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

LUẬN VĂN THẠC SỸ KHOA HỌC

LẬP TRÌNH SIP CHO THIẾT BỊ DI ĐỘNG BẰNG JAVA

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

TRẦN XUÂN THẢO

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

HÀ NỘI 2006

MỤC LỤC

MỞ ĐẦU

Ngày nay công nghệ thông tin di động đang phát triển. Các máy điện thoại di động ngoài việc thực hiện chức năng thoại bình thường còn được tích hợp thêm nhiều tính năng khác như cho phép người sử dụng có thể cài đặt thêm chương trình. Hãng Sun MicroSystem đã phát triển phần mềm Java cho lập trình di động (J2ME) mà hiện nay nhiều nhà sản xuất thiết bị đã tích hợp vào. Song song với thông tin di động thì mạng IP cũng đang phát triển nhanh chóng. Đã có nhiều nhà cung cấp dịch vụ thoại qua mạng IP nhưng thoại qua mạng IP sử dụng thiết bị đầu cuối di động còn ít. Giao thức điều khiển báo hiệu phiên (SIP) là một giao thức báo hiệu đơn giản nhưng có khả năng cao để điều khiển báo hiệu trong mạng IP.

Trong quá trình học cao học ngành xử lý thông tin và truyền thông, em rất tâm đắc với môn học lập trình hệ phân tán của thầy giáo, TS Hà Quốc Trung. Do vậy em quyết định chọn đề tài "Lập trình SIP cho thiết bị di động bằng Java". Em xin chân thành cảm ơn thầy giáo TS Hà Quốc Trung tận tình hướng dẫn em trong quá trình thực hiện luận văn. Em cũng xin cảm ơn bạn bè, đồng nghiệp đã hỗ trợ em trong quá trình thực hiện luận văn này.

Luận văn gồm 5 chương:

Chương 1 nghiên cứu về giao thức điều khiển báo hiệu phiên (SIP).

Chương 2 nghiên cứu về lập trình cho thiết bị di động.

Chương 3 nghiên cứu sử dụng bộ công cụ để phát triển các MIDlet.

Chương 4 nghiên cứu về các giao diện ứng dụng chương trình SIP.

Chương 5 là lập một chương trình SIP có các chức năng đăng nhập, gọi đến một thiết bị SIP khác, chờ và trả lời cuộc gọi từ một thiết bị SIP khác đến.

CHƯƠNG 1: GIAO THỨC ĐIỀU KHIỂN PHIÊN (SIP)

1.1. Khái niệm

SIP là một giao thức lớp ứng dụng được thiết kế và phát triển bởi IETF. Đặc tả SIP có trong một vài RFC, trong đó quan trọng nhất là RFC 3261.

SIP là một giao thức báo hiệu cho điều khiển các phiên đa phương tiện. Nói cách khác, SIP cung cấp một cách thiết lập truyền thông thoại, video và tin nhắn giữa các thiết bị.

SIP dựa trên HTTP (Hyper Text Transfer Protocol – giao thức truyền siêu văn bản). SIP là một giao thức Client/Server, trong đó các yêu cầu được bên gọi (client) đưa ra và bên bị gọi (server) trả lời.

1.2. Các đặc điểm của SIP

- Tính di động: SIP cho phép một client một vị trí cố định bất kỳ, do đó cuộc gọi có thể được định tuyến tới nó sử dụng một địa chỉ đã biết như một địa chỉ email.
- Cấu trúc bản tin mềm dẻo: bản tin của SIP có dạng văn bản (text) làm cho nó dễ dàng mở rộng thêm các ứng dụng mới.
- Phân tán chức năng giữa các thiết bị: SIP cho phép các yêu cầu có thể được định tuyến động qua các thiết bị khác nhau, cho phép chức năng được phân tán và các yêu cầu định tuyến qua các thiết bị liên quan.
- Sự thỏa thuận của các tính năng được hỗ trợ: điều này làm cho SIP rất có thể thích nghi như sự mở rộng phương tiện và giao thức được sử dụng cho một cuộc gọi riêng biệt được thỏa thuận giữa các client trong cuộc gọi đó. Kết quả là SIP có thể thiết lập bất cứ kiểu hội thoại nào bao gồm thoại, video, tin nhắn.
- Tách riêng báo hiệu và thông tin: trong SIP đường đi của báo hiệu và thông tin hoàn toàn độc lập.

- Sự chia nhánh: điều này cho phép các thiết bị được liên kết với một địa chỉ đơn, do đó tất cả hoặc là một sự lựa chọn của các thiết bị này có thể được liên lạc đồng thời hoặc tuần tự tùy thuộc chính sách địa phương.

1.3. Các phần tử mạng SIP

1.3.1. User agent (UA)

UA là thiết bị đầu cuối trong mạng SIP. UA có thể là một máy tính cài phần mềm SIP, có thể là điện thoại SIP, điện thoại di động, PDA …

UA thường được đề cập tới là UA server (UAS) và UA client (UAC). UAS và UAC chỉ là các thực thể logic, mỗi UA đều chứa 1 UAS và UAC. UAC là một phần của UA mà gửi yêu cầu và nhận trả lời. UAS là một phần của UA mà nhận yêu cầu và gửi trả lời.

1.3.2. Proxy Server

Proxy server là thiết bị trung gian giữa các UA. Proxy server chuyển các yêu cầu và trả lời giữa các UA.

Có 2 loại proxy server là proxy server trạng thái (stateful) và proxy server không trạng thái (stateless).

1.3.2.1. Proxy server không trạng thái

Proxy server không trạng thái đơn giản chỉ là một bộ chuyển tiếp bản tin. Nó chuyển tiếp các bản tin độc lập với nhau. Mặc dù các bản tin được sắp xếp vào các giao dịch nhưng nó không quan tâm đến giao dịch.

Proxy server không trạng thái đơn giản nhưng nhanh hơn proxy server trạng thái. Một trong những hạn chế của proxy server không trạng thái là nó không có khả năng truyền lại các bản tin và thực hiện các định tuyến phức tạp hơn ví dụ như chia nhánh .

1.3.2.2. Proxy server trạng thái

Proxy server trạng thái phức tạp hơn. Khi nhận được một yêu cầu, proxy server tạo ra một trạng thái, trạng thái này được duy trì cho tới khi kết thúc phiên giao dịch. Một số giao dịch, đặc biệt là giao dịch được tạo bởi "INVITE" có thể kéo dài hơi lâu (đến khi bị gọi nhấc máy hoặc từ chối cuộc gọi). Bởi vì máy chủ phải quản lý trạng thái trong suốt thời gian giao dịch nên sự thực thi của nó bị giới hạn.

Khả năng liên kết các bản tin SIP vào trong các giao dịch làm cho proxy server có một số tính năng thú vị. Proxy server có thể thực hiện việc chia nhánh, tức là trong khi nhận một bản tin thì hai hay nhiều bản tin khác có thể được gửi đi.

Proxy server có thể thực hiện việc truyền lại các bản tin bởi vì từ trạng thái của giao dịch nó biết được là đã nhận được cùng bản tin đó chưa. Proxy server có thể thực hiện các phương pháp phức tạp để tìm kiếm một người sử dụng, ví dụ khi máy của người sử dụng ở cơ quan không trả lời thì nó có thể chuyển cuộc gọi đến máy di động của người đó.

Hầu hết các SIP proxy hiện nay là trạng thái vì cấu hình của chúng thường phức tạp.

1.3.3. Registrar server

Registrar server là một thiết bị nhận các yêu cầu đăng ký và lưu trữ thông tin của người sử dụng.

1.3.4. Redirect server

Redirect server là một thiết bị nhận bản tin yêu cầu và gửi trả lại bản tin trả lời chứa danh sách vị trí của một người sử dụng.

1.4. Các bản tin SIP

Truyền thông sử dụng SIP (thường được gọi là báo hiệu) bao gồm một dãy các bản tin. Các bản tin có thể được truyền độc lập bởi mạng. Thông thường mỗi bản tin được truyền trong một gam dữ liệu UDP. Mỗi bản tin bao gồm phần dòng đầu tiên, phần đầu đề và phần thân bản tin. Phần dòng đầu tiên chỉ ra loại của bản tin. Có hai loại bản tin SIP. Bản tin yêu cầu được sử dụng để khởi tạo một số hành động hoặc là báo cho người nhận yêu cầu nào đó. Bản tin trả lời để xác nhận một yêu cầu đã được nhận và được xử lý và chứa trạng thái của việc xử lý.

1.4.1. Các bản tin yêu cầu

- **INVITE** : bản tin này sử dụng để thiết lập một phiên. Ví dụ một bản tin INVITE như sau:

INVITE sip:7170@iptel.org SIP/2.0
Via: SIP/2.0/UDP 195.37.77.100:5040;rport
Max-Forwards: 10
From: "jiri" <sip:jiri@iptel.org>;tag=76ff7a07-c091-4192-84a0-
d56e91fe104f
To: <sip:jiri@bat.iptel.org>
Call-ID: d10815e0-bf17-4afa-8412-d9130a793d96@213.20.128.35
CSeq: 2 INVITE
Contact: <sip:213.20.128.35:9315>
User-Agent: Windows RTC/1.0
Proxy-Authorization: Digest username="jiri", realm="iptel.org",
algorithm="MD5", uri="sip:jiri@bat.iptel.org",
nonce="3cef753900000001771328f5ae1b8b7f0d742da1feb5753c",
response="53fe98db10e1074
b03b3e06438bda70f"

Content-Type: application/sdp

Content-Length: 451

v=0

o=jku2 0 0 IN IP4 213.20.128.35

s=session

c=IN IP4 213.20.128.35

b=CT:1000

t=0 0

m=audio 54742 RTP/AVP 97 111 112 6 0 8 4 5 3 101

a=rtpmap:97 red/8000

a=rtpmap:111 SIREN/16000

a=fmtp:111 bitrate=16000

a=rtpmap:112 G7221/16000

a=fmtp:112 bitrate=24000

a=rtpmap:6 DVI4/16000

a=rtpmap:0 PCMU/8000

a=rtpmap:4 G723/8000

a=rtpmap: 3 GSM/8000

a=rtpmap:101 telephone-event/8000

a=fmtp:101 0-16

Phần đầu dòng cho ta biết đây là bản tin INVITE. Sau đó là địa chỉ của bước truyền tiếp theo của bản tin.

Trường đầu đề "Via" được sử dụng để ghi lại đường đi của bản tin yêu cầu. Sau đó nó được sử dụng để định tuyến bản tin trả lời theo chính xác cùng một đường đi. Bản tin INVITE chỉ chứa một trường "Via" là được tạo ra bởi UA gửi bản tin.

Trường đầu đề “From” và “To” là địa chỉ của người gọi và người bị gọi.

Trường “Call-ID” để nhận dạng các bản tin trong cùng một cuộc gọi. Các bản tin này có cùng môt “Call-ID”.

Trường “Cseq” dùng để chỉ thứ tự của các yêu cầu.

Trường “Contact” chứa địa chỉ IP và cổng mà người gửi đang đợi các yêu cầu từ người bị gọi.

Đầu đề của bản tin được phân cách với phần thân bởi một dòng trống.

Phần thân của bản tin INVITE chứa mô tả kiểu dữ liệu được chấp nhận bởi người gửi và được mã hóa trong SDP.

- **ACK** : bản tin này công nhận đã nhận bản tin trả lời cuối cùng cho INVITE. Thiết lập một phiên sử dụng bắt tay 3 bên. Việc này tốn thêm thời gian trước khi bị gọi chấp nhận hoặc từ chối cuộc gọi. Bị gọi sẽ gửi lại bản tin trả lời cuối cùng theo chu kỳ cho đến khi nhận được bản tin ACK.
- **BYE** : bản tin này dùng để kết thúc một phiên đa phương tiện. Bên nào muốn kết thúc phiên thì gửi BYE cho bên kia.
- **CANCEL** : dùng để hủy bỏ một phiên không được thiết lập đầy đủ.
- **REGISTER** : dùng để máy chủ registrar biết vị trí hiện tại của user. Thông tin về địa chỉ IP và cổng hiện tại của một user chứa trong bản tin REGISTER. Máy chủ registrar lấy thông tin này và đưa vào cơ sở dữ liệu vị trí. Cơ sở dữ liệu này sau đó được sử dụng bởi các proxy server để định tuyến cuộc gọi tới user. Việc đăng ký là hạn chế thời gian và cần phải được làm tươi theo chu kỳ.

1.4.2. Các bản tin phúc đáp

Khi một UA hoặc proxy server nhận được một yêu cầu thì nó gửi lại một phúc đáp. Tất cả yêu cầu phải được phcs đáp trừ yêu cầu ACK. Một phúc đáp điển hình như sau:

SIP/2.0 200 OK

Via: SIP/2.0/UDP 192.168.1.30:5060;received=66.87.48.68

From: sip:sip2@iptel.org

To: sip:sip2@iptel.org;tag=794fe65c16edfdf45da4fc39a5d2867c.b713

Call-ID: 2443936363@192.168.1.30

CSeq: 63629 REGISTER

Contact: <sip:sip2@66.87.48.68:5060;transport=udp>;q=0.00;expires=120

Server: Sip EXpress router (0.8.11pre21xrc (i386/linux))

Content-Length: 0

Warning: 392 195.37.77.101:5060 "Noisy feedback tells:

pid=5110 req_src_ip=66.87.48.68 req_src_port=5060 in_uri=sip:iptel.org

out_uri=sip:iptel.org via_cnt==1"

Dòng đầu tiên của bản tin phúc đáp chứa phiên bản của giao thức (SIP2.0), mã phúc đáp và lý do.

Mã phúc đáp là một số nguyên từ 100 đến 699 và chỉ loại phúc đáp. Có 6 lớp của phúc đáp:

- *1xx* là phúc đáp tạm thời. Một phúc đáp tạm thời là phúc đáp mà báo cho bên nhận biết là yêu cầu tương ứng đã nhận được nhưng kết quả của xử lý là không biết. Phúc đáp tạm thời được gửi chỉ khi việc xử lý không hoàn thành ngay lập tức. Người gửi phải dừng việc gửi lại yêu cầu trên.

Thông thường các proxy server gửi phúc đáp với mã là 100 khi chúng bắt đầu xử lý một INVITE và các UA gửi phúc đáp với mã là 180 (đổ chuông).

