**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

A close up of a sign

Description automatically generated

**ĐỒ ÁN TỐT NGHIỆP**

**Xây dựng ứng dụng hỗ trợ quản trị thiết bị mạng trong hệ thống quy mô nhỏ**

**NGUYỄN TRẦN BẢO DANH**

danh.ntb140645@sis.hust.edu.vn

**Ngành Công nghệ thông tin**

**Chuyên ngành Truyền thông và Mạng máy tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | Ths. Bùi Trọng Tùng |
| **Bộ môn:** | Truyền thông và Mạng máy tính |
| **Viện:** | Công nghệ thông tin và Truyền thông |

**HÀ NỘI, 12/2019**

Phiếu giao nhiệm vụ đồ án tốt nghiệp

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Trần Bảo Danh

Điện thoại liên lạc: 0979328111

Email: danh.ntb140645@sis.hust.edu.vn

Lớp: CNTT 1.2-K59

Hệ đào tạo: Kỹ Sư

Đồ án tốt nghiệp được thực hiện tại: Bộ môn Truyền thông và Mạng máy tính

Thời gian làm ĐATN: Từ ngày 27/09/2019 đến 27/ 12/ 2019

2. Mục đích nội dung của ĐATN

Xây dựng ứng dụng hỗ trợ quản trị thiết bị mạng trong hệ thống quy mô nhỏ

3. Các nhiệm vụ cụ thể của ĐATN

- Lý thuyết: Tìm hiểu về giao thức SNMP

- Thực hành: Viết công cụ thu thập thông tin thiết bị mạng dựa trên giao thức SNMP.

4. Lời cam đoan của sinh viên:

Tôi Nguyễn Trần Bảo Danh cam kết đồ án tốt nghiệp này là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của Ths. Bùi Trọng Tùng. Các nội dung đã nêu trong đồ án tốt nghiệp là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 27 tháng 12 năm 2019*  Tác giả đồ án  *Nguyễn Trần Bảo Danh* |

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 27 tháng 12 năm 2019*  *Giảng viên hướng dẫn* |

**Lời cảm ơn**

Sau hơn 5 năm học đại học của mình, em đã học được rất nhiều những kiến thức và kỹ năng quan trọng để ứng dụng trong công việc, trong cuộc sống từ những người thầy, người cô giáo của viện Công nghệ thông tin và Truyền thông nói chung, bộ môn Truyền thông và Mạng máy tính nói riêng. Vì vậy, đầu tiên em xin được gửi lời cảm ơn chân thành đến các thầy cô của viện Công nghệ thông tin và Truyền thông. Em xin chúc các thầy cô luôn mạnh khỏe để tiếp tục công việc giảng dạy, truyền đạt kiến thức, kinh nghiệm của mình.

Đặc biệt, em xin được gửi lời cảm ơn sâu sắc đến thầy Bùi Trọng Tùng, người không chỉ giảng dạy cho em các kiến thức mà còn trực tiếp hướng dẫn, giúp đỡ em hoàn thành đồ án này.

Em cũng xin gửi lời cảm ơn đến gia đình, bạn bè đã giúp đỡ, tạo điều kiên trong suốt quá trình học tập và thực hiện đồ án của mình.

Em xin chân thành cảm ơn!

**Tóm tắt nội dung đồ án**

Đề tài đồ án được đưa ra trong bối cảnh quản lý hoạt động của các thiết bị mạng đã trở thành một vấn đề cần thiết với sự phát triển của mạng máy tính ngày nay. Mục đích chính của đề tài là xây dựng một ứng dụng hỗ trợ quản lý thiết bị mạng. Chương trình được xây dựng cần đạt được các mục tiêu cơ bản trong việc hỗ trợ quản trị thiết bị như thu thập các thông tin thiết bị cho người dùng quan sát để có thể đánh giá được hiệu quả hoạt động của thiết bị, thông báo cho người dùng những sự kiện bất thường có thể xảy ra. Để đạt được các mục tiêu trên, đồ án đã được thực hiện và trải qua các giai đoạn đặt vấn đề, định hướng giải pháp, công nghệ, tìm hiểu cơ sở lý thuyết của giao thức sử dụng, phân tích thiết kế chương trình, cài đặt và triển khai thử nghiệm. Chương trình sản phẩm kết quả đã đạt được các mục tiêu cơ bản đưa ra, nhưng vẫn cần nhiều những cải tiến để có thể đáp ứng được sự phát triển, biến đổi không ngừng của môi trường mạng máy tính hiện nay.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 27 tháng 12 năm 2019*  Tác giả đồ án |

**Mục lục**

[PHẦN MỞ ĐẦU 8](#_Toc28269753)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 10](#_Toc28269754)

[1.1 Giới thiệu về giao thức SNMP 10](#_Toc28269755)

[1.1.1 Mô hình hoạt động 10](#_Toc28269756)

[1.1.2 Một số khái niệm cơ bản 11](#_Toc28269757)

[1.1.3 Đặc điểm hoạt động của giao thức 12](#_Toc28269758)

[1.1.4 Khuôn dạng thông điệp truy vấn, hồi đáp và phân loại 13](#_Toc28269759)

[1.2 Management information base 15](#_Toc28269760)

[1.2.1 Giới thiệu chung 15](#_Toc28269761)

[1.2.2 Phạm vi ứng dụng 16](#_Toc28269762)

[1.3 Trap 19](#_Toc28269763)

[1.3.1 Khuôn dạng thông điệp 19](#_Toc28269764)

[1.3.2 Các loại Trap 20](#_Toc28269765)

[CHƯƠNG 2. PHÂN TÍCH VÀ ĐẶC TẢ CHỨC NĂNG 22](#_Toc28269766)

[2.1 Các chức năng cơ bản 22](#_Toc28269767)

[2.2 Các chức năng mở rộng 27](#_Toc28269768)

[CHƯƠNG 3. THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH 30](#_Toc28269769)

[3.1 Thiết kế chức năng 30](#_Toc28269770)

[3.1.1 Các chức năng cơ bản 30](#_Toc28269771)

[3.1.2 Các chức năng mở rộng 40](#_Toc28269772)

[3.2 Thiết kế cấu trúc chương trình 43](#_Toc28269773)

[3.2.1 Thiết kế các class của package gui 43](#_Toc28269774)

[3.2.2 Thiết kế các class trong package controllers 50](#_Toc28269775)

[3.2.3 Thiết kế các class trong package data 53](#_Toc28269776)

[3.2.4 Thiết kế các class trong package snmpd 56](#_Toc28269777)

[3.3 Thiết kế giao diện 61](#_Toc28269778)

[3.4 Thiết kế cơ sở dữ liệu 68](#_Toc28269779)

[CHƯƠNG 4. TRIỂN KHAI THỬ NGHIỆM 73](#_Toc28269780)

[4.1 Môi trường triển khai thử nghiệm 73](#_Toc28269781)

[4.1.1 Công cụ sử dụng 73](#_Toc28269782)

[4.1.2 Môi trường triển khai 73](#_Toc28269783)

[4.2 Mô hình triển khai thử nghiệm 73](#_Toc28269784)

[4.2.1 Biểu đồ triển khai và mô tả 73](#_Toc28269785)

[4.2.2 Thông số các thiết bị sử dụng 74](#_Toc28269786)

[4.3 Thử nghiệm và kết quả. 74](#_Toc28269787)

[4.4 Đánh giá kết quả thử nghiệm 75](#_Toc28269788)

[PHẦN KẾT LUẬN 76](#_Toc28269789)

[TÀI LIỆU THAM KHẢO 77](#_Toc28269790)

**Danh mục hình vẽ**

[Hình 1.1: Minh họa mô hình giao thức SNMP 10](#_Toc28257299)

[Hình 1.2: Dạng cây mô phỏng của MIB 16](#_Toc28257300)

[Hình 2.1: Biểu đồ ca sử dụng chức năng cơ bản 22](#_Toc28257301)

[Hình 2.2: Biểu đồ ca sử dụng các chức năng mở rộng 27](#_Toc28257302)

[Hình 3.1: Biểu đồ trình tự của ca sử dụng “Import device info” 30](#_Toc28257303)

[Hình 3.2: Biểu đồ trình tự của ca sử dụng “Update device basic info” 31](#_Toc28257304)

[Hình 3.3: Biểu đồ trình tự 1 của ca sử dụng “Query device info automatically” 32](#_Toc28257305)

[Hình 3.4: Biểu đồ trình tự 2 của ca sử dụng “Query device info automatically” 34](#_Toc28257306)

[Hình 3.5: Biểu đồ trình tự của ca sử dụng “Observe device network interfaces” 35](#_Toc28257307)

[Hình 3.6: Biểu đồ trình tự của ca sử dụng “Observe device resources” 36](#_Toc28257308)

[Hình 3.7: Biểu đồ trình tự của ca sử dụng “Observe statistics” 36](#_Toc28257309)

[Hình 3.8: Biểu đồ trình tự của ca sử dụng “Process Traps” 37](#_Toc28257310)

[Hình 3.9: Biểu đồ trình tự của ca sử dụng “Observe Notifications” 38](#_Toc28257311)

[Hình 3.10: Biểu đồ trình tự của ca sử dụng “Access device” 39](#_Toc28257312)

[Hình 3.11: Biểu đồ trình tự của ca sử dụng “Import template” 40](#_Toc28257313)

[Hình 3.12: Biểu đồ trình tự của ca sử dụng “Edit template” 41](#_Toc28257314)

[Hình 3.13: Biểu đồ của ca sử dụng “Query with template” 42](#_Toc28257315)

[Hình 3.14: Các package trong thiết kế của chương trình 43](#_Toc28257316)

[Hình 3.15: Biểu đồ lớp 1 của package gui 44](#_Toc28257317)

[Hình 3.16: Biểu đồ lớp 2 của package gui 47](#_Toc28257318)

[Hình 3.17: Biểu đồ lớp 3 của package gui 49](#_Toc28257319)

[Hình 3.18: Biểu đồ lớp của package controllers 50](#_Toc28257320)

[Hình 3.19: Biểu đồ lớp 1 của package data 53](#_Toc28257321)

[Hình 3.20: Biểu đồ lớp 2 của package data 55](#_Toc28257322)

[Hình 3.21: Biểu đồ lớp 1 của package snmpd 57](#_Toc28257323)

[Hình 3.22: Biểu đồ lớp 2 của package snmpd 58](#_Toc28257324)

[Hình 3.23: Biểu đồ lớp 3 của package snmpd 60](#_Toc28257325)

[Hình 3.24: Giao diện thông tin cơ bản của một thiết bị 62](#_Toc28257326)

[Hình 3.25: Giao diện hiển thị thông tin sử dụng tài nguyên trên thiết bị 63](#_Toc28257327)

[Hình 3.26: Giao diện các biểu đồ thống kê 64](#_Toc28257328)

[Hình 3.27: Giao diện hiển thị kết quả truy vấn với template 65](#_Toc28257329)

[Hình 3.28: Giao diện chương trình hiển thị thông tin về giao diện mạng 66](#_Toc28257330)

[Hình 3.29: Giao diện hiện thị chi tiết template 67](#_Toc28257331)

[Hình 3.30: Giao diện danh sách các thông điệp Notifications 68](#_Toc28257332)

[Hình 3.31: Biểu đồ thực thể quan hệ của chương trình 69](#_Toc28257333)

Danh mục các bảng

[Bảng 1.1: Một số khái niệm cơ bản 11](#_Toc28257334)

[Bảng 1.2: Một số lỗi cơ bản xảy ra khi xử lý truy vấn trong SNMP 14](#_Toc28257335)

[Bảng 1.3: Các loại thông điệp cơ bản của SNMP 14](#_Toc28257336)

[Bảng 1.4: Các loại thông điệp Trap cơ bản 20](#_Toc28257337)

[Bảng 2.1: Bảng đặc tả ca sử dụng “ Import device info” 23](#_Toc28257338)

[Bảng 2.2: Bảng đặc tả ca sử dụng “Update device basic info” 23](#_Toc28257339)

[Bảng 2.3: Bảng đặc tả ca sử dụng “Query device info automatically” 23](#_Toc28257340)

[Bảng 2.4: Bảng đặc tả ca sử dụng “Observe device resources” 24](#_Toc28257341)

[Bảng 2.5: Bảng đặc tả ca sử dụng “Observe device network interfaces” 24](#_Toc28257342)

[Bảng 2.6: Bảng đặc tả ca sử dụng “Observe device statistics” 25](#_Toc28257343)

[Bảng 2.7: Bảng đặc tả ca sử dụng “Observe notifications” 25](#_Toc28257344)

[Bảng 2.8: Bảng đặc tả ca sử dụng “Access device” 26](#_Toc28257345)

[Bảng 2.9: Bảng đặc tả ca sử dụng “Process response” 26](#_Toc28257346)

[Bảng 2.10: Bảng đặc tả ca sử dụng “Process Traps” 26](#_Toc28257347)

[Bảng 2.11: Bảng đặc tả ca sử dụng “Import template” 27](#_Toc28257348)

[Bảng 2.12: Bảng đặc tả của ca sử dụng “Edit template” 28](#_Toc28257349)

[Bảng 2.13: Bảng đặc tả ca sử dụng “Query with template” 28](#_Toc28257350)

**Danh mục từ viết tắt**

|  |  |
| --- | --- |
| SNMP | Simple Network Management Protocol |
| NMS | Network Management Station |
| MIB | Management Information Base |
| OID | Object Identifier |
| PDU | Protocol Data Unit |

# PHẦN MỞ ĐẦU

Ngày nay, do tính phổ biến và các lợi ích đem lại, mạng máy tính dần đóng một vai trò không nhỏ trong sự phát triển của mỗi cá nhân, mỗi tổ chức và doanh nghiệp, cũng như của toàn xã hội. Nhờ vậy, hệ thống mạng và các thiết bị chuyên dụng dành được rất nhiều sự quan tâm, chú ý trong việc bảo trì, cải tiến và nâng cấp. Từ đấy nảy sinh ra nhu cầu về mặt quản lý các thiết bị trong mạng, đặc biệt là quản lý về hiệu năng hoạt động và quản lý lỗi. Quản lý hiệu năng hoạt động là quản lý khả năng hoạt động, cung ứng dịch vụ của các thiết bị trong mạng như PC, router, switch, … Còn quản lý lỗi là theo dõi, thông báo các lỗi đã và có thể xảy ra, cũng như đề xuất hướng sửa chữa, khắc phục.

Đề tài này được thực hiện với mục đích chính là xây dựng một ứng dụng nhằm đáp ứng phần nào các nhu cầu và vấn đề đưa ra ở trên. Để đạt được điều đấy, quá trình thưc hiện cần hoàn thành các mục tiêu, nhiệm vụ cơ bản sau. Một là xây dựng một hệ thống thu thập thông tin về sử dụng tài nguyên phần cứng, đường truyền, phục vụ việc đánh giá trạng thái và khả năng cung ứng dịch vụ của các thiết bị đang hoạt động. Hai là cung cấp một cơ chế giám sát, thông báo lỗi khi có vấn đề xảy ra và có thể đưa ra những cảnh báo, định hướng sửa chửa, khắc phục cho người dùng. Bên cạnh đấy, chương trình cũng nên được phát triển theo định hướng hỗ trợ người sử dụng trong việc quản trị hệ thống mạng của họ.

Do quy trình phát triển, thực hiện có quy mô nhỏ nên sản phẩm kết quả đạt được sẽ hướng đến việc ứng dụng trong các doanh nghiệp, tổ chức có mô hình hoạt động, kinh doanh không lớn. Các doanh nghiệp này thường không có ngân sách để chi trả cho các phần mềm quản trị mạng chuyên dụng trong khi hệ thống mạng của họ thường ổn định, không quá phức tạp, phù hợp với cơ chế hoạt động căn bản của sản phẩm làm ra. Ngoài ra, chương trình cũng được thiết kế theo hướng đơn giản về mặt sử dụng, thích hợp cho những người quản trị mạng có trình độ thấp. Bởi vậy, sản phẩm đầu ra của dự án hoàn toàn phù hợp với các loại hình doanh nghiệp, tổ chức quy mô nhỏ, tiết kiệm cho họ chi phí trong quá trình tuyển dụng nhân lực.

Nhắc đến việc giám sát hiệu năng mạng, hiệu quả hoạt động của thiết bị có thể nhắc đến hai phương pháp là giám sát bị động và giám sát chủ động. Phương pháp đầu tiên thường thực hiện với chỉ giao thức SNMP trong khi phương pháp còn lại được thường triển khai với RMON. Trong mô hình thưc hiện của phương pháp giám sát mạng bị động, các thiết bị mạng sẽ lưu lại thông tin trạng thái của mình vào cơ sở dữ liệu cục bộ (SNMP MIB) và chỉ trả lời khi có các thông điệp yêu cầu từ trạm quản lý. Do vậy, trạm quản lý cần gửi di các thông điệp thu thập thông tin một cách thường xuyên và sẽ gặp khó khăn trong quá trình tổng hợp, thống kê khi kích thước mạng là lớn, với lượng tài nguyên mạng nhiều. Phương pháp còn lại, giám sát chủ động thường gắn với việc sử dụng mô hình của RMON. Trong mô hình này, một số thiết bị với thành phần phần cứng đặc biệt sẽ đóng vai trò là probe. Các Probe liên tục thăm dò, thu thập thông tin tài nguyên trong mạng nội bộ, rồi tổng hợp, thống kê lại và lưu trữ vào cơ sở dữ liệu (RMON MIB). Khi liên kết với trạm quản lý khả dụng, các probe sẽ chủ động gửi các dữ liệu tổng hợp được về trạm quản lý để phục vụ cho việc giám sát, đáng giá tình trạng, hiệu năng hoạt động của mạng. Có thể nhận thấy phương pháp giám sát chủ động với RMON hỗ trợ tốt hơn trong việc thu thập, thống kê thông tin tài nguyên mạng, nhưng tiêu tốn nhiều băng thông hơn so với phương pháp còn lại sử dụng SNMP. Chưa kể các thiết bị thăm dò của RMON cần cấu hình phức tạp mới có thể sử dụng trong khi sản phẩm của để tài chỉ hướng đến việc quản trị thiết bị, sử dụng cho người dùng cơ bản nên đề tài này sẽ lựa chọn phương pháp giám sát bị động với giao thức SNMP làm định hướng giải pháp.

