gáo rò để giới hạn trễ tối đa	48
Hình 2.18 Ví dụ chức năng định dạng lưu lượng của thuật toán gáo rò	49

Hình 3.1 Mô hình mạng của công ty cổ phần Sách giáo dục điện tử EDC	51
Hình 3.2 Mô hình Web Cluster	53
Hình 3.3a Nội dung file ifcfg-eth0 cho máy Node 1	54
Hình 3.3b Nội dung file ifcfg-eth0 cho máy Node 2	55
Hình 3.4 Cấu hình httpd	58
Hình 3.5 : Trước khi điều khiển lưu lượng	62
Hình 3.6 Sau khi điều khiển lưu lượng	63

MỞ ĐẦU

Ngày nay, nền kinh tế của đất nước đang ngày một phát triển và đang hoà nhập với nền kinh tế của khu vực cũng như của thế giới. Cùng với sự phát triển đó mạng máy tính đã và đang trở nên rất quan trọng đối với chúng ta trong mọi lĩnh vực như: Khoa học, quốc phòng, thương mại, giáo dục...hiện nay ở nhiều nơi, mạng đã trở thành một nhu cầu không thể thiếu được.

Việc truyền dữ liệu trong mạng phụ thuộc vào nhiều yếu tố, trong đó có chiến lược cung cấp tài nguyên của mạng (đường truyền, bộ nhớ đệm...). Nếu khả năng tài nguyên có hạn, không có chiến lược cấp phát phù hợp và thích nghi với trạng thái luôn thay đổi của mạng thì dẫn đến tình trạng tắc nghẽn do khả năng tài nguyên trên các thiết bị mạng đáp ứng không nổi. Trong khi đó, có một số thiết bị mạng nào đó có ít dữ liệu truyền qua lại không được tận dụng. Tắc nghẽn có thể xuất hiện ở nhiều vị trí khác nhau trong mạng và đó là kết quả của một số nguyên nhân sau:

Thời gian chờ xử lý, các gói tin xếp hàng trong hàng đợi quá lớn. Nếu các luồng gói tin đột ngột đi vào từ nhiều Interface và tất cả đều muốn đi ra cùng một đường nên hàng đợi sẽ bị đầy (do phải lưu gói tin và chuyển tiếp gói tin ...). Nếu khả năng xử lý của các nút mạng yếu sẽ dẫn đến tắc nghẽn. Kích thước bộ đệm của hàng đợi quá nhỏ. Nếu bộ đệm không đủ dung lượng để lưu các luồng gói tin thì một số gói tin sẽ bị mất. Độ trễ lớn, tần suất lỗi mạng cao và sự chênh lệch về băng thông giữa các liên kết: Làm tăng số lượng gói tin tại các interface đầu vào của các router biên trong mạng làm cho khả năng tắc nghẽn trong mạng tăng lên, và cũng đồng nghĩa với việc số lượng gói tin bị loại bỏ cũng tăng lên nếu các Router biên không có cơ chế hành xử hợp lý.

Trong quá trình học tập và nghiên cứu tôi đã tìm hiểu được một số kiến thức về mạng và tìm hiểu các thuật toán để điều khiển lưu lượng mạng để đảm bảo dữ liệu trong mạng không bị tắc nghẽn. Chính vì thế tôi đã lựa chọn đề tài “Kỹ thuật điều khiển lưu lượng trong mạng máy tính cụm”.

✓ **Đối tượng và phạm vi nghiên cứu**

Đối tượng:

- Kiến trúc hệ thống máy tính cụm
- Thuật toán gáo rò

Phạm vi nghiên cứu:

- Điều chỉnh lưu lượng trong mạng máy tính cụm bằng thuật toán gáo rò

✓ **Hướng nghiên cứu của đề tài**

- Tìm hiểu thành phần, chức năng của hệ thống máy tính cụm, và các vấn đề nảy sinh trong quá trình khởi động mạng máy tính cụm
- Tìm hiểu thuật toán gáo rò.
- Giải quyết bài toán điều khiển lưu lượng dựa trên thuật toán gáo rò

✓ **Những nội dung nghiên cứu chính:**

Luận văn được trình bày trong 3 chương:

Chương 1: Giới thiệu tổng quan về máy tính cụm và các vấn đề điều khiển lưu lượng trong thực tế

Chương 2: Kỹ thuật điều khiển lưu lượng trong mạng máy tính cụm

Chương 3: Đây là chương demo việc triển khai điều khiển lưu lượng trên máy tính cụm bằng thuật toán gáo rò. Trong chương này sẽ hướng dẫn tạo ra một cluster, dùng thuật toán gáo rò để điều khiển lưu lượng.

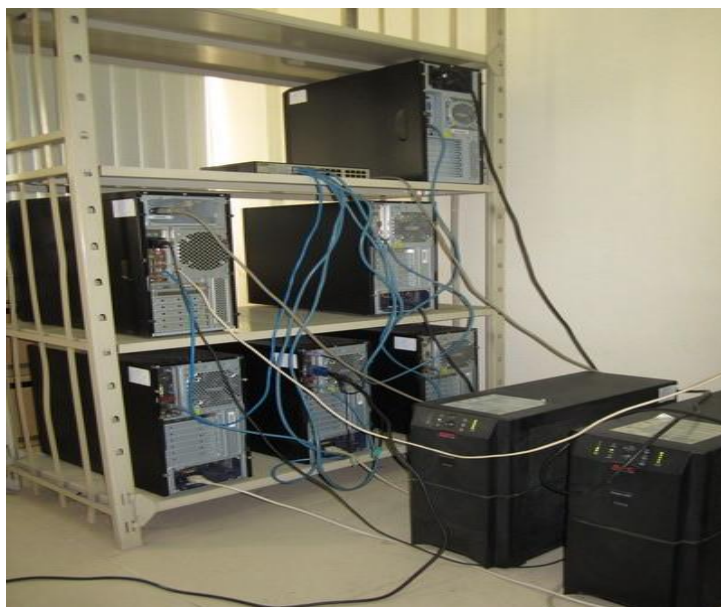
CHƯƠNG I: TỔNG QUAN MÁY TÍNH CỤM VÀ VẤN VỀ ĐIỀU KHIỂN LƯU LƯỢNG

1.1 Tổng quan về hệ thống máy tính cụm

1.1.1 Khái niệm máy tính cụm

Máy tính cụm (Cluster computers) là một cụm gồm nhiều máy tính hoạt động cùng nhau, xử lý các công việc như 1 máy tính logic hay còn gọi là cluster (cụm máy tính). Chúng thường (nhưng không nhất thiết) có chung cấu hình phần cứng và phần mềm. Để đảm bảo việc toàn bộ cụm máy tính (cluster computer) có chung một hệ thống điều hành cũng như các tùy chọn về phần mềm và cấu hình, bạn có thể dùng chung file ảo hóa để khởi động từ đó (ví dụ file .iso). Việc này có thể hoàn thành nhờ sử dụng mạng như một phương thức khởi động, thay vì sử dụng ổ đĩa thông thường. [1]

Các bộ phận của hệ thống máy tính cụm được liên kết với nhau thông qua các loại cáp dùng cho mạng LAN có thể là cáp thường, tốt nhất là cáp quang, để đảm bảo việc truyền dữ liệu được nhanh hơn. Máy tính cụm cùng chạy một hệ điều hành giống nhau. Hệ thống máy tính cụm là 1 giải pháp để nâng cao khả năng của các máy tính không mạnh để chúng trở nên mạnh hơn khi kết hợp với nhau và đặc biệt là tiết kiệm nguồn tài chính.



Hình 1.1 Hệ thống máy tính cụm của trung tâm vật lý lý thuyết,

1.1.2 Các loại máy tính cụm

Hệ thống máy tính cụm có độ sẵn sàng cao (High-availability clusters/HA)

Hệ thống máy tính cụm có độ sẵn sàng cao còn được gọi là failover cluster được sử dụng chủ yếu để cung cấp các hệ thống cluster cần có các dịch vụ yêu cầu độ sẵn sàng cao.

Chúng hoạt động bằng cách cung cấp các nút dự thừa (tức là cung cấp nhiều nút cùng làm 1 nhiệm vụ cùng thực hiện 1 chức năng). Sẽ có tối thiểu 1 nút thừa, nếu 1 nút bị ngừng hoạt động thì ngay lập tức nút kia sẽ đảm nhận nhiệm vụ của nó. Loại hệ thống cluster này cần chi phí khá cao.

Hệ thống cluster chia tải (Load-Balancing clusters)

Đây là loại cluster mà các nút thực hiện việc chia tải công việc để chúng cùng xử lý các công việc .

Đặc điểm của loại hệ thống này là người dùng sẽ chỉ thấy 1 máy chủ

Hệ thống cluster tính toán (Compute clusters)

Thường loại hệ thống này thường sử dụng cho các mục đích tính toán chứ không sử dụng cho các hoạt động mang tính vào ra như các dịch vụ web hoặc cơ sở dữ liệu.

Hệ thống tính toán lưới (Grid computing)

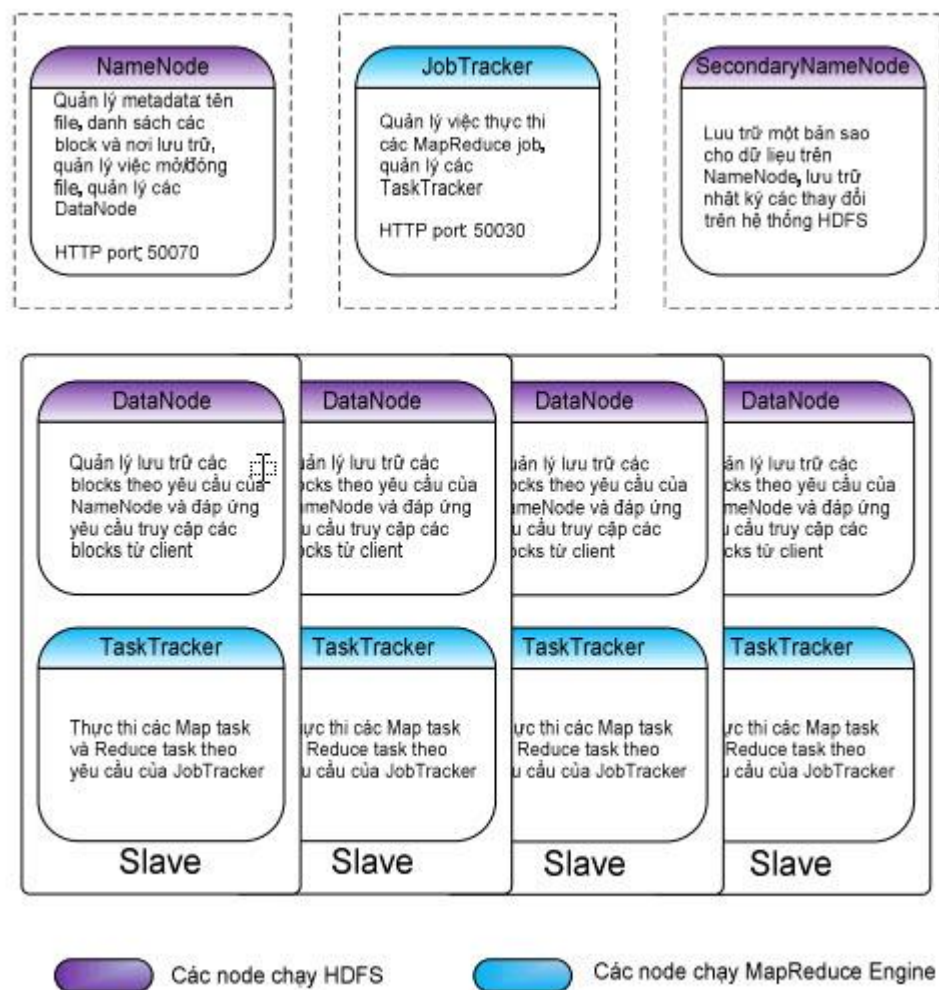
Hệ thống này là hệ thống cluster cực lớn thường bao gồm các máy chủ từ nhiều nơi nối với nhau bằng cáp quang tạo thành 1 mạng lưới, thường chúng sử dụng vào các mục đích yêu cầu phải tính toán phức tạp như các bài toán về dự báo thời tiết hoặc các công ty trong 1 tập đoàn cần xử lý dữ liệu của nhau.

1.1.3 Kiến trúc của một Cluster

Cluster được tổ chức thành các nhóm gọi là các farm hay pack. Trong hầu hết các trường hợp, các dịch vụ ở tầng trước và giữa (front-end and middle-tiers services) được tổ chức thành các farm sử dụng các clone, trong khi đó các dịch vụ tầng sau (back-end services) được tổ chức thành các pack. Các khái niệm farm, pack và clone trong hệ thống cluster sẽ được làm rõ ngay dưới đây.

Farm là một nhóm các máy chủ chạy các dịch vụ giống nhau, nhưng không dùng chung cơ sở dữ liệu. Được gọi là farm (trang trại) bởi vì chúng xử lý bất cứ yêu cầu nào gửi đến cho chúng bằng các bản sao cơ sở dữ liệu (tài nguyên) giống hệt nhau được lưu giữ cục bộ, chứ không dùng chung một bản cơ sở dữ liệu. Cũng bởi tính chất này nên các máy chủ thành viên của farm làm việc độc lập và chúng được gọi là clone (clone là máy tính được thiết kế để mô phỏng chức năng của máy tính khác).

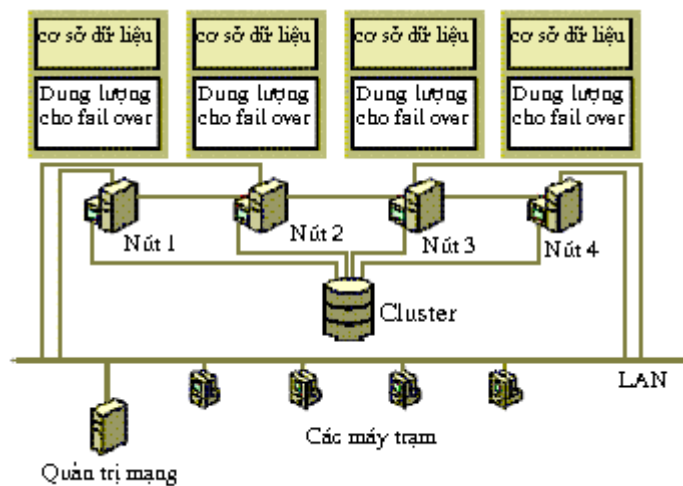
Cluster Pack là một nhóm các máy chủ hoạt động cùng với nhau và chia sẻ với nhau các phần của cơ sở dữ liệu. Được gọi là pack (khối) vì sự hoạt động của các máy chủ thành viên của pack có liên hệ chặt chẽ với nhau và chúng làm việc theo một phương thức thống nhất để quản lý và duy trì các dịch vụ. [5]



Hình 1.2 Kiến trúc một Hadoop Cluster

1.1.4 Chế độ hoạt động của Cluster

Mỗi máy chủ trong cluster được gọi là một nút (cluster node), và có thể được thiết lập ở chế độ chủ động (active) hay thụ động (passive). Khi một nút ở chế độ chủ động, nó sẽ chủ động xử lý các yêu cầu. Khi một nút là thụ động, nó sẽ nằm ở chế độ dự phòng nóng (standby) chờ để sẵn sàng thay thế cho một nút khác nếu bị hỏng.



Hình 1.3 Nguyên lý hoạt động của một Cluster

Trong một cluster có nhiều nút có thể kết hợp cả nút chủ động và nút thụ động. Trong những mô hình loại này việc quyết định một nút được cấu hình là chủ động hay thụ động rất quan trọng. Để hiểu lý do tại sao, hãy xem xét các tình huống sau:

- Nếu một nút chủ động bị sự cố và có một nút thụ động đang sẵn sàng, các ứng dụng và dịch vụ đang chạy trên nút hỏng có thể lập tức được chuyển sang nút thụ động. Vì máy chủ đóng vai trò nút thụ động hiện tại chưa chạy ứng dụng hay dịch vụ gì cả nên nó có thể gánh toàn bộ công việc của máy chủ hỏng mà không ảnh hưởng gì đến các ứng dụng và dịch vụ cung cấp cho người dùng cuối (Ngầm định rằng các các máy chủ trong cluster có cấu trúc phần cứng giống nhau).

- Nếu tất cả các máy chủ trong cluster là chủ động và có một nút bị sự cố, các ứng dụng và dịch vụ đang chạy trên máy chủ hỏng sẽ phải chuyển sang một

máy chủ khác cũng đóng vai trò nút chủ động. Vì là nút chủ động nên bình thường máy chủ này cũng phải đảm nhận một số ứng dụng hay dịch vụ gì đó, khi có sự cố xảy ra thì nó sẽ phải gánh thêm công việc của máy chủ hỏng. Do vậy để đảm bảo hệ thống hoạt động bình thường kể cả khi có sự cố thì máy chủ trong cluster cần phải có cấu hình dư ra đủ để có thể gánh thêm khối lượng công việc của máy chủ khác khi cần.

Trong cấu trúc cluster mà mỗi nút chủ động được dự phòng bởi một nút thụ động, các máy chủ cần có cấu hình sao cho với khối lượng công việc trung bình chúng sử dụng hết khoảng 50% CPU và dung lượng bộ nhớ.

Trong cấu trúc cluster mà số nút chủ động nhiều hơn số nút bị động, các máy chủ cần có cấu hình tài nguyên CPU và bộ nhớ mạnh hơn nữa để có thể xử lý được khối lượng công việc cần thiết khi một nút nào đó bị hỏng.