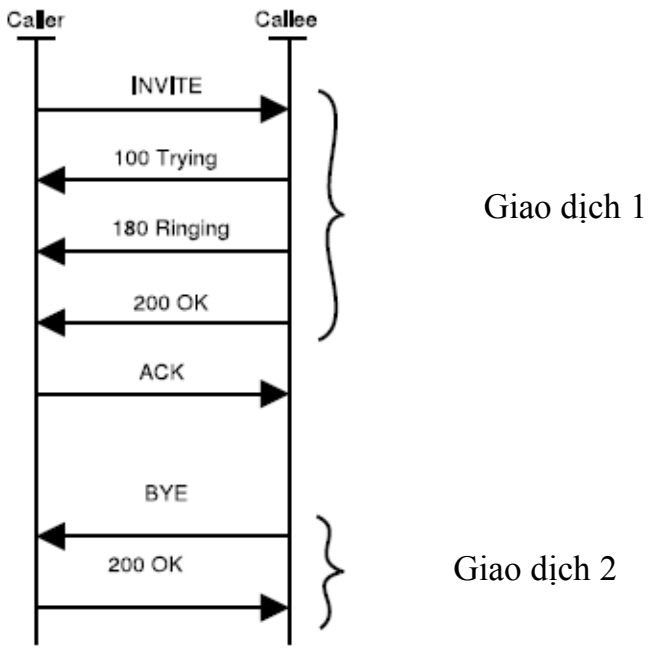
- *2xx* là các phúc đáp xác thực cuối cùng. Phúc đáp cuối cùng đồng thời kết thúc giao dịch. Phúc đáp với mã từ 200 đến 299 là các phúc đáp xác thực có nghĩa là yêu cầu đã được xử lý thành công và được chấp nhận. Ví dụ phúc đáp 200 OK được gửi khi một user chấp nhận lời mời tới một phiên (yêu cầu INVITE).
- *3xx* dùng để định hướng lại một người gọi. Một phúc đáp định hướng lại cho thông tin về vị trí mới của user hoặc một dịch vụ mà người gọi có thể sử dụng. Phúc đáp định hướng lại thường được gửi bởi proxy server. Khi một proxy nhận một yêu cầu và không muốn hoặc không thể xử lý vì bất cứ lý do gì, nó sẽ gửi một phúc đáp định hướng lại tới người gọi và đặt vị trí khác vào trong phúc đáp mà người gọi có thể muốn thử lại. Nó có thể là vị trí của một proxy khác hoặc là vị trí hiện tại của bị gọi (từ cơ sở dữ liệu vị trí của registrar). Chủ gọi sau đó có thể gửi lại yêu cầu tới vị trí mới. Các phúc đáp 3xx là cuối cùng.
- *4xx* là các phúc đáp cuối cùng từ chối. Một phúc đáp 4xx có nghĩa rằng vấn đề ở phía chủ gọi. Yêu cầu không thể xử lý vì nó chứa cú pháp sai hoặc nó không thể được thực hiện ở server.
- *5xx* có nghĩa là vấn đề nằm ở phía server. Yêu cầu có thể hợp lệ nhưng server lỗi không thực hiện được.
- *6xx* có nghĩa là yêu cầu không thể được đáp ứng ở bất kỳ server nào. Phúc đáp này thường được gửi bởi server mà có thông tin cuối cùng về một user cụ thể. Các UA thường gửi bản tin 603 từ chối trả lời khi user đó không muốn tham dự vào phiên.

1.5. Các giao dịch SIP

Mặc dù chúng ta nói rằng các bản tin SIP được gửi một cách độc lập qua mạng, chúng thường được sắp xếp vào các giao dịch bởi các UA và các proxy server. Vì vậy SIP được gọi là các giao thức giao dịch.

Một giao dịch là một chuỗi các bản tin SIP được trao đổi giữa các phần tử mạng SIP. Một giao dịch bao gồm một yêu cầu và tất cả phúc đáp cho yêu cầu đó. Đó bao gồm không hoặc nhiều phúc đáp tạm thời và một hoặc nhiều bản tin cuối cùng.

Nếu một giao dịch được khởi tạo bởi một yêu cầu INVITE thì giao dịch đó cũng bao gồm ACK nhưng chỉ khi phúc đáp cuối cùng không phải là phúc đáp 2xx. Nếu phúc đáp cuối cùng là 2xx thì ACK không phải là một phần của giao dịch.

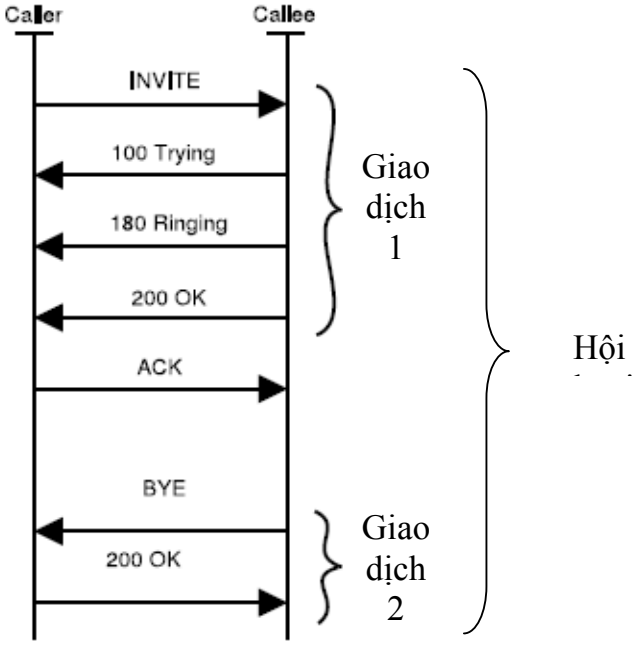


Hình 1.1. Các giao dịch SIP

1.6. Các hội thoại SIP

Một hội thoại tương ứng với một quan hệ SIP đồng đẳng (peer-to-peer) giữa hai UA. Các hội thoại làm cho việc sắp xếp thứ tự và định tuyến các bản tin giữa các điểm cuối SIP được thuận tiện hơn.

Các hội thoại được định danh sử dụng các trường Call-ID, From và To. Các bản tin có ba trường này giống nhau thì cùng trong một hội thoại. Một hội thoại là một dãy các giao dịch. Hình 1.2 chỉ ra các bản tin trong một hội thoại.



Hình 1.2. Hội thoại SIP

Một số bản tin thiết lập một hội thoại, một số thì không. Điều này cho phép biểu diễn rõ ràng mối quan hệ của các bản tin và đồng thời để gửi các bản tin mà không liên quan tới các bản tin khác ngoài hội thoại. Điều này thực hiện dễ dàng hơn bởi vì UA không phải giữ trạng thái hội thoại.

Ví dụ bản tin INVITE thiết lập một hội thoại bởi vì nó sẽ được kèm theo sau bởi yêu cầu BYE để kết thúc phiên được thiết lập bởi INVITE. Bản tin BYE này được gửi trong cùng một hội thoại được thiết lập bởi INVITE. Nhưng nếu một UA gửi một yêu cầu MESSAGE thì yêu cầu này không thiết lập hội thoại. Các bản tin sau sẽ được gửi độc lập với bản tin trước.

1.6.1. Các hội thoại làm cho định tuyến thuận tiện

Chúng ta biết rằng các hội thoại cũng được sử dụng để định tuyến các bản tin giữa các UA.

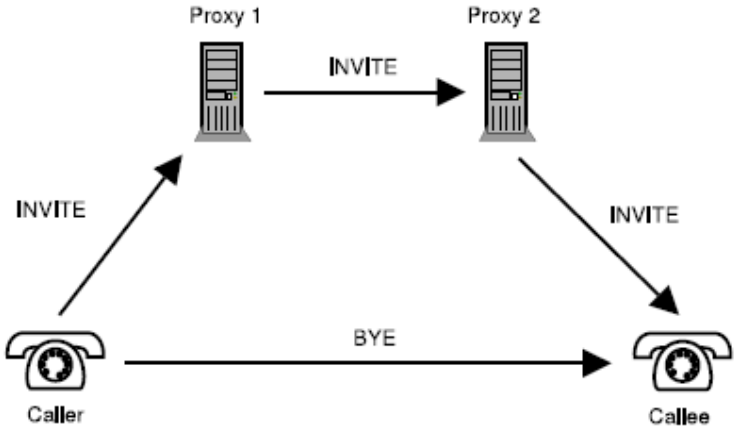
Giả sử rằng user sip:bob@a.com muốn nói chuyện với user:pete@b.com. Chủ gọi biết địa chỉ SIP của bị gọi nhưng địa chỉ này không nói bất kỳ điều gì về vị trí hiện tại của user – chủ gọi không biết phải gửi yêu cầu tới host nào. Vì vậy yêu cầu INVITE được gửi tới một proxy server.

Yêu cầu sau đó được gửi từ proxy đến proxy tới khi nó đến một proxy mà biết vị trí hiện tại của bị gọi. Quá trình này được gọi là định tuyến. Khi yêu cầu đã đến được bị gọi, bị gọi sẽ tạo một phúc đáp gửi lại cho chủ gọi. Bị gọi đồng thời cũng đưa trường đầu đề "Contact" vào trong phúc đáp và sẽ chứa địa chỉ hiện tại của user. Yêu cầu gốc cũng chứa trường đầu đề "Contact" có nghĩa là cả hai UA biết vị trí hiện tại của nhau.

Bởi vì các UA biết vị trí của nhau nên không cần thiết phải gửi yêu cầu qua các proxy nữa, chúng có thể được gửi trực tiếp từ UA đến UA. Đó chính là hội thoại làm cho định tuyến thuận tiện.

Các bản tin trong cùng một hội thoại sau đó sẽ được gửi trực tiếp từ UA đến UA. Điều này giúp cải thiện hiệu năng bởi vì các proxy không xem tất cả các bản tin trong cùng một hội thoại, chúng thường được sử dụng để định tuyến yêu cầu đầu tiên mà thiết lập hội thoại. Các bản tin trực tiếp đồng thời cũng được truyền với độ trễ nhỏ hơn nhiều bởi vì một proxy thường thực hiện

các phép toán logic định tuyến phức tạp. Hình 1.3 là một ví dụ minh họa những điều trên.



Hình 1.3. Hình thang SIP

1.6.2. Nhận dạng hội thoại

Chúng ta đã biết rằng sự nhận dạng hội thoại bao gồm ba phần : Call-Id, From tag và To tag. Nhưng nó không rõ ràng tại sao các phần nhận dạng hội thoại lại được tạo chính xác và ai đóng góp phần nào.

Call-Id được gọi là phần nhận dạng cuộc gọi. Nó phải là một chuỗi ký tự duy nhất để nhận dạng một cuộc gọi. Một cuộc gọi bao gồm một hay nhiều hội thoại. Đa UA có thể phúc đáp cho một yêu cầu khi một proxy phân nhánh yêu cầu đó. Mỗi UA gửi một phúc đáp 2xx thiết lập một hội thoại riêng rẽ với chủ gọi. Tất cả các hội thoại này là một phần của cùng một cuộc gọi và có cùng tham số “Call-Id”.

From tag được tạo ra bởi chủ gọi và nó nhận dạng duy nhất hội thoại trong UA của chủ gọi.

To tag được tạo ra bởi bị gọi và nó nhận dạng duy nhất hội thoại trong UA của bị gọi.

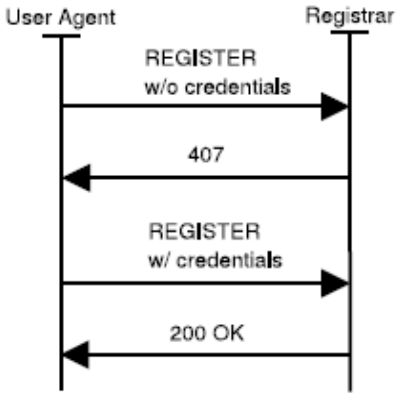
Sự nhận dạng hội thoại phân cấp này là cần thiết bởi vì một sự mời cuộc gọi có thể tạo ra vài hội thoại và chủ gọi phải có khả năng phân biệt chúng.

1.7. Những kịch bản SIP điển hình.

Phần này trình bày một cách khái quát các kịch bản SIP điển hình thường xảy ra.

1.7.1. Đăng ký

Người sử dụng phải đăng ký với một registrar để những người sử dụng khác có thể liên lạc đến được. Một sự đăng ký bao gồm một bản tin "REGISTER" và một phúc đáp "200 OK" từ registrar nếu sự đăng ký là thành công. Sự đăng ký thường phải xác thực do đó một phúc đáp "407" có thể được gửi nếu người sử dụng không cung cấp sự tin cậy hợp lệ. Hình 1.4 là một ví dụ về sự đăng ký.



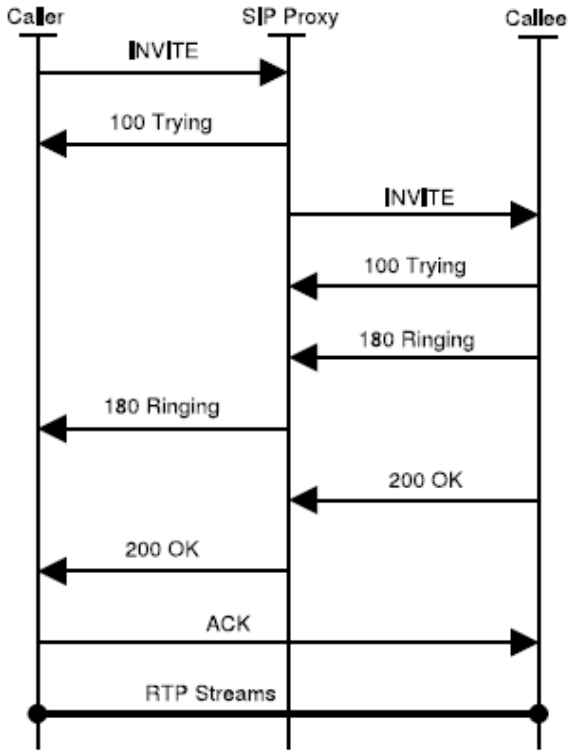
Hình 1.4. Sự đăng ký

1.7.2. Khởi tạo phiên

Một sự khởi tạo phiên bao gồm một yêu cầu "INVITE" mà thường là gửi tới một proxy. Proxy gửi ngay một phúc đáp "100 Trying" để ngừng việc gửi lại và chuyển yêu cầu sau này.

Tất cả phúc đáp tạm thời được tạo bởi bị gọi được gửi lại cho chủ gọi. Khi bị gọi đổ chuông nó gửi phúc đáp cho chủ gọi bản tin “180 Ringing”.