Quá trình thực hiện đề tài đồ án trải qua nhiều giai đoạn khác nhau nhưng có thể khái quát thành các giai đoạn sau. Đầu tiên là giai đoạn tìm hiểu cơ sở lý thuyết. Chương trình của dự án được phát triển trên nền tảng của giao thức SNMP nên phần tìm hiểu lý thuyết này chủ yếu là về đặc điểm, mô hình hoạt động và các thông điệp trao đổi của giao thức đấy. Tiếp theo trong quá trình thực hiện dự án là khâu phân tích các chức năng cần có cho ứng dụng để đáp ứng các mục tiêu đã đề ra ở trên và thiết kế, cài đặt thành chương trình sản phẩm. Cuối cùng sau khi sản phẩm hoàn thành thì chương trình sẽ được tiến hành đem ra thử nghiệm và đánh giá trong các trường hợp khác nhau để tiếp tục sửa chữa và cải thiện.

Quá trình thực hiện đề tài sử dụng các công cụ khác nhau tùy theo từng giai đoạn. Giai đoạn xây dựng chương trình sử dụng môi trường của ngôn ngữ lập trình Java để phát triển. Đây là một ngôn ngữ phổ biến trên thế giới vì sự đơn giản và tính hướng đối tượng của mình, phù hợp cho việc xây dựng một ứng dụng đa nền tảng cùng với khả năng mở rộng, cải tiến dễ dàng. Với việc sử dung Java làm ngôn ngữ lập trình thì JDK 1.8 cùng với Netbeans IDE được lựa chọn làm môi trường phát triển. Netbeans IDE là một công cụ tuy không còn phổ biến như Eclipse nhưng hiện vẫn đang được duy trì và tiếp tục phát triển bởi Apache. JDK 1.8 chỉ cung cấp những chức năng căn bản của Java nên quá trình xây dựng chương trình còn cần sử dụng thêm các thư viện bên ngoài như snmp4j (cung cấp các chức năng hoạt động theo giao thức SNMP), mibble (thư viện phân tích mib), tnm4j (giảm thiểu sự phức tạp cho hệ thống), xchart (dùng trong vẽ đồ thị), jch (cung cấp các chức năng liên quan đến SSH) và hibernate (framework để tương tác với cơ sở dữ liệu). Ngoài ra, do chương trình cần phải lưu lại một số loại thông tin nên quá trình phát triển và triển khai sẽ sử dụng thêm hệ quản trị cơ sở dữ liệu MySQL để quản lý việc lưu trữ.

# CƠ SỞ LÝ THUYẾT

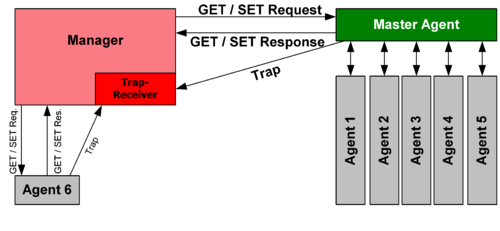
Chương trình của dự án được xây dựng dựa trên giao thức SNMP, Simple Network Management Protocol. Đây là một giao thức cơ bản của mạng IP cho việc thu thập và quản lý thông tin trạng thái của các thiết bị trong mạng. Ngoài ra, qua việc thay đổi các thông số kỹ thuật này, người quản trị mạng cũng có thể cấu hình thay đổi hoạt động của thiết bị được quản lý. Do giao thức này là một giao thức chuẩn và phổ biến nên được nhiều loại thiết bị từ các nhà cung cấp khác nhau hỗ trợ, có thể kể đến như modem, router, workstation, printer, …

Sau khoảng thời gian dài từ khi ra đời đến nay, SNMP đã trải qua nhiều thế hệ khác nhau. SNMPv1 là phiên bản thế hệ đầu tiên của giao thức. Các thế hệ sau SNMPv2 và SNMPv3 là bản cải tiến của phiên bản này về mặt hiêu năng hoạt động, sự linh hoạt và cả tính bảo mật nhưng đều không thay đổi nhiều về mặt mô hình hoạt động. Để có thể sử dụng rộng rãi, chương trình sản phẩm được xây dựng dựa trên phiên bản SNMPv2c, một biến thể phổ biến ở thế hệ thứ hai của giao thức.

## Giới thiệu về giao thức SNMP

### Mô hình hoạt động

Một hệ thống triển khai giao thức SNMP thường gồm một hoặc nhiều máy quản lý gọi là SNMP manager kết nối với các máy trạm, thiết bị chịu sự quản lý khác trong mạng, được minh họa cơ bản như hình bên dưới.



Hình 1.1: Minh họa mô hình giao thức SNMP

Trong đó, trên các máy quản lý thường triển khai một ứng dụng, gọi là NMS, network management station. Ứng dụng này thông qua giao thức SNMP sẽ gửi đi các truy vấn đến các thiết bị khác trong mạng. NMS thường cung cấp nhiều loại truy vấn khác nhau, tùy theo phiên bản giao thức hỗ trợ, nhưng chủ yếu vẫn là hai loại chính: Get Request để lấy thông tin thiết bị về hiển thị và Set Request để thay đổi thông tin cấu hình.

Bên cạnh manager, trong mạng còn có các thiết bị khác chịu sự quản lý, các thiết bị này có thể là router, switch, server, cable modem, … còn đang hoạt động. Các thiết bị này sẽ triển khai một ứng dụng gọi là SNMP agent. Ứng dụng này cung cấp một giao diện SNMP để tương tác với trạm quản lý, mà chủ yếu là hồi đáp các request được gửi đến. Khi có request đến, agent sẽ phân tích các thông tin trạng thái cục bộ trên thiết bị và chuyển đổi các thông tin này theo quy định, đưa vào thông điệp hồi đáp tương ứng để trả lời cho máy quản lý. Ngoài ra, nếu được cấu hình thì các agent còn có khả năng thông báo các sự kiện thất thường xảy ra trên thiết bị để manager có thể nắm được.

Đối với kiến trúc tồn tại nhiều phiên bản khác nhau của SNMP thì có thể triển khai thêm các proxy agent. Chức năng chính của proxy agent là chuyển đổi thông điệp từ version này sang version khác rồi chuyển tiếp đến nơi nhận. Ví dụ như trong một môi trường mà cả SNMPv1 và SNMPv2c cùng tồn tại thì các thông điệp truy vấn thông thường như Get, Set Request có thể được chuyển trực tiếp từ manager cho agent. Nhưng với các thông điệp Trap từ agent, sẽ phải qua proxy để chuyển sang dạng ứng với version của manager trước khi gửi đến máy quản lý. Một phương pháp khác cho mô hình triển khai nhiều phiên bản SNMP là manager sẽ hỗ trợ các phiên bản khác nhau. Phương pháp này không cần dùng đến proxy agent nhưng yêu cầu manager phải có thông tin trước về thiết bị. Khi muốn gửi truy vấn cho một thiết bị, manager sẽ kiểm tra thông tin agent trong cơ sở dữ liệu của mình rồi chọn phiên bản hoạt động phù hợp trước khi trao đổi với agent.

### Một số khái niệm cơ bản

Dưới đây là một số khái niệm cơ bản khác đi kèm với SNMP, được dùng khá nhiều trong phần mô tả hoạt động cơ bản bên dưới của giao thức.

Bảng 1.1: Một số khái niệm cơ bản của SNMP

|  |  |
| --- | --- |
| **Khái niệm** | **Ý nghĩa** |
| SNMP application entity | Là các ứng dụng chạy trên các nút mạng và giao tiếp với nhau thông qua giao thức SNMP (Agent và NMS ở trên) |
| Protocol entity | Là các thành phần thực thi giao thức SNMP và cung cấp tính năng cho application entity sử dụng |
| SNMP community | Là một cặp giữa một agent với nhóm các snmp application entities, xác định bởi một community name. |
| SNMP message | Là các thông điệp trao đổi của giao thức. Các thông điệp này thường được xác thực trước khi xử lý. Authentic message của một community là thông điệp được tạo ra từ application entity của community đấy. |
| Authentication scheme | Là một bộ các điều kiện, khoản để xác thực thông điệp |
| Authentication service | Là thành phần thực thi authentication scheme và có chức năng xác thực SNMP message. |
| SNMP MIB view | Là một phần của MIB lưu thông tin đặc trưng cho thiết bị mạng |
| MIB object | Đối tượng lưu giữ một loại thông tin của thiết bị |
| SNMP community profile | Là profile của một community, cho biết quyền truy cập vào thông tin trong MIB view |
| SNMP access policy | Là chính sách về quyền truy cập được tạo ra khi áp dụng community profile xác định lên một community |
| MIB variable | Là instance của MIB object. |

### Đặc điểm hoạt động của giao thức

#### Đặc điểm tổng quan

SNMP hoạt động trên tầng ứng dụng của mô hình TCP/IP. Giao thức này sử dụng UDP của tầng dưới để vận chuyển các thông điệp trao đổi giữa agent và manager. Mỗi thông điệp giữa hai bên sẽ được gói gọn trong một UDP datagram và chuyển đi độc lập với nhau. Đây cũng là một đặc điểm cho thấy sự đơn giản của giao thức này, khi SNMP không quan tâm đến việc yêu cầu của mình gửi đi có đến được Agent không. Ngoài ra, SNMP cơ bản không chấp nhận các thông điệp có kích thước vượt quá 484 bytes từ cả hai phía, nhưng giới hạn này đã được bỏ đi trong một vài phiên bản mở rộng.

Thông thường, SNMP manager sẽ sử dụng cổng bất kỳ để gửi request đến agent với đích là cổng UDP/161. Agent xử lý request trong PDU, rồi gửi hồi đáp về cho manager theo cổng nguồn của gói tin nhận được. Trường hợp có event đặc biệt xảy ra, agent sẽ gửi các thông điệp thông báo (Inform hoặc Trap) về cổng UDP/162 của máy quản lý để báo cho NMS. Cổng 161 và 162 là cổng mặc định nhưng SNMP cũng có thể được cấu hình để sử dụng cổng khác. Trong trường hợp triển khai cùng với TLS và DTLS, manager sẽ được cấu hình nhận trap ở UDP 10162 còn agent sẽ nhận request ở cổng 10161

#### Chi tiết hoạt động

Dưới đây là mô tả chi tiết cơ bản quá trình hoạt động trao đổi thông điệp giữa manager và agent trong quá trình gửi nhận, xử lý request.

Đầu tiên, protocol entity (định nghĩa trong mục 1.1.2) ở phía manager sẽ xây dựng một PDU phù hợp với loại yêu cầu từ phía ứng dụng người dùng. Do unit này được định dạng bởi ASN.1 nên có thể xem như là một ASN.1 object. Sau đấy protocol entity sẽ đưa object này cùng với community name, thông tin địa chỉ nguồn và đích cho authentication service để áp dụng authentication scheme. Service này trả cho protocol entity một ASN.1 object khác mang theo tính xác thực, và protocol entity tiếp tục tiến hành xây dựng một đối tượng thông điệp bằng object này với community name đã có. Đối tượng thông điệp tạo ra được đóng gói và chuẩn hóa với các luật cơ bản của ASN.1, trước khi chuyển tiếp xuống tầng dưới để UDP gửi đi sang agent.

Bên phía agent, sau khi nhận được UDP datagram, protocol entity sẽ tiến hành phân tách sơ lược để khôi phục lại đối tượng thông điệp, trường hợp lỗi thì agent sẽ loại bỏ gói tin và dừng lại. Đối tượng thông điệp được phân tách thành công sẽ tiếp tục trải qua quá trình so sánh phiên bản SNMP, khi không hợp thì gói tin sẽ bị loại bỏ và agent cũng sẽ dừng lại. Tiếp đến, protocol entity sẽ gửi đối tượng thông điệp cùng với community name, địa chỉ nguồn và đích cho một authentication service để áp dụng authentication scheme. Service này hoặc trả về một đối tượng ASN.1 đại diện cho PDU, hoặc thông báo lỗi của quá trình xác thực. Trường hợp có lỗi, protocol entity sẽ loại bỏ gói tin và sinh ra một Authentication failure trap gửi về cho manager theo địa chỉ được cấu hình trước. Trong trường hợp còn lại, protocol entity ở agent tiếp tục tiến hành phân tách để lấy ra PDU truy vấn. Nếu phân tách lỗi thì loại bỏ ASN.1 object và dừng lại, còn không thì xử lý yêu cầu trong PDU, sinh ra bản tin hồi đáp và gửi về phía truy vấn thông qua địa chỉ gửi ở gói tin nhận được.

### Khuôn dạng thông điệp truy vấn, hồi đáp và phân loại

#### Khuôn dạng thông điệp truy vấn, hồi đáp

Thông điệp truy vấn, hồi đáp của SNMP thường có kích thước 484 bytes và gói tin của giao thức có dạng như bên dưới.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IP Header | UDP Header | Version | Community | PDU-type | PDU |

Trong đấy, trường version là phiên bản của giao thức SNMP còn community là community name, dùng như mật khẩu để xác thực giữa manager và agent. Trường tiếp theo PDU-type sẽ chỉ ra loại PDU trong gói tin như là Get, GetNext , Set Request, …. Phần còn lại trong gói tin là PDU mang truy vấn mà manager muốn gửi, thường tuân theo định dạng sau.

PDU ::= IMPLICIT SEQUENCE {

request-id

INTEGER,

error-status

ErrorStatus,

error-index

ErrorIndex,

variable-bindings

VarbindList

}

Do manager thường gửi đi nhiểu truy vấn cùng một lúc đến nhiều thiết bị khác nhau trong mạng nên trong PDU có thêm trường Request ID để phân biệt các truy vấn với nhau. Trường này còn có tác dụng để nhận biết thông điệp trả lời ứng với mỗi truy vấn gửi đi khi agent thường dùng Request Id của truy vấn từ manager để xét cho gói tin hồi đáp. Trong bối cảnh SNMP sử dụng giao thức vận chuyển UDP không tin cậy ở tầng dưới, trường này còn có thể sử dụng để phát hiện các thông điệp bị lặp.

Trường tiếp theo Error Status dùng bởi agent để chỉ các lỗi có thể xảy ra khi xử lý truy vấn từ phía manager. Trường hợp có lỗi thì giá trị của trường này sẽ khác 0, và trường Error-Index sẽ chứa thông tin về MIB object sinh lỗi trong truy vấn. Với các thông điệp truy vấn từ phía manager, Error Status được xét bằng 0 (noError). Một số lỗi cơ bản, có thể xảy trong quá trình xử lý request được liệt kê trong bảng dưới đây.

Bảng 1.2: Một số lỗi cơ bản xảy ra khi xử lý truy vấn trong SNMP

|  |  |
| --- | --- |
| **Error Status** | **Nguyên nhân lỗi** |
| Too big (1) | Kích thông điệp trả lời truy vấn vượt quá giới hạn cho phép |
| No such name (2) | Đối tượng trong truy vấn không tồn tại |
| Bad value (3) | Giá trị của đối tượng trong Set Request hoặc là sai cú pháp hoặc là giá trị không phù hợp |
| Read only (4) | Truy vấn đang cố thay đổi giá trị của một MIB object chỉ có quyền truy cập là read-only |
| Gen err (5) | Một lỗi khác các lỗi trên xảy ra trong quá trình xử lý truy vấn ở agent. |

Khái niêm “variable” là để chỉ thực thể của một loại MIB object. Varbind hay variable binding là một cặp nối giữa định danh của variable và giá trị tương ứng. Tập hợp các cặp này lại trong thông điệp sẽ tạo thành một danh sách các đối tượng được truy vấn. Trong một varbind, thành phần giá trị là khá quan trong nhưng với một số loại thông điệp như các thông điệp truy vấn lấy giá trị, thành phần này thường không được chú ý và xử lý. Tuy nhiên, với các thông điệp đấy, khi giá trị trường này được đưa ra thì vẫn cần đảm bảo đúng cú pháp hoặc quy định về miền giá trị.

#### Phân loại thông điệp

Trải qua các phiên bản khác nhau, một số loại thông điệp mới đã được thêm vào nhưng mọi phiên bản của SNMP đều hỗ trợ các loại thông điệp cơ bản sau

Bảng 1.3: Các loại thông điệp cơ bản của SNMP

|  |  |
| --- | --- |
| **Loại thông điệp** | **Mô tả cơ bản** |
| Get Request | Thông điệp gửi từ manager để truy vấn lấy giá trị các đối tượng trong MIB. Trường hợp không có lỗi khi xử lý truy vấn, mỗi thực thể đối tượng được nhắc đến trong variable-bindings sẽ ứng với một cặp varbind trả về trong thông điệp hồi đáp |
| GetNext Request | Truy vấn từ manager để lấy ra thực thể kế tiếp của các thực thể được nhắc đến trong variable-bindings. Trường hợp không có lỗi xảy, với mỗi một thực thể được nhắc đến ở truy vấn, một cặp varbind của thực thể đối tượng kế tiếp trong MIB được lấy ra và tập hợp vào thông điệp hồi đáp |
| Set Request | Thông điệp để thay đổi giá trị thực thể trong MIB. Khi không có lỗi xử lý, mỗi một giá trị của thực thể trong truy vấn sẽ được xét thay thế giá trị hiện tại trong MIB. Thông điệp trả về cho truy vấn loại này cũng chứa các cặp tên - giá trị mới được xét như để xác nhận kết quả xử lý. |
| Response | Thông điệp hồi đáp từ phía agent cho các truy vấn từ phía manager. |
| Trap | Thông điệp từ phía agent để thông báo cho manager khi có một sự kiện cụ thể xảy ra trên thiết bị. |

Ngoài các loại thông điệp trên, SNMP còn có một số loại thông điệp khác được thêm vào ở các phiên bản sau này như GetBulk Request, Inform Request…. GetBulk (thêm vào ở SNMPv2) là loại dùng để lấy một loạt nhiều giá trị thực thể của đối tượng MIB ra với chỉ một thông điệp. Các thông điệp Get, GetNext cũng cho phép lấy ra nhiều đối tượng bằng cách đưa nhiều MIB object vào variable bindings hoặc gửi đi nhiều thông điệp truy vấn một đối tượng cùng lúc trong khi GetBulk thì chỉ cần một object và một thông điệp. Bằng việc sử dụng thông điệp loại này, manager có thể giảm được kích thước thông điệp truy vấn, tiến kiệm được tài nguyên tính toán trong quá trình tạo thông điệp.

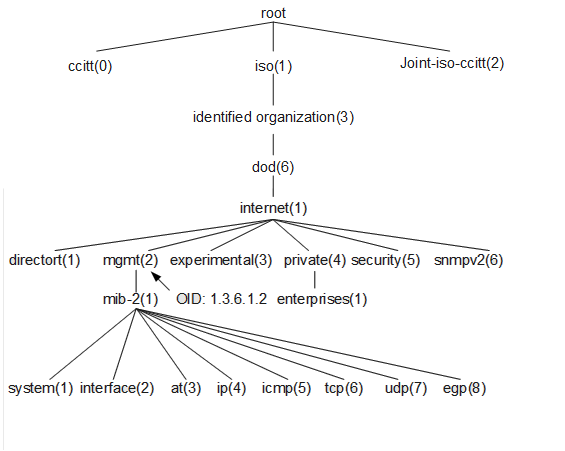
## Management information base

### Giới thiệu chung

MIB (Management information base) là một vùng nhớ lưu trữ thông tin của thiết bị. Thông tin của thiết bị được nhìn nhận dưới dạng một tập hợp các MIB object. Các object có liên hệ với nhau thường được định nghĩa bởi một module. MIB module có loại chuẩn chung và cũng có loại do các tổ chức, công ty sản xuất thiết bị đưa ra, nên trên mỗi thiết bị từ các hãng khác nhau có thể có các MIB view khác nhau.

