TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

ĐỒ ÁN

**TỐT NGHIỆP ĐẠI HỌC**

NGÀNH CÔNG NGHỆ THÔNG TIN

**Xây dựng hệ thống Voip trên hệ điều hành Android**

Sinh viên thực hiện:  **Nguyễn Khắc Vinh**

Lớp CNPM-K51

Giáo viên hướng dẫn: **ThS Vũ Đức Vượng**

HÀ NỘI 5-2011

# PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên:  *Nguyễn Khắc Vinh*

Điện thoại liên lạc *01696890500* Email: *nkvinh@gmail.com*

Lớp:  *CNPM*  Hệ đào tạo: *Đại học chính quy*

Đồ án tốt nghiệp được thực hiện tại: *Trung tâm phần mềm Viettel*

Thời gian làm ĐATN: Từ ngày *28 /2 /2011* đến *28/5 /2011*

2. Mục đích nội dung của ĐATN

* Tìm hiểu kiến trúc Voip và hệ điều hành cho di động Android
* Xây dựng hoàn thiện một hệ thống Voip, demo phía client trên máy Android

3. Các nhiệm vụ cụ thể của ĐATN

* Tìm hiểu về kiến trúc của Voip
* Tìm hiểu về hệ điều hành Android
* Tìm hiểu thư viện hỗ trợ để lập trình Voip Client trên Android
* Chọn proxy server để xây dựng hệ thống
* Dùng thư viện và server đã chọn xây dựng một hệ thống Voip
* Bảo mật cho tài khoản khi đăng kí với server
* Tìm hiểu và lựa chọn giao thức vượt NAT

4. Lời cam đoan của sinh viên:

Tôi – *Nguyễn Khắc Vinh* - cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *thạc sĩ Vũ Đức Vượng*

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác

|  |  |
| --- | --- |
|  | *Hà Nội, ngày 25 tháng 05 năm 2011*  Tác giả ĐATN  *Nguyễn Khắc Vinh* |

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

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng năm*  Giáo viên hướng dẫn  *Thạc sĩ Vũ Đức Vượng* |

# 

# MỤC LỤC

[PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP 2](#_Toc294267134)

[MỤC LỤC 4](#_Toc294267135)

[KHÁI NIỆM VÀ THUẬT NGỮ 5](#_Toc294267136)

[DANH MỤC CÁC BẢNG 6](#_Toc294267137)

[DANH MỤC CÁC HÌNH 7](#_Toc294267138)

[LỜI CẢM ƠN 8](#_Toc294267139)

[MỞ ĐẦU 8](#_Toc294267140)

[1. Giới thiệu đề tài 8](#_Toc294267141)

[2. Môi trường thực hiện đồ án tốt nghiệp 9](#_Toc294267142)

[3. Bố cục của đồ án tốt nghiệp 9](#_Toc294267143)

[PHẦN I: ĐẶT VẤN ĐỀ VÀ ĐỊNH HƯỚNG GIẢI PHÁP 10](#_Toc294267144)

[1 Tổng quan về đề tài 10](#_Toc294267145)

[1.1 Các vấn đề cần giải quyết 10](#_Toc294267146)

[2 Giới thiệu tóm tắt về cơ sở lý thuyết 10](#_Toc294267147)

[2.1 Giới thiệu về VOIP 10](#_Toc294267148)

[2.1.1 Giới thiệu về SIP 11](#_Toc294267149)

[2.1.2 Giới thiệu về RTP 12](#_Toc294267150)

[2.2 Đăng ký, thiết lập cuộc gọi với SIP proxy server 13](#_Toc294267151)

[2.2.1 Đăng ký mới: 13](#_Toc294267152)

[2.2.2 Thiết lập cuộc gọi 15](#_Toc294267153)

[2.3 Cơ bản về Android 19](#_Toc294267154)

[2.3.1 Activity: 19](#_Toc294267155)

[2.3.2 Service: 20](#_Toc294267156)

[2.3.3 Intent: 20](#_Toc294267157)

[2.3.4 Broadcast Receiver: 21](#_Toc294267158)

[2.3.5 Notification: 21](#_Toc294267159)

[2.3.6 SQLLite 21](#_Toc294267160)

[2.3.7 XML trong Android: 21](#_Toc294267161)

[2.3.8 Android Manifest: 21](#_Toc294267162)

[3 Các vấn đề đặt ra 21](#_Toc294267163)

[3.1 Xác thực dữ liệu 22](#_Toc294267164)

[3.2 NAT và ảnh hưởng của nó tới VoIP 22](#_Toc294267165)

[4 Giải pháp và công cụ được lựa chọn để giải quyết các vấn đề đặt ra 22](#_Toc294267166)

[4.1 Giải pháp xác thực dữ liệu 22](#_Toc294267167)

[4.2 Giải pháp cho các thiết bị nằm trong NAT 22](#_Toc294267168)

[4.2.1 Giải pháp qua NAT cho các tín hiệu SIP 22](#_Toc294267169)

[4.2.2 Giải pháp truyền các gói tin RTP qua NAT 23](#_Toc294267170)

[4.3 Chọn lựa chuẩn xây dựng hệ thống VoIP 24](#_Toc294267171)

[4.4 Lựa chọn thư viện phát triển ứng dụng SIP trên client 24](#_Toc294267172)

[4.5 Lựa chọn SIP proxy 24](#_Toc294267173)

[PHẦN II: XÂY DỰNG HỆ THỐNG VOIP 25](#_Toc294267174)

[1 Khảo sát, phân tích yêu cầu 25](#_Toc294267175)

[1.1 Mục tiêu của hệ thống 25](#_Toc294267176)

[1.2 Phân tích yêu cầu 25](#_Toc294267177)

[1.2.1 Các ràng buộc thiết kế (Design Constraints) 25](#_Toc294267178)

[1.2.2 Các yêu cầu về hiệu năng (Performance) 25](#_Toc294267179)

[1.3 Thiết kế tổng quan 25](#_Toc294267180)

[1.3.1 Thiết kế kiến trúc hệ thống 25](#_Toc294267181)

[1.4 Thực hiện, cài đặt và thử nghiệm hệ thống 26](#_Toc294267182)

[1.4.1 Triển khai phía server: 26](#_Toc294267183)

[1.4.2 Phía Client 30](#_Toc294267184)

[1.4.2.2.1 Phân tích cụ thể từng chức năng: 31](#_Toc294267185)

[1.4.2.2.2 Các luồng thông tin trong ứng dụng Voip Android 32](#_Toc294267186)

[1.4.2.2.3 Thiết kế cơ sở dữ liệu 33](#_Toc294267187)

[a. Bảng Account: 33](#_Toc294267188)

[b. Bảng CallLogs 34](#_Toc294267189)

[c. Bảng Messages: 35](#_Toc294267190)

[1.4.2.2.4 Thiết kế giải thuật 36](#_Toc294267191)

[1.4.2.2.5 Kiến trúc ứng dụng Voip Android 39](#_Toc294267192)

[1.4.2.2.6 Chi tiết cài đặt 40](#_Toc294267193)

[❖ Voip Android Service: 40](#_Toc294267194)

[❖ Voip Android Models: 47](#_Toc294267195)

[❖ Voip Android UI 51](#_Toc294267196)

[TÀI LIỆU THAM KHẢO 62](#_Toc294267197)

[PHỤ LỤC 62](#_Toc294267198)

# 

# KHÁI NIỆM VÀ THUẬT NGỮ

| Thuật ngữ | Định nghĩa | Ghi chú |
| --- | --- | --- |
| VOIP | Voice over Internet Protocol | Công nghệ cho phép truyền thoại sử dụng giao thức mạng IP |
| VOIP over 3G | Voice over Internet Protocol over 3G | Công nghệ cho phép truyền thoại sử dụng giao thức mạng IP trên hạ tầng mạng 3G (third-generation) |
| TDM | Time Division Multiplexing | Ghép kênh phân chia thời gian: Thời gian sử dụng đường truyền được chia làm nhiều khung. Mỗi khung được chia thành nhiều khe thời gian(Ts time slot) mỗi người sử dụng một khe thời gian dành riêng cho mình để phục vụ cho việc truyền tin. |
| IP | Internet Protocol | Giao thức mạng được sử dụng rộng rãi ngày nay |
| End point |  | Các thiết bị đầu cuối: bao gồm IP phone và SIP phone. |
| SIP | Session Initiation Protocol | Giao thức khởi tạo session |
| UA/ UAC/ UAS | User Agent/ Uer Agent Client/ User Agent Server | Nơi gửi và nơi đáp trả các bản tin SIP, thông thường là các thiết bị SIP phone |
| G7xx | G711/ G729, … | Các codec dùng số hóa âm thanh |
| Proxy server |  | Server trung chuyển các bản tin SIP |
| Redirect server |  | Server trả lời điểm tiếp theo mà thiết bị phải gửi bản tin tới. |
| Registrar server |  | Server đăng ký cho các user agent. |
|  |  |  |

Thuật ngữ và định nghĩa

DANH MỤC CÁC BẢNG

[Table 1: Bảng Account 37](#_Toc294141273)

[Table 2: Bảng CallLogs 38](#_Toc294141274)

[Table 3: Bảng Messages 39](#_Toc294141275)

DANH MỤC CÁC HÌNH

[Figure 1: Kiến trúc của hệ thống Voip 12](#_Toc294222185)

[Figure 2: Đăng kí với Sip Server thành công 15](#_Toc294222186)

[Figure 3: Thiết lập thành công cuộc gọi 17](#_Toc294222187)

[Figure 4: Cơ chế xác thực username và password 24](#_Toc294222188)

[Figure 5 : Giải pháp qua NAT cho các tín hiệu SIP 25](#_Toc294222189)

[Figure 6: Giải pháp truyền các gói tin RTP qua NAT 25](#_Toc294222190)

[Figure 7: Mô hình hệ thống cần xây dựng 28](#_Toc294222191)

[Figure 8: Mô hình Use case của ứng dụng Voip Android 34](#_Toc294222192)

[Figure 9: Biểu đồ tuần tự cho chức năng thêm và xem các account 39](#_Toc294222193)

[Figure 10: Biểu đồ tuần tự cho chức năng xem và xóa lịch sử cuộc gọi 39](#_Toc294222194)

[Figure 11 : Biểu đồ tuần tự cho chức năng xem và xóa tin nhắn 40](#_Toc294222195)

[Figure 12: : Biểu đồ tuần tự cho chức năng gửi tin nhắn và tạo cuộc gọi 41](#_Toc294222196)

[Figure 13 : Biểu đồ tuần tự cho chức năng xóa và sửa tài khoản 42](#_Toc294222197)

[Figure 14: Kiến trúc ứng dụng Voip Android 42](#_Toc294222198)

[Figure 15 Class UAStateReceiver 44](#_Toc294222199)

[Figure 16 SipService 46](#_Toc294222200)

[Figure 17 SipNotifications 48](#_Toc294222201)

[Figure 18 Sơ đồ lớp xử lí Media 50](#_Toc294222202)

[Figure 19 Class SipMessage 51](#_Toc294222203)

[Figure 20 Class CallInfo 52](#_Toc294222204)

[Figure 21 Class IAccount 53](#_Toc294222205)

[Figure 22 AccountState 54](#_Toc294222206)

[Figure 23 Sơ đồ lớp cho giao diện thêm account 55](#_Toc294222207)

[Figure 24 Sơ đồ lớp của Voip Android widget 57](#_Toc294222208)

[Figure 25 Sơ đồ lớp của giao diện xử lí cuộc gọi trong Voip Android 58](#_Toc294222209)

[Figure 26 Sơ đồ lớp giao diện chính của Voip Android 59](#_Toc294222210)

[Figure 27 Sơ đồ lớp giao diện cài đặt của Voip Android 60](#_Toc294222211)

[Figure 28 Sơ đồ lớp giao diện quản lí tài khoản của Voip Android 61](#_Toc294222212)

[Figure 29 Sơ đồ lớp giao diện quản lí lịch sử cuộc gọi của Voip Android 62](#_Toc294222213)

[Figure 30 Sơ đồ lớp chọn tài khoản để gọi hoặc gửi tin của Voip Android 63](#_Toc294222214)

[Figure 31 Quản lí tài khoản 66](#_Toc294222215)

[Figure 32 Màn hình cài đặt và gửi tin nhắn 67](#_Toc294222216)

[Figure 33 Màn hình quản lí cuộc gọi 68](#_Toc294222217)

LỜI CẢM ƠN

Trước tiên em xin gửi lời cám ơn chân thành sâu sắc tới các thầy cô giáo trong trường Đại học Bách Khoa Hà Nội nói chung và các thầy cô giáo trong viện Công nghệ Thông tin, bộ môn Công nghệ Phần mềm nói riêng đã tận tình giảng dạy, truyền đạt cho em những kiến thức, kinh nghiệm quý báu trong suốt thời gian qua.

Đặc biệt, em xin gửi lời cảm ơn đến thầy Vũ Đức Vượng. Thầy đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn em trong suốt quá trình làm đồ án tốt nghiệp. Trong thời gian làm việc với thầy, em không ngừng tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần làm việc, thái độ nghiên cứu khoa học nghiêm túc, hiệu quả, đây là những điều rất cần thiết cho em trong quá trình học tập và công tác sau này.