Khi bị gọi nhấc máy nó gửi lại bản tin “200 OK”và nó được gửi lại cho đến khi nhận được một “ACK” từ chủ gọi. Phiên được thiết lập ở điểm này. Hình 1.5 là một ví dụ minh họa sự khởi tạo phiên.



Hình 1.5. Khởi tạo phiên

1.7.3. Kết thúc phiên

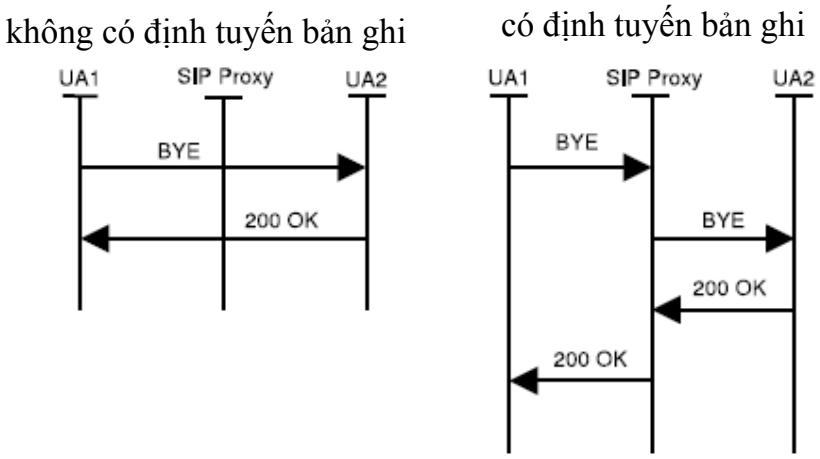
Kết thúc phiên được thược hiện bằng cách gửi bản tin “BYE”. Bản tin “BYE” được gửi trực tiếp từ một UA đến UA khác trừ khi một proxy trên đường đi của yêu cầu “INVITE” chỉ ra rằng nó phải đi theo bằng cách sử dụng định tuyến bản ghi.

Bên muốn kết thúc phiên gửi một yêu cầu “BYE” tới bên kia. Bên kia sẽ gửi lại một phúc đáp “200 OK” để xác nhận yêu cầu “BYE” và phiên chấm dứt.

1.7.4. Định tuyến bản ghi

Tất cả yêu cầu trong một hội thoại mặc định được gửi trực tiếp từ một UA đến UA khác. Chỉ những yêu cầu ngoài một hội thoại là đi qua các proxy. Cách này làm cho mạng SIP mềm dẻo hơn bởi vì chỉ có một số nhỏ bản tin đến proxy.

Có những tình huống mà proxy cần lưu lại đường đi của tất cả bản tin. Ví dụ proxy điều khiển hộp NAT hoặc proxy thực hiện tính cước cần phải lưu lại đường đi của yêu cầu "BYE".



Hình 1.6. Chấm dứt phiên

Kỹ thuật mà một proxy có thể cho các UA biết là nó muốn lưu lại đường đi của tất cả bản tin được gọi là định tuyến bản ghi. Proxy này sẽ chèn trường đầu đề "Record-Route" vào các bản tin SIP mà có chứa địa chỉ của proxy đó. Các bản tin được gửi trong một hội thoại sẽ đi qua tất cả proxy mà chèn một trường "Record-Route" vào bản tin đó.

Bên nhận yêu cầu đó nhận được một tập các trường "Record-Route" trong bản tin đó. Nó phải ánh xạ lại tất cả trường đó vào trong phúc đáp bởi vì bên phát yêu cầu đó cũng cần phải biết tập proxy đó.

1.8. So sánh SIP và H.323

Ngoài SIP còn có một giao thức báo hiệu khác được sử dụng phổ biến hiện nay là H323.

Giữa H.323 và SIP có nhiều điểm tương đồng. Cả hai đều cho phép điều khiển, thiết lập và huỷ cuộc gọi. Cả H.323 và SIP đều hỗ trợ tất cả các dịch vụ cần thiết, tuy nhiên có một số điểm khác biệt giữa hai chuẩn này. Đó là:

- H.323 hỗ trợ hội nghị đa phương tiện rất phức tạp. Hội nghị H.323 về nguyên tắc có thể cho phép các thành viên sử dụng những dịch vụ như bảng thông báo, trao đổi dữ liệu, hoặc hội nghị video.
- SIP hỗ trợ điều khiển cuộc gọi từ một đầu cuối thứ 3. Hiện nay H.323 đang được nâng cấp để hỗ trợ chức năng này.

Dưới đây là bảng so sánh giữa hai giao thức:

	SIP	H.323
Tổ chức	IETF	ITU
Quan hệ kết nối	Ngang cấp	Ngang cấp
Khởi điểm	Dựa trên mạng Internet và Web. Cú pháp và bản tin tương tự như HTTP.	C sở là mạng thoại. Giao thức báo hiệu tuân theo chuẩn ISDN Q.SIG.
Đầu cuối	Đầu cuối thông minh SIP	Đầu cuối thông minh H.323
Các server lõi	SIP proxy, redirect, location, và registration servers.	H.323 Gatekeeper
Tình hình hiện nay	Giai đoạn thử nghiệm khả	Đã được sử dụng rộng rãi

	năng cùng hoạt động của thiết bị của các nhà cung cấp khác nhau đã kết thúc. SIP nhanh chóng trở nên phổ biến.	
Khuôn dạng bản tin	Text, UTF-8	Nhị phân ASN.1 PER
Trễ thiết lập cuộc gọi	1.5 RTT (round-trip time, tức chu kỳ gửi bản tin và nhận bản tin trả lời hay xác nhận)	6-7 RTT hoặc hơn
Giám sát trạng thái cuộc gọi	Có 2 lựa chọn: chỉ trong thời gian thiết lập cuộc gọi hoặc suốt thời gian cuộc gọi	Phiên bn 1 và 2: máy chủ phi giám sát trong suốt thời gian cuộc gọi và phi giữ trạng thái kết nối TCP -> hạn chế khả năng mở rộng và gim độ tin cậy.
Báo hiệu quảng bá	Có hỗ trợ	Không
Chất lượng dịch vụ	Sử dụng các giao thức khác như RSVP, OPS, OSP để đảm bảo chất lượng dịch vụ	Gatekeeper điều khiển băng thông. H.323 khuyến nghị dùng RSVP để lưu trữ tài nguyên mạng
Bảo mật	Đăng ký tại registrar server,	Chỉ đăng ký khi trong

	có xác nhận đầu cuối và mã hoá	mạng có gatekeeper, xác nhận và mã hóa theo chuẩn H.235
Định vị đầu cuối và định tuyến cuộc gọi	Dùng SIP URL để đánh địa chỉ. Định tuyến nhờ sử dụng redirect và location server	Định vị đầu cuối sử dụng E.164 hoặc tên ảo H.323 và phương pháp ánh xạ địa chỉ nếu trong mạng có gatekeeper. Chức năng định tuyến do gatekeeper đảm nhiệm
Tính năng thoại	Hỗ trợ các tính năng của cuộc gọi cơ bản	Hỗ trợ các tính năng của cuộc gọi cơ bản
Hội nghị	Hội nghị cơ sở, quản lý phân tán	Được thiết kế nhằm hỗ trợ rất nhiều tính năng hội nghị, kể cả thoại, hình ảnh và dữ liệu, quản lý tập trung -> MC có thể tắc nghẽn
Tạo tính năng và dịch vụ mới	Dễ dàng, sử dụng SIP-CGI và CPL	H.450.1
Khả năng mở rộng	Dễ dàng	Hạn chế
Tích hợp với web	Rất tốt, hỗ tợ click to dial	Kém

CHƯƠNG 2 : CƠ BẢN VỀ LẬP TRÌNH CHO THIẾT BỊ DI ĐỘNG BẰNG JAVA

2.1. Giới thiệu

Java được hãng Sun Microsystem giới thiệu vào năm 1995. Ban đầu là phiên bản chuẩn được thiết kế để chạy trên destop và máy trạm. Hai năm sau hãng đưa ra phiên bản mới gọi là phiên bản xí nghiệp dùng cho những ứng dụng lớn. Năm 1999 Sun đưa ra phiên bản nhỏ gọn dùng cho những thiết bị nhúng và cầm tay mà không hỗ trợ thực hiện phiên bản chuẩn. Từ tháng 12/1998 Sun giới thiệu tên mới "Java 2" thay cho phiên bản Java 1.2. Tên mới này được dùng để quy ước cho cho tất cả các phiên bản của Java: phiên bản chuẩn (J2SE), phiên bản xí nghiệp (J2EE) và phiên bản nhỏ gọn (J2ME).

2.2. Máy ảo Java (JVM – Java Virtual Machine)

Các chương trình Java được viết dưới dạng văn bản, sau đó được dịch sang dạng mã byte vào các file lớp (các file có đuôi .class). JVM sẽ biên dịch mã byte này sang dạng mã máy để thực hiện.

2.3. Cấu hình thiết bị

Một cấu hình định nghĩa môi trường chạy J2ME cơ bản. Môi trường này bao gồm máy ảo mà có thể hạn chế hơn máy ảo dùng cho J2SE và một tập các lớp lõi được lấy từ J2SE. Mỗi cấu hình được hướng tới một họ các thiết bị có khả năng tương đương nhau. Hiện tại có hai cấu hình được định nghĩa.

2.3.1. Cấu hình thiết bị kết nối

Cấu hình thiết bị kết nối (CDC – Connected Device Configuration) là cấu hình các thiết bị có kết nối mạng băng thông rộng và thường trực, yêu cầu có tối thiểu 512kb bộ nhớ để chạy Java và 256kb bộ nhớ thực thi chương trình. CDC yêu cầu đầy đủ tính năng của JVM trong J2SE.

2.3.2. Cấu hình thiết bị hạn chế kết nối

Cấu hình thiết bị hạn chế kết nối (CLDC – Connected Limited Device Configuration) có yêu cầu kết nối mạng (thường là không dây) với băng thông và khả năng truy nhập internet thấp. CLDC không yêu cầu nhiều tài nguyên. Nó chỉ yêu cầu tối thiểu 128kb bộ nhớ dùng để chạy Java và 32kb bộ nhớ chạy chương trình. Cấu hình này là cho các thiết bị hạn chế cả về giao diện giao diện người dùng và nguồn năng lượng (pin).

2.3.2.1. Những khác biệt của CLDC so với Java chuẩn

- Dấu chấm động toán học: dấu chấm động toán học đòi hỏi bộ xử lý phải mạnh. Để có thể chạy được trên nhiều đặc tả phần cứng, cài đặt của ngôn ngữ Java trên CLDC không hỗ trợ biến, toán tử, hằng số, hàm… liên quan đến dấu chấm động.
- Không có phương thức hủy: để đơn giản công việc của bộ thu gom rác thì phương thức hủy (finalize) được loại bỏ.
- Quản lý lỗi: JVM dành cho CLDC hỗ trợ một tập hợp rất hạn chế các xử lý ngoại lệ lỗi. CLDC chỉ định nghĩa ba lớp lỗi là java.lang.Error, java.lang.OutOfMemoryError và java.lang.VirtualMachineError. Các lỗi khác được xử lý bởi máy ảo. Một giải pháp đơn giản thường xử lý lỗi là khởi động lại phần cứng (tắt máy bật lại).

2.3.2.2. Các lớp CLDC kế thừa từ J2SE

java.lang.Boolean

java.lang.Byte

java.lang.Character

java.lang.Class

java.lang.Integer

java.lang.Long

java.lang.Math

java.lang.Object

java.lang.Runnable

java.lang.Runtime

java.lang.String

java.lang.StringBuffer

java.lang.System

java.lang.Thread

java.lang.Throwable

java.io.ByteArrayInputStream

java.io.ByteArrayOutputStream

java.io.DataInput

java.io.DataInputStream

java.io.DataOutput

java.io.DataOutputStream

java.io.InputStream

java.io.InputStreamReader

java.io.OutputStream

java.io.OutputStreamWriter

java.io.PrintStream

java.io.Reader

java.io.Writer

java.util.Calendar

java.util.Date

java.util.Enumeration

java.util.Hashtable

java.util.Random

java.util.Stack

java.util.Time

java.util.Vector

2.3.2.3. Khung kết nối chung (GCF – Generic Connection Framework)

GCF là một bộ khung nền mở rộng thư viện vào/ra (I/O) và nối mạng cho thiết bị di động. Nó là một tập con của thư viện java.io và java.net trong J2SE. Tất cả các lớp của GCF được định nghĩa trong gói javax.microedition.io. các lớp trong GCF bao gồm:

Connection

ConnectionNotFoundException

Connector

ContentConnection

Datagram

DatagramConnection

InputConnection

OutputConnection

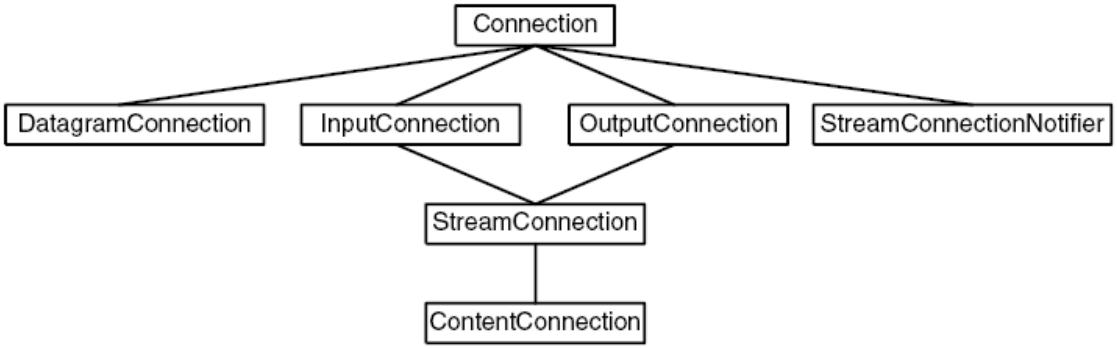
StreamConnection

StreamConnectionNotifier

Thay vì phải tạo các lớp riêng biệt để tạo kết nối, với GCF chỉ có một lớp duy nhất là Connector để tạo các loại kết nối như file, http, datagram… Phương thức mở kết nối có dạng sau:

Connector .Open(“giao thức:địa chỉ; các tham số”);

Hình 2.1 cho thấy phân cấp kết nối trong GCF:



Hình 2.1. Phân cấp kết nối trong GCF

2.4. Profile

Định nghĩa về cấu hình cho các thiết bị là khá tốt cho hầu hết mọi thiết bị. Ví dụ như điện thoại di động, PDA đều có thể xếp vào phân loại CLDC. Tuy nhiên giữa điện thoại di động và PDAvẫn có thiết bị với nhiều khả năng xử lý hơn cái kia. Nhằm mô tả những khả năng khác biệt này và cũng để cung cấp nhiều tính linh hoạt hơn khi công nghệ thay đổi, Sun giới thiệu khái niệm profile dành cho nền J2ME. Một profile là định nghĩa mở rộng thêm cho một phân loại cấu hình. Profile cung cấp những thư viện cho phép người phát triển dùng để viết những ứng dụng chạy trên một kiểu thiết bị đặc biệt.

