BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

LUẬN VĂN THẠC SỸ KHOA HỌC

Quản trị mạng tập trung trên nền WEB sử dụng công nghệ SNMP, CGI và CORBA cho hệ thống cung cấp dịch vụ Digital Subscriber Line (DSL) của Bưu điện Hà nội

NGÀNH: XỬ LÝ THÔNG TIN VÀ TRUYỀN THÔNG
MÃ SỐ:

TRẦN VĨNH THANH

Người hướng dẫn khoa học: **TS. HÀ QUỐC TRUNG**

HÀ NỘI 2006

LỜI CẢM ƠN

Trước hết, xin được gửi lời cảm ơn đến thầy giáo hướng dẫn tôi là tiến sĩ Hà Quốc Trung, người đã giúp đỡ tôi trong quá trình nghiên cứu hoàn thành luận văn này.

Cho phép tôi gửi lời cảm ơn đến Trung tâm tin học Bưu điện Hà nội, đặc biệt là các anh chị em đồng nghiệp tại Đài Điều Hành Mạng VNN, nơi tôi đang công tác đã tích cực cộng tác, tham gia vào các thử nghiệm, tìm hiểu hệ thống và tạo điều kiện để tôi được thử nghiệm các giải pháp liên quan đến đề tài.

Tôi cũng xin gửi lời cảm ơn đến các bạn cùng học trong khóa đào tạo thạc sỹ chuyên ngành Xử Lý Thông Tin Và Truyền Thông 2004-2006 đã cung cấp các tài liệu cần thiết trong quá trình nghiên cứu và đã giúp đỡ tôi rất nhiều trong quá trình học tập, chuẩn bị luận án.

Cuối cùng cho phép tôi cảm ơn các bạn bè, gia đình đã giúp đỡ, ủng hộ tôi rất nhiều trong toàn bộ quá trình học tập cũng như nghiên cứu hoàn thành luận văn này.

LỜI CAM ĐOAN

Tôi xin cam đoan luận văn này là công trình nghiên cứu của chính bản thân. Các nghiên cứu trong luận văn này dựa trên những tổng hợp lý thuyết và hiểu biết thực tế, không sao chép.

Tác giả

Trần Vĩnh Thanh

Mục lục

Danh sách các thuật ngữ và từ viết tắt

ADSL	Asymmetric Digital Subscriber Line
API	Application Program Interfaces
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
BOA	Basic Object Adapter
BGP	Border Gateway Protocol
CCITT	International Telegraph and Telephone Consultative Comittee
CGI	Common Gateway Interface
CORBA	Common Object Request Broker Architecture
CSDL	Cơ Sở Dữ Liệu
DII	Dynamic Invocation Interface
DNS	Domain Name Service
DSI	Dynarnic Skeleton Invocation
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
IDL	Interface Definition Language
IETF	Intemet Engineering Task Force
IIOP	Intemet Inter-ORB protocol
IOR	Interoperable Object Reference
IOS	International Organization for Standardization
IOS	Internetworking Operating System
IP	Internet Protocol

JAR	Java ARchive
MTU	Maxium Transfer Unit
NMS	Network Management System
NNM	Network Node Manager
MIME	Multipurpose Internet Mail Extensions
OID	Object Identifier
OMG	Object Management Group
PDU	Protocol Data Unit
PPP	Point-to-Point Protocol
RADIUS	Remote Authentication Dial In User Service
RDBMS	Relational database management system
RFC	Request For Comment
RMON	Remote Monitoring
SGMP	Simple Gateway Monitor Protocol
SHA	Secure Hash Algorithm
SMB	Server Message Block
SHDSL	Symmetric High-speed Digital Subscriber Line
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
STDIN	Standard Input
STDOUT	Standard Output
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USM	User-based Security Model
WWW	World Wide Web

Danh mục hình vẽ

Danh mục các bảng

Lời nói đầu

Cuộc cách mạng Internet trong những năm gần đây và sự lấn át của các dịch vụ truy nhập internet qua ADSL trước các dịch vụ truy nhập truyền thống qua Dial-up đã đặt ra nhiều bài toán lớn cho các nhà cung cấp dịch vụ (ISP) trong việc xây dựng quản lý một số lượng khổng lồ các thiết bị DSLAM phục vụ lắp đặt ở khắp nơi trong địa bàn cung cấp.

Bên cạnh đó, sự bùng nổ mạnh mẽ của các dịch vụ Web và khả năng sử dụng được web ở mọi nơi, mọi lúc, vào mọi thời điểm mà không phụ thuộc vào hệ thống nền hay khoảng cách địa lý đã tạo ra một trào lưu web hóa các loại hình dịch vụ, kể cả các loại dịch vụ có tính chất chuyên môn cao, xưa nay vẫn gói gọn trong các phòng thí nghiệm hay các trung tâm máy tính lớn như quan trắc và quản lý các dịch vụ mạng.

Trong luận văn này, chúng tôi sẽ đề cập đến vấn đề sử dụng công nghệ web (CGI, CORBA) và công nghệ quản trị mạng truyền thống (SNMP) để theo dõi và quản trị các thiết bị cung cấp dịch vụ DSLAM với mục đích xây dựng một cổng giao tiếp trên nền WEB phục vụ công tác quản trị các thiết bị DSLAM của các nhà sản xuất khác nhau hiện đang được khai thác tại Bưu điện Hà nội.

Về phương diện lý thuyết, luận án này sẽ đi vào tìm hiểu giao thức quản trị mạng SNMP và mô hình quản trị mạng dựa trên giao thức này; công nghệ cổng giao tiếp chung CGI trên WWW và CORBA cũng sẽ được giới thiệu ở các khía cạnh chính, có liên quan đến việc phát triển ứng dụng quản trị mạng trên nền web.

Chương I. TỔNG QUAN

I.1. Một số vấn đề cơ bản

Giao thức quản trị mạng SNMP đã được đưa ra từ những năm 80 của thế kỷ trước nhưng đến nay vẫn được sử dụng rộng rãi trong lĩnh vực quản trị của các mạng TCP/IP. Mặc dù khi mới được đưa ra, SNMP chỉ được thiết kế như một giải pháp tạm thời để quản trị mạng TCP/IP nhưng do TCP/IP đã quá phổ biến và thành chuẩn giao tiếp de-factor của thế giới, SNMP cũng trở thành một chuẩn đóng vai trò cực kỳ quan trọng trong việc thiết kế các phần mềm quản trị mạng của các thiết bị cung cấp dịch vụ.

Common Object Request Broker Architecture (CORBA) được OMG (Object Management Group) đưa ra như là một bộ khung kiến trúc chuẩn cho các ứng dụng hướng đối tượng trên mạng. CORBA đưa ra nhiều xác lập quan trọng như là trong suốt hóa tính địa phương của các đối tượng, gắn kết ngôn ngữ bậc cao cũng như đưa ra các phương thức gọi hàm động.

Như chúng ta đã biết, các trang web tĩnh sẽ không đủ khả năng cung cấp các thông tin cần được chất cập nhật thường xuyên như các ứng dụng dựa trên GUI (Graphical User Interface) của windows. Công nghệ sử dụng JavaApplet nhúng trong các trình duyệt đã khắc phục được điểm yếu này, và có khả năng cung cấp đầy đủ các thông tin cập nhật thời gian thực, kể cả thông tin dưới dạng đồ họa. Sử dụng Java trong các trình duyệt trên thực tế đã mở rộng khả năng của web lên nhiều lần, khiến cho web trở thành một môi trường vạn năng truyền tải thông tin không bị giới hạn về khoảng cách hay sự khác biệt về cấu hình hệ nền.

I.2. Lý do chọn đề tài

Dịch vụ truy nhập Internet băng thông rộng sử dụng công nghệ ADSL lần đầu tiên được Tập đoàn Bưu chính Viễn thông Việt nam (VNPT) thử nghiệm vào năm 2001 và được triển khai rộng rãi từ tháng 7 năm 2003 với tên thương hiệu là MegaVNN. Dịch vụ này từ khi ra đời đến nay đã có những bước phát triển nhảy vọt, đáp ứng được yêu cầu của người dùng về băng rộng, và dần dần thay thế dịch vụ truy cập Internet gián tiếp (Dial-up) qua đường dây điện thoại truyền thống.

Là một thành viên của VNPT, hiện nay trên địa bàn thành phố, Bưu điện TP Hà nội đang cung cấp 2 dịch vụ chính sử dụng công nghệ xDSL là dịch vụ truy nhập Internet băng rộng qua ADSL và dịch vụ dịch vụ mạng riêng ảo - MegaWan trên cả 2 loại đường truyền ADSL và SHDSL.

Để có thể cung cấp dịch vụ xDSL trên địa bàn thành phố Hà nội, hiện nay Bưu điện Hà nội đang quản lý một hạ tầng mạng lưới bao gồm một hệ thống phục vụ truy nhập hiện đại với các thiết bị DSLAM (Digital Subscriber Line Access Multiplexer) phân bổ ở khắp nơi trên địa bàn thành phố (hơn 140 điểm lắp đặt, gần 200 DSLAM …) của nhiều nhà cung cấp thiết bị nổi tiếng. Nhu cầu sử dụng xDSL trên địa bàn vẫn đang tiếp tục phát triển rất nhanh, số lượng các thiết bị DSLAM khai thác trên mạng liên tục được đầu tư mới nhằm đáp ứng được nhu cầu của khách hàng, mạng lưới được mở rộng và độ phức tạp tăng lên. Đến nay, trên địa bàn Hà nội hiện có 8 chủng loại thiết bị của 4 nhà sản xuất khác nhau Siemens, Huawei, Tailyn, ZTE … với các công nghệ khác nhau như ATM DSLAM, IP DSLAM…

Hệ thống các DSLAM thuộc 4 hãng sản xuất này được quản trị, giám sát, khai thác mạng từ xa bởi 04 hệ thống quản lý NMS (Network Management System) tập trung do từng hãng sản xuất thiết bị cung cấp. Các hệ thống

NMS này đều là môi trường đóng, được thiết kế hướng tới đối tượng là các kỹ thuật viên vận hành mạng nên không cung cấp giao diện ra bên ngoài và không có mối liên hệ với nhau.

Với những hạn chế trên, cùng với sự phát triển của mạng lưới xDSL cả về số lượng và chủng loại thiết bị đã đặt ra một thách thức lớn đối với Bưu điện Hà nội trong việc vận hành, khai thác hệ thống; cũng như ảnh hưởng đến chất lượng các quy trình cung cấp dịch vụ của đơn vị, cụ thể như sau:

Không có chức năng để cho phép các hệ thống hỗ trợ bên ngoài giao tiếp với phần quản lý mạng

Do không có chức năng giao tiếp với các hệ thống hỗ trợ bên ngoài (ví dụ hệ thống quản lý khách hàng, hệ thống hỗ trợ dịch vụ.…), quá trình cung cấp dịch vụ (đóng mở cổng dịch vụ, khởi tạo dịch vụ, tháo hủy dịch vụ…) đều phải chuyển đến kỹ thuật viên khai thác mạng thực hiện bằng nhân công thông qua hệ thống NMS của mỗi hãng; không cho phép kết nối, thực hiện tự động hóa dây chuyền sản xuất, cũng như không thể xây dựng và phát triển thành một giải pháp tổng thể. Điều đó đã dẫn đến các hệ quả tất yếu sau:

- *Số lượng thao tác hàng ngày tăng lên theo số lượng thuê bao và dịch vụ:* Một ngày phải thực hiện nhiều yêu cầu đóng/mở cổng (khi có khách hàng mới hòa mạng, huỷ hợp đồng, nợ, trả nợ cước, vv…). Có những ngày, số lượng yêu cầu lên đến hơn 300; thời gian thực hiện trong từ 7:00 cho đến 21:00 với các quy định chặt chẽ về thời gian để hạn chế tối đa việc mất liên lạc của khách hàng;
- *Tạo một sức ép không nhỏ đối với quá trình vận hành và khai thác hệ* thống do phải sử dụng nhiều loại phần mềm quản lý NMS đối với những công việc hàng ngày (kiểm tra thông số cổng, đóng, mở, reset

cổng) . Thực tế là đã có lúc, cán bộ quản lý mạng phải ngồi trước 04 màn hình NMS và phải thao tác qua lại giữa 4 NMS này;

Công tác hỗ trợ và chăm sóc khách hàng gặp nhiều khó khăn:

Vì lý do an ninh, bảo mật nên phần quản lý mạng NMS nên kỹ thuật viên tại bộ phận hỗ trợ không có thông tin về trạng thái thiết bị để trả lời và hỗ trợ khách hàng mà phải hỏi thông tin từ bộ phận quản lý mạng NMS, ảnh hưởng không tốt đến chất lượng chăm sóc khách hàng, tốn nhiều nhân lực và mất nhiều thời gian chờ đợi..

Khó khăn trong việc tích hợp ứng dụng, nâng cao chất lượng, tùy biến của dịch vụ:

Các phần mềm quản lý thiết bị DLSAM được thiết kế cho các nhu cầu quản lý chung nên có nhiều điểm không phù hợp với nhu cầu sử dụng của Bưu điện Hà nội; không tích hợp với các CSDL hiện có của Bưu điện Hà nội, do vậy gặp nhiều khó khăn trong việc tích hợp ứng dụng, nâng cao chất lượng của dịch vụ.

Không có một giải pháp tổng thể cho toàn hệ thống:

Không có một hãng cung cấp thiết bị DSLAM nào có khả năng cung cấp một giải pháp tổng thể thỏa mãn các yêu cầu trên, do giải pháp thiết bị của mỗi hãng đều khác nhau, các hãng chỉ có thể có khả năng cung cấp giải pháp đối với thiết bị của họ khi có yêu cầu, mà không quan tâm đến thiết bị của các hãng sản xuất khác. Thực tế tại mạng do Bưu điện Hà nội quản lý đã tồn tại thiết bị của 4 hãng sản xuất, trong khi số hãng cung cấp thiết bị trên thị trường Việt nam ước tính lớn hơn 10 hãng.

Sự phát triển ngày càng mạnh mẽ của dịch vụ xDSL với xu hướng nâng cao chất lượng dịch vụ mà vẫn tiết kiệm nguồn nhân lực kỹ thuật cao đòi hỏi phải có một giải pháp giải quyết triệt để các vấn đề đã nêu trên.

Là một cán bộ kỹ thuật đang công tác tại một đơn vị cung cấp dịch vụ lớn với, tôi có cơ hội được tiếp xúc với những công nghệ tiên tiến của thế giới cũng như được va chạm nhiều với các vấn đề nảy sinh mà một nhà cung cấp dịch vụ phải đối mặt khi tiến hành cung cấp dịch vụ mạng trên quy mô rộng, đặc biệt là vấn đề quản trị mạng và những rắc rối nảy sinh trong thực tế khi phải phối hợp hoạt động giữa nhiều đơn vị, sử dụng nhiều loại thiết bị của nhiều nhà cung cấp khác nhau. Lựa chọn đề tài "*Quản trị mạng tập trung trên nền WEB sử dụng công nghệ SNMP, CGI và CORBA cho hệ thống cung cấp dịch vụ Digital Subscriber Line (DSL) của Bưu điện Hà nội*", chúng tôi đang hướng tới mục tiêu tìm hiểu công nghệ quản trị mạng dựa trên WEB và xây dựng một giải pháp phần mềm ứng dụng trong thực tế phù hợp với mô hình khai thác, quản lý nơi tôi đang công tác nói riêng và có thể áp dụng cho các nhà nhà cung cấp dịch vụ khác. Phần mềm cần phải đáp ứng các yêu cầu đã đặt ra với các khả năng:

- Cho phép tự động hóa các thao tác khai thác hàng ngày;
- Cung cấp giao tiếp cho phép các ứng dụng/dịch vụ hỗ trợ bên ngoài được giao tiếp với các thiết bị DSLAM. Có thể theo dõi trạng thái thiết bị từ xa, tuỳ theo phân quyền của các đơn vị tham gia khai thác phù hợp với quy trình quản lý dịch vụ của nhà cung cấp dịch vụ, tạo tiền để để tiến tới thực hiện các chức năng quản lý phức tạp hơn…
- Nhất thể hóa giao diện quản lý, giúp người sử dụng tránh việc phải thao tác với nhiều phần mềm quản lý khác nhau;

Nhận thức được ý nghĩa quan trọng của việc tin học hóa, tự động hóa dần các thao tác đơn giản, giải phóng nguồn nhân lực có trình độ cao khỏi các thao tác đơn điệu, cũng như nâng cao chất lượng cung cấp dịch vụ, nhóm thực hiện đề tài sẽ cố gắng hoàn thành đề tài hướng tới khả năng áp dụng vào thực tế không chỉ đối với đơn vị mình, mà có thể áp dụng vào các đơn vị khác.

I.3. Cấu trúc của luận án

Luận án được chia thành 5 chương với các nội dung chính sau:

- Chương 1: Tổng quan, trình bày những vấn đề cơ bản sẽ được trình bày trong đề tài, lý do lựa chọn đề tài và trình bày sơ qua về cấu trúc luận án
- Chương 2 sẽ trình bày những vấn đề cơ bản của giao thức quản trị mạng SNMP và mô hình quản trị mạng thông thường, sự ra đời và phát triển của ; các vấn đề liên quan đến SNMP như SMI, MIB, OID cũng như các chuẩn cơ bản của SNMP, các hạn chế của SNMP và khắc phục…
- Chương 3 sẽ trình bày những vấn đề cơ bản của CGI và CORBA. Các vấn đề sẽ được trình bày ở đây là chuẩn CGI, các đặc trưng của CGI, mô hình quan hệ Client/Server ba bên sử dụng CGI, mô hình quản trị mạng qua web, cơ bản về lập trình CGI… . Chương 3 cũng sẽ khái lược về CORBA, giải pháp sử dụng CORBA làm môi trường xây dựng ứng dụng quản trị mạng qua web. Các vấn đề sẽ được trình bày ở đây là chuẩn CORBA, tổng quan về kiến trúc CORBA, bộ phận trung gian xử lý các yêu cầu trên đối tượng (Object Request Broker –

ORB), mô hình bốn bên giữa Web client, Web server, NMS Agent và DSLAM trên CORBA.

- Chương 4 Áp dụng thực tế hệ thống quản trị hệ thống cung cấp dịch vụ xDSL của Bưu điện Hà nội. Giới thiệu hệ thống quản lý mạng cung cấp dịch vụ xDSL của Bưu điện Hà nội đang được triển khai thực tế và các giải pháp xây dựng công cụ quản trị các thiết bị DSLAM thông qua giao thức SNMP dựa trên trên nền web bằng CGI và CORBA. Chương này sẽ trình bày những phần cơ bản liên quan đến xây dựng giải pháp quản trị mạng tập trung qua WEB sử dụng CGI cũng như CORBA, giới thiệu sơ bộ về gói phần mềm VisiBroker và trình bày cụ thể phương pháp xây dựng công cụ quản trị mạng DSLAM sử dụng CORBA
- Chương 5 Kết quả thực tiễn và áp dụng, trình bày những kết quả đạt được của đề tài, một số so sánh giữa hai công cụ quản trị mạng dựa trên CGI và CORBA. Chương 5 cũng sẽ trình bày những khả năng phát triển, mở rộng của đề tài, để có thể ứng dụng được nhiều hơn trong thục tế trong việc, đặc biệt là áp dụng vào hệ thống quản trị mạng DSL của Bưu điện Hà nội.

Chương II. Giao thức SNMP

SNMP (Simple Network Management Protocol): là giao thức được sử dụng rất phổ biến để giám sát và điều khiển thiết bị mạng như switch, router... Với những văn phòng nhỏ chỉ có vài thiết bị mạng và đặt tập trung một nơi thì có lẽ ta không thấy được lợi ích của SNMP; Nhưng với các hệ thống mạng lớn, thiết bị phân tán nhiều nơi, đặc biệt là trong các hệ thống mạng của các nhà cung cấp dịch vụ với mô hình quản lý tập trung thì việc sử dụng SNMP dường như là bắt buộc.

Giao thức SNMP được thiết kế để cung cấp một phương thức đơn giản để quản lý tập trung mạng TCP/IP. Nếu muốn quản lý các thiết bị từ 1 vị trí tập trung, giao thức SNMP sẽ vận chuyển dữ liệu từ client (thiết bị mà đang giám sát) đến server nơi mà dữ liệu được lưu trong log file nhằm phân tích dễ dàng hơn. Các phần mềm ứng dụng dựa trên giao thức SNMP như: MOM của Microsft và HP Openview vv…

II.1. Một số vấn đề cơ bản về SNMP

Bản chất của SNMP là tập hợp một số lệnh đơn giản và các thông tin mà lệnh cần thu thập để giúp người quản trị thu thập dữ liệu và thay đổi cấu hình của các thiết bị tương thích với SNMP.

Ví dụ, SNMP có thể dùng để kiểm tra tốc độ hay ra lệnh shutdown một cổng Ethernet, theo dõi nhiệt độ của switch và cảnh báo khi nó lên quá cao.… SNMP có thể quản trị rất nhiều thiết bị, từ phần cứng đến phần mềm như Web server hay cơ sở dữ liệu, từ thiết bị đắt tiền như router đến một số hub rẻ tiền, hay các hệ thống Unix, Window, các máy in, nguồn điện… miễn là các thiết bị đó hỗ trợ SNMP. Các thiết bị được gọi là hỗ trợ hay tương thích

SNMP tức là nó được cài đặt một phần mềm để có thể thu thập một số thông tin và trả lời các yêu cầu của người quản trị.

II.1.1. Sự ra đời và phát triển của SNMP

Giao thức Simple Netwok Management Protocol (SNMP) ra đời vào năm 1988 để đáp ứng đòi hỏi cấp bách về một chuẩn chung cho quản trị mạng Internet. SNMP cung cấp cho người dùng một tập các lệnh đơn giản nhất để có thể quản trị được các thiết bị từ xa.

Được phát triển từ giao thức Simple Gateway Monitoring Protocol (SGMP), SNMP đã được mở rộng cho phù hợp với các yêu cầu của một hệ thống quản trị mạng đa dụng. Ban đầu, SNMP chỉ được xem như là một giải pháp tạm thời cho việc quản trị các mạng máy tính dựa trên nền TCP/IP trong khi chờ đợi chuyển hẳn sang một giao thức dựa trên kiến trúc mạng của OSI.

Tuy nhiên, do sự phát triển mạnh mẽ của các ứng dụng trên nền TCP/IP, nhất là từ năm 1990, đã khiến cho TCP/IP trở thành một giao thức truy nhập mạng *de factor* của thế giới. Điều đó cũng khiến cho SNMP trở thành giao thức quản trị mạng được sử dụng chính và không còn bị xem là một giải pháp tạm thời nữa [Stallings 96].

Các hoạt động và quy cách dữ liệu của SNMP được chỉ định dựa trên các tiêu chuẩn được đưa ra trong các bộ RFC (Request For Comment) và hiện chúng vẫn đang được phát triển. Trong số các RFC xây dựng nên chuẩn SNMP, có ba bộ tiêu chuẩn quan trọng được dùng làm cơ sở cho SNMP. Chúng là:

- RFC 1156 - Cấu trúc và định danh của các thông tin quản trị của internet trên nền TCP/IP (*Structure and Identification of Management Information for TCP/IP based internets*).

- RFC 1157 - A Simple Network Management Protocol (SNMP).
- RFC 1213 – Cơ sở thông tin quản trị mạng cho Internet trên nền TCP/IP (Management Information Base for Network Management of TCP/IP-based internets: MIB-II)

Phiên bản đầu tiên của SNMP (SNMPv1) ra đời năm 1988 được quy định trong RFC 1157. Ở phiên bản đầu tiên này, tiêu chí của SNMP đúng như tên gọi của nó, đó là sự đơn giản trong thực thi [Stallings 96] . Đó là lý do chính khiến cho tính bảo mật trong SNMPv1 rất lỏng lẻo, phụ thuộc vào một xâu chia sẻ tương tự như mật khẩu ở dạng thuần văn bản gọi là "commutitiy string". Điều này cho phép tất cả các ứng dụng SNMP nếu biết xâu này có thể truy cập thông tin quản trị trên thiết bị.

Mặc dù chuẩn SNMPv1 đã thuộc về quá khứ (historical standard) nhưng hiện nay nó vẫn là phiên bản mà rất nhiều các nhà sản xuất hỗ trợ.

Phiên bản tiếp theo của SNMP là SNMPv2 hay SNMPv2c. Được quy định trong RFC 3416, RFC 3417 và RFC 3418, SNMPv2 thêm các khuôn dạng dữ liệu, các MIB và PDU mới, làm tăng khả năng cho giao thức.

Tuy nhiên hai phiên bản đầu tiên này của SNMP vẫn thiếu các tính năng bảo mật, xác thực cần thiết nên vẫn có thể dễ dàng bị khai thác [Stallings 96] . SNMPv3 là phiên bản cuối cùng, chủ yếu tăng cường bảo mật trong quản trị mạng [Stallings 98] . Phiên bản này hỗ trợ giao thức xác thực mạnh và kênh giao tiếp được mã hóa giữa các thực thể được quản trị. Năm 2002, phiên bản này được chuyển từ bản thảo sang thành chuẩn, bao gồm các RFC 3410, RFC 3411, RFC 3412, RFC 3413, RFC 3414, RFC 3415, RFC 3416, RFC 3417, RFC 3418, và RFC 2576. Vì SNMPv3 là chuẩn mới được công bố, do vậy chỉ có một số hãng lớn như Cisco mới hỗ trợ SNMPv3. Tuy nhiên với

nhu cầu ngày càng cao của bảo mật trong quản trị mạng, sẽ có thêm ngày càng nhiều các hãng hỗ trợ SNMPv3 trong các sản phẩm của mình.

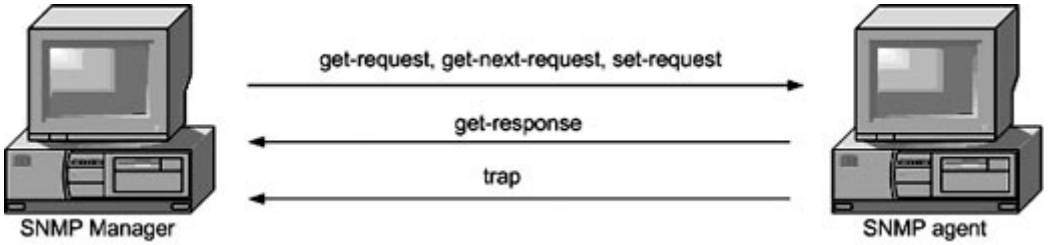
II.1.2. Mô hình SNMP

Chuẩn SNMP đưa ra một mô hình cơ sở cho các định nghĩa dữ liệu thông quản trị và chuẩn cho các giao thức trao đổi thông tin đó.

Trong kiến trúc của SNMP có hai loại thực thể là manager và agent. Manager là server chạy một phần mềm có khả năng điều khiển các công việc quản trị cho một mạng. Manager thường được gọi là trạm quản trị - Network Management Station (NMS). Trong một mạng, trạm quản trị chịu trách nhiệm thăm dò (polling) và nhận các trap từ agent. Thăm dò là hành động truy vấn một agent (router, switch, server Unix…) yêu cầu một số thông tin. Các thông tin này được trạm quản trị lưu trữ, phân tích và hiển thị. Trap cho phép agent thông báo cho trạm quản trị nếu có điều gì đó vượt khỏi phạm vi cho phép xảy ra. Khi nhận được trap, tùy theo thông tin mà trap cung cấp, trạm quản trị sẽ thực hiện một số thao tác đã được cấu hình từ trước. Chẳng hạn, nếu đường T1 kết nối ra Internet có sự số, ngay lập tức router gửi trap cho trạm quản trị, khi đó trạm quản trị có thể thực hiện hành động như thông báo lại cho người quản trị.

Thực thể thứ hai là agent, là một phần mềm nhỏ chạy trên thiết bị được quản trị [SnmpFAQ]. Nó có thể là một chương trình độc lập như một tiến trình daemon trong Unix, có thể là thành phần tích hợp bên trong hệ điều hành như IOS của router Cisco hay là hệ điều hành cấp thấp điều khiển UPS. Agent cung cấp thông tin về rất nhiều hoạt động của thiết bị. Ví dụ, agent trong router có thể theo dõi trạng thái up/down của các interface. Trạm quản trị có thể truy vấn trạng thái của các interface này và thức hiện các hành động tương ứng nếu interface down. Hoặc là nếu agent được cấu hình để có

khả năng nhận biết một số sự kiện xấu, agent có thể gửi trap đến trạm quản trị, nơi mà các tác vụ tương ứng sẽ được thực hiện. Một vài thiết bị. Hình II.1 minh họa mối quan hệ giữa trạm quản trị và agent.