Các nút trong một cluster thường là một bộ phận của cùng một vùng (domain) và có thể được cấu hình là máy điều khiển vùng (domain controllers) hay máy chủ thành viên. Lý tưởng nhất là mỗi cluster nhiều nút có ít nhất hai nút làm máy điều khiển vùng và đảm nhiệm việc failover đối với những dịch vụ vùng thiết yếu. Nếu không như vậy thì khả năng sẵn sàng của các tài nguyên trên cluster sẽ bị phụ thuộc vào khả năng sẵn sàng của các máy điều khiển trong domain. [5]

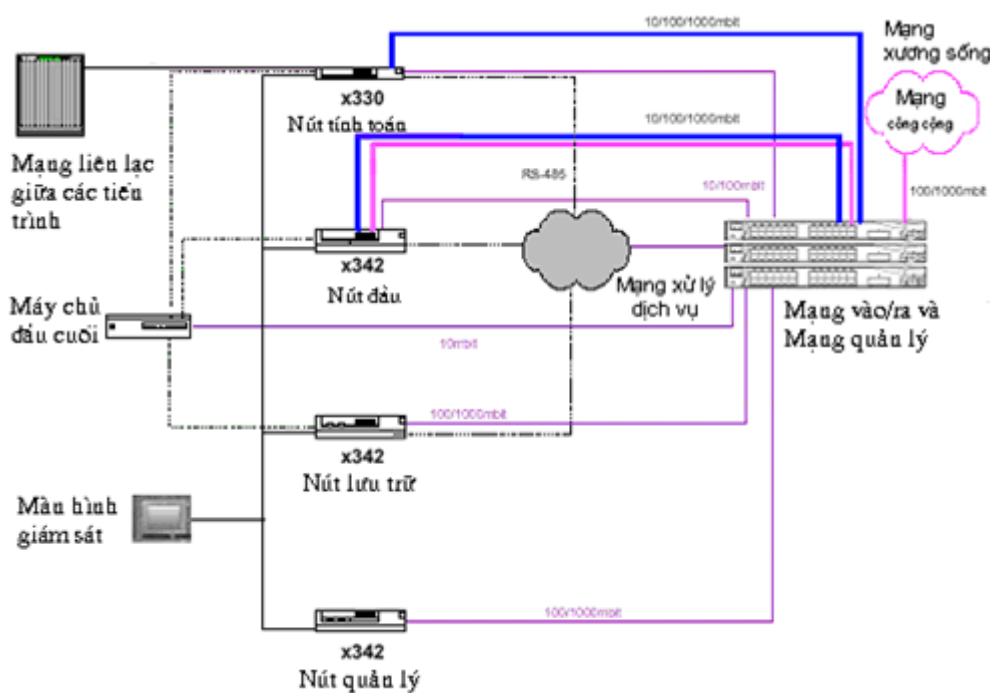
1.1.5 Linux Cluster

Mặc dù công nghệ clustering hiện nay vẫn phổ biến dùng hệ điều hành nguồn đóng, nhưng các thống kê về thị phần và mức tăng trưởng của thị trường máy chủ cho thấy rõ ràng là sự chuyển dịch sang các hệ điều hành nguồn mở như Linux đang ngày càng trở nên hiện thực (IBM đã đầu tư khoảng 1 tỷ USD để phát triển hệ thống IBM Linux cluster. Bởi vậy khi thảo luận về công nghệ clustering, việc tìm hiểu về Linux clustering là một vấn đề rất cần thiết).

Linux cluster được chú ý phát triển nhờ có các đặc điểm như: Giá thành rẻ do phát triển từ hệ điều hành UNIX có mã nguồn mở; Tốc độ tính toán nhanh; Độ tin cậy cao. Linux cluster trên cơ sở bộ xử lý Intel đã trở nên thông dụng trong các viện nghiên cứu. Đó là một phương án không quá tốn kém đối với

những vấn đề của công nghệ thông tin như lập trình song song, phát triển công cụ song song và quản lý các hệ thống phân tán. Đồng thời, Intel Linux cluster cũng đang xuất hiện trong các dự án nghiên cứu trong công nghiệp, mới đầu dưới dạng hộp mẫu hoặc thử nghiệm hệ thống đang thiết kế.

Về nguyên lý hoạt động nói chung hệ thống Linux cluster cũng giống như các hệ thống cluster dùng phần mềm nguồn đóng, tuy nhiên hệ điều hành cơ sở cho Linux cluster là hệ điều hành Linux, được cài đặt trên từng nút của cluster. Chương trình quản lý được dùng trong các Linux cluster tùy theo yêu cầu của khách hàng có thể hỗ trợ các chức năng bao gồm việc cung cấp giao diện dòng lệnh hoặc cửa sổ; Các chức năng quản trị từ xa như thiết đặt lại hệ thống; giám sát các tham số quan trọng; kiểm soát nguồn; xem tệp nhật ký hệ thống; thao tác đơn tác động song song đến nhiều nút v.v.



Hình 1.4 Sơ đồ nguyên lý của một Linux cluster lớn

Như đã nói ở trên, Linux cluster có độ tin cậy và tính ổn định khá cao, tuy nhiên việc thiết kế một Linux cluster hay một siêu cluster không phải là đơn giản, nó đòi hỏi phải xác định được các lớp rất trừu tượng và độ phức tạp tăng theo kích thước của cluster. Các đề án về giải pháp Linux cluster phải do những người

có hiểu biết cần thiết về các vấn đề này xây dựng nên. Việc xác định các nút cần thiết phải theo các nguyên tắc sau:

- Cứ 32 đến 64 nút tính toán cần có một nút đầu mối.
- Mỗi hệ thống cần có một nút quản lý
- Việc vào/ra bên ngoài cần có một hay nhiều nút lưu trữ.

Có ba mạng chức năng cần phải có:

- Mạng dành cho việc liên lạc giữa các tiến trình IPC (inter process communication) với tốc độ phụ thuộc vào bài toán được đặt ra.

- Mạng dùng cho vào/ra tệp (file I/O). Mạng IPC cũng có thể kiêm luôn nhiệm vụ này

- Mạng phục vụ cho việc quản lý hệ thống, thường là mạng được thiết lập bởi các chuyên mạch 10/100 Ethernet. Cũng cần phải có cả máy chủ phục vụ đầu cuối trong mạng này.

1.2 Lý thuyết lưu lượng

1.2.1 Khái niệm về lưu lượng và đơn vị Erlang

Trong lý thuyết lưu lượng viễn thông chúng ta thường sử dụng thuật ngữ lưu lượng để biểu thị cường độ lưu lượng, tức là lưu lượng trong một đơn vị thời gian. Thuật ngữ về lưu lượng có nguồn gốc từ tiếng ý và có nghĩa là “độ bận rộn”.

Theo (ITU-T,1993) định nghĩa như sau:

Cường độ lưu lượng: Mật độ lưu lượng tức thời trong một nhóm tài nguyên dùng chung là số tài nguyên bận tại thời điểm đó.

Nhóm tài nguyên dùng chung có thể là một nhóm phục vụ như đường trung kế. Tiến hành thống kê mật độ lưu lượng hiện tại có thể tính toán cho một chu kỳ T, ta có cường độ lưu lượng trung bình là:

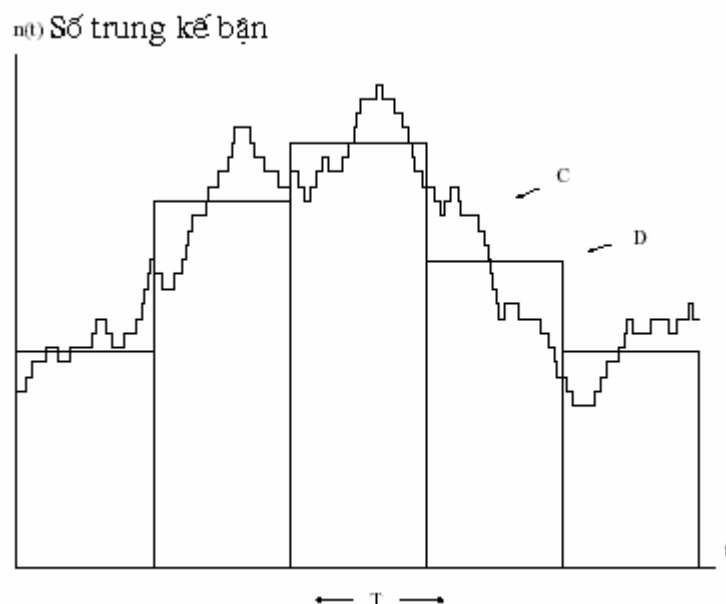
$$Y(T) = \frac{1}{T} \int_0^T n(t) dt$$

Với $n(t)$ là số thiết bị sử dụng tại thời điểm t

Lưu lượng mang

$A_c = Y = A'$ được gọi là lưu lượng được thực hiện bởi một nhóm phục vụ trong khoảng thời gian T (hình 1.5).

Trong thực tế, thuật ngữ cường độ lưu lượng thường có nghĩa là cường độ lưu lượng trung bình.



Hình 1.5: Lưu lượng mang (mật độ) (bằng số thiết bị bận) là một hàm thời gian (đường cong C). Lưu lượng trung bình trong khoảng thời gian T (đường cong D)

Đơn vị của cường độ lưu lượng là Erlang (kí hiệu là Erl), đây là đơn vị không có thứ nguyên. (Ra đời 1946 để ghi nhớ công ơn của nhà toán học người Đan mạch A.K Erlang (1878-1929), người đã tìm ra lý thuyết lưu lượng điện thoại).

Khối lượng lưu lượng: là tổng lưu lượng mang trong chu kỳ T và được đo bằng đơn vị Erlang - giờ (Eh) (theo như tiêu chuẩn ISO những đơn vị tiêu chuẩn có thể là Erlang giây, nhưng thông thường đơn vị Erlang giờ thường sử dụng nhiều hơn).

Lưu lượng mang không thể vượt quá số lượng của đường dây. Một đường dây chỉ có thể mang nhiều nhất một Erlang. Doanh thu của các nhà khai thác tỷ lệ với lưu lượng mang của mạng viễn thông.

Đối với điện thoại cố định thường thì có $A_c = 0,01 \div 0,04$ Erl

Đối với cơ quan : $0,04 \div 0,06$ Erl

Tổng đài cơ quan: 0,6 Erl

Điện thoại trả tiền : 0,7 Erl

Lưu lượng phát sinh A

Lưu lượng phát sinh là lưu lượng được mang nếu không có cuộc gọi nào bị từ chối do thiếu tài nguyên, ví dụ như với số kênh không bị giới hạn.

Lưu lượng phát sinh là một giá trị lý thuyết không đo lường được chỉ có thể ước lượng thông qua lưu lượng mang.

Ta gọi mật độ cuộc gọi là λ , là số cuộc gọi trung bình đến trong một đơn vị thời gian và gọi s là thời gian phục vụ trung bình. Khi đó lưu lượng phát sinh là:

$$A = \lambda \cdot s$$

Từ phương trình này ta thấy rằng đơn vị lưu lượng không có thứ nguyên. Định nghĩa này phù hợp với định nghĩa trên với điều kiện kênh phục vụ không bị giới hạn. Nếu sử dụng cho một hệ thống với năng lực giới hạn ta có sự xác định phụ thuộc vào hệ thống.

Ngoài ra có thể được tính: $A = \lambda / \mu$ (μ : tốc độ phục vụ)

Lưu lượng tổn thất A_r

Lưu lượng tổn thất là độ chênh lệch giữa lưu lượng phát sinh và lưu lượng mang. Giá trị này của hệ thống giảm khi năng lực của hệ thống tăng.

$$A_r = A - A_c$$

Lưu lượng phát sinh là một tham số sử dụng trong tính toán lý thuyết định cỡ. Tuy nhiên, chỉ có lưu lượng mang thường phụ thuộc vào hệ thống thực mới là tham số đo lường được trong thực tế.

Trong hệ thống truyền dẫn số ta không nói về thời gian phục vụ mà chỉ nói về các tốc độ truyền dẫn. Một cuộc giao dịch có thể là quá trình truyền s đơn vị (như bits hay bytes).

Năng lực hệ thống là φ , nghĩa là tốc độ báo hiệu số liệu, được tính bằng đơn vị trên giây (ví dụ bit/s). Như vậy thời gian phục vụ cho một giao dịch như thế tức là thời gian truyền sẽ là s / φ đơn vị thời gian (ví dụ như giây-s); nghĩa là phụ thuộc vào φ .

Nếu trung bình có λ cuộc giao dịch đến trong một đơn vị thời gian, thì độ sử dụng hệ thống sẽ là:

$$\theta = \frac{\lambda \cdot s}{\varphi} \quad \text{Với } 0 \leq \theta \leq 1$$

1.2.2 Hệ thống tổn thất (Loss System) và công thức Erlang B

Công thức Erlang B

Công thức Erlang được mô tả bằng ba thành phần: cấu trúc, chiến lược và lưu lượng.

Cấu trúc: Ta xem xét một hệ thống có n kênh đồng nhất hoạt động song song và được gọi là nhóm đồng nhất (các server, kênh trung kế, khe slot).

Chiến lược: Một cuộc gọi tới hệ thống được chấp nhận nếu còn ít nhất một kênh rỗi (mọi cuộc gọi chỉ cần một kênh rỗi). Nếu tất cả các kênh đều bận thì cuộc gọi sẽ bị huỷ bỏ và nó sẽ bị loại bỏ mà không gây một ảnh hưởng nào sau đó (cuộc gọi bị loại bỏ có thể được chấp nhận trên một tuyến khác). Chiến lược này được gọi là mô hình Loss (tổn thất) Erlang hay mô hình LCC (Lost Calls Cleared).

Lưu lượng: Giả sử rằng trong khoảng thời gian dịch vụ được phân bố theo hàm mũ (số mũ λ), và tiến trình sử dụng là tiến trình Poisson với tốc độ λ . Loại lưu lượng này được gọi là PCT - I (Pure Chance Traffic Type I). Tiến trình lưu lượng này sẽ trở thành tiến trình Markov đơn giản xử lý bằng toán học.

Công thức Erlang B biểu thị mối quan hệ giữa lưu lượng xuất hiện, lượng thiết bị, và xác suất tổn hao như một hàm số được sử dụng rộng rãi như là lý thuyết tiêu chuẩn cho việc lập kế hoạch trong hệ thống viễn thông, vì vậy công thức Erlang B chứa đựng những tiêu chuẩn sau:

Các cuộc gọi xuất hiện một cách ngẫu nhiên:

- Xác suất xảy ra sự cố cuộc gọi là luôn cố định bất chấp thời gian (xác suất cố định xảy ra sự cố của cuộc gọi).
- Xác suất xảy ra sự cố của cuộc gọi không bị ảnh hưởng bởi các cuộc gọi trước (không còn sót lại những đặc điểm của cuộc gọi trước).
- Trong thời gian rất ngắn, không có cuộc gọi nào xuất hiện hoặc chỉ có một cuộc gọi xuất hiện (các cuộc gọi rải rác).

Dạng tổn hao trong khi vận hành khi tất cả các mạch đều bận:

- Trong dạng tổn hao vận hành này, cuộc gọi không thể liên lạc được khi tất cả các mạch đều bận. Trong trường hợp đó tín hiệu được gửi ra ngoài và dù

đường ra trở nên thông suốt sau khi tín hiệu bận được gửi ra thì cuộc gọi vẫn không được kết nối.

- Nhóm mạch ra là nhóm trung kế có khả năng sử dụng hết.
- Thời gian chiếm dụng của các cuộc gọi gần đúng với phân bố hàm mũ.
- Các mạch vào thì vô hạn, còn các mạch ra thì hữu hạn.

Xác suất tổn hao cuộc gọi trong công thức Erlang B được trình bày trong công thức sau:

$$E_n(A) = E_{1,n}(A) = P(n) = \frac{\frac{A^n}{n!}}{1 + A + \frac{A^2}{2!} + \dots + \frac{A^n}{n!}} = \frac{\frac{A^n}{n!}}{\sum_{i=0}^n \frac{A^i}{i!}}$$

Với A - Lưu lượng phát sinh ($A = \lambda.s$)

n - Số kênh

Việc tính toán công thức trên không phù hợp cả khi cả An và $n!$ tăng quá nhanh, khi đó máy tính sẽ bị tràn số do vậy người ta thường áp dụng một số kết quả tính toán và đưa ra công thức sau:

$$E_x(A) = \frac{A.E_{x-1}(A)}{x + A.E_{x-1}(A)} \text{ với } E_0(A) = 1$$

Từ quan điểm toán ứng dụng, hàm tuyến tính có độ ổn định cao nhất ta có:

$$I_x(A) = 1 + \frac{x}{A} I_{x-1}(A) \text{ với } I_0(A) = 1$$

Ở đây $I_n(A) = 1/E_n(A)$

Công thức này hoàn toàn chính xác, thậm chí với các giá trị $(n.A)$ lớn vẫn không xuất hiện lỗi. Đây là công thức cơ bản cho rất nhiều bảng số của công thức Erlang B

Các đặc tính lưu lượng của công thức Erlang B

Biết được xác suất trạng thái ta có thể biết được các số đo hiệu năng.

- Độ nghẽn theo thời gian: là xác suất mà tất cả các trung kế bị chiếm tại một thời điểm bất kỳ bằng với phần thời gian tất cả các trung kế bị chiếm trên tổng thời gian.

- Độ nghẽn theo cuộc gọi: xác suất mà một cuộc gọi bất kỳ bị mất bằng tỷ lệ số cuộc gọi bị chặn trên tổng các cuộc gọi.

$$\text{Độ nghẽn lưu lượng: } C = \frac{A - Y}{A} = E_n(A)$$

Ta có $E = B = C$, bởi vì cường độ cuộc gọi độc lập với trạng thái, đây chính là tính chất PASTA (Poisson Arrival See Time Average), nó phù hợp với tất cả các hệ thống tuân theo tiến trình Poisson. Trong tất cả các trường hợp khác, ít nhất có ba tham số đo tắc nghẽn là khác nhau.

1.2.3 Hệ thống trễ (Delay) và công thức Erlang C

Xét lưu lượng với tiến trình poisson (Không giới hạn về tài nguyên). Phân bố thời gian phục vụ là PCT-1. Hệ thống hàng đợi này có tên là hệ thống trễ Erlang. Trong hệ thống này thì lưu lượng mang sẽ bằng lưu lượng phát sinh và không có khách hàng nào bị nghẽn.

Công thức Erlang C

w là biến ngẫu nhiên của thời gian đợi thì ta có xác suất để biến $w > 0$ là:

$$E_{2,n}(A) = P(w > 0) = \frac{\frac{A^n}{n!} \cdot \frac{n}{n-A}}{1 + A + \frac{A^2}{2!} + \dots + \frac{A^{n-1}}{(n-1)!} + \frac{A^n}{n!} \cdot \frac{n}{n-A}} \quad (A < n)$$

Cho biết xác suất cuộc gọi đến hệ thống thì nó phải bị xếp vào hàng đợi (do số kênh giới hạn).

Xác suất để 1 khách hàng đợi phục vụ ngay :

$$S_n = E_{2,n}(A)$$

Công thức hồi quy:

$$\frac{1}{E_{2,n}(A)} = \frac{1}{E_{1,n}(A)} - \frac{1}{E_{1,n-1}(A)}$$

$$I_{2,n}(A) = I_{1,n}(A) - I_{1,n-1}(A)$$

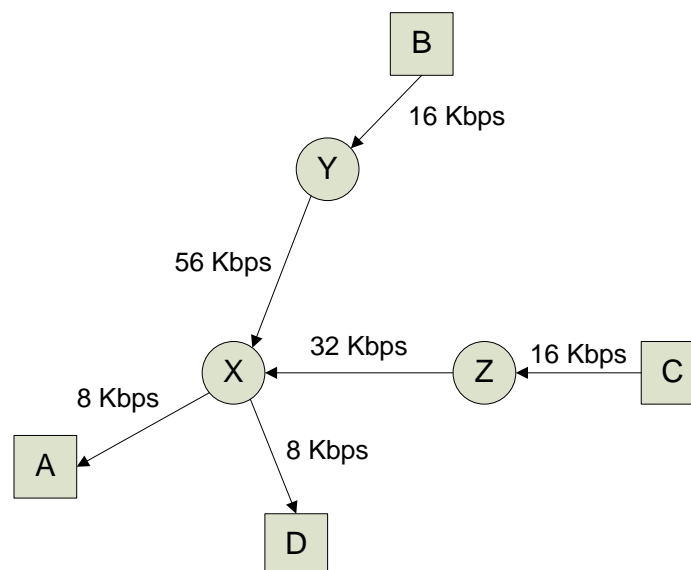
$$I_{2,n}(A) = \frac{1}{E_{2,n}(A)}$$

Lưu lượng phát sinh bằng lưu lượng mang: $A=Y$ (Chỉ áp dụng cho mô hình trể).