Profile cho thiết bị thông tin di động (MIDP – Mobile Information Device Profile) định nghĩa tập những hàm API cho phép xử lý những thành phần giao diện người dùng nhập liệu trên thiết bị điện thoại di động, cách xử lý sự kiện, nơi chứa dữ liệu, giao thức kết nối mạng, đối tượng định giờ, quản lý những hạn chế về kích thước màn hình và bộ nhớ đặc thù của điện thoại di động.

2.5. Máy ảo Java cho CLDC

Đối với các thiết bị cấu hình dạng CLDC, Sun cài đặt một phiên bản thu nhỏ hơn dành cho JVM gọi là K virtual machine (KVM). KVM được thiết kế để điều khiển và chạy trên những thiết bị có nguồn tài nguyên hạn chế.

2.6. Xác minh file lớp (.class)

Xác minh sự toàn vẹn và an toàn của những file lớp dùng thực thi (.class) không phải là việc đơn giản. Trong J2SE, bộ kiểm tra mã chiếm khoảng 20kb, chưa kể những yêu cầu không gian heap và thời gian xử lý. Để giảm tải công việc nặng nề này trên thiết bị di động, công việc kiểm tra xác minh độ an toàn của mã được thực hiện làm hai bước:

2.6.1. Tiền xác minh

Trước khi một file lớp được tải về thiết bị di động, một chương trình phần mềm của hệ thống máy ảo được chạy để chèn những thuộc tính bổ sung vào trong file lớp .class. Thông tin này được dùng để giảm bớt về thời gian và bộ nhớ khi JVM thực hiện bước công việc xác minh và kiểm tra mã ở bước 2. Những file lớp này sẽ lớn thêm xấp xỉ khoảng 5% so với file gốc. Những thuộc tính thêm vào một file lớp được gọi là bản đồ ngăn xếp, những thông tin dùng mô tả những biến và toán hạng sẽ chiếm dụng các phần trong ngăn xếp trong quá trình JVM diễn dịch.

2.6.2. Xác minh bởi thiết bị

Khi thiết bị tải file lớp đã qua tiền xác minh ở bước 1 thiết bị sẽ kiểm tra từng đoạn mã thông qua từng chỉ thị lệnh. Có rất nhiều thao tác kiểm tra được thực hiện để xác định tính hợp lệ của mã. Ở tại bất kỳ thời điểm nào, bộ kiểm tra này cũng có thể báo lỗi và loại bỏ file lớp khỏi quá trình thực thi nếu file này không hợp lệ.

2.7. MIDLET

2.7.1. Cơ bản về MIDlet

MIDlet là một ứng dụng Java được thiết kế để chạy trên thiết bị di động, đặc biệt hơn, một MIDlet chứa các lớp Java được dùng bởi CLDC và MIDP. Một bộ MIDlet gồm một hoặc nhiều MIDlet được đóng gói cùng nhau trong file nén JAR.

2.7.1.1. Quản lý ứng dụng và môi trường thực thi Runtime

Bộ quản lý ứng dụng là phần mềm trên thiết bị di động chịu trách nhiệm thiết đặt , chạy và loại bỏ các MIDlet. Phần mềm này là phụ thuộc vào thiết bị (được thiết kế và thực hiện bởi nhà sản xuất thiết bị). Khi bộ quản lý ứng dụng bắt đầu khởi động một MIDlet, nó sẽ chuẩn bị tất cả tài nguyên sau cho ứng dụng:

+ Cho phép truy nhập tới CLDC và KVM: MIDlet có thể sử dụng bất kỳ lớp nào được định nghĩa bên trong CLDC.

+ Cho phép truy nhập tới những lớp MIDP: những thư viện này định nghĩa và cài đặt giao diện người dùng, nơi lưu dữ liệu, mạng, hỗ trợ sử dụng HTTP, thiết bị định giờ và bộ quản lý tương tác người dùng với thiết bị.

+ Cho phép truy nhập tới các file JAR: nếu MIDlet được đóng gói trong file JAR thì tất cả những lớp nào hoặc những tài nguyên khác bên trong file JAR (như hình ảnh) phải sẵn sàng cho MIDlet.

+ Cho phép tới file mô tả ứng dụng Java (JAD): cùng với file JAR, một MIDlet có thể truy nhập tới một file JAD. Nếu file JAD hiện diện thì nội dung phải sẵn sàng cho MIDlet.

2.7.1.2. File lưu trữ Java (JAR)

Một ứng dụng đóng gói khi chuyển giao gồm có nhiều file. Ngoài những file lớp của Java, những file khác như file hình ảnh và dữ liệu ứng dụng, thường được gọi là những file tài nguyên. Các file này được nén cùng nhau vào một file duy nhất được gọi là file JAR (file nén dạng Zip). Ngoài những file lớp và tài nguyên, một file JAR còn chứa đựng một file gọi là file thống kê hay manifest file. File này mô tả nội dung của JAR. File manifest có tên manifest.mf. File này không yêu cầu mọ thuộc tính phải được định nghĩa. Tuy nhiên nếu sáu thuộc tính đầu tiên không có trong file manifest, bộ quản lý ứng dụng sẽ từ chối tải file JAR vào thực thi:

MIDlet-Name

MIDlet-Version

MIDlet-Vendor

MIDlet-<n> (một mục cho mỗi MIDlet trong file JAR).

MicroEdition-Profile

MicroEdition-Configuration

Thuộc tính MIDlet-<n> tham chiếu đến MIDlet cụ thể bên trong bộ đóng gói ứng dụng gồm nhiều MIDlet. Thông số <n> có thể chứa 3 thông tin sau:

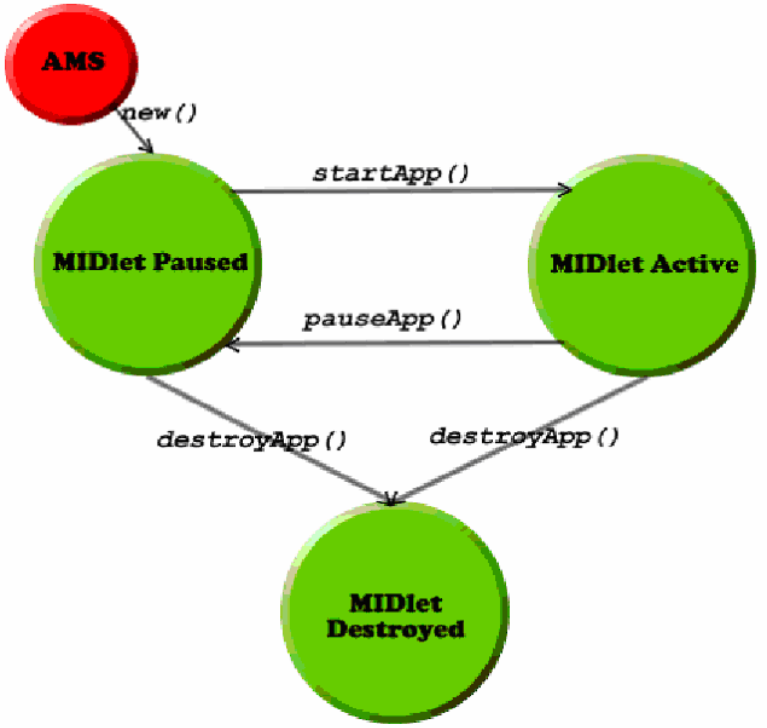
- Tên MIDlet.
- Biểu tượng cho MIDlet này (tùy chọn)
- Tên lớp mà bộ quản lý ứng dụng sẽ gọi tải MIDlet này.

2.7.1.3. Bộ mô tả ứng dụng Java (file JAD)

Ngoài file JAR, một file JAD có thể dùng chứa thông tin về MIDlet. Nhiệm vụ chính của file JAD như sau:

- Cung cấp thông tin cho bộ quản lý ứng dụng về nội dung của một file JAR. Với thông tin này, hệ thống có thể ra những quyết định xem liệu một MIDlet có thích hợp để chạy trên thiết bị hay không?
- Cung cấp phương tiện chuyển tham số cho một MIDlet mà không phải thực hiện thay đổi cho file JAR. Bộ quản lý ứng dụng yêu cầu file JAD phải có tên mở rộng là .jad.

2.7.2. Vòng đời của MIDlet



Hình 2.2. Vòng đời của MIDlet

Một MIDlet đi qua nhiều chu trình của vòng đời hoạt động và luôn ở một trong ba trạng thái sau:

- Paused (tạm ngừng): một MIDlet được đặt trong trạng thái Paused sau khi phương thức khởi tạo đã được gọi, nhưng trước khi được khởi động bởi bộ quản lý ứng dụng. Khi MIDlet đã được khởi động, nó có thể chuyển đổi xen kẽ giữa trạng thái Paused và Active (kích hoạt) bất kỳ thời điểm nào trong suốt vòng đời của nó.
- Active (kích hoạt): MIDlet đang chạy.

- Destroyed (hủy): MIDlet chấmdứt, giải phóng tài nguyên mà nó đang giữ và bị đóng lại bởi bộ quản lý ứng dụng.

2.7.3. Tạo ra một MIDlet

Một MIDlet được tạo ra bằng cách kế thừa lớp MIDlet. Đây là lớp trừu tượng và bao gồm ba phương thức trừu tượng là destroyApp(), pauseApp() và startApp(). Dưới đây là một bộ khung vỏ tạo nên MIDlet. Nó bao gồm tất cả các phương thức yêu cầu bởi MIDlet.

```
public class Shell extends MIDlet
{
public Shell( )
{
}
//gọi bởi bộ quản lý ứng dụng để khởi động MIDlet
public void startApp( )
{
}

//gọi bởi bộ quản lý ứng dụng trước khi tạm ngừng MIDlet
public void pauseApp( )
{
}
//gọi bởi bộ quản lý ứng dụng trước khi shutdown
public void destroyApp(boolean unconditional)
 {
 }
}
```

2.7.4. MIDlet API

MIDlet class: javax.microedition.midlet.MIDlet

Phương thức	Mô tả
Giao tiếp từ bộ quản lý ứng dụng đến MIDlet	
abstract void destroyApp(boonlean unconditional)	MIDlet chuẩn bị shutdown
abstrach void pauseApp( )	MIDlet chuẩn bị tạm dừng
abstract void startApp( )	MIDlet được đặt vào trạng thái kích hoạt
Giao tiếp từ MIDlet đến bộ quản lý ứng dụng	
final void notifyDestroyed( )	MIDlet yêu cầu được shutdown
final void notifyPause( )	MIDlet yêu cầu được tạm dừng
final void resumeRequest( )	MIDlet yêu cầu được kích hoạt
Các thuộc tính yêu cầu từ MIDlet đến bộ quản lý ứng dụng	
final String getAppProperty(String key)	Lấy thuộc tính từ file JAR hoặc JAD

2.7.5. Giao tiếp từ bộ quản lý ứng dụng

Khi một MIDlet sắp sửa được đặt vào trạng thái hoạt động, bộ quản lý ứng dụng sẽ gọi startApp(). Phương thức này có thể được gọi suốt vòng đời của một MIDlet. Phương thức pauseApp() thông báo cho MIDlet sắp sửa được đặt vào trạng thái tạm ngừng. Lúc này MIDlet có thể tranh thủ giải phóng những tài nguyên chưa cần đến. Phương thức destroyApp() báo hiệu cho MIDlet ứng dụng sắp sửa chấm dứt. Bất kỳ những tài nguyên nào còn đang sử dụng bởi MIDlet cần phải giải phóng vào thời điểm này.

2.7.6. Giao tiếp tới bộ quản lý ứng dụng

Khi một MIDlet cần đóng lại (chấm dứt), phương thức notifyDestroyed() có thể được gọi để báo hiệu cho bộ quản lý ứng dụng yêu cầu này. Tuần tự có thể diễn ra như sau:

- Người gọi yêu cầu thoát.
- Gọi destroyApp() để quét dọn bất kỳ những tài nguyên nào còn đang dùng.
- Gọi notifyDestroyed() để thông báo cho bộ quản lý ứng dụng có thể đóng MIDlet an toàn. Điều quan trọng cần nhớ trước khi notifyDestroyed() là MIDlet phải có trách nhiệm giải phóng bất kỳ tài nguyên nào nó đang sử dụng.

Nếu một MIDlet được đặt vào trạng thái tạm ngừng, notyfyPaused()sẽ gửi yêu cầu cho bộ quản lý ứng dụng. Khi đang ttrong ttrạng thái tạm ngừng, resumeRequest() có thể thông báo cho bộ quản lý ứng dụng rằng MIDlet sẵn sàng kích hoạt lại.

2.7.7. Truy vấn thuộc tính MIDlet

Hàm getAppProperty() có thể yêu cầu bộ quản lý ứng dụng truy vấn thuộc tính từ file JAD và file manifest.

CHƯƠNG 3: BỘ CÔNG CỤ KHÔNG DÂY J2ME

Việc biên dịch, tiền xác minh mã và đóng gói các MIDlet từ dòng lệnh thường phức tạp nhất là với các dự án lớn. Sun đã tạo ra bộ công cụ không dây J2ME (J2ME Wireless Toolkit) để hỗ trợ việc này. Chương này sẽ giới thiệu cách sử dụng bộ công cụ này để tạo các ứng dụng.

3.1. Giới thiệu

Bộ công cụ không dây J2ME là một tập các công cụ để dễ dàng tạo các ứng dụng cho điện thoại di động và các thiết bị không dây khác.

3.1.1. Các công cụ trong bộ công cụ

Bộ công cụ không dây J2ME gồm có ba phần chính:

- Ktoolbar tự động hóa nhiề công việc phức tạp trong việc tạo các ứng dụng MIDP.
- Bộ mô phỏng là một điện thoại di động mô phỏng. nó rất hữu ích trong việc thử các ứng dụng.
- Một tập các tiện ích cung cấp các tính năng hữu ích khác.