Hình II.1 Mối quan hệ giữa manager và agent

Chú ý là trap và thăm dò có thể xảy ra đồng thời. Không có hạn chế gì về thời điểm trạm quản trị có thể thăm dò agent và thời điểm agent gửi trap

Mô hình SNMP của một hệ thống quản trị mạng bao gồm bố thành phần trọng yếu (các thành phần này được mô tả ở Hình II.2):

- Trạm quản trị;
- Thực thể bị quản trị (node hay Network Element - NE)
- Cơ sở thông tin quản trị
- giao thức quản trị.

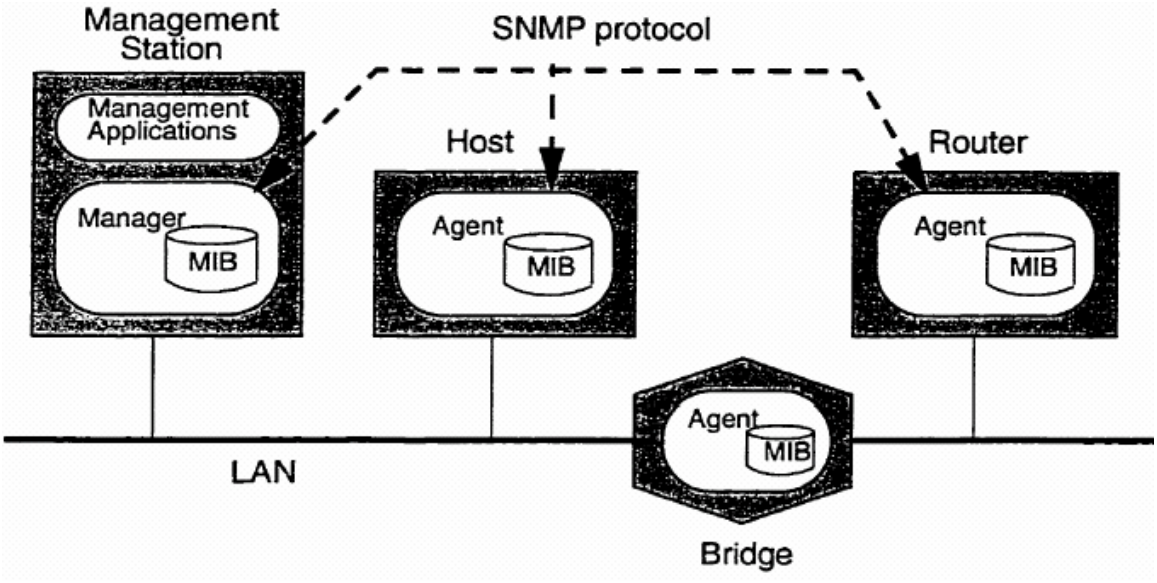
Việc quản trị mạng được thực hiện bởi các trạm máy tính quản trị. Các máy tính này sử dụng các phần mềm quản trị có nhiệm vụ quản lý một phần hoặc toàn bộ cấu hình của mạng theo yêu cầu của các ứng dụng quản trị hoặc các nhà quản trị mạng. Các phần mềm này có thể có giao diện đồ học cho phép các nhà quản trị theo dõi trạng thái của mạng và thực hiện các thao tác cần thiết khi có yêu cầu.

Các "điểm" quản trị (NE) có thể là các trạm làm việc, các thiết bị định tuyến, cầu hoặc chuyển mạch hoặc là bất kỳ một thiết bị nào có khả năng

trao đổi dữ liệu về trạng thái của mình với thế giới bên ngoài. Để có thể thực hiện được các chức năng "bị quản lý", các NE phải có được các tính năng cơ bản của một SNMP agent, thực chất đó là một modul phần mềm có chức năng lưu trữ và cập nhật các thông tin quản trị của thiết bị cũng như có khả năng gửi các thông tin đó đến cho trạm quản trị khi được yêu cầu.

Cấu trúc của các thông tin được xác định bởi thành phần Cơ sơ thông tin quản trị (Management Information Base - MIB).

Mỗi một hệ thống trên mạng duy trì một MIB phản ánh các trạng thái của các tài nguyên cần quản trị trong hệ thống đó.



Hình II.2 Các thành phần cơ bản của SNMP

Việc trao đổi dữ liệu giữa Manager và Agent được thực hiện trên giao thức SNMP [ietf]. Giao thức này cho phép các thực thể quản trị gửi các đến Agent các truy vấn về trạng thái các tài nguyên (còn gọi là các đối tượng). Các đối tượng này được định nghĩa trong MIB của các agent và có thể được thay đổi khi có yêu cầu. SNMP cung cấp ba tác vụ cơ bản như sau:

- *Get*: Trạm quản lý yêu cầu nhận giá trị của một hoặc nhiều đối tượng quản lý (MO) từ trạm bị quản lý;
- *Set*: Trạm quản lý yêu cầu thay đổi giá trị của một hoặc nhiều đối tượng quản lý (MO) tại trạm bị quản lý;
- *Trap*: Trạm bị quản lý gửi thông tin về trạng thái của một đối tượng quản lý khi có một biến cố đã được định nghĩa trước xảy ra

Theo quy định của giao thức SNMP, Get bao gồm 2 tác vụ *GetRequest* và *GetNextRequest*, trong đó:

- *GetRequest*: lấy giá trị của một hoặc nhiều biến
- *GetNextRequest*: lấy giá trị của biến kế tiếp

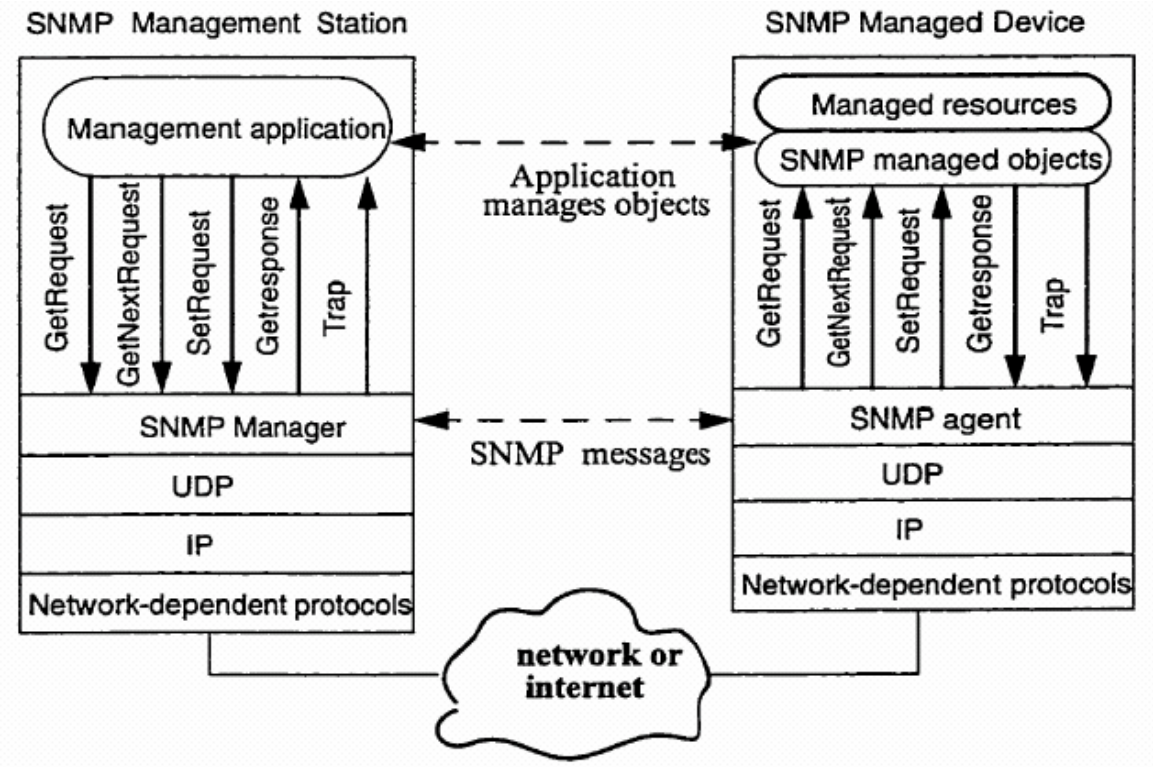
Từ phiên bản SNMP v2, có thêm một tuỳ chọn nữa được đưa vào, đó là *GetBulkRequest*. Câu lệnh này được sửu dụng chính để lấy một lượng lớn dữ liệu dạng ma trận

Bên cạnh đó, SNMP còn định nghĩa các tác vụ khác như:

- *GetResponse*: trả về giá trị của một hoặc nhiều biến sau khi phát lệnh G*etRequest* hoặc *GetNextRequest*, hoặc *SetRequest*.
- *InformRequest*: Cho phép các trạm quản trị gửi thông tin dạng trap đến các trạm quản lý khác (từ SNMP v2)

Trong mạng TCP/IP, SNMP là một giao thức hoạt động ở tầng ứng dụng và sử dụng giao thức UDP. Do đó, SNMP là một giao thức phi kết nối, tức là giữa manager và agent không có sự duy trì kết nối trong suốt quá trình trao đổi dữ liệu.

Hình II.3 là một minh họa của giao thức SNMP và các ứng dụng SNMP trong kiến trúc mạng, trong đó, network-dependent protocols có thể là Ethemet, FDDI hay X.25, vv…



Hình II.3 SNMP trong mô hình mạng

II.1.3. Cổng dịch vụ và dịch vụ truyền tải phi hồi đáp

SNMP được thiết kế để dử dụng trên các dịch vụ phi kết nối [SnmpFAQ]. Nguyên nhân dẫn đến quyết định này là do SNMP được thiết kế để có thể duy trì được liên lạc trong các trường hợp xuất hiện lỗi thiết bị hoặc lỗi mạng.

Nếu SNMP sử dụng các loại dịch vụ hướng kết nối (connection-oriented), việc mất kết nối sẽ giảm hiệu năng trao đổi dữ liệu của SNMP. Chính vì lý do đó, SNMP sử dụng giao thức UDP (User Datagram Protocol) trong kiến trúc TCP/IP. Trong mô hình OSI, SNMP cũng có được hỗ trợ bởi dịch vụ truyền vận phi kết nối (Comectioless Transport Service). Các phân đoạn

UDP được truyền đi trong các gói tin IP. UDP header có bao gồm cả địa chỉ nguồn và địa chỉ đích, cho phép các thực thể SNMP định danh địa chỉ của nhau. CÁc thực thể SNMP tiếp nhận các gói tin đến trên cổng UDP 116 ngoại trừ các gói tin TRAP. Trạm quản lý "nghe" các gói tin TRAP trên cổng 162.

Trong môi trường SNPM, các gói tin không nên có độ dài vượt quá 484 byte [ietf]. Tuy nhiên, các thực thể vẫn nên chấp nhận các gói dữ liệu lớn hơn nếu như hệ thống cho phép.SNMP sử dụng User Datagram Protocol (UDP) làm giao thức ở tầng giao vận để truyền dữ liệu giữa manager và agent vì rất nhiều lý do. Thứ nhất vì UDP là giao thức đơn giản, không liên kết nên :

- Gói tin có kích thước header nhỏ, thích hợp với truyền thông tin quản trị;
- Không tốn thời gian và công sức để thiết lập, duy trì và ngắt liên kết;
- Không tốn băng thông của mạng;
- Nhiều thiết bị được quản trị có tài nguyên CPU, bộ nhớ rất hạn chế, nên chỉ có thể cài đặt UDP làm giao thức ở tầng giao vận.

Ngoài ra, UDP không đòi hỏi tin cậy. SNMP được thiết kế để thông báo khi có lỗi xảy ra vì nếu mạng không bao giờ lỗi thì ta cũng không cần thiết phải giám sát. Sẽ là một ý tưởng tồi trong trường hợp mạng xảy ra tắc nghẽn hay bị lỗi, ta lại cố gắng truyền đi truyền lại để đảm bảo tính tin cậy như của TCP. Điều này chỉ làm cho mạng càng tắc nghẽn hơn.

Tuy nhiên không tin cậy cũng là một vấn đề của UDP. Điều này đòi hỏi các ứng dụng SNMP phải xử lý trường hợp gói tin bị mất và truyền lại nếu cần. Công việc này thường được thực hiện một các đơn giản với timeout. Trạm quản trị gửi một gói tin yêu cầu tới agent và chờ đợi trả lời trong một khoảng

thời gian được thiết lập trước gọi là timeout. Nếu sau thời gian timeout, trạm quản trị không nhận được gói tin trả lời từ agent, nó có thể giả sử rằng gói tin này bị mất và truyền lại yêu cầu nếu cần. Số lần truyền lại cũng có thể được cấu hình trước. Ta có thể thấy rằng không tin cậy không phải là vấn đề thực sự của UDP. Trong trường hợp tồi nhất trạm quản trị gửi đi một yêu cầu và không bao giờ nhận được trả lời. Tương tự với trap, nếu agent gửi đi một trap và nó không đến nơi nhận, trạm quản trị cũng không có cách nào biết được trap đã được gửi đi hay chưa và agent cũng không thể biết được trap có đến đích hay không. Do vậy thậm chí agent cũng không cần truyền lại trap.

SNMP sử dụng cổng UDP 161 để truyền và nhận yêu cầu và cổng 162 để nhận trap từ thiết bị được quản trị. Các cổng này là mặc định, các sản phẩm SNMP thường cho phép người sử dụng thay đổi cổng vì lý do an ninh. Ví dụ cổng nhận trap của manager có thể đổi thành 1999, khi đó agent cũng phải được cấu hình để gửi trap đến đúng cổng này.

II.1.4. SNMP community

SNMP sử dụng khái niệm community là một xâu dùng chung để thiết lập mối quan hệ tin cậy giữa manager và agent. Có ba loại community là : read-only, read-write và trap. Như tên gọi đã chỉ ra, ba community này cho phép giới hạn thực hiện ba công việc. Read-only chỉ cho phép đọc mà không được thay đổi nội dụng, chẳng hạn ta có thể đọc số lượng gói tin truyền qua một cổng của router nhưng không được phép thay đổi giá trị này. Read-write cho phép đọc và thay đổi giá trị, do vậy có thể đọc giá trị một biến đếm, thiết lập lại giá trị này, thậm chí thay đổi biến trạng thái của một interface hay thay đổi các cấu hình của router…. Community trap cho phép manager nhận trap từ agent.

Về bản chất community chính là mật khẩu, cả manager và agent đều sử dụng ba xâu giống nhau để đặt tên cho 3 loại community này. Hầu hết các hãng đều sử dụng xâu mặc định là public cho community read-only, private cho community read-write. Theo giá trị mặc định này, khi manager muốn đọc giá trị của một biến, manager trình xâu public trong gói tin yêu cầu. Agent sẽ kiểm tra xâu public và xác định là trùng với community read-only, như vậy manager có community cho phép đọc giá trị. Tuy nhiên agent còn phải thực hiện xác thực manager và xét đến khả năng cho phép truy cập dựa trên MIB của biến mới quyết định là manager có thể đọc giá trị của biến đó hay không. Vì community có bản chất là mật khẩu nên cần thay đổi giá trị mặc định. Khi cấu hình SNMP agent, ta phải cấu hình địa chỉ nơi nhận trap. Thêm vào đó, vì SNMP community được gửi đi dưới dạng thuần văn bản, ta nên cấu hình agent gửi trap authentication-failure khi ai đó cố gắng truy vấn thiết bị với một community không chính xác.

Do sử dụng community như là mật khẩu nên SNMPv1 là giao thức rất yếu về bảo mật. Các gói tin được gửi đi dưới dạng thuần văn bản nên không chống đỡ được kiểu tấn công bằng cách nghe lén – sniffer.

SNMPv2 cố gắng giải quyết vấn đề này dựa trên các cách tiết cận chặt chẽ hơn. Một phiên bản gọi là SNMPv2 party-based tiếp cận theo hướng: tuy từng yêu cầu về xác thực và tính mật mà có thể sử dụng các kênh khác nhau để trao đổi thông tin. Hình 2.3. minh họa 3 kênh với các yêu cầu về bảo mật khác nhau bằng cách thay thế community (chia sẻ dùng chung giữa tất cả các bên tham gia) bằng party (chia thành nhiều nhóm, mỗi nhóm trao đổi theo cách thức riêng). Kênh thứ nhất sử dụng để truyền số liệu không quan trọng giữa A và B, do vậy sử dụng cặp Party 1.A và Party 1.B có tính chất mở - open. Kênh thứ hai để đọc và thay đổi cấu hình thông thường, yêu cầu

có xác thực nên sử dụng cặp Party 2.A và Party 2.B có tính chất xác thực – authenticated. Kênh thứ ba truyền cấu hình rất quan trọng, yêu cầu phải bảo mật nên sử dụng cặp Party 3.A và Party 3.B có tính mật. Tuy nhiên, với nhiều nỗ lực để tăng cường bảo mật trong SNMP đã dẫn tới ba phiên bản không tương thích với nhau là: SNMPv2p hay SNMPv2 party-based, SNMPv2u hay SNMPv2 user-based và SNMPv2*. Các phiên bản này đã thất bại trong việc tìm được sử hỗ trợ của các nhà sản xuất và dừng lại ở bản thảo, rồi chuyển sang quá khứ. Cuối cùng, một sự thỏa hiệp được thực hiện và kết quả là chuẩn SNMPv2c hay SNMP community-string-based. Đây là một bước tụt lùi khi quay lại sử dụng community như SNMPv1, tuy nhiên chuẩn này lại được hỗ trợ của IETF cũng như cách nhà sản xuất. Trong tài liệu này, khi nói đến SNMPv2 là ám chỉ SNMPv2c. Vấn đề về bảo mật chỉ được giải quyết triệt để chỉ khi xuất hiện phiên bản SNMPv3.

SMNPv3 ra đời chủ yếu để giải quyết vấn đề bức xúc về bảo mật trong hai phiên bản trước [Stallings 98]. Phiên bản này không có sự thay đổi về giao thức, không có thêm PDU mới, chỉ có một vài quy chuẩn mới, khái niệm và thuật ngữ mới, cũng không nằm ngoài việc làm tăng tính chính xác [Stallings 98]. Thay đổi quan trọng nhất trong SNMPv3 này là sử dụng khái niệm SNMP entity thay cho cả manager và agent. Mỗi SNMP entity gồm một SNMP engine và một hoặc nhiều SNMP application. Sự thay đổi về khái niệm này quan trọng ở chỗ thay đổi về kiến trúc, tách biệt hai phần của hệ thống SNMP, giúp cho việc thực hiện các chính sách bảo mật. Điểm quan trọng là SNMPv3 vẫn tương thích ngược với các phiên bản trước.

II.2. Cấu trúc thông tin quản trị (SMI) và cơ sở thông tin quản trị (MIB)

Để manager và agent có thể trao đổi thông tin cho nhau thì giữa manager và agent phải có định nghĩa về khuôn dạng dữ liệu trao đổi chung.

Cấu trúc thông tin quản trị (Structure of Management Information-SMI) được định nghĩa trong RFC 1155 xác định phương pháp cơ bản để định danh các đối tượng được quản trị và hành vi của chúng [perkins]. Agent sở hữu danh sách các đối tượng nó giám sát. Các đối tượng này có thể là trạng thái hoạt động (up/down/testing) của một interface của router, số gói tin truyền/nhận của interface… Danh sách này cũng cung cấp thông tin mà trạm quản trị có thể sử dụng để xác định trạng thái của thiết bị chứa agent.

Lưu ý là SMI chỉ là cú pháp để định nghĩa các đối tượng được quản trị, còn các đối tượng được quản trị định nghĩa bằng SMI gọi là Cơ sở thông tin quản trị (Management Information Base-MIB. MIB có thể được coi là cơ sở dữ liệu về các đối tượng được quản trị mà agent giám sát. Tất cả trạng thái hay thông tin thống kê có thể truy nhập bởi trạm quản trị đều được định nghĩa trong MIB.

Phiên bản đầu tiên của SNMP đưa ra MIB-I định nghĩa trong RFC 1066, Phiên bản tiếp theo (MIB II) được đưa ra vào năm 1991 (RFC 1213) cùng với SNMPv2 bổ sung thêm danh sách các các thông tin cơ bản, bắt buộc phải có đối và đã được chuẩn hóa trên mọi thiết bị tương thích SNMP.

MIB được cấu trúc dạng hình cây [perkins]. Trong cấu trúc này, tất cả các biến SNMP hay các đối tượng được mô tả dưới dạng *cành và lá* và được đặt tên theo kiểu OBJECT IDENTIFIER (OID) của ASN.1. Các đối tượng quản lý được tập hợp lại thành các nhóm liên hệ logic với nhau tính từ gốc (root). Từ điểm root, ta sẽ có các cành tiếp theo ở mức 1: iso (l), ccitt (0) and joint-

iso-ccitt (2), trong đó, iso nhánh theo quy định của tổ chức International Organization for Standardization, ccitt là của Intemational Telegraph and Telephone Consultative Cornmittee, và joint-iso-ccitt giành cho các quy định được quản lý bởi cả hai tổ chức ISO và CCITT [ietf].

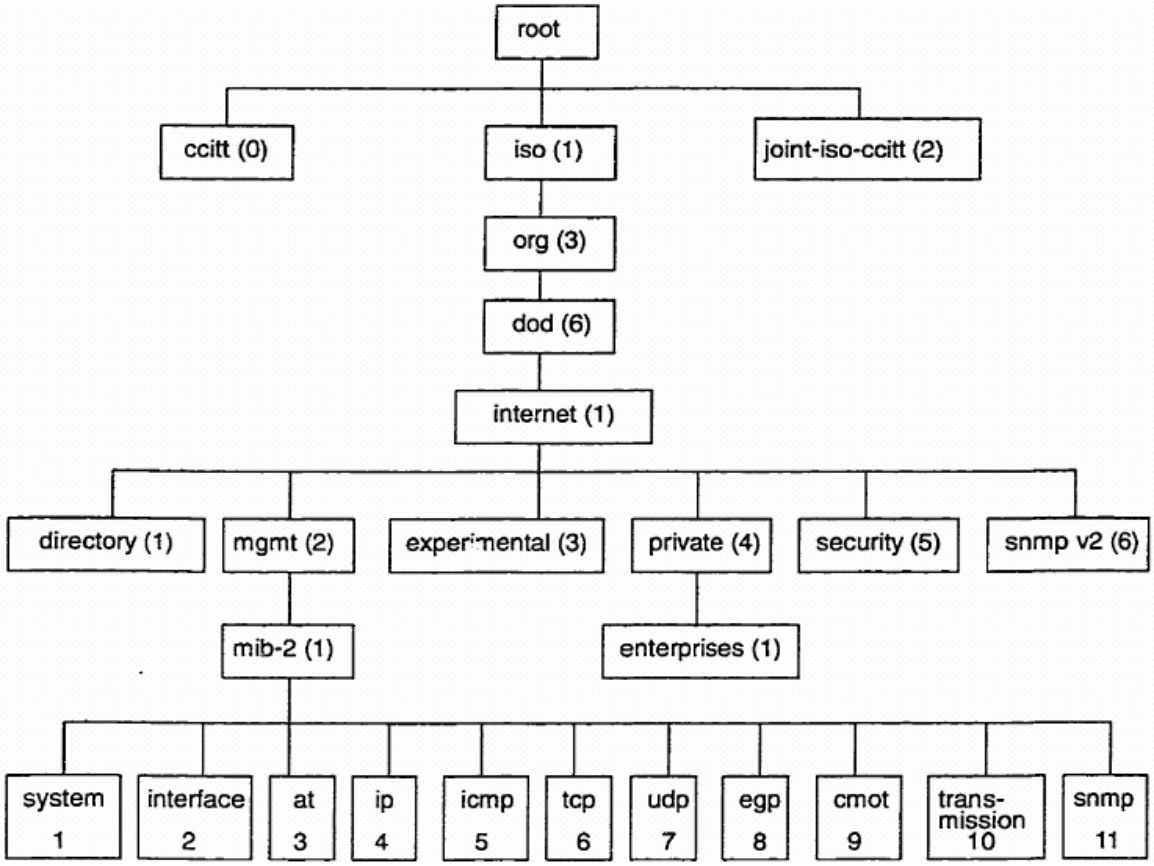
Một agent có thể cài đặt nhiều MIB, nhưng tất cả các agent đều phải cài đặt một MIB đặc biệt gọi là MIB-II (RFC 1213). Chuẩn này định nghĩa những rất nhiều thông tin chung về hệ thống (vị trị của thiết bị, người liên hệ…), về số liệu thống kê của interface (tốc độ, MTU, lượng octet gửi, lượng octet nhận…). Mục đích của MIB-II là cung cấp các thông tin quản trị chung về TCP/IP. MIB-I là phiên bản đầu tiên nhưng từ khi MIB-II phát triển nó, nó đã không còn được sử dụng nữa. Để có thể giám sát được những vấn đề cụ thể liên quan đến các công nghệ mạng khác nhau, các tính năng đặc biệt của các hãng khác nhau thì agent và manager phải được cài đặt các MIB tương ứng. Chẳng hạn, một số bản thảo và đề nghị được đưa ra để quản trị các công nghệ như Frame Relay, ATM, FDDI và các dịch vụ như email, DNS …:

- ATM MIB (RFC 2515)
- Frame Relay DTE Interface Type MIB (RFC 2115)
- BGP Version 4 MIB (RFC 1657)
- RDBMS MIB (RFC 1697)
- RADIUS Authentication Server MIB (RFC 2619)
- Mail Monitoring MIB (RFC 2789)
- DNS Server MIB (RFC 1611)

Ngoài ra, một điểm rất mở nữa của SNMP là các hãng sản suất và cá nhân đều có thể định nghĩa các MIB cho riêng mình. Ví dụ, một agent trong một router được cài đặt MIB-II (bắt buộc) và các MIB cho các loại interface mà nó có (như RFC 2515 cho ATM và RFC 2115 cho Frame Relay). Ngoài ra, router này còn có thêm một số chức năng mới rất hữu ích trong quản trị mà chữa được đề cập đến trong các MIB chuẩn nào, do vậy nhà sản xuất định nghĩa MIB của riêng mình, cài đặt các đối tượng được quản trị cho các chức năng mới này. Có rất nhiều các lại MIB, nhưng mỗi agent chỉ được hỗ trợ một số MIB, do vậy ở trạm quản trị ta cũng chỉ cần cài đặt các MIB cần thiết.

II.2.1. Nhóm hệ thống trong MIB II

Thông tin trong nhóm hệ thống có ý nghĩa rất quan trọng trong quả trị mạng. Như đã mô tả trong RFC 1213, nhóm hệ thống đưa ra các thông tin về hệ thống quản trị. Nhóm này bao gồm bảy đối tượng (xem Hình II.4 NhómCấu trúc của MIB). Nếu không được cấu hình để chưa các thông tin này thì agnt sẽ trả về giá trị độ dài bằng 0.



Hình II.4 NhómCấu trúc của MIB

Bảng II-1 Khuôn dạng một số đối tượng

Đối tượng	Khuôn dạng	Truy nhập	Mô tả
SysDescr	Displaystring (size 0 ... 255)	RO	Tên, phiên bản của hệ thống
SysObjectID	OBJECT IDENTIFIER	RO	Tên nhà sản xuất, hoặc định danh của nhà quản trị phân đoạn mạng
sysUpTime	TimeTicks	RO	Thời gian tính từ khi phần quản trị mạng được khởi động
syscontact	Displaystring (size 0 ... 255)	RW	Thông tin về người quản trị thiết bị
SysName	Displaystring (size 0 ... 255)	RW	Tên của người quản trị
SysLocation	Displaystring (size (0 ... 255)	RW	Vị trí, nơi đặt thiết bị
SysServices	INTEGER (0 … 127)	RO	Mô tả các dịch vụ mà thiết bị cung cấp

* RW - đọc/Ghi (Read & Write) RO – Chỉ đọc (Read Only)

II.2.2. Nhóm các tổ chức trong MIB-II

Trong hình trên, chúng ta thấy nhóm các đối tượng "tổ chức" - enterprise được xếp ở dưới nhánh "Private". Nhóm Enterrprise được sử dụng để cho phép các tổ chức (nhà sản xuất) cung các các hệ thống mạng có thể đăng ký cho các sản phẩm của mình và công bố để các nhà quản trị mạng có thể sử dụng chúng trong tổ chức mạng của mình.

Các cành ở trong nhóm enterrprise được sử dụng cho các tố chức đăng ký các OID theo mục đích riêng của tổ chức đó.