1.3 Khái niệm điều khiển lưu lượng

Điều khiển luồng là cơ chế nhằm đảm bảo việc truyền thông tin của phía phát không vượt quá khả năng xử lý của phía thu.

Chống tắc nghẽn là cơ chế kiểm soát thông tin đi vào mạng nhằm đảm bảo tổng lưu lượng thông tin đi vào mạng không vượt quá khả năng xử lý của toàn mạng.



Hình 1.6 Hoạt động của mạng khi không có sự kiểm soát

Chống tắc nghẽn liên quan đến việc kiểm soát thông tin trên toàn mạng, trong khi điều khiển luồng là việc kiểm soát thông tin giữa hai đầu cuối cụ thể. Hai kỹ thuật này có điểm tương đồng là phải giới hạn lưu lượng thông tin nhằm tránh khả năng quá tải của hệ thống đích. Do tính chất gắn kết của hai khái niệm này, đa phần các tài liệu đều sử dụng lẫn (hoặc kết hợp) các khái niệm điều khiển luồng (flow control) và điều khiển tắc nghẽn (congestion control).

Vì lý do đó, trong luận văn này, tôi sử dụng khái niệm điều khiển lưu lượng để diễn tả cả hai phạm trù. Trong trường hợp cụ thể cần phải phân biệt làm rõ hai khái niệm, tôi sẽ có những chú thích rõ ràng.

1.4 Nhiệm vụ điều khiển lưu lượng

Điều khiển lưu lượng được sử dụng khi có sự giới hạn về tài nguyên giữa người sử dụng với điểm truy nhập mạng, hay giữa hai thiết bị mạng để kiểm soát thông tin trên mạng. Điều khiển lưu lượng được sử dụng nhiều nhất tại các lớp liên kết dữ liệu(data link), lớp mạng(network), và lớp giao vận(transport).

Mục đích chính của việc sử dụng điều khiển lưu lượng trong mạng là nhằm:

Tối ưu hóa thông lượng sử dụng của mạng: trong trường hợp thông tin chỉ truyền giữa hai người dùng, việc tối ưu hóa tốc độ truyền tin không cần đặt ra. Tuy nhiên, trong một hệ thống mạng với sự tham gia trao đổi thông tin của nhiều nút mạng, việc tối ưu hóa thông lượng của hệ thống mạng phức tạp hơn nhiều.

Giảm trễ gói khi đi qua mạng: đứng trên phương diện người sử dụng, trễ gói từ đầu cuối đến đầu cuối càng nhỏ càng tốt. Tuy nhiên, điều khiển luồng (ở lớp mạng) không nhằm thực hiện điều đó. Điều khiển luồng chỉ đảm bảo trễ của gói tin khi đi qua mạng nằm ở một mức chấp nhận được thông qua việc giới hạn số lượng gói tin đi vào mạng (và do đó, giảm trễ hàng đợi). Vì lý do đó, điều khiển luồng không có tác dụng với những ứng dụng đòi hỏi trễ nhỏ trong khi lại truyền trên hệ thống hạ tầng tốc độ thấp. Trong trường hợp này, việc đáp ứng yêu cầu của người sử dụng chỉ có thể được thực hiện thông qua việc nâng cấp hệ thống hay sử dụng các giải thuật định tuyến tối ưu hơn. Mục đích chính của việc giảm trễ gói là để giảm sự lãng phí tài nguyên khi phải truyền lại gói. Việc truyền lại có thể do hai nguyên nhân: (1) hàng đợi của các nút mạng bị đầy dẫn đến gói thông tin bị hủy và phải truyền lại; (2) thông tin báo nhận quay trở lại nút nguồn quá trễ khiến phía phát cho rằng thông tin truyền đi đã bị mất và phải truyền lại.

Đảm bảo tính công bằng cho việc trao đổi thông tin trên mạng: đảm bảo tính công bằng trong trao đổi thông tin là một trong những yếu tố tiên quyết của kỹ thuật mạng. Việc đảm bảo tính công bằng cho phép người sử dụng được dùng tài nguyên mạng với cơ hội như nhau. Trong trường hợp người sử dụng

được chia thành các nhóm với mức độ ưu tiên khác nhau thì bảo đảm tính công bằng được thực hiện đối với các người dùng trong cùng một nhóm.

Đảm bảo tránh tắc nghẽn trong mạng: tắc nghẽn là hiện tượng thông lượng của mạng giảm và trễ tăng lên khi lượng thông tin đi vào mạng tăng. Điều khiển luồng cung cấp cơ chế giới hạn lượng thông tin đi vào mạng nhằm tránh hiện tượng tắc nghẽn kể trên. Có thể hình dung điều khiển luồng như hoạt động của cảnh sát giao thông trên đường phố vào giờ cao điểm.

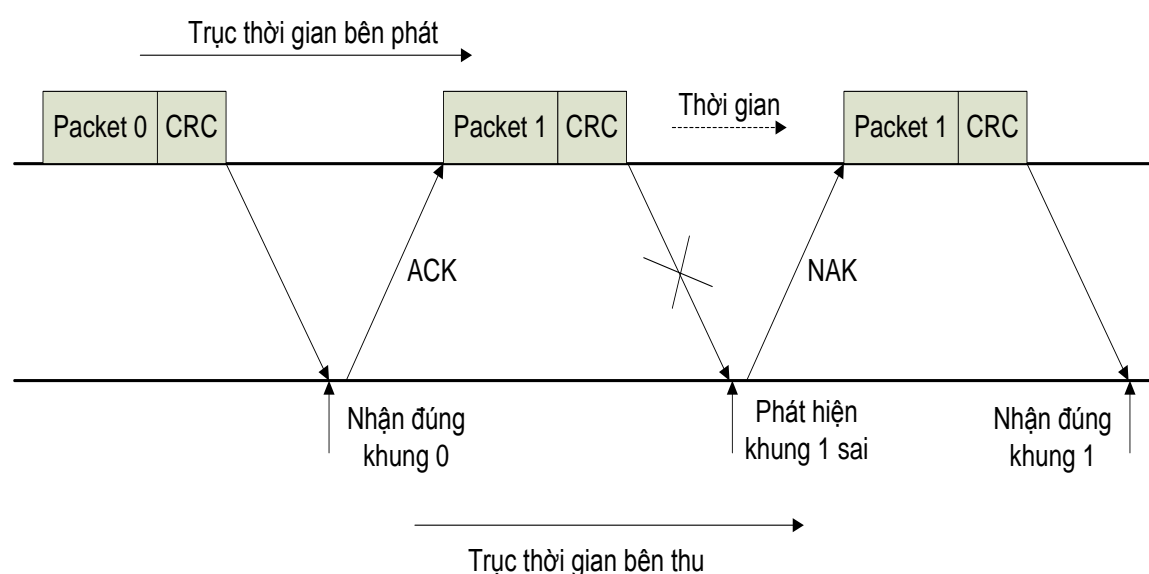
1.5 Các cơ chế điều khiển lưu lượng điển hình

Các cơ chế điều khiển lưu lượng được phân ra làm ba loại chính:

- Các cơ chế cấp phát lại ARQ
- Các cơ chế cửa sổ
- Các cơ chế điều khiển truy nhập mạng

1.5.1 Cơ chế cấp lại ARQ

Các cơ chế điều khiển lưu lượng theo phương pháp cửa sổ được hoạt động tương tự như các cơ chế phát lại ARQ (Automatic Repeat Request).



Hình 1.7 Phát lại theo cơ chế dừng và đợi

Khi truyền thông tin trong mạng, thông tin truyền từ phía phát sang phía thu có thể bị sai lỗi hoặc mất. Trong trường hợp thông tin bị mất, cần phải thực hiện truyền lại thông tin. Với trường hợp thông tin bị sai, có thể sửa sai bằng một trong hai cách:

Sửa lỗi trực tiếp bên thu: phía thu sau khi phát hiện lỗi có thể sửa lỗi trực tiếp ngay bên thu mà không yêu cầu phải phát lại. Để có thể thực hiện được điều này, thông tin trước khi truyền đi phải được cài các mã sửa lỗi (bên cạnh việc có khả năng phát hiện lỗi, cần có khả năng sửa lỗi).

Yêu cầu phía phát truyền lại: phía thu sau khi kiểm tra và phát hiện có lỗi sẽ yêu cầu phía phát truyền lại thông tin.

Đặc điểm của hai phương pháp sửa lỗi trên:

Sửa lỗi trực tiếp bên thu (Forward Error Correction – FEC): chỉ cần truyền thông tin một lần, không yêu cầu phải truyền lại thông tin trong trường hợp có lỗi. Tuy nhiên, số lượng bit thông tin có thể sửa sai phụ thuộc vào số loại mã sửa sai và số bit thông tin thêm vào cho mục đích sửa sai. Nhìn chung, số bit thông tin thêm vào càng lớn thì số bit có thể sửa sai càng nhiều, tuy nhiên hiệu suất thông tin (số bit thông tin hữu ích trên tổng số bit truyền đi) lại thấp.

Sửa lỗi bằng cách truyền lại: khác với sửa lỗi trực tiếp bên thu, trong trường hợp sửa lỗi bằng cách truyền lại, thông tin trước khi phát chỉ cần thêm các bit thông tin phục vụ cho mục đích phát hiện lỗi (số bit thêm vào ít hơn so với trường hợp sửa lỗi) do đó hiệu suất truyền thông tin cao hơn so với trường hợp trên. Tuy nhiên, trong trường hợp có lỗi xảy ra với khung thông tin thì toàn bộ khung thông tin phải được truyền lại (giảm hiệu suất truyền tin).

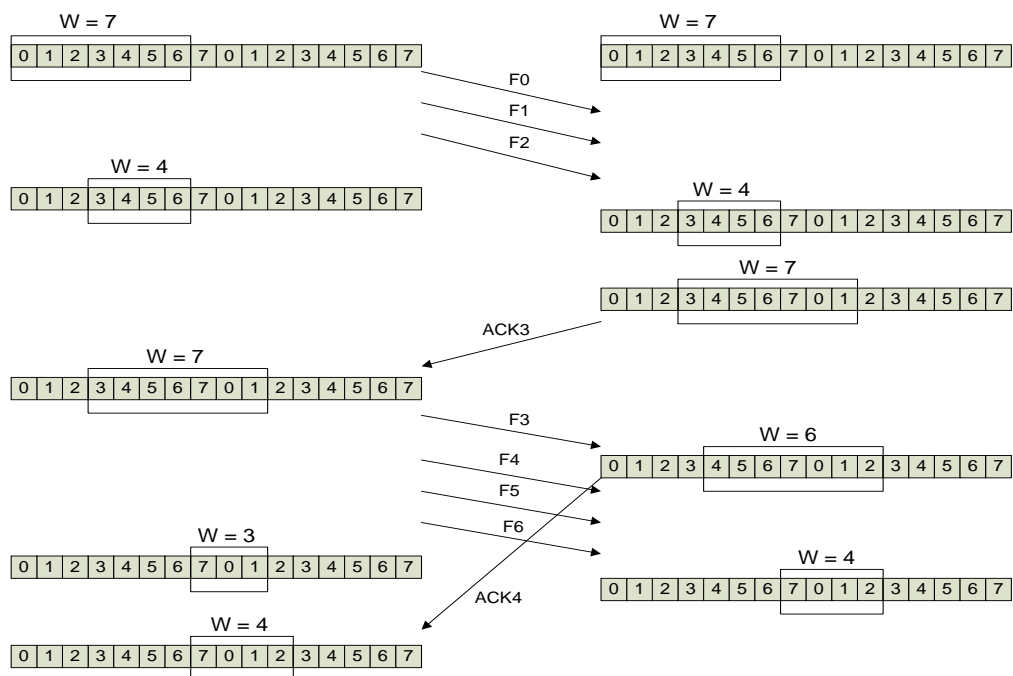
Với ưu nhược điểm của các phương pháp trên, sửa lỗi bằng cách truyền lại thường được dùng trong môi trường có tỷ lệ lỗi bit thấp (truyền dẫn hữu tuyến) trong khi sửa lỗi bên thu thường được dùng trong trường hợp môi trường truyền dẫn có tỷ lệ lỗi bit cao (vô tuyến). Để có thể đối phó với trường hợp lỗi chùm (burst noise), có thể áp dụng một số cơ chế như ghép xen kẽ thông tin (interleaving).

Các cơ chế phát lại được chia ra làm 3 loại chính:

- Cơ chế phát lại dừng và đợi (Stop-and-Wait ARQ)
- Cơ chế phát lại theo nhóm (Go-back-N ARQ)
- Cơ chế phát lại có lựa chọn (Selective repeat ARQ)

1.5.2 Cơ chế cửa sổ

Cơ chế điều khiển lưu lượng dựa trên phương pháp cửa sổ được thực hiện bởi việc giới hạn số lượng gói tin được truyền ở phía phát nhằm đảm bảo thông tin này không vượt quá khả năng xử lý của phía thu.



Hình 1.8 Nguyên tắc hoạt động của cơ chế cửa sổ trượt

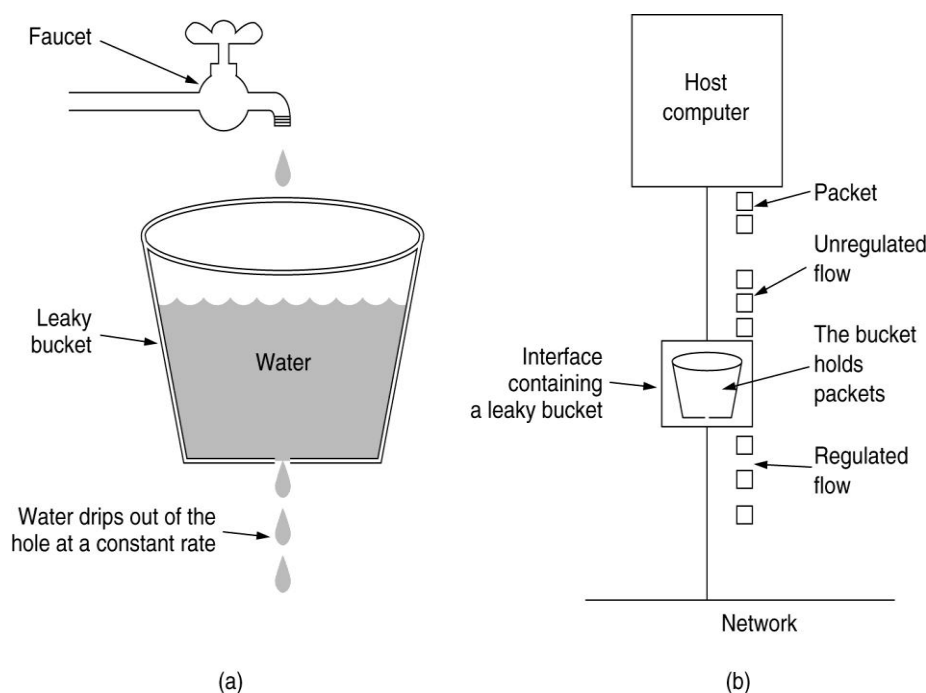
Theo cơ chế này, phía phát sẽ không thực hiện phát tin chừng nào phía thu còn chưa xử lý xong gói tin (hoặc một số gói tin) trước đó. Khi phía thu xử lý xong thông tin do phía phát gửi đến thì nó sẽ báo cho phía phát biết và lúc này, phía phát sẽ tiếp tục gửi các gói tin tiếp theo. Cơ chế này đảm bảo việc truyền tin không bao giờ vượt quá khả năng xử lý của phía thu.

Với việc kết hợp hoạt động nhịp nhàng giữa phía phát và phía thu (có sử dụng báo nhận), số lượng gói tin đồng thời tồn tại trên đường truyền nằm trong giới hạn nhất định. Nếu phía thu có bộ đệm với dung lượng lớn hơn tổng kích thước các gói tin này thì bộ đệm phía thu sẽ không bao giờ bị tràn.

Các tiến trình thông tin có thể chịu sự ảnh hưởng của điều khiển luồng gồm có các kênh ảo độc lập, một nhóm các kênh ảo hay toàn bộ luồng thông tin từ một nút mạng này đến một nút mạng khác.

1.5.3 Cơ chế điều khiển truy cập mạng (hạn chế băng thông)

Cơ chế kiểm soát băng thông (truy cập mạng) đảm bảo lượng thông tin của người dùng đưa vào mạng không vượt quá một mức nào đó nhằm tránh tắc nghẽn trong mạng. Trong một số trường hợp cụ thể, thông tin của người dùng đưa vào mạng có thể vượt quá lượng thông tin giới hạn ở một mức độ nào đó cho phép.



Hình 1.9: (a) thuật toán gáo rò với nước, (b) thuật toán gáo rò với các gói tin

Cơ chế kiểm soát băng thông của thông tin đi vào mạng chia làm hai loại:

- Kiểm soát chặt (strict implementation) – với tốc độ thông tin vào mạng trung bình là r gói/s, thì hệ thống kiểm soát sẽ chỉ cho một gói vào cứ sau mỗi $1/r$ giây. Phương pháp này không phù hợp cho các thông tin có thay đổi với biên độ lớn (bursty traffic). Ví dụ điển hình của phương pháp này là cơ chế TDMA.

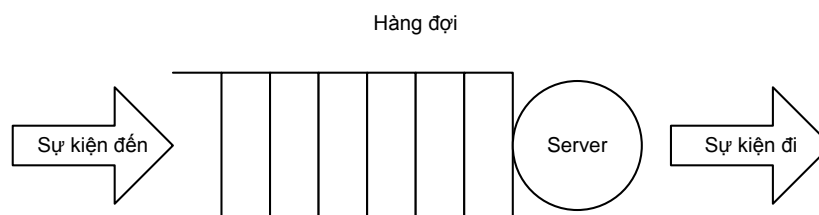
- Kiểm soát lỏng (less-strict implementation) – với tốc độ thông tin vào mạng trung bình là r gói/s thì hệ thống kiểm soát sẽ cho W gói vào mạng trong khoảng thời gian W/r giây. Trong phương pháp này, tốc độ dữ liệu trung bình là không đổi nhưng cho hệ thống cho phép nhận tối đa W gói tại một thời điểm (bursty traffic). Cơ chế này thường được triển khai với việc sử dụng gáo rò (leaky bucket)

CHƯƠNG 2: KỸ THUẬT ĐIỀU KHIỂN LƯU LƯỢNG TRONG MẠNG MÁY TÍNH CỤM

Chúng ta đã biết nền tảng của thuật toán gáo rò là việc điều khiển hàng đợi tại nút mạng nên trong chương này, chúng ta sẽ (1) tìm hiểu về hệ thống hàng đợi, (2) nghiên cứu thuật toán gáo rò về phương diện toán học, kỹ thuật, và ứng dụng của thuật toán gáo rò cho điều khiển lưu lượng.