Đồng thời em xin chân thành cảm ơn anh Cao Xuân Tuấn, trưởng phòng R&D, Trung tâm phần mềm Viettel và anh Đỗ Đình Thắng trưởng ban công nghệ đã tạo mọi điều kiện về cơ sở vật chất giúp em có một môi trường tốt để thực hiện đề tài đã giúp em hoàn thành đề tài này

Hà Nội, ngày 25 tháng 5 năm 2011

Người thực hiện

Nguyễn Khắc Vinh

MỞ ĐẦU

## Giới thiệu đề tài

Voice over Internet Protocol (VOIP), còn gọi là IP Telephony, đang nhanh chóng trở thành một thuật ngữ thông thuộc và công nghệ này đang đi sâu vào các doanh nghiệp, các tổ chức .VOIP được thiết kế để thay thế mạng lưới và công nghệ TDM bằng một mạng dữ liệu dựa trên giao thức mạng IP .Tiếng nói được số hóa và truyền đi trong các gói tin IP .Trong các doanh nghiệp hiện nay nhất là các công ty mạng Internet và cả wifi đã len lỏi đến mọi phòng ban .Mặt khác nhu cầu liên lạc nội bộ trong doanh nghiệp là rất lớn .Sẽ là rất tuyệt vời nếu có một hệ thống Voip ngay trong công ty ,giúp cho việc liên lạc trở nên dễ dàng hơn mà chi phí gần như bằng không vì tận dụng được hạ tầng mạng IP có sẵn .Vì lí do đó em chọn đề tài

**Xây dựng hệ thống Voip trên hệ điều hành Android**

## Môi trường thực hiện đồ án tốt nghiệp

Địa điểm: Trung tâm phần mềm Viettel, Phòng R&D

Thành viên đội thực hiện dự án:

Quản trị dự án: Đỗ Đình Thắng

Thành viên dự án: Đào Hải Hưng

Thành viên dự án: Nguyễn Khắc Vinh

Dự án: Nghiên cứu thử nghiệm công nghệ Voip

## Bố cục của đồ án tốt nghiệp

Đồ án tốt nghiệp được chia thành các phần như sau:

**MỞ ĐẦU**

Giới thiệu đề tài, môi trường thực hiện đồ án, tóm tắt bố cục của đồ án tốt nghiệp

**PHẦN 1**: Đặt vấn đề và định hướng giải pháp

1. Các nhiệm vụ cần thực hiện trong đề tài
2. Cơ sở lý thuyết và công cụ sử dụng.

**PHẦN 2**: Các kết quả đạt được

1. Phân tích yêu cầu
2. Thiết kế hệ thống
3. Cài đặt
4. Đánh giá ưu, nhược điểm của hệ thống

**KẾT LUẬN**

Kết luận chung: tự đánh giá công việc làm được và chưa làm được, định hướng phát triển và hoàn thiện hệ thống trong tương lai.

# PHẦN I: ĐẶT VẤN ĐỀ VÀ ĐỊNH HƯỚNG GIẢI PHÁP

## Tổng quan về đề tài

* 1. **Các vấn đề cần giải quyết**
     + Tìm hiểu về kiến trúc của Voip
     + Tìm hiểu về hệ điều hành Android
     + Tìm hiểu thư viện hỗ trợ để lập trình Voip Client trên Android
     + Chọn server để xây dựng hệ thống
     + Dùng thư viện và server đã chọn xây dựng một hệ thống Voip cơ bản
     + Bảo mật cho tài khoản khi đăng kí với server
     + Tìm hiểu và lựa chọn giao thức vượt NAT

1. **Giới thiệu tóm tắt về cơ sở lý thuyết**

### Giới thiệu về VOIP

Trong một mạng VOIP, có 2 giao thức được sử dụng chính .Giao thức đầu tiên là Signaling Protocol (giao thức báo hiệu): dùng trong việc khởi tạo và quản lý các tương tác giữa các user session bao gồm voice, video, instant messaging…. Giao thức thứ hai là speech transmission protocol: giao thức truyền tiếng nói. Cả hai giao thức yêu cầu tất cả các thông tin được truyền tải trong các gói tin IP. Cho tới nay có một vài chuẩn cho signaling protocols, gồm có H.323, SIP .Còn RTP là chuẩn cho speech transmission protocol được sử dụng trong các mạng VOIP .Tín hiệu tiếng nói được số hóa, đóng gói trong các gói tin và được truyền thông qua mạng IP .Ta cần rất nhiều các gói tin mới có thể truyền tải được một âm đơn do một người phát ra .Tín hiệu voice được số hóa bằng các sử dụng một trong các chuẩn G.7xx.

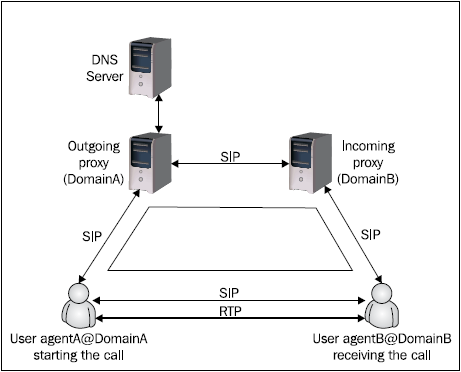


Figure 1: Kiến trúc của hệ thống Voip

Proxy server là một thiết bị trung gian nhận SIP request từ một client và sau đó chuyển tiếp request cho tới khi tới được sip client cần tới .Proxy server là thành phần quan trọng trong hệ thống VoIP, cho phép các user agent trao đổi thông tin, đăng ký người dùng và quản lý trạng thái user agent .Proxy server KHÔNG thực hiện chuyển mạch điện thoại (telephone switch) .Khi proxy server thiết lập một cuộc gọi (peer-to-peer), server sẽ không tham gia vào quá trình truyền tải tiếng nói .Các gói tin chứa tiếng nói được truyền trực tiếp từ một phone tới một phone khác

Ngoài ra còn có Registrar server Xử lý các requests từ các UAC để đăng ký vị trí (ip, port) của UAC đó .Registrar server thường được đặt cùng với proxy server

#### Giới thiệu về SIP

SIP là giao thức tầng application trong mô hình OSI nó được sử dụng để thiết lập, thay đổi, và kết thúc các session, SIP là giao thức thuần text (clear text) .Thiết lập các tham số về media được sử dụng (codec, ip, port) giữa các UA thông qua Session Description Protocol (SDP) .Giao thức SIP được thiết kế là một phần của một kiến trúc đa phương tiện bao gồm các giao thức khác như RTP, SDP .Tuy nhiên SIP không phụ thuộc vào các giao thức khác để làm việc .Địa chỉ SIP giống như một địa chỉ email, ví dụ như: [johndoe@sipA.com](mailto:johndoe@sipA.com). Các phương thức có trong SIP gồm có:

INVITE = Thiết lập phiên   
ACK = Xác nhận yêu cầu INVITE  
BYE = Kết thúc phiên  
CANCEL = Hủy bỏ việc thiết lập phiên  
REGISTER = Trao đổi thông tin địa điểm người dùng (tên máy, IP)  
OPTIONS = Trao đổi các thông tin về khả năng của các điện thoại SIP gọi và nghe trong phiên

Đáp úng cho các phương thức trên dưới dạng là text như trong HTML, có 6 loại Phản hồi SIP:

1xx = phản hồi thông tin, ví dụ 180, có nghĩa là đang đổ chuông  
2xx = phản hồi thành công  
3xx = phản hồi chuyển hướng  
4xx = yêu cầu bị thất bại  
5xx = lỗi máy chủ  
6xx = thất bại toàn cục

#### Giới thiệu về RTP

Giao thức truyền tín hiệu giọng nói, hình ảnh chuẩn được sử dụng trên mạng Internet hiện nay là giao thức RTP (Real-time transport protocol) .RTP được sử dụng rộng rãi trong các ứng dụng liên lạc và giải trí cần tới streaming dữ liệu

Tài liệu đặc tả của RTP mô tả hai giao thức con:

* Giao thức truyền dữ liệu, RTP, chịu trách nghiệm truyền dữ liệu thời gian thực. Thông tin cung cấp cho giao thức này bao gồm timestamps (để đồng bộ hóa), chuỗi số (cho kiểm tra gói tin bị mất) và tham số chỉ định dạng của dữ liệu.
* Một giao thức điều khiển, RTCP, dùng để tính toán các tham số trên đường truyền RTP để tối ưu đường truyền. Băng thông của RTCP tương đối nhỏ so với RTP, nằm trong khoảng 5%.

Có một vài chuẩn số hóa dữ liệu âm thanh và một trong số chúng được sử dụng chủ yếu trong VoIP .Đa số các nhà cung cấp dịch vụ và thiết bị hỗ trợ một hoặc nhiều các chuẩn đưa ra bởi ITU như dưới đây:

* G.711 là chuẩn mặc định cho tất cả các nhà cung cấp (dịch vụ và thiết bị), cũng như cho PSTN. Chuẩn này số hóa voice thành dữ liệu không nén có bitrate là 64 Kbps
* G.729 được hỗ trợ bởi nhiều nhà cung cấp cho phép nén dữ liệu âm ở bitrate 8 Kbps. Chất lượng của chuẩn này kém hơn so với G.711, nó cũng là chuẩn phổ biến thứ 2 sau G.711
* G.723.1 là chuẩn nén được khuyến cáo. Bitrate của dữ liệu theo chuẩn này là 6.3 và 5.3 kbps. Mặc dù chuẩn này làm giảm gắng nặng băng thông, nhưng chất lượng thấp hơn đáng kể so với G.729 và không được phổ biến trong VoIP.
* G.722 có bitrate 64kbps nhưng có độ trung thực về tiếng nói cao. Trái với ba chuẩn trước sử dụng tần số lấy mẫu vào khoảng 3.4 kHz, G.722 lấy mẫu với tần số 7 kHz. Chuẩn này sẽ được sử dụng phổ biến trong tương lai.

Trong mọi trường hợp, IP phone lấy dữ liệu số của 10 đến 30 ms và đặt nó trong gói tin RTP và truyền đi

### Đăng ký, thiết lập cuộc gọi với SIP proxy server

* + 1. **Đăng ký mới:**

Khi User muốn sử dụng IP phone để sử dụng dịch vụ VoIP họ phải thực hiện bước đăng ký địa chỉ sip của họ với một registrar server.Registrar server cung cấp các thông tin về user cho location server (location database) dựa trên các request đã nhận được từ user .SIP phone sử dụng phương thức REGISTER để register tới registrar server. Server nhận và lưu các thông tin trong request nhận được vào location database .Registrar server có thể chấp nhận các loại thông tin khác nhau, không chỉ địa chỉ IP của client .Database có chứa tất cả các ánh xạ giữa địa chỉ SIP tương ứng với địa chỉ IP của client, ví dụ: SIP user [8590@ekiga.net](mailto:8590@ekiga.net) đăng ký ở địa chỉ IP 200.180.1.1.

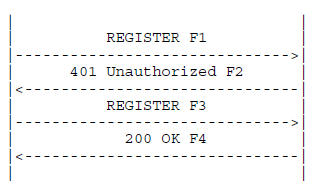


Figure 2: Đăng kí với Sip Server thành công

Caller gửi SIP REGISTER request tới SIP server .Request này bao gồm trong nó danh sách liên lạc (contact list) của user đó (danh sách các địa chỉ SIP của Caller). SIP server yêu cầu user nhập thông tin (user ID và password) để xác thực .SIP client mã hóa thông tin người dùng và gửi tới server .Server xác thực thông tin người dùng, nếu thành công nó trả về chuỗi 200 OK cho SIP client .Với cơ chế này client sử dụng giá trị nonce (Do server sinh ra và gửi cho client và là dãy số chỉ dùng 1 lần, tương tự như one time password) để tạo ra thông tin gửi lên server.

Thông tin gửi lên server gồm có checksum của username, password, giá trị nonce (username, cnonce, hash (nonce +cnonce + password)) và kết hợp với một số giá trị khác .Theo cách này, password của người dùng được gửi đi không bao giờ tồn tại dưới dạng clear text và cũng không thể giải mã được ra password

*F1 REGISTER sip: 192.168.133.193 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060; rport; branch=z9hG4bKBuZ0keykB*

*Max-Forwards: 70*

*To: <sip:1000@192.168.133.193>*

*From: <sip:1000@192.168.133.193>;tag=1mzIc5Cd*

*Call-ID: Ne4rmYWF-1295248430029@cn.vinhnk*

*CSeq: 1 REGISTER*

*Contact: sip:1000@192.168.133.193:6060;transport=UDP*

*F2 SIP/2.0 401 Unauthorized*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bKBuZ0keykB*

*To: <sip:1000@192.168.133.193>;tag=b27e1a1d33761e85846fc98f5f3a7e58.436b*

*From: <sip:1000@192.168.133.193>;tag=1mzIc5Cd*

*Call-ID: Ne4rmYWF-1295248430029@cn.vinhnk*

*CSeq: 1 REGISTER*

*WWW-Authenticate: Digest realm="192.168.133.193", nonce="TTPtWU0z7C2nKObujoCYR89YfUq3YDxC"*

*Server: kamailio (3.1.0 (i386/linux))*

*Content-Length: 0*

Server đòi username và password nó gửi về giá trị nonce="TTPtWU0z7C2nKObujoCYR89YfUq3YDxC"

*F3 REGISTER sip:192.168.133.193 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport;branch=z9hG4bK2LKuG1vwn*

*Max-Forwards: 70*

*To: <sip:1000@192.168.133.193>*

*From: <sip:1000@192.168.133.193>;tag=bYwlyuoF*

*Call-ID: Ne4rmYWF-1295248430029@cn.vinhnk*

*CSeq: 2 REGISTER*

*Contact: <sip:1000@192.168.133.193:6060;transport=UDP>*

*Authorization: Digest username="1000", realm="192.168.133.193", nonce="TTPtWU0z7C2nKObujoCYR89YfUq3YDxC", uri="sip:192.168.133.193", response="ba7b6b7988a263cba8de358cc3eac7e1"*

Client dùng MD5 tính checksum của password, nonce … rồi gửi response="ba7b6b7988a263cba8de358cc3eac7e1" trở lại server.