Nhiều tổ chức đã tự tạo lập cho riêng mình một MIB như là Proteion, IBM, CMU, Cisco vv…

Bảng II-2 Tên của các tổ chức và OID

Tên của tổ chức	OID
Dự phòng	1.3.6.1.4.1.0
Proteon	1.3.6.1.4.1.1
Cisco	1.3.6.1.4.1.9
NSC	1.3.6.1.4.1.10
Novell	1.3.6.1.4.1 23
…	
Sun Microsystems	1.3.6.1.4.1.42
…	

Mỗi một MIB của các tổ chức cũng được định nghĩa theo chuẩn SMI và ASN.1. Ví dụ: file định dạng CISCO-SMI.my của hãng Cisco System Inc có dạng như sau:

```
…
ciscoProducts OBJECT IDENTIFIER ::= { cisco 1 )
-- OBJECT-IDENTIY
        Status: mandatory
        Descr:
                ciscoProducts is the root OBJECT IDENTIFIER from which sysObjectID
                values are assigned.
                Actual values are defined in CISCO-PRODUCTS-MIB.

local OBJECT IDENTIFIER ::= { cisco 2 )
```

```
-- OBJECT-IDENTITY
        Status: mandatory
        Descr:
                Subtree beneath which pre-10.2 MIBS were built.
…
```

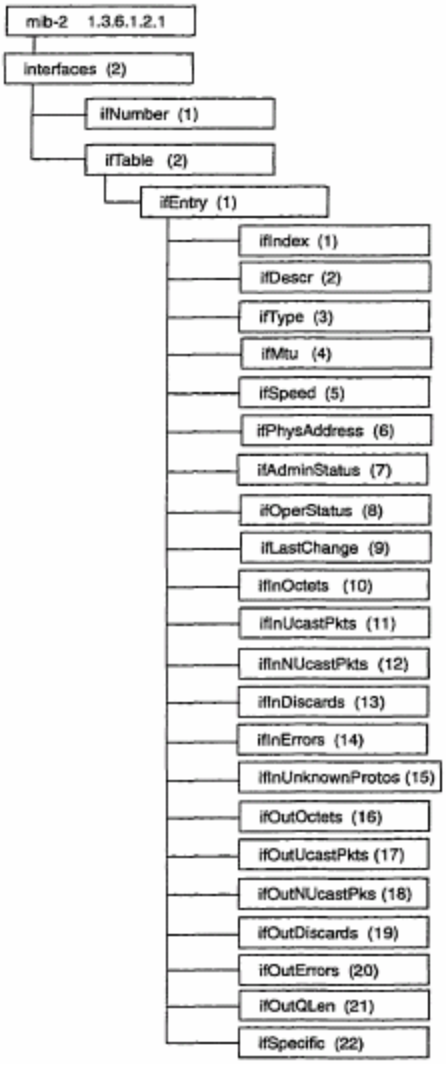
II.2.3. Nhóm giao diện (interface trong MIB-II)

Các thông tin quan trong được chưa trong nhóm giao diện (interface) như là số lương các giao diện vật lý, kiểu, loại giao diện được lắp đặt trong thiết bị cũng như số lượng các giao diện đang hoạt động (up) cũng như số lượng các giao diện đang tắt (down).

Hình II-1 minh họa cây OID bên dưới nhóm giao diện và các nhánh, lá bên dưới

Bảng II-3 Một số định nghĩa của các OID

Đối tượng	khuông dạng	truy nhập	Mô tả
IfNumber	INTEGER	RO	Số lượng các giao diện mạng
IfTable	sequence of ifEntry	NA	Danh sách các điểm vào của giao diện
Iflndex	SEQUENCE	NA	điểm vào của một giao diện có chứa các đối tượng là các giao diện lớp dưới
…			
IfOutOctets	Counter	RO	Tổng số octes đã được chuyển qua giao diện, kể cả các ký tự khung



Hình II-1 Cấu trúc nhóm các giao diện trong MIB-II

II.3. Đặc tả SNMP

Theo RFC 1157, giao thức quản trị mạng được định nghĩa là một giao tiếp tầng ứng dụng, thông qua đó để theo dõi hoặc thay đổi các biến (đối tượng điều khiển) trong MIB của các Agent.

SNMP cung cấp 03 tác vụ cơ bản là: GET, SET và TRAP, thông qua đó, các thiết bị quản lý mạng có thể yêu cầu nhận, thay đổi các cài các giá trị điều khiển của Agent cũng như được thông báo về các sự kiện bất thường xảy ra tại thiết bị điều khiển.

II.3.1. Khuôn dạng của SNMP

Trong khuôn khổ của SNMP, liên lạc giữa các thực thể được thực hiện thông qua việc trao đổi các thông điệp SNMP được biểu diễn dưới dạng các gói tin UDP trên nguyên tắc mã hóa cơ bản của ASN.1. Các thông điệp mang theo mình thông tin về phiên bản SNMP hiện đang sử dụng, community name được sử dụng để "xác thực" và một trong năm kiểu dữ liệu (GetRequestPDU, GetNextRequestPDU, SetRequestPDU, GetResponsePDU, TrapPDU)

(1) SNMP message:

Version	Community	SNMP PDU

(2) GetRequest PDU, GetNextRequest PDU, và SetRequest PDU:

PDUtype	RequestID	0	0	variable-bindings

(3) GetResponse PDU:

PDUtype	RequestID	ErrorStatus	Errorindex	variable-bindings

(4) Trap PDU

PDUtype	Enterprise	AgentAddr	GenericTrap	specific Trap	time stamp	Variable-bindings

Bảng II-4 Mô tả các trường của SNMP

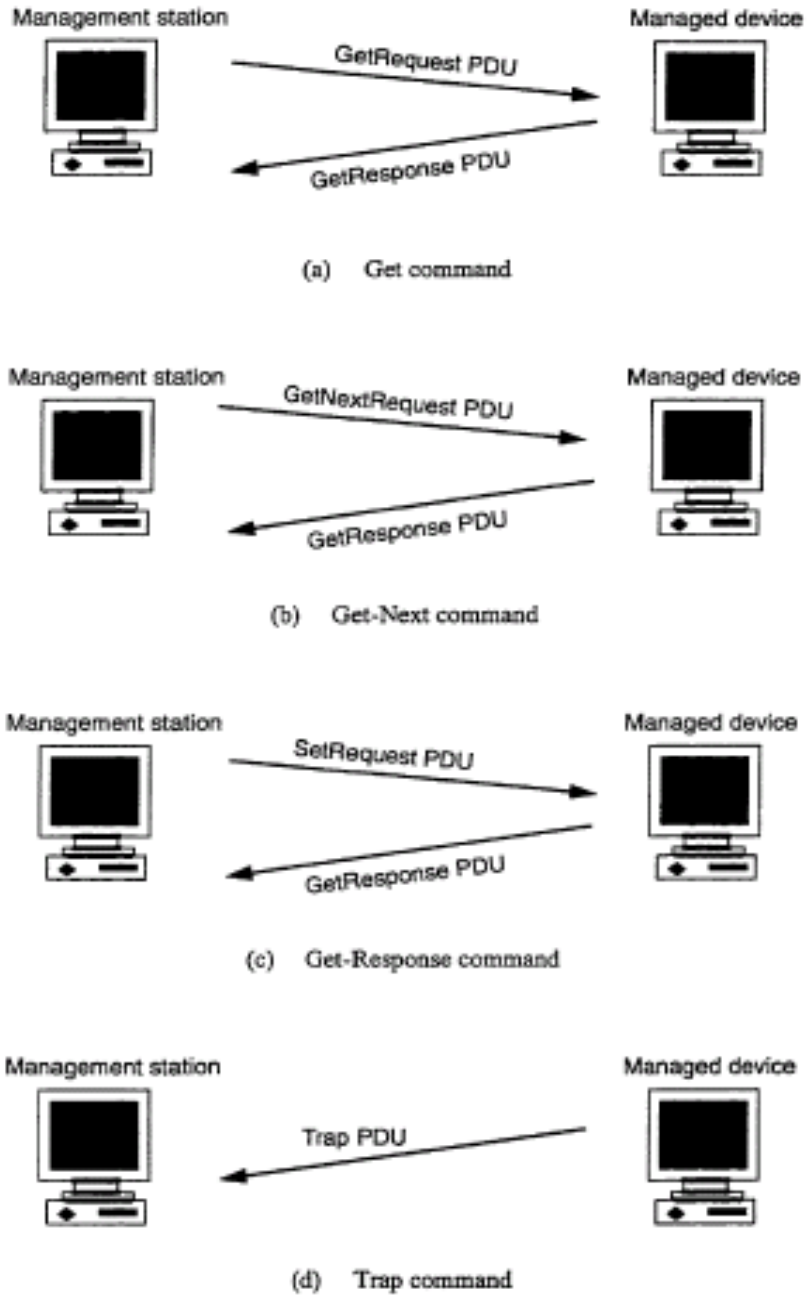
Tên	Mô tả
Community	Được sử dụng như là một dạng mật khẩu để xác thực các gói tin SNMP. từ khóa "public" thường được sử dụng mặc định
ErrorStatus	Giá trị nguyên được sử dụng để thông báo về trạng thái lỗi xuất hiện khi xử lý một yêu cầu. Các giá trị có thể là: • noError (0) • tooBig (l) • noSuchName (2) • badVaIue (3) • readOnly (4) • genEn(5)
ErrorIndex	Giá trị được sử dụng khi ErrorStatus khác không để mô tả bổ sung các

	thông tin về lỗi
GenericTrap	Giá trị nguyên mô tả sự kiện xảy ra ở thiết bị. Chúng có thể là: • ColdStart(0); • WarmStart(1) • LinkDown(2) • LinkUp(3); • AuthenticationFailure(4) • EgpNeighborLoss(5) • EnterpriseSpecific(6)
SpecificTrap	Sự kiện xảy ra không nằm trong quy định của nhà sản xuất

II.3.2. Các lệnh SNMP và trình tự thực hiện

Như chúng ta đã biết, SNMP có 5 lệnh cơ bản là: Get, Get-Next, Get-Response, Set và Trap. Tương ứng với năm lệnh đó là năm gói tin: GetRequestPDU, GetNexRequestPDU, GetResponsePDU, SetRequestPDU và TrapPDU.

Khuôn dạng của chúng như đã được mô tả trong phần trước. Phương thức vận hành của chúng được mô tả ở hình sau:

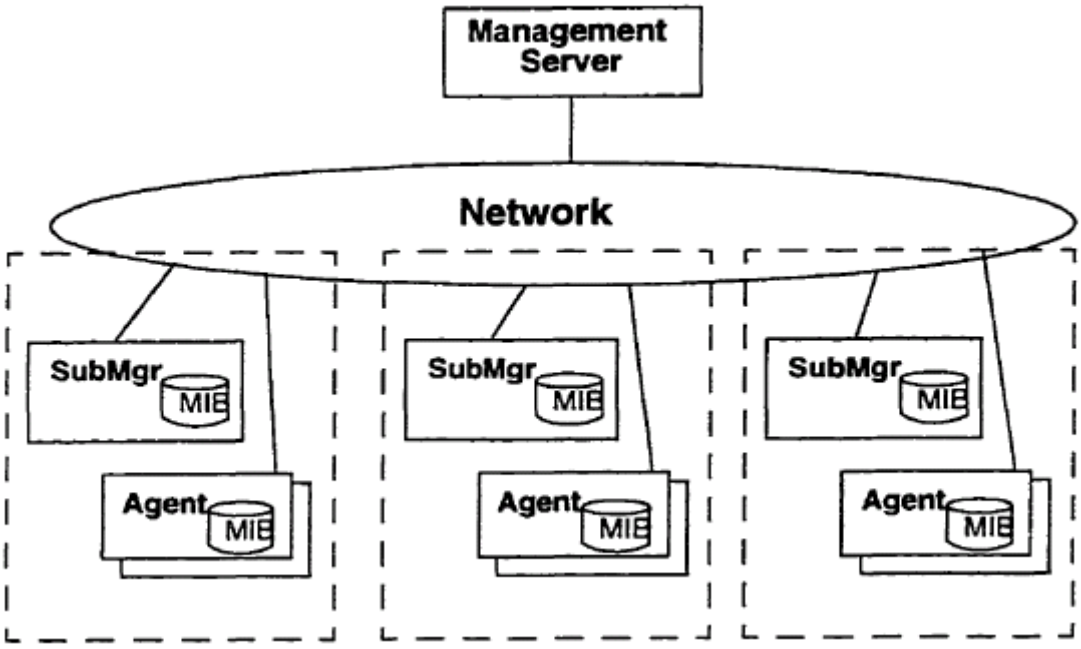


Hình II.5 Chu trình SNMP

II.3.3. Kiến trúc quản trị mạng

Hình II.2 đưa ra một mô hình đơn giản trong quản lý mạng nội bộ. Tham gia vào mô hình đó chỉ có hai thực thể đơn giản là Trạm quản lý và thiết bị được quản lý (Agent). Tất nhiên là cả hai thực thể đều phải dùng giao thức quản trị mạng để liên lạc với nhau (SNMP) và thông tin cần gửi là các giá trị của

các biến trong MIB. Sự thắng thế của mô hình tính toán phân tán đã kéo theo phong trào phân tán hóa việc quản trị mạng [Mazumdar]. Một hệ thống quản trị mạng phân tán thông thường sẽ có một số trạm làm việc tương tác với nhau thông qua liên mạng, trong đó, các trạm làm việc này sẽ đóng vai trò quản trị mạng của phân đoạn mạng đó, hoặc của đơn vị (thực thế) đó. Trong mô hình này, chúng ta ta cũng sẽ thấy có một trạm quản trị chính làm nhiệm vụ tương tác với trạm quản trị địa phương và trách nhiệm quản trị chính sẽ được giao cho các trạm quản trị địa phương này. Tuy theo cấu hình và yêu cầu cụ thể mà Trạm quản lý trung tâm có thể làm việc trực tiếp với các Agent ở mức thấp hơn.



Hình II.6 Kiến trúc quản trị hệ thống phân tán thông thường

II.3.4. Những hạn chế của SNMP

SNMP được thiết kế theo hướng đơn giản hóa các tác vụ nên có một số các điểm hạn chế:

- Chỉ có một gói thông tin đối với từng yêu cầu, không phù hợp với các mạng phức tạp, có nhiều sữ liệu cần phải kiểm tra [Stallings 96]

- SNMP là giao thức phi hồi đáp, nghĩa là agent không thể chắc chắn là các gói tin trap do mình gửi đi đến được đích. Bốn trong năm thông điệp của SNMP là các nghi thức hồi-đáp đơn giản (máy trạm gửi yêu cầu, máy agent phản hồi kết quả) nên SNMP sử dụng giao thức UDP. Điều này nghĩa là một yêu cầu từ máy trạm có thể không đến được máy agent và hồi đáp từ máy agent có thể không trả về cho máy trạm. Vì vậy máy trạm cần cài đặt thời gian hết hạn (timeout) và cơ chế phát lại [Stallings 96].
- Tính bảo mật kém, tên cộng đông (community) được sử dụng như là mật khẩu để xác thực các thông điệp SNMP [Stallings 98]. Quản lý mạng dựa trên SNMP có mức bảo mật thấp. Vì dữ liệu không mã hóa và không có thiết lập cụ thể để ngưng bất kỳ truy nhập mạng trái phép nào khi tên community name và địa chỉ IP bị sử dụng để gửi yêu cầu giả mạo tới agent. Do đó, SNMP phù hợp với mô hình quan trắc hơn là với mô hình điều khiển.
- Chỉ có các cấu dữ liệu đơn giản. Không phù hợp với các yêu cầu về giá trị hay kiểu của đối tượng
- Không hỗ trợ giao tiếp từ trạm quản lý đến trạm quản lý
- Không hỗ trợ các lệnh thực thi tức thời.
- Quản lý mạng dựa trên SNMP có mức khả chuyển thấp giữa các kiến trúc khác nhau. Vì cấu trúc thông tin quản lý của SNMP chỉ hỗ trợ giới hạn các kiểu dữ liệu.
- Không thân thiện.

Nhiều nhược điểm này đã được khắc phục hoặc giải quyết trong các phiên bản tiếp theo của SNMP (version 2, 3)

Chương III.Quản trị mạng trên web với CGI và CORBA

III.1. Chuẩn CGI

CGI là viết tắt của từ tiếng anh Common Gateway Interface. CGI là một giao diện chuẩn cho phép trao đổi thông tin giữa phần mềm Web Server với các chương trình (ứng dụng) bên ngoài [Weinman].

Nguyên thuỷ, Web server chỉ là một phần mềm xử lý các yêu cầu http đơn thuần nhận được và trả về các trang html với các nội dung tĩnh. Do sự phát triển của mạng và nhu cầu tương tác cao của người sử dụng đối với các nguồn thông tin trên web, thông tin có tính chất "động" như truy vấn cơ sở dữ liệu…

III.1.1. CGI - sự mở rộng của HTTP

Chuẩn CGI đã được đưa ra và mô tả bởi các tác giả chính của HTTP server: Tony Sander, Ari Luotonen, George Phillips và John Franks. Ban đầu dịch vụ của các HTTP server khá bị giới hạn và chúng chỉ có thể trả về cho các tình duyệt web các tài liệu HTML cố định (tĩnh). Để đáp ứng các yêu cầu ngày càng tăng về các tính năng của web như là cung cấp các thông tin cập nhật (động) cho trình duyệt client, các tác giả nêu trên đã đưa ra một phương pháp mới, mở rộng các dịch vụ và năng lực từ gốc rễ của các Web server. Đó chính là chuẩn CGI [Weinman].

CGI là một giao diên đơn giản giành cho việc chạy các chương trình bên ngoài (CGI script - các kịch bản CGI) bên dưới nền HTTP server. Khi có một yêu cầu của khách hàng được gửi đến đến Web Server thông qua trình duyệt Web, Web Server sẽ gửi tới CGI gateway. CGI sẽ thực hiện công việc

của mình và chuyển thông tin về cho Web Server dưới dạng chuẩn HTML và Web server sẽ gửi tiếp các thông tin này về cho khách hàng [Tittel96].

Sau đây là tóm lược bốn bước xử lý của CGI:

- Bước 1: Xử lý dữ liệu được truyền từ Client tới Server.
- Bước 2: Server sẽ hướng các yêu cầu mà Client gửi tới đến các chương trình CGI để thực hiện.
- Bước 3: Gửi lại các dữ liệu và kết quả mà chương trình CGI thực hiện trở lại cho Server.
- Bước 4: Server gửi lại dữ liêu mà nó nhận từ chương trình CGI cho Client.

III.1.2. Các đặc trưng của CGI

CGI cho phép bạn mở rộng các chức năng của Web server, là một phương thức để cho HTTP server trao đổi thông tin với chương trình người dùng.

Trên quan điểm lý thuyết: CGI sẽ xử lý dữ liệu đưa vào thông qua browser và trả lại thông tin cho người sử dụng [Tittel96].

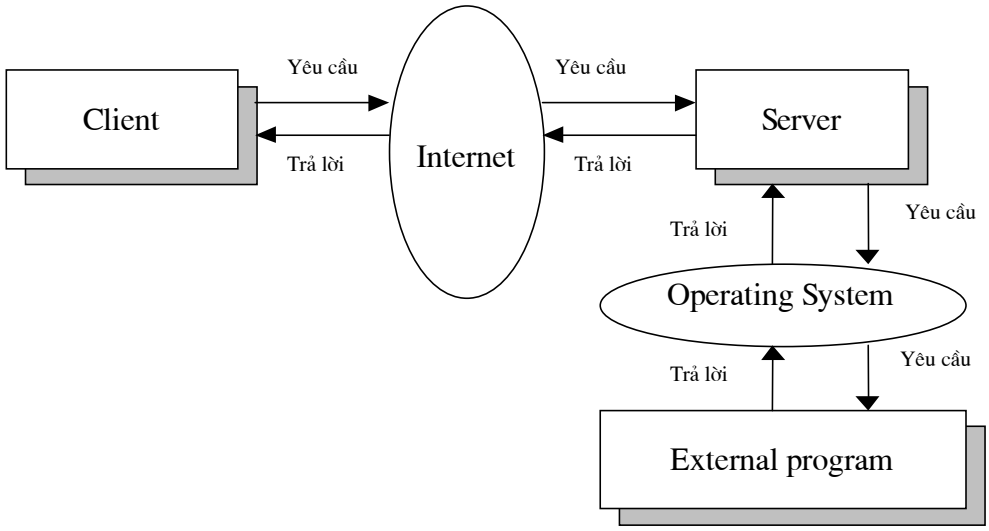
Trên quan điểm thực hành: CGI là trình giao diện cho phép người lập trình viết chương trình thực hiện truyền thông với Server [CGI201].

- CGI cung cấp cách giải quyết vấn đề một cách dễ dàng và đơn giản.
- Giao thức CGI được định nghĩa theo một chuẩn, nó cung cấp cách truyền thông với Web server.
- Sử dụng CGI bạn không cần dùng nhiều tri thức đặc biệt, có thể viết chương trình với bất kỳ ngôn ngữ máy tính nào để thực hiện giao tiếp và truyền thông với Web server [CGI201].

- Sự truyền đạt của CGI là dựa trên các chuẩn vào ra.

III.1.3. Mô hình quan hệ Client/Server sử dụng CGI

Thông thường, một hệ thống khách/phục vụ được gọi là hệ thống quan hệ cấp 2. Trong hệ thống này, giao diện người dùng và các quan hệ logic nằm về bên thứ nhất. Các chức năng phục vụ công việc và dữ liệu trên server thuộc về bên thứ hai. Đó có thể được xem là các hệ nền (platform) và các hệ thống mạng phần cứng cũng như phần mềm liên kết khách/phục vụ được gọi là các trung gian.



Hình III-1 Chu trình thực hiện một CGI request

CGI tạo ra cầu nối giữa web server và các dịch vụ internet khác như là cập nhật số liệu cho các server mạng back-end hoặc quản lý máy tính khác trong mạng đằng sau web server. Trong trường hợp này, CGI hoạt động như một trung gian (middleware) giữa web server và cơ sở dữ liệu bên ngoài hoặc các dịch vụ thông tin khác [CGIPerl]. Với sự tham gia của CGI, chúng ta đã có quan hệ tay ba trong kiến trúc Client/Server được cấu thành bởi cơ sở dữ liệu bên ngoài và các hệ thống dịch vụ thông tin. Hình III-2 mô tả kiến trúc ba lớp mới này. Thông thường, lớp đầu tiên là màn hình của người sử dụng.

Lớp giữa được tạo thành bởi các đối tượng server, là các dữ liệu cố định và các khối logic thực thi. Lớp thứ ba là các dịch vụ truyền thống thông thường. Trong Hình III-2 , lớp ở giữa là web server được "tăng cường" thêm bởi dịch vụ ứng dụng CGI .

Các nhà cung cấp dịch vụ mạng thường sử dụng công nghệ này để kết nối các trình duyệt web của họ đến các thực thể quản trị để theo dõi tình trạng của các thiết bị. Công nghệ này đã được ứng dụng trong web site của Bay, cho phép các trình duyệt được truy nhập vào các ứng dụng quản lý Optiviti.

III.1.4. Cách thức và phương pháp truyền dữ liệu trong CGI

Với việc sử dụng nhiều kỹ thuật khác nhau client có thể truyền đối số hoặc dữ liệu tới chương trình gateway thông qua HTTP server. Chương trình gateway thay vì phải bắt đầu với một chương trình tĩnh (static program) ở thời điểm hiện tại thì nó sẽ được thay thế với một thực thể động (dynamic entity) để tiến hành trả lời tới người sử dụng cuối cùng.

Có bốn phương pháp chủ yếu để để server liên lạc với các CGI script. Ba phương pháp đầu tiên là cách để các CGI script nhận được thông tin từ server và cách cuối cùng là để các CGI script gửi thông tin cho server.

Sau đây, chúng ta sẽ lần lượt xem xét các phương pháp đó: phương pháp biến môi trường (Enviroment variables), tham số dòng lệnh (Command Line) bằng hoặc bằng dòng nhập chuẩn (Standard input).

III.1.4.1 Phương pháp thứ nhất - Biến môi trường

Các biến môi trường là các biến được thiết lập bởi phần mềm server và có thể được truy nhập từ các chương trình bên ngoài. Các biến này chứa đựng các thông tin về server, chương trình bên ngoài và yêu cầu của khách hàng. Xem bảng

Bảng III-1 Các biến môi trường chuẩn

Tên biến	Mô tả
AUTH_TYPE	kiểu xác thực truy nhập
CONTENT-LENGTH	độ lớn của các dữ liệu (thực thể) tính theo byte
CONTENT-TYPE	Định kiểu MIME của thực thể: application/octet-stream, text/pain, …
GATEWAY-INTERFACE	Phiên bản CGI của server
HTTP_(string)	Dữ liệu header của khách
PATH-INFO	đường dẫn mở rộng để các CGI scrript biên dịch
PATH-TRANSLATED	Ánh xạ từ ảo đến đường dẫn thực trong hệ thống file
QUERY-STRING	Xâu tìm kiếm đã được viết dưới dạng chuẩn URL
REMOTEADDR	địa chỉ IP của phía đưa ra yêu cầu
REMOTEHOST	Tên miền đầy đủ của phía đưa ra yêu cầu
REMOTEINDENT	Dữ liệu phân biệt về phía kết nối đến server
REMOTE-USER	Định danh người dùng do phía client gửi
REQUEST_METHOD	Yêu cầu của phía client (“GET” hay là “POST”…)
SCRIPT-NAME	Univeral Resource Identifier (URI) - đường dẫn để nhận biết một CGI script.
SERVER-NAME	Tên server, phần tên máy chủ trong URI hoặc DSN
SERVER-PORT	Cổng dịch vụ nhận yêu cầu
SERVER-PROTOCOL	Tên và phiên bản của giao thức yâu cầu
SERVER-SOFWARE	Tên và phiên bản của phần mềm phục vụ trả lời yêu cầu

III.1.4.2 Phương pháp thứ hai – Dòng lệnh

Phương pháp này thường được sử dụng cho các truy vấn HTML ISINDEX. Thông tin có thể được chuyển cho các chương trình bên ngoài thông qua các tham số dòng lệnh khi chương trình được gọi (để chạy).

III.1.4.3 Phương pháp ba – dòng dữ liệu vào chuẩn (standard input)

Khi phía client sử dụng phương pháp POST hoặc PUT để gửi các yêu cầu đến các chương trình bên ngoài, thông tin sẽ được server chuyển cho các chương trình bên ngoài thông qua dòng nhập chuẩn (standard input). Nếu sử dụng GET, dữ liệu sẽ được đưa vào biến môi trường QUERY_STRING. Server cũng sẽ thiết lập biến CONTENT_TYPE và CONTENT_LENGTH

tương ứng về định dạng kiểu dữ liệu MIME và độ lớn của dữ liệu (tính bằng byte).

III.1.4.4 Phương pháp dòng dữ liệu ra chuẩn (standard input)

Khi chương trình bên ngoài sau khi thực hiện yêu cầu, thông tin sẽ gửi dữ liệu ngược lại cho web server (để web server chuyển tiếp cho trình duyệt client) thông qua dòng dữ liệu ra chuẩn (standard output). Các dữ liệu này phải được viết ở khuôn dạng HTML.

III.1.5. Lập trình CGI

Chương trình CGI gateway có thể viết bằng một ngôn ngữ lập trình nào đó, chẳng hạn như C/C++, Visual Basic, Perl đây là một chương trình thực hiện được (executable). Mỗi khi có yêu cầu thực hiện CGI từ phía khách hàng thì máy chủ (server) sẽ tạo một tiến trình (process) mới cho gateway và truyền thông tin từ khách hàng cho tiến trình này.

Khi lập trình CGI, có hai nguyên tắc sau cần phải được tuân thủ:

1. Kết quả trả về cho Client (dữ liệu hay text) phải được ghi ra dòng dữ liệu ra chuẩn (Standard output, STDOUT);
2. Các dữ liệu ra cần phải được bắt đầu bằng chuỗi “Content-type” và một dòng trống

Sau đây là một ví dụ về một đoạn mã chương trình CGI bằng C:

```
#include <stdlib.h>
void main(void)
{
      printf(“Content-type: text/html\n\n”);
      printf(“Hello World!”);
}
```

Như đã đề cập ở phần trước, chương trình chỉ đơn giản gửi thông điệp ra STDOUT theo đúng yêu cầu của CGI: một dòng định kiểu "MIME type: text/html" và tiếp theo là một dòng trống.

III.1.6. Cài đặt các chương trình CGI

Sau đây là một số bước cơ bản để cài đặt một chương trình CGI trên server và cách thực hiện chúng

Yêu cầu đầu tiên là người sử dụng phải có quyền truy nhập web server và có quyền ghi vào thư mục có tên là cgi-bin, nơi đặt các chương trình cgi. Các chương trình CGI sẽ được web server gọi ra thực hiện tự động khi web server nhận được yêu cầu từ phía người sử dụng.

Thứ hai, chương trình CGI cần phải chạy được và có thể truy nhập từ phía người dùng bình thường.

Đối với các chương trình dùng ngôn ngữ Perl, phần mềm Perl cũng phải được cài đặt trong mạng [Tittel96].