2.1 Hệ thống hàng đợi

2.1.1 Các đặc trưng của hệ thống hàng đợi []



Hình 2.1 Mô hình chung của hệ thống hàng đợi

Bất kì hệ thống hàng đợi nào cũng được mô hình hóa như hình 2.1 và được đặc trưng bởi các yếu tố sau: Quá trình đến, thời gian phục vụ, số lượng server, chiều dài hàng đợi, kích cỡ bộ đệm cực đại, quy tắc phục vụ

Đặc điểm của hệ thống hàng đợi

- Miêu tả của tiến trình đến (phân bố khoảng thời gian đến)
- Miêu tả của tiến trình phục vụ (phân bố thời gian phục vụ)
- Số lượng server
- Số lượng các vị trí đợi

Các quy tắc hàng đợi đặc biệt:

- Quy tắc phục vụ (FCFS, LCFS, RANDOM)
- Thời gian rỗi (phân bố thời gian rỗi, khi mà thời gian rỗi bắt đầu)
- Mức độ ưu tiên
- Những luật khác

Thông số quan trọng cho người sử dụng:

- Trễ hàng đợi
- Tổng trễ (bao gồm trễ hàng đợi và trễ phục vụ)

- Số lượng khách hàng trong hàng đợi
- Số lượng khách hàng trong hệ thống (gồm khách hàng chờ và khách hàng đang được phục vụ)
- Xác suất nghẽn mạng (khi kích thước bộ đệm hữu hạn)
- Xác suất chờ để phục vụ
- Thông số quan trọng cho các nhà cung cấp dịch vụ:
- Khả năng sử dụng server
- Khả năng sử dụng bộ đệm
- Lợi ích thu được (thông số dịch vụ và các xem xét về kinh tế)
- Lợi ích bị mất (thông số dịch vụ và các xem xét về kinh tế)
- Đáp ứng nhu cầu của người sử dụng

Chất lượng dịch vụ (QoS):

- Tồn thất (PDF, mean)
- Trễ (PDF, mean)
- Jitter (PDF, mean)

Đưa ra các thông số trên để thu được:

- Hàm phân bố xác suất
- Các giá trị trung bình
- Đo được các thời điểm cực đại, cực tiểu

Các hàm phân bố xác suất chứa đựng đầy đủ các thông tin liên quan đến các thông số quan tâm. Tuy nhiên, việc thiết lập được các hàm này là khó thực hiện.

2.1.2 Phân tích một số mô hình hàng đợi

2.1.2.1 Ký hiệu Kendall []

Bất kỳ hệ thống hàng đợi nào cũng được mô tả bởi :

Tiến trình đến

Nếu các khách hàng đến vào các thời điểm $t_1, t_2 \dots t_j$ thì các biến số ngẫu nhiên $P_j = t_j - t_{j-1}$ được gọi là các thời điểm giữa các lần đến. Các thời điểm này thường được giả thiết là các biến số ngẫu nhiên độc lập và được phân bố đồng

nhất IID (Independent and Identically distributed). Các tiến trình đến thông dụng nhất là :

M: Tiến trình mũ (là tiến trình Markov hay tiến trình không nhớ)

Er: Tiến trình Erlang bậc r

Hr: Tiến trình siêu số mũ bậc r

D: Tiến trình tất định (deterministic)

G: Tiến trình chung

Tiến trình phục vụ

Thời gian mà mỗi công việc tiêu tốn cần thiết tại server gọi là thời gian phục vụ. Các thời gian phục vụ thường giả thiết là các biến số ngẫu nhiên IID. Các tiến trình phục vụ thông dụng nhất cũng giống như thời gian đến.

Số lượng các bộ server

Số lượng các server phục vụ cho hàng đợi

Dung lượng hệ thống

Kích thước bộ nhớ đệm cực đại

Qui mô mật độ

Số lượng các công việc đến tại hàng đợi. Qui mô mật độ luôn là hữu hạn trong các hệ thống thực. Tuy nhiên phân tích hệ thống với qui mô mật độ lớn sẽ dễ dàng hơn nếu giả thiết rằng qui mô mật độ là vô hạn.

Qui tắc phục vụ

Thứ tự mà theo đó các công việc trong hàng xếp được phục vụ. Các qui tắc phổ biến nhất là đến trước phục vụ trước FCFS (First Come First Served), đến sau phục vụ trước LCFS (Last Come First Served), theo vòng tròn RR (Round Robin), thời gian xử lý ngắn nhất phục vụ trước SPT (Shortest Processing Time First) và thời gian xử lý ngắn nhất được đề cử SRPT (Shortest Remaining Processing Time First)

Ký hiệu Kendall

A/S/m/B/K/SD được sử dụng rộng rãi để mô tả hệ thống xếp hàng

A: Phân bố thời gian giữa các lần đến

S: Phân bố thời gian phục vụ

m: Số lượng server

B: Kích thước bộ đệm

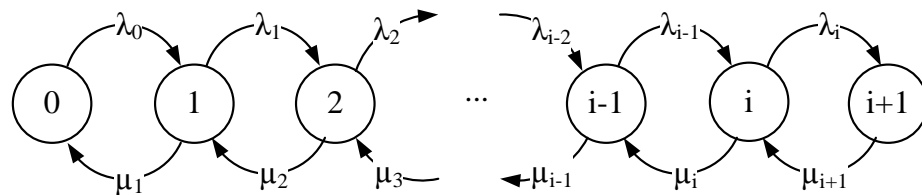
K: Quy mô mật độ

SD: Quy tắc phục vụ

Ví dụ hàng đợi M/D/1: M có nghĩa tiến trình đến là tiến trình Markov không nhớ (với thời gian giữa các lần đến theo hàm mũ); D thời gian phục vụ luôn như nhau (tất định); 1 có một server duy nhất phục vụ. Phần B/K/SD của ký hiệu bị loại trừ để cho thấy rằng dung lượng của hệ thống và qui mô mật độ là vô hạn và qui tắc phục vụ là FCFS.

2.1.2.2 Quá trình Sinh-Tử (Birth-Death)

Trạng thái của hệ thống được biểu diễn bằng số các khách hàng n trong một hệ thống. Khi có một khách hàng mới đến thì trạng thái của hệ thống sẽ thay đổi sang n+1, khi có một khách hàng ra đi thì trạng thái hệ thống sẽ thay đổi sang n-1, ta có lược đồ chuyển tiếp trạng thái là quá trình sinh tử.



Hình 2.2 Chuỗi Markov của một quá trình sinh-tử

λ_n : Tốc độ của lần đến n

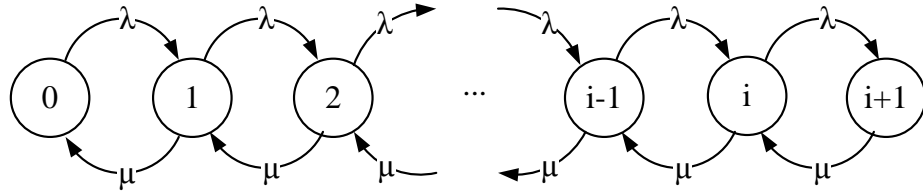
μ_n : Tốc độ của lần đi

P_n : Xác suất ổn định trạng thái n của quá trình sinh – tử tại trạng thái n

$$P_n = \frac{\lambda_0 \lambda_1 \dots \lambda_{n-1}}{\mu_1 \mu_2 \dots \mu_n} P_0$$

P_0 - xác suất ở trạng thái 0, P_n - xác suất ở trạng thái n

2.1.2.3 Hàng đợi M/M/1



Hình 2.3 Chuỗi Markov của hàng đợi M/M/1

Tất cả các tốc độ đến đều là λ , μ

λ : Tốc độ của lần đến

μ : Tốc độ của lần đi

$$P_n = \left(\frac{\lambda}{\mu}\right)^n P_0 = \rho^n P_0$$

P_n : Xác suất ổn định trạng thái n

P_0 : Xác suất ổn định trạng thái 0

ρ : Mật độ lưu lượng $\rho = \frac{\lambda}{\mu}$

Trong trường hợp này số kênh phục vụ bằng 1, chỉ có 1 server

Các công thức tính toán:

Xác suất có n khách hàng trong hệ thống

$$P_n = (1 - \rho) \rho^n ; n=1,2,\dots$$

$$P_0 = (1 - \rho)$$

Số lượng trung bình các khách hàng trong hệ thống:

$$L = E(n) = \frac{\rho}{1 - \rho}$$

Phương sai: $\sigma_n^2 = \frac{\rho}{(1 - \rho)^2}$

Tham số thời gian

Thời gian trung bình của 1 khách hàng trong hệ thống: W

$$W = \frac{L}{\lambda} = \frac{\rho}{\lambda(1 - \rho)} = \frac{1}{\mu - \lambda}$$

Thời gian phục vụ trung bình cho một khách hàng : W_s

$$W_s = \frac{1}{\mu} = \frac{\rho}{\lambda}$$

Thời gian trung bình của khách hàng trong hàng đợi

$$W_q = W - W_s = \frac{\rho}{\lambda(1-\rho)} - \frac{\rho}{\lambda} = \frac{\rho^2}{\lambda(1-\rho)}$$

Chiều dài hàng đợi

Số lượng trung bình các khách hàng trong hệ thống

$$L = \frac{\rho}{1-\rho}$$

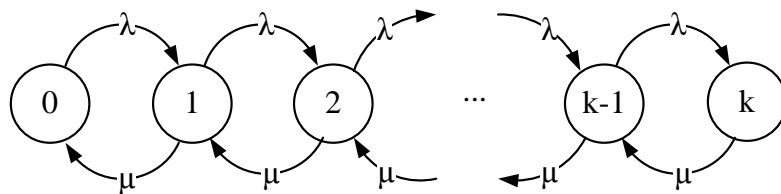
Số lượng trung bình các job trong server: L_s

$$L_s = 1P(n \geq 1) = 1 - P(n=0) = 1 - (1-\rho) = \rho$$

Số lượng trung bình của các công việc trong hàng đợi L_q

$$L_q = L - L_s = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho}$$

2.1.2.4. Hàng đợi M/M/1/K



Hình 2.4 Chuỗi Markov của hàng đợi M/M/1

Với số khách hàng là k

$$P_n = \left(\frac{\lambda}{\mu}\right)^n \cdot P_0 \quad ; \quad 0 \leq n \leq k$$

$$P_n = \rho^n (1-\rho)(1-\rho^{k+1})$$

$$L = \frac{\rho}{1-\rho} - \frac{(k+1)\rho^{k+1}}{1-\rho^{k+1}}$$

Xác suất khách hàng đến hệ thống bị từ chối là P_k

Tốc độ thực tế đến hệ thống

$$\lambda' = (1 - P_K) \lambda$$

Mật độ lưu lượng

$$\rho' = \frac{\lambda'}{\mu} = (1 - P_K) \rho$$

2.1.3 Kỹ thuật hàng đợi

Các kỹ thuật hàng đợi được dùng phổ biến là:

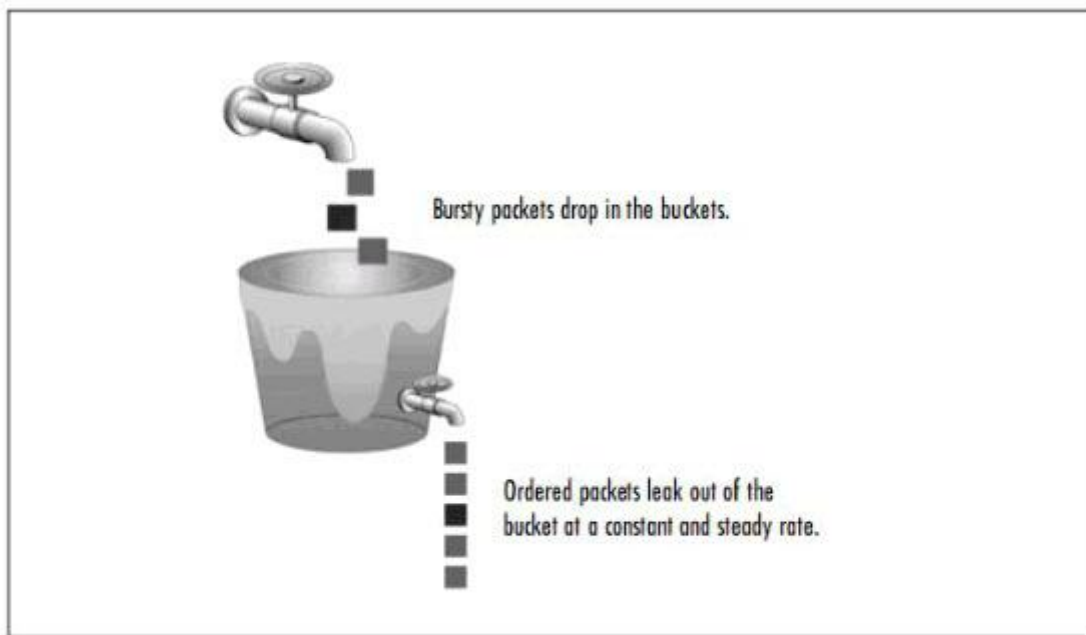
- First in First out Queuing
- Priority Queuing
- Custom Queuing
- Weighted Fair Queuing (WFQ)

Trong 4 kỹ thuật hàng đợi trên, Priority Queuing và Custom Queuing đòi hỏi nhà quản trị phải lên kế hoạch tính toán cẩn thận để thực thi và cấu hình chính xác trên router. Nhà quản trị phải có 1 sự am hiểu tốt về các dòng lưu lượng và cách để đánh dấu độ ưu tiên trên các dòng lưu lượng đó để có thể tạo ra các chiến lược hàng đợi hiệu quả. FIFO và WFO thì không cần cấu hình nhiều để có thể hoạt động 1 cách thích hợp.

Hàng đợi tồn tại trong bộ định tuyến để giữ gói tin cho đến khi có đủ tài nguyên để chuyển tiếp gói tin ra khỏi cổng ra. Nếu không có tắc nghẽn, gói tin sẽ lập tức được chuyển đi.

•Leaky bucket (Gáo rò)

Leaky bucket là khái niệm khóa cho việc hiểu về lý thuyết hàng đợi. Một hàng đợi mạng có thể được so sánh như một thùng mà những gói tin được đổ vào. Chiếc thùng này có một lỗ ở dưới đáy để gói tin có thể đi ra với một tốc độ liên tục. Trong môi trường mạng, tốc độ gói tin đi ra sẽ là tốc độ của card giao tiếp. Nếu tốc độ của gói tin đi vào lớn hơn tốc độ của gói tin đi ra thì thùng sẽ bị đầy và sẽ xảy ra hiện tượng rót gói, các gói tin đi vào sẽ bị đánh rơi.



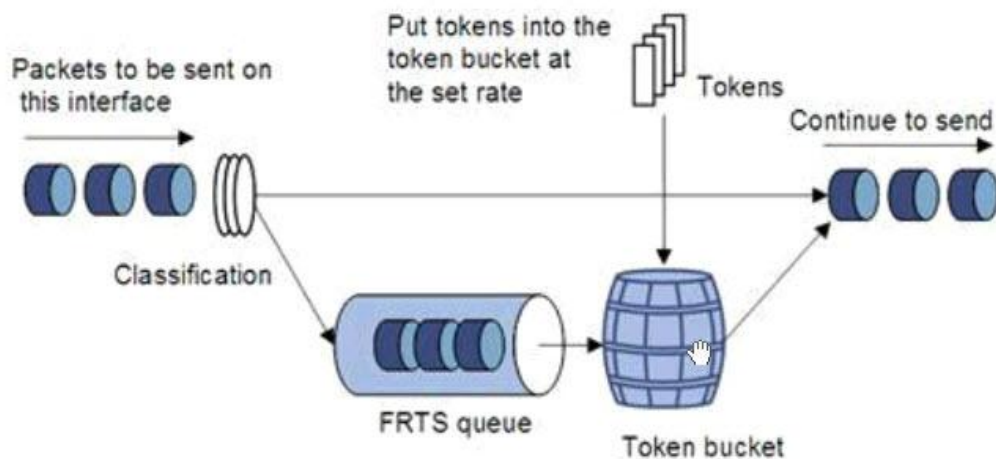
Hình 2.5 Leaky bucket

•**Tail Drop (cắt đuôi)**

Khi một lượng lớn các gói tin đi vào hàng đợi vượt quá dung lượng của hàng đợi thì sẽ xảy ra tắc nghẽn và các gói tin đi vào sau sẽ bị đánh rớt cho đến khi hết tắc nghẽn. Các giao thức ở tầng cao hơn sẽ dùng phương thức của nó để phát hiện và xử lý truyền lại các gói tin đã bị đánh rớt. Việc đánh rớt gói tin là việc xảy ra thường xuyên trong ạng vì tốc độ của gói tin đi vào 100Mbps Fast Ethernet) nhanh hơn tốc độ của gói tin đi ra khỏi bộ định tuyến.

•**Token bucket (thùng đựng thẻ)**

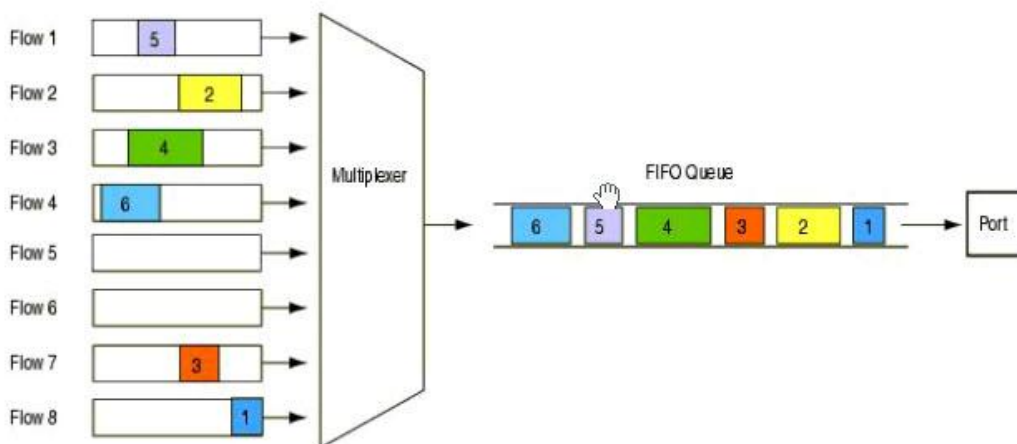
Token bucket là một thuật toán khác được dùng trong QoS, nó được mô tả như một hồ tài nguyên được dùng bởi một dịch vụ bất cứ khi nào cần thiết. Khác với Leaky bucket, gói tin đi vào sẽ được đi ra từ đỉnh chứ không phải từ đáy thùng leaky bucket. Dần dần thẻ sẽ được thêm vào thùng. Khi một ứng dụng muốn gửi dữ liệu ra ngoài mạng, nó sẽ lấy đi một lượng thẻ tương ứng với lượng dữ liệu mà nó muốn gửi. Nếu trong thùng không có đủ thẻ, ứng dụng sẽ phải đợi cho đến khi nào đủ thẻ mới có thể gửi dữ liệu.



Hình 2.6 Token Bucket

•First in First out Queue (Hàng đợi FIFO)

Hàng đợi FIFO là hàng đợi đơn giản nhất dựa trên khả năng lưu trữ và chuyển tiếp của bộ định tuyến, gói tin nào vào trước thì sẽ được ra trước. FIFO là thuật toán sắp xếp truyền thông mặc định trong một số trường hợp, vì vậy không cần phải cấu hình, nhưng nó có những hạn chế nhất định. Hạn chế lớn nhất của nó là không có sự phân lớp cho các gói tin, không có sự thỏa thuận về băng thông truyền. FIFO là thuật toán đầu tiên được thiết kế cho việc truyền thông trên mạng, nhưng đối với những mạng thông minh ngày nay đòi hỏi những thuật toán tốt hơn cũng như phức tạp hơn. FIFO được dùng chủ yếu trong mô hình cố gắng tối đa (best – effort).