Kiểu của một MIB object được định dạng theo một mẫu (ASN.1 OBJECT TYPE macro) với 4 trường cơ bản: Syntax - định nghĩa, mô tả cấu trúc dữ liệu (kiểu dữ liệu) của đối tượng; Access (Max-Access) - cho biết quyền truy cập vào đối tượng loại này; Description - mô tả ý nghĩa và vai trò của MIB object.; Status - trạng thái hiện tại của MIB object.

MIB được tổ chức dưới dạng mô phỏng hình cây và để xác định các MIB object trong đó phải dùng một thành phần là OID (object identifier). Thông thường OID được biểu diễn dưới dạng một dãy các số nguyên cách nhau bởi dấu “.”, ví dụ 1.3.6.1.2.1.1.3.0. Dãy này là path từ nút gốc dẫn đến MIB object tương ứng nên với dạng cây khung thì mỗi OID chỉ xác định một object duy nhất. Ngoài ra, OID còn được biểu diễn dưới dạng kết hợp giữa xâu và số, chẳng hạn sysUpTime.0. Cả hai dạng trên đều thể hiện một OID với 2 thành phần. Thành phần đầu tiên đại diện kiểu object và thành phần còn lại là để xác định thực thể của object.



Hình 1.2: Dạng cây mô phỏng của MIB

Khi các MIB object có liên hệ chặt chẽ với nhau thì tập hợp của các object này có thể xem như một bảng MIB. Một vài bảng phổ biến có thể kể đến như ifTable (bảng lưu thông tin liên quan đến interface của thiết bị), ipRouteTable (bảng lưu các thông tin định tuyến), ...

Với mỗi object trong MIB thì quyền truy cập vào được xác định bởi 3 kiểu. Nếu quyền truy cập của đối tượng là “none” thì giá trị của object đấy không thể bị lấy ra hoặc thay đổi được bất kể community profile là thế nào. Nếu đối tượng được định nghĩa với “read - write” hoặc “write - only” nhưng community profile thuộc loại “read–write” thì giá trị object đấy thỏa mãn cho cả Get và Set request. Còn lại MIB object kiểu “read-only” chỉ có thể dùng để lấy ra cho Get response (hồi đáp của Get request) hoặc thông điệp Trap.

### Phạm vi ứng dụng

Một MIB thông thường có kích thước khá lớn với nhiều module định nghĩa khác nhau nhưng đề tài sẽ hướng đến sử dụng 4 module chính cho các chức năng cơ bản của chương trình sản phầm là SNMPv2-MIB, IF-MIB, IP-MIB và HOST-RESOURCE-MIB. Trong đó, các MIB object định nghĩa bởi SNMPv2-MIB sử dụng trong quá trình thực hiện đề tài được liệt kê trong bảng dưới đây.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **OID** | **Phạm vi truy cập** | **Mô tả** |
| sysDescr | 1.3.6.1.2.1.1.1 | Read-only | Mô tả hệ thống của thiết bị |
| sysUpTime | 1.3.6.1.2.1.1.3 | Read-only | Thời gian kể từ khi khởi động |
| sysName | 1.3.6.1.2.1.1.5 | Read-write | Tên của thiết bị, dùng cho quản lý |
| sysLocation | 1.3.6.1.2.1.1.6 | Read-write | Vị trí của thiết bị |

Với IF-MIB, các MIB object được sử dụng sẽ gồm các object từ bảng ifTable lưu một số thông tin của các giao diện mạng trên thiết bị. Các object này được liệt kê trong bảng sau.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **OID** | **Phạm vi truy cập** | **Mô tả** |
| ifIndex | 1.3.6.1.2.1.2.2.1.1 | Read-only | ID của giao diện mạng trên thiết bị |
| ifOperStatus | 1.3.6.1.2.1.2.2.1.8 | Read-only | Trạng thái hoạt động |
| ifDescr | 1.3.6.1.2.1.2.2.1.2 | Read-only | Mô tả giao diện mạng |
| ifPhysAddress | 1.3.6.1.2.1.2.2.1.6 | Read-only | Địa chỉ vật lý (MAC address) |
| ifType | 1.3.6.1.2.1.2.2.1.3 | Read-only | Kiểu của giao diện |
| ifMtu | 1.3.6.1.2.1.2.2.1.4 | Read-only | Kích thước gói tin lớn nhất có thể |
| ifSpeed | 1.3.6.1.2.1.2.2.1.5 | Read-only | Giá trị ước lượng của băng thông |
| ifInUCastPkts | 1.3.6.1.2.1.2.2.1.11 | Read-only | Số gói tin Unicast đi vào |
| ifOutUCastPkts | 1.3.6.1.2.1.2.2.1.17 | Read-only | Số gói tin Unicast truyền ra |
| ifInNUCastPkts | 1.3.6.1.2.1.2.2.1.12 | Read-only | Số gói tin Non-unicast đi vào |
| ifOutNUCastPkts | 1.3.6.1.2.1.2.2.1.18 | Read-only | Số gói tin Non-unicast truyền ra |
| ifInOctets | 1.3.6.1.2.1.2.2.1.10 | Read-only | Số byte dữ liệu đi vào |
| ifOutOctets | 1.3.6.1.2.1.2.2.1.16 | Read-only | Số byte dữ liệu truyền ra |
| ifInDiscards | 1.3.6.1.2.1.2.2.1.13 | Read-only | Số gói tin đi vào bị loại bỏ |
| ifOutDiscards | 1.3.6.1.2.1.2.2.1.19 | Read-only | Số gói tin đi ra bị loại bỏ |

Các MIB object, định nghĩa trong IP-MIB được sử dụng là từ các bảng ipAddrTable, ipNetToMediaTable và được liệt kê trong bảng sau

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **OID** | **Phạm vi truy cập** | **Mô tả** |
| ipAdEntIfIndex | 1.3.6.1.2.1.4.20.1.2 | Read-only | ID của giao diện mạng |
| ipAdEntAddr | 1.3.6.1.2.1.4.20.1.1 | Read-only | Địa chỉ IP |
| ipAdEntNetmask | 1.3.6.1.2.1.4.20.1.3 | Read-only | Subnet mask |
|  |  |  |  |
| ipNetToMediaIfIndex | 1.3.6.1.2.1.4.22.1.1 | Read-only | ID của giao diện mạng |
| ipNetToMediaPhysAddress | 1.3.6.1.2.1.4.22.1.2 | Read-only | Địa chỉ vật lý nút liên kết với giao diện |
| ipNetToMediaNetAddress | 1.3.6.1.2.1.4.22.1.3 | Read-only | Địa chỉ IP của nút liên kết với giao diện |

Ở module còn lại HOST-RESOURCES-MIB, các đối tượng được sử dụng trong quá trình thực hiện đề tài sẽ từ 3 bảng hrProcessorTable, hrDeviceTable và hrStorageTable. Các object này được liệt kê theo bảng như bên dưới.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **OID** | **Phạm vi truy cập** | **Mô tả** |
| hrProcessorFrwID | 1.3.6.1.2.1.25.3.3.1.1 | Read-only | Firmware ID của processor |
| hrProcessorLoad | 1.3.6.1.2.1.25.3.3.1.2 | Read-only | Tải của processor |
|  |  |  |  |
| hrDeviceType | 1.3.6.1.2.1.25.3.2.1.2 | Read-only | Loại của thành phần phần cứng |
| hrDeviceDescr | 1.3.6.1.2.1.25.3.2.1.3 | Read-only | Mô tả của thành phần phần cứng |
| hrDeviceID | 1.3.6.1.2.1.25.3.2.1.4 | Read-only | ID của nhà sản xuất |
|  |  |  |  |
| hrStorageType | 1.3.6.1.2.1.25.2.3.1.2 | Read-only | Loại của vùng lưu trữ |
| hrStorageDescr | 1.3.6.1.2.1.25.2.3.1.3 | Read-only | Mô tả của vùng lưu trữ |
| hrStorageSize | 1.3.6.1.2.1.25.2.3.1.5 | Read-write | Kích thước tổng của vùng lưu trữ |
| hrStorageUsed | 1.3.6.1.2.1.25.2.3.1.6 | Read-only | Kích thước của phần đã sử dụng |

## Trap

### Khuôn dạng thông điệp

Trap là thông điệp từ phía agent để thông báo cho manager khi có một sự kiện cụ thể xảy ra trên thiết bị chịu sự quản lý. Ở phiên bản đầu tiên SNMPv1, thông điệp Trap thường có PDU dạng theo mô tả bên dưới.

PDU ::= IMPLICIT SEQUENCE {

enterprise

OBJECT IDENTIFIER,

agent-addr

NetworkAddress,

generic-trap

INTEGER,

specific-trap

INTEGER,

time-stamp

TimeTicks,

variable-bindings

VarbindList,

}

Trong đấy, trường Enterprise xác định loại đối tượng sinh ra thông điệp và Agent-address là địa chỉ của agent sinh ra Trap. Hai trường kế tiếp dùng để xác định kiểu Trap với giá trị Generic-trap dùng để xác định các loại phổ biến như coldStart(0), warmStart(1), linkDown(3) … Khi trường generic-trap có giá trị là enterpriseSpecific(6) thì nghĩa là Trap này được định nghĩa bởi các công ty, tổ chức khác nhau và không theo chuẩn chung. Lúc này, kiểu của Trap sẽ được xác định cụ thể qua trường specific-trap. Time-stamp là trường chứa thông tin về khoảng thời gian đã trôi qua từ lần cuối cùng reboot của agent. Trường này có thể dùng trong việc xác định thời điểm xảy ra sự kiện trên thiết bị. Ngoài ra, với mỗi loại sự kiện còn có thêm các thông tin mô tả kèm theo được gửi về trong variable-bindings của thông điệp, dùng trong quá trình phân tích, hiển thị.

Phần trên là mô tả thông điệp Trap cho SNMPv1. Ở phiên bản thế hệ sau SNMPv2c, Trap được định nghĩa thành Notification và có một định dạng PDU mới khác biệt hoàn toàn với phiên bản ban đầu.

PDU ::= IMPLICIT SEQUENCE {

sysUpTime

TimeTicks,

snmpTrapOid

OBJECT IDENTIFIER,

variable-bindings

VarbindList

}

Trong các trường của thông điệp, trường sysUpTime mang thông tin thời gian giống với Time-stamp của phiên bản trước. Trường kế tiếp snmpTrapOID đùng dể xác đinh loại Trap. Với các loại phổ biến thông thường, giá trị của các trường này được định nghĩa trước. Còn với Enterprise Trap, giá trị trường này là tập hợp của thành phần xác định loại Enterprise tổng quát, một giá trị “0” liền kề và một giá trị cuối cùng là mã loại cụ thể. Phần cuối của thông điệp là danh sách variable-bindings mang các thông tin kèm theo, mô tả chi tiết hơn về sự kiện xảy ra trên thiết bị.

Ngoài ra, cũng ở phiên bản SNMPv2c, giao thức giới thiệu thêm một loại thông điệp nữa là Inform. Do sự thiếu tin cậy của giao thức tầng vận chuyển UDP, nên thông điệp này được dùng để thông báo sự kiện từ SNMP agent, giống như thông điệp Trap nhưng có thêm cơ chế ACK

### Các loại Trap

Dưới đây là bảng mô tả của các loại Trap cơ bản của SNMP và đặc điểm, sự kiện nguyên nhân tương ứng.

Bảng 1.4: Các loại thông điệp Trap cơ bản

|  |  |  |  |
| --- | --- | --- | --- |
| **Loại Trap** | **Giá trị Generic-Trap** | **Giá trị Snmp Trap OID** | **Đặc điểm** |
| Cold Start | 0 | 1.3.6.1.6.3.1.1.5.1 | Thông điệp này được sinh ra khi agent bị khởi động lại với cấu hình có thể bị thay đổi |
| Warm Start | 1 | 1.3.6.1.6.3.1.1.5.2 | Trap được sinh ra khi agent bị khởi đông lại và cấu hình không thay đổi |
| Link Down | 2 | 1.3.6.1.6.3.1.1.5.3 | Thông điệp này được sinh ra khi có một liên kết ở agent bị lỗi hoặc mất đi. Trong phần nội dung của Trap thường chứa cặp varbind mang thông tin giao diện mạng (ifIndex) bị ảnh hưởng |
| Link Up | 3 | 1.3.6.1.6.3.1.1.5.4 | Thông điêp được sinh ra khi có một liên kết ở agent chuyển từ trạng thái down sang up. Phần nội dung tương đồng với của Trap loại trên khi có thông tin giao diện mạng bị ảnh hưởng |
| Authentication Failure | 4 | 1.3.6.1.6.3.1.1.5.5 | Trap được sinh ra khi xảy ra lỗi trong quá trình xác thực nguồn gửi truy vấn đến agent. Các thiết bị được quản lý nên thực thi cơ chế sinh Trap này để tăng tính bảo mật cho hệ thống |
| EGP Neighbor Loss | 5 | 1.3.6.1.6.3.1.1.5.6 | Được sinh ra khi một nút hàng xóm xác định qua giao thức EGP được xác định là down và mối liên kết giữa hai bên không còn. Thông thường trong thông điệp loại này có chứa cặp varbind mang địa chỉ của EGP neighbor ảnh hưởng |
| Enterprise | 6 | EnterpriseType.0. SpecificCode | Trap được định nghĩa bởi các doanh nghiệp, tổ chức cho từng kiểu sự kiện cụ thể` |

Các Enterprise Trap được định nghĩa ra bởi các công ty, tổ chức nên rất khó để có thể phân loại và xử lý được toàn bộ. Bởi vậy chương trình sản phẩm của đề tài được phát triền mới chỉ hướng đến việc phân loại và xử lý các thông điệp Generic Trap, bao gồm Cold Start, Warm Start, Link Up, Link Down, Authentication Failure và EGP Neighbor Loss. Còn với các thông điệp Enterprise, chương trình sản phẩm sẽ thu nhập, và hiển thị nội dung nhưng không tiến hành xử lý, phân loại cụ thể.

# PHÂN TÍCH VÀ ĐẶC TẢ CHỨC NĂNG

Tương tác với hệ thống của chương trình chủ yếu là hai tác nhân chính, User và Device. User ở đây thường là các quản trị viên về mạng, có kinh nghiệm về SNMP. Ngoài ra, do chương trình hướng đến tính đơn giản trong sử dụng nên người dùng thông thường, có chút hiểu biết về mạng máy tính cũng có thể dùng trong việc quản lý thiết bị của họ. Tác nhân còn lại, Device có thể xem là các thiết bị được quản lý. Các thiết bị này phải đang hoạt động và đã được triển khai giao thức SNMP, phục vụ cho quá trình hoạt động, trao đổi thông điệp với chương trình ứng dụng.

## Các chức năng cơ bản

Thành phần cung cấp chức năng quản trị mạng được chia thành 2 phần là cơ bản và mở rộng nhằm hướng đến hai nhóm người dùng khác nhau. Dưới đây là biểu đồ chức năng cho thành phần dùng bởi nhóm người dùng cơ bản, những người không cần hiểu biết quá sâu về giao thức SNMP và MIB.

A picture containing text, map

Description automatically generated

Hình 2.1: Biểu đồ ca sử dụng chức năng cơ bản

Đặc tả của ca sử dụng “Import device info”

Bảng 2.1: Bảng đặc tả ca sử dụng “ Import device info”

|  |  |
| --- | --- |
| Tên ca sử dụng | Import device info |
| Mục đích ca sử dụng | Nhập thông tin của các thiết bị muốn giám sát vào trong chương trình |
| Điều kiện đầu vào | Chương trình đang hoạt động và người dùng có file csv chứa thông tin thiết bị |
| Điều kiện kích hoạt | Người dùng lựa chọn nhập thiết bị vào chương trình qua file |
| Luồng thực thi | 1. Người dùng vào giao diện có danh sách các thiết bị  2. Người dùng lựa chọn nhập thiết bị vào chương trình thông qua file  3. Một giao diện chọn file xuất hiện và người dùng chọn file muốn đưa vào trong chương trình  4. Chương trình xử lý file, lưu lại thông tin các thiết bị và hiển thị lên trên danh sách trên giao diện. |

Đặc tả của ca sử dụng “Update device basic info”

Bảng 2.2: Bảng đặc tả ca sử dụng “Update device basic info”

|  |  |
| --- | --- |
| Tên ca sử dụng | Update device basic info |
| Mục đích ca sử dụng | Cập nhật thông tin cơ bản của thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động với một số thiết bị đã được nhập vào |
| Điều kiện kích hoạt | Người dùng lựa chọn thiết bị từ danh sách thiết bị đang có để sửa đổi thông tin. |
| Luồng thực thi | 1. Người dùng vào giao diện danh sách các thiết bị.  2. Người dùng lựa chọn một thiết bị từ danh sách.  3. Giao diện hiển thị thông tin thiết bị hiện ra.  4. Người dùng thay đổi các thông tin cơ bản rồi chọn lưu lại.  5. Chương trình lưu trữ các thay đổi của người dùng. Trong trường hợp thiết bị hoạt động thì một số thông tin có thể được đưa vào lưu trong thiết bị. |

Đặc tả của ca sử dụng “Query device info automatically”

Bảng 2.3: Bảng đặc tả ca sử dụng “Query device info automatically”

|  |  |
| --- | --- |
| Tên ca sử dụng | Query device info automatically |
| Mục đích ca sử dụng | Truy vấn và cập nhật thông tin thiết bị một cách tự động cho người dùng |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường với một số thiết bị đã được đưa vào |
| Điều kiện kích hoạt | Người dùng lựa chọn cập nhật tự động qua một số thành phần giao diện |
| Luồng thực thi | 1. Người dùng vào một giao diện quan sát thông tin thu được trên thiết bị.  2. Người dùng thay đổi chu kỳ thời gian cập nhật tự động rồi lưu lại.  3. Chương trình bắt đầu tự động truy vấn đến thiết bị lấy thông tin với chu kỳ mới. Thông điệp hồi đáp trải qua quá trình xử lý sẽ được lưu trữ và hiển thị lên một số giao diện người dùng. |

Đặc tả ca sử dụng “Observe device resources”