*F4 SIP/2.0 200 OK*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bK2LKuG1vwn*

*To: <sip:1000@192.168.133.193>;tag=b27e1a1d33761e85846fc98f5f3a7e58.8f80*

*From: <sip:1000@192.168.133.193>;tag=bYwlyuoF*

*Call-ID: Ne4rmYWF-1295248430029@cn.vinhnk*

*CSeq: 2 REGISTER*

*Contact: <sip:1000@192.168.133.193:6060;transport=UDP>;expires=3600*

*Server: kamailio (3.1.0 (i386/linux))*

*Content-Length: 0*

Khi user 1000@192.168.133.193 Log out khỏi server 192.168.133.193 nó tạo request:

*REGISTER sip:192.168.133.193 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport;branch=z9hG4bKZOmdS4hLG*

*Max-Forwards: 70*

*To: <sip:1000@192.168.133.193>*

*From: <sip:1000@192.168.133.193>;tag=FLjcVEm6*

*Call-ID: Ne4rmYWF-1295248430029@cn.vinhnk*

*CSeq: 4 REGISTER*

*Contact: <sip:1000@192.168.133.193:6060;transport=UDP>;expires=0*

*Authorization: Digest username="1000", realm="192.168.133.193", nonce="TTPt+k0z7M6EsW/CXuH6wCl9ZHF244HG", uri="sip:192.168.133.193", response="d18e2c819303f864d9b09280015267df"*

Sau đó server trả về thành công:

*SIP/2.0 200 OK*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bKZOmdS4hLG*

*To: <sip:1000@192.168.133.193>;tag=b27e1a1d33761e85846fc98f5f3a7e58.d2d1*

*From: <sip:1000@192.168.133.193>;tag=FLjcVEm6*

*Call-ID: Ne4rmYWF-1295248430029@cn.vinhnk*

*CSeq: 4 REGISTER*

*Server: kamailio (3.1.0 (i386/linux))*

*Content-Length: 0*

* + 1. **Thiết lập cuộc gọi**

Khi user thiết lập một cuộc gọi, một SIP request được gửi tới SIP server (một proxy server hoặc một redirect server) .Trong request này bao gồm địa chỉ của người gọi (trường From) và địa chỉ của người nhận (trường To).Khác với phương thức REGISTER, phương thức INVITE được sử dụng để thiết lập các media session giữa các user agent .Một media session được thiết lập khi các bản tin INVITE ,200 OK, ACK được trao đổi giữa UAC và UAS .Phương thức BYE được sử dụng bởi một trong các UA để kết thúc media session đã được thiết lập.

UAC mà khởi tạo một INVITE request sẽ tạo ra một Call-ID duy nhất, sử dụng cho việc xác định thời gian của cuộc gọi. Biến đếm CSeq được khởi tạo (không cần giá trị phải bằng 1, nhưng phải là một số nguyên) và tăng khi có một request mới từ cùng một Call-ID. Trường To và From trong header của request là địa chỉ SIP của người nhận và người gọi. Trường Expires trong header của một INVITE request cho UAS biết request này có hiệu lực trong khoảng thời gian bao lâu .Một khi media session được thiết lập thì trường này sẽ không còn ý nghĩa nữa .Trường Session-Expires có thể được sử dụng để giới hạn thời gian của một session.



Figure 3: Thiết lập thành công cuộc gọi

Trong phần này sẽ đi chi tiết về việc thiết lập cuộc gọi (thiết lập sesssion) giữa hai SIP UA: 1000@192.168.133.193 (Alice) và 1001@192.168.133.193 (Bob) .Sự trao đổi thông tin về media của UA được đặt trong SDP payloads .Trong phần này, cuộc gọi được thiết lập và cuối cùng yêu cầu kết thúc cuộc gọi đến từ phía 1000.

*F1 INVITE sip:1001@192.168.133.193 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport;branch=z9hG4bKrybFW32gj*

*Max-Forwards: 70*

*To: <sip:**1001@192.168.133.193>*

*From: <sip:1000@192.168.133.193>;tag=mXuZXR5C*

*Call-ID: bdpjEqZN-1295245265337@cn.vinhnk*

*CSeq: 4 INVITE*

*Content-Length: 218*

*Content-Type: application/sdp*

*Contact: <sip:1000@192.168.133.193:6060;transport=UDP>*

*Proxy-Authorization: Digest username="1000", realm="192.168.133.193", nonce="TTPg/E0z39DX2sC/XzRzl8PQlpKPQDlQ", uri="sip:1001@192.168.133.193", response="3e671b21bfe90c3e9c94ec63c229041c"*

*v=0*

*o=user1 1782520463 2100547947 IN IP4 192.168.133.193*

*s=-*

*c=IN IP4 192.168.133.193*

*t=0 0*

*m=audio 8000 RTP/AVP 0 8 101*

*a=rtpmap:0 PCMU/8000*

*a=rtpmap:8 PCMA/8000*

*a=rtpmap:101 telephone-event/8000*

*a=sendrecv*

Nội dung bản tin SDP (từ đoạn v=0) ở request này cho biết. SIP phone của 1000 sẽ nhận RTP audio có loại codec là 0, 8, 101 ở 192.168.133.193: 8000

*F2 SIP/2.0 180 Ringing*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bKrybFW32gj*

*Record-Route: <sip:192.168.133.193;lr>*

*Contact: <sip:1001@192.168.133.196:32322;rinstance=2f19f079276fe5f9>*

*To: <sip:1001@192.168.133.193>;tag=ca470244*

*From: <sip:1000@192.168.133.193>;tag=mXuZXR5C*

*Call-ID: bdpjEqZN-1295245265337@cn.vinhnk*

*CSeq: 4 INVITE*

*User-Agent: X-Lite release 1002tx stamp 29712*

*Content-Length: 0*

* + - 1. **Chấp nhận cuộc gọi**

1001@192.168.133.193 trả lời:

*F3 SIP/2.0 200 OK*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bKrybFW32gj*

*Record-Route: <sip:192.168.133.193;lr>*

*Contact: <sip:1001@192.168.133.196:32322;rinstance=2f19f079276fe5f9>*

*To: <sip:1001@192.168.133.193>;tag=ca470244*

*From: <sip:1000@192.168.133.193>;tag=mXuZXR5C*

*Call-ID: bdpjEqZN-1295245265337@cn.vinhnk*

*CSeq: 4 INVITE*

*Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO*

*Content-Type: application/sdp*

*User-Agent: X-Lite release 1002tx stamp 29712*

*Content-Length: 247*

*v=0*

*o=- 6 2 IN IP4 192.168.133.196*

*s=<CounterPath eyeBeam 1.5>*

*c=IN IP4 192.168.133.196*

*t=0 0*

*m=audio 52062 RTP/AVP 0 8 101*

*a=fmtp:101 0-15*

*a=rtpmap:101 telephone-event/8000*

*a=sendrecv*

*a=x-rtp-session-id: 9BF1CA1AD44A4BBDB2F89BE1533DFFB0*

Nội dung bản tin SDP (từ đoạn v=0) ở response trên cho biết. SIP phone của 1001 sẽ nhận RTP audio có loại codec là 0, 8, 101 ở 192.168.133.196: 52062

*F4 ACK sip:1001@192.168.133.196:32322;rinstance=2f19f079276fe5f9 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport;branch=z9hG4bKrybFW32gj*

*To: <sip:1001@192.168.133.193>;tag=ca470244*

*From: <sip:1000@192.168.133.193>;tag=mXuZXR5C*

*Call-ID: bdpjEqZN-1295245265337@cn.vinhnk*

*CSeq: 4 ACK*

*Route: <sip:192.168.133.193;lr>*

*Max-Forwards: 70*

Sau khi thực hiện xong bước 4 .RTP stream được thiết lập giữa 1000 và 1001 .Khi 1000 thực hiện gác máy, BYE được tạo ra bởi 1000

*F5 BYE sip:1001@192.168.133.196:32322;rinstance=2f19f079276fe5f9 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport;branch=z9hG4bK0mvp65wxR*

*To: <sip:1001@192.168.133.193>;tag=ca470244*

*From: <sip:1000@192.168.133.193>;tag=mXuZXR5C*

*Call-ID: bdpjEqZN-1295245265337@cn.vinhnk*

*CSeq: 6 BYE*

*Route: <sip:192.168.133.193;lr>*

*Max-Forwards: 70*

*F6 SIP/2.0 200 OK*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bK0mvp65wxR*

*Contact: <sip:1001@192.168.133.196:32322;rinstance=2f19f079276fe5f9>*

*To: <sip:1001@192.168.133.193>;tag=ca470244*

*From: <sip:1000@192.168.133.193>;tag=mXuZXR5C*

*Call-ID: bdpjEqZN-1295245265337@cn.vinhnk*

*CSeq: 6 BYE*

*User-Agent: X-Lite release 1002tx stamp 29712*

*Content-Length: 0*

* + - 1. **Từ chối cuộc gọi**

1001@192.168.133.193 trả lời như sau

*SIP/2.0 480 Temporarily Unavailable*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport=6060;branch=z9hG4bKRdwKIwYy5*

*To: <sip:1001@192.168.133.193>;tag=6b254d3c*

*From: <sip:1000@192.168.133.193>;tag=mVBPDX6W*

*Call-ID: nFthyBSf-1295248052847@cn.vinhnk*

*CSeq: 4 INVITE*

*User-Agent: X-Lite release 1002tx stamp 29712*

*Content-Length: 0*

1000@192.168.133.193 xác nhận lại:

*ACK sip:1001@192.168.133.193 SIP/2.0*

*Via: SIP/2.0/UDP 192.168.133.193:6060;rport;branch=z9hG4bKRdwKIwYy5*

*Call-ID: nFthyBSf-1295248052847@cn.vinhnk*

*From: <sip:1000@192.168.133.193>;tag=mVBPDX6W*

*To: <sip:1001@192.168.133.193>;tag=6b254d3c*

*CSeq: 4 ACK*

* 1. **Cơ bản về Android**

Việc hiểu được các thành phần (component) tạo nên một ứng dụng Android là rất cần thiết cho việc lập trình trên Android .

* + 1. **Activity*:***

Hiểu một cách đơn giản thì Activity là nền của một ứng dụng. Khi khởi động một ứng dụng Android nào đó thì bao giờ cũng có một main Activity được gọi, hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác .Hệ điều hành Android quản lý Activity theo dạng stack: khi một Activity mới được khởi tạo, nó sẽ được xếp lên đầu của stack và trở thành *running activity*, các Activity trước đó sẽ bị tạm dừng và chỉ hoạt động trở lại khi Activity mới được giải phóng.  
  
Activity bao gồm bốn state:  
- *active (running):* Activity đang hiển thị trên màn hình (foreground).  
- *paused:* Activity vẫn hiển thị (visible) nhưng không thể tương tác (lost focus).   
- *stop:* Activity bị thay thế hoàn toàn bởi Activity mới sẽ tiến đến trạng thái *stop*  
- *killed:* Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên. Các Activity ở trạng thái *stop* hoặc *paused* cũng có thể bị giải phóng và khi nó được hiển thị lại thì các Activity này phải khởi động lại hoàn toàn và phục hồi lại trạng thái trước đó.

* + 1. ***Service:***

Thành phần chạy ẩn trong Android, service sử dụng để update dữ liệu, đưa ra các cảnh báo (Notification) và không bao giờ hiển thị cho người dùng thấy .Nếu một Service cho phép những tiến trình khác kết nối với nó thì sẽ có thêm các phương thức callback dành cho Service đó để thực hiện

* + 1. **Intent*:***
       1. **Khái niệm về Intent:**

Nền tảng để truyền tải các thông báo, intent được sử dụng để gửi các thông báo đi nhằm khởi tạo một Activity hay Service để thực hiện công việc bạn mong muốn. VD: khi mở trang web, bạn gửi một intent đi để tạo activity mới hiển thị trang web đó. Intent được chia làm 2 loại:   
- Explicit Intents: intent đã được xác định thuộc tính component, nghĩa là đã chỉ rõ thành phần sẽ nhận và xử lý intent. Thông thường intent dạng này sẽ không bổ sung thêm các thuộc tính khác như action, data .Explicit Intent thương được sử dụng để khởi chạy các activity trong cùng 1 ứng dụng.  
- Implicit Intents**:** Intent không chỉ rõ component xử lý, thay vào đó nó bổ sung thông tin trong các thuộc tính .Khi intent được gửi đi, hệ thống sẽ dựa vào những thông tin này để quyết định component nào thích hợp nhất để xử lý nó.

* + - 1. Intent Filter là gì

Activity, Service và BroadCast Receiver sử dụng Intent Filter để thông báo cho hệ thống biết các dạng Implicit Intent mà nó có thể xử lý .Nói cách khác, Intent Filter là bộ lọc Intent, chỉ cho những Intent được phép đi qua nó  
Intent Filter mô tả khả năng của component định nghĩa nó .Khi hệ thống bắt được một Implicit Intent (chỉ chứa một số thông tin chung chung về action, data và category...), nó sẽ sử dụng những thông tin trong Intent này, kiểm tra đối chiếu với Intent Filter của các component các ứng dụng, sau đó quyết định khởi chạy ứng dụng nào thích hợp nhất để xử lý Intent bắt được. Nếu có hai hay nhiều hơn ứng dụng thích hợp, người dùng sẽ được lựa chọn ứng dụng mình muốn.

* + 1. ***Broadcast Receiver:***

Thành phần thu nhận các Intent bên ngoài gửi tới. VD: bạn viết một chương trình thay thế cho phần gọi điện mặc định của android, khi đó bạn cần một *Broadcast Receiver* để nhận biết các Intent là các cuộc gọi tới.

* + 1. ***Notification:***

Dưa ra các cảnh báo mà không làm cho các Activity phải ngừng hoạt động

* + 1. **SQLLite**

Mỗi ứng dụng đều sử dụng dữ liệu, dữ liệu có thể đơn giản hay đôi khi là cả 1 cấu trúc .Trong Android thì hệ cơ sở dữ liệu được sử dụng là SQLite Database, đây là hệ thống mã nguồn mở được sử dụng rộng rãi trong các ứng dụng (Mozilla Firefox sử dụng SQLite để lưu trữ các dữ liệu về cấu hình, iPhone cũng sử dụng cơ sở dữ liệu là SQLite)   
Trong Android, cơ sở dữ liệu mà bạn tạo cho 1 ứng dụng thì chỉ ứng dụng đó có quyền truy cập và sử dụng, các ứng dụng khác thì không. Khi đã được tạo, cơ sở dữ liệu SQLite được chứa trong thư mục */data/data/<package\_name>/databases*.

* + 1. **XML trong Android:**

Sử dụng XML sẽ đơn giản công việc thiết kế giao diện đi rất nhiều .Đồng thời sử dụng XML sẽ giúp việc chỉnh sửa ứng dụng sau này trở nên dễ dàng

* + 1. **Android Manifest:**

Activity, Service, Broadcast Receiver là những thành phần chính cấu thành nên ứng dụng Android, bắt buộc phải khai báo trong AndroidManifest .Mỗi ứng dụng đều cần có AndroidManifest.xml để mô tả những thông tin quan trọng của nó cho hệ thống Android biết.