Hình 2.7 Hàng đợi FIFO

Ưu điểm:

- Đây là kỹ thuật đơn giản và nhanh
- Được hỗ trợ trên tất cả các nền

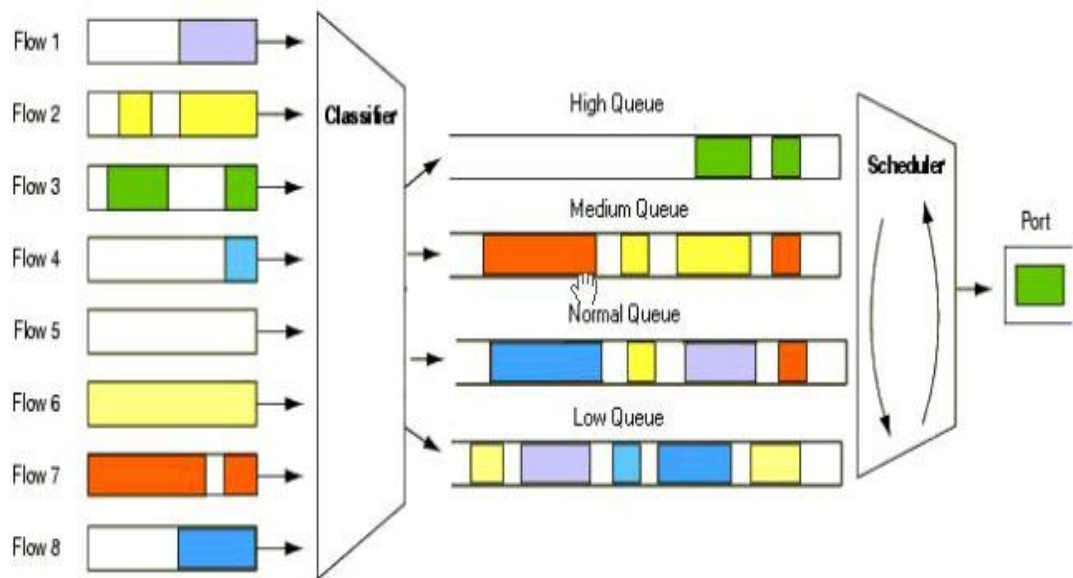
Nhược điểm:

- FIFO không hoàn toàn đáng tin cậy khi một luồng không mong muốn tranh giành với các luồng có độ ưu tiên thấp. Các luồng không mong muốn gửi một số lượng lớn các gói (đa số các gói đều bị hủy bỏ). Trong khi đó, các luồng với độ ưu tiên thấp gửi một số lượng gói xác định và hầu hết chúng bị hủy bởi vì hàng đợi lúc nào cũng đầy bởi các luồng không mong muốn đã chiếm hết không gian hàng đợi.

- Sự bùng nổ cao hay thấp gây ra tình trạng đầy hàng đợi FIFO. Các gói đi vào một hàng đợi đầy phải chờ một thời gian trước khi chúng được truyền. Nhưng ở thời điểm khác, hàng đợi có thể trống và các gói cùng một luồng có thể truyền không bị trì hoãn.

•Hàng đợi ưu tiên (Priority Queue)

Hàng đợi ưu tiên là một loại hình quản lý nghẽn chặt chẽ và mạnh mẽ, cho phép nhà quản trị định nghĩa đến 4 hàng đợi cho lưu lượng mạng, đó là High, Medium, Normal và Low. Bộ định tuyến xử lý những hàng đợi này một cách chặt chẽ dựa trên độ ưu tiên của chúng. Nếu có gói tin nằm trong hàng đợi High, hàng đợi đó sẽ được xử lý cho đến khi nó rỗng, độc lập về trạng thái của những hàng đợi khác. Một khi hàng đợi High rỗng, bộ định tuyến sẽ di chuyển đến hàng đợi Medium, nhưng ngay khi 1 gói tin từ hàng đợi medium được chuyển tiếp đi, bộ định tuyến sẽ lập tức quay trở lại hàng đợi High để kiểm tra rằng nó còn trống hay không. Cứ như vậy cho đến khi tất cả hàng đợi đều rỗng.

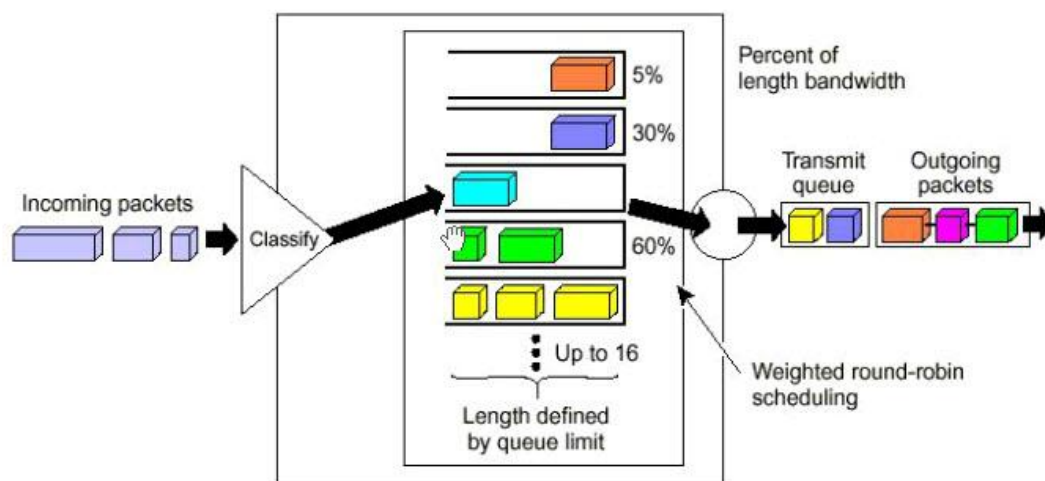


Hình 2.8 Hàng đợi ưu tiên

Hàng đợi ưu tiên cung cấp cho người quản trị khả năng đủ mạnh để có thể từ chối lưu lượng thuộc hàng đợi Low. Khi hàng đợi ưu tiên thuộc thứ hạng thấp không phục vụ vì có quá nhiều lưu lượng ở hàng đợi cao hơn thì sẽ xảy ra tình trạng đói độ ưu tiên (priority starvation).

•Hàng đợi tùy chọn (Custom Queue)

Để khắc phục sự cứng nhắc của hàng đợi ưu tiên, nhà quản trị có thể dùng hàng đợi tùy chọn để thay thế. Hàng đợi tùy chọn cho phép nhà quản trị đánh dấu ưu tiên cho lưu lượng mà không gây ra việc thiếu hàng đợi thấp như ở hàng đợi ưu tiên. Với hàng đợi tùy chọn, nhà quản trị có thể tạo ra 16 hàng đợi cho các lưu lượng ưu tiên, bộ lập lịch của hàng đợi tùy chọn không có tùy chọn để phục vụ một hàng đợi ưu tiên- như hàng đợi ưu tiên cao – vì thế hàng đợi tùy chọn không cung cấp dịch vụ với độ trì hoãn lớn và lưu lượng nhạy với sự thay đổi. Bộ lập lịch hàng đợi tùy chọn lấy các gói từ hàng đợi cho đến khi tổng số byte chỉ định cho mỗi hàng đợi vượt ngưỡng. Sau khi hàng đợi phục vụ nhiều byte hay hàng đợi không có gói nào, hàng đợi tùy chọn di chuyển đến hàng đợi tiếp theo và lặp lại tiến trình.



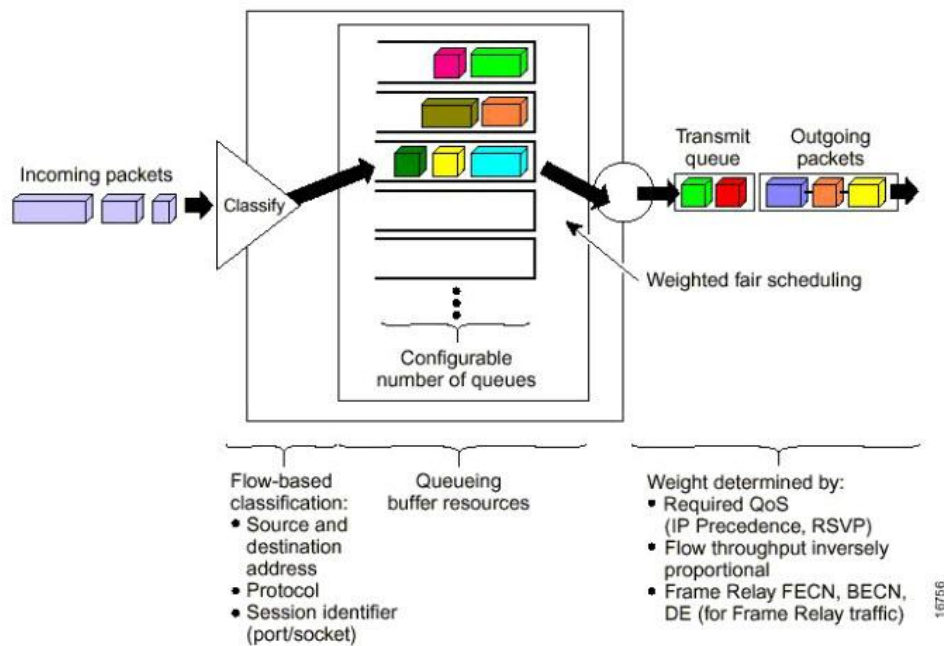
Hình 2.9 Custom Queue

Router thực hiện việc đưa gói tin vào hàng đợi dựa vào thuật toán round-robin, sắp xếp lại từng byte cấu hình được đếm từ mỗi hàng đợi trong từng chu kỳ. Tính năng này đảm bảo rằng không có ứng dụng nào (hay nhóm ứng dụng cụ thể nào) có thể hoạt động cao hơn khả năng mà hệ thống đã cấp phát cho chúng khi đường truyền đang ở tình trạng quá tải. Giống như Priority Queue, Custom Queue được cấu hình tĩnh và không tự động cập nhật lại các thay đổi mạng.

•Hàng đợi cân bằng có trọng số (Weighted Fair Queue):

WFQ là chiến lược hàng đợi mặc định cho các card giao tiếp có tốc độ thấp, WFQ là phương pháp cấp phát băng thông đều cho tất cả các lưu lượng. WFQ sắp xếp các lưu lượng mạng vào những dòng tạo nên cuộc hội thoại trên mạng bằng cách dùng sự kết hợp của các tham số. Các cuộc hội thoại TCP/IP được xác định dựa vào tham số sau:

- Giao thức IP
- Giao thức IP nguồn
- Giao thức IP đích
- Cổng IP nguồn
- Cổng IP đích
- Trường ToS (Type of Services).

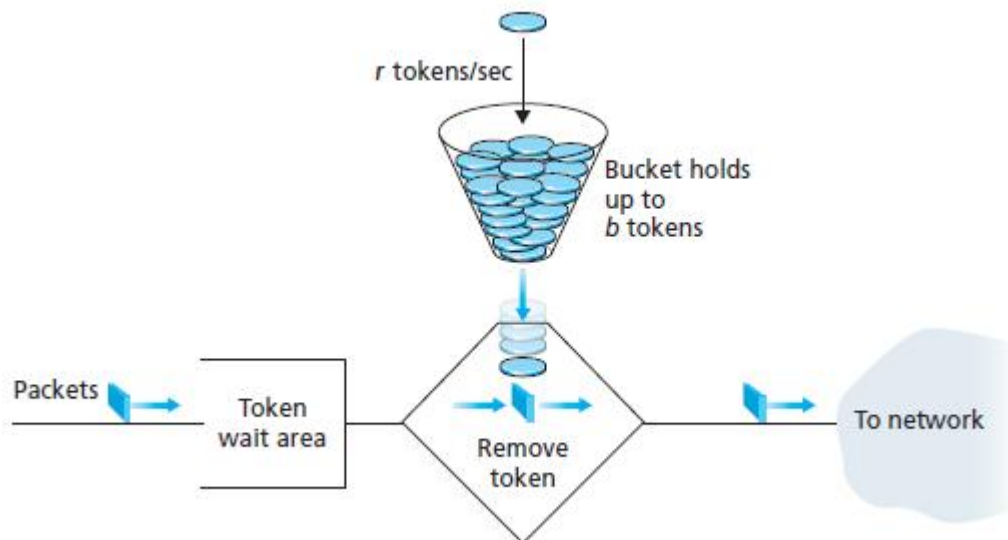


Hình 2.10 Weighted Fair Queue

Qua việc theo dõi các dòng dữ liệu, router có thể biết được dòng nào cần nhiều băng thông như FTP hoặc dòng nào nhạy cảm với độ trễ Telnet. Có 256 hàng đợi mặc định được dùng khi sử dụng WFQ.

2.2 Điều khiển lưu lượng theo thuật toán gáo rò (leaky bucket)

2.2.1 Nguyên lý của thuật toán gáo rò



Hình 2.11 Mô hình gáo rò

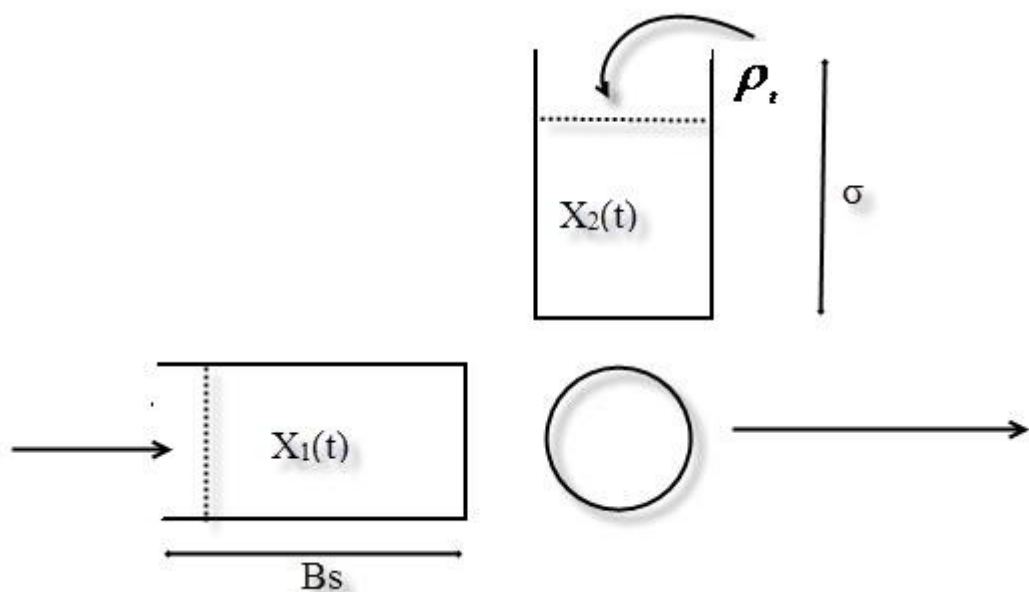
Trong mô hình này, nút mạng được trang bị một gáo rò dùng kiểm soát lưu lượng thông tin đi vào mạng. Gáo là một bộ đệm có khả năng lưu trữ tối đa là b thẻ bài. Các thẻ bài được đưa vào gáo với tốc độ r thẻ bài/s. Khi gáo đã đầy thẻ bài thì thẻ bài sẽ không được điền thêm vào gáo.

Mỗi khi một gói tin đến và để có thể được vào được mạng thì gói tin đó phải nhận được một thẻ bài. Tốc độ trung bình của thông tin vào mạng là r gói tin/s và bằng tốc độ điền thẻ bài vào gáo.

Trong trường hợp gáo rò đầy thẻ bài, nút mạng có thể cho tối đa b gói tin vào mạng tại một thời điểm (burst size). Nếu r nhỏ thì khả năng kiểm soát tốc độ luồng thông tin vào là tốt, nếu r lớn thì khả năng hỗ trợ burst tốt. [3]

2.2.2 Mô hình giải tích

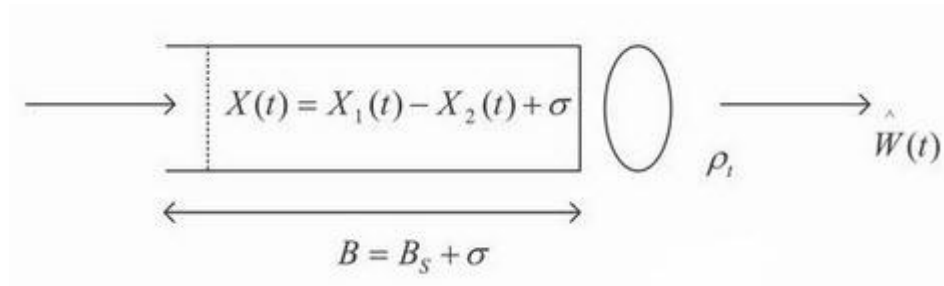
Trên nguyên lý hoạt động của thuật toán cái gáo rò, chúng ta vẽ lại với các ký hiệu toán học như Hình 2.12.



Hình 2.12 Mô hình gáo rò bằng kí hiệu toán học

Xét mô hình điều khiển lưu lượng dạng gáo rò như Hình 2.12: Hàng đợi dữ liệu có kích thước tối đa là B_s , thẻ được phát ra với tốc độ không đổi ρ_t . Lượng dữ liệu và lượng thẻ có trong bộ đệm dữ liệu và bộ đệm thẻ tại thời điểm t tương ứng là $X_1(t)$ và $X_2(t)$.

Mô hình điều khiển lưu lượng dạng gáo rò có thể được chuyển đổi thành mô hình hàng đợi dạng M/D/1/B như Hình 2.13.