Bảng 2.4: Bảng đặc tả ca sử dụng “Observe device resources”

|  |  |
| --- | --- |
| Tên ca sử dụng | Observe device resources |
| Mục đích ca sử dụng | Quan sát các trạng thái các tài nguyên phần cứng của thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường với một số thiết bị đã được đưa vào |
| Điều kiện kích hoạt | Người dùng mở giao diện hiển thị thông tin sử dụng tài nguyên trên thiết bị |
| Luồng thực thi | 1. Người dùng lựa chọn một thiết bị từ danh sách.  2. Người dùng mở giao diện quan sát thông tin tài nguyên được sử dụng của thiết bị được chọn  3. Giao diện xuất hiện với dữ liệu gần nhất thu được.  4. Chương trình tiếp tục cập nhật thông tin thu được từ thiết bị bởi thành phần thực hiện truy vấn tự động lên giao diện. |

Đặc tả ca sử dụng “Observe device network interfaces”

Bảng 2.5: Bảng đặc tả ca sử dụng “Observe device network interfaces”

|  |  |
| --- | --- |
| Tên ca sử dụng | Observe device network interfaces |
| Mục đích ca sử dụng | Quan sát trạng thái, thông tin của các giao diện mạng trên thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường với một vài thiết bị đã được đưa vào |
| Điều kiện kích hoạt | Người dùng mở giao diện hiển thị thông tin của một giao diện mạng trên thiết bị |
| Luồng thực thi | 1. Người dùng lựa chọn một thiết bị từ danh sách thiết bị được nhập vào  2. Danh sách giao diện mạng ứng với thiết bị này xuất hiện.  3. Người dùng chọn một giao diện mạng muốn quan sát.  4. Giao diện chương trình hiển thị các thông tin thu được từ thiết bị về giao diện mạng này xuất hiện.  5. Chương trình tiếp tục cập nhật giao diện hiển thị với dữ liệu thu được từ thành phần thực hiện truy vấn tự động. |

Đặc tả của ca sử dụng “Observe device statistics”

Bảng 2.6: Bảng đặc tả ca sử dụng “Observe device statistics”

|  |  |
| --- | --- |
| Tên ca sử dụng | Observe device statistics |
| Mục đích ca sử dụng | Quan sát các thông tin tổng hợp để đánh giá hoạt động thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường với một vài thiết bị đã được nhập vào |
| Điều kiện kích hoạt | Người dùng kích hoạt giao diện thống kê dữ liệu của thiết bị |
| Luồng thực thi | 1. Người dùng lựa chọn một thiết bị từ danh sách thiết bị được nhâp vào  2. Người dùng mở giao diện hiển thị thống kê cho thiết bị.  3. Hệ thống tính toán giá trị thống kê từ các dữ liệu đã lưu trữ và hiển thị lên giao diện dưới dạng các biểu đồ. |

Đặc tả của ca sử dụng “Observe notifications”

Bảng 2.7: Bảng đặc tả ca sử dụng “Observe notifications”

|  |  |
| --- | --- |
| Tên ca sử dụng | Observe notifications |
| Mục đích ca sử dụng | Quan sát các notification để nắm thông tin về sự kiện đã xảy ra trên thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường và đã nhận, xử lý thông điệp Trap từ thiết bị |
| Điều kiện kích hoạt | Người dùng mở giao diện các thông điệp Notification |
| Luồng thực thi | 1. Người dùng mở giao diện danh sách các Notifications đã được thu và xử lý.  2. Người dùng thiết lập các điều kiện chọn lựa để lọc ra các thông điệp muốn quan sát.  3. Chương trình hiển thị ra các thông điệp tương ứng.  4. Người dùng lựa chọn một thông điệp.  5. Chương trình hiển thị ra nội dung của thông điệp người dùng lựa chọn. |

Đặc tả ca sử dụng dụng “Access device”

Bảng 2.8: Bảng đặc tả ca sử dụng “Access device”

|  |  |
| --- | --- |
| Tên ca sử dụng | Access device |
| Mục đích ca sử dụng | Hỗ trợ người dùng truy cập thiết bị thông qua SSH, phục vụ chủ yếu cho việc cấu hình |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường với một vài thiết bị đã được nhập vào |
| Điều kiện kích hoạt | Người dùng mở giao diện để thiết lập kết nối đến thiết bị |
| Luồng thực thi | 1. Người dùng chọn một thiết bị từ danh sách đã được đưa vào.  2. Người dùng mở giao diện kết nối đến thiết bị.  3. Người dùng điền các thông tin cần cho quá trình thiết lập phiên kết nối đến thiết bị và lựa chọn kết nối.  4. Phiên thiết lập thành công, người dùng có thể nhập lệnh rồi quan sát kết quả trên giao diện |

Đặc tả ca sử dụng “Process responses”

Bảng 2.9: Bảng đặc tả ca sử dụng “Process response”

|  |  |
| --- | --- |
| Tên ca sử dụng | Process responses |
| Mục đích ca sử dụng | Xử lý hồi đáp cho truy vấn từ thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường và thiết bị gửi hồi đáp |
| Điều kiện kích hoạt | Chương trình nhận được thông điệp trả lời cho truy vấn đã gửi |
| Luồng thực thi | 1. Thiết bị gửi một thông điệp hồi đáp cho truy vấn từ máy quản lý  2. Một thành phần được kích hoạt để nhận và xử lý thông điệp.  3. Dữ liệu phân tích, xử lý thành công được lưu lại và hiển thị lên giao diện cho người dùng. |

Đặc tả của ca sử dụng “Process Trap”

Bảng 2.10: Bảng đặc tả ca sử dụng “Process Traps”

|  |  |
| --- | --- |
| Tên ca sử dụng | Process notifications |
| Mục đích ca sử dụng | Xử lý các thông điệp Trap từ thiết bị |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường và thiết bị gửi Trap |
| Điều kiện kích hoạt | Chương trình nhận được thông điệp từ thiết bị |
| Luồng thực thi | 1. Thiết bị gửi thông điệp Trap thông báo sự kiện trên thiết bị.  2. Các thành phần xử lý trong chương trình được kích hoạt và tiến hành xử lý thông điệp nhận được.  3. Thông điệp Trap được phân tích xử lý được lưu trữ dưới dạng Notification trong hệ thống. |

## Các chức năng mở rộng

Bên cạnh thành phần với các chức năng cơ bản, mặc định ở trên, chương trình còn cung cấp một cơ chế sử dụng mở rộng cho những người đã có kinh nghiệm làm việc với SNMP và MIB. Đó là thành phần cung cấp, sử dụng các template. Mỗi template cơ bản là một tập các MIB object được người dùng tự khai báo, đưa vào hệ thống để sử dụng truy vấn thông tin trên thiết bị. Bằng việc sử dụng cơ chế này, người dùng sẽ không bị bó buộc với các MIB object mà hệ thống cơ bản sử dụng, cho phép họ có thể có nhiều tùy chọn hơn trong việc quan sát thông tin của thiết bị.

Với thành phần mô tả như trên, biểu đồ các ca sử dụng cơ bản là như sau.

A picture containing text, map

Description automatically generated

Hình 2.2: Biểu đồ ca sử dụng các chức năng mở rộng

Đặc tả của ca sử dụng “Import template”.

Bảng 2.11: Bảng đặc tả ca sử dụng “Import template”

|  |  |
| --- | --- |
| Tên ca sử dụng | Import template |
| Mục đích ca sử dụng | Thêm các template vào hệ thống để có thể sử dụng |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường và người dùng có file chứa template muốn đưa vào |
| Điều kiện kích hoạt | Người dùng lựa chọn nhập template vào chương trình |
| Luồng thực thi | 1. Người dùng vào giao diện hiển thị danh sách các template và lựa chọn nhập template vào chương trình  2. Giao diện chọn file xuất hiện  3. Người dùng chọn file họ muốn đưa vào chương trình.  4. Chương trình xử lý file thành công, lưu lại template và MIB object được khai báo trong đấy rồi hiển thị template lên giao diện danh sách. |

Đặc tả ca sử dụng “Edit template”

Bảng 2.12: Bảng đặc tả của ca sử dụng “Edit template”

|  |  |
| --- | --- |
| Tên ca sử dụng | Edit template |
| Mục đích ca sử dụng | Thay đổi thông tin cơ bản của template và bật tắt các MIB object được khai báo trong template |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường và đã có template được nhập vào. |
| Điều kiện kích hoạt | Người dùng lựa chọn hiển thị thông tin một template từ danh sách |
| Luồng thực thi | 1. Người dùng lựa chọn một template từ danh sách.  2. Giao diện hiển thị thông tin template xuất hiện.  3. Người dùng thay đổi thông tin template, bật tắt MIB object và chọn lưu lại.  4. Chương trình lưu lại thay đổi của người dùng. |

Đặc tả ca sử dụng “Query with template”

Bảng 2.13: Bảng đặc tả ca sử dụng “Query with template”

|  |  |
| --- | --- |
| Tên ca sử dụng | Query with template |
| Mục đích ca sử dụng | Gửi truy vấn lấy thông tin thiết bị dựa theo các MIB object được khai báo trong template |
| Điều kiện đầu vào | Chương trình đang hoạt động bình thường và đã có template trong hệ thống |
| Điều kiện kích hoạt | Người dùng chọn một thiết bị để truy vấn |
| Luồng thực thi | 1. Người dùng lựa chọn một thiết bị từ danh sách các thiết bị được nhập vào  2. Người dùng mở giao diện chọn template dùng cho truy vấn  3. Giao diện các template cho truy vấn xuất hiện và người dùng chọn một template để sử dụng.  4. Chương trình tiến hành gửi truy vấn đến thiết bị lấy thông tin.  5. Chương trình nhận được hồi đáp, xử lý và hiển thị |

# THIẾT KẾ VÀ XÂY DỰNG CHƯƠNG TRÌNH

## Thiết kế chức năng

### Các chức năng cơ bản

Chức năng cơ bản đầu tiên là cho phép người dùng nhập thông tin các thiết bị vào hệ thống trước khi tiến hành theo dõi. Để thiết bị và chương trình có thể tương tác, người dùng cần cung cấp các thông tin cần thiết như địa chỉ của giao diện SNMP bên cạnh các thông tin cơ bản khác như tên, nhãn của thiết bị, …. Các thông tin này cần được đưa vào với số lượng lớn nên chương trình cung cấp chức năng xử lý nhập vào theo file csv. Chức năng này ứng với ca sử dụng “Import device info” và biểu đồ trình tự như sau.

A screenshot of a social media post

Description automatically generated

Hình 3.1: Biểu đồ trình tự của ca sử dụng “Import device info”

Đặc tả của biểu đồ trình tự trên:

* Trên giao diện danh sách thiết bị, người dùng lựa chọn nhập thiết bị vào để quản lý. Một giao diện chọn file sẽ hiển thị và người dùng chọn file chứa thông tin các thiết bị muốn đưa vào chương trình.
* Thành phần xử lý sự kiện trên giao diện gọi đến thành phần xử lý các tác vụ liên quan đến thiết bị DeviceManagementController. Controller bắt đầu quá trình xử lý việc nhập thông tin thiết bị từ file được chọn bằng cách gọi đến thành phần phân tích thông tin trong file CSVReader.
* Thành phần này tách lấy từng dòng trong file csv ứng với thông tin thiết bị. Với mỗi dòng thông tin thiết bị thu được, Controller gọi đến thành phần xử lý việc lưu trữ DeviceManager để lưu vào cơ sở dữ liệu chương trình
* Sau khi hoàn thành việc lưu trữ các thiết bị, chương trình cập nhật lại danh sách thiết bị ban đầu với các thiết bị mới được đưa vào.

Chức năng kế tiếp hướng đến việc quản lý thiết bị là cho phép người dùng thay đổi các thông tin thiết bị được đưa vào hệ thống, để phù hợp với những thay đổi trong quá trình hoạt động của thiết bị. Ca sử dụng ứng với chức năng này là “Update device basic info” và biểu đồ trình tự của quá trình sử dụng này là như sau.

A close up of a map

Description automatically generated

Hình 3.2: Biểu đồ trình tự của ca sử dụng “Update device basic info”

Đặc tả của biểu đồ trình tự trên:

* Trên giao diện danh sách thiết bị, người dụng lựa chọn một để hiển thị các thông tin cơ bản ứng với thiết bị đấy.
* Người dùng thực hiện các thao tác thay đổi thông tin đang hiện thị rồi chọn lưu lại. Thành phần xử lý sự kiện trên giao diện gọi đến Controller để tiến hành quá trình lưu thông tin.
* Controller gọi đến thành phần xử lý việc lưu trữ DeviceManager để cập nhật thay đổi vào cơ sở dữ liệu.
* Controller tiếp tục gọi đến một thành phần hỗ trợ để đưa thông tin lên lưu trên thiết bị. Thành phần hỗ trợ tạo ra một đối tượng xử lý việc gửi truy vấn SNMPContext dựa trên các thông tin của thiết bị và đối tượng callback để xử lý hồi đáp trả về.
* Context được yêu cầu gửi truy vấn SNMP GetNext đến lấy một số thông tin trên thiết bị và cũng để kiểm tra liên kết đến SNMP Agent được triển khai trên thiết bị. Chương trình nhận được hồi đáp, đối tượng callback được gọi đến để xử lý, lấy ra một số thông tin hiển thị lên cho người dùng.
* Nếu liên kết với SNMP Agent bị khả dụng, Context tiếp tục gửi truy vấn SNMP Set để lưu một số thông tin cơ bản như nhãn thiết bị, vị trí, thông tin người cập nhật lên trên thiết bị nhằm hỗ trợ trong việc xác nhận cũng như backup cho cơ sở dữ liệu của trạm quản lý.

Chức năng tiếp theo cho việc quản lý thiết bị là hỗ trợ người dùng truy vấn thông tin thiết bị một cách tự động. Ca sử dụng tương ứng là “Query device info automatically”. Hệ thống cơ bản cho phép truy vấn tự động và tổng hợp dữ liệu thuộc hai loại khác nhau là thông tin của giao diện mạng thiết bị và thông tin về việc sử dụng các tài nguyên phần cứng như CPU, Memory, … Trong đó, chức năng truy vấn tự động và tổng hợp thông tin giao diện mạng của thiết bị có biểu đồ trình tự sử dụng như sau.

A close up of a map

Description automatically generated

Hình 3.3: Biểu đồ trình tự 1 của ca sử dụng “Query device info automatically”

Đặc tả của biểu đồ trình tự trên:

* Người dùng thiết lập chu kỳ chạy Task cho một đối tượng Timer trong chương trình.
* Mỗi lần chạy Task, Timer lấy ra danh sách ID của một nhóm thiết bị với SNMP Agent đang hoạt động, từ một thành phần lưu trữ dữ liệu ActiveDeviceDataCollector. Danh sách ID này được xây dựng và thường xuyên cập nhật bởi một Timer khác hoạt động trong chương trình.
* Với mỗi ID trong danh sách thu được, Timer gọi đến thành phần xử lý việc lưu trữ dữ liệu DeviceManager để lấy ra thông tin của thiết bị, tạo đối tượng Context để phục vụ gửi đi các SNMP Request, đối tượng InterfaceDataCollector để tổng hợp dữ liệu cũng như callback cho xử lý thông điệp hồi đáp.
* Context gửi truy vấn đến “walk” toàn bộ bảng ifTable trên thiết bị. Cơ chế “Walk” ở đây là sử dụng các thông điệp “SNMP GetBulk” để lấy ra toàn bộ các đối tượng được lưu trong bảng ifTable của thiết bị. Thông điệp trả về được đưa vào Collector tổng hợp lưu trữ theo từng dòng của bảng thông tin lấy được.
* Context gửi truy vấn tiếp đến “walk” bảng ipAddrTable. Dữ liệu thu thập trả về cũng được callback đưa vào InterfaceDataCollector để tổng hợp và ghép cặp theo từng dòng với dữ liệu thu được từ lần trước thông qua giá trị của số hiệu cổng lưu trong đối tượng MIB “ifIndex” và “ipAdEntIfIndex”.
* Context gửi các truy vấn cuối cùng đến “walk” bảng ipNetToMediaTable. Callback đưa dữ liệu trả về vào InterfaceDataCollector để tổng hợp, ghép cặp dựa trên số hiệu cổng lưu ở MIB object “ifIndex” và “ipNetToMediaIfIndex”.
* Đối tượng DataCollector chuyển dữ liệu tổng hợp được cho Controller quản lý tác vụ liên quan đến giao diện mạng thiết bị để tiến hành việc lưu trữ và hiển thị.

Dữ liệu giao diện mạng thu thập được khi lưu trữ chia thành ba loại chính là: loại ít thay đổi như địa chỉ mac, kiểu giao diện; loại dữ liệu hay thay đổi cần lưu trữ trong cơ sở dữ liệu như lượng dữ liệu vào ra và loại thông tin hay thay đổi nhưng không cần lưu trữ trên cơ sở dữ liệu như địa chỉ IP, thông tin các nút liên kết. Các dữ liệu kiểu đầu được lưu lại nhằm xác định các giao diện mạng của thiết bị. Loại thứ hai lưu lại trong cơ sở dữ liệu thông qua thành phần xứ lý việc lưu trữ InterfaceDynamicDataManager để tính toán các giá trị thống kê, phục vụ việc đánh giá hiệu năng hoạt động của thiết bị. Còn loại còn lại, không dùng trong việc tính toán thống kê nên sẽ dược lưu tạm thời lại trong một thành phần ActiveDeviceDataCollector để hiển thị cho người sử dụng có thể quan sát thông tin hiện trạng, kết nối của giao diện mạng được chọn.

Với quá trình truy vấn tự động thông tin về việc sử dụng tài nguyên trên thiết bị, chương trình cũng hoạt động theo một trình tự tương tự như trên với biểu đồ trình tự như sau.

A close up of a map

Description automatically generated

Hình 3.4: Biểu đồ trình tự 2 của ca sử dụng “Query device info automatically”

Điểm khác biệt so với quá trình cập nhật thông tin trạng thái thiết bị là các bảng được truy vấn đến để lấy thông tin là các bảng hrProcessorTable, hrDeviceTable và hrStorageTable. Ngoài ra, lượng dữ liệu thu được sau xử lý sẽ được lưu trữ trong cơ sở dư liệu theo từng loại tài nguyên CPU và Memory, nhằm phục vụ cho quá trình tính toán giá trị thống kê và đánh giá hoạt động của thiết bị.

Chức năng cơ bản tiếp theo trong việc quản lý thiết bị là cho phép người dùng quan sát các thông tin về giao diện mạng thu được. Chức năng này ứng với ca sử dụng “Observe device network interfaces” và biểu đồ trình tự sử dụng như sau.