1. **Các vấn đề đặt ra**

### *Xác thực dữ liệu*

Như ta đã biết các bản tin SIP được truyền đi trong mạng là dạng clear text. Do đó nếu không có cơ chế bảo vệ thì thông tin về username và password của toàn bộ người dùng sẽ bị phơi bày

* 1. ***NAT và ảnh hưởng của nó tới VoIP***

Internet ngày càng được sử dụng rộng rãi, các dịch vụ web, tương tác trực tuyến ngày càng phát triển, số lượng máy tính kết nối Internet ngày càng nhiều dẫn đến số lượng địa chỉ IPv4 thiếu hụt. IPv6 ra đời giúp tăng thêm số lượng địa chỉ IP tuy nhiên còn phải mất thêm nhiều thời gian để được sử dụng rộng rãi .Việc dùng NAT có nhiều ưu điểm như giúp giải quyết thiếu hụt IPv4, bảo vệ các máy tính trong mạng nội bộ tránh sự xâm nhập trái phép từ Internet và nhiều ưu điểm khác .Tuy nhiên NAT phá vỡ các giao tiếp trong VoIP

1. **Giải pháp và công cụ được lựa chọn để giải quyết các vấn đề đặt ra**
   1. **Giải pháp xác thực dữ liệu**

SIP digest authentication dựa trên digest authentication được định nghĩa ở RFC 2617. Với cơ chế này client sử dụng giá trị nonce (Do server sinh ra và gửi cho client và là dãy số chỉ dùng 1 lần, tương tự như one time password) để tạo ra thông tin gửi lên server

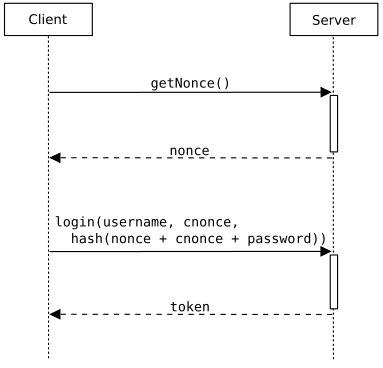


Figure 4: Cơ chế xác thực username và password

Thông tin gửi lên server gồm có checksum của *username*, *password*, giá trị *nonce* và kết hợp với một số giá trị khác .Theo cách này, password của người dùng được gửi đi không bao giờ tồn tại dưới dạng clear text và cũng không thể giải mã được ra password

### *Giải pháp cho các thiết bị nằm trong NAT*

* + 1. **Giải pháp qua NAT cho các tín hiệu SIP**

Vì các User Agent phải kết nối, gửi các SIP request tới một SIP server chạy trên một địa chỉ có ip và port cố định .Do vậy phía server sẽ ghi nhận được ip: port mặt ngoài của thiết bị kết nối tới nó, SIP server sẽ dùng thông tin này để gửi lại các SIP response như hình dưới:

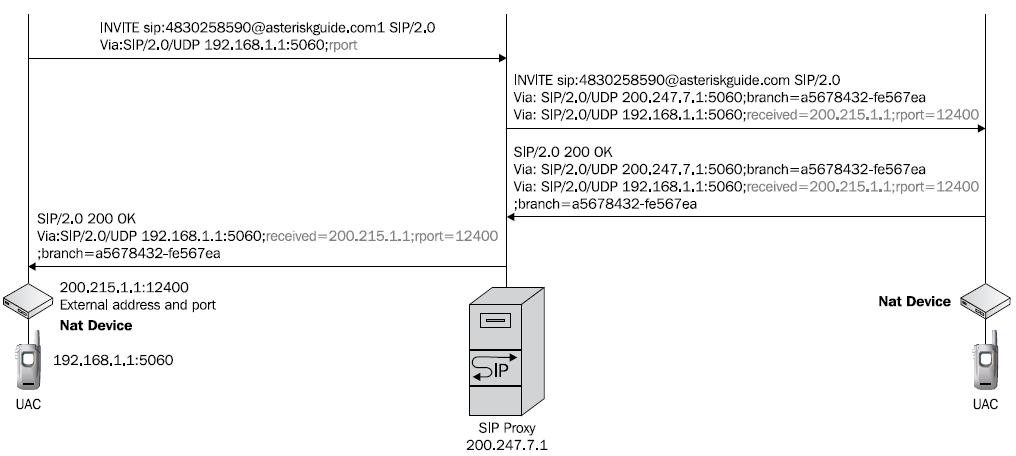


Figure 5 : Giải pháp qua NAT cho các tín hiệu SIP

* + 1. **Giải pháp truyền các gói tin RTP qua NAT**

Sử dụng STUN để xác định trước cặp ip:port của thiết bị NAT, gửi cặp này cho user agent kia, để nó có thể kết nối tới:

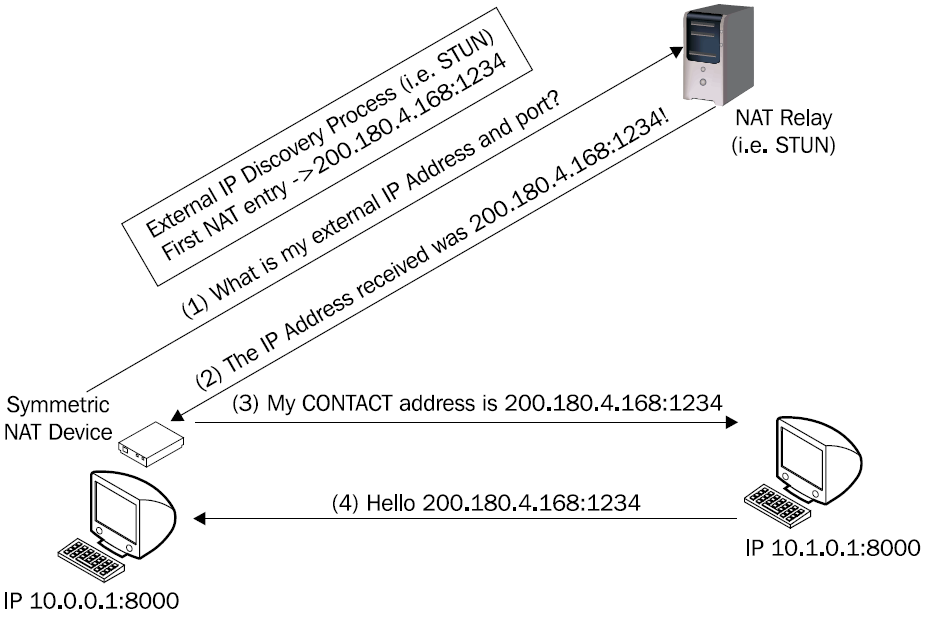


Figure 6: Giải pháp truyền các gói tin RTP qua NAT

Tuy nhiên với giải pháp này, ta không thể đảm bảo 100% các user agent có thể giao tiếp với nhau vì nếu user agent ở dưới một symmetric nat thì dùng STUN sẽ không giải quyết được vấn đề.

### *Chọn lựa chuẩn xây dựng hệ thống VoIP*

SIP hiện đang thể hiện là chuẩn được dùng và sẽ dùng phổ biến trong tương lai bởi sự đơn giản và hỗ trợ tương tác tốt giữa các hệ thống khác nhau. Các bản tin SIP được thể hiện dưới dạng text base và truyền đi trong mạng IP tương tự như HTTP. Ta có thể phát triển và xây dựng dịch vụ voice dựa trên SIP dễ và nhanh như phát triển các web pages .Nó không yêu cầu các hạ tầng mạng đặt biệc mà tận dụng các thiết bị, server của mạng IP có sẵn .Hơn nữa ngày nay đa số các thiết bị IP phone đều hỗ trợ SIP. Do vậy ta sẽ sử dụng SIP như là một giải pháp cho signaling protocol.

Để truyền tải và số hóa dữ liệu ta sử dụng giao thức RTP, số hóa âm thanh dùng codec theo chuẩn G.711 (cho chất lượng) và G.729 (cho việc cân bằng giữa chất lượng và băng thông) vì 2 codec trên cũng được hỗ trợ bởi đa số các IP phone ngày nay.

### *Lựa chọn thư viện phát triển ứng dụng SIP trên client*

Môi trường phía client được lựa chọn để xây dựng demo là Android OS (android API 8).

PJSIP là thư viện có thể sử dụng trên nhiều nền tảng, thư viện kích thước nhỏ (small footprint), Tích hợp sẵn các gói thư viện media, NAT để xây dựng ứng SIP client.

PJSIP là thư viện viết bằng C có hiệu năng cao, ổn định, hỗ trợ nhiều nền tảng và hiện tại đang được phát triển mạnh mẽ. Ngoài ra, thư viện đi kèm PJSIP đã được đóng gói sẵn các thư viện xử lý media, NAT. Với những ưu điểm này, ta sử dụng PJSIP để phát triển ứng dụng demo phía client.

### *Lựa chọn SIP proxy*

SIP server là thành phần quan trọng trong hệ thống VoIP, cho phép các user agent trao đổi thông tin, đăng ký người dùng và trong suốt với hạ tầng mạng giữa nhiều người dùng .SIP server còn cung cấp các dịch vụ quan trọng khác như định tuyến cuộc gọi, các chính sách bảo mật, xác thực người dùng và quản lý trạng thái user agent .SIP server cần phải làm việc trên nhiều topo mạng (như các mạng công cộng, mạng di động, 3G…). SIP server cần có độ sẵn sàng cao, làm việc với nhiều giao dịch đồng thời và cần có khả năng mở rộng.

Kamailio (tên gọi cũ OpenCER) là hệ thống open source dùng để triển khai sip server .Một SIP proxy xây dựng từ Kamailio có khả năng quản lý hàng nghìn cuộc gọi mỗi giây .Kamailio được duy trì và phát triển bởi một cộng đồng lớn các lập trình viên có kinh nghiệm, ngày càng hỗ trợ kết nối tới nhiều hệ thống viễn thông khác.

# PHẦN II: XÂY DỰNG HỆ THỐNG VOIP

1. Khảo sát, phân tích yêu cầu
   1. Mục tiêu của hệ thống

Xây dựng hệ thống để thử nghiệm tính khả thi của công nghệ Voip phục vụ cho các dự án tiếp theo của Trung tâm phần mềm Viettel.

* 1. Phân tích yêu cầu
     1. **Các ràng buộc thiết kế (Design Constraints)**

Sử dụng ngôn ngữ Java

Server chạy nền tảng UNIX

Công cụ phát triển là Eclipse tích hợp Android SDK

* + 1. Các yêu cầu về hiệu năng (Performance)

Đảm bảo nhiều người sử dụng đồng thời

Chất lượng cuộc thoại trong phạm vi cho phép

* 1. Thiết kế tổng quan
     1. Thiết kế kiến trúc hệ thống

Hệ thống Voip được bao gồm hai thanh phần chính: Server và Client

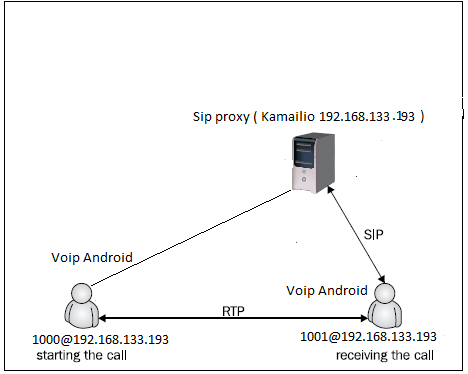


Figure 7: Mô hình hệ thống cần xây dựng

Trong các phần trên ta đã nêu ra được các thành phần cần thiết mà hệ thống VoIP phải đáp ứng .Chúng ta có các user agent và server .Hệ thống sử dụng một mô hình phân tán peer-to-peer cùng với một signaling server

* + - 1. **Client**

Client trong hệ thống là ứng dụng Voip Android được gọi là các User Agent (UA) .Voip Adroid có các vai trò là:

* User Agent Client (UAC) – Khi mà nó khởi tạo các SIP request (start SIP signaling) tới server
* User Agent Server (UAS) – Khi nó trả lời SIP signaling đến từ các UAC
  + - 1. **Server**

Server được lựa chọn là Kamailio .Có hai chức năng chính làm proxy server và registrar server.Registrar server cung cấp các thông tin về user cho location server (location database) dựa trên các request đã nhận được từ user

* 1. **Thực hiện, cài đặt và thử nghiệm hệ thống**

#### Triển khai phía server:

* + - 1. **Cài đặt**

Ta có thể sử dụng các nhà cung cấp dich vụ miễn phí như ekiga.net , iptel.org …hoặc cũng có thể tự cài cho riêng mình một sip server .

Kamailio (trước kia là OpenSER) là một SIP server mã nguồn mở với khả năng thiết lập hàng nghìn cuộc gọi trong một giây .Kamailio được sử dụng để xây dựng dịch vụ cỡ VoIP lớn

Nội dung phần này tập trung vào việc hướng dẫn cài đặt Kamailio v3.1.0 có hỗ trợ MySQL trên hệ thống Debian.