Đối với các chương trình sử dụng Java, trong thư mục cgi_bin cần phải có một file thực thi với dòng lệnh:

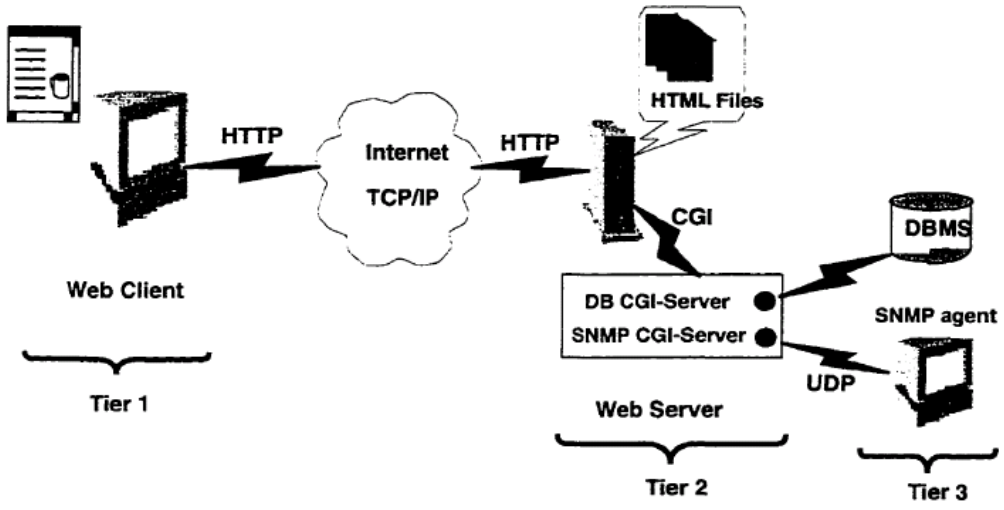
```
java ProgramFileName
```

trong đó ProgramFileName là tên của java class. Trong các hệ thống unix, file này có thể được viết dưới dạng ngôn ngữ kịch bản (shell script). Đối với họ windows, file nói trên có thể là một file batch. Trong mọi trường hợp, phần mềm hỗ trợ java phải được cài đặt trong cùng một thư mục [CGI2].

Trên đây là các bước cần thiết để cài đặt các chương trình CGI. Sau các bước cài đặt cần thiết, các chương trình CGI sẽ được gọi ngay khi có yêu cầu từ các applet code trong trình duyệt client.

III.1.7. Mô hình quản trị mạng ba bên sử dụng Web - CGI

Trên Hình III-2 là sơ đồ tương tác của một hệ thống quản trị mạng qua web sử dụng CGI. Như đã để cập ở phần trước, web server sẽ chuyển các thông tin cho các chương trình CGI back-end thông qua các biến môi trường, dòng nhập dữ liệu chuẩn và sẽ yêu cầu thực hiện các chương trình (thường là các chương trình SNMP server) này tùy theo yêu cầu cụ thể. Các chương trình này được khởi chạy và sẽ đọc các dữ liệu từ dòng nhập dữ liệu chuẩn. Sau đó, chương trình này sẽ liên hệ với các SNMP agent để lấy các thông tin cần thiết. Sau khi hoàn thành, chương trình sẽ trả thông tin lại cho web server dưới dạng mã HTML thông qua giao diện STDOUT. Cuối cùng, web server sẽ trả lại file HTML này cho Client.



Hình III-2 Mô hình web Client/Server ba bên sử dụng CGI

III.2. Chuẩn CORBA

CORBA là viết tắt của cụm từ Common Object Request Broker Architecture - được đưa ra bởi tổ chức quản lý đối tượng (Object Management Group - OMG) với mục đích tạo ra một môi trường (khung) làm việc chung cho các ứng dụng hướng đối tượng [Rosenberger]. Chuẩn này nhắm tới một môi trường tính toán phân tán không đồng nhất và tạo ra một cơ chế liên lạc chuẩn giữa các đối tượng trong môi trường không đồng nhất đó [Rosenberger].

III.2.1. Giới thiệu chuẩn CORBA

CORBA được đưa ra nhằm giải quyết hai vấn đề cơ bản nhất mà ngành công nghiệp phần mềm phải đối mặt ngày hôm nay, đó là những khó khăn trong việc phát triển các ứng dụng Client/Server và cách thức tích hợp các hệ thống đã sẵn có (kế thừa chúng) cũng như yêu cầu tương thích ngược của các hệ thống sẽ phát triển .

Tương tự như các RFC (Request For Comments) của IETF (Intemet Engineering Task Force, OMG cũng đưa ra các yêu cầu kiến nghị (Request For Proposals - RFP) đối với CORBA. Từ "Common" trong COBRA thể hiện sự hợp nhất của hai kiến nghị API quan trọng. Một kiến nghị đến từ HyperDesk and Digital, hỗ trợ API động; kiến nghị khác do Sun và HP (Hewlett Packard) hỗ trợ APIS tĩnh. Kết quả là CORBA được tinh chỉnh và thừa hưởng cả hai tính năng của hai kiến nghị nêu trên [CORBA14] .

Theo OMG, CORBA là giải pháp cho việc *phải có được khả năng tương tác giữa các hệ thống phần cứng phần mềm đang ngày càng phát triển cả về số lượng và chủng loại ngày nay* ("to the need for interoperability among the rapidly proliferating number of hardware and software products available today" [OMG].

III.2.2. Sơ lược về lịch sử CORBA

OMG là một tổ chức chuyên ngành vô vụ lợi, do 8 công ty quốc tế thành lập tháng 5 năm 1989, trong đó đáng kể là Hewlett-Packard và SUN. OMG đáp ứng đúng nhu cầu chung và có một phương pháp làm việc khá khách quan, nên đã được hưởng ứng mạnh mẽ. Từ chỗ chỉ có tám thành viên ban đầu, đến nay OMG đã phát triển lên tới hơn 800 thành viên chính thức.

Hoạt động của OMG nhằm mục đích cung cấp một khung làm việc chung cho các ứng dụng hướng đối tượng nhằm thiết lập một khung cảnh khái niệm chung về hướng tiếp cận sự vật phân tán, để có thể cho phép các hệ áp dụng hướng sự vật, đã và sẽ được phát triển trên những hệ điều hành và thiết bị khác nhau, có thể trao đổi với nhau. Thành tựu lớn nhất của OMG là đã thiết lập được kiến trúc quản lý đối tượng (Object Management Architecture - OMA) mà CORBA là một phần trong đó. Nói một cách ngắn gọn, OMA bao gồm các bộ phận trung gia xử lý yêu cầu trên đối tượng (Object Request Broker - ORB), các dịch vụ về đối tượng (còn gọi là các dịch vụ CORBA), các tiện ích chung, các giao diện miền (domain Interface) và các đối tượng ứng dụng. Vai trò của CORBA trong OMA là quản lý việc thực hiện các chức năng của ORB [OMG].

Ra đời từ tháng 5/1989, CORBA đã có nhiều bước phát triển mạnh mẽ, trong đó đáng chú ý nhất là hai phiên bản 1.0 và 2.0 với các tính năng đột phá.

CORBA 1.0

Đây là phiên bản đầu tiên của CORBA, được giới thiệu vào tháng 12/1990, tức là ngay sau khi tổ chức OMG được thành lập. Đầu năm 1991, phiên bản 1.1 ra đời, trong đó có đưa ra khái niệm về ngôn ngữ định nghĩa giao diện (Interface Definition Language - IDL) và các công cụ API giúp cho các ứng

dụng có thể giao tiếp được với các ORB. Một thời gian ngắn sau, OMG công bố phiên bản 1.2 với một số đặc tính bổ sung so với các phiên bản trước đó. Có thể coi các phiên bản 1.x là bước khởi điểm quan trọng cho khả năng tương tác giữa các thành phần đối tượng, cho phép các đối tượng trên nhiều máy có kiến trúc khác nhau, được viết bằng các ngôn ngữ khác nhau có thể trao đổi được với nhau.

CORBA 2.0

Các phiên bản 1.x còn thiếu hoàn chỉnh vì chưa đưa ra được những quy định đầy đủ về sự tương tác giữa các thành phần đối tượng. Mặc dù chúng cũng đã đưa ra được các chuẩn cho ngôn ngữ IDL và cho việc truy nhập ORB thông qua một ứng dụng nhưng chúng lại mắc phải một hạn chế cơ bản đó là chưa quy định được các giao thức chuẩn để các ORB có thể trao đổi được với nhau [TL_CORBA].Chính vì có hạn chế này mà CORBA ORB của một nhà cung cấp này không thể trao đổi được với CORBA ORB của một nhà cung cấp khác, từ đó thu hẹp khả năng tương tác giữa các đối tượng trong môi trường phân tán.

Để khắc phục hạn chế trên, tháng 12/1994, phiên bản CORBA 2.0 ra đời. Trong phiên bản này, OMG đã đưa ra Giao thức chung giữa các ORB (Internet Inter-ORB Protocol - IIOP) - đây là giao thức chuẩn giúp cho các ORB của các nhà cung cấp khác nhau có thể trao đổi được với nhau. Chuẩn mới này được ứng dụng trên các mạng có sử dụng giao thức TCP/IP. OMG cũng đã quy định rõ rằng tất cả các nhà cung cấp muốn sản phẩm của mình hoạt động được trong kiến trúc CORBA đều phải cài đặt giao thức IIOP.

Tiếp sau phiên bản 2.0, OMG tiếp tục công bố một số phiên bản kế tiếp: CORBA 2.1 (12/1997), CORBA 2.2 và 2.3 (1998). Các phiên bản sau này đã từng bước phát triển và mở rộng các ưu điểm của phiên bản 2.0 trước đó.

CORBA 3.0

Corba 3.0 được chính thức ra mắt vào tháng 10/2002. Mặc dù được thiết kế khác gọn nhẹ, CORBA 3.x không phải là một tiêu chuẩn đơn mà đó thực chất là một họ các tiêu chuẩn được thêm vào chuẩn 2.x [CORBA3.0]

Về cơ bản, CORBA là một công nghệ tích hợp các ứng dụng tính toán phân tán. Hạt nhân của các hệ thống CORBA – ORB – đã "che dấu" các chi tiết thành phần ở mức dưới về hệ thống như platform, mạng, … cho phép các nhà lập trình tập trung chính vào giải quyết các vấn đề của mình thay vì việc phải phát triển một hệ thống hạ tầng tính toán phân tán [CORBA3.0].

Cũng như các tiêu chuẩn công nghệ khác, CORBA cũng phải tự mình phát triển để đáp ứng được các yêu cầu ngày càng cao của nhu cầu. Việc bổ sung thêm ba tính năng chính của CORBA trong phiên bản 3.0 như Portable Object Adapter - POA, CORBA Messaging, và Objects By Value cho phép áp dụng CORBA cho những lĩnh vực mà trước đây còn chưa phù hợp.

III.2.3. Tổng quan về kiến trúc CORBA

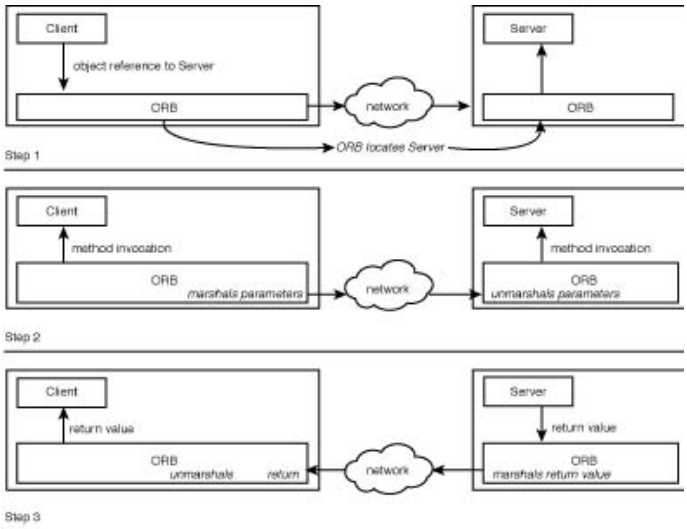
Trong phần này, chúng ta sẽ đề cập đến các thành phần chính của CORBA:

- Bộ phận trung gian xử lý các yêu cầu trên đối tượng (Object Request Broker – ORB): cấu trúc của phần cốt lõi của kiến trúc CORBA.
- Ngôn ngữ định nghĩa Giao diện (Interface Definition Language – IDL) – thành phần cơ bản của kiến trúc CORBA.
- Mô hình truyền thông trong kiến trúc CORBA (CORBA communication model) – lý giải cách thức hoạt động của các đối tượng CORBA trong kiến trúc mạng máy tính.

- Mô hình đối tượng trong kiến trúc CORBA, bao gồm các tham chiếu đối tượng và các Bộ điều hợp đối tượng cơ bản (Basic Object Adapters – BOAs).
- Khái niệm và vai trò của các thực thể Client và Server trong kiến trúc CORBA.
- Khái niệm và vai trò của các client stubs và server skeletons trong việc xây dựng các ứng dụng trong kiến trúc CORBA.

III.2.4. Bộ phận trung gian xử lý yêu cầu trên đối tượng (ORB)

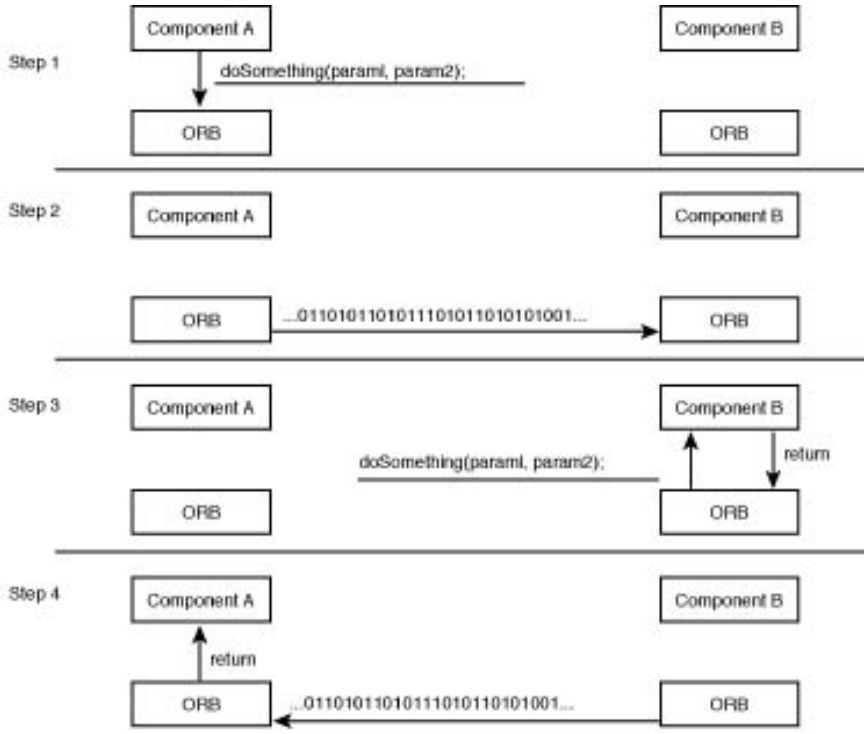
Bộ phận trung gian xử lý các yêu cầu trên đối tượng (Object Request Broker – ORB) chính là bộ phận căn bản cấu thành kiến trúc CORBA, còn được biết đến như là object bus hoặc thư viện các đối tượng ảo (main object library) [OMG_ARCH] Sau khi có ORB, các thành phần khác khi muốn trao đổi thông tin với nhau thì không cần phải kết nối trực tiếp. Thay vào đó, chúng chỉ cần giao tiếp với ORB thông qua các hàm API của CORBA. Các giao tác tiếp theo sẽ do CORBA đảm nhận. Cụ thể, khi một thành phần ứng dụng muốn sử dụng một dịch vụ được cung cấp bởi một thành phần ứng dụng khác, trước tiên nó cần có được một sự tham chiếu tới đối tượng cung cấp dịch vụ đó. Sau khi đã có được sự tham chiếu này, nó có quyền gọi các phương thức của đối tượng được tham chiếu đến và có thể truy cập vào các dịch vụ mà nó mong muốn do đối tượng đó cung cấp. Chức năng đầu tiên của CORBA là giải quyết các yêu cầu về tham chiếu đối tượng, cho phép các thành phần đối tượng có thể thiết lập kết nối với nhau (Hình III.1) [OMG_ARCH]



Hình III.1 ORB giải quyết các yêu cầu về đối tượng

Để dễ theo dõi, từ đây trở về sau chúng ta sẽ quy ước gọi thành phần ứng dụng có yêu cầu sử dụng dịch vụ là Client, còn phía cung cấp dịch vụ được gọi là Server.

Sau khi đã có được sự tham chiếu tới đối tượng cung cấp dịch vụ cần sử dụng, Client có thể bắt đầu gọi các phương thức trên đối tượng này. Thông thường, các phương thức trên đối tượng đều cần có các tham số đầu vào và trả về các kế quả đầu ra nên chức năng tiếp theo của ORB là nhận các tham số đầu vào từ phía Client, đổi chúng sang một khuôn dạng phù hợp để có thể truyền tới được đối tượng được tham chiếu ở xa thông qua mạng máy tính. Quá trình này được gọi là "tập hợp" (*marshal*). Sau đó, ORB lại có trách nhiệm thực hiện một sự chuyển đổi ngược: nó nhận các giá trị tham số đầu ra được trả về và chuyển chúng sang một khuôn dạng mà phía Client có thể hiểu được. Quá trình này được gọi là "phân tách" (*unmarshal*). Xem Hình III.2.



Hình III.2 Quá trình marshal và unmarshal

Toàn bộ quá trình marshal và unmarshal được thực hiện mà không cần có sự can thiệp của người lập trình. Phía Client chỉ việc đưa ra yêu cầu về một phương thức từ xa và sẽ nhận được các kết quả trả về giống như khi thực hiện các phương thức trong lòng nó vậy. Tất cả các công việc đều do ORB đảm trách và "trong suốt" đối với phía Client [OMG].

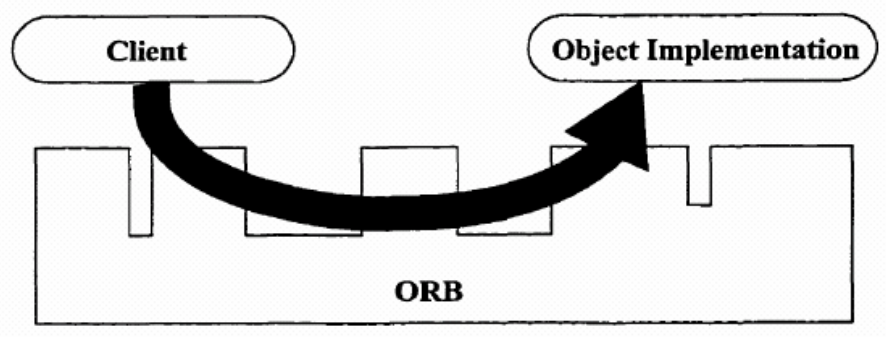
Như vậy, các quá trình marshal và unmarshal giúp cho việc giao tiếp giữa các thành phần ứng dụng trong mạng trở nên độc lập đối với môi trường (platform-independent). Điều đó có nghĩa là một ứng dụng chạy trên hệ Macintosh có thể gọi các phương thức trên một ứng dụng khác chạy trên hệ UNIX. Không chỉ có vậy, những sự khác biệt về phần cứng cũng không gây trở ngại gì bởi lẽ ORB sẽ tự động thực hiện các sự chuyển đổi nếu thấy cần thiết. Có thể nói mọi sự khác nhau về môi trường của các ứng dụng đều được ORB xử lý và giải quyết [CORBA].

Một lần nữa xin được nhắc lại rằng toàn bộ quá trình marshal và unmarshal đều hoàn toàn được thực hiện bởi ORB và hoàn toàn trong suốt đối với cả phía Client và phía cung cấp dịch vụ (Server). Người lập trình cũng tuyệt đối không có liên quan gì tới các quá trình trên.

Như vậy, có thể thống kê tóm tắt các chức năng của ORB như sau:

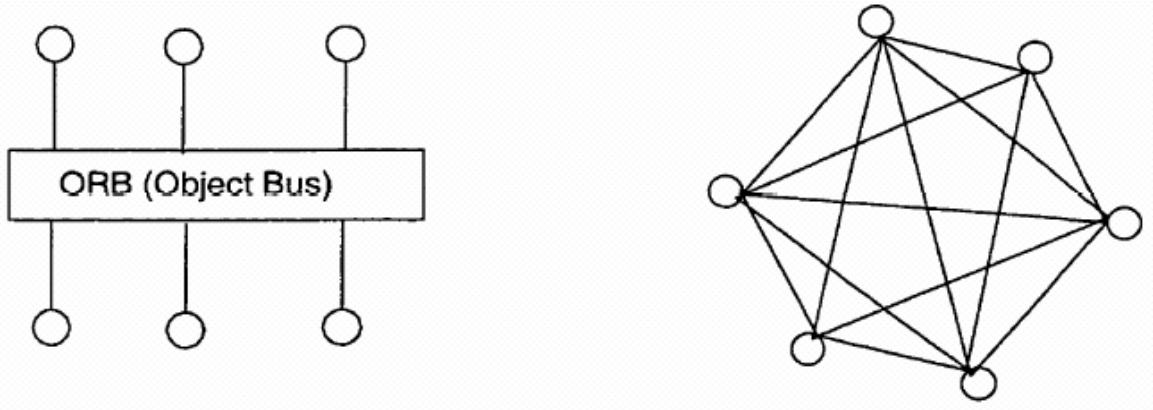
- Nhận một tham chiếu đối tượng từ phía Client, thay mặt Client xác định vị trí của Server tương ứng – là nơi sẽ thi hành dịch vụ mà Client yêu cầu. (Lưu ý rằng việc làm thế nào để có được tham chiếu đối tượng là thuộc trách nhiệm của phía Client).
- Khi đã xác định được vị trí của Server, ORB phải đảm bảo rằng phía Server đã sẵn sàng nhận yêu cầu.
- Bộ phận ORB ở phía Client có trách nhiệm nhận các tham số đầu vào từ Client, sau đó thực hiện quá trình marshal.
- Bộ phận ORB ở phía Server thực hiện quá trình unmarshal các tham số và chuyển chúng cho Server để xử lý.
- Trong trường hợp có các tham số trả về, ORB lại tiến hành các quá trình marshal và unmarshal giống như trên.

Vai trò trung gian của ORB có thể được minh họa bằng hình vẽ dưới đây:



Hình III.3 Vai trò trung gian của ORB

Để thấy được lợi ích có được khi sử dụng ORB, chúng ta hãy xem xét trường hợp có N thành phần Client/Server trong môi trường ứng dụng. Nếu không sử dụng ORB, ta sẽ cần phải định nghĩa N^2 giao diện để chúng có thể làm việc được với nhau (hình a). Trong khi đó, sử dụng ORB, số giao diện cần phải định nghĩa chỉ là N. Chưa kể là khi không có ORB, các giao diện phải được đồng bộ về ngôn ngữ cũng như về hệ nền (platform)

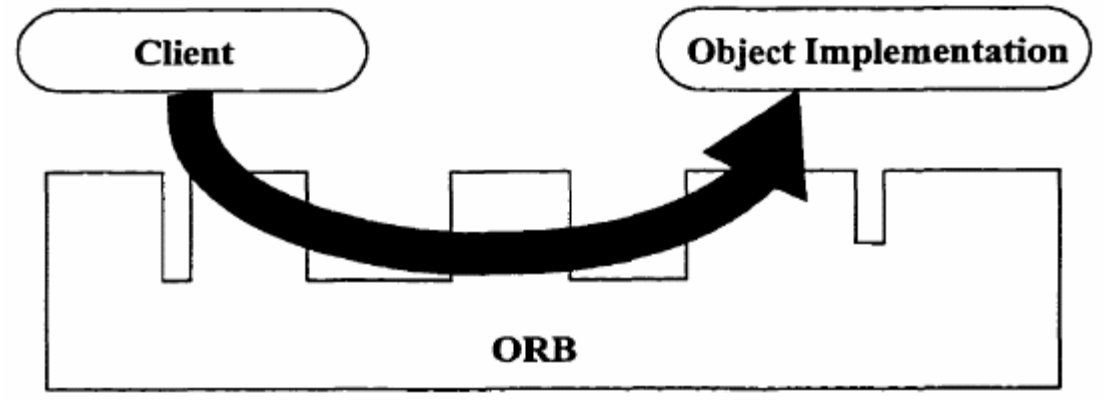


Hình III.4 Tương tác giữa các thành phần qua ORB và không qua ORB

Bản chất của ORB là một thành phần phần mềm chạy giữa các máy tính Client Server và cung cấp cơ chế liên lạc giữa chúng [CORBA14]. ORB có 02 chức năng chính:

- Cung cấp tham chiếu đến các đối tượng mà client yêu cầu
- marshals và unmarshals các tham số đi/đến các đối tượng

Trong CORBA, chúng ta sử dụng tham chiếu đối tượng để xác định đối tượng trong ORB. Có thể xem ORB hoạt động như một "bộ sưu tập" các đối tượng và tài nguyên mạng, liên quan đến các phần mềm ứng dụng, cho phép các ứng dụng này định vị và sử dụng chúng trong môi trường ORB.

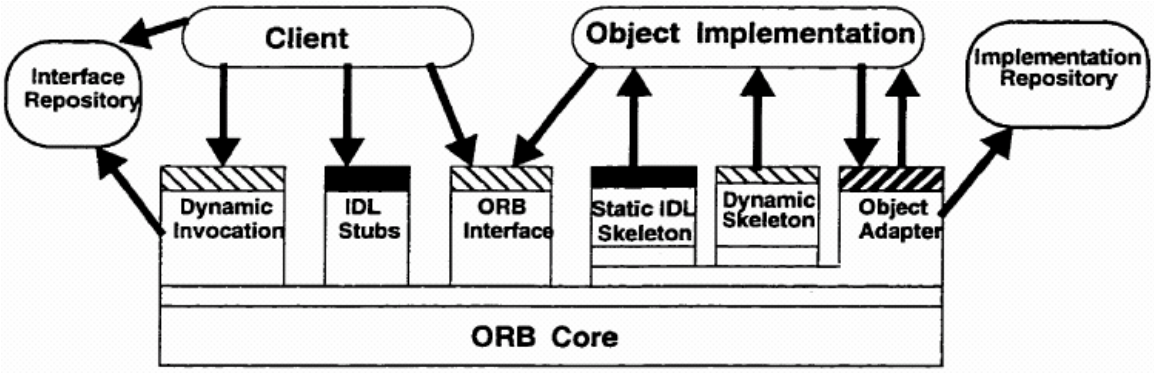


Hình III-3 Mô hình gửi yêu cầu qua Object Request Broker

Hình III-3 minh họa một yêu cầu được gửi bởimột client đến một đối tượng thực thi thông qua một ORB. Thực thể Client đã gửi một yêu cầu thực hiện một tác vụ trên đối tượng mà không cần phải biết đến vị trí của đối tượng cũng như cách thức thực hiện tác vụ đó. ORB sẽ có nhiệm vụ tìm đối tượng sẽ thực hiện yêu cầu, tập hợp các tham số cần gửi đến đối tượng cũng như phân tách các kết quả gửi về từ đối tượng đó.

Như đã trình bày ở trên, *marshalling* là quá trình dịch các tham số từ phía Client thành khuôn dạng của dữ liệu sẽ được truyền đi trong mạng. Unmarshalling là quá trình ngược lại của *marshall*: chuyển đổi số liệu từ mạng sang khuôn dạng mà phía client có thể hiểu được [Rosenberger 98].

Trong CORBA, Client có thể gửi yêu cầu đến server qua nhiều con đường: thông qua IDL (Interface Definition Language) tĩnh hoặc qua DII (Dynamic Invocation Interface) – Giao diện yêu cầu động. Ngoài ra, client còn có thể gửi yêu cầu trực tiếp đến ORB như mô tả chi tiết ở Hình III.5



Hình III.5 Cấu trúc của ORB

Các IDL stub cung cấp các giao diện tĩnh cho các đối tượng phục vụ và phụ thuộc vào đối tượng đó (tuỳ theo loại đối tượng cụ thể). Nói một cách đơn giản, IDL stub là một đoạn mã nhỏ được biên dịch cùng với phần mềm client tạo cho client khả năng truy cập server.

Dynamic Invocation Interface (DII) cung cấp phương thức "phát hiện động" các giao diện server và có thể yêu cầu truy cập các đối tượng mà thậm chí client còn chưa được biết đến (hoặc chưa tồn tại) ngay từ khi biên dịch phần mềm client. Giá phải trả cho đặc tính này là mức độ phức tạp bị tăng lên đáng kể.

Về phía server, nơi chứa các đối tượng thực thi, cũng có các thành phần (được gọi là bộ khung - skeleton) tương ứng với hai giao diện trên. Đó là static IDL skeletons và Dynamic Skeleton Invocation (DSI) tương ứng với IDL stub và DII của phía client.

Các server skeleton này cũng là các đoạn mã lệnh nhỏ được sinh ra khi biên dịch các đặc tả giao diện IDL. Chúng cung cấp các giao diện tĩnh cho từng dịch vụ, được server cung cấp (export).