A screenshot of a social media post

Description automatically generated

Hình 3.5: Biểu đồ trình tự của ca sử dụng “Observe device network interfaces”

Đặc tả biểu đồ trình tự:

* Người dùng chọn một giao diện mạng trong danh sách các giao diện của một thiết bi đang xét, giao diện hiển thị thông tin thu thập được của chương trình sẽ xuất hiện.
* Trong quá trình khởi tạo dữ liệu hiển thị của mình, thành phần giao diện này sẽ gọi đến Controller chịu trách nhiệm để lấy ra dữ liệu mới nhất từ cơ sở dữ liệu và ActiveDeviceDataCollector rồi hiển thị cho người dùng khi thành phần cập nhật tự động chưa có dữ liệu mới.
* Thành phần cập nhật tự động của chương trình có dữ liệu mới sau mỗi lượt thu thập, xử lý sẽ thông qua Controller để lưu trữ và tiến hành cập nhật lại giao diện hiển thị.

Chức năng cơ bản kế tiếp của chương trình là cho phép người dùng quan sát các thông tin về sử dụng tài nguyên phần cứng của thiết bị. Ca sử dụng của chức năng này là “Observe device resources” và được mô tả theo biểu đồ trình tự sau.

A screenshot of a social media post

Description automatically generated

Hình 3.6: Biểu đồ trình tự của ca sử dụng “Observe device resources”

Đặc tả biểu đồ trình tự trên:

* Người dùng mở giao diện hiển thị thông tin sử dụng tài nguyên phần cứng thu thập được cho một thiết bị được chọn trong danh sách các thiết bị.
* Sau khi thành phần truy vấn thông tin tự động trên thiết bị xử lý xong dữ liệu thu thập được sẽ thông qua Controller để liên tục cập nhật lên giao diện hiển thị.

Các thông tin trên thiết bị được thu thập và lưu trữ với mục đích chính là để phục vụ cho việc thống kê, hỗ trợ cho người dùng trong quá trình đánh giá hoạt động của thiết bị. Chức năng thống kê, ứng với ca sử dụng “Observe device statistics” của thiết bị, có biểu đồ trình tự như sau.

A close up of a map

Description automatically generated

Hình 3.7: Biểu đồ trình tự của ca sử dụng “Observe statistics”

Đặc tả của biểu đồ trình tự trên.

* Người dùng mở giao diện hiển thị các giá trị thống kê cho dữ liệu thu thập được của một thiết bị được chọn từ danh sách các thiết bị.
* Thành phần giao diện này trong quá trình khởi tạo dữ liệu hiển thị sẽ gọi đến một Controller chịu trách nhiệm trong chương trình để lấy mô hình đồ thi.
* Thành phần Controller lần lượt yêu cầu các đối tượng xử lý việc lưu trữ thông tin thu thập về CPU, Memory, lưu lượng giao diện mạng để lấy ra dữ liệu đã lưu theo khoảng thời gian lụa chọn bởi người dùng.
* Với dữ liệu nhận được, Controller tiến hành tính toán ra các giá trị thống kê như tải trung bình của CPU, tỉ lệ sử dụng trung bình của các loại Memory và hiệu suất sử dụng băng thông trung bình trên mỗi giao diện mạng ở thiết bị được chọn.
* Sau đấy, Controller xây dựng mô hình biểu đồ ứng với các giá trị tính toán được và trả về giao diện để hiển thị lên cho người dùng.

Một trong các chức năng cơ bản mà quan trọng của chương trình là khả năng phân tích Trap nhận được từ thiết bị và thông báo lên cho người dùng. Chức năng này ứng với ca sử dụng “Process Traps” và là một ca sử dụng mà tác nhân chủ yếu là thiết bị được quản lý. Ca sử dụng này có biểu đồ trình tự như sau.

A close up of a map

Description automatically generated

Hình 3.8: Biểu đồ trình tự của ca sử dụng “Process Traps”

Đặc tả của biểu đồ trình tự trên.

* Khi thiết bị gửi thông điệp Trap về để thông báo cho một sự kiện xảy ra, thành phần Listener trong chương trình sẽ kích hoạt một đối tượng NotificationProcessor được thiết lập từ trước để tiến hành quá trình xử lý.
* Thành phần xử lý Processor gọi đến một đối tượng NotificationParser để tiến hành phân tích nội dung thông điệp nhập được. Parser dựa trên trường SnmpTrapOID để phân loại Trap nhận được và có thể tiến hành thêm một số thao tác xử lý tùy từng trường hợp trước khi trả về kết quả cho đối tượng Processor.
* Với kết quả từ phân tích, Processor đưa dữ liệu nhận được cho một Controller xử lý các tác vụ liên quan đến Notification để bắt đầu việc lưu trữ. Trong quá trình lưu trữ, Controller cũng đưa một phần nội dung thông điệp sau xử lý lên một thành phần giao diện của chương trình để thông báo cho người dùng quan sát.

Các thông điệp Trap sau khi được xử lý sẽ hiển thị lên trong một giao diện dạng danh sách các Notification. Người dùng có thể tìm kiếm và lựa chọn một thông điệp trên danh sách này để quan sát các thông tin chi tiết hơn. Đây là chức năng của ca sử dụng “Observe Notifications” với biểu đồ trình tự như sau.

A screenshot of a map

Description automatically generated

Hình 3.9: Biểu đồ trình tự của ca sử dụng “Observe Notifications”

Đặc tả của biểu đồ trình tự trên.

* Trên giao diện hiển thị danh sách các thông điệp Notification nhận được sau xử lý, khi người dùng thay đổi các điều kiện chọn lọc thì thành phần xử lý sự kiện giao diện sẽ gọi đến một Controller chịu trách nhiệm xử lý để lấy ra các thông điệp tương ứng.
* Controller đưa các điều kiện chọn lọc này cho thành phần xử lý việc lưu trữ của các Notification để truy vấn lấy dữ liệu đã lưu trữ và cập nhật lại cho danh sách hiển thị.
* Người dùng chọn lựa một thông điệp cụ thể từ danh sách thì giao diện hiển thị thông tin chi tiết sẽ xuất hiện. Trong quá trình khởi tạo dữ liệu hiển thị của mình, thành phần giao diện gọi tiếp đến Controller để lấy ra thông tin cho thông điệp được chọn.
* Controller lại thông qua thành phần xử lý việc lưu trữ để lấy dữ liệu của thông điệp được chọn rồi đưa cho giao diện để hiển thị lên cho người dùng quan sát.

Ngoài các chức năng cơ bản bên trên, chương trình cũng cung cấp một chức năng cho phép người dùng truy nhập vào thiết bị đang được quản lý bằng SSH. Chức năng này ứng với biểu đồ trình tự sử dụng như sau.

A close up of a map

Description automatically generated

Hình 3.10: Biểu đồ trình tự của ca sử dụng “Access device”

Đặc tả của biểu đồ trình tự.

* Người dùng chọn liên kết đến một thiết bị trong danh sách các thiết bị được đưa vào. Một giao diện mới sẽ xuất hiện cho phép người dùng có thể nhập các thông tin cơ bản cho việc thiết lập phiên kết nối đến thiết bị này.
* Người dùng điền xong thông tin và lựa chọn kết nối, thành phần xử lý sự kiện ở giao diện gọi đến một đối tượng SSHClient trong chương trình để thiết lập phiên liên kết tới thiết bị.
* Nếu phiên thiết lập thành công, giao diện nhập lệnh xuất hiện và quá trình tương tác người dùng với thiết bị được quản lý bắt đầu. Với mỗi lệnh người dùng đưa vào, SSHClient tạo một channel để gửi đi, nhận kết quả trả về rồi đưa lên giao diện hiển thị của chương trình.

Chức năng hỗ trợ truy cập thiết bị này được đưa thêm vào chương trình, chủ yếu để tiện cho người dùng trong việc cấu hình thiết bị. Mặc dù SNMP có hỗ trợ truy vấn loại Set để thay đổi giá trị các đối tượng lưu trong MIB, qua đó thay đổi trạng thái, hoạt động của thiết bị được quản lý, nhưng cơ chế này khá phức tạp và đòi hỏi người sử dụng phải có hiểu biết sâu về MIB và các MIB object. Chưa kể chương trình phát triển dựa trên SNMPv2c, một giao thức thiếu tính bảo mật, nên cần sử dụng một giao thức khác, có tính bảo mật cao hơn là SSH để có thể chuyển đi các thông tin cấu hình cho thiết bị được quản lý.

### Các chức năng mở rộng

Các chức năng của phần này liên quan đến cơ chế sử dụng template của chương trình. Chương trình định nghĩa ra hai kiểu template cho người dùng sử dụng là singular và tabular. Với kiểu đầu tiên, chương trình sẽ chỉ sử dụng GetNext để lấy ra thông tin của của một thực thể ứng với mỗi MIB object khai báo trong template. Còn kiểu thứ hai, chương trình sẽ sử dụng cơ chế “Walk” trong quá trình truy vấn. Bởi vậy, kiểu thứ hai thường phù hợp hơn cho việc lấy thông tin của các bảng object ở trong MIB.

Chức năng mở rộng đầu tiên là cho phép cập nhập vào chương trình một template được người dùng định nghĩa trong file csv và lưu trữ để sử dụng nhiều lần. Chức năng này ứng với ca sử dụng “Import template” và biểu đồ trình tự như sau.

A screenshot of a social media post

Description automatically generated

Hình 3.11: Biểu đồ trình tự của ca sử dụng “Import template”

Đặc tả của biểu đồ trình tự.

* Trên giao diện danh sách các template, người dùng lựa chọn nhập thêm template vào chương trình. Một giao diện chọn file xuất hiện và nười chọn file chứa template muốn đưa vào chương trình.
* Thành phần xử lý kiện trên giao diện gọi đến một Controller chịu trách nhiệm và bắt đầu quá trình xử lý file.
* Controller gọi đến một thành phần xử lý việc phân tách lấy dữ liệu trong file, CSVReader để đọc ra các thông tin cơ bản của template cùng với các MIB object người dùng khai báo, rồi lưu lại trong cơ sở dữ liệu dưới dạng template và các template item qua thành phần quản lý việc lưu trữ TemplateManager.
* Sau khi quá trình lưu trữ kết thúc, chương trình cập nhật lại danh sách đang hiển thị với template mới đã được thêm vào.

Chức năng tiếp theo ở phần mở rộng là cho phép người dùng thay đổi template mà họ đã đưa vào hệ thống. Chức năng này chủ yếu là để người dùng có thể bật tắt các item ứng với các MIB object mà họ đã khai báo trong template và thay đổi các thông tin cơ bản để người sử dụng không cần phải tạo lại template mới. Chức năng này ứng với ca sử dụng “Edit template” và có biểu đồ trình tự sử dụng sau.

A screenshot of a social media post

Description automatically generated

Hình 3.12: Biểu đồ trình tự của ca sử dụng “Edit template”

Đặc tả của biểu đồ trình tự.

* Trên danh sách các template được đưa vào, người dùng chọn hiển thị thông tin của một template. Môt giao diện mới hiển thị thông tin của template với các item được người dùng khai bao xuất hiện.
* Người dùng chỉnh sửa thông tin của các template, bật, tắt các item rồi chọn lưu lại.
* Thành phần xử lý sự kiện trên giao diện sẽ gọi đến Controller chịu trách nhiệm và bắt đầu quá trình cập nhật lại thông tin của template với thành phần xử lý việc lưu trữ TemplateManager.

Chức năng tiếp theo, quan trọng nhất của cơ chế dùng template của chương trình là cho phép người dùng sử dụng template để gửi truy vấn đến thiết bị với các MIB object ở template item mà họ đã khai báo. Chức năng này ứng với ca sử dụng “Query with template” và biểu đồ trình tự sau.

A close up of a map

Description automatically generated

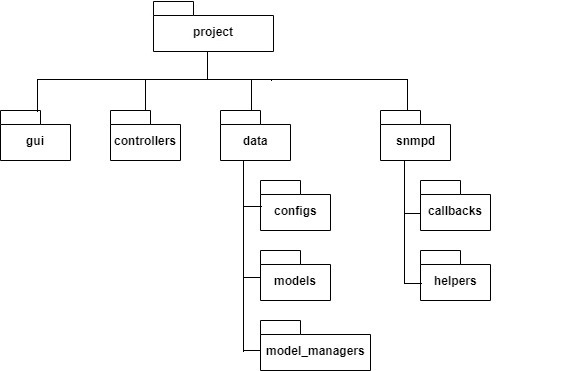
Hình 3.13: Biểu đồ của ca sử dụng “Query with template”

Đặc tả của biểu đồ trình tự

* Người dùng bắt đầu quá trình truy vấn với template cho một thiết bị đã chọn từ danh sách. Một giao diện hiển thị danh sách các template xuất hiện và người dùng lựa chọn template muốn sử dụng.
* Sau khi template được chọn, một giao diện mới để hiển thị kết quả xuất hiện. Thành phần giao diện mới trong quá trình khởi tạo thông tin hiển thị gọi đến một Controller xử lý của chương trình và bắt đầu quá trình gửi đi truy vấn.
* Controller gọi đến các thành phần xử lý việc lưu trữ dư liệu của thiết bị và template để lấy ra thông tin thiết bị, template được chọn. Với dữ liệu nhận được, Controller tạo ra một đối TemplateQuery để lưu tạm thời nhằm dùng trong quá trình phân tích và hiển thị hồi đáp trả về rồi gọi đến một đối tượng hỗ trợ quá trình gửi truy vấn QueryHelper.
* Đối tượng hỗ trợ truy vấn tạo đối tượng Context phục vụ việc gửi truy vấn với các thông tin thiết bị lưu trong TemplateQuery và callback để xử lý hồi đáp trả về. Sau đấy, đối tượng hỗ trợ gọi đến Context để gửi đi truy vấn lấy thông tin các MIB object ứng với những item được bật trong template được chọn.
* Khi có thông điệp trả về, đối tượng callback được kích hoạt và bắt đầu quá trình xử lý, lấy ra các giá trị trong thông điệp hồi đáp ứng với những đối tượng MIB dùng truy vấn rồi lưu vào TemplateQuery, xong đưa cho Controller để tiến hành việc cập nhật, hiển thị lên trên giao diện kết quả của chương trình.

## Thiết kế cấu trúc chương trình

Chương trình được thiết kế thành 4 thành phần cơ bản ứng với 4 package khác nhau, được mô tả theo sơ đồ bên dưới



Hình 3.14: Các package trong thiết kế của chương trình

Trong đó, package gui bao gồm các class giao diện của chương trình còn package controllers chứa các class xử lý nhiều tác vụ khác nhau, nhưng đều liên quan đến một loại đối tượng trong hệ thống. Đối tượng của các class này cũng chính là các Controller được mô tả trong biểu đồ trình tự ở phần trên. Tiếp theo, data, package chứa các lớp liên quan đến xử lý dữ liệu của chương trình. Package này được chia thành 3 package nhỏ hơn. Package đầu, configs chứa các file cấu hình cho Hibernate, framework dùng trong tương tác với cơ sở dữ liệu mà hệ thống sử dụng. Package kế, models chứa các class thực thể. Các class này được tạo ra để hệ thống sử dụng cơ chế ORM (Object Relational Mapping) của Hibernate. Package còn lại, model\_managers, bao gồm các class xử lý các tác vụ liên quan đến việc lưu trữ, sửa xóa các bản ghi ứng với các class thực thể ở models. Package kế tiếp, snmpd là một package bao gói nhiều class sử dụng trong quá trình gửi, xử lý truy vấn SNMP của chương trình. Package này lại được chia thành 2 package khác nhau để chứa các class callback và class hỗ trợ cho quá trình truy vấn lấy thông tin trên thiết bị được quản lý.

### Thiết kế các class của package gui

Các class của package này là class giao diện của chương trình và được chia thành các cấp theo biểu đồ lớp sau.

A close up of text on a black background

Description automatically generated

Hình 3.15: Biểu đồ lớp 1 của package gui

Điểm đầu tiên có thể nhận thấy là các class giao diện đều có chứa các thành phần giao diện cơ bản cung cấp bởi thư viện Swing của JDK. Bởi vậy, các class này đều có phương thức khởi tạo các thành phần giao diện cơ bản, khởi tạo đối tượng xử lý sự kiện trên giao diện, và các getter, setter. Các class giao diện mà có thành phần giao diện con là đối tượng của class định nghĩa bởi chương trình thì còn có thêm các phương thức để trình diễn cũng như che đi giao diện thành phần. Ngoài ra, với các giao diện hiển thị thông tin, các class tương ứng sẽ có thêm có thêm phương thức initViewData, một phương thức dùng chủ yếu để truy vấn đến các thành phần dữ liệu và lấy ra thông tin được lưu trữ trong chương trình về hiển thị.

ApplicationWindow, class giao diện cấp trên cùng, một singleton đại diện cho cửa sổ chính của chương trình. Bên cạnh các thành phần giao diện cơ bản, class này có hai thành phần giao diện con là đối tượng của class PanelInitial và PanelMain. Các phương thức của class này phần lớn là các phương thức cơ bản chung cho class giao diện của chương trình.

PanelInitial là class ứng với panel xuất hiện đầu tiên khi chương trình khởi động, dùng cho quá trình đăng nhập và đăng ký tài khoản người sử dụng. Bên cạnh các phương thức và thuôc tính căn bản chung, class này còn có phương thức và thuôc tính khác mô tả theo bảng sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Ý nghĩa** |
| refreshPanel() | Phương thức để làm mới lại panel đang được hiển thị |
| getDataForRegistration() | Phương thức lấy thông tin người dùng nhập vào form đăng ký trong quá trình tạo tài khoảng người dùng |

Class kế tiếp PanelMain là class ứng với panel chính của chương trình. Class này có các phương thức, thuôc tính quan trọng sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Ý nghĩa** |
| panelNotificationInfo | Panel con hiển thị thông tin chi tiết của một thông điệp Notification |
| panelUserProfile | Panel con hiển thị thông tin người dùng |
| panelImportedDevices | Panel con hiển thị danh sách các thiết bị được quản lý |
| panelImportedTemplates | Panel con hiển thị danh sách các template được đưa vào |
| panelNotificationList | Panel con hiển thị danh sách liệt kê các Notification đã nhận được |
| panelNotificationBoard | Bảng thông báo Notification vừa nhận được, hoạt đông tương tự một popup |
| listNotifications | Tập thành phần giao diện đại diện các thông điệp Notification trên bảng thông báo. |
| showNotification() | Phương thức dùng hiển thị notification vừa nhận được lên bảng thông báo |

Trong các class giao diện con của PanelMain, PanelUserProfile là class của hiển thị thông tin tài khoản người sử dụng. Bên cạnh các phương thức cơ bản như class của mọi thành phần giao diện hiển thị thông tin khác, class này có thêm phương thức để lấy ra các thông tin người dùng, phục vụ trong quá trình cập nhật thông tin tài khoản.