* Yêu cầu:

Có quyền root với hệ thống. Và các công cụ sau đây là cần thiết cho các bước tiếp theo:

* + *git* client: apt-get install git-core: Sử dụng để lấy mã nguồn mới nhất của kamailio
  + *gcc* compiler: apt-get install gcc
  + *flex* : apt-get install flex
  + *bison* : apt-get install bison
  + *libmysqlclient15-dev* : apt-get install libmysqlclient15-dev
  + *make* : apt-get install make
  + Nếu như muốn sử dụng thêm nhiều module của Kamailio hơn thì các thư viện dưới đây là cần thiết:
    - libcurl : apt-get install libcurl4-openssl-dev
    - libxml2 : apt-get install libxml2-dev
    - libpcre3 : apt-get install libpcre3-dev
* Lấy mã nguồn mới nhất của Kamailio từ GIT

Tạo thư mục chứa mã nguồn:

* mkdir -p /usr/local/src/kamailio-3.1.0
* cd /usr/local/src/kamailio-3.1.0

Download mã nguồn sử dụng lệnh git:

* git clone --depth 1 git://git.sip-router.org/sip-router kamailio
* cd kamailio
* git checkout -b 3.1 origin/3.1
* Sửa Makefiles

Bước quan trọng đầu tiên cầu thực hiện đó là set biết flavour cho Kamailio:

* make FLAVOUR=kamailio cfg

Cho phép module MySQL hoạt động:

* nano -w modules.lst

Gỡ bỏ **db\_mysql khỏi biến exclude\_modules Lưu file và tắt nano**

Hai bước trên có thể thực hiện bằng cách đơn giản hơn:

* make FLAVOUR=kamailio include\_modules="db\_mysql dialplan" cfg

Nếu ta muốn cài đặt mọi thứ của kamailio vào thư mục /usr/local/kamailio-3.1 thì:

* make PREFIX="/usr/local/kamailio-3.1" FLAVOUR=kamailio include\_modules="db\_mysql dialplan" cfg
* Biên dịch Kamailio

Dùng lệnh make để thực hiện biên dịch

* make all
* Cài đặt Kamailio

Lệnh để cài đặt

* make install
* Vị trí các file được cài đặt trên hệ thống

Kamailio sau khi được biên dịch và cài đặt thì các file của nó được đặt tại:

* /usr/local/sbin

Trong thư mục này có:

* + kamailio – Kamailio (OpenSER) server
  + kamdbctl – Đoạn script để tạo và quản lý database
  + kamctl – Đoạn script để quản lý và điều khiển Kamailio (OpenSER) server
  + sercmd - CLI - command line tool để tương tác với Kamailio (OpenSER) server

Thêm đường dẫn '/usr/local/sbin' vào biến môi trường PATH . Ta có thể kiểm tra bằng lệnh 'echo $PATH'. Nếu chưa thêm thì mởi file /root/.bash\_profile và thêm vào cuối file:

* PATH=$PATH:/usr/local/sbin
* export PATH

Các modules của kamailio được cài đặt ở:

* /usr/local/lib/kamailio/modules/
* /usr/local/lib/kamailio/modules\_k/

File tài liệu và readme file ở:

* /usr/local/share/doc/kamailio/

File cấu hình:

* /usr/local/etc/kamailio/kamailio.cfg

Hướng dẫn sử dụng:

* /usr/local/share/man/man5/
* /usr/local/share/man/man8/
* Tạo MySQL database

Để tạo MySQL database, ta chạy đoạn script sau:

* /usr/local/sbin/kamdbctl create
* Cấu hình Kamailio

Cấu hình trong file dưới đây để có thể chạy được các modules cần thiết.

* /usr/local/etc/kamailio/kamailio.cfg

Để chạy MySQL thêm các dòng dưới đây vào đầu file:

* #!define WITH\_MYSQL
* #!define WITH\_AUTH
* #!define WITH\_USRLOCDB
* Start, stop, restart server:

Để start, stop server chạy các lệnh dưới đây:

* kamctl start
* kamctl stop
* kamctl restart
* Thêm user vào server

Thêm user ‘1000’ với password là ‘123456’

* kamctl add 1000 123456

Nếu được hỏi về biến môi trường SIP\_DOMAIN thì thực hiện lệnh sau:

* export SIP\_DOMAIN=mysipserver.com

Sau khi thực hiện các bước trên, ta có thể dùng sip phone và đăng nhập, thực hiện cuộc gọi tới user khác.

* + 1. **Phía Client**
       1. **Build thư viện pjsip cho ứng dụng Voip Android**

Đầu tiên download ndk và giải nén Android Ndk

* wget http://www.crystax.net/data/android-ndk-r4-linux-x86-crystax-4.tar.bz2tar -xvf android-ndk-r4-linux-x86-crystax-4.tar.bz2rm android-ndk-r4-linux-x86-crystax-4.tar.bz2

Download thư viện pjsip:

* http://www.pjsip.org/download.htm/ chọn pjsip\_android

Copy thư mục prebuilt trong android-ndk-r4-crystax/build vào thư mục pjsip\_android/build/

* cp -r android-ndk-r4-crystax/build/prebuilt pjsip\_android/build/

Build file thư viện liên kết động .so, vào thư mục pjsip\_android

* cd pjsip\_android

Gọi lệnh make

* make APP=pjsip

Sau khi xong các bước trên file libpjsipjni.so sẽ được tạo ra trong thư mục /VoipAndroid/libs/ armeabi & armeabi-v7a

* + - 1. **Xây dựng ứng dụng Voip Android phía client**

Voip Android là một ứng dụng trên điện thoại chạy hệ điều hành Android cho phép người dùng có thể tạo cuộc gọi, gửi tin nhắn thông qua hạ tầng mạng IP có sẵn có thể là wifi hoặc 3G .Voip Android được viết bằng thư viện PJSIP, cách build PJSIP cho nền tảng Android đã được trình bày ở trên.

###### Phân tích cụ thể từng chức năng:

Các chức năng của hệ thống được trình bày như biểu đồ dưới đây:



Figure 8: Mô hình Use case của ứng dụng Voip Android

Quản lí các tài khoản:

* Thêm xóa chỉnh sửa Account:
* Register / Unregister một Account với sip server

Quản lí việc nhận và gửi tin nhắn

* Gửi tin nhắn (instant message)
* Quản lí lưu trữ tin nhắn

Quản lí cuộc gọi

* Thiết lập cuộc gọi đến một địa chỉ URI xác định
* Quản lí lưu trữ calllogs
  + - * 1. **Các luồng thông tin trong ứng dụng Voip Android**

Hình dưới đây minh họa luồng thông tin được xử lí bên trong hệ thống

Như hình dưới ta thấy, người dùng thông qua giao diện gửi yêu cầu tới ứng dụng, tùy theo loại yêu cầu ứng dụng xác định xem có cần truy nhập cơ sở dữ liệu hay có cần kết nối tới SipService không .Việc kết nối vào SipService thông qua đối tượng ISipservice .SipService có trách nhiệm tương tác với sip server thông qua bản tin sip, các softphone khác thông qua bản tin RTP .

* User interface: Giao diện người dùng , các màn hình như: Tạo cuộc gọi, tin nhắn , xem lịch sử cuộc gọi, thêm tài khoản …
* Database: Dùng SQLLite để quản lí các bảng như: tài khoản , tin nhắn , calllogs
* SipService: là nơi thực sự xử lí các yêu cầu người dùng, nó chạy ngầm trong ứng dụng, tương tác với các sip client và sip server, cung cấp các hàm api cho các thành phần khác kết nối đến.
* ISipService: nó đóng vai trò giống như cầu nối để các activity kết nối tới SipService

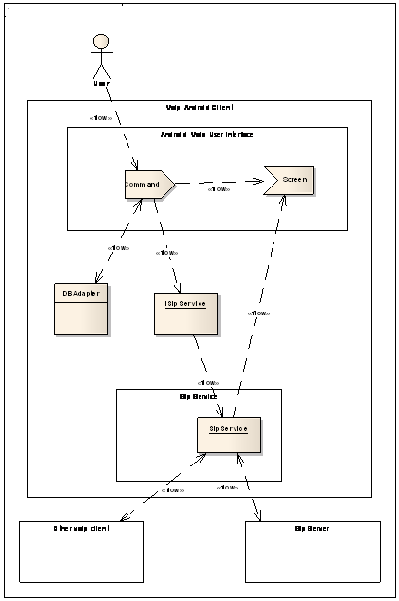


Figure 9: Luồng thông tin trong ứng dụng Voip Android

* + - * 1. **Thiết kế cơ sở dữ liệu**

Trong Voip Android tên của database là: “com.voip.db”. Class DBAdapter trong package utils để xử lý tất cả các thao tác liên quan đến CSDL .Có 4 bảng là: Account, CallLog, and Message. Cấu trúc của các bảng như sau:

* 1. **Bảng Account:**

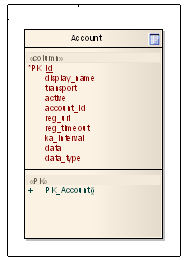


Table 1: Bảng Account

Ý nghĩa của các trường được mô tả bên dưới (Class IAccount)

Các thao tác trên bảng Account:

* Xóa một account trong bảng, hàm này phục vụ cho chức năng xóa một tài khoản
* Cập nhật môt account ,hàm này dành cho chức năng chỉnh sửa tài khoản
* Lấy ra một danh sách các account theo một tiêu chí nào đó VD: chỉ lấy ra các account đã từng được đăng kí (active)
  1. **Bảng CallLogs**

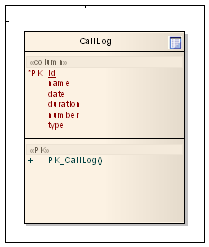


Table 2: Bảng CallLogs

Ý nghĩa các trường:

* name: Tên người thực hiện cuộc gọi
* date: Ngày giờ cuộc gọi diễn ra
* duration: thời gian đàm thoại
* number: sip uri của người gọi
* type: loại cuộc gọi (cuộc gọi đã nhận , cuộc gọi nhỡ ,cuộc gọi đi)

Các thao tác trên bảng:

* Thêm một cuộc gọi vào bảng , dùng để cập nhật khi thực hiện một cuộc gọi hoặc có cuộc gọi đến hoặc cuộc gọi nhỡ
* Lấy ra các cuộc gọi có trong bảng dùng cho chức năng xem lịch sử các cuộc trong quá khứ
* Xóa cuộc gọi được dùng đến khi người dùng thực hiện thao tác xóa một hay nhiều lựa chon
  1. **Bảng Messages:**

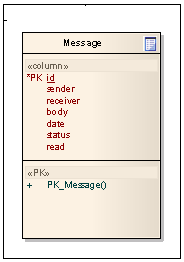


Table 3: Bảng Messages

Ý nghĩa của các trường:

* sender: sip uri của người gửi
* receiver: sip uri của người nhận
* body: Nội dung của tin nhắn
* date: Ngày gửi hoặc nhận
* status: Trạng thái của tin nhắn (đã gửi thành công, thất bại)
* read: Tin nhắn đã được đọc hay chưa

Các thao tác trên bảng:

* Thêm một tin nhắn vào bảng ,được dùng khi có tin nhắn mới đến hoặc khi người dùng tự họ gửi
* Lấy ra các tin nhắn trong bảng dùng cho chức năng xem các tin nhắn có trong máy
* Xóa một tin nhắn trong bảng dùng cho chức năng xóa một tin nhắn.
* Cập nhật trạng thái của tin nhắn: đã đọc, gửi thành công hay thất bại .Trạng thái của tin nhắn được cập nhật dựa vào thư viện PJSIP
  + - * 1. **Thiết kế giải thuật**

Hình dưới đây minh họa tuần tự của người dùng với hệ thống ứng với mỗi chức năng

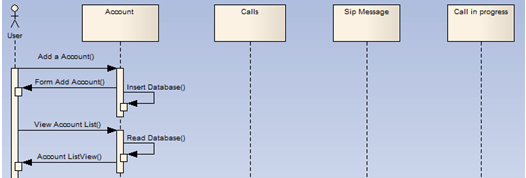


Figure 9: Biểu đồ tuần tự cho chức năng thêm và xem các account

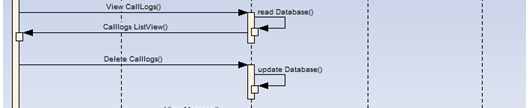


Figure 10: Biểu đồ tuần tự cho chức năng xem và xóa lịch sử cuộc gọi

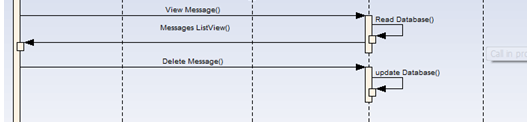


Figure 11 : Biểu đồ tuần tự cho chức năng xem và xóa tin nhắn

Các chức năng xem, xóa, sửa đổi thông tin tài khoản, calllogs, danh sách các tin nhắn có thể được thực hiện mà không cần có tài khoản đã đăng kí thành công với server .Riêng các chức năng gửi tin nhắn, thiết lập cuộc gọi bắt buộc có ít nhất một tài khoản đã đăng kí thành công với server .Việc đăng kí tài khoản mới do giới hạn của đồ án nên việc này phải đăng kí qua giao diện web của nhà cung cấp dịch vụ VD: ekiga.net. hoặc được tạo ra khi cài đặt Sip Server .Ứng với mỗi thao tác của người dùng hệ thống đều cập nhật vào Database để tiện cho việc quản lí .Cơ chế quản lí cơ sở dữ liệu của Android sẽ được đề cập chi tiết ở phần sau.

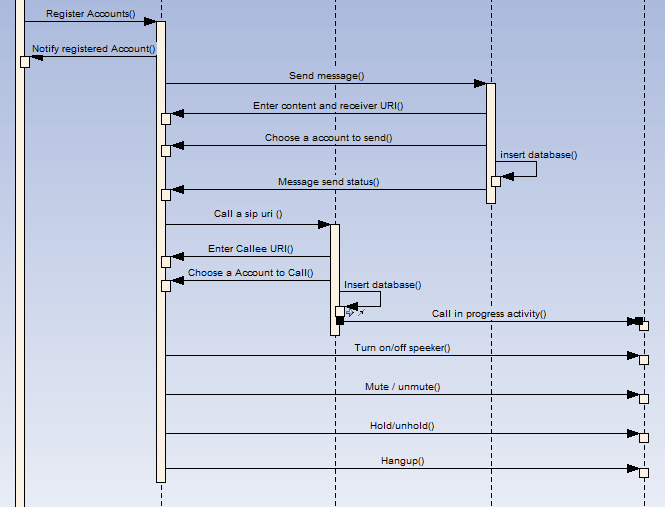


Figure 12: : Biểu đồ tuần tự cho chức năng gửi tin nhắn và tạo cuộc gọi

Đối với trường hợp lúc người dùng đang đàm thoại họ có thể lựa chọn các chức năng giống như cuộc gọi thông thường với mạng GSM .Có thể bật hoặc tắt loa, mute hoặc unmute, hold hoặc unhold, cúp máy.