Hình 2.13 Mô hình chuyển đổi sang hàng đợi

Như vậy mô hình cái gáo rò được chuyển sang mô hình hàng đợi có quá trình sinh tử mô tả tương đồng như Hình 2.2. Đây là trường hợp có bộ đệm có kích thước hữu hạn ($B_s = K$) [6] [7]

Lưu lượng của hệ thống được tính theo công thức []:

$$T = P_o \left(\sum_{i=0}^{M+K} i a_i + (M+K) \bar{a}_{M+K} \right) + \sum_{j=1}^{M+K} i a_i \left(\sum_{i=1}^{M+K-j+1} i a_i + (M+K-j+1) \bar{a}_{M+K-j+1} \right)$$

Với:

$$\bar{a}_j = 1 - \sum_{i=0}^j a_i$$

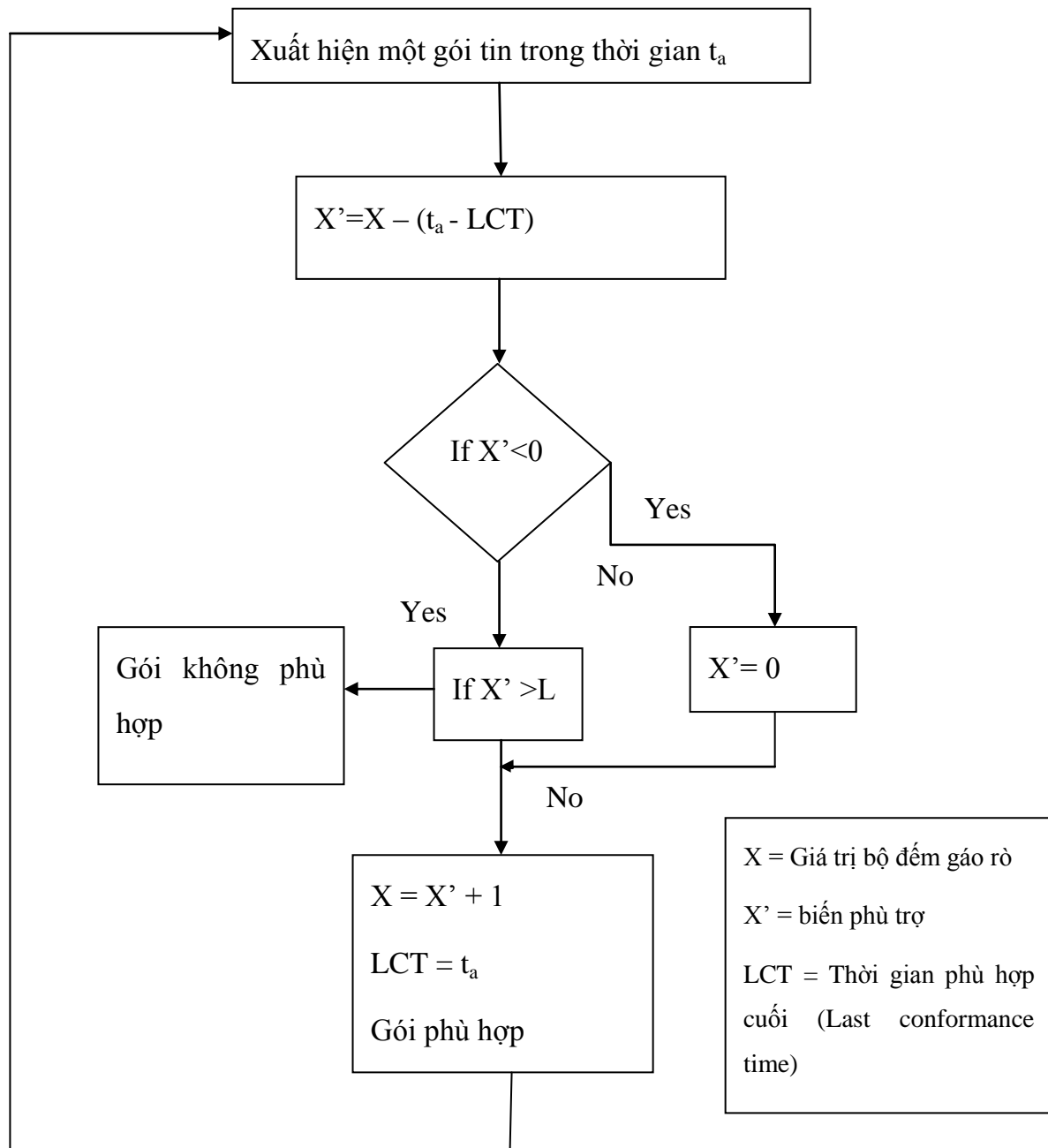
Xác suất một gói tin bị mất là:

$$P_{loss} = 1 - \frac{T}{\lambda D}$$

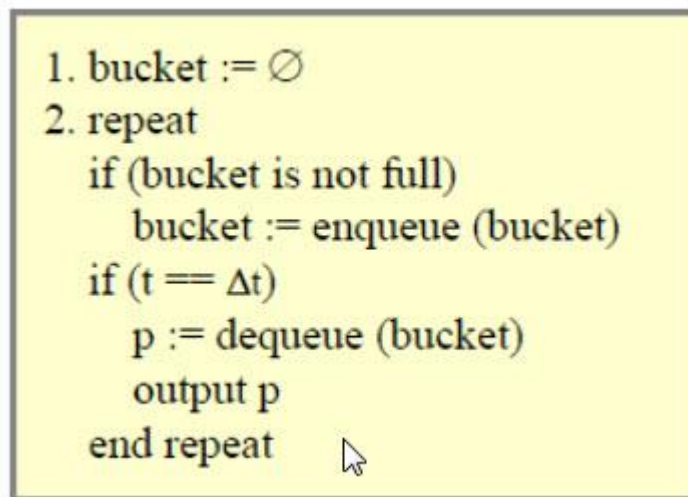
2.2.3 Thuật toán gáo rò (Leaky bucket)

❖ Mô tả thuật toán : [6] [7]

- Thuật toán gáo rò dùng để quản lý lưu lượng
- Khởi tạo với giá trị gáo rỗng (bucket = \emptyset)
- Khi content đi vào gáo và gáo chưa đầy thì xếp content vào hàng đợi, sau một khoảng thời gian t (do người dùng thiết lập) thì content sẽ được chuyển ra khỏi gáo. Việc này được lặp lại
- Khi gáo đầy (bucket = full) thì có thể xảy ra mất mát dữ liệu



Hình 2.14 Lưu đồ thuật toán gáo rò



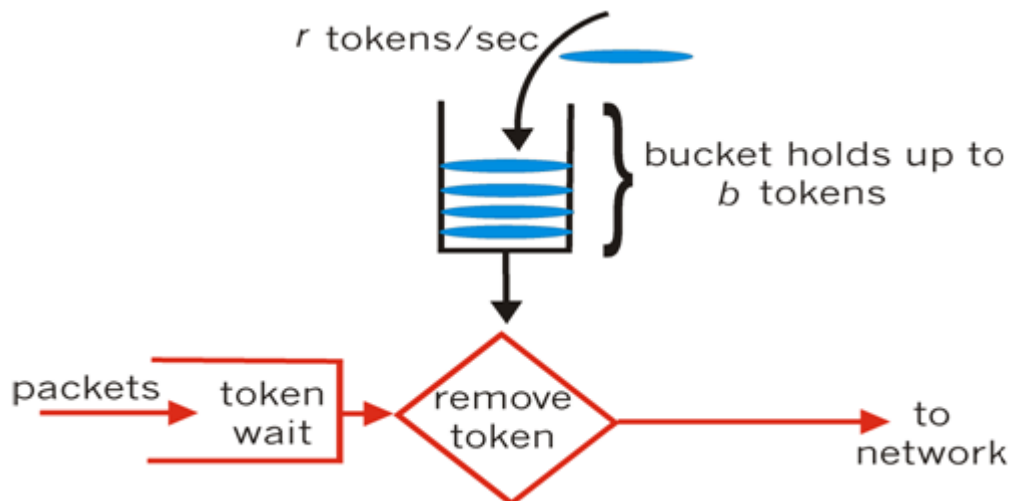
Hình 2.15 Thuật toán gáo rò

2.2.4 Thuật toán gáo rò trong điều khiển lưu lượng

Giới hạn lưu lượng đưa vào mạng

Với các thông số như Hình 2.16, chúng ta có nhận thấy lưu lượng đưa vào mạng được giới hạn bởi công thức : $\leq r \times t + b$

Trong đó r là tốc độ thẻ đưa vào gáo và b là độ lớn của gáo. Khi gáo rỗng thì tốc độ dữ liệu đưa vào mạng bằng tốc độ thẻ đưa vào gáo. Chùm tin (Burst) lớn nhất được đưa vào mạng là b

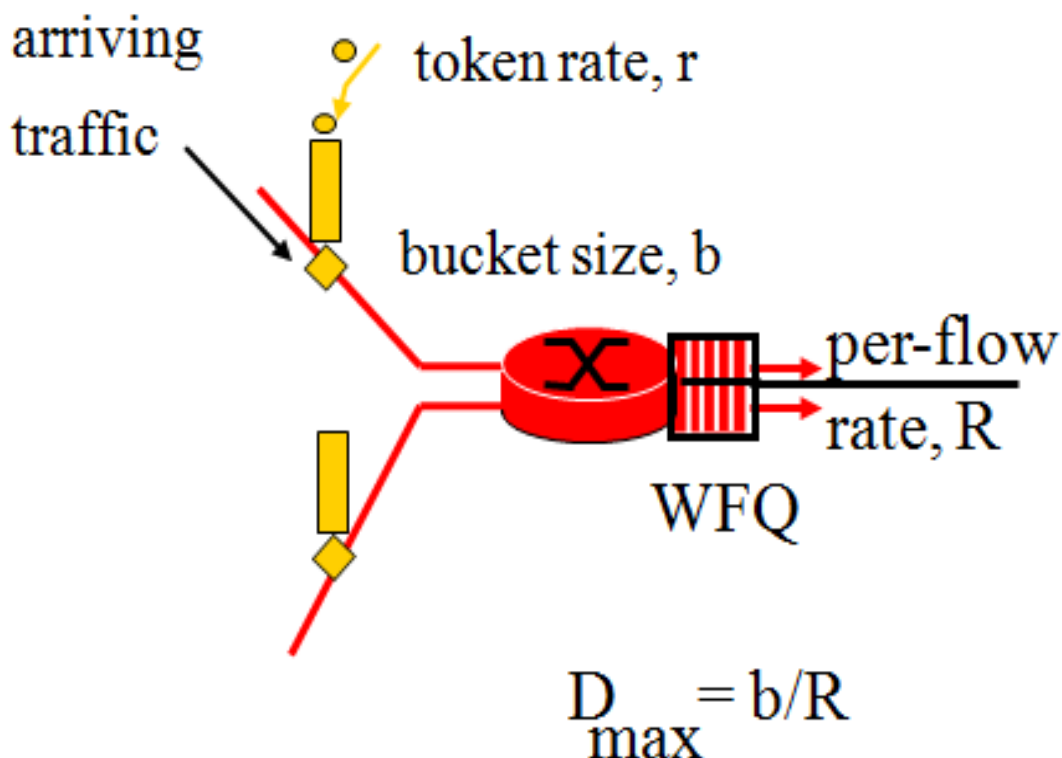


Hình 2.16 Điều khiển lưu lượng đưa vào mạng bằng thuật toán cái gáo rò

Giới hạn trễ tối đa của gói tin

Kết hợp thuật toán cái gáo rò và thuật toán lập lịch xoay vòng có trọng số, chúng ta có thể giới hạn trễ tối đa gói tin phải chờ trước khi được đưa vào mạng.

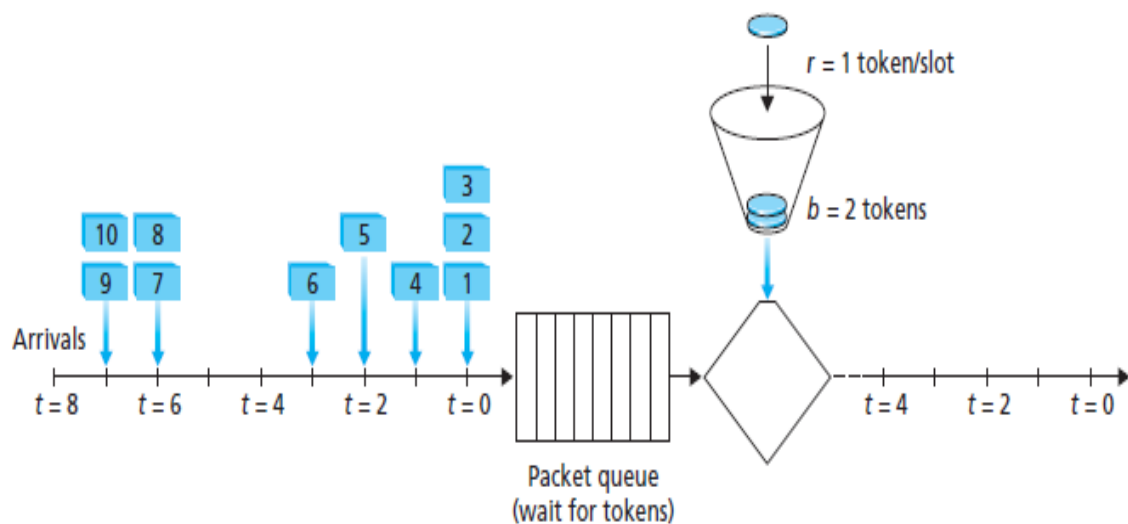
Hình 2.17 minh họa cơ chế tính trễ D_{\max}



Hình 2.17 Sử dụng thuật toán cái gáo rò để giới hạn trễ tối đa

Chức năng định dạng lưu lượng

Với việc sử dụng gáo rò, luồng thông tin vào mạng có tốc độ không vượt quá r gói/s. Nếu mạng có nhiều nút mạng để giao tiếp với bên ngoài (entry point), mỗi nút mạng được trang bị một gáo rò để kiểm soát lưu lượng thông tin vào mạng thì cho dù tốc độ thông tin đến các nút có thể thay đổi, nhưng tốc độ thông tin trong mạng khá ổn định. Với đặc điểm này, người ta nói gáo rò thực hiện chức năng định dạng lưu lượng. Để hiểu cơ chế định dạng lưu lượng của thuật toán chúng ta minh họa qua một ví dụ với các giá trị số cụ thể như trên sơ đồ Hình 2.11.



Hình 2.18 Ví dụ chức năng định dạng lưu lượng của thuật toán gáo rò

Trong ví dụ trên Hình 2.18, luồng các gói tin đi vào gáo rò trong 9 thời điểm (tại đầu các khe thời gian), chúng ta theo dõi số gói tin trong bộ đệm (Packet queue), số thẻ trong gáo và số gói tin tại cổng ra trong mỗi khe thời gian. Để dễ theo dõi, chúng ta giả thiết các giá trị tốc độ thẻ $r = 1$ thẻ/mỗi khe thời gian và số thẻ tối đa trong gáo là $b = 2$. Tại $t = 0$ gáo đầy. Tốc độ ra có thể đạt 2 gói tin/mỗi khe thời gian. Các chi tiết về thời gian của hệ thống như sau:

- Các gói tin đến tại đầu của khe thời gian. Như vậy trong hình, các gói 1, 2, và 3 đến trong khe thời gian $t = 0$. Nếu có gói tin trong hàng đợi, các gói tin mới đến được xếp vào cuối hàng đợi. Cơ chế lập lịch chuyển tiếp gói tin từ hàng đợi đến cổng ra là FIFO.

- Sau khi đến hàng đợi, một hoặc hai trong số những gói dữ liệu sẽ lấy thẻ và đi đến cổng đầu ra trong khe thời gian. Như vậy, các gói 1 và 2 sẽ lấy thẻ và đi đến cổng ra trong khe thời gian 0.

- Nếu gáo không đầy thì một thẻ mới được thêm vào gáo vì tốc độ thẻ là $r = 1$ thẻ/mỗi khe thời gian.

- Các bước trên được lặp lại cho các khe thời gian tiếp theo.

Trên cơ sở các thông số đã cho trên, chúng ta cần:

- Xác định các gói tin có trong hàng đợi và số lượng thẻ trong gáo, ngay sau khi đến và trước khi chuyển tiếp cho mỗi khe thời gian,.

- Xác định các gói dữ liệu xuất hiện ở cổng ra cho mỗi khe thời gian..

Khe thời gian	Gói tin trong hàng đợi	Số thẻ trong gáo	Gói tin tại cổng ra
0	1, 2, 3	2	1, 2
1	3, 4	1	3
2	4,5	1	4
3	5,6	1	5
4	6	1	6
5	-	1	-
6	7, 8	2	7, 8
7	9, 10	1	9
8	10	1	10

Bảng 2.1: Kết quả chức năng định dạng luồng thông tin của thuật toán gáo rò

Kết quả thống kê số gói tin tại các khe thời gian cho thấy: Số gói tin được đưa vào mạng tại các khe thời gian đã thay đổi, kích thước tối đa của chùm gói tin đã được hạn chế, nói cách khác luồng các gói tin đưa vào gáo đã được định dạng lại và đưa vào mạng.

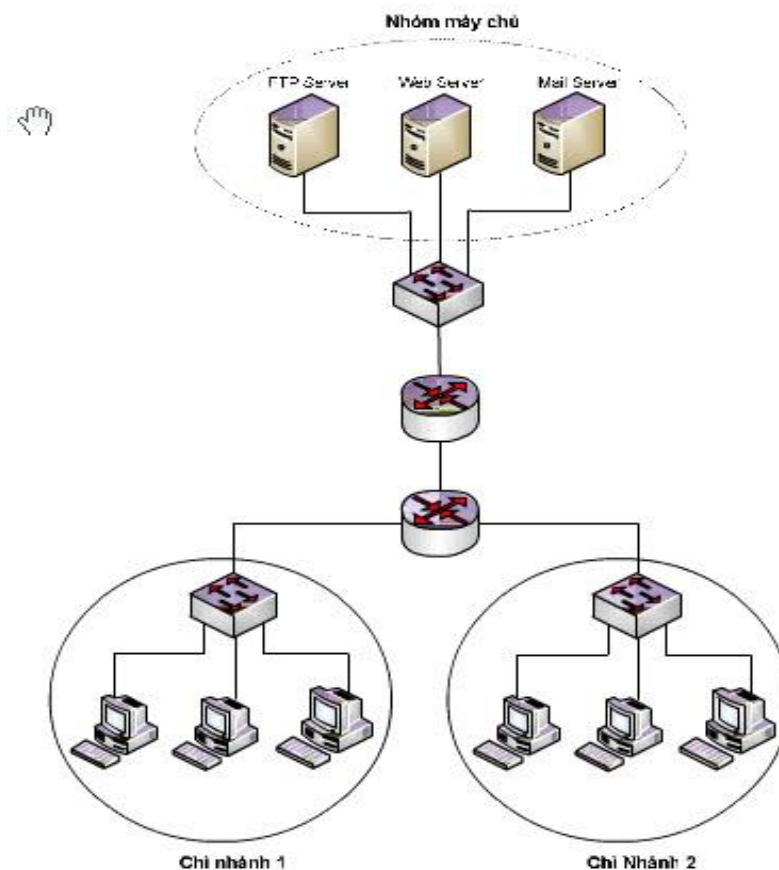
Chương 2 đã trình bày nguyên lý của thuật toán gáo rò, mô hình toán học, giải mã của thuật toán. Chúng ta cũng đã thấy khả năng điều khiển lưu lượng của các nút mạng có cài đặt thuật toán gáo rò. Chương tiếp theo sẽ xây dựng cài đặt thuật toán và ứng dụng trong việc giám sát và kiểm soát lưu lượng mạng trong quá trình khởi động máy tính cụm nhằm nâng cao hiệu quả của hệ thống.

CHƯƠNG 3: CHƯƠNG TRÌNH THỰC NGHIỆM

3.1 Nhiệm vụ của luận văn

3.1.1 Bài toán đặt ra

Hệ thống mô tả mô hình mạng của công ty cổ phần Sách giáo dục EDC. Công ty này có hai chi nhánh tại Hà Nội và t.p Hồ Chí Minh. Mỗi chi nhánh có một mạng nội bộ và bao gồm rất nhiều phòng ban như : phòng nội dung, phòng công nghệ, phòng kinh doanh, phòng sản xuất... Riêng trụ sở chính đặt tại Hà Nội sẽ có hệ thống máy chủ đó là : Web Server, FTP và Mail Server. Mô hình mạng của công ty như sau :



Hình 3.1 Mô hình mạng của công ty cổ phần Sách giáo dục điện tử EDC

Hàng ngày sẽ có rất nhiều lưu lượng khác nhau (Web, FTP, Mail...) giữa hai chi nhánh với hệ thống máy chủ.