3.1.2. Đặc điểm bộ công cụ

Bộ công cụ không dây J2ME hỗ trợ việc tạo các ứng dụng MIDP với các đặc điểm chính sau:

- Xây dựng và đóng gói: lập trình viên chỉ việc viết mã nguồn, phần còn lại do bộ công cụ thực hiện. Chỉ việc bấm một nút là bộ công cụ sẽ dịch mã nguồn, tiền xác minh các file lớp và đóng gói một bộ MIDlet.
- Chạy và giám sát: một bộ MIDlet có theer chạy trên bộ mô phỏng hoặc là cài đặt nó sử dụng một phương pháp giống với việc cài đặt ứng dụng trên thiết bị thực.

- Đánh dấu bộ MIDlet: bộ công cụ chứa những công cụ để ghi mật mã vào các MIDlet. Điều này rất hữu ích để kiểm tra sự hoạt động của MIDlet trong các vùng bảo vệ khác nhau.

3.1.3. Các công nghệ hỗ trợ

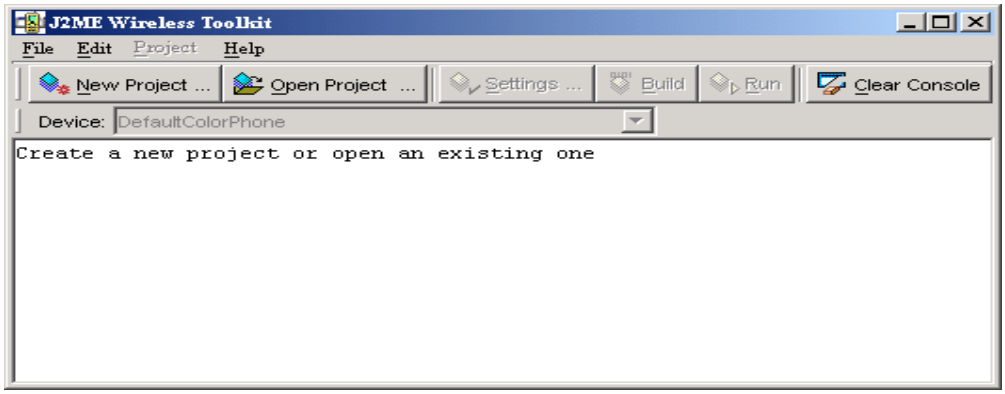
J2ME Wireless Toolkit hỗ trợ các API sau:

- CLDC 1.1
- MIDP 2.0
- JTWI 1.0 : công nghệ Java cho công nghiệp không dây.
- WMA 2.0 : nhắn tin không dây.
- MMAPI 1.1: API phương tiện di động.
- PIM : gói tùy chọn cho J2ME.
- Bluetooth
- 3D graphic

3.2. Phát triển các bộ MIDlet

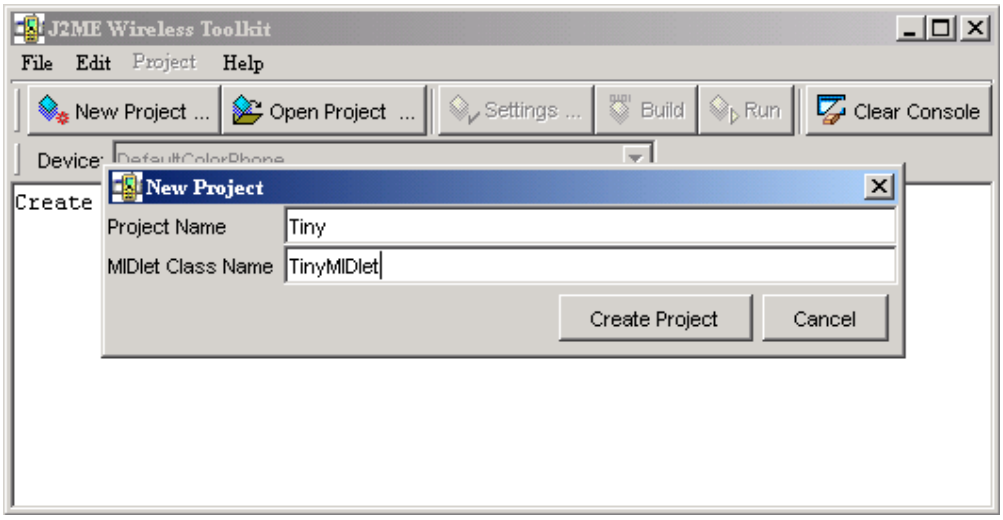
3.2.1. Dự án (Project)

Trong J2ME Wireless Toolkit các bộ MIDlet được tổ chức vào các "dự án". Một dự án chứa tất cả các file mà sẽ sử dụng để xây dựng một bộ MIDlet, bao gồm các file Java nguồn, các file tài nnguyên và bộ mô tả MIDlet. Để tạo một project mới trước hết phải chạy KToolbar.



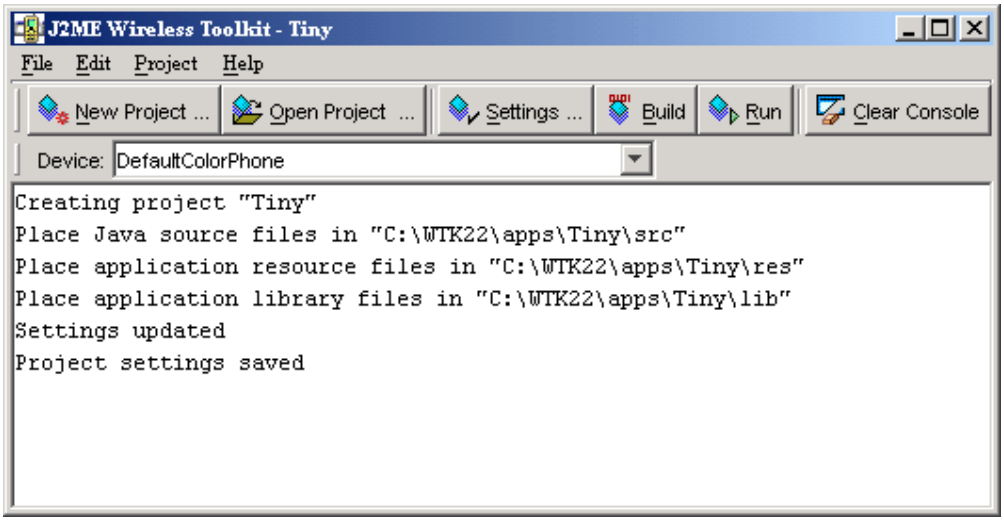
Hình 3.1. Màn hình KToolbar

Sau đó bấm chuột vào nút “New Project”, màn hình sẽ hiện ra như hình 3.2. Nhập tên dự án và lớp vào ô tương ứng.



Hình 3.2 Tạo một dự án mới

Các tùy chọn của dự án mới sẽ tự động hiện lên để thiết lập môi trường môi trường cho dự án. Trên màn hình sẽ có thông báo chính xác các thư mục để copy các file nguồn và tài nguyên cho dự án vào.



Hình 3.3. Vị trí các file

3.2.2. Quy trình phát triển đơn giản

Quy trình phát triển đơn giản như sau:

Viết mã nguồn → xây dựng → chạy

- Viết mã nguồn: tạo các file nguồn Java và các file tài nguyên mà được sử dụng trong ứng dụng. bước này không được J2ME Wireless Toolkit hỗ trợ mà phải sử dụng bộ soạn thảo văn bản.
- Xây dựng (Build): J2ME Wireless Toolkit dịch và tiền xác minh các file nguồn java.
- Chạy (Run): Các file lớp Java đã được dịch chạy trên bộ mô phỏng.

Nếu có lỗi xảy ra khi bộ công cụ dịch các file nguồn thì phải quay lại sửa mã nguồn. Nếu tìm thấy lỗi khi chạy thử chương trình trên bộ mô phỏng thì phải sửa mã nguồn để sửa lỗi đó.

3.2.3. Quy trình phát triển đầy đủ

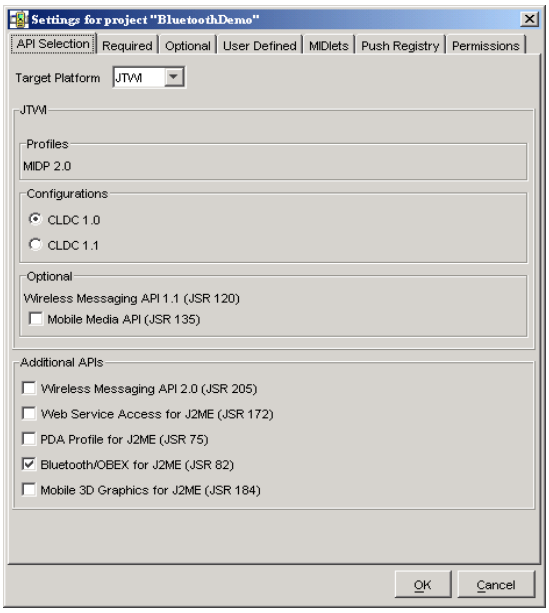
Quy trình này phức tạp hơn quy trình trên một chút:

Viết mã nguồn → Đóng gói → Cài đặt → Chạy

- Viết mã nguồn: giống như quy trình trên.
- Đóng gói (Package): J2ME Wireless Toolkit dịch và tiền xác minh các file nguồn (giống như bước xây dựng ở trên). Sau đó nó đóng gói các file lớp và các file tài nguyên vào một file JAR và file JAD.
- Cài đặt: các bộ MIDlet cần phải được cài đặt trước khi chạy.

3.3. Làm việc với các project

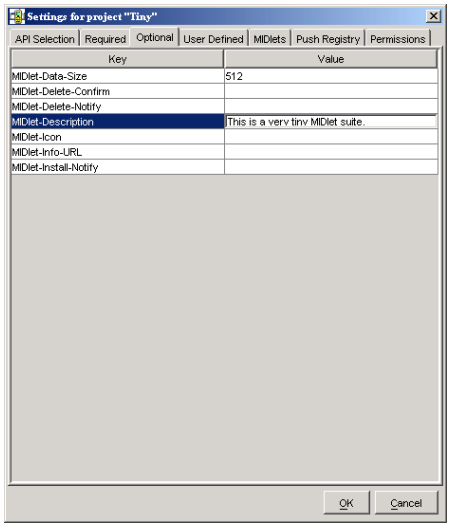
3.3.1. Lựa chọn các API



Hình 3.4. Cửa sổ chọn API

3.3.2. Thay đổi các thuộc tínhcủa bộ MIDlet

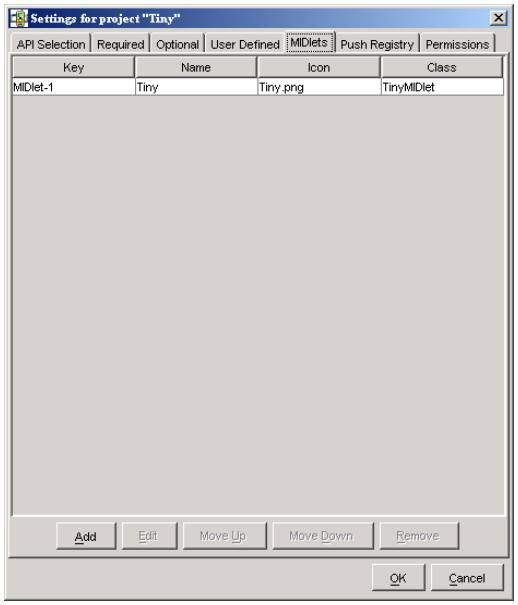
Lập trình viên có thể thay đổi các thuộc tính lưu trữ trong các file JAD và manifest.



Hình 3.5 Thay đổi các thuộc tính của MIDlet

3.3.3. Thao tác MIDlet

Các thiết lập dự án cung cấp cách thêm vào hoặc sửa đổi các MIDlet chứa trong dự án các bộ MIDlet hiện tại.



Hình 3.6. Danh sách các MIDlet trong một dự án

3.3.4. Cấu trúc thư mục dự án

Bên trong mỗi thư mục dự án có các thư mục con sau:

- bin : Chứa các file JAD, JAR sau khi đóng gói dự án.
- classes : chứa các file lớp đã dịch.
- lib : chứa thư viện của bên thứ 3 sử dụng trong dự án.
- res : chứa các file tài nguyên.
- src : chứa các file nguồn
- tmpclasses : được sử dụng bởi bộ công cụ
- tmpsrc : được sử dụng bởi bộ công cụ

3.3.5. Sử dụng các thư viện của bên thứ ba

Bộ công cụ cho phép lập trình viên kết hợp các thư viện của bên thứ ba vào trong các ứng dụng của họ. sử dụng các thư viện của bên thứ ba có thể

làm giảm thời gian phát triển bằng cách cung cấp các chức năng mà lập trình viên không muốn tự xây dựng.

Khi sử dụng một thư viện của bên thứ ba vào ứng dụng thì file JAR sẽ bị tăng thêm kích thước của thư viện đó.

3.3.5.1. Các thư viện của bên thứ ba cho một project

Bất cứ file thư viện nào đặt trong thư mục lib của dự án sẽ được đưa vào xây dựng và đóng gói dự án. Các thư viện nên là các file JAR hoặc Zip của các lớp Java.

3.3.5.2. *Các thư viện của bên thứ ba cho tất cả project*

Một số thiết bị có sẵn các thư viện cho tất cả các bộ MIDlet cài đặt. Ví dụ nhà sản xuất có thể đưa các API vào tất cả thiết bị của họ. trong trường hợp này lập trình viên muốn có thể sử dụng các thư viện này khi xây dựng và test ứng dụng của họ. họ không muốn đưa các thư viện này vào bộ MIDlet của họ sẽ cài đặt bộ MIDlet này lên thiết bị mà đã có sẵn các thư viện này.

Để thực hiện điều này thì đặt các thư viện vào trong thư mục \apps\lib trong thư mục cài đặt bộ công cụ. Tất cả thư viện trong thư mục này sẽ dành cho tất cả các dự án.