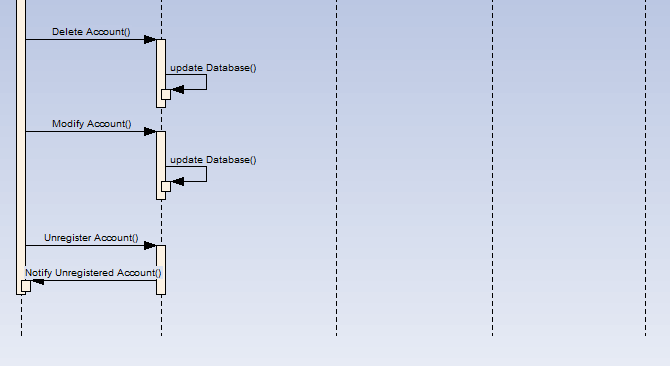


Figure 13 : Biểu đồ tuần tự cho chức năng xóa và sửa tài khoản

* + - * 1. **Kiến trúc ứng dụng Voip Android**

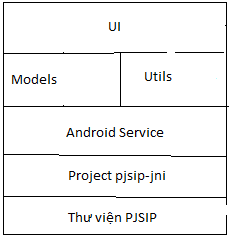


Figure 14: Kiến trúc ứng dụng Voip Android

Ứng dụng Voip Android được phân chia theo các package như sau:

* package ui: Chứa các class về xử lí về giao diện người dùng ,các màn hình quản lí account (thêm ,sửa, xóa) , quản lí cuộc gọi (lịch sử , màn hình lúc đang đàm thoại) ,quản lí tin nhắn (các màn hình danh sách tin nhắn,màn hình tạo tin nhắn mới) ,cài đặt …
* package utils: Chứa các class tiện ích dùng cho việc truy xuất database , cập nhật giao diện , kiểm tra sử tương thích giữa các dòng máy , các thông số dành cho việc cài đặt và khởi động hệ thống.
* package service: Đây có thể gọi là trái tim của Voip Android ,package này xử lí gần như toàn bộ chức năng của hệ thống ,nó có nhiệm vụ nạp thư viện PJSIP vào bộ nhớ .Đưa ra các dịch vụ để cho ứng dụng chính gọi tới: Đăng kí tài khoản ,cập nhật trạng thái , thiết lập cuộc gọi , ngắt kết nối , hold , xử lí điều khiển media:bật tắt loa ,mic … (voice) thông qua các hàm của PJSIP
* package models: Là các lớp dùng để chứa các thông tin cần thiết để xử lí trong hệ thống như tài khoản , tin nhắn , cuộc gọi

Voip Android cũng dựa vào hai project là pjsip và pjsip-jni

* PJSIP là thư viện viết bằng C có hiệu năng cao, ổn định, hỗ trợ nhiều nền tảng và hiện tại đang được phát triển mạnh mẽ. Ngoài ra, thư viện đi kèm PJSIP đã được đóng gói sẵn các thư viện xử lý media, NAT. trang chủ dự án: <http://www.pjsip.org>
* Pjsip –jni là dự án tạo ra giao diên bằng Java ( JNI – Java native interface ) cho phép gọi các hàm pjsip viết bằng C từ Java . Project Pjsip-jni được để trong package org.pjsip.pjsua trong thư mục **src.**

Dùng System.Load (libpjsipjni.so) ta sẽ nạp thư viện pjsip vào bộ nhớ, dùng các hàm trong project pjsip-jni để gọi tới PJSIP từ trong chương trình java .PJSIP đươc nạp và chạy trong một tiến trình khác với ứng dụng chính .Cụ thể thì nó chạy ngầm trong một service cung cấp các “dịch vụ” để ứng dụng chính kết nối vào sử dụng .Để PJSIP chạy trong một service đảm bảo người dùng vẫn nhân được cuộc gọi và tin nhắn đến ngay cả khi ứng dụng chính không chạy .Việc thông báo dựa vào cơ chế notification của Android .

* + - * 1. **Chi tiết cài đặt**

Dưới đây là một số Class tiêu biểu

* **Voip Android Service:**
  + **UAStateReceiver**

Lớp này được xem như là một cầu nối giữa ứng dụng Voip Android và PJSIP, kế thừa lớp CallBack của PJSIP, override các hàm của CallBack .Nó sẽ được PJSIP gọi khi có các sư kiện xảy ra

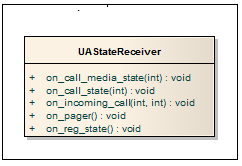


Figure 15 Class UAStateReceiver

* + on\_incoming\_call(acc\_id, callId)

Được gọi khi có cuộc gọi đến cho account có id là acc\_id, callId là định danh duy nhất của cuộc gọi đó, hàm này được dùng để thông báo cho người dùng khi nào có cuộc gọi đến ( rung , chuông , notification …) và khởi tạo Activity xử lí cuộc gọi .

* + on\_call\_state ( callId)

Được gọi khi có cuộc gọi có định danh là callId đang được xử lí (in progress), hàm này được gọi mỗi khi cuộc gọi đó thay đổi trạng thái, các trạng thái được định nghĩa như sau

* + - *CALLING:* là trạng thái khi bên gọi đang cố gắng gửi bản tin invite kết nối tới sip server.
    - *INCOMING****:*** là trạng thái báo có cuộc gọi tới.
    - *EARLY:* khi bên nhận cuộc gọi trả lời cho bên gọi bằng bản tin sip có mã là 1xx.
    - *CONNECTING:* là trạng thái mà bên gọi chờ để nhận bản tin 200 OK, còn bên nhận chờ để nhận bản tin xác nhận ACK.
    - *CONFIRMED:* đây là trạng thái thực sự hai bên đang đàm thoại với nhau .Một luồng RTP được thiết lập giữa hai bên để truyền dữ liệu voice
    - *DISCONNECTED:* là trạng thái kết thúc cuộc gọi, người chủ động kết thúc gửi bản tin BYE còn người kia xác nhận lại bằng bản tin 200 OK
  + on\_pager (call\_id, from, to, body)

Được gọi khi có tin nhắn tới:

From: nguồn của tin nhắn (uri của người gửi) VD: vinhnk@ekiga.net

To: nơi nhận tin nhắn (uri của ngưởi nhận) VD: vinh.soict@ekiga.net

Body: nội dung tin nhắn.

Ta sử dụng hàm này để báo cho người dùng về tin nhắn mới và thêm nó vào DB.

* + on\_pager\_status (call\_id, to, body, status)

Được gọi khi trạng thái của tin nhắn từ to với nội dung body thay đổi, nếu tin nhắn thành công thì status có mã 2xx. Hàm này dùng để cập nhật trạng thái của tin nhắn trong DB

* + on\_reg\_state ( accountId)

Hàm này để cập nhật trạng thái khi SipService đăng kí account với id là accountId vơi sip server: lỗi khi đăng kí, đang đăng kí, time out, đã đăng kí thành công …

* + on\_call\_media\_state ( callId)

Hàm này được gọi khi trạng thái của cuộc gọi có callId thay đổi, các trạng thái được định nghĩa:

* + - ACTIVE: Trạng thái đang đàm thoại
    - LOCAL\_HOLD: Đây là trạng thái khi người dùng ấn phím hold.
    - REMOTE\_HOLD: Đây là trạng thái khi người bên kia đầu dây ấn phím hold.

Trong đó trạng thái ACTIVE là quan trọng nhất, trong trạng thái này dựa vào cơ chế conference port của pjsip để đưa âm thanh đến loa và điều chỉnh âm lượng theo cài đặt của người dùng.

* + **SipService:**

SIP Service là một thành phần quan trọng nhất trong ứng dụng Voip Android, thành phần này chạy ngầm trong một tiến trình và làm những công việc không cần tới giao diện như duy trì đảm bảo kết nối liên tục với sip server, broadcast cho các activity khi có cuộc gọi, tin nhắn tới, trạng thái lúc đăng kí tài khoản, trạng thái cuộc goi ….

SipService được vận hành theo như đã được lập trình việc sử dụng một Interface mà nó định nghĩa. Các tiến trình khác thiết lập một kết nôi tới đối tượng SipService và sử nó đó để gọi các hàm dịch vụ được cung cấp bởi SipService

Trong Voip Android ta sử dụng AIDL **(**Android Interface Definition Language**)** để định nghĩa giao diện (các dịch vụ mà Sipservice cho phép các tiến trình khác sử dụng) của Sipservice .Việc sử dụng aidl cho phép ta định nghĩa các hàm mà Sipservice cho phép các tiến trình khác gọi ở trong file ISipService.aidl. Một khi tạo ra file aidl công cụ Eclipse + AIDL sẽ tạo ra một file có cùng tên, nhưng phần mở rộng là .java .Trong Voip Android file được tự đông sinh ra là ISipService.java . Ta sẽ dùng lớp ISipService được sinh ra này để thực hiện kết nối tới SipService.

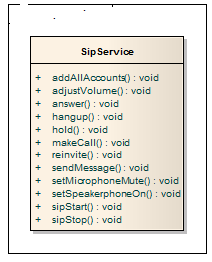


Figure 16 SipService

Dưới đây là các “*dịch vụ*” mà các sipsevice cung cấp cho các tiến trình khác:

* sipStart

Tạo sip stack load thư viện vào bộ nhớ, khởi tạo PJSUA đó là điều rất quan trọng trước khi bất kỳ hàm nào của PJLIB có thể được gọi .Start SipService theo các cấu hình tùy theo sự lựa chọn của người dùng

* + Cho media: chất lượng âm thanh, clock rate …
  + Cho transport: loại giao thức để truyền dữ liệu (UDP, TCP, TLS, RTP) cổng để truyền ứng với từng loại giao thức.
  + NAT: Giao thức để vượt NAT ( các server ứng với một trong các giao thức sau : STUN , TURN , ICE ) .

Khởi tạo các bộ codec để mã hóa âm thanh, sắp xếp các codec theo thứ tự ưu tiên về băng thông.

* sipStop

Stop SipService, giải phóng bộ nhớ

* addAllAccounts

Thêm account từ database vào thư viện pjsip và đăng kí nó với sip server, trạng thái của việc đăng kí (thành công, thất bại …) sẽ được thông báo cho người dùng biết.

* getSipProfileState (accountId):

Lấy thông tin về Account có id là accountId: có được active không, mã trạng thái, tên hiển thị …

* makeCall(callee, accountId)

Dùng Account có id là accountId để gọi đến một người có uri là callee

* answer(callId, status):
* hangup(callId, status):

Trả lời hoặc từ chối (dập máy) một cuộc gọi có id là callId, status là mã được gởi đi trong bản tin sip: 403 từ chối, 200 trả lời, 180 ringing.

* hold( callId)

Giữ máy cho cuộc gọi có id là callId

* getCallInfo(callId)

Lấy thông tin của remote party, trạng thái cuộc gọi: *CALLING*, INCOMING, *EARLY*, CONNECTING, *CONFIRMED*, And DISCONNECTED.Conference port … của cuộc gọi có Id là callId

* setMicrophoneMute (on)
* setSpeakerphoneOn(on)

Turn on hoặc off speaker (chế độ aloud)

* sendMessage(msg, toNumber, accountId)

Send message có nội dung msg tới remote party có uri là toNumber sử dụng account có Id là accountId

Trong Voip Android khi khởi tạo service thì kiểm tra kết nối mạng của điện thoại (có bật mạng 3G hoặc wifi), Service chỉ được tạo nếu sẵn có kết nối .Tiến hành đăng kí tài khoản với sip server đặt trạng thái online cho tài khoản đó .Quá trình đăng kí như được mô tả trong phần lí thuyết .Đồng thời đăng kí với android OS để nhận được thông tin về thay đổi trạng thái kết nối mạng, cuộc gọi GSM.

* Trong trường hợp thay đổi loại kết nối, ví dụ thay đổi từ 3G sang Wifi hoặc ngược lại dẫn tới địa chỉ Ip local thay đổi theo .Trong trường hợp này SipService cần phải khởi động lại.
* Trong trường hợp ứng dụng Voip Android nhận được thông tin cuộc gọi GSM. Cuộc goi gsm được ưu tiên hơn cuộc gọi IP vì thế nếu nếu cuộc goi gsm đang active thì ip tạm dừng (hold) sau khi gsm kết thúc thì kết nối lại
  + **SipNotifications:**

Notificationlà một trong những thành phần quan trọng của Android đưa ra các cảnh báo mà không làm cho các Activity phải ngừng hoạt động .Trong Voip Android ta sử dụng SipNotification để thông báo cho người dùng khi có cuộc gọi đến, cuộc gọi bị nhỡ, tin nhắn đến .Do sử dụng cơ chế Intent của Android nên khi người dùng click vào notification đó thì sẽ dẫn đến activity xử lí sự kiện

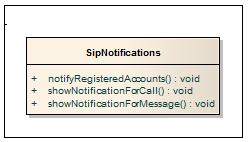


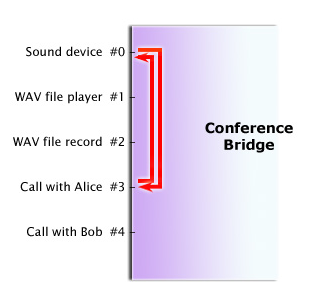
Figure 17 SipNotifications

* notifyRegisteredAccounts:

Thông báo cho người dùng biết những tài khoản nào đã được đăng kí thành công với server

* showNotificationForCall: Thông báo khi có cuộc gọi đên, sau đó chuyển qua activity xử lí cuộc gọi
* showNotificationForMissedCall: Thông báo khi có cuộc gọi nhỡ
* showNotificationForMessage: Thông báo khi có tin nhắn đến , khi click vào chuyển sang activity tạo message
  + **MediaManager:**

Xử lí voice là một phần quan trọng trong Voip Android .Trong PJSIP sử dụng cơ chế conference bridge để định tuyến



giữa các bên trong một cuộc gọi .Với conference bridge sẽ tạo cầu nối cho “dòng chảy” voice tử nguồn tới đích .Như hình trên sound device mặc đinh được đánh số là 0 .Một file wav để được phát lại được đánh số 1 .Một file wav dành cho việc ghi lại cuộc đàm thoại được đánh số 2 .Khi có cuộc goi đang active (đang nói) sẽ được dành cho nó .Ví dụ như hình trên, khi đang nói chuyện với Alice một conference port có số hiệu là 3 được dành cho cuộc gọi này .Để thiết lập luồng âm thanh hai chiều với Alice có thể thực hiện một cách dễ dàng bằng cách kết nối các thiết bị âm thanh với cuộc gọi tại port đó .