Vấn đề thứ nhất xảy ra đó là có những thời điểm một số máy trạm truy xuất rất chậm, một số máy khác truy xuất bình thường. Lý do chính là máy nào

truy xuất dữ liệu trước và truy xuất dữ liệu có dung lượng lớn sẽ chiếm hết băng thông của những máy trạm truy xuất sau.

Vấn đề thứ hai xảy ra là một lý do khách quan máy chủ hỏng phần cứng, không thể truy cập được hệ thống, công việc của nhân viên trong công ty bị gián đoạn.

Từ hai vấn đề đó, giải pháp được đưa ra là tạo ra 2 máy chủ webserver có cùng chức năng nếu một máy chủ nào đó chẳng may bị hỏng hóc tự động máy chủ còn lại sẽ đóng vai trò thay thế. Bên cạnh đó điều khiển lưu lượng (quản lý giới hạn băng thông cho từng loại dịch vụ) trên hệ thống, áp dụng thuật toán gáo rò cho lượng băng thông truy cập là đều nhau trên mỗi đơn vị thời gian.

3.1.2 Mô hình hệ thống

Từ nhu cầu của công ty cổ phần Sách giáo dục điện tử EDC, luận văn cần triển khai như sau :

- Xây dựng một web Cluster
- Điều khiển lưu lượng sử dụng thuật toán Gáo rò

Vì mô hình mạng của công ty cổ phần Sách giáo dục điện tử EDC gồm rất nhiều máy tính, do vậy để thuận tiện trong việc demo, luận văn cài đặt Web cluster trên máy ảo vmware (gồm 2 máy server sử dụng hệ điều hành linux – centos).

Mô hình thực nghiệm có hơi khác so với mô hình mạng thực tế từ công ty cổ phần Sách điện tử giáo dục EDC :

- Mô hình thực nghiệm chỉ xây dựng Webserver và FTP Server (Không xây dựng mô hình Mail Server) trên máy chủ.
- Băng thông được điều khiển sẽ nhỏ hơn so với nhu cầu thực tế, do mô hình thực nghiệm được demo qua hệ thống máy ảo.

3.2 Xây dựng một Web Cluster

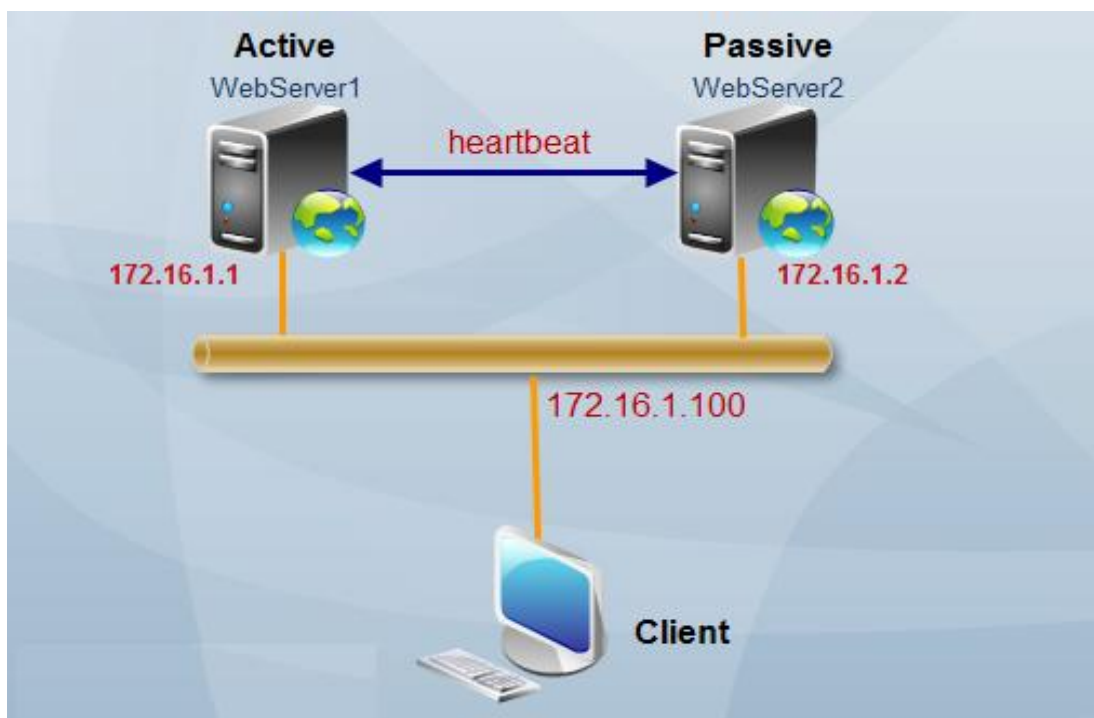
3.2.1 Mô hình

Chúng ta có giải pháp tạo ra 2 máy chủ webserver cùng chức năng nếu một máy chủ nào đó chẳng may bị hỏng hóc tự động máy chủ còn lại sẽ đóng vai trò thay thế. Đó chính là giải pháp Cluster Apache High Availability.

Cluster sửa chữa lỗi (failover cluster) được dùng để đảm bảo tính sẵn sàng cho các dịch vụ và ứng dụng hệ thống khi bị tấn công, xử lý các lỗi phần cứng và rủi ro do môi trường. Trong luận văn này, tôi sẽ xây dựng một cluster Apache hai nút, chắc chắn tin cậy và hiệu quả cao với ứng dụng của dự án The High-Availability Linux.

Trong môi trường cluster, hệ thống 'có tính sẵn sàng cao' (high ability - HA) chịu trách nhiệm bắt đầu và kết thúc các dịch vụ, cài đặt và gỡ bỏ tài nguyên, giám sát khả năng sẵn sàng của hệ thống trong môi trường cluster và điều khiển quyền sở hữu địa chỉ IP ảo chia sẻ giữa các nút cluster. Dịch vụ heartbeat (trung tâm) cung cấp các tính năng cơ sở cần thiết cho hệ thống HA.

Cấu hình cluster phổ biến nhất là standby, sẽ được mô tả dưới đây. Trong cấu hình cluster này, một nút thực hiện tất cả các việc, còn các nút khác ở trạng thái nghỉ ngơi. Heartbeat giám sát "sức khỏe" của từng dịch vụ cụ thể, thông thường qua một giao diện Ethernet phân tách vốn chỉ dùng cho hệ thống HA sử dụng câu lệnh đặc biệt ping. Nếu vì một lý do nào đó, nút đang thực hiện bị hỏng, heartbeat sẽ chuyển tất cả thành phần HA sang nút khỏe mạnh khác. Khi nút cũ phục hồi, nó có thể khôi phục lại tình trạng cũ trước đó của mình.



Hình 3.2 Mô hình Web Cluster

3.2.2 Cài đặt hai server chạy hệ điều hành Linux trên VMware Workstation

Phần mềm :

❖ **Vmware workstation 10.0**

Link tải: <https://my.vmware.com/web/vmware/downloads>

❖ **Linux Centos 6.6 32 bit**

Link tải : <http://isoredirect.centos.org/centos/6/isos/i386/>

Cài đặt

Bước 1 : Cài đặt vmware workstation 10.0 (xem chi tiết ở [9])

Bước 2 : Cài đặt 2 máy linux Centos trên vmware đặt tên máy lần lượt là Node01 và Node 2 (Xem chi tiết ở [8])

Bước 3 : Cấu hình card mạng cho Centos (làm trên cả 2 node)

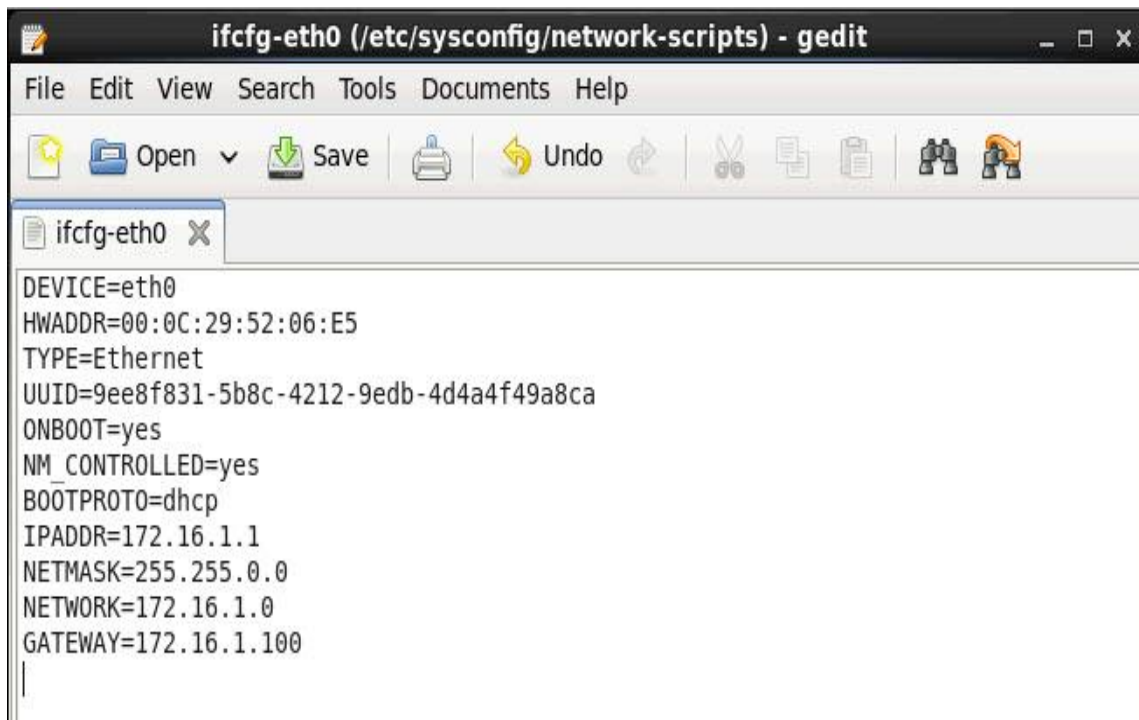
(1) Kiểm tra card mạng

ifconfig -a | grep eth

(2) Cấu hình card eth0

vi /etc/sysconfig/network-scripts/ifcfg-eth0

(3) Thay đổi nội dung file như hình 3.3



Hình 3.3a Nội dung file ifcfg-eth0 cho máy Node 1



Hình 3.3b Nội dung file ifcfg-eth0 cho máy Node 2

(4) Cần khởi động lại máy hoặc sử dụng lệnh sau để các khai báo trên file ifcfg-eth0 được áp dụng :

/etc/init.d/networking restart

(5) Kiểm tra địa chỉ Ip và routing đã được hay chưa

Sử dụng lệnh #route -n

(6) Khai báo DNS cho một server centos

Sử dụng lệnh : # vi /etc/resolv.conf

Nội dung trong file:

nameserver 8.8.8.8

nameserver 8.8.4.4

(7) Khởi động lại hai node

3.2.3 Cài đặt Apache và PHP trên Linux CentOS

(1) Cài đặt Apache

Sử dụng lệnh để tải gói và cài đặt :

yum install httpd -y

Sau đó khởi động Apache

service httpd start

Test thử quá trình cài đặt đã thành công hay chưa. Vào trình duyệt gõ <http://localhost>

Trên CentOS, thư mục lưu trữ web mặc định của Apache được lưu ở đường dẫn /var/www/html và tệp tin cấu hình được lưu tại /etc/httpd/conf/httpd.conf. Các file cấu hình khác được lưu tại thư mục /etc/httpd/conf.d/

(2) Cài đặt PHP

Cài đặt php bằng lệnh :

```
# yum install php
```

Sau đó khởi động lại Apache

```
# service httpd restart
```

3.2.4 Cài đặt và cấu hình heartbeat trên các Node

Trên cả hai node ta cài đặt các gói heartbeat. Heartbeat cơ bản là một phần của Linux, là hệ thống High Availability nó sẽ đảm bảo tính sẵn sàng cho hệ thống.

3.2.4.1 Cài đặt

Bước 1 : Tạo thư mục download bên trong root

```
# mkdir /download
```

```
# cd /download/
```

Bước 2 : Tải EPEL repository

```
# wget http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
```

Bước 3 : Cài đặt Epel RPM

```
# rpm -ivh epel-release-6-8.noarch.rpm
```

Bước 4 : Chỉnh sửa tệp tin epel.repo '/' etc / yum.repos.d / epel.repo' thay đổi dòng thứ 6: 'enabled = 1 thành enable = 0'.

```
# vi /etc/yum.repos.d/epel.repo
```

Bước 5 : Cài đặt Heartbeat với lệnh Yum

```
#yum --enablerepo=epel install heartbeat
```

Kết quả hiển thị : xem thêm ở phần Phụ lục 1

3.2.4.2 Cấu hình Heartbeat

Sau khi cài đặt thành công Heartbeat, ta thực hiện cấu hình Heartbeat trên hai node.

Bước 1 : Sao chép các tệp tin cấu hình authkeys, ha.cf, haresources vào thư mục /etc/ha.d

```
# cp /usr/share/doc/heartbeat-2.1.3/authkeys /etc/ha.d
```

```
# cp /usr/share/doc/heartbeat-2.1.3/ha.cf /etc/ha.d
```

```
# cp /usr/share/doc/heartbeat-2.1.3/haresources /etc/ha.d
```

Bước 2 : Thực hiện mở file authkeys :

```
# vi /etc/ha.d/authkeys
```

Thêm vào 2 dòng sau:

```
auth 1
```

```
1 sha1 test-ha
```

Bước 3 : Thực hiện mở file ha.cf:

```
# vi /etc/ha.d/ha.cf
```

Thêm vào các dòng sau:

logfile /var/log/ha-log	#file log HA
logfacility local0	#tieu ich dung cho syslog hoac logger
keepalive 2	#thoi gian kiem tra giua cac heartbeat
deadtime 30	#thoi gian quyet dinh mot host da die hay chua?
initdead 120	#thoi gian chet dau tien
bcast eth1	#card mang de gui tin hieu heartbeat
udpport 694	#Port de gui tin hieu heartbeat
auto_failback on	#tu dong gui tai nguyen sai tro lai node chinh
node node1.cluster.com	#ten node 1
node node2.cluster.com	#ten node 2

Bước 4 : Mở file haresources

vi /etc/ha.d/haresources

Thêm vào cuối file dòng sau:

node1.cluster.com 172.16.1.100 httpd

3.2.4.3 Cấu hình dịch vụ httpd

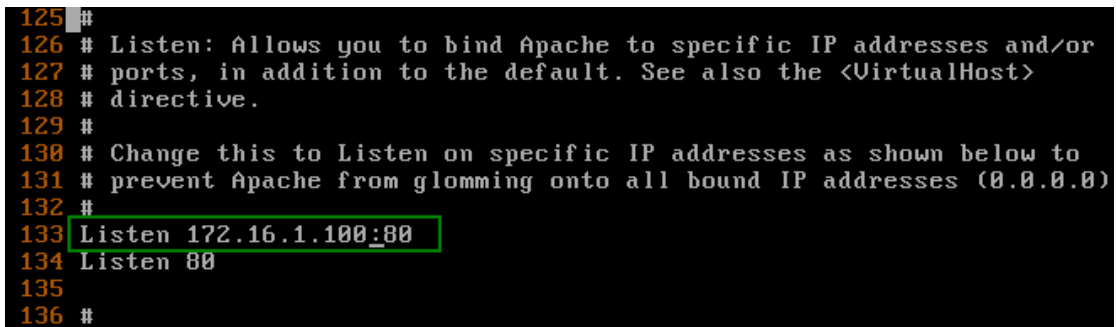
Trên cả hai node, mở file

vi /etc/httpd/conf/httpd.conf

Tại dòng 133 cấu hình như sau:

Listen 172.16.1.100:80

Cấu hình này sẽ “bảo” dịch vụ httpd lắng nghe trên địa chỉ IP ảo của mô hình Cluster, thay vì lắng nghe trên IP của node đó.



```
125 #
126 # Listen: Allows you to bind Apache to specific IP addresses and/or
127 # ports, in addition to the default. See also the <VirtualHost>
128 # directive.
129 #
130 # Change this to Listen on specific IP addresses as shown below to
131 # prevent Apache from glomming onto all bound IP addresses (0.0.0.0)
132 #
133 Listen 172.16.1.100:80
134 Listen 80
135
136 #
```

Hình 3.4 Cấu hình httpd

Sau khi cấu hình xong, ta khởi động dịch vụ heartbeat lên, và thiết đặt tự động bật mỗi khi hệ thống khởi động:

service heartbeat start

chkconfig heartbeat on

Đồng thời, ta cũng tắt chế độ tự khởi động của dịch vụ httpd đi, vì khi dịch vụ heartbeat được khởi động thì nó cũng yêu cầu dịch vụ httpd khởi động theo rồi!

chkconfig httpd off

3.3 Chương trình thuật toán gáo rò

Dựa vào thuật toán gáo rò được mô tả trong 2.4.3 và 2.4.4 trong tài liệu, từ đó triển khai thuật toán trên ngôn ngữ Java.

Dưới đây là chương trình code thuật toán gáo rò trong java để điều khiển lưu lượng

```
// Leaky Bucket Implementation

import java.util.*;

public class leaky
{
    public static void main(String[] args)
    {
        Scanner my = new Scanner(System.in);

        int no_groups,bucket_size;

        System.out.print("\n Enter the bucket size : \t");

        bucket_size = my.nextInt();

        System.out.print("\n Enter the no of groups : \t");

        no_groups = my.nextInt();

        int no_packets[] = new int[no_groups];

        int in_bw[] = new int[no_groups];

        int out_bw,reqd_bw=0,tot_packets=0;

        for(int i=0;i<no_groups;i++)
        {
            System.out.print("\n Enter the no of packets for group " + (i+1) + "\t");

            no_packets[i] = my.nextInt();

            System.out.print("\n Enter the input bandwidth for the group " + (i+1) +
"\t");

            in_bw[i] = my.nextInt();
        }
    }
}
```

```

        if((tot_packets+no_packets[i])<=bucket_size)
        {
            tot_packets += no_packets[i];
        }
        else
        {
            do
            {
                System.out.println(" Bucket Overflow ");
                System.out.println("  Enter  value  less  than  "  +  (bucket_size-
tot_packets));
                no_packets[i] = my.nextInt();
            }while((tot_packets+no_packets[i])>bucket_size);
            tot_packets += no_packets[i];
        }
        reqd_bw += (no_packets[i]*in_bw[i]);
    }

    System.out.println("\nThe total required bandwidth is " + reqd_bw);
    System.out.println("Enter the output bandwidth ");
    out_bw = my.nextInt();
    int temp=reqd_bw;
    int rem_pkts = tot_packets;
    while((out_bw<=temp)&&(rem_pkts>0))

```

```

{
    System.out.println("Data Sent \n" + (--rem_pkts) + " packets remaining");
    System.out.println("Remaining Bandwidth " + (temp -= out_bw));
    if((out_bw>temp)&&(rem_pkts>0))
        System.out.println(rem_pkts + " packet(s) discarded due to insufficient
bandwidth");
    }
}
}
}

```

3.4 Một số kịch bản thử nghiệm

Kịch bản 1 : Kiểm tra việc truy cập hệ thống khi 1 node trong cluster bị lỗi

Thử nghiệm 1 :

Lần lượt tạo 2 file index.html trên 2 node với nội dung khác nhau :

Trên node1 : # echo "<h1>Day la node1 "> /var/www/html/index.html

Trên node2 : # echo "<h1>Day la node1 "> /var/www/html/index.html

Từ phía máy Client, mở trình duyệt lên và truy cập vào địa chỉ IP ảo 172.16.1.100 ta sẽ thấy nội dung web là “Day la node1 ”. Bởi vì Node 1 đang là Node chính nên nó được ưu tiên phản hồi cho Client

Tiếp theo, thực hiện tắt dịch vụ heartbeat trên Node1 đi :

```
# service heartbeat stop
```

Sau đó từ client ta tiến hành truy cập địa chỉ Ip ảo 172.16.1.100.