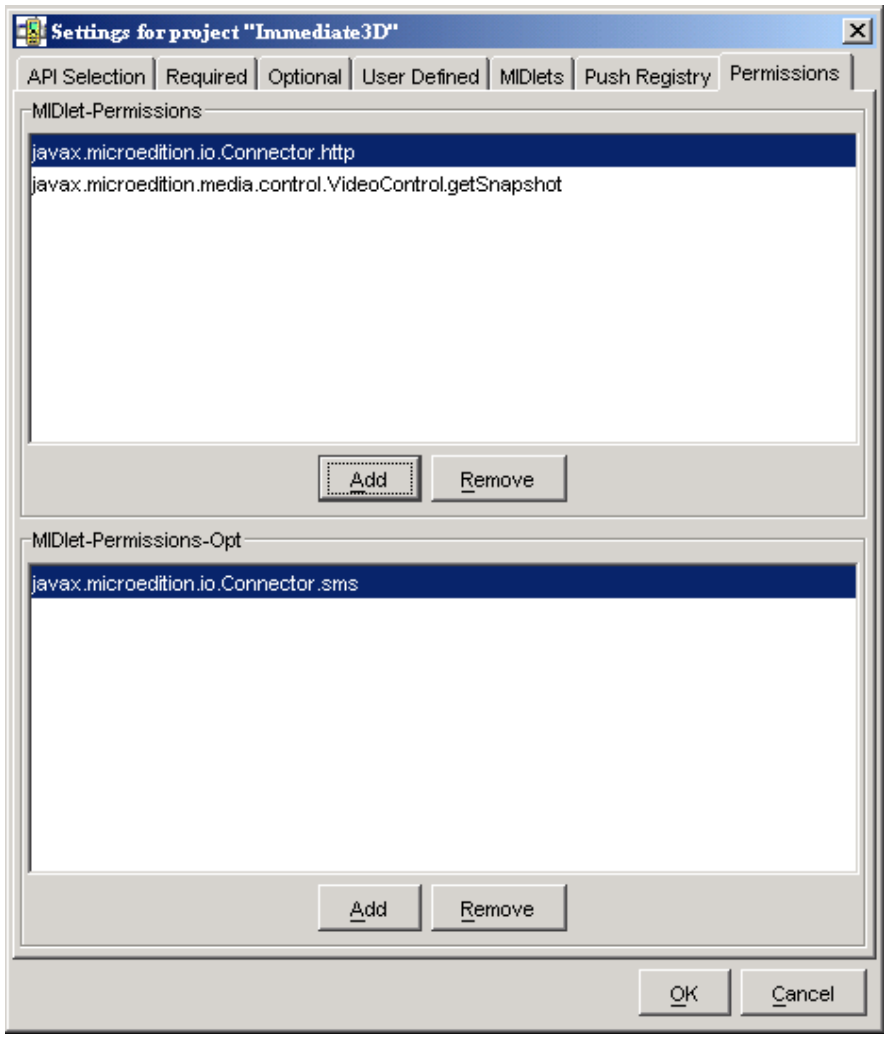
3.4. An toàn và đánh dấu MIDlet

MIDP 2.0 có một mô hình an toàn toàn diện dựa vào các vùng bảo vệ. Các bộ MIDlet được cài đặt vào một vùng bảo vệ mà xác định truy nhập tới các chức năng bảo vệ. Đặc tả MIDP 2.0 cũng bao gồm khuyên nghị cho việc sử dụng khóa mã chung để xác nhận và xác thực MIDlet.

3.4.1. Sự cho phép (permission)

MIDlet phải có sự cho phép thực hiện các hoạt động nhạy cảm như kết nối mạng. Sự cho phép có các tên rõ ràng, và các bộ NIDlet có thể chỉ ra sự cần thiết của chúng với những loại nhất định của sự cho phép qua các thuộc tính trong bộ mô tả MIDlet.

Hình 3.4 cho ta thấy sự cho phép các MIDlet:



Hình 3.7. Sự cho phép MIDlet

3.4.2. Các vùng bảo vệ (protect domain)

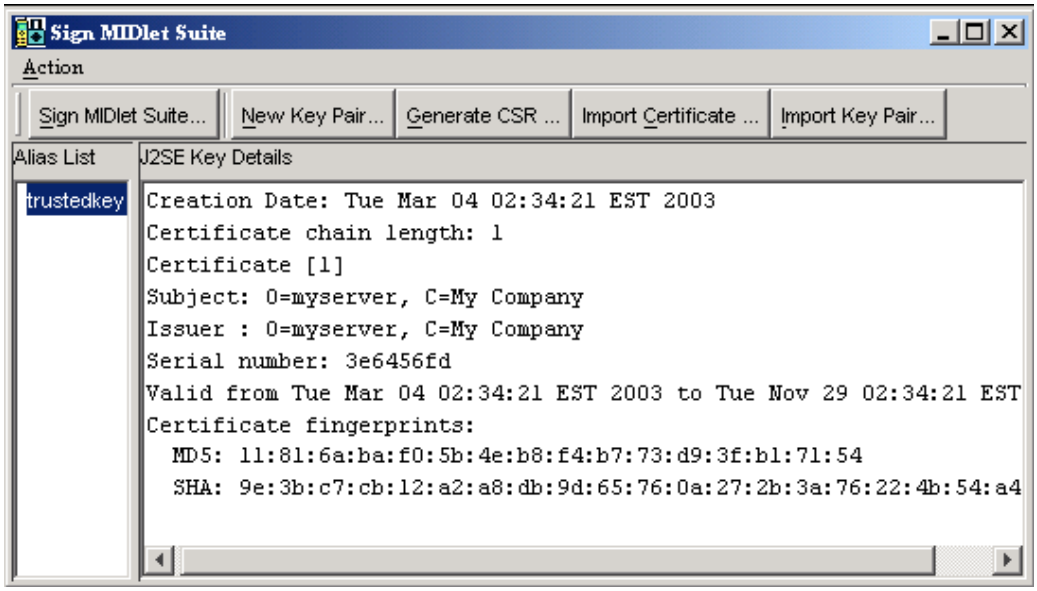
Bộ công cụ có bốn vùng bảo vệ :

- MIDlet trong vùng tối thiểu bị cấm tất cả.
- Vùng không tin tưởng cung cấp một mức cao an toàn cho các ứng dụng mà nguồn và tính xác thực không được xác định. Người sử dụng thường xuyên được hỏi khi ứng dụng muốn thực hiện một hoạt động nhạy cảm.
- Vùng tin tưởng : các MIDlet có tất cả quyền.

- Vùng cực đại tương đương vùng tin tưởng.

3.4.3. Đánh dấu một bộ MIDlet

Để đánh dấu một bộ MIDlet, trước hết phải đóng gói nó. Sau đó chọn Project → Sign từ thực đơn của KToolbar. Màn hình đánh dấu xuất hiện:



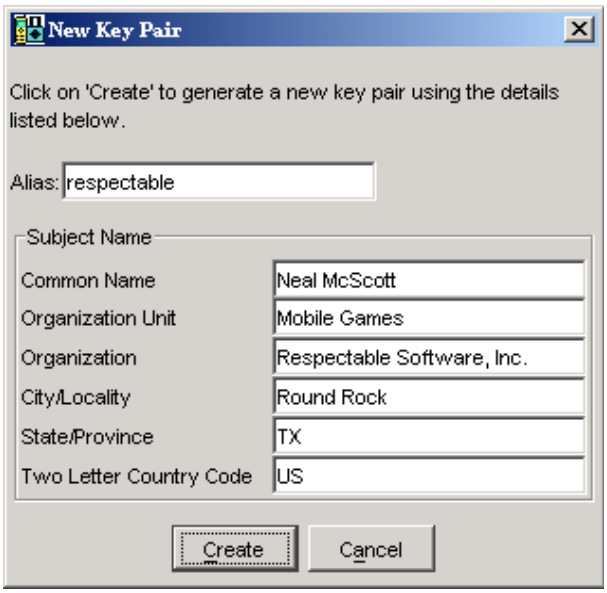
Hình 3.8. Đánh dấu MIDlet

3.4.4. Quản lý khóa

Cửa sổ đánh dấu MIDlet có theer sử dụng để quản lý các khóa.

3.4.4.1. Tạo một cặp khóa mới

Để tạo một cặp khóa mới, nhấp chuột vào "New Key Pair". Bộ công cụ sẽ hỏi cho một bí danh khóa và thông tin sẽ liên kết với cặp khóa.



Hình 3.9. Tạo cặp khóa mới

Sau khi chọn "Create", bộ công cụ sẽ hỏi chọn một vùng bảo vệ. kết nối giữa cặp khóa và vùng bảo vệ dường như có dạng chéo, nhưng nó làm cho hoàn hảo:

- Bộ công cụ tạo một chứng chỉ gốc tự ký sử dụng cặp khóa vừa tạo.
- Chứng chỉ gốc này được đưa vào danh sách bộ mô phỏng của các chứng chỉ gốc.
- Bộ công cụ cần liên kết chuỗi chứng chỉ này trong bộ mô tả MIDlet.

Khi cài đặt một MIDlet được đánh dấu với khóa mới sẽ xảy ra:

- Thực hiện kiểm tra chuỗi chứng chỉ trong bộ mô tả MIDlet. Trong trường hợp này chuỗi chứng chỉ là một chuỗi đơn, gốc tự đánh dấu.
- Thực hiện cố gắng tìm gốc của chuỗi chứng chỉ trong danh sách. Điều này sẽ thành công vì chứng chỉ gốc vừa được thêm vào khi tạo cặp khóa.

- Thực hiện cân nhắc chứng chỉ hợp lệ và sử dụng nó để xác minh chữ ký trên bộ MIDlet.
- Bộ MIDlet được cài đặt vào vùng bảo vệ được chọn.

3.4.4.2. Nhận các khóa thực

Khả năng tạo một cặp khóa và đánh dấu một MIDlet trong môi trường bộ công cụ chỉ để cho chức năng test. Khi chạy chương trình trên một thiết bị thực thì phải nhận một cặp khóa từ một chứng chỉ authority đwocj nhận dạng bởi thiết bị.

Thủ tục đánh dấu các bộ MIDlet với các khóa thực như sau:

1. Tạo một cặp khóa mới.
2. Tạo một yêu cầu đánh dấu chứng chỉ (CSR).
3. Gửi CSR đó tới một certificate authority (CA). CA sẽ cần thêm thông tin để xác nhận định danh.
4. Sau khi CA xác nhận sẽ gửi một chứng chỉ chứng nhận khóa chung.
5. Đưa khóa vào bộ công cụ bằng cách chọn “Import Certificate” trong cửa sổ đăng ký MIDlet.

CHƯƠNG 4 : GIAO TIẾP LẬP TRÌNH ỨNG DỤNG CHO J2ME

Chương này giới thiệu đặc tả JSR 180, SIP API cho J2ME. Đặc tả này định nghĩa một gói tuỳ chọn J2ME mà cho phép các thiết bị hạn chế tài nguyên (sau đây gọi là thiết bị đầu cuối) gửi và nhận các bản tin SIP. API này cung cấp chức năng SIP ở mức giao dịch. SIP API được sử dụng bởi các chương trình ứng dụng để thực hiện các chức năng SIP UA.

Các lớp và giao diện trong SIP API:

Lớp / giao diện	Chức năng
SipClientConnection	SipClientConnection mô tả giao dịch SIP client.
SipClientConnectionListener	Giao diện nghe các phúc đáp SIP đến.
SipConnection	SipConnection là giao diện cơ sở cho các kết nố SIP.
SipConnectionNotifier	Giao diện này định nghĩa một thông báo kết nối SIP server.
SipDialog	SipDialog mô tả một SIP Dialog.
SipRefreshListener	Giao diện nghe các sự kiện RefreshHelpe.
SipServerConnection	SipServerConnection mô tả giao dịch SIP server.
SipServerConnectionListener	Giao diện nghe các yêu cầu SIP gửi đến.
SipAddress	SipAddress cung cấp một bộ phân tích địa chỉ SIP chung.
SipHeader	SipHeader cung cấp bộ trợ giúp phân tích đầu đề SIP.
SipRefreshHelper	Lớp này thực hiện chức năng làm thuận tiện việc xử lý làm tươi yêu cầu của ứng.
SipException	Đây là một lớp ngoại lệ cho các lỗi SIP cụ thể.

4.1. SipConnection

SipConnection là giao diện cơ bản cho các kết nối SIP. SipConnection giữ các thuộc tính và chung cho các giao diện con SipClientConnection và SipServerConnection.

Khai báo :

public interface SipConnection extends javax.microedition.io.Connection

Các giao diện cha :

javax.microedition.io.Connection

Các giao diện con :

SipClientConnection, SipServerConnection

4.2. Tích hợp vào khung kết nối chung

Tích hợp vào javax.microedition.io.Connector

Một kết nối SIP mới được tạo ra bởi Connector.open(). Một ứng dụng phải gọi close ( ) khi muốn kết thúc kết nối. Chú ý rằng Connector.open()cho một kết nối chế độ client (SipClientConnection) hoặc kết nối chế độ server (SipConnectionNotifier) tuỳ thuộc vào chuỗi (SIP URI) trong Connector.open(). SIP URI được định nghĩa trong RFC3261. Mẫu chung như sau:

{scheme}:[{target}][{params}]

Trong đó:

- scheme là lược đồ SIP được hỗ trợ bởi hệ thống SIP.
- target là địa chỉ mạng người sử dụng có dạng là {user_name}@{target_host}[:{port}] hoặc {telephone_number}
- params: các tham số khác cho SIP URI ví dụ như transport=udp

4.3. Định tuyến yêu cầu gửi đến

Luật định tuyến sau dựa vào thông tin trong yêu cầu gửi đến. Việc thực hiện có thể định nghĩa chính sách địa phương mà thay thế các luật này vì lý do an toàn. Yêu cầu gửi đến được định tuyến dựa vào các thông tin sau:

- Trường đầu đề Accept-Contact
- Trường đầu đề Reject-Contact
- SDP media : tên phương tiện và cổng địa.

Yêu cầu được định tuyến theo luật sau, dựa vào thông tin nào ở trên có sẵn. Tất cả yêu cầu có thể có các trường đầu đề Accept-Contact và Reject-Contact. Yêu cầu INVITE có thể cũng có tải SDP.

	Accept - Contact	SDP media	Routing
1)	Có	-	Trường đầu đề chứa tập các thuộc tính mô tả các UA mà chủ gọi muốn gọi đến. Trong đặc tả này yêu cầu tối thiểu là ứng dụng kiểu MIME được chỉ định bởi tham số “type”
2)	Có	Có	Giống trường hợp 1)
3)	-	Có	(Tuỳ chọn) Yêu cầu được định tuyến dựa vào thông tin phương tiện SDP.

Luật 1) định nghĩa yêu cầu tối thiểu hoạt động cho đặc tả này. Người gửi SIP phải thực hiện luật quyết định định nghĩa trong đặc tả RFC 3841. Vì các lý do an toàn người gửi có thể định nghĩa các luật định tuyến mà không bị thay thế bởi các ứng dụng khác.