Kết nối sounddevice/mic tới port của cuộc gọi

1. lib**.**conf\_connect(0, 3)

Và kết nối port của cuộc gọi tới sound device/speaker của máy điện thoại

1. lib**.**conf\_connect(3, 0)

Trong Voip Android thông tin về conference port của một cuộc gọi cụ thể được chứa trong thuộc tính confPort của class CallInfo , giá trị của nó được cập nhật từ PJSIP .Hình trên minh họa luồng âm thanh hai chiều giữa hai bên (đường màu đỏ)

Chi tiêt tại <http://trac.pjsip.org/repos/wiki/Python_SIP/Media>

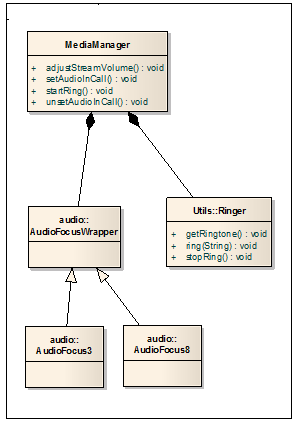
****

Figure 18 Sơ đồ lớp xử lí Media

Hình trên mô tả sơ đồ lớp xử lí media của Voip Android .AudioFocus dùng để xử lí sự kiện khi nút headset được bấm bởi user,nếu trạng thái của cuộc gọi la INCOMING hoặc EARLY thì khi ấn sẽ có chức năng trả lời ,nếu trạng thái là CALLING hoặc CONFIRMED hoặc CONNECTING thì tùy theo cài đặt của người dùng nút này có thể có chức năng là hangup hoặc mute loa .AudioFocusWrapper dùng dể tạo ra hoặc là một thể hiện của AudioFocus3 hoặc AudioFocus8 tùy theo từng dòng máy .Class Ringer dùng để rung và đổ chuông khi có cuộc gọi đến ,nhạc chuông được dùng là được mặc định trong máy .Việc rung và đổ chuông sẽ được thực hiện trong mỗi thread

Lớp MediaManager chạy trong tiến trình của SipService, có nhiệm vụ điều khiển media như bật tắt loa, mic truyền âm thanh nhận được từ PJSIP đưa ra loa, điều chỉnh âm lượng bằng cách sử dụng các hàm của SipService.

* **Voip Android Models:**

1. **SipMessage:**

Khi gửi tin một bản tin sip với Type là MESSAGE được tạo ra, sipservice sẽ chuyển tiếp lên sip server , server có trách nhiệm forward cho địa chỉ sip được chỉ ra trong trường To của bản tin sip

Khi nhận được tin nhắn người nhận sẽ gửi lại bản tin với mã là 200 để thông báo với bên gửi rằng tin nhắn đã nhận thành công . Trong thư viện PJSIP cung cấp cho ta hàm pjsua.*im\_send với các tham số là* accountID dùng để gửi (cho vào trong trường From của sip message), URI của người nhận ví dụ: [vinhnk@ekiga.net](mailto:vinhnk@ekiga.net) , thông điệp dạng text để chèn vào trong sip message .

Trạng thái của bản tin (đã gửi thành công, thất bại) sẽ được PJSIP thông báo cho ta biết qua hàm on\_pager\_status, trong Voip Android ta dùng hàm này để cập nhật trạng thái của bản tin nhắn trong Database (Bảng Message).

Trong Voip Android mỗi tin nhắn là một đối tượng của class SipMessage. Các thuộc tính trong SipMessage tương ứng với các trường trong bản tin sip

* Body: Nội dung tin nhắn.
* From: Địa chỉ URI của ngưởi gửi VD: [vinhnk@ekiga.net](mailto:vinhnk@ekiga.net)
* To: Địa chỉ URI của người nhận VD: [vinh.soict@ekiga.net](mailto:vinh.soict@ekiga.net)
* Status: Trạng thái của tin nhắn.

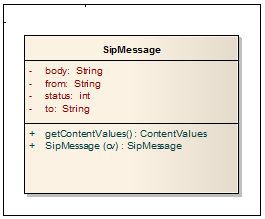


Figure 19 Class SipMessage

Class này tương ứng với Bảng Message trong Database .Hai hàm trên dùng để lấy ra và lưu trữ đối tượng trong Database.

1. **CallInfo:**

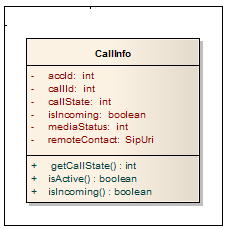
****

Figure 20 Class CallInfo

Class này chứa thông tin về cuộc goi Voip:

1. accid: Id của Account đã thực hiện cuộc gọi , id này là giá trị được lấy ra từ thư viện PJSIP , khi thêm một account vào PJSIP thành công một id sẽ được gán cho account đó.
2. callId: id của cuộc gọi đó , giá trị này được lấy ra từ bản tin INVITE ( trong phần lí thuyết ) , giá trị này là duy nhất cho từng cuộc gọi dùng làm định danh .
3. mediaStatus: trạng thái media của cuộc gọi , các trạng thái được định nghĩa trong PJSIP là : ACTIVE, LOCAL\_HOLD, REMOTE\_HOLD

Trong đó trạng thái ACTIVE là trạng thái quan trọng nhất .Voip Android dựa vào trạng thái này để thiết lập luồng RTP (voice) giữa hai bên, đổ chuông … Giá trị của thuộc tính này được PJSIP cập nhật thông qua hàm on\_call\_media\_state với tham số là id của cuộc gọi.

1. callState: trạng thái của cuộc gọi , các trạng thái được định nghĩa trong PJSIP là : CALLING, INCOMING, EARLY, CONNECTING, CONFIRMED, DISCONNECTED .

Các trạng thái này được thư viện PJSIP cập nhật thông qua các hàm sau:

* on\_call\_state **:** Hàm này sẽ cập nhật trạng thái của cuộc gọi có định danh xác định .Ta dựa vào hàm này để cập nhật thuộc tinh callState
* on\_incoming\_call**:** Báo cho ta biết khi nào có cuộc gọi đến (PJSIP nhận được bản tin INVITE), cuộc gọi này là của account nào (trường hợp có nhiều account cùng dược đăng kí với sip server)

1. remoteContact: thuộc tính này chứa Sip URI của người bên kia đầu dây ,lấy ra từ trường From (trong trường hợp có cuộc gọi đến) ,trường To (cuộc gọi đi) của bản tin INVITE
2. confPort: số hiệu của conference port dùng để giao tiếp với remoteContact

Các thông tin trong lớp này được dùng để cập nhật bảng CallLogs trong Database

1. **IAccount:**

Các account được nhập vào từ Form trong màn hình add account chứa các thông tin quan trọng cho việc đăng kí với Sip Server .Việc đăng kí được thực hiện thông qua bản tin REGISTER .

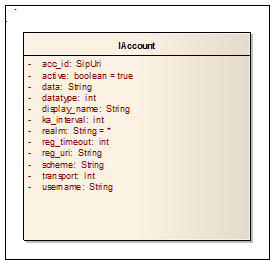


Figure 21 Class IAccount

* acc\_id: Địa chỉ URI của account VD: [vinhnk@ekiga.net](mailto:vinhnk@ekiga.net)
* active: Trường này xác định xem account này đã từng đang kí thành công với sip server lần nào chưa , nếu có thì sẽ tự động đăng kí khi chương trình khởi động hoặc lúc máy khởi động xong
* data: Chứa thông tin về password của account
* datatype: Có hai loại plain pass và digest
* username: tên tài khoản để đăng nhập
* reg\_uri: Địa chỉ của sip server và cổng để đăng kí mặc định là 5060

VD: ekiga.net:5060

* reg\_timeout: thời gian tối đa mà Voip Android cố gắng để đăng nhập
* ka\_interval: khoảng thời gian để đảm bảo keep alive kết nối
* transport: Kiểu để truyền bản tin sip có thể là kiểu UDP hoặc là TCP

Class này tương ứng với bảng Account trong Database.

1. **AccountState:**

Lớp này chứa thông tin về trạng thái của account .Nó có thể là: Đã đăng kí thành công, đang đăng kí, lỗi khi đăng kí, đăng kí hết hiệu lực … ứng với nó là các mã lỗi, thời gian có hiệu lực của account

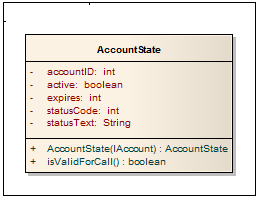
****

Figure 22 AccountState

Giá trị của lớp này được cập nhật từ trong thư viên PJSIP .Lớp này đóng vai trò quan trọng cho việc xác định các account nào đang active, cho phép người dùng chọn để thiết lập cuộc gọi hoặc gửi sip message.

* **Voip Android UI**

Activity hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác, khái niệm activity giống như form trên window   
Các file giao diện trong Voip Android được để trong thư mục res/layout .Ứng với mỗi file này sẽ có một id duy nhất ở trong Class layout của R.java (trong thư mục gen) file này được Android sdk sinh ra .Mỗi màn hình trong Voip Android sẽ có một file xml cho nó .File này sẽ được bung ra khi hàm khởi tạo activity được gọi .Để bung file giao diện ra gọi hàm setContentView với tham số là id của file đó

1. **Account:**

Phần này cho phép người dùng nhập thông tin về username, password, tên hiển thị trong các giao dịch thông qua bản tin sip, địa chỉ của sip server để đăng kí .Sau đó thêm account này vào thư viện PJSIP, Sip service sẽ đăng kí nó với server đã được chỉ ra .Trạng thái của quá trình đang kí sẽ được hiện ra thông báo cho người dùng.

Dưới đây là biểu đồ lớp cho phần giao diện liên quan đến Account trong Voip Android

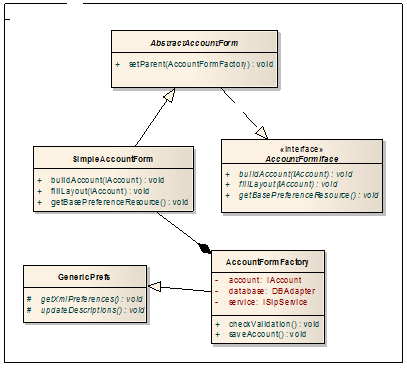
****

Figure 23 Sơ đồ lớp cho giao diện thêm account

Thiết kế này dựa vào Factory design pattern cho phép ta có thể tạo ra nhiểu loại form để thêm account tùy theo nhu cầu sử dụng. Mặc định Factory này sẽ tạo ra giao diện từ class SimpleAccountForm kế thừ từ lớp cơ sở là AbstractAccountForm thực thi giao diện AccountFormIface .Màn hình này cũng được dùng để sửa đổi thông tin tài khoản nếu thuộc tính account của AccountFormFactory khác null

* 1. getBasePreferenceResource: Xác định resource (file xml) sẽ dùng để tạo giao diện
  2. fillLayout: dùng khi chỉnh sửa thông tin tài khoản đã có
  3. checkValidation: kiểm tra xem thông tin mà người dùng nhập vào có hợp lệ hay không
  4. saveAccount: thực hiện việc thêm hoặc cập nhật cơ sở dữ liệu và kết nối với Sipservice thông qua cầu nối ISipService để đăng kí với sip server.

File giao diện cho màn hình tài khoản này được đặt trong thư mục res/xml tên là account\_add\_form.xml, file này chứa các EditText để cho người dùng nhập các thông tin vào

1. **Widgets:**

Tuy các widget của Android là rất đa dạng nhưng đôi khi ta cũng cần những widget khác phù hợp với nhu cầu sử dụng của mình .Trong Voip Android có hai widget cho việc xử lí cuộc gọi đang active .Hai widget này đều kế thừa từ FrameLayout

* IncallInfo: widget này dùng để hiển thị các thông số của cuộc gọi như:Tên hiển thị của người gọi ,địa chỉ URI của họ ,thời gian dàm thoại , trạng thái của cuộc gọi
  + updateRemoteName: Cập nhật tên hiển thị , số điện thoại của remote party
  + updateTitle: Cập nhật trạng thái của cuộc gọi lên màn hình
  + updateElapsedTimer: Hiển thị thời gian dàm thoại đã trôi qua

File giao diện của widget này là in\_call\_info.xml trong thư mục layout

* IncallControls: widget dùng để điều khiển cuộc gọi như: cúp máy , bât/tắt loa/mic, giữ cuộc gọi , từ chối hoặc trả lời khi có cuộc gọi tới
  + setCallState: với tham số là một đối tượng của CallInfo chứa thông tin của cuộc gọi hiện thời .Tùy vào trạng thái mà chỉnh sửa giao diện sao cho phù hợp VD: khi có người gọi đến thì hiện ra hai nút là Answer và Decline còn ẩn các nút điều khiển khác, còn khi trạng thái là CONFIRMED thì ẩn hai nút đó và hiện ra các nút điều khiển (hold, hangup, turn on/off…)
  + Trong widget này có một tham chiếu đến một Interface là OnTriggerListener .Interface này có một hàm callback là onTrigger sẽ được gọi khi người dùng ấn các phím điều khiển trong widget này

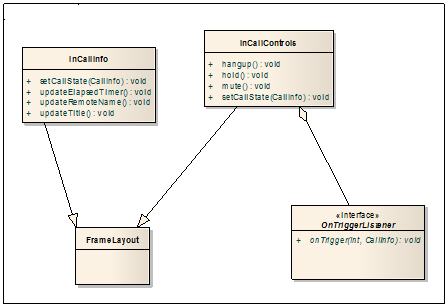
****

Figure 24 Sơ đồ lớp của Voip Android widget

1. **Main UI:**

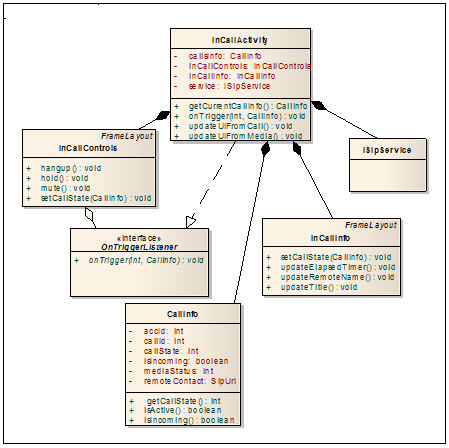


Figure 25 Sơ đồ lớp của giao diện xử lí cuộc gọi trong Voip Android

Đây là activity xử lí cuộc gọi đang đàm thoại của Voip Android .Thực thi giao diện OntriggerListener, implement hàm onTrigger, hàm này sẽ được gọi khi người dùng ấn các phím điều khiển (đã được mô tả ở trên) trong màn hình này .Thông tin về cuộc gọi được lấy ra từ Sip Service thông qua cầu nối ISipService, lưu gữi trong lớp CallInfo .Từ các thông tin này widget IncallInfo sẽ được cập nhật để hiển thị lên cho người dùng .Việc update được thực hiện nhờ hai phương thức

* updateUIFromCall: cập nhật dựa vào trạng thái cuộc gọi
* updateUIFromMedia: cập nhật màu sắc của màn hình dựa vào trạng thái media

Mỗi khi ấn một phím điều khiển hàm onTrigger được thực thì trong InCallActivity

Sẽ được gọi với các tham số là định danh của hành động, dựa vào đó cùng với cầu nối ISipService sẽ gọi các dịch vụ của SipService phù hợp với yêu cầu của người dùng

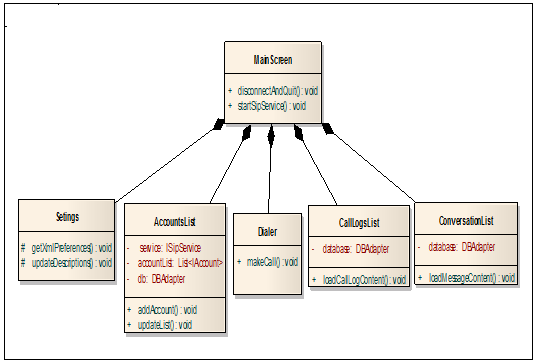


Figure 26 Sơ đồ lớp giao diện chính của Voip Android

MainScreen là màn hình chính của ứng dụng Voip Android, cung cấp các chức năng như: cài đặt, thao tác với các account, lịch sử cuộc gọi, tin nhắn, thực hiện cuộc gọi.