Class giao diện con kế tiếp, PanelImportedTemplates là class ứng với giao diện chứa danh sách các template trong hệ thống. Class này có các phương thức và thuộc tính ngoài căn bản khác như sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Ý nghĩa** |
| panelTemplateInfo | Panel con hiển thị thông tin, item của của một template |
| panelItem | Panel hiển thị thông tin chi tiết của một template item |
| labelTemplates | Tập thành phần giao diện đại diện cho các template trên danh sách. |
| isSingular | Thuôc tính xác định loại của các template đang được hiển thị trên danh sách. |
| initViewData() | Phương thức lấy ra danh sách tên các template đã nhập vào để hiển thị lên. |
| updateTemplateList() | Phương thức dùng để cập nhât danh sách các template hiển thị |

Class giao diện PanelImportedDevices là class ứng với panel có danh sách các thiết bị đã được đưa vào quản lý. Class này có các phương thức và thuôc tính quan trọng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Ý nghĩa** |
| panelDeviceInfo | Panel con hiển thị thông tin cơ bản của một thiết bị |
| panelDeviceResources | Panel con hiển thị tình trạng sử dụng một số loại tài nguyên phần cứng |
| panelDeviceStatistics | Panel con hiển thị biểu đồ thống kê hiệu suất sử dụng của một số loại tài nguyên. |
| panelMonitoringDevices | Panel hiển thị kết quả truy vấn thông tin với template |
| labelDevices | Tập thành phần giao diện đại diện cho các thiết bị trên danh sách hiển thi |
| labelInterfaces | Tập thành phần giao diện hiển thị các giao diện mạng của một thiết bị được chọn |
| initViewData() | Phương thức lấy ra danh sách nhãn các thiết bị đã được nhập vào để hiển thị lên. |
| updateDeviceList() | Phương thức cập nhật danh sách các thiết bị đang được hiển thị |
| updateLabelInterfaces() | Phương thức cập nhật danh sách các giao diện mạng đang được hiển thị cho một thiết bị được chọn |
| updateLabelDeviceState() | Phương thức cập nhật trạng thái cho thiết bị đang hiển thị trong danh sách |
| updateLabelInterfaceState() | Phương thức cập nhật trạng thái cho một giao diện mạng đang hiển thị của một thiết bị được lựa chọn |

Class tiếp theo, PanelNotificationList là class ứng với giao diện có danh sách tổng hợp các thông điệp Notification. Class này có một số thuộc tính và phương thức sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Ý nghĩa** |
| notificationIds | Danh sách ID lưu trữ của các thông điệp Notification |
| initViewData() | Phương thức lấy ra danh sách các notification đã được lưu lên hiển thị. |
| updateNotificationList() | Phương thức để cập nhật danh sách các thông điệp đang hiển thị |
| getSelectedNotificationTypes() | Phương thức lấy ra loại thông điệp mà người dùng lựa chọn để hiển thị |

Class còn lại PanelNotificationInfo là class ứng với giao diện hiển thị thông tin chi tiết của một thông điệp Notification. Do giao diện này không cung cấp nhiều chức năng tương tác cho người dùng nên class này không có phương thức gì khác ngoài các phương thức cơ bản của một class hiển thị thông tin với phương thức initViewData() dùng để lấy thông tin chi tiết của Notificiaton đã lưu. Mặc dù giao diện của class PanelNotificationInfo hiển thị thông tin chi tiết của một thông điệp Notification nhưng đối tượng class này lai được để là thuộc tính của class PanelMain với mục đích để người dùng có thể hiển thị chi tiết hơn thông điệp Notification được thu gọn, đưa vào bảng thông báo dạng popup nhắc đến trong phần mô tả class ứng với panel chính của chương trình.

Class PanelImportedDevices, panel chứa danh sách thiết bị có khá nhiều thành phần thuôc tính ứng với các giao diện con quan trọng, cung cấp chức năng hỗ trợ quản trị thiết bị cho chương trình. Các class ứng với các giao diện con này được mô tả theo biểu đồ thiết kế bên dưới đây.

A close up of a map

Description automatically generated

Hình 3.16: Biểu đồ lớp 2 của package gui

Trong đấy, class PanelDeviceInfo là class ứng với giao diện hiển thị thông tin cơ bản của thiết bị, có các phương thức, thuôc tính quan trọng sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Ý nghĩa** |
| deviceId | ID lưu trữ trong cơ sở dữ liệu của thiết bị được chọn |
| initViewData() | Phương thức lấy và hiển thị thông tin cơ bản đã lưu của thiết bị được chọn. |
| getDeviceDataForUpdate() | Phương thức lấy ra các thông tin để cập nhật lại cơ sở dữ liệu khi người dùng chỉnh sửa |
| updateDeviceDescription() | Phương thức để hiển thị lên thông tin mô tả hệ thống thu được từ thiết bị. |

Class kế tiếp, PanelDeviceResources, một class ứng với thành phần giao diện dùng trong quan sát các thông tin sử dụng tài nguyên trên thiết bị. Class này ngoài các phương thức, thuôc tính cơ bản thì còn có thêm một phương thức, thuộc tính sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Ý nghĩa** |
| deviceId | ID lưu trữ trong cơ sở dữ liệu của thiết bị được chọn |
| initViewData() | Phương thức khởi tạo dữ liệu cho giao diện hiện thị. |
| updateView() | Phương thức cập nhật lại thông tin tài nguyên hiển thị |

Class DialogChoosingTemplates, class này ứng với một hôp thoại hiển thị danh sách các template đã có, bao gồm các thuộc tính và phương thức dùng trong quá trình người dùng tìm kiếm và lựa chọn template cho truy vấn bên cạnh căn bản chung. Tiếp theo, class PanelMonitoringDevices, ứng với giao diện hiển thị kết quả truy vấn qua template, có các phương thức và thuộc tính quan trọng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Ý nghĩa** |
| deviceId | ID của thiết bị được lựa chọn |
| templateId | ID của template được lụa chọn cho truy vấn |
| initViewData() | Phương thức bắt đầu quá trình truy vấn đến thiết bị lấy thông tin về hiển thị. |
| updateViewStage1() | Cập nhật tên hiển thị của các MIB object được truy vấn lên trên bảng kết quả |
| updateViewStage2() | Cập nhật giá trì đối tượng lấy được lên trên bảng kết quả |

Class PanelDeviceStatistics, class giao diện hiển thị thông tin thống kê của thiết bị, gồm các thuộc tính, phương thức quan trọng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| deviceId | ID của thiết bị được chọn |
| initViewData() | Phương thức khởi tạo dữ liệu hiển thị cho một số thành phần giao diện |
| displayAllCharts() | Phương thức gọi đến Controller để lấy các mô hình biểu đồ về |
| displayChart() | Phương thức biểu diễn từng mô hình biểu đồ lên giao diện hiển thị |
| switchChartViewerVisibility() | Phương thức bật, tắt một giao diện dùng cho phóng to biểu đồ được chọn |

Class tiếp theo, PanelInterfaceInfo, là class ứng với panel hiển thị thông thu thập được của một giao diện mạng trên thiết bị được chọn. Thuôc tính, phương thức chủ yếu gồm.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| deviceId | ID của thiết bị được chọn |
| interfaceListID | ID của giao diện mạng này trong danh sách các giao diện mạng mà thiêt bị có |
| initViewData() | Phương thức lấy dữ liệu giao diện mạng mới nhất thu thập được về hiển thị |
| updateView() | Cập nhật dữ liệu thu thập được tự động lên giao diện |
| displayConnectedNodeInformation() | Thay đổi thông tin đang hiển thi ở phần nút liên kết theo lựa chọn người dùng, |

PanelSSHClient là class giao diện cung cấp chức năng thiết lập và sử dụng SSH để truy cập vào thiết bị, Class này ngoài các phương thức cơ bản thì có phương thức initViewData() để lấy ra một số thông tin đã lưu của thiết bị lên hiển thị, hỗ trợ người dùng trong quá trình nhập thông tin để thiết lập phiên làm việc.

Các class còn lại trong các package gui là class của panel hiển thị thông tin của một template và các MIB object được khai báo trong template đấy. Biểu đồ thiết kế của các class này cơ bản là như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.17: Biểu đồ lớp 3 của package gui

Class PanelTemplateInfo là class ứng vơi giao diện hiển thị thông tin của một template. Class này có các phương thức, thuôc tính cơ bản sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Mô tả** |
| templateId | ID trong lưu trữ của template được chọn |
| initViewData() | Phương thức lấy thông tin của template được chọn trong lưu trữ ra |
| updateView() | Phương thức để hiển thị thông tin template lấy được |
| getDataForUpdate() | Phương thức lấy dữ liệu người dùng thay đổi để đưa vào lưu trữ |

Class còn lại, PanelItemInfo là class giao diện hiển thị thông tin của template item được chọn. Giao diện của class này dùng trong hiển thị, thay đổi thông tin là chủ yếu nên có các phương thức, thuộc tính tương tự như của class PanelTemplateInfo, được mô tả ở bảng trên.

### Thiết kế các class trong package controllers

Package controllers bao gồm các class cung cấp đối tượng Controller xử lý tác vụ liên quan đến một loại thực thể trong hệ thống. Biểu đồ thiết kế của các class này cơ bản như hình bên dưới.

A close up of a map

Description automatically generated

Hình 3.18: Biểu đồ lớp của package controllers

Các class trong package đang xét đều được kế thừa từ một class Controller gốc. Thuộc tính và phương thức của class này dùng cho việc hiển thị thông báo kết quả việc xử lý tác vụ được yêu cầu.

Class con đầu tiên, UserManagementController là class xử lý các tác vụ liên quan đến người dùng, có phần lớn các phương thức được gọi trong các quá trình người dùng tạo tài khoản, đăng nhập, đăng xuất khỏi hệ thống và thay đổi thông tin cá nhân.

Tiếp đến là class DeviceManagementController, class xử lý tác vụ liên quan đến thiết bị được quản lý. Các phương thức của class này được mô tả trong bảng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| processImportingDevicesFromFile() | Phương thức thực hiện quá trình xử lý nhập thông tin thiết bị từ file. |
| processGettingImportedDevices() | Phương thức lấy ra nhãn các thiết bị được nhập vào để hiển thị lên danh sách |
| processGettingDeviceInfo() | Phương thức lấy ra thông tin cơ bản của thiết bị đã được đưa vào hệ thống |
| processSavingDeviceInfo() | Phương thức xử lý việc lưu lại thông tin đã chỉnh của thiết bị. |
| processPushingDeviceInfo() | Phương thức tiến hành quá trình đưa một số thông tin cơ bản lên thiết bị. |
| processSendingQueryBasedOnTemplate() | Phương thức tiến hành quá trình gửi truy vấn lấy thông tin theo template |
| processCompletedTemplateQuery() | Phương thức xử lý tiếp dữ liệu trong đối tượng TemplateQuery sau khi có hồi đáp cho truy vấn với template. |

InterfaceManagementController, class với các phương thức xử lý các công việc liên quan đến giao diện mạng được mô tả trong theo trong bảng sau.

|  |  |
| --- | --- |
| **Phương thức / Thuộc tính** | **Mô tả** |
| processGettingInterfaceNames() | Phương thức lấy ra danh sách tên các interface để hiển thị lên giao diện cho môt thiết bị được chọn |
| processChangingInterfaceCheckingPeriod() | Phương thức thực hiện thay đổi chu kỳ của thành phần cập nhật tự động |
| processGettingInterfacesOfActiveDevices() | Phương thức khởi động thành phần lấy thông tin giao diện mạng tự động |
| processGettingInterfaceFromDatabase() | Phương thức lấy ra thông tin của giao diện mạng được chọn mới nhất |
| processCollectedData() | Phương thức xử lý thông tin thu thập được bởi hệ thống tự động |

Tương tự, class DeviceResourceManagementController, class xử lý các tác vụ liên quan đến tài nguyên sử dụng của thiết bị cũng có những phương thức như sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| processGettingResourcesOfActiveDevices() | Phương thức khởi động thành phần lấy thông tin tài nguyên tự động |
| processChangingResourceCheckingPeriod() | Phương thức thực hiện thay đổi chu kỳ cập nhật tự động |
| processCollectedResourceData() | Phương thức xử lý thông tin thu thập bởi hệ thống tự động |

Class NotificationManagementController là Controller class cung cấp các phương thức thực hiện công việc liên quan đến Notification trong chương trình, class này có các phương thức sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| processPushingDeviceNotification() | Phương thức đưa Notification sau xử lý vào lưu trữ và hiển thị |
| processGettingNotifications() | Phương thức lấy ra danh sách các Notification đã lưu trữ lên cho giao diện |
| processGettingNotificationInfo() | Phương thức lấy ra thông tin của Notification được chọn cho giao diện |

Class kế tiếp TemplateManagementController là class Controller sử dụng trong tương tác của người dùng với phần cung cấp cơ chế template của chương trình. Class này có các phương thức xử lý sau.

|  |  |
| --- | --- |
| **Thuộc tính / Phương thức** | **Mô tả** |
| processImportingTemplateFromFile() | Phương thức xử lý việc nhập template từ file |
| processGettingImportedTemplates() | Phương thức lấy ra danh sách các template cho hiển thị |
| processSavingTemplateInfo() | Phương thức tiến hành lưu lại các thay đổi của người dùng |
| processGettingTemplateInfo() | Phương thức lấy ra thông tin đã lưu của template được chọn cho hiển thị |

Còn lại, ChartManagementController, là một class xử lý việc tính toán giá trị thống kê và xây dựng mô hình biểu đồ cho giao diện hiển thị. Các phương thức xử lý của class này là như sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| processGettingChart() | Phương thức bắt đầu quát trình tính toán, xây dựng mô hình biểu đồ cho giao diện |
| buildChartForDay() | Phương thức xây dựng mô hình biểu đồ cho dữ liệu một ngày |
| buildChartForDays() | Phương thức xây dựng mô hình biểu đồ cho dữ liệu nhiều ngày |
| getDataForChart() | Phương thức lấy ra dữ liệu cho mô hình biểu đồ |
| getAverageCPULoadForTime() | Phương thức tính tải CPU trung bình với dữ liệu lưu trữ ở khoảng thời gian |
| getAverageMemoryUsageForTime() | Phương thức tính lượng bộ nhớ sử dụng trung bình ở khoảng thời gian |
| getAverageBandwidthUsageForTime() | Phương thức tính lượng băng thông sử dụng trung bình ở khoảng thời gian |

### Thiết kế các class trong package data

Package này gồm ba thành phần chính là các class thực hiện các tác vụ cập nhật, lưu trữ thông tin vào cơ sở dữ liệu, các class thực hiện việc lưu trữ một số loại thông tin thiết bị không lưu trong cơ sở dữ liệu và các class thực thể của hệ thống. Phần đầu tiên, các class thực hiện các tác vụ cập nhật, lưu trữ thông tin thực thể, có biểu đồ thiết kế như sau.

A screenshot of text

Description automatically generated

Hình 3.19: Biểu đồ lớp 1 của package data

Class DataManager là một singleton, chứa toàn bộ đối tượng của các class xử lý việc lưu trữ dữ liệu khác trong package. Class này được có các thuôc tính, phương thức mô tả theo bảng dưới đây.

|  |  |
| --- | --- |
| Thuôc tính / Phương thức | Mô tả |
| sessionFactory | Thuộc tính đối tượng dùng trong tạo phiên trao đổi với cơ sở dữ liệu. Đối tượng này được chia sẻ tham chiếu cho các class thành phần do chi phí dùng để tạo ra đối tượng này. |
| userManager | Thuôc tính đối tượng của class UserManager |
| settingManager | Thuôc tính đối tượng của class SettingManager |
| deviceManager | Thuôc tính đối tượng của class DeviceManager |
| interfaceDynamicManager | Thuôc tính đối tượng của class DeviceInterfaceDynamicDataManager |
| templateManager | Thuộc tính đối tượng của class TemplateManager |
| templateItemManager | Thuôc tính đối tượng của class TemplateItemManager |
| notificationManager | Thuôc tính đối tượng của class NotificationManager |
| deviceCPUManager | Thuôc tính đối tượng của class DeviceCPUStateManager |
| deviceMemoryManager | Thuôc tính đối tượng của class DeviceMemoryStateManager |
| activeAccountID | ID tài khoảng người dùng đang login vào chương trình |
| close() | Đóng lại sessionFactory và các phiên trao đổi dở với cơ sở dữ liệu. |

Các class xử lý công việc lưu trữ dữ liệu thực thể trong package này đều có cùng một loại thuộc tính để tham chiếu đến sử dụng đối tượng được chia sẻ bởi DataManager. Các phương thức của các class này cũng đơn giản, chủ yếu là để lưu, cập nhật và lấy ra thông tin của các loại đối tượng thực thể trong hệ thống, phục vụ cho Controller trong quá trình thực hiện tác vụ cần thiết. Bảng mô tả chung vai trò của các class xử lý công việc lưu trữ dữ liệu thực thể này như sau.

|  |  |
| --- | --- |
| Class | Vai trò tổng quan |
| UserManager | Xử lý việc lưu trữ thông tin người dùng |
| SettingManager | Xử lý việc lưu trữ thông tin cài đặt của người dùng |
| DeviceManager | Xử lý việc lưu trữ thông tin của thiết bị |
| InterfaceDynamicDataManager | Xử lý việc lưu trữ thông tin thiết bị dùng cho tính toán giá trị thống kê |
| NotificationManager | Xử lý việc lưu trữ thông tin thông điệp Notification sau xử lý |
| DeviceCPUStateManager | Xử lý việc lưu trữ thông tin trạng thái tài nguyên CPU |
| DeviceMemoryStateManager | Xử lý việc lưu trữ thông tin trạng thái tài nguyên bộ nhớ |
| TemplateManager | Xử lý việc lưu trữ thông tin template |
| TemplateItemManager | Xử lý việc lưu trữ thông tin template item |

Phần thứ hai của package này là các class dùng lưu trữ thông tin mà không cần lưu trong cơ sở dữ liệu. Các thông tin này chủ yếu là của giao diện mạng trên thiết bị mà không dùng cho chức năng tính toán, thống kê như địa chỉ ip, mac, lượng gói tin vào ra, thông tin của các node liên kết đến. Các class xử lý việc lưu trữ các thông tin này có biểu đồ thiết kế như sau.