4.4. SipClientConnection

Khai báo:

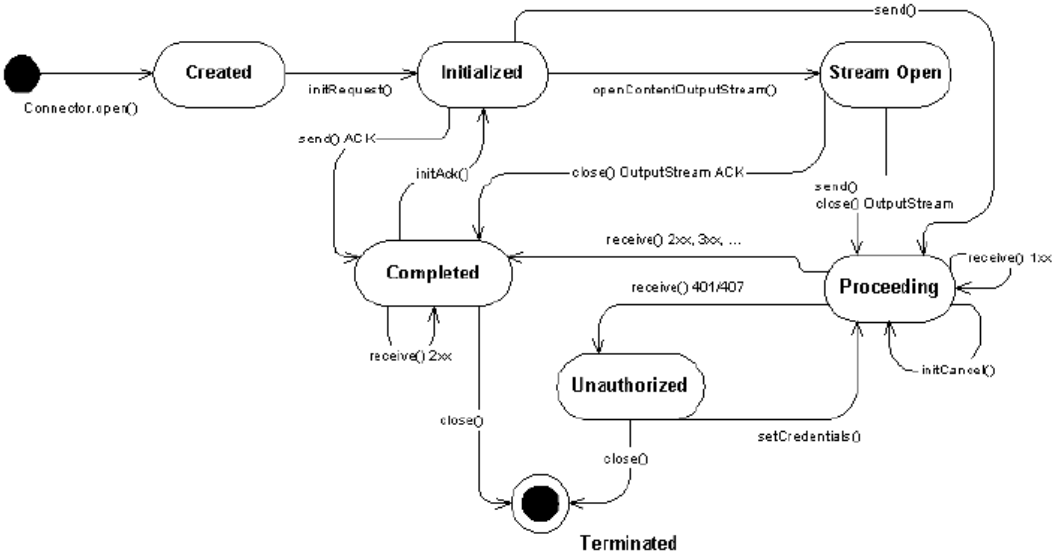
public interface SipClientConnection extends SipConnection

Các giao diện cha :

javax.microedition.io.Connection, SipConnection

SipClientConnection đại diện cho giao dịch SIP client. Ứng dụng có thể tạo SipClientConnection mới với đối tượng Connector hoặc SipDialog.

SipClientConnection có biểu đồ trạng thái như hình 4.1:



Hình 4.1. Biểu đồ trạng thái SipClientConnection

Các trạng thái:

- Created: SipClientConnection được tạo bởi Connector.
- Initialized: yêu cầu được khởi tạo với initRequest(...) hoặc initAck hoặc initCancel() hoặc SipDialog.getNewClientConnection(...)
- Stream Open: OutputStream được mở với openContentOutputStream(). Mở InputStream để nhận phản hồi không làm thay đổi trạng thái.

- Proceeding: yêu cầu đã được gửi và đang đợi phúc đáp, hoặc là đã nhận phúc đáp tạm thời 1xx. initCancel() có thể được gọi mà sẽ xuất hiện một SipClientConnection mới ở trạng thái Initialized.
- Completed: giao dịch được hoàn thành với phúc đáp cuối cùng (2xx, 3xx, 4xx, 5xx, 6xx). Trong trạng thái này ACK có thể được khởi tạo. Nhiều phúc đáp 200 OK có thể được nhận. Riêng phúc đáp 401 và 407 chuyển trạng thái khác.
- Unauthorized: giao dịch hoàn thành với phúc đáp 401 (Unauthorized) hoặc 407 (Proxy Authentication Required). Ứng dụng có thể bắt đầu lại yêu cầu với các khả năng thích hợpbằng cách gọi setCredentials() method. Sau đó SipClientConnection trở lại trạng thái Proceeding.
- Terminated: trạng thái cuối cùng mà kết nối SIP kết thúc bởi lỗi hay do đóng lại.

4.5. SipServerConnection

Khai báo:

public interface SipServerConnection extends SipConnection

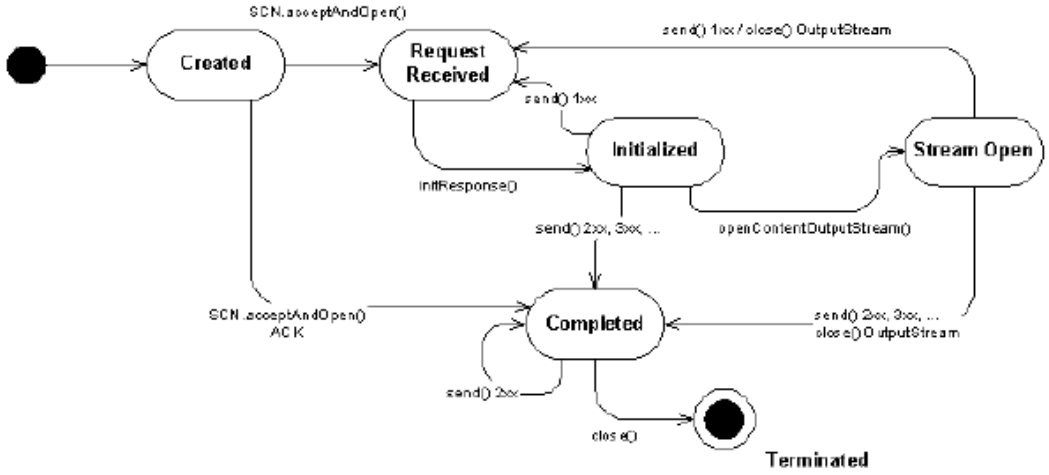
Các giao diện cha:

javax.microedition.io.Connection, SipConnection

SipServerConnection đại diện cho giao dịch SIP server. SipServerConnection được tạo bởi SipConnectionNotifier khi một yêu cầu mới nhận được. Các trạng thái của SipServerConnection:

- Created: SipServerConnection được tạo.
- Request Received: SipServerConnection trả lại từ SipConnectionNotifier (không ACK) hoặc phúc đáp tạm thời (1xx).

- Initialized: phúc đáp được khởi tạo gọi initResponse()
- Stream Open: OutputStream được mở với openContentOutputStream(). Việc mở InputStream để nhận yêu cầu không làm thay đổi trạng thái.
- Completed: giao dịch được hoàn thành với việc gửi phúc đáp cuối cùng (2xx, 3xx, 4xx, 5xx, 6xx) hoặc gửi lại 2xx hoặc SipServerConnection cho ACK từ SipConnectionNotifier.
- Terminated: trạng thái cuối cùng mà kết nối SIP chấm dứt bởi lỗi hoặc được đóng.



Hình 4.2. Biểu đồ trạng thái của SipServerConnection

4.6. SipConnectionNotifier

Khai báo:

public interface SipConnectionNotifier extends javax.microedition.io.Connection

Các giao diện cha:

javax.microedition.io.Connection

Giao diện này định nghĩa một bộ thông báo kết nối SIP server. Kết nối SIP server được mở với Connector.open() sử dụng một chuỗi SIP URI với việc bỏ qua host và user.

4.7. SipClientConnectionListener

Khai báo: *public interface SipClientConnectionListener*

Đây là giao diện nghe các phản hồi SIP gửi đến.

4.8. SipServerConnectionListener

Khai báo: *public interface SipServerConnectionListener*

Đây là giao diện nghe các yêu cầu SIP gửi đến.

4.9. SipDialog

Khai báo: *public interface SipDialog*

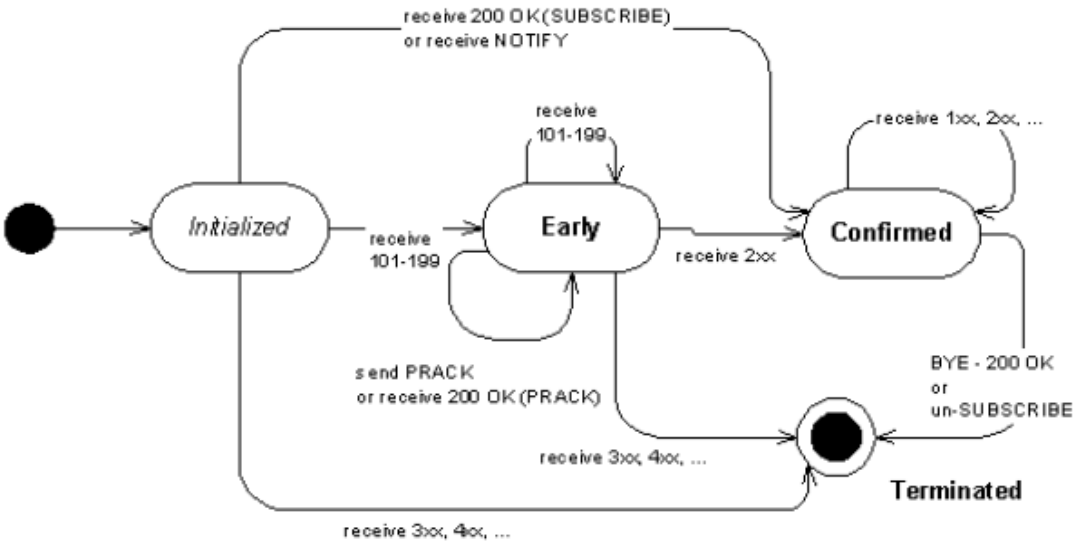
SipDialog diễn đạt một hội thoại SIP. SipDialog có thể nhận được từ một đối tượng SipConnection. Có hai yêu cầu SIP có thể mở một hội thoại:

- INVITE-1xx-2xx-ACK sẽ mở một hội thoại. SipClientConnection trong cùng một hội thoại có thể thu được bằng cách gọi phương thức getNewClientConnection(String method). Hội thoại chấm dứt khi giao dịch BYE-200 OK được hoàn thành.
- SUBSCRIBE-200 OK(hoặc khớp NOTIFY) sẽ mở một hội thoại. Tiếp theo sau SipClientConnection trong cùng hội thoại có thể đạt được bằng cách gọi phương thức getNewClientConnection(String method). Hội thoại chấm dứt khi một bộ thông báo gửi một yêu cầu NOTIFY với một "Subscription-State" của "terminated" và không có các subscription nào sống cùng với hội thoại này.

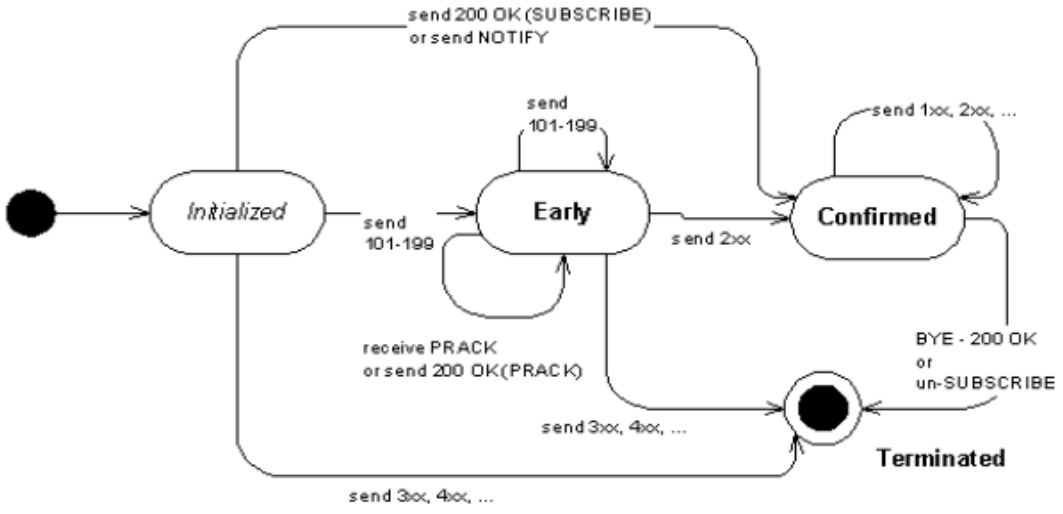
SipDialog có các trạng thái sau (cho cả client và server side):

- Initialized: trạng thái bên trong khi hội thoại được tạo. Trạng thái này không hữu hình với người sử dụng.
- Early: các phúc đáp tạm thời 101-199 nhận được (hoặc gửi)

- Confirmed: phúc đáp cuối cùng 2xx response nhận được (hoặc gửi) cho yêu cầu gốc. Hoặc NOTIFY xác nhận subscription nhận được (hoặc gửi).
- Terminated: không có phúc đáp hoặc phúc đáp lỗi (3xx-6xx) nhận được (hoặc gửi). Hội thoại cũng có thể chấm dứt với BYE hoặc un-SUBSCRIBE.



Hình 4.3. Biểu đồ trạng thái của SipDialog (phía clien)



Hình 4.4. Biểu đồ trạng thái của SipDialog (phía server)

4.10. SipHeader

Khai báo:

public class SipHeader

java.lang.Object

|

+--javax.microedition.sip.SipHeader

SipHeader cung cấp bộ trợ giúp phân tích đầu đề SIP chung. Lớp này có thể được sử dụng để phân tích các giá trị đầu đề ký tự tối thiểu mà đọc từ bản tin SIP sử dụng phương thức SipConnection.getHeader().SipHeader là một lớp trợ giúp riêng rẽ và không bắt buộc sử dụng cho tạo một kết nối SIP. Do đó SIP headers có thể được xây dựng với lớp này. SipHeader sử dụng khuôn dạng chung để phân tích các giá trị header và các tham số.

4.11. SipAddress

Khai báo:

public class SipAddress

java.lang.Object

|

+--javax.microedition.sip.SipAddress

SipAddress cung cấp một bộ phân tích địa chỉ SIP chung. Các địa chỉ hợp lệ có thể được xây dựng với lớp này. SipAddress có các yêu cầu chức năng sau:

- SipAddress không tránh được các chuỗi ký tự địa chỉ.
- SipAddress bỏ qua phần đầu đề của SIP URI.
- Khuôn dạng hợp lệ SipAddress là giống như được định nghĩa trong SIP BNF cho URI tuyệt đối.
- Địa chỉ Contact hợp lệ "*" được chấp nhận trong SipAddress. Trong trường hợp này các thuộc tính sẽ là null và số cổng sẽ là 0.