Để xử lý một yêu cầu, đoạn mã thực thi đối tượng có thể gọi các Object Adapter và giao diện ORB cho các dịch vụ thông qua ORB. Object Adapter

quản lý các lạo hình dịch vụ như là sinh ra và thông dịch các tham chiếu đối tượng cũng như các phương pháp yêu cầu [OMG_ARCH] .

Để quản lý được các đối tượng thực thi, server có thể truy cập vào hai cơ sở dữ liệu: interface repository (kho chứa giao diện) và implementation repository (kho chứa các phương thức). Interface repository cung cấp các đối tượng bền vững (persistent) được đặc tả đầy đủ và chính là các thông tin IDL trong toàn bộ tiến trình (run-time).

Implementation repository chứa các thông tin cho phép ORB định vụ và kích hoạt các phương thức của đối tượng.

III.2.5. Ngôn ngữ định nghĩa giao diện (IDL)

Ngôn ngữ định nghĩa giao diện IDL (Interface Definition Language) là một thành phần khác của CORBA và được sử dụng để định nghĩa kiểu của các đối tượng băng cách đặc tác các giao diện của nó. Là ngôn ngữ được sử dụng để mô tả các giao diện giữa các đối tượng CORBA nhưng IDL không phải là 1 ngôn ngữ lập trình theo đúng ý nghĩa của nó mà chỉ tự giới hạn ở mức định nghĩa các giao diện.

IDL không phải là ngôn ngữ thủ tục, nó chỉ có thể định nghĩa các giao tiếp không có phần cài đặt. Nó tương tự như phần header của các class trong C++, phần header không chứa bất kì cài đặt của lớp nào nhưng mô tả được lớp và giao diện của lớp.

IDL hoàn toàn không phụ thuộc vào ngôn ngữ sử dụng, nói cách khác IDL có thể được thực thi trong một số ngôn ngữ có tồn tại ánh xạ ngôn ngữ đến nó, ví dụ như Java, C, C++ và một số ngôn ngữ khác.

IDL hoàn toàn chỉ là ngôn ngữ mô tả nên các chương trình phía Client không nhất thiết phải viết bằng IDL mà có thể được viết bằng ngôn ngữ bất

kì, nhưng ngôn ngữ đó phải được ánh xạ vào IDL. IDL sẽ có trách nhiệm chuyển đổi dữ liệu, kiểu dữ liệu một cách tương thích giữa các ngôn ngữ khác nhau [TL_CORBA] .

Một giao diện bao gồm một nhóm các tác vụ được đặt tên (các hàm hoặc là phương thức) và nhờ đó, các client có thể yêu cầu chúng để được phục vụ.

IDL là một loại ngôn ngữ đặc tả, nó hỗ trợ chính tả C++ cho khai báo hằng, kiểu biến và tác vụ. Tuy nhiên như đã nói ở trên, IDL không có cấu trúc thủ tục và biến, từ đó, nó không có các câu lệnh điều khiển rẽ nhánh như if-else, while…

IDL được viết theo phong cách của C++, hỗ trợ naming space, tiền xử lý, thừa kế đơn và đa thừa kế (multiple inheritance)… và tất nhiên là có thêm một số từ khóa hỗ trợ mô hình tính toán phân tán.

- *module* – tương tự như các Java package
- *interface* – giao diện của CORBA, tương tự như Java interface
- *exception* - tương tự như Java exception
- *attribute* – Các biến thành viên được tạo tự động khi tạo(get đối với các thuộc tính chỉ đọc (readonly attributes), get và set cho các thuộc tính bình thường)
- *operations* - tương tự như method trong các ngôn ngữ lập trình khác
- *Các kiểu dữ liệu đơn giản* như là short, unsigned short, char, ...
- *Các kiểu dữ liệu có cấu trúc* như mảng, array, sequence, struct, enum…

Sequence là kiểu dữ liệu tương tự như mảng nhưn các thành viên của nó có kích thước không giống nhau và có thể thay đổi được trong quá trình thực

thi. Như vậy, seqquense có thể xem là tương tự như biến kiểu vector của Java, có một sự khác biệt là các thành viên phải là cùng có một kiểu, trong khi Java cho phép thành viên là các đối tượng có kiểu khác nhau.

Sau khi đã khai báo xong file IDL, người sử dụng có thể dựa vào bộ tiền xử lý (precompiler) – biên dịch các file IDL sang ngôn ngữ được sử dụng để hoàn thành quá trình thực hiện.

Đoạn mã sau đây minh họa một ví dụ về file IDL

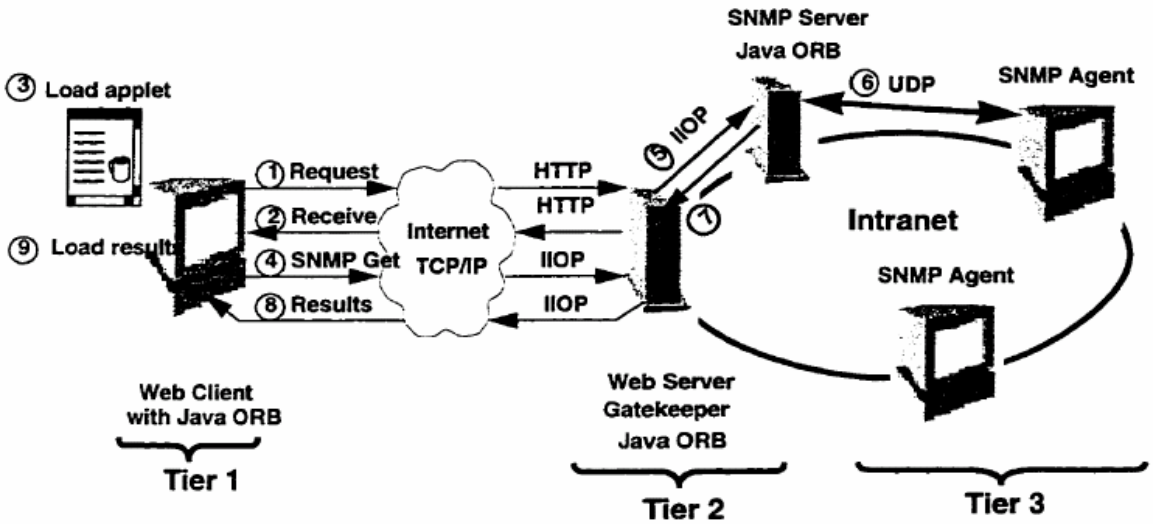
```
module Bank {
     interface Account{
          float balance();
     };
     interface AccountManager {
          Account open( in string name);
     };
};
```

Ví dụ trên đưa ra một định nghĩa về giao diện Account và AccountManager. Từ khóa trong phương thức open có ý nghĩa là biến name sẽ phải được chuyển từ phía client đến server. Theo định nghĩa của IDL, các tham số trong các phương thức phải được mô tả đầy đủ là chúng được chuyển từ phí client đến server (in) hay là ngược lại (out) hay là theo cả hai hướng (inout). Đó là một sự khác biệt so với C và Java.

III.2.6. Mô hình bốn bên giữa Web client và server với CORBA

Chúng ta đã tham khảo vai trò của CGI trong mô hình ba bên Client/Server sử dụng CGI và đã thấy được các nhược điểm của CGI: ứng với mỗi một yêu cầu từ phía client, web server phải tạo ra một tiến trình hoàn toàn mới bất kể yêu cầu đó là gì.

Ở phần trước, chúng ta cũng đã biết ORB của CORBA cũng cấp các tham chiếu đến các đối tượng và chuyển đi, trả lại các tham số từ client đến các đối tượng. Trong phần này, chúng ta sẽ tìm hiểu cách thức thực hiện của CORBA trong môi trường client/server ba bên.



Hình III-4 Mô hình client/server 4 bên trong ứng dụng CORBA SNMP

Hình III-4 minh họa một ứng dụng quan trắc mạng sử dụng kiến trúc Clent/server ba bên sử dụng CORBA của VisiBroker.

(1) Trên hình minh họa, ta thấy phía web client gửi yêu cầu đến web server thông qua HTTP;

(2) Web client dowwnload trang html (trong đó có một Java applet và một file Jar). Việc này cũng được thực hiện qua HTTP;

(3) Web browser tải applet và file Jar vào bộ nhớ của máy client và bắt đầu chạy applet;

(4) Sau khi người sử dụng click vào 1 nút bấm trong cửa sổ của applet để yêu cầu nhận thông tin về mạng, yêu cầu được gửi đến gatekeeper (chạy trên web server thông qua ORB/IIOP)

(5) VisiBroker Gatekeeper cho phép applet được liên lạc với SNMP server thông qua môi trường mạng như là một cổng giao tiếp từ phía client đến đối tượng server nằm ở phía bên kia của mạng

(6) SNMP server sẽ giao tiếp với các Agent trong mạng nội bộ để thực hiện các yêu cầu lấy thông tin

(7) SNMP server gửi trả kết quả về cho Gatekeeper trên web server

(8) Gatekeeper tự động chuyển tiếp kết quả về phía Web client thông qua ORB/IIOP.

(9) Kết quả được nạp và trình bày cho người sử dụng ở trình duyệt web.

III.3. Tóm tắt về CGI và CORBA

Chuẩn CGI cho phép Web server liên lạc với các chương trình bên ngoài, để tiến hành thực hiện các nhiệm vụ riêng biệt. Sự khác nhau chính giữa các chương trình ứng dụng là hệ điều hành nó đảm bảo yêu cầu về tốc độ hoạt động của chương trình.

Trong môi trường Unix, dữ liệu từ các from gửi tới chương trình bên ngoài thông qua chuẩn vào. Một vài tham số có thể được dùng để yêu cầu các tham số truyền qua các biến môi trường. Server sẽ gửi các dữ liệu ra của chương trình bên ngoài hướng tới client. Do đó đòi hỏi các chương trình bên ngoài phải sinh ra thông tin header cho client. Một trong các thành phần hết sức qua trọng của header là trường Content-type. Chính nhờ nó mà client sẽ biết xử lý như thế nào với dữ liệu.

Trong môi trường Window, file-base truyền tới chương trình bên ngoài , tương tự, dữ liệu của chương trình được đưa vào trong file cho server đọc. Tất cả dữ liệu được lưu vào trong file riêng profile.

Từ những xem xét trên, chúng ta thấy rằng các chương trình CGI sẽ được gọi thực hiện mỗi khi có yêu cầu từ phía khách hàng. Mỗi lần như vậy, máy chủ (server) sẽ tạo một tiến trình (process) mới cho gateway và truyền thông tin từ khách hàng cho tiến trình này thông qua các biến môi trường và dòng nhập dữ liệu chuẩn. Sau khi chương trình CGI được thực hiện xong, kết quả sẽ được gửi ngược lại cho phần mềm web server thông qua dòng dữ liệu ra chuẩn. Như vậy, đối với mỗi một yêu cầu từ phía Client, máy chủ web server sẽ phải tạo ra một tiến trình (process) hoàn toàn mới, tiêu tốn khá nhiều tài nguyên hệ thống. Càng có nhiều yêu cầu của khách hàng, càng có nhiều tiến trình CGI do máy chủ tạo ra; điều đó dẫn đến sự chiếm dụng tài nguyên, gây chậm trễ cho hệ thống. Mặt khác, khả năng tương tác với người sử dụng của CGI cũng có những hạn chế [Mazumdar].

Tuy nhiên, CGI cũng có ưu điểm là có thể cài đặt trên các hệ điều hành và các Web server khác nhau, giao diện được chuẩn hóa và đáp ứng được các yêu cầu cơ bản.

Từ những phân tích trên, chúng ta có thể tạm rút ra một số nhận xét về sự khác nhau giữa các ứng dụng CORBA và các ứng dụng dựa trên CGI như sau:

(1) Trong ứng dụng CGI, web server phải tạo một tiến trình hoàn toàn mới ứng vỡi mỗi một yêu cầu, tiêu tốn nhiều tài nguyên hệ thống cũng như thời gian thực hiện. Ngoài ra, toàn bộ thông điệp gửi giữa client và server được thực hiện trên HTTP, dẫn đến phải có có nhiều bước xử lý (overhead)

(2) Trong ứng dụng CORBA, web server và HTTP chỉ được sử dụng để truyền trang HTML và Java applet vào lúc bắt đầu của tiến trình. Sau đó thì client và đối tượng server đã thiết lập và trao đổi thông tin với

nhau thông qua IIOP, nhờ đó, giảm được khác nhiều overhead so với HTTP [Mazumdar] .

(3) Đối với các ứng dụng CORBA, server đối tượng chỉ chạy ở bên trong trong (đằng sau) web server và chỉ cung cấp các dịch vụ thông qua các giao diện đã được quy định cho các client. Phiên làm việc giữa applet và server đối tượng sẽ được sẽ giữ trong toàn bộ tiến trình và toàn bộ các thông tin trạng thái sẽ được giữ cho đến khi một trong hai phía ngắt kết nối.

(4) CORBA bảo đảm sự trong suốt về vị trí (nội bộ hay ở xa) đối với các phương thức truy vấn dịch vụ, nghĩa là đối với client, công việc được thực hiện như nhau dù server đối tượng nằm ở mạng khác, trên cùng một mạng hay trong cùng một máy tính [CORRBA14].

(5) Với mô hình hạ tầng kiến trúc phân tán các đối tượng, CORBA cho phép các Java applet liên lạc với các đối tượng được viết bằng các ngôn ngữ lập trình khác thông qua mạng.

Chương IV. Xây dựng hệ thống quản trị DSLAM qua web

IV.1.Khảo sát hệ thống mạng cung cấp dịch vụ ADSL

IV.1.1. Giới thiệu hệ thống mạng cung cấp dịch vụ ADSL của Bưu điện Hà nội

Từ tháng 7-2003, một dịch vụ mới được triển khai rộng khắp trên mạng VNN là dịch vụ truy cập qua đường dây thuê bao số xDSL, bao gồm các công nghệ ADSL, SHDSL và VDSL (gọi tắt là xDSL). Các dịch vụ xDSL ra đời mang lại khả năng truy cập Internet băng rộng giá rẻ cho khách hàng. Tới thời điểm hiện tại chỉ tính riêng tại địa bàn Hà nội đã có khoảng trên 35.000 khách hàng (cổng truy cập) sử dụng dịch vụ xDSL.

Để có thể cung cấp dịch vụ xDSL trên địa bàn thành phố Hà nội, hiện nay Bưu điện Hà nội đang quản lý một hạ tầng mạng lưới bao gồm một hệ thống phục vụ truy nhập hiện đại với các thiết bị DSLAM (Digital Subscriber Line Access Multiplexer) phân bổ ở khắp nơi trên địa bàn thành phố (hơn 140 điểm lắp đặt, gần 200 DSLAM …) của nhiều nhà cung cấp thiết bị nổi tiếng. Đến nay, trên địa bàn Hà nội hiện có 8 chủng loại thiết bị của 4 nhà sản xuất khác nhau Siemens, Huawei, Tailyn, ZTE … với các công nghệ khác nhau như ATM DSLAM, IP DSLAM…

Hệ thống các DSLAM thuộc 4 hãng sản xuất này được quản trị, giám sát, khai thác mạng từ xa bởi 04 hệ thống quản lý NMS (Network Management System) tập trung do từng hãng sản xuất thiết bị cung cấp. Các hệ thống NMS này đều là môi trường đóng, được thiết kế hướng tới đối tượng là các kỹ thuật viên vận hành mạng nên không cung cấp giao diện ra bên ngoài và không có mối liên hệ với nhau.

Tuy khác nhau về chủng loại thiết bị, nhà sản xuất, phần mềm quản lý, nhưng tất cả các hệ thống NMS đều được thiết kế với các thành phần chính như sau:

1. Hệ thống DSLAM:
    - Thiết bị của hãng Siemens: *XpressLink Standard, XpressLink Mini, XpressLink M200 và M1200 (IP DSLAM)*
    - Thiết bị của hãng Huawei: *MA5100* và *MA5600V3* (IP DSLAM)
    - Thiết bị của hãng Tailyn: *UMAP 2100*
    - Thiết bị của hãng ZTE: *ZTE 8203*
2. Hệ thống BRAS (BoardBand Remote Access System): ERX 1410 của hãng Juniper
3. Hệ thống máy chủ phục vụ vận hành khai thác và quản lý: ACI server, FTP server, DHCP server, NMC-RX Server, SDx server, …
4. Hệ thống máy chủ cung cấp dịch vụ: Radius, Billing server, VoD server, Catching server, TV server, …
5. Hệ thống bảo vệ Firewall, log server, …
6. Mạng truyền dẫn: truyền dẫn STM-1 và E1
7. Mạng truy cập cơ sở: sử dụng đôi dây cáp đồng điện thoại sẵn có từ nhà thuê bao đến các tổng đài điện thoại.

IV.1.2. Cơ bản về thiết bị DSLAM

Hệ thống ghép kênh truy nhập đường thuê bao số DSLAM (Digital Subscriber Line Access Multiplexer) là "điểm" giao tiếp giữa người sử dụng đầu cuối và nhà cung cấp dịch vụ băng rộng.

Trong mạng quản lý DSLAM, chúng ta có thể phân biệt theo vai trò của chúng trong mạng: DSLAM-HUB và DSLAM (stanalone, sub-DSLAM):

- DSLAM-HUB thì có hoạt động với vai trò là thiết bị ghép kênh cấp 1, kết nối lên BRAS bằng luồng STM-1 và có các DSLAM cấp 2 kết nối vào.
- DSLAM thường là các thiết bị ghép kênh cấp 2 được kết nối lên DSLAM cấp 1 hoặc BRAS mà không có các DSLAM thứ cấp. đầu ra của nó được kết nối về các DSLAM chính. Sub-DSLAM chịu sự quản lý của DSLAM chính (Main DSLAM).

Kết nối giữa giữa MainDSLAM với SubDSLAM, hoặc MainDSLAM – SubDSLAM với mạng IP bằng các loại giao tiếp sau:

- n X Ethernet 100/1000 (cho lưu lượng IP).
- 155Mbps SDH.
- 34Mbps PDH.
- n X 2Mbps.

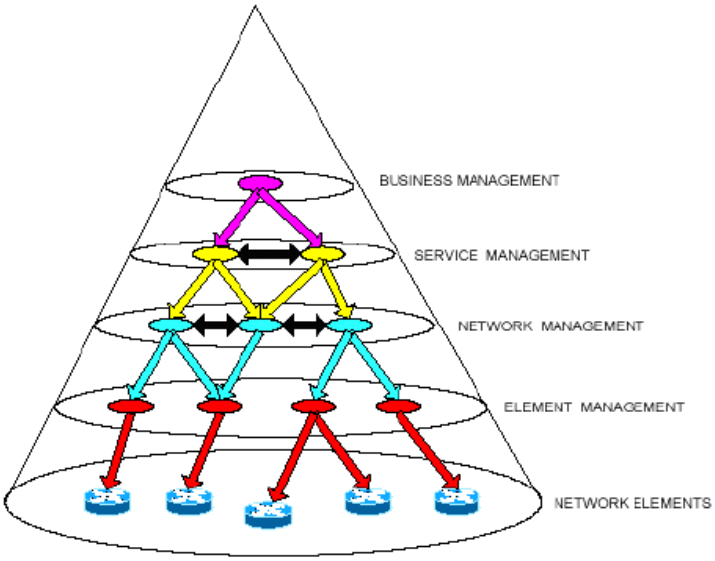
IV.1.3. Hệ thống quản lý mạng xDSL

Hệ thống quản lý mạng NMS (Network Management System) sẽ làm việc theo khuyến nghị của ITU: G.992.1, G.992.2, G.997.1 và IETF RFC 2662. NMS là hệ thống mở, kết hợp với tiêu chuẩn cấu trúc quản lý mạng viễn thông TMN được xác định bởi khuyến nghị M.3010 của ITU.

Hệ thống ADSL sẽ được quản lý bởi một hệ thống quản lý tập trung được hình thành một Nhà quản lý các phần tử mạng ; tổng hợp với hệ thống quản lý mạng lưới trên:

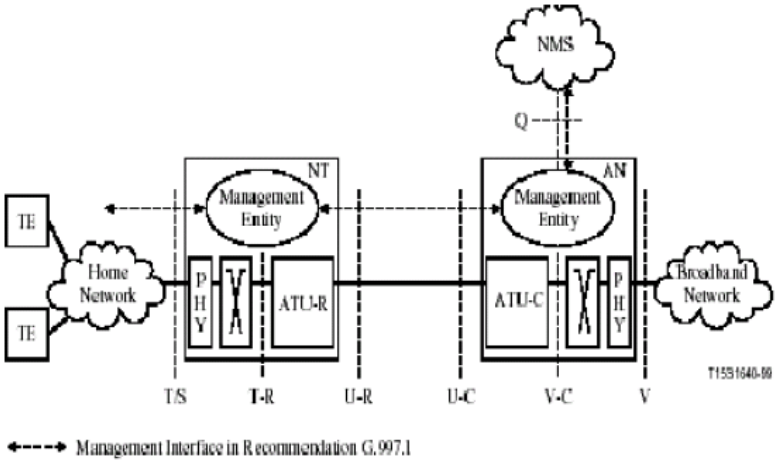
- Lớp quản lý phần tử mạng.

- Lớp quản lý mạng.
- Lớp quản lý dịch vụ.
- Lớp quản lý kinh doanh.

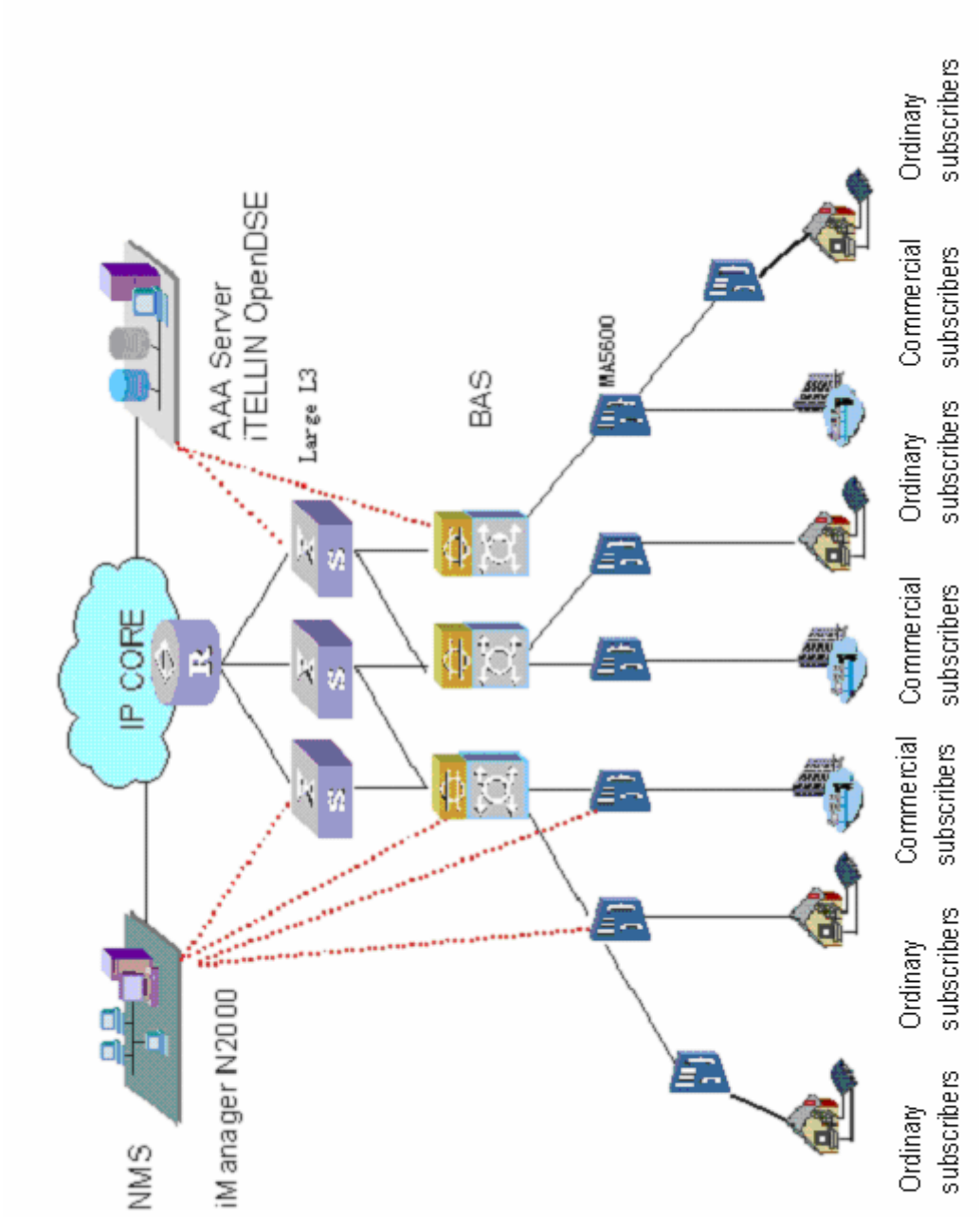


Hình IV-1 CẤu trúc quản lý mạng

Các phần tử của mạng truy cập sẽ được quản lý bởi hệ thống quản lý mạng NMS dùng giao thức SNMP. Phụ thuộc vào kính cỡ mạng trong tương lai và người khai thác yêu cầu giải pháp quản lý có thể tuỳ chọn bao gồm sự kết hợp quản lý phần tử EM (Element Management), trợ giúp thủ công LCT (Local Craft Tool) và các thiết bị giao tiếp truyền thông IMD (Interface Mediation Device).



Hình IV-2 Mô hình tham chiếu quản lý mạng



Hình IV-3Mô hình hệ thống quản lý DSLAM của HUAWEI tại Bưu điện Hà nội

Yêu cầu hệ thống NMS phải hỗ trợ tính năng quản lý phần tử, quản lý mạng và quản lý dịch vụ. Quản lý phần tử mạng nên hỗ trợ đa truy cập từ người khai thác được kết nối trên nền IP, môi trường LAN/WAN.

Giải pháp quản lý mạng phải hỗ trợ cho cả Mức độ thấp (ứng các mạng nhỏ) và Mức độ cao, rộng với sự tổng hợp của cả quản lý mạng và dịch vụ. Xem hình 8-2-1 Mô hình tham chiếu quản lý mạng.

IV.1.4. Công việc quản lý mạng

Khuôn khổ công việc quản lý mạng sẽ bao trùm ít nhất 4 bộ phận:

- **Hệ thống quản lý mạng**: Tất cả các phần tử của mạng truy cập ADSL phải được quản lý bởi NMS. Cần cấu hình dư cho NMS.
- **Các node quản lý**: Các node quản lý, mỗi node chứa một phần tử đại diện, có thể là một bộ định hướng router, cầu –Bridge, chuyển mạch - Switch, Modem ADSL...
- Giao thức quản lý mạng: Giao thức quản lý mạng được dùng bởi người quản lý và các nhân viên để chuyển mạch các tin tức quản lý sẽ là SNMP, giao thức quản lý mạng internet (Lớp ứng dụng) cho phép nhà quản lý quản lý hoạt động của mạng lưới, tìm và giải quyết các vấn đề và lập kế hoạch cho việc phát triển mạng.
- Quản lý cơ sở thông tin MIB (Management Information Base): Quản lý cơ sở thông tin MIB như việc sưu tập các đơn vị tin tức trong quản lý để phụ vụ việc lưu trữ, xử lý công việc khi cần.

IV.1.5. Chức năng quản lý phần tử mạng

NMS sẽ cung cấp tất cả các chức năng để quản lý mạng truy cập ADSL. Quản lý phần tử sẽ cung cấp các chức năng sau:

- **Quản lý cấu hình**: Các chức năng quản lý cấu hình sẽ cho phép hình thành các phần tử mạng với:
- Tự động xắp đặt các trường trạng thái.

- Khai báo card mới lắp đặt trong các DSLAM.
- Khái lược cấu hình mạng cho tất cả mức độ liên lạc: đường dây ADSL, truy cập ATM, mạng ATM, kiểm tra chấp nhận kết nối CAC (Connection Admission Control)
- Cấu hình các kết nối ATM (VPI/VCI).
- Mở/Khoá dịch vụ ADSL.
- Download phần mềm trên các bảng mạch DSLAM và CPE.
- Quản lý phần tử phải hỗ trợ: Xem cấu hình mạng; Xác định tài nguyên của node; Thống kê theo dõi thiết bị đã lắp đặt; Phần mềm quản lý.
- Phần tử mạng được yêu cầu hỗ trợ các tính năng Cấu hình tự động và cấu hình bởi người khai thác mạng. Dữ liệu cấu hình cần phải xác định: vị trí node (Tên + vị trí node), subrack, bảng mạch (card, Slot), xác định các đường nối nội bộ (local port).
- Cấu hình hệ thống được thực hiện trước khi bắt đầu lắp đặt thiết bị. Cái này cho phép chức năng:
- cắm - chạy card, các đầu cuối mạng NT hay bất kỳ thiết bị khác nào được thêm vào hệ thống
- Tự động nhận biết (detect) và tự động cấu hình.
- cho phép cài đặt các mạch ảo thường trực PVC không cần sự hiện diện của card phần cứng và một NT có thể được đưa vào hoạt động sớm khi đã được lắp đặt.