A picture containing screenshot

Description automatically generated

Hình 3.20: Biểu đồ lớp 2 của package data

Trong đó, ActiveDeviceDataCollector là một singleton, lưu giữ thông tin của các thiết bị có SNMP Agent đang hoạt động. Class này có các phương thức, thuôc tính trong bảng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| managerDevice | Đối tượng chứa thông tin giao diện mạng của trạm quản lý |
| importedDevices | Danh sách lưu trữ thông tin các thiết bị với SNMP Agent đang hoạt động |
| initManagerData() | Phương thức tìm và lưu lại địa chỉ các giao diện mạng của máy trạm quản lý |
| addImportedDevice() | Phương thức thêm một thiết bị có SNMP Agent đang hoạt động vào danh sách |
| removeImportedDevice() | Phương thức xóa đi một thiết bị có SNMP Agent ngừng hoạt động khỏi danh sách lưu trữ |
| updateInterfaceData() | Phương thức cập nhật dữ liệu giao diện mạng thu được bởi hệ thống cập nhật tự động thiết bị trong danh sách |
| getImportedDeviceIds() | Phương thức cung cấp cho thành phần cập nhật tự động danh sách ID của thiết bị có Agent hoạt động |

Class ActiveDeviceData dùng lưu thông tin của thiết bị với SNMP Agent đang hoạt động. Class này có các thuôc tính, phương thức được mô tả trong bảng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| id | ID của thiết bị có SNMP Agent đang hoạt động |
| interfaces | Danh sách lưu dữ liệu các giao diện mạng thu được |
| updateInterface() | Phương thức cập nhật dữ liệu giao diện mạng thu được |
| removeConnectedNodes() | Phương thức loại đi dữ liệu của nút liên kết đến giao diện |

Class InterfaceData lưu các thông tin của giao diện mạng không phục vụ cho việc tính toán giá trị thông kê, thu thập được bởi thành phần cập nhật tự động của chương trình. Class này có các thuôc tính và phương thức sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| mac | Tên giao diện |
| ipAddress | Địa chỉ IP của giao diện |
| netmask | Netmask của giao diện |
| connectedNode | Danh sách lưu thông tin các nút mạng kết nối với giao diện này |
| updateConnectedNodes() | Cập nhật danh sách các nút mạng liên kết vơi giao diện |

Class còn lại, ConnectedNodeData dùng để lưu thông tin của nút mạng kết nối đến một giao diện trên thiết bị. Các thông tin được class lưu trong thuộc tính gồm có ID thiết bị ứng với nút mạng nếu có, địa chỉ IP và địa chỉ MAC. Class này chỉ phục vụ lưu trữ và truy vấn dữ liệu nên không có phương thức gì đặc biệt ngoài getter và setter.

Phần còn lại trong package là các class thực thể của chương trình. Mỗi class đại diện cho một loại thực thể trong cơ sở dữ liệu với mỗi thuộc tính là một trường giá trị tương ứng. Phương thức của các class này chủ yếu gồm constructor, getter và setter, do các đối tượng này chỉ để phục vụ việc lưu trữ dữ liệu trong quá trình trao đổi với hệ quản trị cơ sở dữ liệu của chương trình.

### Thiết kế các class trong package snmpd

Package này được chia thành 2 package nhỏ hơn là callbacks và helpers trong quá trình cài đặt nhưng khi nhìn về mặt chức năng và quan hệ giữa các class thì có thể chia package này theo 3 phần dưới đây: các class cơ sở, các class hỗ trợ truy vấn thông tin thiết bị không liên quan đến giao diện mạng và các class hỗ trợ truy vấn thông tin giao diên mạng.

Phần đầu, các class cơ sở là các class cung cấp chức năng và đối tượng căn bản trong hoạt động liên quan đến giao thức SNMP trong chương trình. Phần này có biểu đồ thiết kế như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.21: Biểu đồ lớp 1 của package snmpd

Trong đấy, class SnmpManager là một singleton, nắm giữ một số thành phần cần thiết trong hệ thống. Các thuôc tính, phương thức của class này được mô tả theo bảng sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| mib | Đối tượng dùng để tải các mib module sử dụng trong chương trình |
| notificationListener | Listener cho thông điệp Trap từ thiết bị |
| notificationProcessor | Thành phần bắt đầu quá trình xử lý thông điệp Trap |
| queryTimerManager | Thành phần quản lý các Timer được sử dụng trong hệ thống |
| initMib() | Phương thức tải các MIB module vào chương trình để sử dụng |
| initOtherComponents() | Phương thức thiết lập các notificationListener và Processor |
| createContext() | Phương thức tiện ích dùng trong tạo đối tượng Context phục vụ việc truy vấn |
| close() | Phương thức gọi đến TimerManager để dừng tất cả Timer trong chương trình còn đang hoạt động |

Class QueryTimerManager dùng trong việc quản lý các Timer sử dụng ở quá trình hoạt động của chương trình. Class này có các phương thức, thuộc tính sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| deviceActiveTimer | Timer dùng trong quá trình cập nhật trạng thái của SNMP Agent trên thiết bị |
| deviceResourceTimer | Timer dùng trong quá trình truy vấn lấy thông tin tài nguyên tự động |
| interfaceTimer | Timer dùng trong quá trình truy vấn lấy thông tin giao diện mạng tự động |
| startTimer() | Phương thức khởi động một Timer |
| cancelTimer() | Phương thức dừng hoạt động của một Timer |

Hai class còn lại, NotificationProcessor và NotificationParser là các class dùng trong quá trình xử lý thông điệp Notification từ thiết bị. Class đầu là một callback class với phương thức chính để bắt đầu quá trình xử lý thông điệp nhận được, còn class kia cung cấp một phương thức hỗ trợ cho quá trình phân tách nội dung của thông điệp Notification.

Phần thứ hai, các class hỗ trợ chức năng truy vấn đến lấy thông tin trên thiết bị không liên quan đến giao diện mạng. Các class này có biểu đồ thiết kế như bên dưới.

A close up of text on a white background

Description automatically generated

Hình 3.22: Biểu đồ lớp 2 của package snmpd

Class DeviceQueryHelper là class hỗ trợ gửi truy vấn cho thành phần Controller trong hệ thống. Class này gồm một số phương thức, thuôc tính sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| processorLoadTable | Danh sách các đối tượng MIB sử dụng từ bảng MIB hrProcessorTable |
| deviceTable | Danh sách các đối tượng MIB sử dụng từ bảng MIB hrDeviceTable |
| memoryTable | Danh sách các đối tượng MIB sử dụng từ bảng MIB hrStorageTable |
| startTemplateQuery() | Phương thức thực hiện quá trình gửi truy vấn với template |
| pushInfoIntoDevice() | Phương thức tiến hành quá trình gửi thông tin cơ bản lên thiết bị |
| startQueryDeviceResource() | Phương thức tiến hành quá trình truy vấn thu thập thông tin trạng thái tài nguyên thiết bị |

Các class callback được sử dụng để xử lý hồi đáp từ thiết bị trong các trường hợp gửi đi truy vấn. Các class này đều có phương thức onSnmpResponse để lấy ra danh sách Varbind trong thông điệp trả về rồi đưa cho các đối tượng khác trong quá trình thực thi chức năng của chương trình xử lý.

DeviceResourceDataCollector là class hỗ trợ class Callback xử lý, tổng hợp các MIB object trong thông điệp hồi đáp từ thiết bị. Class này có các phương thức, thuôc tính cơ bản sau.

|  |  |
| --- | --- |
| Thuôc tính / Phương thức | Mô tả |
| deviceID | ID của thiết bị được truy vấn đến |
| deviceCPUData | Danh sách lưu thông tin trạng thái của CPU trên thiết bị thu được |
| deviceMemoryData | Danh sách lưu thông tin trạng thái vùng nhớ thu được từ thiết bị |
| processCPUData() | Phương thức tổng hợp dữ liệu từ bảng MIB hrProcessorTable và lưu vào deviceCPUData theo từng dòng. |
| processDeviceData() | Phương thức xử lý dữ liệu từ bảng MIB hrDeviceTable để đưa thông tin mô tả CPU vào trong deviceCPUData. |
| processMemoryData() | Phương thức xử lý, tổng hợp dữ liệu từ bảng MIB hrStorageTable để lưu vào trong deviceMemoryData. |

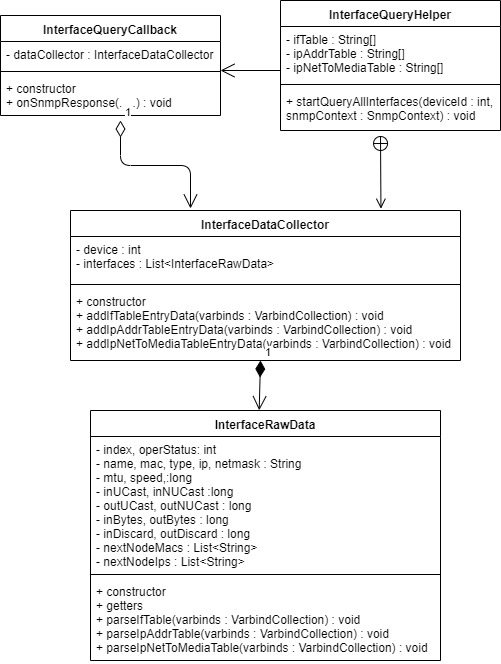
Hai class DeviceMemoryData và DeviceCpuData được dùng để lưu thông tin trạng thái tài nguyên thu được từ thiết bị nên các class này không có phương thức gì đặc biệt. Với class DeviceMemoryData, các thông tin trạng thái vùng nhớ được lưu trữ bao gồm type (kiểu), description(thông tin mô tả), totalSize (kích thước tổng cộng), usedSize(kích thước phần được sử dụng). Class còn lại DeviceCpuData lưu trữ các thông tin như firmwareId, description(thông tin mô tả) và load ( tải hiện tại của CPU).

Tương tự class TemplateQuery cũng dùng chủ yếu lưu trữ thông tin của quá trình truy vấn với template nên không có phương thức gì đặc biệt. Class này lưu lại các loại thông tin sau của quá trình truy vấn.

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| deviceId | ID của thiết bị được chọn |
| templateId | Id của template sử dụng |
| isTable | loại của template |
| itemList | danh sách các MIB object dùng trong truy vấn và xử lý hồi đáp |
| receivedTime | Thời gian nhận được hồi đáp |
| Result | Danh sách các giá trị của các MIB object được truy vấn |

Class cuối của thành phần thứ hai là ResponseDataProcessor, class dùng bởi callback của quá trình truy vấn với template. Class này có các phương thức processGetNextData() để xử lý, lấy ra giá trị các MIB object cho truy vấn với template loại Singular và processWalkData() cho các truy vấn với template loại Tabular.

Phần còn lại trong package snmpd là các class hỗ trợ truy vấn thông tin của giao diện mạng trên thiết bị. Phần này có biểu đồ thiết kế như sau.



Hình 3.23: Biểu đồ lớp 3 của package snmpd

Class đầu tiên, InterfaceQueryHelper, tương tự như DeviceQueryHelper là class hỗ trợ các thành phần Controller của chương trình trong quá trình gửi truy vấn lấy thông tin giao diện mạng. Class này gồm các phương thức, thuộc tính sau.

|  |  |
| --- | --- |
| **Thuôc tính / Phương thức** | **Mô tả** |
| ifTable | Danh sách các đối tượng được sử dụng trong bảng MIB ifTable |
| ipAddrTable | Danh sách các đối tượng được sử dụng trong bảng MIB ipAddrTable |
| ipNetToMediaTable | Danh sách các đối tượng được sử dụng trong bảng MIB ipNetToMediaTable |
| startQueryAllInterfaces() | Phương thức bắt đầu quá trình truy vấn đến lấy thông tin của toàn bộ giao diện mạng |

InterfaceDataCollector, tương tự như DeviceResourceDataCollector là class hỗ trợ class Callback trong trong việc tổng hợp dữ liệu. Các phương thức, thuôc tính của class này được mô tả theo bảng bên dưới.

|  |  |
| --- | --- |
| Thuộc tính / Phương thức | Mô tả |
| deviceId | ID của thiết bị được truy vấn đến |
| interfaces | Danh sách lưu thông tin giao diện thu được |
| addIfTableEntryData() | Phương thức xử lý lưu trữ dữ liệu của bảng ifTable vào interfaces theo từng dòng |
| addIpAddrTableEntryData() | Phương thức xử lý, tổng hợp dữ liệu thu được của bảng ipAddrTable vào interfaces, ghép với các dòng đã lưu của ifTable |
| addIpNetToMediaTableEntryData() | Phương thức xử lý, tổng hợp dữ liệu thu được của bảng ipNetToMediaTable vào interfaces, ghép với các dòng lưu trong đấy |

Hai class còn lại là class callback để xử lý hồi đáp trả về và class để lưu trữ thông tin của từng interface thu được. Class callback chỉ có một phương thức để đưa danh sách giá trị (variable-bindings) trong hồi đáp cho thuôc tính dataCollector xử lý. Còn InterfaceRawData có các thuộc tính và phương thức sau dùng cho việc thu, lưu thông tin giao diện mạng thu được.

|  |  |
| --- | --- |
| Thuộc tính / Phương thức | Mô tả |
| index, name, … | Các trường thuộc tính lưu thông tin thu được từ các bảng |
| parseIfTable() | Phương thức lấy giá trị trong một dòng thu được từ ifTable và lưu vào các trường thuộc tính |
| parseIpAddrTable() | Phương thức lấy các giá trị trong một dòng từ ipAddrTable và lưu vào thuộc tính |
| parseIpNetToMediaTable() | Phương thức lấy giá trị trong một dòng từ ipNetToMediaTable và lưu vào trường thuộc tính |

## Thiết kế giao diện

Giao diện cơ bản đầu tiên là giao diện thông tin các thiết bị đang được quản lý trong hệ thống. Thiết kế của giao diện này như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.24: Giao diện thông tin cơ bản của một thiết bị

Giao diện đang xét gồm 3 thành phần chính là thanh menu của chương trình ở phía trên cùng, panel chứa danh sách các thiết bị ở bên trái và panel hiển thị thông tin cơ bản của một thiết bị được chọn ở bên phải. Các thành phần giao diện con trên giao diện đang xét được mô tả theo bảng sau.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| mitem1 | Nút trên menu chính để mở giao diện chứa danh sách các thiết bị |
| mitem2 | Nút trên menu chính để mở giao diện chứa danh sách các template |
| mitem3 | Nút trên menu chính để mở giao diện chứa danh sách các thông điệp Notification |
| tfieldSearch | Ô điền tìm lọc thiết bị trong danh sách |
| panelDeviceList | Danh sách các thiết bị được quản lý |
| labelDevice1, labelDevice2, labelDevice3, … | Nhãn biểu diễn thiết bị được quản lý trong danh sách |
| labelInterface1, labelInterface2, labelInterface3 | Nhãn biểu diễn giao diện mạng của thiết bị được chọn, trong ví dụ là thiết bị ứng với labelDevice2 |
| buttonImport | Nút để mở giao diện chọn file nhập thiết bị |
| tfieldLabel | Ô text hiển thị nhãn thiết bị |
| tfieldName | Ô text hiển thị tên thiết bị |
| cboxType | Ô box chọn loại thiết bị |
| tfieldLocation | Ô text hiển thị thông tin vị trí của thiết bị |
| tareaDescription | Vùng text hiển thị thông tin mô tả hệ thống của thiết bị |
| labelLastAccess | Nhãn hiển thị thời gian cuối cùng truy vấn đến thiết bị |
| cboxSnmpVersion | Ô box hiển thị SNMP version của thiết bị |
| tfieldIpAddress | Ô text nhập địa chỉ IP của giao diện SNMP |
| tfieldPort | Ô text nhập số cổng của giao diện SNMP |
| tfieldCommunity | Ô text nhập community cho giao diện SNMP |
| buttonResources | Nút mở giao diện hiển thị tình trạng sử dụng tài nguyên trên thiết bị |
| buttonStatistics | Nút mở giao diện hiển thị các biểu đồ thống kê dữ liệu thu được từ thiết bị |
| buttonQuery | Nút mở giao diện chọn template để truy vấn |
| buttonSave | Nút lưu lại thông tin thiết bị được thay đổi |
| buttonCancel | Nút hủy bỏ việc thay đổi thông tin thiết bị |

Giao diện đầu tiên có thể mở từ giao diện thông tin cơ bản của thiết bị là giao diện cho phép người dùng quan sát các thông tin tài nguyên trên thiết bị. Giao diện này có thiết kế như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.25: Giao diện hiển thị thông tin sử dụng tài nguyên trên thiết bị

Một số thành phần giao diện con quan trọng trong thiết kế trên được mô tả ở bảng sau.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| tableCPU | Bảng hiển thị trạng thái hoạt động của CPU lấy được từ thiết bị với 3 cột thông tin Firmware ID, Description và Load |
| labelTotalRam | Nhãn hiển thị thông tin kích thước của bộ nhớ RAM. |
| labelUsedRam | Nhãn hiển thị kích thước phần được sử dụng của bộ nhớ RAM |
| labelTotalVirtual | Nhãn hiển thị thông tin kích thước vùng nhớ ảo. |
| labelUsedVirtual | Nhãn hiển thị thông tin kích thước phần được sử dụng của vùng nhớ ảo |
| tableDisks | Bảng thông tin trạng thái các ổ đĩa của thiết bị gồm 3 cột: tên ổ, kích thước tổng cộng, kích thước phần được dùng. |
| labelUpdatedTime | Nhãn hiển thị thời gian chương trình nhận được những dữ liệu hiển thị |
| tfieldUpdatePeriod | Ô nhập chu kỳ thời gian cập nhật tự động |
| buttonChange | Nút dùng thay đổi chu kỳ cập nhật tự động |
| buttonStop | Nút dừng cập nhật tự động |

Giao diện cơ bản tiếp theo có thể mở từ giao diện thông tin cơ bản của thiết bị là giao diện hiển thị các biểu đồ thống kê dữ liệu thu thập được. Giao diện này có thiết kế như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.26: Giao diện các biểu đồ thống kê

Các thành phần giao diện con của giao diện hiển thị biểu đồ thống kê được mô tả trong bảng sau.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| buttonToday | Nút kích hoạt hiển thị biểu đồ với dữ liệu thu thập trong ngày hiện tại |
| buttonYesterday | Nút kích hoạt hiển thị biểu đồ thống kê với dữ liệu của ngày trước ngày hiên tại |
| button3days | Nút kích hoạt hiển thị biểu đồ thống kê với dữ liệu tổng hợp trong 3 ngày trước |
| button7days | Nút kích hoạt hiển thị biểu đồ thống kê với dữ liệu thu được trong 7 ngày trước |
| cpuPanel | Panel hiển thị biểu đồ thống kê tải trung bình của CPU |
| memoryPanel | Panel hiển thị biểu đồ thống kê lượng bộ nhớ sử dụng trung binh |
| bandwidthPanel | Panel hiển thị biểu đồ thống kê cho lượng băng thông sử dụng trung bình |
| cboxMemory | Box để lựa chọn loại bộ nhớ cho quan sát riêng |
| cboxInterfaces | Box để lựa chọn giao diện mạng cho quan sát riêng |

Giao diện còn lại có thể xuất hiện từ giao diện thông tin cơ bản của thiết bị được chọn là giao diện hiển thị kết quả truy vấn với template. Giao diện này có thiết kế như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.27: Giao diện hiển thị kết quả truy vấn với template

Các thành phần con cơ bản của giao diện trên được liệt kê trong bảng dưới đây.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| tfieldTemplateName | Ô text hiển thị tên của template được chọn cho truy vấn |
| tfieldDeviceName | Ô text hiển thị nhãn thiết bị |
| tableResult | Bảng hiển thị kết quả truy vấn. Với template loại Singular, bảng kết quả có hai cột dùng hiển thị tên và giá trị của MIB object được truy vấn. Còn với loại Tabular, bảng kết quả có nhiều cột với mỗi cột hiển thị giá trị của một MIB object và mỗi dòng ứng với một dòng thông tin lấy ra được từ MIB |
| labelUpdatedTime | Nhãn hiển thị thời gian nhận được hồi đáp từ thiết bị |
| tfieldUpdatePeriod | Ô text chỉnh chu kỳ thời gian cập nhật tự động |
| buttonStart | Nút kích hoạt cập nhật tự động |
| buttonStop | Nút dừng việc cập nhật tư động |

Giao diện cơ bản tiếp theo của chương trình là giao diện hiển thị thông tin thu thập được của giao diện mạng được chọn. Giao diện này có thiết kế sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.28: Giao diện chương trình hiển thị thông tin về giao diện mạng

Giao diện trên có các thành phần giao diện con được mô tả theo bảng bên dưới.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| labelName, labelMacAddress, labelType, … | Các nhãn hiển thị thông tin giao diện mạng thu thập được, tương ứng với các nhãn hiển thị ý nghĩa bên trái. |
| labelUpdatedTime | Nhãn hiển thị thời gian chương trình nhận được các thông tin đang hiển thị |
| tfieldUpdatePeriod | Ô nhập chu kỳ thời gian cập nhật tự động |
| buttonChange | Nút thay đổi chu kỳ cập nhật tự động |
| buttonStop | Nút dừng việc cập nhật tự động |

Giao diện cơ bản tiếp theo là giao diện hiển thị thông tin của một template với thiết kế được biểu diễn bên dưới.