1. **Setting**:

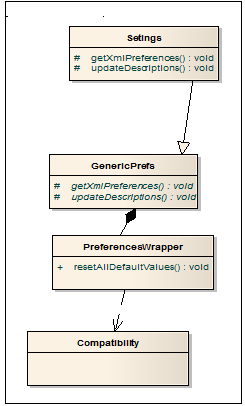


Figure 27 Sơ đồ lớp giao diện cài đặt của Voip Android

Activity này dùng cho người dùng cài đặt các thông số theo nhu cầu của họ .Các thông số này quan trọng cho việc khởi tạo Sip Service .Vì vậy mỗi khi người dùng chọn xong cài đặt của họ Voip Android bắt buộc phải khởi động lại Sip Service.

Các thông số cho việc cài đặt ở trong lớp PreferenceWrapper .Voip Android được thiết kế cho nhiều dòng máy khác nhau nên việc kiểm tra tính tương thích là điều quan trọng, việc này được thực hiện trong class Compatiblity, nó kiểm mic/loa loại CPU, API level mà dòng máy đang dùng .Một số thông số cài đặt đáng lưu ý:

* 1. Loại giao thức dùng để truyền dữ liệu có thể là UDP hoặc TCP,cổng (phía local dùng để kết nối)
  2. Cài đặt về giao thức để vượt NAT, mặc định trong Voip Android dùng giao thức STUN ,người dùng có thể bật tắt chức năng này tùy theo ngữ cảnh họ đang dùng: nếu ở trong mạng LAN thì nên bỏ chọn STUN, nếu không thì phải chọn STUN, họ có thể lựa chọn STUN server để dùng .Một số STUN server: stun.ekiga.net hoặc stun.counterpath.com
  3. Lựa chọn loại mạng để dùng, trong Voip Android hỗ trợ cả 3G và wifi ,tuy nhiên dùng wifi cho chất lượng thoại tốt hơn

1. **AccountList**:

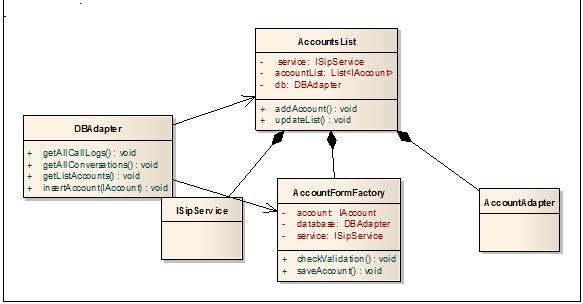


Figure 28 Sơ đồ lớp giao diện quản lí tài khoản của Voip Android

Đây là activity quản lí tài khoản, cho phép người dùng xem thông tin, trạng thái của từng tài khoản .Các Account được trình bày dưới dạng listview, AccountAdapter dùng để hiển thị tên cùng trạng thái đăng nhập .Các Account được lấy ra từ database (hàm getListAccount) .Activity này cũng cho phép người dùng sửa, xóa, thêm tài khoản .Bất cứ sự thay đổi nào đều được cập nhật vào cơ sở dữ liệu và báo cho SipService biết thông qua cầu nối ISipService

1. **Dialer:**
2. **CallLogList:**
3. **ConversationList:**

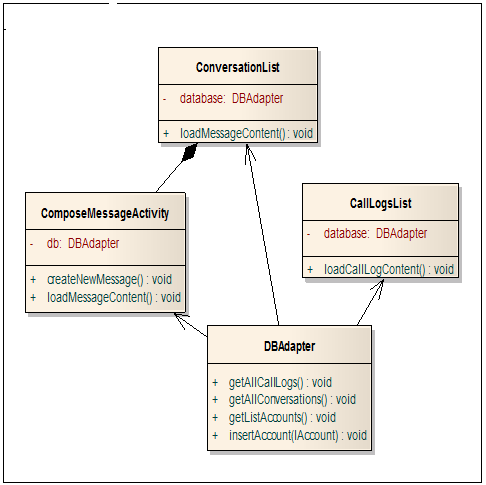


Figure 29 Sơ đồ lớp giao diện quản lí lịch sử cuộc gọi của Voip Android

Việc quản lí các tin nhắn, lịch sử cuộc gọi được thực hiện trong activity này .Nó cho phép người dùng xem thông tin về lịch sử cuộc gọi (Ngày giờ, độ dài đàm thoại, loại cuộc gọi…), các tin nhắn đã nhận và gửi (nội dụng, ngày giờ, loại tin nhắn) .Các cuộc gọi và tin nhắn được lấy ra từ bảng Messages và CallLogs trong database thông qua các hàm getAllCallLog và getAllConversation của lớp DBAdapter .Người dùng cũng có thể tạo một tin nhắn mới trong activity này .Mọi thao tác như xóa, tạo thêm tin nhắn sẽ được lưu vào DB để tiện cho việc quản lí .

1. **OutgoingCallChooser:**

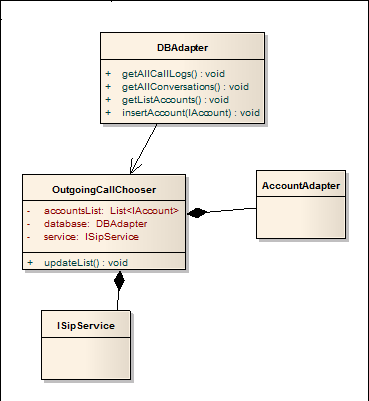


Figure 30 Sơ đồ lớp chọn tài khoản để gọi hoặc gửi tin của Voip Android

Activity này dùng để chọn một sip uri trong các account đã đăng kí với server để tạo một cuộc gọi mới hoặc gửi một tin nhắn mới .DBAdapter sẽ truy vấn DB và chỉ lấy ra các account ở trạng thái active .Sử dụng AccountAdapter để hiển thị chúng lên một listview cho người dùng lựa chọn .Khi người dùng chọn một account nào đó ,thông qua ISipService sẽ gọi dịch vụ gọi hoặc gửi message của SipService

* + - * 1. **Đánh giá**

**Ưu điểm:**

* Kamailio server trong hệ thống này có độ sẵn sàng cao, làm việc với nhiều giao dịch đồng thời và cần có khả năng mở rộng .Có khả năng quản lý hàng nghìn cuộc gọi mỗi giây .Kamailio được duy trì và phát triển bởi một cộng đồng lớn các lập trình viên có kinh nghiệm, ngày càng hỗ trợ kết nối tới nhiều hệ thống viễn thông khác .Vì thế hiệu năng của hệ thống được đảm bảo.
* Mặt khác, sử dụng bộ thư viện PJSIP giúp xây dựng ứng dụng Voip phía client dễ dàng và nhanh chóng giống các webpages .Hơn nữa PJSIP được viết từ C nên có hiệu năng cao, hỗ trợ xử lí media, các giao thức vượt NAT … giúp giảm chi phí phát triển.
* Ứng dụng Voip Android được thiết kế cho nhiều dòng máy nên có thể chạy trên nhiều máy android ,có thể kết nối tới các IP Phone khác sử dụng cùng giao thức Sip (Đã test với Xlite 3.0 cho kết quả khá tôt)
* Hệ thống tận dụng hạ tầng mạng IP có sẵn giúp tiết kiệm chi phí đàm thoại khi có kết nối mạng .Với mạng wifi cho chất lượng thoại tương đối tốt.

**Nhược điểm:**

Bên cạnh các ưu điểm vừa liệt kê trên thì hệ thống cũng còn tồn tại một số nhược điểm:

* Giao diện của một số chức năng trong hệ thống chưa được thiết kế một cách tối ưu nhất, nên những chức năng chưa đem lại được sự tiện dụng nhất cho người dùng của hệ thống.
* Thiếu chức năng quản lí danh bạ (các sip URI)
* Chất lượng đàm thoại còn thấp , độ trể còn lớn (hạn chế cố hữu của mạng IP) với mạng 3G
* Chưa có hỗ trợ về bảo mật bản tin sip, luồng voice
* Chưa có giao diện người dùng để quản lí server

**KẾT LUẬN**

***Kết luận chung:***

Đồ án đi sâu vào tìm hiểu các khái niệm, kiến trúc của Voip .Qua đó xây dựng một hệ thống làm thử nghiệm tính khả thi của công nghệ voip .Ngoài ra các công cụ được chọn lựa để phát triển hệ thống giúp cho đảm bảm được hiệu năng của hệ thống .Nhìn chung đồ án đã thực hiện hoàn thành các nhiệm vụ được đề ra

Các nhiệm vụ hoàn thành:

* Nghiên cứu công nghệ voip, tìm hiểu lập trình android
* Xây dựng ứng dụng Voip client trên android dựa vào thư viện PJSIP
* Xây dựng một hệ thông Voip cơ bản (cả phía server và client)

***Định hướng phát triển trong tương lai:***

Về cơ bản, hệ thống được xây dựng đã đạt được những mục tiêu đề ra .Tuy nhiên, hệ thống vẫn còn một số tồn tại, nếu có thêm cơ hội, em mong muốn có thể phát triển và hoàn thiện sản phẩm này:

* Cải tiến và chuẩn hóa giao diện cho người dùng được tiện lợi và thân thiện nhất để tăng hiệu quả sử dụng hệ thống
* Nâng cao bảo mật cho hệ thống, bảo mật cho bản tin sip (dùng TLS) , bảo mật cho cuộc gọi, chống lại nghe lén, giả mạo …
* Cho phép quản lí người dùng ở phía server qua giao diện web
* Hiện tại với chất lượng mạng 3G thì độ trễ còn khá lớn, cần cải tiến codecs để cho chất lượng nén tốt hơn

TÀI LIỆU THAM KHẢO

**Sách**

Huỳnh Quyết Thắng, Phân tích và thiết kế phần mềm, 2010.

Martin Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling

Language, Third Edition, Addison Wesley, 2003

**Tài liệu tham khảo từ Internet**

<http://www.sparxsystems.com>.

[http://en.wikipedia.org](http://en.wikipedia.org/" \t "_blank)

http://pjsip.org

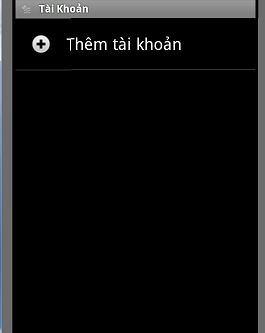
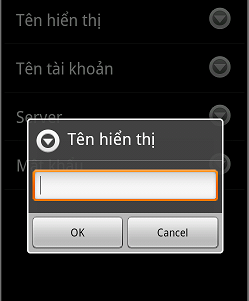
http://google.com/

http://sourceforge.net/projects/pjsip-jni/

http://www.kamailio.org/w/

**PHỤ LỤC**

Một số hình ảnh minh họa

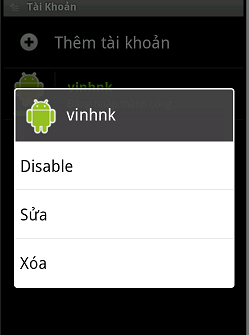
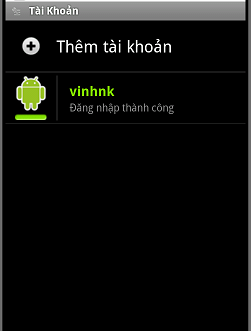
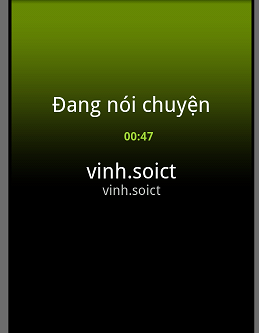
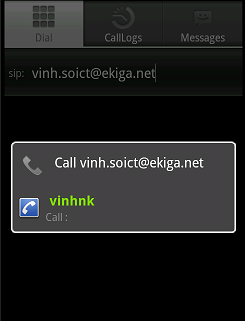


Figure 31 Quản lí tài khoản



Figure 32 Màn hình cài đặt và gửi tin nhắn



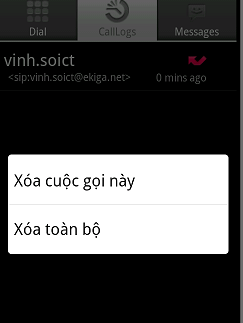
 

Figure 33 Màn hình quản lí cuộc gọi