- Hoạt động/ dừng hoạt động cũng sẽ được kiểm tra bởi người quản lý mạng, có thể thay đổi trạng thái quản lý trên thiết bị và PVC trong hệ thống. Các yêu cầu sẽ được gởi tới các phần tử mạng quản lý để cho hoạt đọng/ dừng hoạt động thiết bị đã xác định. Người khai thác phải được thông tin về kết quả:
    - thực hiện thành công hoặc lỗi.
    - Xử lý trong hệ thống được thực hiện theo các nguyên lý ITU-T X.731.
- Các chủ thể có thể quản lý trong hệ thống sẽ có các trạng thái sau:
    - Trạng thái quản lý (loked = Blocked, unlocked = unblocked, enabled = up, disabled = down)
    - Trạng thái hoạt đọng (Active, Idle)
- Các chủ thể có thể quản lý (danh sách và đẳng cấp sắp đặt) trong hệ thống là:
    - Các bảng mạch điện (Card).
    - Các cổng (điểm vật lý).
    - Các giao tiếp ATM.
    - Các mạch ảo thường trực PVC.
    - Các khe (Slot) sẽ được quản lý hoặc không được quản lý. Một khe được quản lý sẽ được giám sát bởi các thiết bị ứng dụng. Khe không quản lý sẽ được tạo ra khi tái cấu hình như việc xoá một subrack.
- **Quản lý lỗi**: Quản lý lỗi cho phép hình thành yếu tố mạng với:

- Thu nhận được các cảnh báo từ thiết bị.
- Phân lớp cảnh báo trên yếu tố nghiêm ngặt hệ trọng.
- Thực hiện ban hành một cảnh báo dưới dạng thông tin.
- Tiến triển của cảnh báo (đang bị, đã biết, đã khắc phục).
- Hình thành sự tương quan cảnh báo.
- Lọc lựa cảnh báo.

- **Quản lý sự hoạt động:** Chức năng quản lý hoạt động sẽ hình thành lên các phần tử mạng như:
  - Sưu tập dữ liệu hoạt động ở các điểm hoạt động thay đổi (SDH, PDH, các kết nối ATM, các đường nối ADSL).
  - Dữ liệu tường trình được tạo ra ở cùng điểm hoạt động.
  - Giám sát hoạt động được hoạt động và ngưng hoạt động trên một bảng mạch đơn từ người quản lý phần tử (mạng).
  - Dữ liệu hoạt động (các bộ đếm trong các bộ ghi) sẽ được chiếm giữ từ hệ thống bởi người quản lý sau khi giám sát hoạt động xảy ra. Trong mỗi bộ ghi đó có một giá trị ngưỡng có thể được cài đặt do người quản lý. Nếu giá trị bộ ghi xấp xỉ giá trị ngưỡng thì một cảnh báo sẽ xảy ra.
  - Bộ đếm sẽ được sử dụng bởi người quản lý: thu thập các giá trị đã đếm cho người quản lý, bao hàm thêm chức năng "sự kiện chú ý".
- **Quản lý an ninh mạng**: Truy cập tới EM (Element Management) được điều hành bởi tiêu chuẩn các cơ chế an ninh, nó điều hành và

cho phép người quản lý logon hệ thống. Chức năng quản lý an ninh cho phép:

- Tạo ra khái lược cơ sở thẩm quyền truy cập trên Username/ Password. Tin tức truy cập được lưu trữ trong log file để nhận dạng hoạt động của người quản lý.
- Cho phép các định vùng domain quan hệ của người quản lý để kiểm tra việc chia sẻ tài nguyên.
- Người quản lý sẽ xác định tất cả các user, người mà cần một phần hoặc đầy đủ thẩm quyền truy cập, bao gồm password, nhóm thành viên... Nó cũng cho phép giới hạn user truy cập theo thời gian, cài đặt thống kế truy cập...
- Truy cập tới hệ thống NMS sẽ có 3 mức độ khác nhau: Người quản lý hệ thống, người giám sát và người quản lý thông thường. Người quản lý hệ thống: sẽ được phép truy cập tới các cửa sổ EM, được phép thay đổi bất kỳ các tham số. Có thể tạo, xoá, sủa đổi thẩm quyền các user. Người quản lý thông thường: Có thể truy cập tới các cửa sổ EM, được phép thay đổi một vài tham số. Người giám sát: Có thể truy cập tất cả các cửa sổ EM, không được quyền thay đổi bất kỳ tham số nào.
- Cho phép gán (cài đặt, xoá, tạo) thẩm quyền cho user trong mỗi ứng dụng (ADSL, PVC, thiết bị, cảnh báo & sự kiện, dịch vụ, UNI, NB(MD), SNMP).

IV.1.6. Mạng quản lý truy cập

Hệ thống quản lý mạng thông qua người quản lý phần tử (mạng) để hỗ trợ quản lý mạng truy cập, xử lý nhiều phần tử mạng.

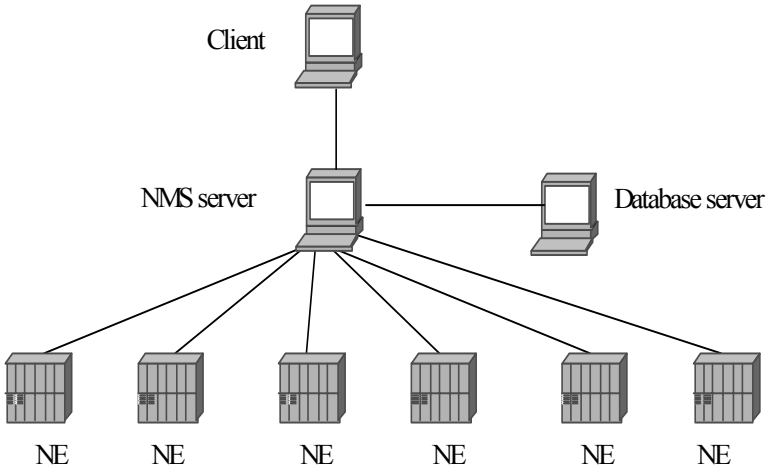
Trong thiết bị ứng dụng các phần tử mạng khác nhau được giám sát từ bản đồ trong Open View Network Element Node Manager. Nó cho phép xem các node hệ thống, các subrack, slot và các bảng mạch.

Các phần tử mạng, node hệ thống và subbrack được cảnh báo trên bản đồ Open View. Nó cho phép tạo ra cảnh báo chung và liệt kê danh sách các sự kiện cảnh báo tới tất cả các điểm kiểm tra (Control Point).

Dịch vụ ứng dụng được dùng để cài đặt một dịch vụ tới người dùng như PVC, ADSL và các chương trình cài đặt mẫu sẽ được tạo cho tất cả các điểm kiểm tra điều khiển CP.

IV.1.7. Cấu hình Client Server NMS

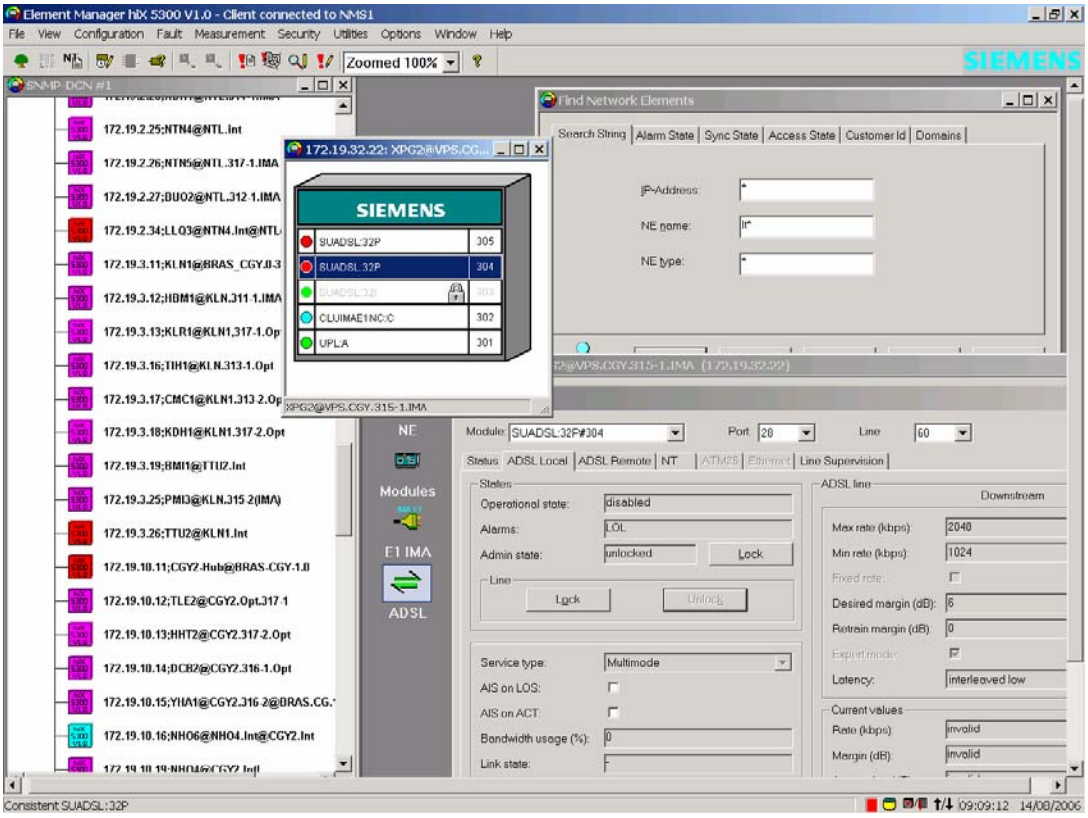
Có 2 sự thiết lập hệ thống quản lý: Client/Server và đứng một mình Standalone. Thiết lập Client/Server đòi hỏi trong trường hợp thiết lập dùng ở mạng lớn (có nhiều điểm điều khiển CP và nhiều đường dây ADSL). Nó bao gồm một server với nhiều màn hình nhỏ với việc xử lý chung về thuê bao và cơ sở dữ liệu. (Các dữ liệu xử lý tại chỗ được lưu trữ tại chỗ, tất cả cấu trúc dữ liệu cho các phần tử mạng được lưu trữ trong CP). Trong trường hợp muốn tìm kiếm thông tin từ CP, các Client sẽ làm việc trực tiếp với CP.



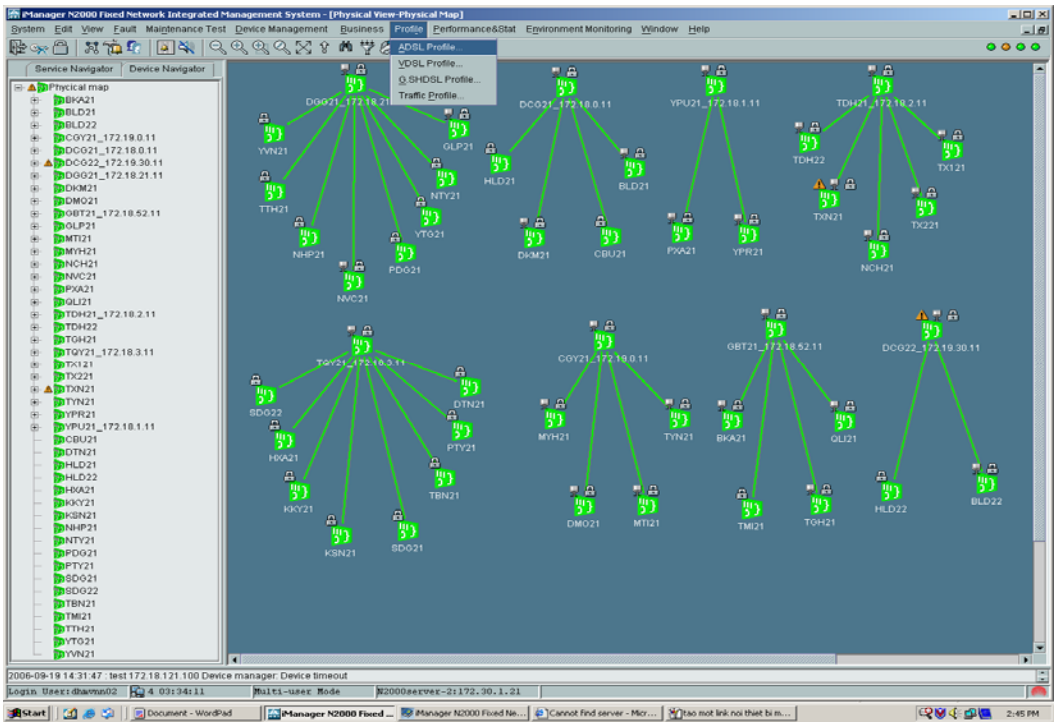
Hình IV-4 Mô hình hệ thống NMS Client/Server

Trong các thiết lập nhỏ, mô hình đứng một mình Standalone được dùng, Client/server được cấu hình trên cùng một Workstation.

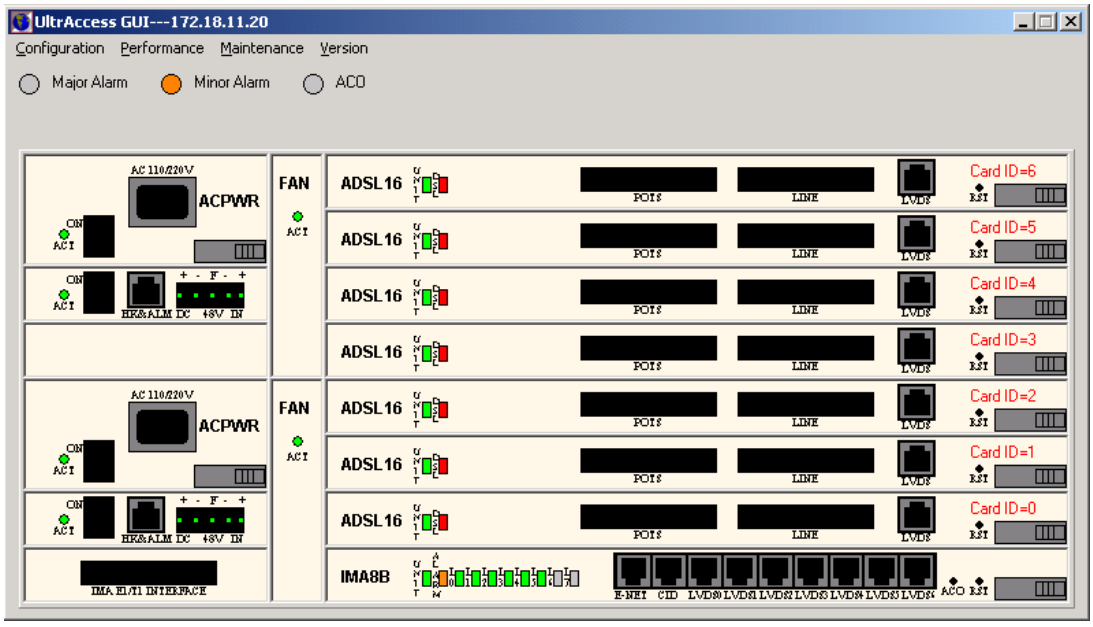
Sau đây là một số hình ảnh giao diện NMS Client của các chủng loại thiết bị DSLAM được sử dụng tại Bưu điện Hà nội



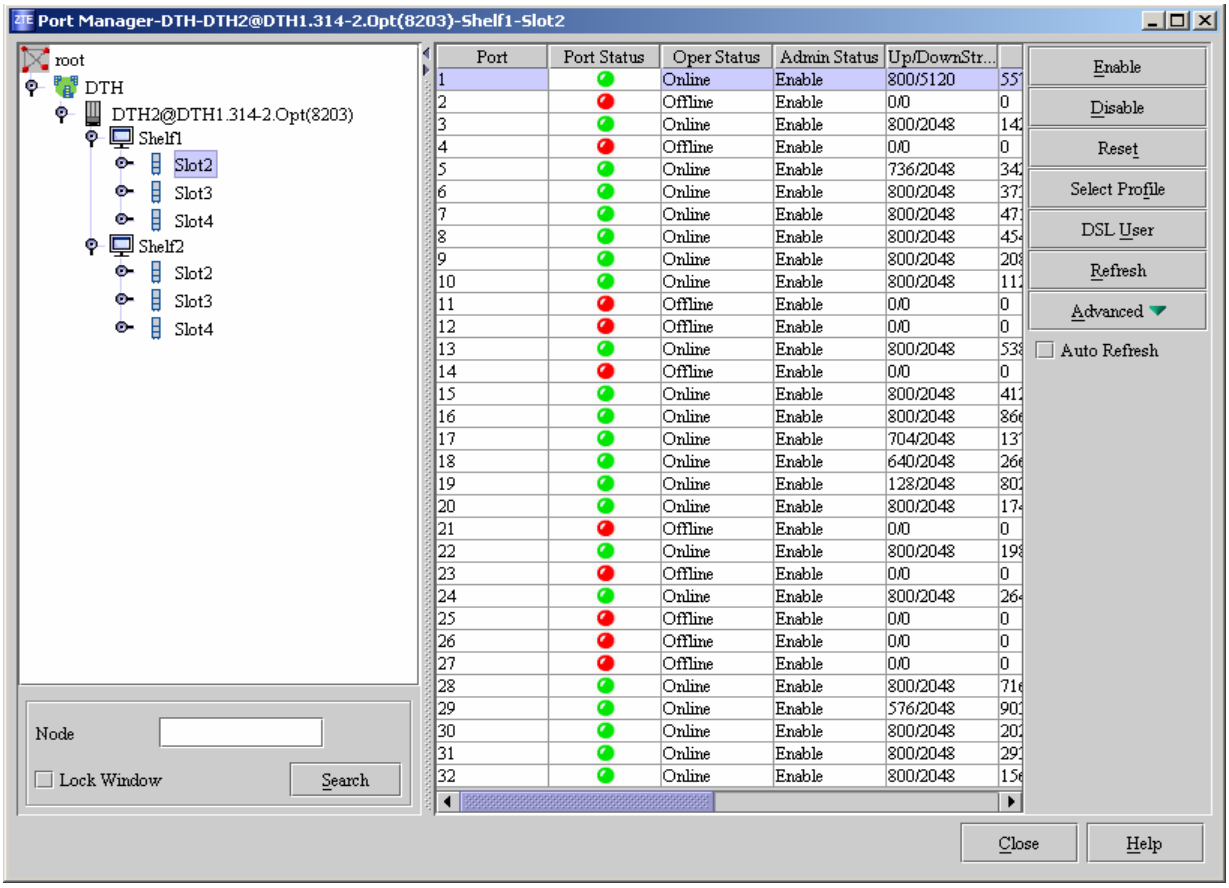
Hình IV-5 Giao diện đồ họa phần mềm quản lý thiết bị SIEMENS (ACI)



Hình IV-6 Giao diện đồ họa phần mềm quản lý thiết bị HUAWEI (iManager N2000)



Hình IV-7 Giao diện đồ họa phần mềm quản lý thiết bị UMAP (UltrAccess GUI)



Hình IV-8 Giao diện đồ họa phần mềm quản lý thiết bị ZTE

IV.1.8. Khảo sát quy trình cung cấp dịch vụ ADSL

Để có thể cung cấp dịch vụ cho khách hàng một cách nhanh chóng, bên cạnh việc thiết lập cấu hình cho các thiết bị DSLAM để có thể hoạt động một cách bình thường theo chỉ dẫn của nhà sản xuất, các DSLAM còn phải được thiết lập cấu hình cho phù hợp với quy trình cung cấp dịch vụ của nhà cung cấp dịch vụ.

Do có một số lượng khách hàng lớn, hệ thống cung cấp rộng khắp trên địa bàn Hà nội, nên

Để phù hợp với mô hình kinh doanh của Bưu điện Hà nội, các DSLAM sẽ được đặt tên theo một chuẩn thống nhất, tạo điều kiện trao đổi thông tin giữa các đơn vị khác nhau. Ví dụ, các DSLAM sẽ được đặt theo bộ mã của các

tổng đài dặt thiết bị: Tổng đài bách khoa được thống nhất đặt tên là BKA và thiết bị DSLAM đặt tại đây sẽ được đặt tên là BKAxy, trong đó xy là chỉ số của DSLAM được lắp đặt tại tổng đài đó (BKA11, BKA12, BKA21…).

Các cổng DSLAM được thống nhất đánh số từ 1 trở đi trên tất cả các loại thiết bị dù thiết bị đó có phương án đánh chỉ mục khác nhau. Ví dụ, DSLAM của Siemens được đánh chỉ số tăng dần theo khe cắm, nhưng các thiết bị khác như Huawei, UMAP thì không có tính năng đó, mà được xác định dưới dạng Shell/Frame/Slot/Port. Điều này tạo ra các bất cập không nhỏ cho bộ phận khai thác mạng khi phải thực hiện động tác chuyển đổi từ thông tin nhận được (Tên DSLAM, số cổng) sang khuôn dạng được quy định bởi nhà sản xuất.

Mặt khác theo khuyến nghị của nhà sản xuất, trước khi cung cấp dịch vụ, các cổng thiết bị nên để ở trạng thái "đóng" – disable để tránh tiêu hao năng lượng và tài nguyên tính toán của DSLAM. Điều đó dẫn đến yêu cầu đối với hệ thống là:

1. Các cổng phải được đưa vào chế độ disable trong các trường hợp:
    a. Chưa có khách hàng đăng ký sử dụng (mặc định khi thiết lập hệ thống)
    b. Khi khách hàng đăng ký tạm ngừng sử dụng
    c. Khi khách hàng nợ cước, buộc phải tạm ngừng cung cấp dịch vụ
    d. Khi khách hàng hủy hợp đồng
    e. Khi khách hàng dịch chuyển sang một vị trí khác (cổng khác)
    f. Khi đấu chuyển thiết bị sang một DSLAM mới (cổng khác)

g. Vv…

2. Các cổng phải được đưa vào chế độ enable trong các trường hợp:
    a. Khách hàng đăng ký sử dụng và đã sẵn sàng sử dụng dịch vụ
    b. Khi khách hàng muốn khôi phục dịch vụ
    c. Khi khách hàng hết nợ cước
    d. Khi khách hàng dịch chuyển đến một vị trí khác (cổng khác)
    e. Khi đấu chuyển thiết bị sang một DSLAM mới (cổng khác)

Bên cạnh việc số lượng các thao tác liên quan đến quá trình phát triển thuê bao ngày càng tăng do nhiều khách hàng sử dụng, hệ thống quản trị mạng còn phải thực hiện nhiều thao tác liên quan đến chính nhà cung cấp dịch vụ.

Do phải thường xuyên cấu hình lại hệ thống mạng để tối đa hóa khả năng cung cấp dịch vụ (điều chuyển dung lượng cổng) số lượng thao tác tăng vọt và kéo theo đó là sự quá tải của các nhân viên kỹ thuật tại trung tâm điều khiển hệ thống.

Một vấn đề khác cũng làm quá tải hệ thống quản lý trị là các yêu cầu cung cấp thông tin về trạng thái cổng của thiết bị.

Các phần mềm quản lý thiết bị của các nhà cung cấp thường được thiết kế cho bộ quản trị nên có rất nhiều tính năng cao cấp, liên quan đến hoạt động của cả hệ thống và chỉ các kỹ thuật viên đã qua đào tạo mới có thể nắm bắt và sử dụng tránh gây mất liên lạc cho toàn hệ thống.

Theo phân cấp quản lý, nhà cung cấp dịch vụ sẽ phải có một bộ phận hỗ trợ khách hàng gián tiếp qua điện thoại và thường bộ phận đó không có quyền quản lý thiết bị mà chỉ có thể thực hiện các thao tác monitoring.

Bên cạnh việc khả năng của các phần mềm quản lý đóng gói do nhà sản xuất cung cấp có những hạn chế nhất định (vì được thiết kế chính cho bộ phận quản lý), nhiều phần mềm còn không có khả năng phân quyền cho người dùng để có thể hạn chế thao tác theo user.

Để dễ hình dung về ta cùng xem xét một ví dụ sau: Khi có khách hàng yêu cầu hỗ trợ do gặp sự cố khi sử dụng dịch vụ, công việc đầu tiên mà bộ phận hỗ trợ khách hàng phải thực hiện là kiểtm tra thông tin về một cổng truy nhập: đóng/mở, trạng thái lỗi, đã được thiết lập các cấu hình cần thiết… Theo cách thức thông thường, các cán bộ kỹ thuật phải thực hiện theo một quy phức tạp bao gồm 8 bước với sự tham gia của 2 cán bộ kỹ thuật của 2 đơn vị khác nhau:

1. Bộ phận hỗ trợ khách hàng nhận được yêu cầu kiểm tra thông số truy nhập
2. Nhân viên hỗ trợ kỹ thuật gọi điện đến bộ phận quản trị mạng
3. Bộ phận quản trị mạng thực hiện xác định chủng loại thiết bị
4. Chuyển sang máy tính hoặc màn hình điều khiển NMS client tương ứng với chủng loại thiết bị đó
5. Tìm đến giao diện màn hình mô tả thiết bị đó
6. Xác định vị trí cổng trên thiết bị (khe số bao nhiêu, cổng thứ mấy trên slot đó)
7. Thực hiện tác vụ được yêu cầu (lấy thông tin trạng thái, đóng, mở cổng…)
8. Thông báo lại cho bên hỗ trợ khách hàng thông tin về trạng thái cổng

Toàn bộ quy trình phức tạp đó đã làm giảm đáng kể chất lượng hỗ trợ khách hàng của mạng MegaVNN cũng như gây quá tải cho các đơn vị tham gia vào quá trình cung cấp dịch vụ.

Sự phát triển của mạng lưới xDSL cả về số lượng và chủng loại thiết bị đã đặt ra một thách thức lớn đối với Bưu điện Hà nội trong việc vận hành, khai thác hệ thống; cũng như ảnh hưởng đến chất lượng các quy trình cung cấp dịch vụ của đơn vị, mà ta có thể tóm tắt lại như sau:

1. Không có chức năng để cho phép các hệ thống hỗ trợ bên ngoài giao tiếp với phần quản lý mạng: Do không có chức năng giao tiếp với các hệ thống hỗ trợ bên ngoài (ví dụ hệ thống quản lý khách hàng, hệ thống hỗ trợ dịch vụ.…), quá trình cung cấp dịch vụ (đóng mở cổng dịch vụ, khởi tạo dịch vụ, tháo hủy dịch vụ…) đều phải chuyển đến kỹ thuật viên khai thác mạng thực hiện bằng nhân công thông qua hệ thống NMS của mỗi hãng; không cho phép kết nối, thực hiện tự động hóa dây chuyền sản xuất, cũng như không thể xây dựng và phát triển thành một giải pháp tổng thể. Điều đó đã dẫn đến các hệ quả tất yếu sau:

    a. Số lượng thao tác hàng ngày tăng lên theo số lượng thuê bao và dịch vụ: Một ngày phải thực hiện nhiều yêu cầu đóng/mở cổng (khi có khách hàng mới hòa mạng, huỷ hợp đồng, nợ, trả nợ cước, vv…). Có những ngày, số lượng yêu cầu lên đến hơn 300; thời gian thực hiện trong từ 7:00 cho đến 21:00 với các quy định chặt chẽ về thời gian để hạn chế tối đa việc mất liên lạc của khách hàng;

    b. Tạo một sức ép không nhỏ đối với quá trình vận hành và khai thác hệ thống do phải sử dụng nhiều loại phần mềm quản lý NMS đối với những công việc hàng ngày (kiểm tra thông số cổng, đóng, mở,

reset cổng) . Thực tế là đã có lúc, cán bộ quản lý mạng phải ngồi trước 04 màn hình NMS và phải thao tác qua lại giữa 4 NMS này;

2. Công tác hỗ trợ và chăm sóc khách hàng gặp nhiều khó khăn: Vì lý do an ninh, bảo mật nên phần quản lý mạng NMS nên kỹ thuật viên tại bộ phận hỗ trợ không có thông tin về trạng thái thiết bị để trả lời và hỗ trợ khách hàng mà phải hỏi thông tin từ bộ phận quản lý mạng NMS, ảnh hưởng không tốt đến chất lượng chăm sóc khách hàng, tốn nhiều nhân lực và mất nhiều thời gian chờ đợi.

3. Khó khăn trong việc tích hợp ứng dụng, nâng cao chất lượng, tùy biến của dịch vụ: Các phần mềm quản lý thiết bị DLSAM được thiết kế cho các nhu cầu quản lý chung nên có nhiều điểm không phù hợp với nhu cầu sử dụng của Bưu điện Hà nội; không tích hợp với các CSDL hiện có của Bưu điện Hà nội, do vậy gặp nhiều khó khăn trong việc tích hợp ứng dụng, nâng cao chất lượng của dịch vụ.