A screenshot of a cell phone

Description automatically generated

Hình 3.29: Giao diện hiện thị chi tiết template

Một số thành phần con hiển thị thông tin của giao diện trên được mô tả trong bảng dưới đây.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| labelTemplateName | Nhãn hiển thị tên của template |
| tareaTemplateDescription | Vùng text hiển thị các thông tin mô tả của template |
| labelImportedTime | Nhãn hiển thị thời gian template được nhập vào chương trình |
| tableItems | Bảng hiển thị các item của template. Bảng này có 4 cột gồm tên hiển thị, MIB name, OID và cột chứa các nút bật tắt item. Mỗi dòng trong bảng ứng với một template item, chứa thông tin về MIB object dùng trong truy vấn |
| buttonSave | Nút lưu thông tin thay đổi |
| buttonCancel | Nút hủy việc sửa đổi thông tin |

Giao diện chính còn lại trong chương trình là giao diện danh sách của các Notification thu nhận được. Giao diện này có thiết kế như sau.

A screenshot of a cell phone

Description automatically generated

Hình 3.30: Giao diện danh sách các thông điệp Notifications

Các thành phần con trong giao diện trên được liệt kê trong bảng phía dưới.

|  |  |
| --- | --- |
| Tên đối tượng giao diện | Ý nghĩa |
| cboxColdStart, cboxWarmStart, … | Các nút để lựa chọn kiểu thông điệp Notification muốn quan sát |
| tfieldSearch | Ô text nhập thông tin thiết bị gửi để lọc thông điệp |
| tableNotfications | Bảng danh sách các thông điệp được người dùng lọc ra, với 4 cột thông tin như thiết bị gửi, kiểu thông điệp, nội dung cơ bản và thời gian nhận được thông điệp. |

## Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống được thiết kế theo biểu đồ thực thể quan hệ sau.

A close up of text on a white background

Description automatically generated

Hình 3.31: Biểu đồ thực thể quan hệ của chương trình

Bảng tổng quan cơ sở dữ liệu của chương trình.

|  |  |
| --- | --- |
| **Tên bảng** | **Ý nghĩa** |
| Device | Bảng lưu thông tin cơ bản của thiết bị được quản lý |
| ContactInteface | Bảng lưu thông tin giao diện SNMP dùng trong quá trình trao đổi với trạm quản lý |
| DeviceNetworkInterface | Bảng lưu một số thông tin cơ bản, ít thay đổi của các giao diện mạng ở mỗi thiết bị |
| DeviceCPUState | Bảng lưu thông tin trạng thái của CPU thu được từ thiết bị, dùng cho việc tính toán, thống kê |
| DeviceMemoryState | Bảng lưu thông tin trạng thái của vùng nhớ thu được từ thiết bị, dùng cho việc tính toán, thống kê |
| DeviceInterfaceDynamicData | Bảng lưu một số thông tin trạng thái của các giao diện mạng dùng cho việc tính toán, thống kê |
| Notification | Bảng lưu thông điệp Trap nhận được sau xử lý |
| NotificationExtraData | Bảng lưu các cặp varbind trong phần nội dung của thông điệp Notification |
| Template | Bảng lưu thông tin của các template được nhập vào |
| TemplateItem | Bảng lưu thông tin các item của template trong thông điệp |

Đặc tả của bảng Device.

|  |  |  |
| --- | --- | --- |
| **Tên trường** | **Ý nghĩa** | **Ràng buộc** |
| Id | Khóa chính | Non-null, unique |
| Label | Nhãn thiết bị | Non-null, unique |
| Name | Tên thiết bị |  |
| Type | Kiểu của thiết bị |  |
| Description | Mô tả hệ thống của thiết bị |  |
| Location | Vị trí của thiết bị |  |
| Snmp\_version | SNMP version thiết bị hỗ trợ |  |
| Imported\_time | Thời gian thiết bị nhập vào |  |

Đặc tả của bảng ContactInterface.

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Device\_id | Khóa ngoài tham chiếu đến ID của thiết bị sở hữu giao diện | Non-null |
| Ip\_address | Địa chỉ IP của giao diện |  |
| Port | Số hiệu cổng nhận truy vấn của thiết bị | >= 0 |
| Community | Community |  |
| Update\_time | Thời gian giao diện được cập nhật |  |

Đặc tả của bảng DeviceNetworkInterface

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Device\_id | Khóa ngoài tham chiếu đến ID của thiết bị sở hữu giao diện mạng | Non-null |
| Id\_x | Số thứ tự của giao diện mạng trong các giao diện mạng mà thiết bị có | >= 0 |
| Name | Tên của giao diện mạng |  |
| Mac\_address | Địa chỉ vật lý của giao diện |  |
| Type | Kiểu của giao diện mạng |  |

Đặc tả của bảng DeviceInterfaceDynamicData.

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Network\_interface\_id | Khóa ngoài tham chiếu đến ID của giao diện mạng mà thông tin này thu thập được từ | Non-null |
| Bandwidth | Giá trị băng thông của giao diện mạng | >= 0 |
| Inbound\_bytes | Lượng byte dữ liệu vào giao diện mạng | >= 0 |
| Outbound\_bytes | Lượng byte dữ liệu ra khỏi giao diện mạng | >= 0 |
| Updated\_time | Thời giant nhận được thông tin |  |

Đặc tả của bảng DeviceCPUState

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Device\_id | Khóa ngoài tham chiếu đến ID của thiết bị gửi thông tin này | Non-null |
| Firmware\_id | Firmware ID của CPU |  |
| Description | Mô tả của CPU |  |
| Cpu\_load | Tải của CPU thu được thông qua thông tin trạng thái | >= 0 |
| Updated\_time | Thời gian nhận được các thông tin |  |

Đặc tả của bảng DeviceMemoryState

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Device\_id | Khóa ngoài tham chiếu dến ID của thiết bị gửi thông tin này | Non-null |
| Type | Loại bộ nhớ |  |
| Description | Mô tả của bộ nhớ |  |
| total\_size | Kích thước tổng cộng | >= 0 |
| used\_size | Kích thước phần được sử dụng | >= 0 |
| Updated\_time | Thời gian nhận được thông tin |  |

Đặc tả của bảng Notification

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Device\_id | Khóa ngoài tham chiếu đến ID của thiết bị gửi thông điệp | Non-null |
| NotificationType | Loại thông điệp Trap nhận được |  |
| Content | Nội dung cơ bản của thông điệp Notification |  |

Đặc tả của bảng NotificationExtraData

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Notification\_id | Khóa ngoài tham chiếu đến ID của thông điệp Notification chứa dữ liệu này | Non-null |
| Id\_x | Số thứ tự của cặp varbind trong các cặp ở nội dung thông điệp |  |
| Name | Tên của MIB object trong cặp |  |
| Value | Giá trị của MIB object trong cặp |  |

Đặc tả của bảng Template.

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Name | Tên của template |  |
| Description | Mô tả của template |  |
| Is\_table | Dùng xác định loại template là Singular hay Tabular |  |
| Imported\_time | Thời gian template được đưa vào chương trình |  |

Đặc tả của bảng TemplateItem.

|  |  |  |
| --- | --- | --- |
| Tên trường | Ý nghĩa | Ràng buộc |
| Id | Khóa chính | Non-null, unique |
| Template\_id | Khóa ngoài tham chiếu đến ID của template sở hữu item này | Non-null |
| Id\_x | Số thứ tự của item này trong danh sách các item nhận được | >= 0 |
| Mib\_name | Tên của MIB object theo định nghĩa bởi MIB module |  |
| OID | Object ID của MIB object ứng với item |  |
| Display\_name | Tên dùng trong hiển thị của item |  |
| Description | Mô tả ý nghĩa của MIB object |  |
| Is\_enabled | Trạng thái bật tắt của item |  |

# TRIỂN KHAI THỬ NGHIỆM

## Môi trường triển khai thử nghiệm

### Công cụ sử dụng

Do những hạn chế về mặt kinh phí, trang thiết bị nên quá trình triển khai thử nghiệm của chương trình sản phẩm được thực hiện trên môi trường ảo cung cấp bởi hai chương trình GNS3 và VirtualBox. GNS3 được sử dụng để xây dựng topology mạng còn VirtualBox để cung cấp một số máy ảo dùng trong thử nghiệm. Ngoài ra chương trình còn sử dụng IOS image của Cisco để tạo router trong topology mạng và JDK (JRE) để chạy chương trình sản phẩm.

Môi trường thực hiện

### Môi trường triển khai

Các chương trình trên và topology của quát trình triển khai thử nghiệm được thực hiện với máy tính có một số thông tin cấu hình sau.

* Hệ điều hành: Ubuntu phiên bản 16.04 LTS.
* RAM: 8 GB.
* Bộ nhớ: 1 TB.
* Vi xử lý: 4 nhân với tốc độ tối đa 2.60 GHz.

## Mô hình triển khai thử nghiệm

### Biểu đồ triển khai và mô tả

Quá trình triển khai thử nghiệm được thực hiện trên mô hình mạng theo biểu đồ sau.

A close up of a map

Description automatically generated

Hình 4.1: Biểu đồ triển khai thử nghiệm của chương trình

Topology triển khai trên bao gồm có 2 mạng đại diện cho một mạng nội bộ và mạng bên ngoài. Các thiết bị trong mạng nội bộ gồm có một máy quản lý, một máy trạm đại diện, một switch liên kết và hai router nối ra mạng bên ngoài. Trên máy trạm quản lý có cài đặt môi trường để chạy chương trình sản phẩm của đề tài. Các máy còn lại trong mạng nội bộ được cấu hình SNMP để Agent hoạt động, phục vụ trong quá trình trao đổi thông tin với trạm quản lý. Các thiết bị trong mạng nội bộ được kết nối với switch. Ngoài giao diện nối với mạng nội bộ, các router cũng có một giao diện được đấu nối ra một mạng ở bên ngoài.

### Thông số các thiết bị sử dụng

Thông số của các thiết bị sử dụng trong topology thử nghiệm trên được liệt kê dưới đây.

Trạm quản lý: máy ảo cung cấp bởi VirtualBox

* Hệ điều hành: Ubuntu phiên bản 16.04LTS.
* RAM: 2 GB.
* Bộ nhớ ảo được cấp phát: 40 GB.
* Vi xử lý được cấp: 2 nhân
* Cổng kết nối: 1 cổng Ethernet

Máy trạm PC: máy ảo cung cấp bởi VirtualBox

* Hệ điều hành: Centos phiên bản 7.0 Minimal
* RAM: 1 GB.
* Vi xử lý được cấp: 1 nhân
* Bộ nhớ ảo được cấp phát: 12 GB.
* Cổng kết nối: 1 cổng Ethernet

Hai router: router ảo Cisco 3725 cung cấp bởi GNS 3.

* RAM: 256 MB.
* Hệ điều hành: Cisco IOS c3725 phiên bản 12.4.15T14
* Cổng kết nối: 2 cổng Ethernet và 6 cổng Serial.

Switch: Switch ảo, cung cấp bởi GNS 3.

* Cổng kết nối: 8 cổng Ethernet.

## Thử nghiệm và kết quả.

Thử nghiệm chức năng quan sát thông tin trạng thái giao diện mạng.

Cách thức thực hiện:

* Tiến hành ping từ máy trong mạng ra mạng bên ngoài
* Trên chương trình quản lý, quan sát trạng thái giao diện mạng của máy bắt đầu quá trình ping và Router.

Kết quả thu được: Một số thông số trạng thái của giao diện mạng như lượng gói tin, dữ liệu vào ra tăng dần theo thời gian.

Thử nghiệm chức năng quan sát thông tin trạng thái môt số tài nguyên trên thiết bị.

Cách thức thực hiện:

* Tiến hành chạy một số ứng dụng trên máy trạm đại diện.
* Quan sát trên giao diện hiển thị thông tin trạng thái tài nguyên của máy trạm.

Kết quả thu được: Trong thời gian ứng dụng được chạy, một số thông tin tài nguyên như tải của CPU, bộ nhớ RAM sẽ tăng lên và giảm xuống một mức thấp hơn khi các ứng dụng dừng lại.

Thử nghiệm chức năng xử lý, phân tích thông điệp Trap.

Các thức thực hiện:

* Bật, tắt một giao diện mạng của Router, ví dụ như giao diện nối với mạng ngoài.
* Quan sát trên bảng thông báo hoặc danh sách thông điệp của chương trình.

Kết quả thu được: Một thông điệp xuất hiện với một số thông tin của thiết bị gửi như địa chỉ, nhãn và thông tin về sự kiện như tên giao diện mạng và trạng thái tương ứng.

Thử nghiệm cơ chế tính toán, thống kê.

Cách thức thực hiện:

* Thực hiện quá trình truyền gửi dữ liệu giữa các nút mạng
* Thu thập thông tin từ thiết bị trong nhiều ngày, giờ với hệ thống truy vấn tự động rồi quan sát.

Kết quả thu được: Các biểu đồ khác nhau cho các dữ liệu thu được.

Thử nghiệm cơ chế template

Cách thức thực hiện:

* Chọn một thiết bị với Agent đang hoạt động rồi chọn một template cho truy vấn.
* Có thể thay đổi lượng item trong template rồi gửi truy vấn lại.

Kết quả thu được: Chỉ các giá trị của đối tượng MIB ứng với item được bật trong template được lấy về hiển thị lên giao diện.

## Đánh giá kết quả thử nghiệm

Sau quá trình thử nghiệm, có thể nhận thấy là một số chức năng cơ bản hệ thống đã hoạt động ổn định. Chương trình đã cho phép người dùng quan sát được trạng thái hoạt động của các cổng trên thiết bị, lượng sử dụng của một số loại tài nguyên. Bên cạnh đấy, cơ chế cập nhật tự động cũng cho phép người dùng có thể quan sát sự thay đổi của các thông tin này theo thời gian hoạt động của thiết bị. Thành phần xử lý và hiển thị Trap tuy hoạt động nhưng còn nhiều mặt hạn chế, mới chỉ phân tích được các loại thông điệp cơ bản, chưa hiệu quả với các thông điệp Trap Enterprise do tổ chức, nhà cung cấp thiết bị, ứng dụng định nghĩa. Các cơ chế phụ trợ khác cũng đã hoạt động ổn định nhưng vẫn chưa thực sự đem lại được nhiều ích lợi cho quá trình sử dụng. Tổng kết lại, chương trình đã có thể đem vào ứng dụng nhưng còn cần rất nhiều những sự phát triển và cải tiến để hỗ trợ được nhiều tính năng hơn cho người dùng.

# PHẦN KẾT LUẬN

Chương trình sản phẩm sau quá trình phát triển đã hoàn thành được các mục tiêu cơ bản đưa ra bởi đề tài. Chương trình đã đạt được mục tiêu thu thập được các thông tin về sử dụng tài nguyên phần cứng, giao diện mạng trên thiết bị đang hoạt động. Chương trình cũng đã có thể tính toán được một số loại giá trị thống kê cơ bản, phục vụ việc đánh giá trạng thái hoạt động, khả năng cung ứng dịch vụ của thiết bị. Với cơ chế nhận và xử lý thông điệp Trap từ thiết bị, chương trình cũng đã phần nào đạt được mục tiêu cảnh báo khi có sự kiện bất thường xảy ra. Ngoài ra, chương trình cũng hỗ trợ thêm một số chức năng khác cho người dùng để họ có thể có thêm nhiều lựa chọn và giảm thiểu được các chi phí như về mặt thời gian, công sức trong quá trình sử dụng.

Bên cạnh những kết quả đạt được bên trên, chương trình vẫn còn một số măt hạn chế cả về mặt tính năng và hiệu năng hoạt động. Chương trình cũng chưa cung cấp được các tính năng hỗ trợ quản trị hệ thống mạng cho người dùng. Bởi vậy, chương trình còn cần rất nhiều những cải tiến để có thể đáp ứng được những vấn đề đã, đang và có thể xảy ra trong môi trường mạng máy tính ngày nay.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | J. Case, K. McCloghrie, M. Rose and S. Waldbusser, Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework, RFC 1452. |
| [2] | J. Case, K. McCloghrie, M. Rose and S. Waldbusser, Introduction to version 2 of the Internet-standard Network Management Framework, RFC 1441. |
| [3] | J. Case, M. Fedor, M. Schoffstall and J. Davin, A Simple Network Management Protocol, RFC 1157, 1990. |
| [4] | T. T. H. Hải, Slide bài giảng môn Quản trị mạng. |