⇒ Kết quả : Website vẫn truy cập bình thường, nội dung được hiển thị là “Day la node2 ” – chính là nội dung từ Webserver trên Node2.

Trong thử nghiệm này, để mô tả việc đáp ứng của Cluster nên tôi để nội dung của hai node khác nhau. Trên thực tế, các nút trên Cluster sẽ được đồng bộ dữ liệu để người dùng luôn nhìn thấy một và chỉ một nội dung mà thôi.

Kịch bản 2 : Điều khiển lưu lượng bằng thuật toán gáo rò

Thực nghiệm 2 :

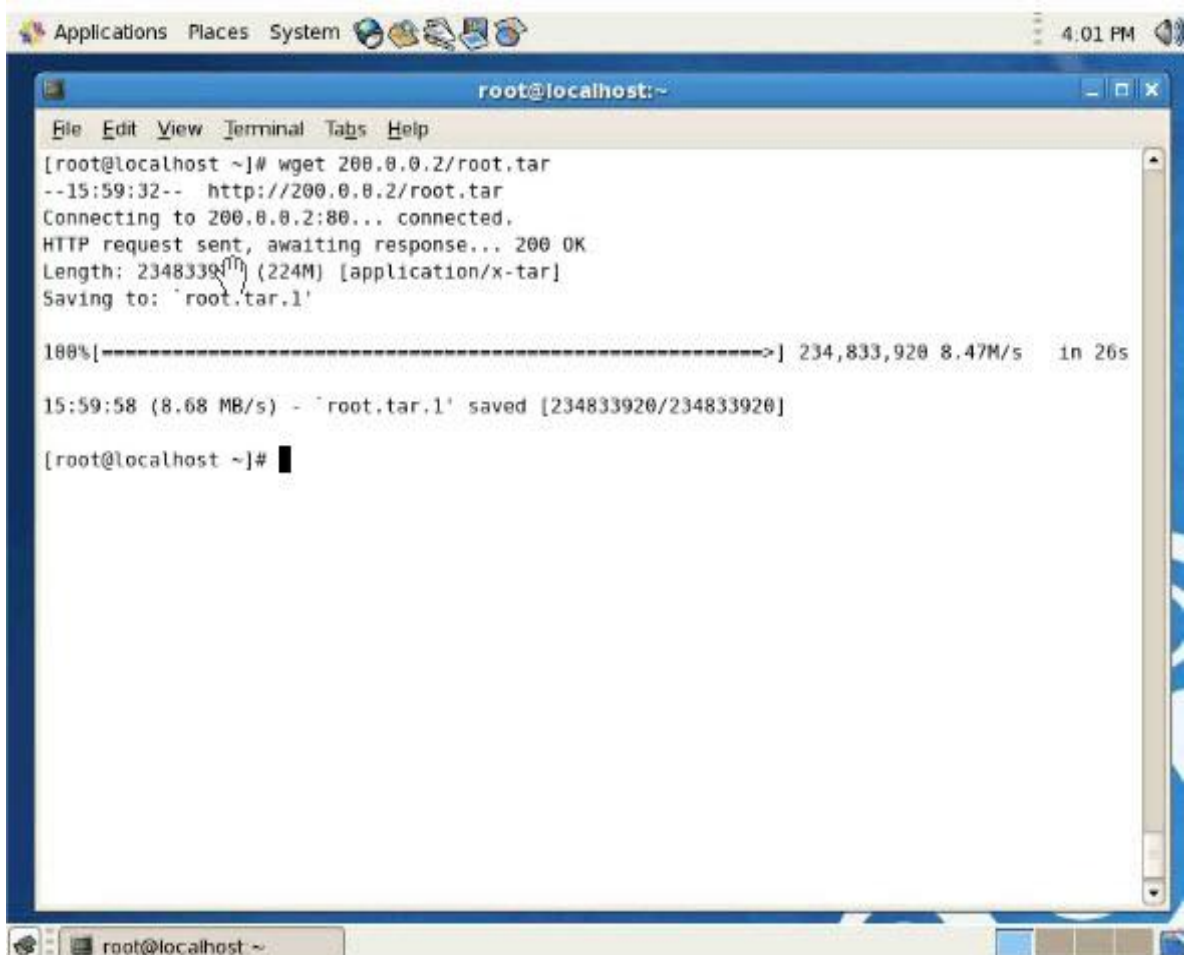
Tổng băng thông trên interface eth0 của client

$$BW1 = BW \times 70\% = 1024 \times 70\% = 717 \text{ kbit/s}$$

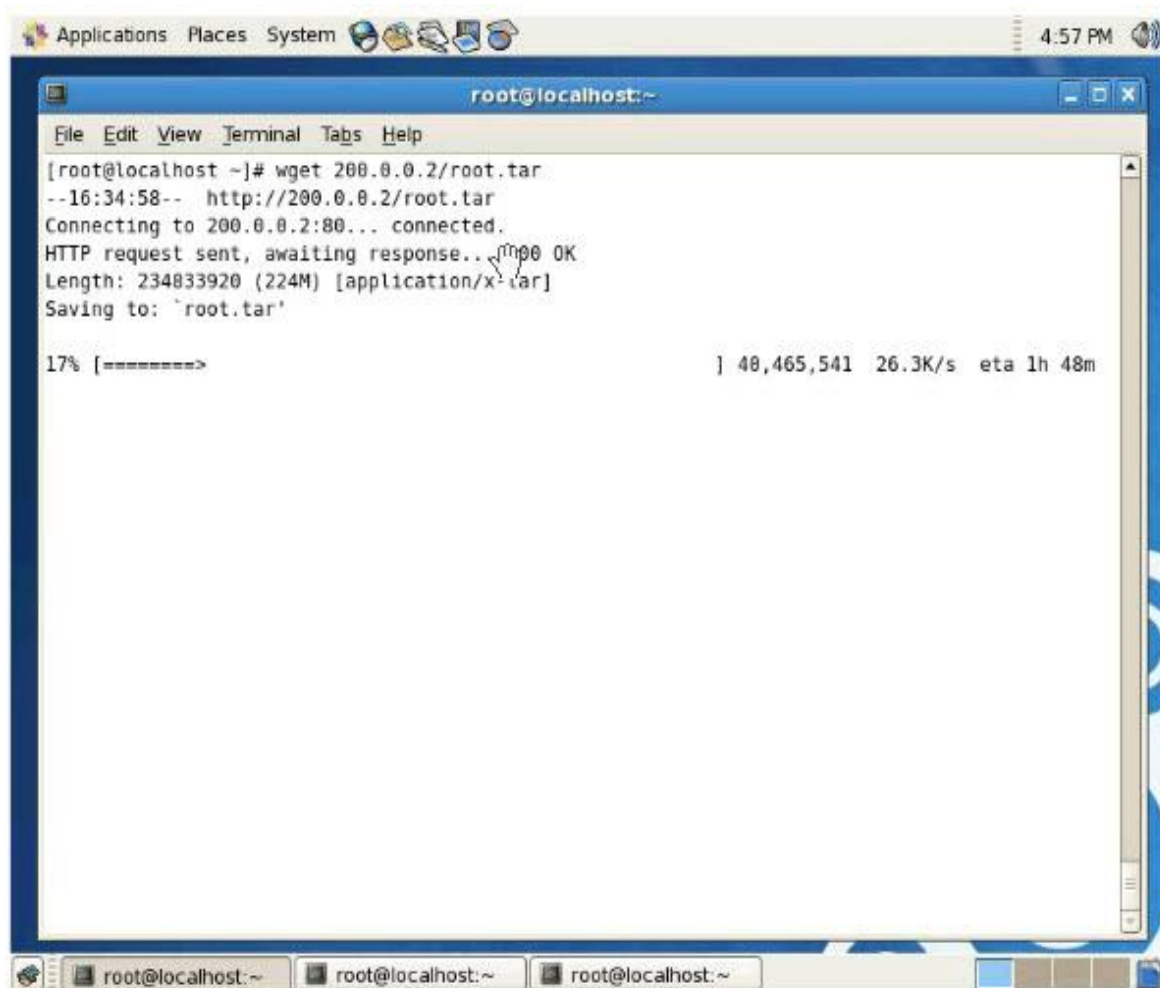
Trong đó :

- Dịch vụ FTP là : 359 kbit/s (50%)
- Dịch vụ HTTP là : 215 kbit/s (30%)
- Các dịch vụ còn lại là 143 kbit/s

Tại client, lần lượt tạo lưu lượng FTP, HTTP đến máy chủ. Để cho việc kiểm tra băng thông được chính xác, cần chọn dữ liệu có dung lượng lớn để tải từ máy chủ xuống.



Hình 3.5 : Trước khi điều khiển lưu lượng



Hình 3.6 Sau khi điều khiển lưu lượng

3.5 Đánh giá kết quả

Như vậy qua hai kịch bản thực nghiệm ở trên đã giải quyết được bài toán đưa ra lúc đầu về vấn đề hệ thống mạng của công ty cổ phần giáo dục sách điện tử EDC. Trong kịch bản 1, nhờ Cluster, hệ thống mạng của công ty sẽ không bị gián đoạn hoặc sẽ không lo sợ khi có một sự cố bất kì xảy ra.

Trong kịch bản 2 đã cho thấy, nhà quản trị mạng có thể đưa ra các chính sách ưu tiên cho từng loại dịch vụ khi điều khiển lưu lượng, để đảm bảo cho hệ thống mạng luôn được ổn định (đặc biệt là thời điểm có nhiều dịch vụ mạng hoạt động cùng truy xuất đến máy chủ)

KẾT LUẬN

Luận văn đề cập đến thành phần, chức năng của hệ thống máy tính cụm, các vấn đề trong quản lý tắc nghẽn, điều khiển lưu lượng, ứng dụng thuật toán gáo rò giải quyết bài toán điều khiển lưu lượng trong máy tính cụm. Các kết quả của luận văn gồm có :

1. Trình bày các thành phần, chức năng của hệ thống máy tính cụm, các vấn đề về điều khiển lưu lượng và điều khiển lưu lượng trong mạng máy tính cụm.

2. Nghiên cứu lý thuyết hàng đợi - nền tảng của thuật toán gáo rò – các khả năng điều khiển tốc độ, định dạng luồng thông tin, xác định độ trễ của dữ liệu vào và ứng dụng thuật toán gáo rò giải quyết bài toán điều khiển lưu lượng trong máy tính cụm.

3. Xây dựng hệ thống thử nghiệm (Web cluster) và thực hiện hai kịch bản để minh họa khả năng chịu lỗi và ứng dụng thuật toán gáo rò trong việc điều khiển lưu lượng.

Trên cơ sở những kiến thức và kết quả đã trình bày trong luận văn, trong thời gian tiếp theo sẽ phát triển nghiên cứu để thử nghiệm nhiều kịch bản sát với thực tiễn hơn. Từ đó sẽ từng bước đưa mô hình nghiên cứu vào ứng dụng trong thực tiễn.

TÀI LIỆU THAM KHẢO

Tài liệu tiếng Việt

[1] Trần Xuân Trường, Nghiên cứu phương pháp điều khiển chống tắc nghẽn trong mạng ATM bằng mạng Neural, luận án tiến sĩ tại trường Đại học Giao thông Vận tải, ngày bảo vệ 14/04/2012

Tài liệu tiếng Anh

[2] Matthew Steggink, Reliable network booting of cluster computers, University of Amsterdam, 2008

[3] Alexey N. Kuznetsov, Traffic control Token Bucket Filter

[4] VMware Inc. VMWare Workstation User's manual, 6.0 edition, 09 2007

Website :

[5] <http://kenhgiaiphap.vn/Detail/156/Cong-nghe-Clustering-cho-cac-he-thong-may-tinh.html>

[6] <http://www.slideshare.net/vimal25792/leaky-bucket-tocken-buckettraffic-shaping>

[7] <http://whatis.techtarget.com/definition/leaky-bucket-algorithm>

[8] <https://www.youtube.com/watch?v=CizMSni5yzY>

[9] <http://thonghoang.com/tong-hop/huong-dan-cai-dat-su-dung-phan-mem-vmware-10-0-2-full.html>

PHỤ LỤC : KẾT QUẢ HIỂN THỊ SAU KHI CÀI ĐẶT XONG HEARTBEAT

```
Loaded plugins: fastestmirror, presto
Loading mirror speeds from cached hostfile
epel/metalink | 4.1 kB 00:00
* base: ftp.dgn.net.tr
* epel: ftp.jaist.ac.jp
* extras: ftp.dgn.net.tr
* updates: ftp.dgn.net.tr
epel | 4.0 kB 00:00
epel/primary_db | 3.7 MB 00:21
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package heartbeat.i686 0:3.0.4-1.el6 will be installed
--> Processing Dependency: libapphb.so.2 for package: heartbeat-3.0.4-1.el6.i686
--> Processing Dependency: PyXML for package: heartbeat-3.0.4-1.el6.i686
--> Processing Dependency: cluster-glue for package: heartbeat-3.0.4-1.el6.i686
--> Processing Dependency: resource-agents for package: heartbeat-3.0.4-1.el6.i686
--> Processing Dependency: libhbclient.so.1 for package: heartbeat-3.0.4-1.el6.i686
--> Running transaction check
```

```
---&gt; Package PyXML.i686 0:0.8.4-19.el6 will be installed
---&gt; Package cluster-glue.i686 0:1.0.5-2.el6 will be installed
---&gt; Package heartbeat-libs.i686 0:3.0.4-1.el6 will be installed
---&gt; Package resource-agents.i686 0:3.9.2-7.el6 will be installed
--&gt; Processing Dependency: /sbin/quotaon for package: resource-agents-
3.9.2-7.el6.i686
--&gt; Processing Dependency: /sbin/quotacheck for package: resource-agents-
3.9.2-7.el6.i686
--&gt; Processing Dependency: /sbin/mount.cifs for package: resource-agents-
3.9.2-7.el6.i686
--&gt; Running transaction check
---&gt; Package cifs-utils.i686 0:4.8.1-5.el6 will be installed
--&gt; Processing Dependency: keyutils for package: cifs-utils-4.8.1-5.el6.i686
--&gt; Processing Dependency: libtalloc.so.2 for package: cifs-utils-4.8.1-
5.el6.i686
---&gt; Package quota.i686 1:3.17-16.el6 will be installed
--&gt; Processing Dependency: tcp_wrappers for package: 1:quota-3.17-
16.el6.i686
--&gt; Running transaction check
---&gt; Package keyutils.i686 0:1.4-3.el6 will be installed
---&gt; Package libtalloc.i686 0:2.0.1-1.1.el6 will be installed
---&gt; Package tcp_wrappers.i686 0:7.6-57.el6 will be installed
--&gt; Finished Dependency Resolution

Dependencies Resolved
```

=====				
=====				
Package	Arch	Version	Repository	Size
=====				
=====				
Installing:				
heartbeat	i686	3.0.4-1.el6	epel	161 k
Installing for dependencies:				
PyXML	i686	0.8.4-19.el6	base	892 k
cifs-utils	i686	4.8.1-5.el6	base	43 k
cluster-glue	i686	1.0.5-2.el6	base	69 k
heartbeat-libs	i686	3.0.4-1.el6	epel	260 k
keyutils	i686	1.4-3.el6	base	38 k
libtalloc	i686	2.0.1-1.1.el6	base	18 k
quota	i686	1:3.17-16.el6	base	202 k
resource-agents	i686	3.9.2-7.el6	base	470 k
tcp_wrappers	i686	7.6-57.el6	base	61 k
Transaction Summary				
=====				
=====				
Install	10 Package(s)			
Total download size: 2.2 M				
Installed size: 7.8 M				

Is this ok [y/N]: y

Downloading Packages:

Setting up and reading Presto delta metadata

Processing delta metadata

Package(s) data still to download: 2.2 M

(1/10): PyXML-0.8.4-19.el6.i686.rpm	892 kB	00:04
(2/10): cifs-utils-4.8.1-5.el6.i686.rpm	43 kB	00:00
(3/10): cluster-glue-1.0.5-2.el6.i686.rpm	69 kB	00:00
(4/10): heartbeat-3.0.4-1.el6.i686.rpm	161 kB	00:01
(5/10): heartbeat-libs-3.0.4-1.el6.i686.rpm	260 kB	00:01
(6/10): keyutils-1.4-3.el6.i686.rpm	38 kB	00:00
(7/10): libtalloc-2.0.1-1.1.el6.i686.rpm	18 kB	00:00
(8/10): quota-3.17-16.el6.i686.rpm	202 kB	00:01
(9/10): resource-agents-3.9.2-7.el6.i686.rpm	470 kB	00:02
(10/10): tcp_wrappers-7.6-57.el6.i686.rpm	61 kB	00:00

Total 118 kB/s | 2.2 MB 00:18

warning: rpmts_HdrFromFdno: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY

Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6

Importing GPG key 0x0608B895:

Userid : EPEL (6)

Package: epel-release-6-5.noarch (installed)

From : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6

Is this ok [y/N]: y

Running rpm_check_debug

Running Transaction Test

Transaction Test Succeeded

Running Transaction

Warning: RPMDB altered outside of yum.

Installing : PyXML-0.8.4-19.el6.i686	1/10
Installing : keyutils-1.4-3.el6.i686	2/10
Installing : libtalloc-2.0.1-1.1.el6.i686	3/10
Installing : cifs-utils-4.8.1-5.el6.i686	4/10
Installing : tcp_wrappers-7.6-57.el6.i686	5/10
Installing : 1:quota-3.17-16.el6.i686	6/10
Installing : resource-agents-3.9.2-7.el6.i686	7/10
Installing : cluster-glue-1.0.5-2.el6.i686	8/10
Installing : heartbeat-libs-3.0.4-1.el6.i686	9/10
Installing : heartbeat-3.0.4-1.el6.i686	10/10

Installed:

heartbeat.i686 0:3.0.4-1.el6

Dependency Installed:

PyXML.i686 0:0.8.4-19.el6	cifs-utils.i686 0:4.8.1-5.el6
cluster-glue.i686 0:1.0.5-2.el6	heartbeat-libs.i686 0:3.0.4-1.el6
keyutils.i686 0:1.4-3.el6	libtalloc.i686 0:2.0.1-1.1.el6
quota.i686 1:3.17-16.el6	resource-agents.i686 0:3.9.2-7.el6

tcp_wrappers.i686 0:7.6-57.el6