4. Không có một giải pháp tổng thể cho toàn hệ thống: Không có một hãng cung cấp thiết bị DSLAM nào có khả năng cung cấp một giải pháp tổng thể thỏa mãn các yêu cầu trên, do giải pháp thiết bị của mỗi hãng đều khác nhau, các hãng chỉ có thể có khả năng cung cấp giải pháp đối với thiết bị của họ khi có yêu cầu, mà không quan tâm đến thiết bị của các hãng sản xuất khác. Thực tế tại mạng do Bưu điện Hà nội quản lý đã tồn tại thiết bị của 4 hãng sản xuất, trong khi số hãng cung cấp thiết bị trên thị trường Việt nam ước tính lớn hơn 10 hãng.

Sự phát triển ngày càng mạnh mẽ của dịch vụ xDSL với xu hướng nâng cao chất lượng dịch vụ mà vẫn tiết kiệm nguồn nhân lực kỹ thuật cao đòi hỏi phải có một giải pháp giải quyết triệt để các vấn đề đã nêu trên.

Để giải quyết thỏa đáng các vấn đề đã nêu trên, đề tài đang hướng tới mục tiêu xây dựng một giải pháp phần mềm phù hợp với mô hình khai thác, quản lý của nhà cung cấp dịch vụ, đáp ứng các yêu cầu đã đặt ra với các khả năng:

- Cho phép tự động hóa các thao tác khai thác hàng ngày;
- Cung cấp giao tiếp cho phép các ứng dụng/dịch vụ hỗ trợ bên ngoài được giao tiếp với các thiết bị DSLAM. Có thể theo dõi trạng thái thiết bị từ xa, tuỳ theo phân quyền của các đơn vị tham gia khai thác phù hợp với quy trình quản lý dịch vụ của nhà cung cấp dịch vụ, tạo tiền để để tiến tới thực hiện các chức năng quản lý phức tạp hơn…
- Nhất thể hóa giao diện quản lý, giúp người sử dụng tránh việc phải thao tác với nhiều phần mềm quản lý khác nhau;

Nhận thức được ý nghĩa quan trọng của việc tin học hóa, tự động hóa dần các thao tác đơn giản, giải phóng nguồn nhân lực có trình độ cao khỏi các thao tác đơn điệu, cũng như nâng cao chất lượng cung cấp dịch vụ, nhóm thực hiện đề tài sẽ cố gắng hoàn thành đề tài hướng tới khả năng áp dụng vào thực tế không chỉ đối với đơn vị mình, mà có thể áp dụng vào các đơn vị khác trong phạm vi tập đoàn.

IV.2.Quản trị mạng tập trung qua WEB sử dụng CGI

Để xây dựng ứng dụng quản trị mạng tập trung qua web, chúng tôi đã sử dụng các hàm SNMP API trên nền Java của hãng AdventNet.

Java được sử dụng vì những tính chất rất cơ bản của ngôn ngữ lập trình này, đó là tính đơn giản, hướng đối tượng, phù hợp với ngôn ngữ lập trình mạng, có độ bảo mật cao, multi-thread và có thể chạy trên mọi loại máy tính khác nhau.

Đối với một ngôn ngữ lập trình máy tính, Java tuy ra đời muộn (nắm 1995) nhưng đã nhanh chóng trở thành một ngôn ngữ lập trình được phát triển nhanh nhất và được chào đón nhiều nhất trong lịch sử.

Như đã trình bày ở trên, Java có trong nó nhiều đặc tính quý báu, phù hợp với lập trình đa nền và lập trình mạng trên internet: an toàn, không phụ thuộc vào chủng loại máy tính (chạy được trên mọi hệ nền), hướng đối tượng, đa luồng, là ngôn ngữ lập trình mạng và một điều rất quan trọng, đó là Java được tích hợp vào các trình duyệt mạng.

Các ứng dụng quản trị mạng có thể thu hoạch được rất nhiều lợi ích từ Java. Java có một hệ thống thư viện hàm phong phú cho các ứng dụng trên mạng và có thể dễ dàng lập trình cho các ứng dụng dựa trên TCP cũng nhu UDP.

Do là ngôn ngữ lập trình không phụ thuộc vào hệ nền (máy tính và hệ điều hành), các mã chương trình Java có thể chạy trên bất kỳ một máy tính nào trên Internet. Ngoài ra, với các Java applets chạy trong các trình duyệt web, nhà quản trị mạng có thể quản trị các loại thiết bị khác nhau từ xa. Điều đó có được do ngày nay, tất cả máy tính mạng đều được cài dặt sẵn các trình duyệt và tất cả các trình suyệt đều hỗ trợ Java.

Một applet là một chương trình mini chạy bên trong một trình duyệt web. Applets sẽ được tự động download như là một phần của trang web và được nạp vào bộ nhớ của web browser client và cho thực hiện. Thông thường, các applet này sẽ tạo ra các hiệu ứng đồ họa bên trong khu vực hiển thị của trang web.

Nhờ vào cơ chế này mà chúng ta có thể cung cấp các phần mềm nhỏ từ một server trung tâm đến các máy trạm và do các phần mềm nhỏ này được chạy

tại phía client, chũng ta có được một mô hình tính toán phân tán server đến các máy trạm.

Giới thiệu gói phần mềm SNMP của AdventNet SNMP

AdventNet Java SNMP API là một hệ thống các hàm công cụ dạng client-server giành cho các ứng dụng quản trị mạng. AdventNet cung cấp các hàm API cơ bản giúp chúng ta có thể xây dựng được các giải pháp quản trị mạng dựa trên sự kết hợp giữa SNMP và công nghệ Java.

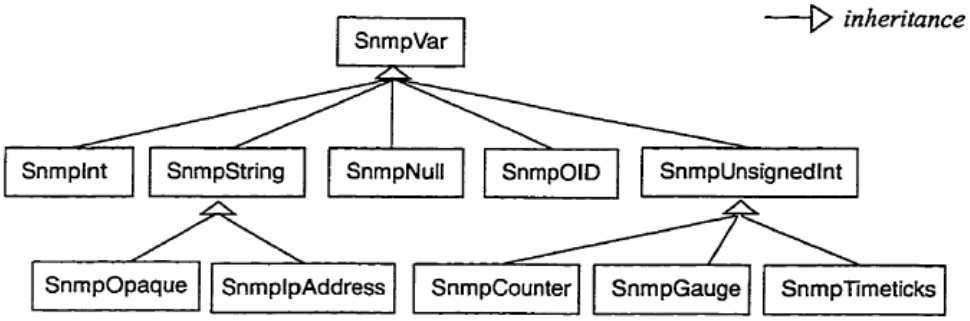
Gói phần mềm này có thể chia thành bốn lớp chính sau:

1. Lớp các biến SNMP
2. Lớp giao tiếp SNMP
3. Lớp liên quan đến các MIB
4. Lớp phụ trợ

Các lớp biến SNMP

Chúng ta đã biết là chuẩn SNMP định nghĩa một số kiểu biến như Integer, Bit String, Octet String, Object Identifier và NULL

AdventNet SNMP Package cũng đưa ra các lớp biến tương ứng với các kiểu biến trên. Đầu tiên là một lớp dạng abstract với tên gọi SnmpVar, chứa các abstract methods giành cho việc in ấn, mã hóa ASN và giải mã, vv… SnmpVar có các lớp con tiếp theo là SnmpInt, SnmpString, SnmpNull, SnmpOID, SnmpUnsignedInt. Các lớp con này lại có các lớp con tiếp theo như: SnmpOpaque, SnmpIpAddess, … xem hình



Hình IV-9 Cấu trúc phân lớp của SnmpVar

Lớp SNMP SNMP Communication

Chúng ta có năm lớp liên quan đến việc trao đổi thông tin qua SNMP:

SnmpAPI: lớp này được tạo ra để quản lý các phiên SNMP được tạo ra bởi các ứng dụng của người dùng, quản lý các modul MIB đã được nạp và lưu trữ các dữ liệu quan trọng trong giao tiếp SNMP. Đây là một lớp rấ t quan trọng của Advent SNMP Package. Trước khi sử dụng bất kỳ một hàm SnmpAPI nào, chúng ta cũng đều phải khởi tạo và chạy lớp này.

SnmpSession: Được dùng để quản lý các phiên làm việc trong một cặp giao tiếp SNMP. lớp này cung cấp các hàm cho phép mở các phiên làm việc đồng bộ hoặc bất đồng bộ; gửi và nhận các yêu cầu SNMP, kiểm tra trả lời hoặc timeout và đóng phiên làm việc. SnmpSession cần phải được khởi tạo trước khi bắt đầu thực hiện liên lạc giữa hai bên SNMP.

SnmpCallback: Được sử dụng khi có một kết quả trả lời không đồng bộ đến một thread nhưng cần được xử lý thêm bởi một thread khác

SnmpPDU: cung cấp các hàm và biến cần thiết để tạo và sử dụng các SNMP PDU. Các method được cung cấp bao gồm thêm các liên kết biến với các OID và các biến Null cho các PDU, in ra tất cả các liên kết biến…

Các lớp liên quan đến SNMP MIB

Chúng ta có một số lớp liên quan đến xử lý các SNMP MIB như sau:

MibModule: Đưa ra một cách phân tích cú pháp và sử dụng các biến dữ liệu trong các file modul MIB. Mỗi một phiên bản MIB được tạo ra từ một MIBModul File và chúng ta có thể nạp vào và loại bỏ các MIBModul này bằng cách tạo hay xóa các phiên bản (instance) này. Thông qua phân tích các modul MIB này, chúng ta có thể nhận được các giá trị do các Agent trả về.

MibMacro: Được sử dụng để phân tích cú pháp các MIB macros. Hiện thời chỉ hỗ trợ các macro dạng OBJECT-TYPE và TRAP-TYPE.

MIBTrap: Được sử dụng với các dữ liệu dạng trap

MibNode: dạng trình bày của một node trong cây MIB. Có nhiều method và thuộc tính được phát triển để đơn giản hóa việc phát triển các ứng dụng có sử dụng các định nghĩa của MIB.

LeafSyntax: Sử dụng để trình bày dạng mô tả của các lá trong cây MIB

Các lớp phụ trợ:

Có một số các lớp được thiết kế nhưng không nằm trong các phân loại trên:

SnmpClient: Đây là một giao diện được sử dụng để thay đổi các diễn biến theo ngầm định của các hàm callbacks, xác thực và đưa ra các thông báo debug.

MibException: Là một lớp dùng để bắt lỗi và diễn giải lỗi (nếu có) trong quá trình thực hiện.

Phần tiếp theo, chúng ta sẽ đi vào các bước cơ bản áp dụng quản trị mạng qua web dựa trên gói AdventNet SNMP.

Xây dựng công cụ quản trị mạng dựa trên CGI

Các yêu cầu cơ bản của hệ thống:

Hệ thống quản trị mạng DSLAM có thể được bắt đầu từ một thiết kế đơn giản: Hệ thống giao tiếp SNMP dựa trên CGI, có khả năng thực hiện các câu lệnh truy vấn đơn giản để lấy các thông tin về hệ thống trong MIB-II. Đây la các thông tin cực kỳ cơ bản đối với một hệ thống quản trị mạng.

Các yêu cầu cần thiết cho NMS_CGI:

(1) Người sử dụng có thể dùng trình duyệt web (NetScape, Internet Explorer,…) để lấy các thông tin cần thiết của hệ thống. Do sử dụng trình duyệt web nên hệ thống nên cung cấp các giao diện đồ họa phù hợp với người sử dụng.

(2) Người sử dụng có thể nhập vào tên hoặc IP của DSLAM.

(3) Tùy theo tên hoặc địa chỉ IP của DSLAM mà NMS_CGI sẽ trả lời với các thông tin tương ứng được định nghĩa trong MIB-II hoặc là thông báo chờ hoặc báo lỗi

(4) Theo RFC 1213, kết quả thông báo về của sysObjectD là OID và SysServices là một số nguyên nên NMS_CGI cần phải thông dịch lại để người sử dụng có thể hiểu được.

(5) Thời gian thực hiện một yêu cầu theo khuyến nghị là 15 giây, sau thời gian trên, hệ thống sẽ đưa ra thông báo timeout.

Dựa vào các yêu cầu trên, hệ thống sẽ được thiết kế theo hướng đối tượng.

IV.2.1. Xây dựng chương trình trên CGI

Trong phần này, chúng ta sẽ luận bàn về cách xây dựng công cụ quản trị mạng dựa trên CGI.

Khi người sử dụng bấm vào một liên kết (link) trên trang web và link đó được trỏ đến một địa chỉ (URL) của một chương trình nằm trên web server,

server sẽ gọi chương trình đó ra để thực hiện. Trước đó, server sẽ chuyển các tham số được yêu cầu vào chương trình thông qua Bô nhập chuẩn (standard input) và các biến môi trường theo đúng quy tắc của CGI.

Sau khi thực hiện xong, chương trình sẽ gửi các kết quả trả về thông qua bộ ra chuẩn (Standard Output) cho web server và đến lượt mình, web serer sẽ trả lại kết quả cho web client.

Trong mô hình này, http client và server cùng đều phải sử dụng chuẩn đặc tả dữ liệu MIME (Multipurpose Intemet Mail Extensions) của Interrnet để mô tả (và thỏaa thuận) về nội dung của các thông điệp. Theo quy tắc của giao thức HTTP, HTTP client và server phải tự thỏa thuận với nhau về cách thức trình bày dữ liệu mỗi khi kết nối được thiết lập.

Một yêu cầu HTTp thông thường có ba phần nhu sau:

```
<method> <resource identifier> <HTTP version><crlf>
```

- <method> là câu lệnh HTTP như GET hay là POST
- <resource identifier> mô tả tên của tài nguyên đích
- <HTTP version> mô tả phiên bản mà phía client sử dụng, ví dụ như HTTP/1.0
- <crlf> là ký tự hết-chuyển về đầu dòng

(2) Các trường chứa thông tin về yêu cầu: <Header>:<Value><Crlf>

Trường header dùng để chưa các thông tin liên quan đến phần header của yêu cầu, có định dạng là tên của header, tiếp theo là giá trị của header và cuối cùng là Crlf. Trường header cuối cùng được kết thúc bằng 2 ký tự CrLf liên tiếp.

(3) Phần thân thực thể (the entity body):

Phần này được client sử dụng để chuyển các thông tin cần thiết liên quan trực tiếp đến yêu cầu lên server

HTTP GET được server sử dụng để tiếp nhận địa chỉ URL và gửi dữ liệu về cho client. HTTP POST cũng tương tự như HTTP GET nhưng có một điểm khác biệt là chúng ta không thể gửi một lượng dữ liệu lớn hơn 256 ký tự (hay 1024 ký tự tùy theo hệ thống) thông qua lệnh GET. Do đó, trong thực tế, khi có các yêu cầu gửi về server, chúng ta thường hay sử dụng lệnh HTTP POST hơn. Listing sau là một đoạn mã java sử dụng Java Socket để tạo một kết nối giữa client và server.

```
import java.lang.*;
import java.util.*;
import java.net.*;
import java.io.*;

class ClientCGI {
      Socket socket = null;
   private String[] msg = {“”, “”, “”, “”, “”, “”, “”, “”, “”, “”};
   private boolean MSG = false, ERR = false;
   private String script = "/cgi-bin/dslamnet/snmpGet”;
   private String line = “”;

   public ClientCGI(String str) {
      String data = new String(“hostname “ + str);
      try (
            socket = new Socket( “172.30.1.2”, 80);
            DataoutputStream ostream = new
                        DataOutputStream(socket.getOutputStream());
            DataInputStream istream = new
                        DataInputStream(socket,getInputStream());
            ostream,writeBytes("POST " + script + " HTTP/l.0\r\n”
                  + "Content-type: application/octet-stream\r\n”
                  + "Content-length: + data.length() + "\r\n\r\n”;)
            ostream, writeBytes (data) ;
            ostream,close();
            line = istream.readLine();
            if(line.equals("Warning!”) (
                  ... // Xu ly loi
            ) else if(line.equals("Messaqe!")){
                  ... // Co ket qua gui ve
            }
            istream.close() ;
```

```
      } catch (Exception e) {
            ... // Xu ly loi
      }
   ...
}
```

Chúng ta cũng có thể sử dụng lớp URLConnebtion đã được viết sẵn trong môi trường Java.net. Khi đó thì mã lệnh sẽ được giản lược đi rất nhiều. Thay vì phải làm việc với TCP connection với các dữ liệu gốc, chúng ta chỉ cần làm chỉ rõ URL và gửi thẳng đến server. URLClass sẽ thực hiện phần công việc còn lại.

```
public ClientCGI (String str) {
      try {
            URL snmpserver = new URL("172.30.1.2' +
                        "/cgi-bin/DSLAMnet/snmpGet”);
            URLConnection               connection              =
            snmpServer.openConnection();
            connection.setDoOutput(true);
            PrintStream ostream = new
                        PrintStream(connection.getOutputStream());
            ostream.println(str);
            ostream.close ();
            BufferedReader istream = new BufferedReader(
                  new InputStreamReader(
                        connection.GetInputStream()));
            line = istream.readLine();
            ...
      }
      ...
}
```

Listing ClientCGI.java sử dụng URL Class

Phần mã java chạy ở phía Client còn đơn giản hơn nhiều so với đoạn mã chạy trên server. Tất cả các công việc cần làm là nhận số liệu từ Standard input và gửi kết quả ra standard outphut

```
class RequestHandler {
      public static void main(String[] args) {
            String line = null, error = null, rdata;
            RequestHandler request_handler = new RequestHandler();
            try {
                  BufferedReader istream = new BufferedReader(
                        new InputStreamReader(System.in.Read())
```

```
                line = istream.readline();
                if(line != null ) {
                      ... ; // Lay du lieu va ghi vao rdata
                      System.out.println("Content-Type:
                            text/plain\n\nMeseage!\n" + rdata);
                } else {
                      System.out.println("Content-Type:
                            text/plain\n\nMeseage!\n" + error);
                }
                istream.close();
          } catch (Exception e){
                System.out.println("Content-Type:
                            text/plain\n\nWarning!\n" + e);
          }
          System.exit(0) ;
     }
     ..   // Bat dau tien trinh phan tich rData va
          //gui cac snmp query den cac DSLAM
}
```

Listing ServerCGI.java RequestHandler.java

IV.2.2. Xây dựng chương trình gửi nhận SNMP

Sử dụng gói phần mềm AdventNet SNMP, lớp SnmpTask.java đã được viết với mục đích thực hiện các tác vụ SNMP khi được yêu cầu.

Theo tài liệu hướng dẫn của AvantNet, bất kỳ một ứng dụng nào muốn sử dụng gói phần mềm này đều phải khởi tạo và chạy lớp snmpAPI. Sau đó, ứung dụng sẽ phải nạp modul MIB để có thể nhận được các giá trị tương ứng từ SNMP agent.

Tiếp đó, ứng dụng sẽ phải mở một phiên bản của SnmpSession để liên lạc với các SNMP agent. Theo tài liệu của AvantNet, chúng ta có thể tạo không hạn chế các phiên làm việc nhưng cần phải lưu ý rằng, các phiên làm việc này thực chất là các thread và việc mở quá nhiều hoặc duy trì nhiều thread chạy song song với nhau là không cần thiết.

```
class SnmpTask{
     private MibModule     module = nuil;
     private SnmpOID       oid;
```

```
      private SnmpPDU       pdu = null, re_pdu= null;
      private SmpVarBind    varbind = null;
      private SnmpVar       var = null;
      private SnmpSession   session = null;
      private SnmpAPI       api = null;
      private byte          commnand;
      private String        errMsg = “”;
      private boolean       eStat = false;

      public SnmpTask(String host , String community) {
            // Instantiate and start SnmpAPI
            api = new SnmpAPI();
            api.start() ;
            command = api.GET_REQ_MSG; // change to GET operation
            // Load the MIB Module
            try {
                  module = new MibModule("rfcl213-MIB”, api,
                              api.DEBUG);
            } catch (Exception e) {
                  errMsg = "Loi: Doc/xuly MIB URL: " + e;
                  return;
            }
            // Instantiate SnmpSession
            session = new snmpSession(api);
            session.peername = host;
            session.comunity = community;
            session.remote_port = 161;
            session.timeout = 15000; // 15 seconds
            session.retries = 0;
            opensession();
      }
}     // end of snmpTask
```

Listing Xây dựng lớp snmpTask

Constructor của lớp này sử dụng method openSession, đơn giản là thực hiện việc mở một phiên SNMP và xử lý các lỗi phát sinh

```
// Open session
private void opensession() {
      try {
            session.open();
      } catch (Exception e) {
            errMsg = “khong the mo duoc phien SNMP. Eror: “
                        + e.getMessage
            eStat = true;
      }
}
```

Listing Mở một phiên làm việc của SnmpTask.java

Tiếp theo, ta sẽ cần phải tạo một SNMP PDU và chuyên dữ liệu đến SNMP ở đầu xa. Công việc đầu tiên là khởi tạo và gắn PDU với một lệnh cụ thể (xem ví dụ). Các câu lệnh cụ thể đã được định nghĩa sẵn trong lợp SnmpAPI dưới dạng BYTE: GET_REQ_MSG, SET_REQ_MSG tương ứng với các tác vụ Get và Set của SNMP.

```
// Tao SnmpPDU
private void buildPDU(byte cmd) {
      pdu = new SnmpPDU(api);
      pdu.command = cmd;
}

private void buildPDUex(String poid){
      SnmpOID oid = new snmpOID(poid, api);
      pdu.addNull(oid);
}

//Gui SnmpPDU
private SnmpPDU sendPDU() {
      SnmpPDU response_pdu = null;
      try {
            response_pdu = session.syncSend(pdu);
      } catch (SnmpException e) {
            errMsg = "Sending PDU: " + e .getMessage();
            return null;
      }
      if (response_pdu == null) {
            . . .

      } else {
            return response_pdu
      }
}
```

Listing các method tạo và gửi PDU

Sau khi một phiên làm việc SNMP đã được mở, chúng ta có thể bắt đầu gửi các lệnh SNMP đến các agent thông qua các phiên làm việc đó. Việc gửi các PDU có thể được thực hiện dưới 2 hình thức: đồng bộ (synchronous) và bất đồng bộ (asynchronous). Với phương pháp xử lý đồng bộ, ứng dụng sẽ tạm ngừng tại thời điểm đó và đợi cho đến khi có dữ liệu được gửi về hoặc phiên làm việc bị timeout.

Sau khi đã viết xong các đoạn chương trình gửi nhận thông qua web, công việc tiếp theo sẽ là tạo một hệ thống hoàn chỉnh để kiểm tra.

Công việc đầu tiên là phải tạo một file .html để có thể download từ trên mạng thông qua web client.

```
<HTML>
<HEAD> <TITLE>DSLAMNET SYSTEM</TITLE></HEAD>
<BODY>
<CENTER>
<APPLET CODEBASE="http://172.30.1.2/cgi-bin/DSLAMnet/”,
           CODE= “ClientApplet.class”,
           archive=ClientAppletJar.jar, width=550 height=380>
     <B>Sorry, your web browser should support Java1.1</B>
</APPLET>
</CENTER>
</BODY>
</HTML>
```

JAR là từ viết tắt của Java Archive, được sử dụng trong môi trường Java để nén và trao đối nhiều file khác nhau (ví dụ như các java class) trong một file. Trong trường hợp này, browser sẽ chỉ cần tạo một kết nối đến web server để tải file JAR về để chạy Java applet. Bên cạnh đó, sử dụng file JAR còn tiết kiệm được thời gian tải file do dữ liệu đã được nén từ trước.

Về nguyên tắc, các file đó nên được đặt ở một vị trí mà client có quền truy xuất và nên được để chung trong một thư mục (ví dụ /CGI-BIN/). Các file đặt trong thư mục này là:

(1) các file chứa các class chạy trên server như là RequestHandler.class, SnmpTask.class, vv…

(2) AdventNet SNMP package

(3) Một file .bat hoặc .vbs có dòng lệnh:

```
java RequestHandler
```

Lưu đồ hoạt động:

Hình IV-9 minh họa các giao tác cần thực hiện giữa client terminal và các phần của hệ thống trong mô hình quản trị mạng sử dụng CGI:

(1) Người sử dụng nhập vào tên của DSLAM và số cổng trên thiết bị và chọn thao tác cần thực hiện rồi bấm vào nút submit

(2) Applet lấy số liệu do người sử dụng nhập vào và chuyển đến cho đối tượng ClientCGI

(3) ClientCGI chuyển dữ liệu về cho web server

(4) Web server phân tích yêu cầu, thực hiện một truy vấn vào cơ sở dữ liệu dựa trên tên của DSLAM để xác định:

   a. Địa chỉ IP của DSLAM

   b. Chủng loại của DSLAM

   c. Các SNMP community dùng cho lệnh GET và SET của DSLAM

   d. Các OID cần thiết tương ứng với cổng và câu lệnh cần thực hiện

(5) RequestHandler được gọi bởi web server sẽ lấy dữ liệu thông qua biến môi trường và standard input và gọi snmpTask;

(6) SnmpTask sẽ gửi các gói SMP PDU đến các DSLAM để thực hiện các yêu cầu.

(7) SNMP Agent tại các DSLAM nhận yêu cầu, xử lý và trả lại kết quả cho Web server (SnmpTask)

(8) SnmpTask chuyển kết quả về cho RequestHandler

(9) RequestHandler tạo ra kết quả dưới dạng chuỗi và gửi về cho web server

(10) Web server chuyển kết quả về cho ClientCGI

(11) ClientCGI chuyển kết quả về cho JavaApplet

(12) JavaApplet hiển thị kết quả cho người sử dụng: tình trạng thực hiện câu lệnh hoặc là thông báo lỗi

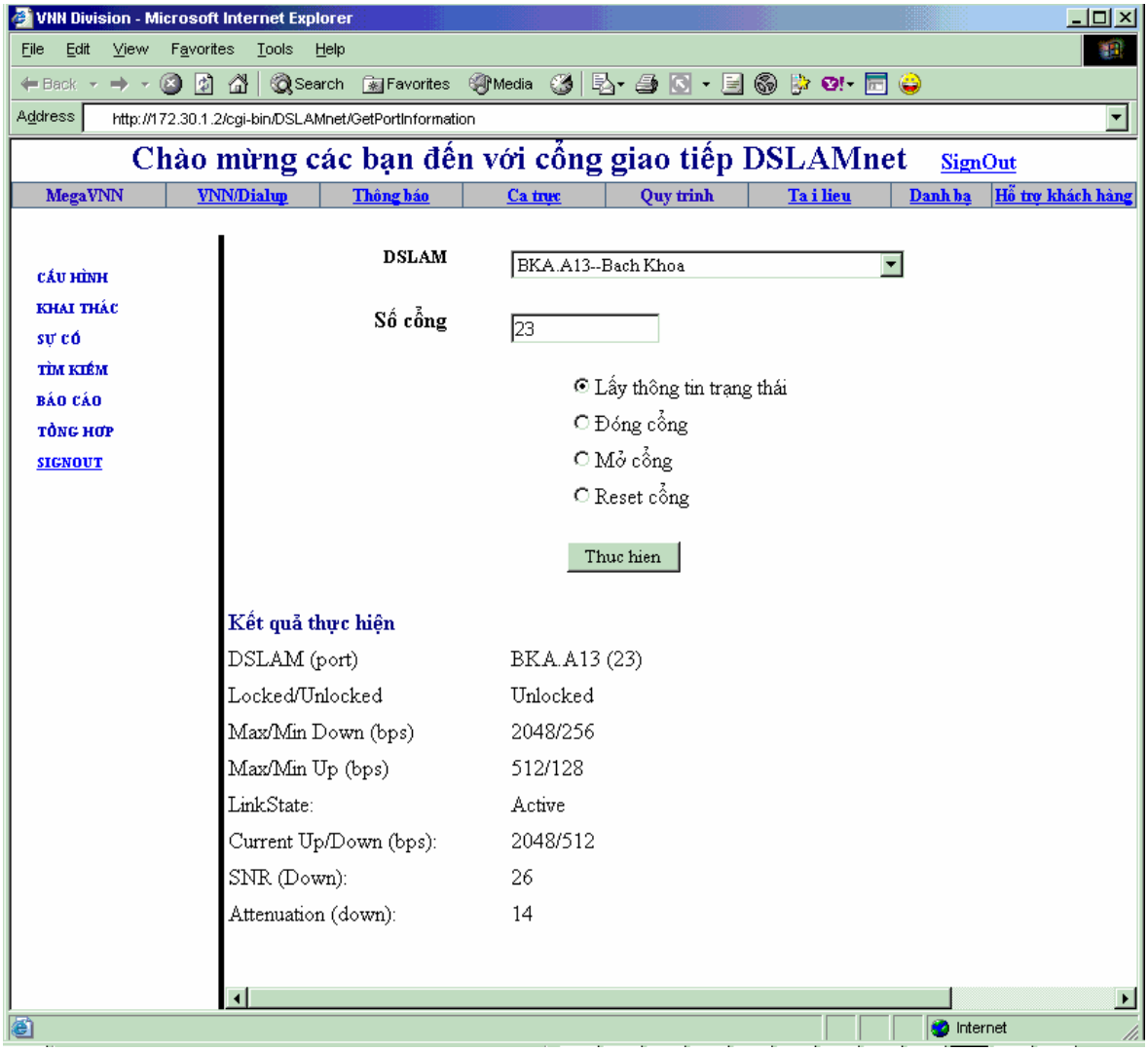
Trên đây là nguyên tắc cơ bản để xây dựng hệ thống theo dõi và quản trị DSLAM dựa trên CGI.

Chúng ta có thể thấy ở đây có sự tham gia của 3 thực thể, đó là:

(1) WebClient tại máy tính của người sử dụng

(2) WebServer tại điểm giao tiếp giữa mạng của người sử dụng và mạng các DSLAM

(3) SNMP Agent tại các DSLAM



Hình IV-10 Giao diện của DSLAMnet

Ở mô hình này, ta có thể nhận thấy gánh nặng tính toán đã được đặt lên Web server do phải làm điểm giao tiếp với các bên và thực hiện các phép tính khác như truy vấn cơ sở dữ liệu, tính toán các tham số thiết bị dựa trên cổng và chủng loại DSLAM.

Trong trường hợp mạng cung cấp dịch vụ có nhiều thiết bị và chủng loại khác nhau, web server sẽ trở thành điểm nghẽn của toàn bộ dịch vụ.

IV.3.Quản trị mạng tập trung qua WEB sử dụng CORBA

Để xây dựng ứng dụng quản trị mạng tập trung qua web, sử dụng CORBA chúng tôi đã sử dụng sản phẩm VisiBroker của hãng Borland

Giới thiệu VisiBroker

Như đã đề cập đến ở các chương trước, CORBA là một trong các công nghệ trung gian (middleware) trong công nghệ tính toán phân tán trên mạng. CORBA là các tiêu chuẩn chung được định nghĩa bởi OMG với mục đích tạo ra một giao tiếp thống nhất cho các ứng dụng hướng đối tượng trên mạng không đồng nhất. Để có thể sử dụng được CORBA, chúng ta sẽ cần có một phần mềm tạo một môi trường nền để phát triển ứng dụng. VisiBroker là một trong các sản phẩm hàng đầu hỗ trợ việc phát triển, triển khai các đối tượng ứng dụng phân tán trên mạng với các phần cứng và phần nền khác nhau và hoàn toàn tương thích với CORBA.

Visibroker có nhiều packet khác nhau cho các ngôn ngữ lập trình khác nhau. Để phục vụ cho công việc của mình, chúng tôi đã sử dụng gói phần mềm hỗ trợ Java để có thể đối chiếu với hệ thống trên CGI.

ORB của VissiBroker được viết hoàn toàn bằng Java nên có thể được sử dụng và phát triển dưới dạng có thể tải về dưới dạng ORBlet.

Có ba thành phần chính được đóng gói kèm theo VissiBroker cho Java. Đó là: dịch vụ tên (Naming Service), dịch vụ sự kiện (Event Service) và GateKeeper.

Naming Service cho phép gán nhiều hơn một tên logic cho một đối tượng thực hiện và lưu trong vùng namespace của dịch vụ.

Event Service cung cấp các tiện ích để chúng ta có thể tách riêng các trao đổi giữa các loại đối tượng khác nhau và nhờ đó, nhiều đối tượng có thể gửi

dữ liệu theo phương thức bất đồng bộ đến nhiều đối tượng sử dụng dữ liều thông qua các kênh riêng sự kiện riêng.

Gatekeeper là dịch vụ chạy trên web server và có thể cho phép client nhận các callback kể cả khi trong hệ thống có sử dụng firewall. Đây là một thành phần rất quan trọng đối với các ứng dụng trên nền web, đặc biệt là trong mô hình ba bên Client/server như đã trình bày ở các chương trước.

VisiBroker còn đưa ra các công cụ phát triển tiên tiến như idl2ir, idl2java, java2iiop và java2idl. IDL2java là một công cụ rất cơ bản để viết các chương trình Java có thể sử dụng được Visibroker ORB. Đó thực chất là các trình tiền biên dịch được sửu dụng để sinh ra các đặc tả Java (stub) cho các đối tượng Client và bộ khung (skeletions) cho các đối tượng server từ một file IDL.

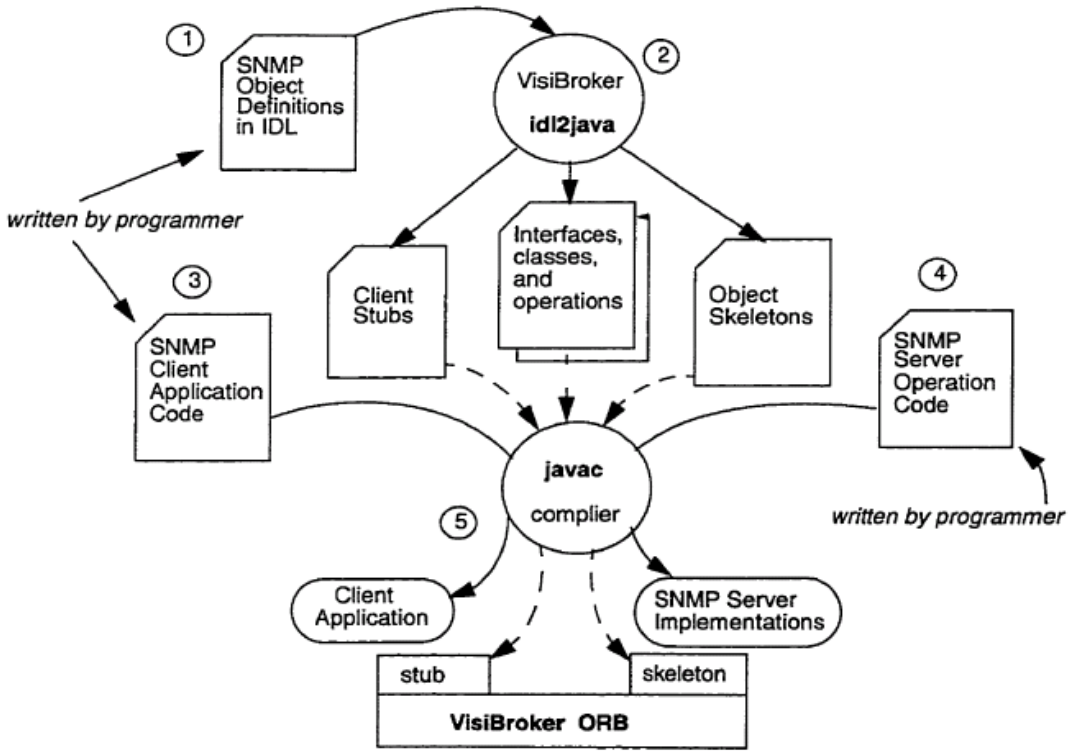
VissiBroker còn đưa ra các công cụ mạnh cho việc xây dựng các ứng dụng CORBA, đặc biệt là các ứng dụng trên nền web.

IV.3.1. Xây dựng ứng dụng với VisiBroker

Hình IV-11 mô tả quy trình xây dựng một ứng dụng CORBA. Như đã đề cập đến ở phần trước, để có thể chạy được trên nền CORBA, các đối tượng phải được mô tả bởi một file IDL – đưa ra các đặc tả của đối tượng dịch vụ sẽ được cung cấp cho các client. Định nghĩa này bao gồm kiểu của đối tượng, các thuộc tính của đối tượng, và danh sách các method mà đối tượng đó cung cấp cũng như các tham số của method đó.

Sau khi đã viết xong file IDL, chúng ta có thể sử dụng trình tiền biên dịch của VissiBroker idl2java, ta sẽ có được các file class, trong đó đã bao gồm các đoạn mã cần thiết để thực hiện chương trình (stub code cho Client và

Skeletion cho Server). Các file class này được chứa trong các đóng gói mà chúng ta hay gọi là các modul trong file DIL



Hình IV-11 Lưu đồ xây dựng hệ thống quản trị mạng DSLAM với VisiBroker

Sau khi đã viết xong các chương trình ứng dụng client và server, công việc tiếp theo là phải biên dịch lại thành một chương trình Java hoàn chỉnh (.class) và cài đặt vào hệ thống

IV.3.2. Xây dựng công cụ quản trị mạng xDSL sử dụng CORBA

Về cơ bản, các yêu cầu đối với hệ thống sử dụng CORBA và CGI là như nhau. Trong mô hình sử dụng CORBA, chúng ta sẽ xem xét mô hình bốn bên, điều vẫn chưa thể thực hiện được ở mô hình CGI truyền thống.

Công việc đầu tiên cần thực hiện là chúng ta phải viết được file mô tả IDL theo đúng yêu cầu của hệ thống.

Trước hết, chúng ta sẽ phải đưa ra định nghĩa của giao diện cho các tác vụ căn bản nhất nhà SnmpGet và SnmpSet. Sau đây là đoạn mã của file DSLAMnet.idl.

```
module SnmpSys {
      struct sysData {
            string s_Descr;
            string s_Oid;
            string s_The;
            string s_Con;
            string s_Name;
            string s_Locat;
            string s_Serv;
            string error;
      };
      exception OprException {
            string reason;
      };
      interface OprInterface{
            sysData snmpGet(in string host) raises (OprException);
      };

      ...
};
```

Chúng ta cần lưu ý rằng từ khóa modul chính là tên của nhóm các giao diện và các dữ liệu chứa trong nó. Có thể nói nó có vai trò tương tự như các java package và bổ sung thêm một mức định nghĩa về cấu trúc phân tầng của IDL namespace.

Trong DSLAMnet.idl, chúng ta đã đưa ra các định nghĩa về tên của modul, cấu trúc dữ liệu, exeption và 02 giao diện giành cho các tác vụ snmp cơ bản là OprInterfaceGet (giành cho GetRequest) và OprInterfaceSet (giành cho SetRequest).

Sau khi đã mô tả xong, chúng ta có thể dụng trình tiền biên dịch idl2Java của VisiBroker để sinh một số file java cần thiết. Các file đó sẽ được lưu trong mộ thư mục con tên file như được mô tả ở DSLAMnet.idl (SnmpSys). Một số file cần thiết của hệ thống:

Oprlnterface.java: Khai báo của các giao diện

OprlnterfaceHelper.java – Khai báo lớp OprlnterfaceHelper. Lớp này đưa ra định nghĩa các hàm tiện ích như là bind, read, write, insert, vv…

OprException.java – Mô tả OprException class, được sử dụng để chuyển thông báo lỗi thông qua ORB

SysData.java: File được sử dụng để tạo đối tượng Sysdata, dùng để chuyển dữ liệu qua ORB

St_Oprlnterface.java - stub code cho đối tượng Oprlnterface ở phía Client.

OprInterfaceImpBase.javaL skeleton code cho đối tượng Oprlnterface ở phía Server.

…

Mã chương trình ở phía Client có thể được sử dụng lại từ phần xây dựng phần mềm quản lý dựa trên CGI. Chỉ cần thực hiện một số thay đổi nhỏ ở đoạn mã applet như sau:

```
//ClientOrbApplet.java
import java.awt.*;
import java.awt.event.*;
...
public class ClientOrbApplet extends Applet \
      implements ActionListener {
      private SnmpSys.OprInterface op_interface;
      . . .
      public void init() {
            ...
      // Initialize the ORB.
            org.omg.CORBA.ORB orb = \
            org.omg.CORBA.ORB.init(this, null);
            interface = SnmpSys.OprInterfaceHelper.bind(orb,\
                  "System Operation");
      }
      ... // Su dung oprInterface.snmpGet de lay thong tin
}
```

Thay đổi chủ yếu ở đây là chúng ta sẽ phải khởi tạo ORB và sử dụng OprInterfaceHelper.bind để tạo ra một đối tượng OprInteface và sau đó mới thực hiện snmpGet thông qua bindhelper này.

ORB class cung cấp các hàm hỗ trợ, được sử dụng ở cả hai phía vlient và server. Để khởi tạo VisiBroker ORB, chúng ta sẽ phải gọi đến hàm init(). Hàm init() này có thể được gọi với cac tham số this (tham chiếu đến chính bản thân applet). Bằng cách này, ORB client sẽ thiết lập một kết nối đến một phiên của Broker Gatekeeper, chạy ở trên máy server, tức là nơi mà applet được tải về. Gatekeeper có nhiệm vụ giúp cho client xác định và sử dụng các đối tượng không nằm trên web server (nằm ở một máy tính khác) và cho phép nhận các callback, điều không thể thực hiện được do yêu cầu bảo mật của các web browser. Các web browser áp dụng hai kiểu hạn chế vì lý do bảo mật đối với các Java applet (còn được gọi là SandBox):

- Các applet chỉ được kết nối ngược lại đến các máy tính mà từ đó, applet được tải về
- Các applet chỉ được chấp nhận các kết nối đến từ host mà applet đó được tải về;

Để có thể có các tham chiếu đến các đối tượng ở xa như OprInterface. chúng ta phải tạo sự gắn kết (dùng hàm bind) của OprInterrfafeHelper. Sau khi applet gọi đến hàm bind, ORB sẽ "nói chuyện" với SmartAgent để xác định server ứng với OprInterface.

Bước tiếp theo ORB sẽ thử thiết lập kết nố giữa applet và server này. Nếu ORB không thể tìm được serer hoặc không thể thiếp lập được kết nối, bind sẽ trả về một lỗi hệ thống CORBA

Công việc tiếp theo là xây dựng đoạn chương trình chạy trên phía server. Chúng ta đã có các file OprInterface.java (lớp OprInterface) nằm trong gói SnmpSys. Cũng giống như ở phần CGI, ở đây chúng ta cũng chỉ cần viết lại đoạn mã chương trình thực thi bên server.

Để thực hiện điều này, chúng ta chia các hàm thực thi phía server thành 2 lớp:

- SnmpServer.java xử lý các yêu cầu từ phía client
- OprInterfaceImp.java phần thực thi của OprInterface.

Sau đây là đoạn mã chương trình của SnmpServer.java.

```
//SnmpServer.java
public class SnmpServer{
public static void main(String[] args) {
     try {
          // Initialize the ORB.
          org.ong.CORBA.ORB orb =
               org.omg.CORBA.ORB,init(args,null);
          // Initialize the BOA.
          org.omg.CORBA.BOA boa = orb.BOA_init();
          // Create the Snmp Operation object.
          SnmpSys.OprInterface opi = new
               OprInterfaceImpl(“System Operation");
          // Export the newly created object.
          boa.obj_is_ready( opi);
          system.out.println(opi + " is ready. " );
          // Wait for incoming requets
          boa.impl_is_ready ();
     } catch (Exception e) {
          // something failed.,,
          Sysrem.out.println(e);
     }
}
```

Đầu tiên ORB và BOA phải được khởi tạo trước khi tạo bất kỳ một đối tượng CORBA nào. BOA là chữ viết tắt của Basic Object Adapter và được các phần thực thi của các đối tượng sử dụng để kích hoạt và deactive các đối tượng mà chúng cung cấp cho các client. nếu sử dụng BOA_init mà không

có tham số vào, chúng ta đơn giản là chấp nhận chính sách chung đối với các thread – thread pooling.

Oprlnterface được tạo ra khi chạy lớp OprInterfaceImpl. Bước tiếp theo là đăng ký với BOA thông qua method obj_is_ready và nhờ đó mà các client trên mạng có thể "nhìn" thấy đối tượng này thông qua ORB.

Cuối cùng, boa.impl_is_ready() sẽ được BOA gọi để đưa server vào vòng lặp vô hạn để đợi các lời gọi đến và chuyển đến đối tượng tương ứng.

Listing sau là đoạng mã minh họa của OprInterfaceImp.java, với các mở rộng của skeletion _OprInterfaceImplBase là phần thực thi lõi của OprInterface, trong đó có các tác vụ thực thi của snmpGet

```
import java.io.*;
import java.net.*;
import javaeutil.*;

public class OprInterfaceImpl extends
                    SnmpSys._OprInterfaceImplBase {
public OprInterfaceImpl(String name) {
     super (name) ;
}

/** Construct a transient object. */
public OprInterfaceImpl() {
     super() ;
}

public SnmpSys.sysData snmpGet(String host){
     throws SnmpSys.0ptException {
          SnmpSys.sysData re_data;
          String warning ;
          try {
               SnmpOpr snmpopr = new SnmpOpr(host, "public");
               . . . // thuc hien gui snmp PDU
               return re_data;
          } catch (Exception e) {
               System.out.println("System      Exception      in
          snmpGet\n"
                              + e);
               return null;
          }
     }
```

```
}
```

Sau khi đã hoàn thành chương trình java, chúng ta sẽ sử dụng trình biên dịch Java để dịch thành byte code. Do đây là ứng dụng qua web nên chúng cũng sẽ phải tạo một trang HTML có "nhúng" các mã java applet cần thiết: Client.java và các file jar của Vbjorb.jar. VBJorb.jar là dile chưa các đối tượng ORB giành cho phía Client.

```
<html>
<head>
<title>DSLAMnet CORBA</title>
</head>
<body bgcolor="#C5C5C5”>
<center>
<applet
      Code=”ClientOrbApplet.class"            ARCHIVE=”Client.jar,
vbjorb.jar”
           width=510 height=360>
      <param name = org.omg.CORBA.ORBClass
                value=com.visigenic.vbroker.orb.ORB>
      <param name = ORBgatekeeperIOR
                value=http://172.18.1.2:15000/gatekeeper.ior”
      <B>Sorry, your web browser should support Javal.l</B>
</applet>
</center>
</body>
</html>
```

Có 2 tham số được sử dụng trong DSLAMnet_CORBA.html, đó là Visibroker và URL của file IOR được sinh ra bởi Gatekeeper/ ORB ở bên client sử dụng giá trị này để tìm IOR file. Trong trường hợp này, GateKeeper được chạy ở cổng 15000 trên chính web server (172.18.1.2). Tất nhiên là gatekeeper phải được chạy ở trên máy chủ này từ trước.

SNMP server cũng phải được khởi tạo để có thể tiếp nhận và chuyển tiếp các yêu cầu từ phía web server. SNMP server có thể được đặt ở trên cùng máy chủ web hoặc ở trên một máy khác. Đây là một tính năng mà chỉ khi dùng khi CORBA ta mới sử dụng được.

Chương V. Kết luận và hướng phát triển

V.1. Các kết quả đã đạt được

Kết quả nghiên cứu của đề tài đã được áp dụng vào thực tế, xây dựng thành công hệ thống phần mềm DSLAMnet quản lý các thiết bị DSLAM trong mạng cung cấp dịch vụ của Bưu điện Hà nội.

Phần mềm đã được triển khai trong thực tế, cung cấp được các thông tin cần thiết cho người sử dụng về trạng thái cổng của các DSLAM cho phép bộ phận hỗ trợ khách hàng có được các thông tin tức thời nhanh chóng chính xác về tình trạng kết nối của khách hàng, tạo điều kiện để phục vụ khách hàng nhanh chóng và hiệu quả, giải phóng bộ phận quản trị mạng khỏi các thao tác hỗ trợ khách hàng thông thường, góp phần rõ rệt trong việc nâng cao hiệu suất làm việc của các bộ phận hỗ trợ khách hàng trực tiếp, gián tiếp, các đơn vị đại lý và giảm tải cho bộ phận quản trị mạng.

Xây dựng trên nền công nghệ CORBA 2.0 và CGI , sử dụng JDK 1.5.06, hệ thống đã được kiểm tra với trình duyệt Internet Explorer và Netscape Navigator trên các máy tính PC sử dụng Windows 2000. Nói chung, trong các trường hợp, hệ thống đã thực hiện được các chức năng thiết kế, đảm bảo được các yêu cầu đã đặt ra.

V.2. Kết luận

Trong đề tài này, chúng ta đã thực hiện việc xem xét chuẩn quản lý mạng SNMP và cách thức xây dựng một ứng dụng quản trị mạng trên nền Web cho các DSLAM dựa trên với công nghệ CGI truyền thống và một hướng tiếp cận mới thông qua CORBA. Qua thực hiện đề tài, ta có thể rút ra các kết luận sau:

- SNMP là một giao thức rất tốt cho việc quản trị mạng, không chỉ trong thực hiện nhiệm vụ theo dõi, giám sát hệ thống mà còn có thể áp dụng một cách khá thành công trong việc điều khiển các thiết bị trên mạng.
- Các chương trình được viết bằng ngôn ngữ Java cho thấy Java là một ngôn ngữ lập trình mạnh, phù hợp với môi trường mạng. Do được thiết kế hướng đối tượng và không phục thuộc vào hệ nền nên các chương trình Java có thể chạy ở trên nhiều hệ thống khác nhau.
- Công nghệ CGI và CORBA đã được áp dụng để xây dựng phần mềm quản trị các DSLAM qua web. Từ góc độ lập trình, CORBA hơn hẳn CGI nhờ tính trong suốt địa phương (local/remote transparency) và các hỗ trợ mức cao trong việc truyền dữ liệu, thực hiện các thủ tục gọi hàm;
- Các ứng dụng chạy trên CGI chậm hơn so với CORBA;
- Hệ thống sử dụng CORBA đã được kiểm thử với các cấu hình khác nhau như sau: (1) Web và SNMP server được đặt trên cùng một máy tính; và (2) Web và SNMP server được đặt trên hai máy tính khác nhau. Trường hợp (1) cho kết quả thực hiện nhanh hơn so với trường hợp (2). Tuy nhiên, điều đó dẫn đến sự tăng tải của máy chủ và nếu đưa hệ thống vào hoạt động trên quy mô rộng thì rất có thế sẽ làm quá tải máy tính và làm chậm tốc độ chung của cả hệ thống;

V.3. Khả năng mở rộng:

- Phân cấp hóa hệ thống quản trị có thể được cài đặt theo từng phân đoạn mạng riêng biệt (phân biệt theo đơn vị quản lý hoặc theo nhà sản

xuất) hoặc mở rộng theo mô hình nhiều SNMP server cũng như nhiều Web server với mục đích phân tải hệ thống;

- Bổ sung thêm tính năng bảo mật như mã hóa dữ liệu trên đường truyền;
- Bổ sung thêm các tính năng phân quyền theo nhóm và người sử dụng theo chức năng cũng như theo phân vùng thiết bị;
- Phát triển một ứng dụng quản trị mạng hoàn chỉnh hơn có khả năng nhận được các thông điệp SNMP trap ngay tại trình duyệt của người sử dụng nhờ vào tính năng callback thông qua IIOP của CORBA. Nhờ đó, người sử dụng có khả năng nhận biết được các bất thường của hệ thống như mất quản lý của một card dịch vụ, bị quá tải, số lượng gói tin lỗi vượt quá một ngưỡng nào đó vv…
- Đóng gói một số thành phần cơ bản của hệ thống,phục vụ cho việc chuyển đổi sang các ngôn ngữ lập trình khác, có hiệu năng cao hơn. Trong ứng dụng này sử dụng Java làm ngôn ngữ lập trình với CORBA. Tuy nhiên, do CORBA có cung cấp chuẩn kết nối cho các ngôn ngữ lập trình cao cấp nên chúng ta cũng có thể thiết kế một số đối tượng bằng các ngôn ngữ lập trình khác như C++ để cải tiến tốc độ thực thi.

V.3.1. Kết luận

Sau một thời gian nghiên cứu và hoàn thành luận văn, tác giả đã nắm bắt được các khái niệm tổng quát và các lý thuyết căn bản về SNMP, CGI và CORBA cũng như ngôn ngữ lập trình Java.

Đề tài cũng đã nêu rõ các chi tiết để áp dụng những cơ sở lý luận này vào phát triển mô hình cụ thể của một giải pháp quản trị mạng các thiết bị

DSLAM dựa trên công nghệ WEB với nền tảng CORBA hoặc CGI và giao thức SNMP.

Luận văn đã thực hiện được các nội dung và đạt được các mục tiêu đề ra như trong bản đề cương đã được duyệt. Các kết quả đạt được bao gồm:

- Nắm được các đặc tả cơ bản của chuẩn SNMP.
- Nắm được mô hình, cơ chế hoạt động, hệ thống quản trị mạng dựa trên SNMP và áp dụng công nghệ CGI vào quản trị mạng.
- Áp dụng công nghệ CGI, CORBA vào quản trị mạng.
- Xây dựng được một ứng dụng để quản trị các thiết bị DSLAM đang được khai thác tại Bưu điện Hà nội trên WEB.

Các kết quả đạt được mở ra nhiều hướng phát triển tiếp cho đề tài, tuy nhiên vẫn còn một số vấn đề mà luận văn chưa đề cập đến. Một số hướng phát triển khác nữa có thể mở rộng như: hoàn thiện hơn hệ giao diện với người sử dụng, danh sách các DSLAM nên được lấy từ một cơ sở dữ liệu, thay vì lấy từ một file text, phát triển thêm các khả năng bảo mật, mã hóa dữ liệu vv…

Mặc dù đã cố gắng trong nghiên cứu và thực hiện đề tài, nhưng vì thời gian và trình độ có hạn, chắc chắn luận văn không tránh khỏi nhiều thiếu sót. Em xin bày tỏ lòng biết ơn sâu sắc tới tiến sỹ Hà Quốc Trung, người đã tận tình giảng dạy và hướng dẫn tôi hoàn thành bản luận văn này. Cũng xin bầy tỏ lòng biết ơn tới các thầy, cô và các anh, chị ở khoa Công nghệ Thông tin và Trung tâm Đào tạo sau Đại học đã nhiệt tình giảng dạy và giúp đỡ em trong suốt thời gian học tập vừa qua.

Xin chân thành cảm ơn các bạn học và đồng nghiệp đã giúp đỡ tôi nhiệt tình trong quá trình trong quá trình học tập, nghiên cứu và thử nghiệm vào thực tế đề tài này.

Tài liệu tham khảo

[ietf] The Internet Engineering Task Force http://www.ietf.org/rfc

[Stallings 96] Stallings W. "*SNMP, SNMP v2 and RMON 2nd edition*", 1996

[Stallings 98] Stallings W. "*SNMPv3: A Security Enhancement for SNMP*", "http://www.comsoc.org/livepubs/surveys/public/4q98issue/stallings.html", 1998

[SnmpFAQ] *SNMP FAQ* http://www.faqs.org/faqs/snmp-faq/

[perkins] Perkins D., McGinnis E., "*Understanding SNMP MIBs*", 1996

[Java] Sun Microsystems, Inc. "*The Java Language: An Oveview*", ""http://java.sun.com/docs/overviews/java/java-overview-l.html"

[AdventNet] AdventNet, Inc, "*AdventNet SNMP API 4*" http://www.adventnet.com"

[CGIPerl] Scott G., Shishir G., Gunther B. *CGI Programming with Perl, Second Edition*, 2000

[Weinman] Weinman W., "*The CGI Book*", 1996

[Tittel96] Tittel E., Gaither M., et al. "*Web Programming Secrets with HTML, CGI, and Perl*", 1996

[CGI2] "*Perl, CGI, and JavaScript Complete, 2nd Edition*", By Sybex Inc. 2000

[CGI201] Hamilton J., "*CGI Programming 201*", By Amazon 12, 2002

[VBJ] *Borland VisiBroker* http://www.borland.com/us/products/visibroker/index.html

[Rosenberger] Rosenberger, J. "*Teach Yourself Corba in 14 Days, Second Edition*", 2000

[Orfali] Robert Orfali R., Harkey D., "*Client/Server Programming with Java and CORBA, 2nd Edition*", 1998

Mazumdar S., "*Inter-Domain Management between CORBA and SNMP: WEB-based Management - Corba/Snmp Gateway Approach*", "http://www.bell-labs.com/project/CorbaSnmp/NeoORBImpl/", 1996

[CORBA] CORBA, "Catalog of OMG CORBA®/IIOP® Specifications", Revision 2.1, 1997

[OMG] Object Management Group, Framingham, Mass, 1998 – "*The Common Object Request Brocker : Architecture and Specification*", Rev. 2.2 ftp://ftp.omg.org/pub/docs/formal/98-07-01.pdf

[CORBA14] Jeremy L. Rosenberger, "*Teach Yourself Corba in 14 Days (Sams Teach Yourself)*", Sams Publishing 1999

[CORBA3.0] Steve V., "New Features for CORBA 3.0", IONA Technologies, Inc.. 2001

[OMG_ARCH] Framingham, Object Management Group, "*The Common Object Request Broker: Architecture and Specification*", 1998.

[Coulouris] Coulouris G., Dollimore J. và Kindberg T. "*Distributed Systems: Concepts and Design (4th Edition)*", August 11, 2000)

[TL_CORBA] Nhóm học viên Cao học Xử lý thông tin và truyền thông 2004 môn học "Hệ phân tán" của lớp cao học Xử lý thông tin và truyền thông 2004, Đại học Bách Khóa Hà nội, Tiểu luận: "*Tìm hiểu kiến trúc CORBA*" .