4.12. SipRefreshHelper

Khai báo:

public class SipRefreshHelper

java.lang.Object

|

+--javax.microedition.sip.SipRefreshHelper

Lớp này thực hiện chức năng làm cho thuận tiện việc làm tươi các yêu cầu của ứng dụng. một số yêu cầu SIP (REGISTER, SUBSCRIBE, ...) cần phải được làm tươi đúng lúc. Ví dụ yêu cầu REGISTER cần phải được gửi lại để đảm bảo điểm khởi tạo vẫn hoạt động. Tính hợp lệ của yêu cầu được đề ra bởi điểm cuối trong yêu cầu và được xác nhận trong phúc đáp bởi registrar/notifier ví dụ trong expires header. Việc xử lý này sẽ làm tăng đáng kể độ phức tạp và kích thước chương trình. SipRefreshHelper có thể đwocj sử dụng để làm cho dễ dàng các công việc này. Khi ứng dụng muốn gửi một yêu cầu có thể làm tươi nó sẽ:

- Thực hiện SipRefreshListener gọi lại giao diện.
- Tạo một SipClientConnection mới và thiết lập nó.
- Gọi phương thức enableRefresh(SipRefreshListener).
- Nếu công việc làm tươi bị lỗi thì một sự kiện lỗi sẽ được gửi đến SipRefreshListener

Một tham chiếu tới đối tượng SipRefreshHelper thu được bằng cách gọi phương thức tĩnh SipRefreshHelper.getInstance(). Cuối cùng sử dụng mã làm tươi từ enableRefresh(SipRefreshListener) ứng dụng có thể:

- Dừng làm tươi: sự liên kết giữa điểm cuối và server bị hủy bỏ.
- Cập nhật làm tươi với các tham số mới

4.13. SipRefreshListener

Khai báo: *public interface SipRefreshListener*

SipRefreshListener là giao diện nghe các sự kiện RefreshHelpe. Giao diện này định nghĩa một sự kiện mà chứa một refreshID để nhận dạng một công việc làm tươi tương ứng, statusCode diễn tả kết quả của quá trình làm tươi này (0 = bị hủy, 200 = thành công, còn lại = không thành công). statusCode tương ứng với phúc đáp nhận được cho yêu cầu nguyên thủy được gửi bởi SipRefreshHelper. reasonPhrase cho một bản tin nguyên bản về thành công hay không của công việc làm tươi này.

4.14. SipException

Khai báo:

public class SipException extends java.io.IOException

java.lang.Object

```
 |
 +--java.lang.Throwable
     |
     +--java.lang.Exception
         |
         +--java.io.IOException
             |
             +--javax.microedition.sip.SipException
```

Đây là một lớp ngoại lệ cho các lỗi cụ thể SIP. Ngoại lệ bao gồm bản tin lỗi nguyên bản khuôn dạng tự do và mã lỗi để phân loại lỗi.

CHƯƠNG 5 : LẬP CHƯƠNG TRÌNH

5.1. Điều kiện thực hiện chương trình

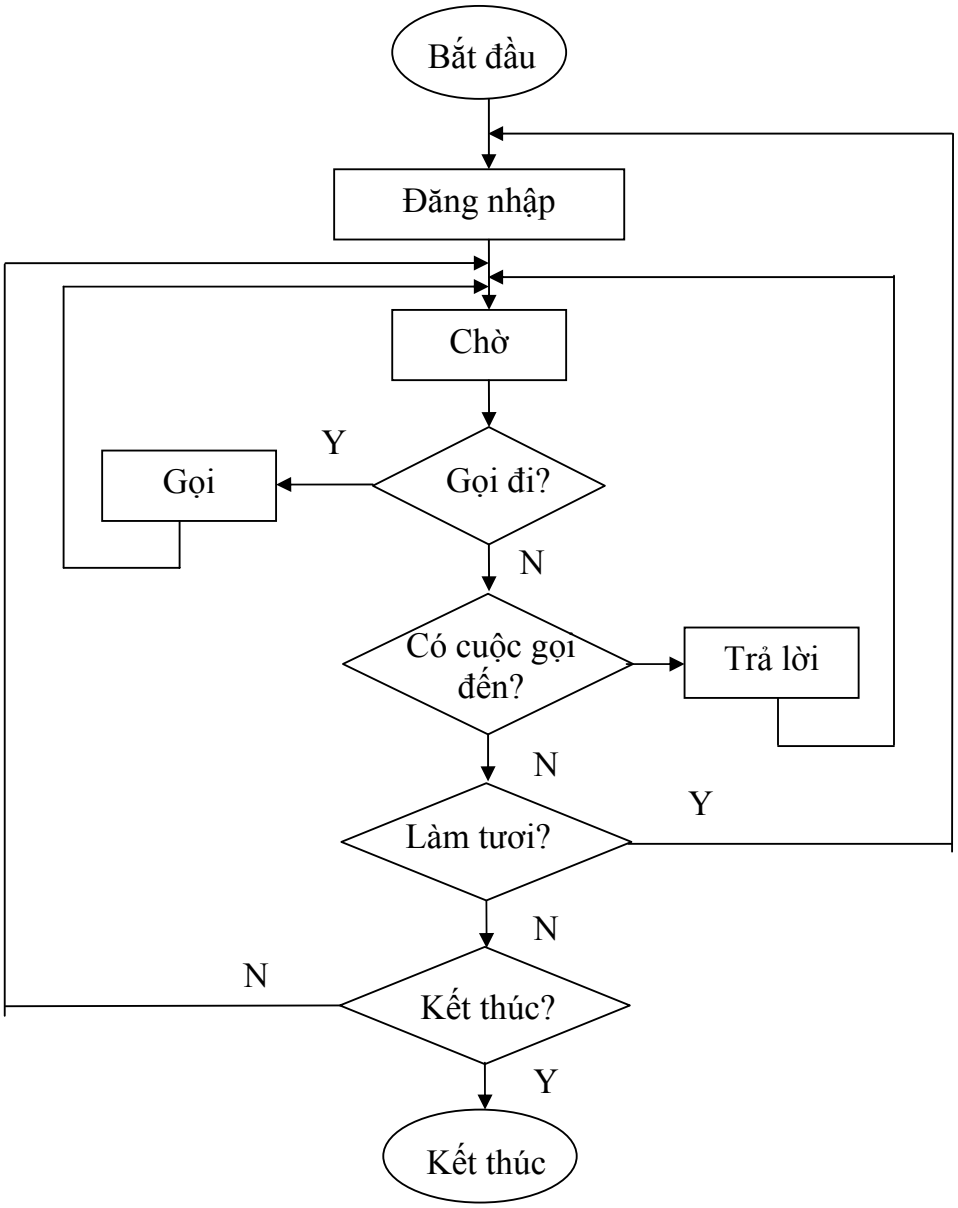
Để thực hiện được chương trình cần phải có những điều kiện sau:

- Thiết bị di động phải có cài đặt CLDC và MIDP. Hiện nay các loại điện thoại di động phổ biến cài đặt CLDC 1.1 và MIDP 2.0. Ngoài ra thiết bị phải có kkhả năng kết nối mạng, có đủ bộ nhớ để cài đặt và thực hiện chương trình.
- Đã có các SIP server (proxy server, registrar server, redirect server).
- Có tài khoản của SIP server trên.

5.2. Thuật toán chương trình

Các bước của thuật toán:

- Khi bắt đầu chạy chương trình, thiết bị đăng nhập vào SIP server.
- Sau đó thiết bị vào trạng thái chờ gọi đến.
- Nếu có cuộc gọi đến thì trả lời hoặc không trả lời.
- Nếu muốn gọi đi thì chuyển sang chế độ gọi đi.
- Nếu đến thời gian phải làm tươi thì thực hiện lại đăng nhập tới server.
- Nếu muốn kết thúc chương trình thì đóng chương trình và giải phóng bộ nhớ.



Hình 5.1. Lưu đồ thuật toán

5.3. Đăng nhập SIP

Khi bắt đầu chạy, chương trình phải thực hiện đăng nhập vào mạng SIP bằng cách gửi bản tin REGISTER. Trước tiên phải nhập tên và mật khẩu. Các bước thực hiện như sau:

+ Ứng dụng khởi tạo yêu cầu REGISTER gốc.

+ Ứng dụng gọi send()

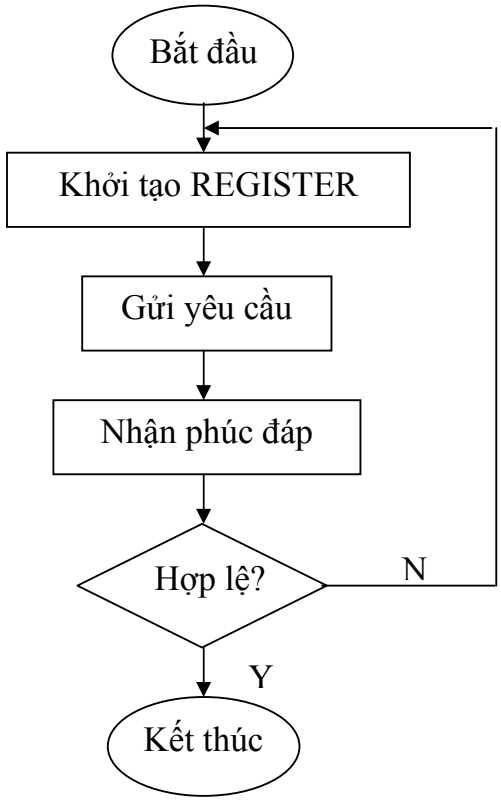
+ SIP API thực hiện gửi yêu cầu REGISTER đến server.

+ Ứng dụng gọi receive( ) để đợi phúc đáp.

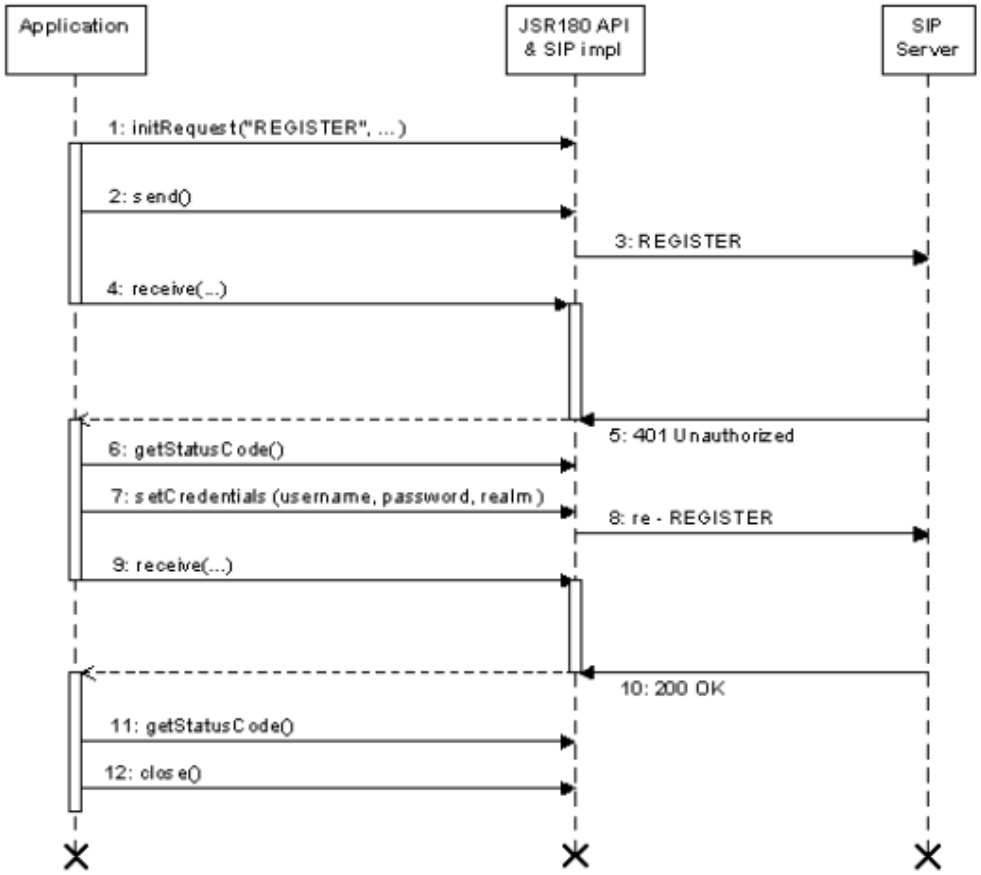
+ Nếu tên và mật khẩu không hợp lệ thì SIP server gửi phúc đáp "401 Unauthorized". Nếu hợp lệ thì SIP server gửi phúc đáp "200 OK".

+ Ứng dụng gọi getStatusCode().

+ Ứng dụng kết thúc kết nối bằng phương thức close( ).



Hình 5.2. Thuật toán đăng nhập SIP



Hình 5.3. Quá trình thực hiện đăng nhập

Mã chương trình:

```
    public void doRegister(String username, String password, String realm)
{
    SipClientConnection scc = null;
    SipConnectionNotifier scn =
    String contact =
    try {
        // open listener in application specific port 5080
        scn = (SipConnectionNotifier) Connector.open("sip:5080");
```

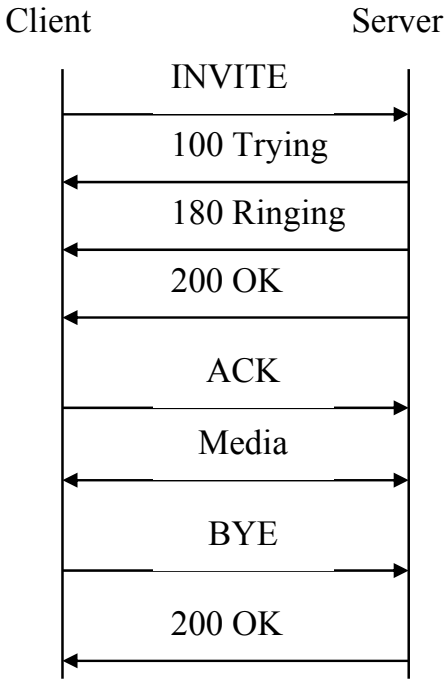
```
        // build the contact URI
        contact =
    new String("sip:user@"+scn.getLocalAddress()+":"+scn.getLocalPort(
));
     open client connection to the SIP registrar in this case "host.com"
        scc =SipClientConnection)open("sip:host.com");
     initialize REGISTER with appropriate headers
        scc.initRequest("REGISTER", scn);
        scc.setHeader("From", "sip:user@host.com");
    setHeader("To",sip:user@host.com");
    setHeader("Contact", contact);
        scc.send();
        boolean handled = false;
        int scode = 0;
        while(!handled) {
           SipHeader sh;
           // wait max 30 secs for response
           scc.receive(30000);
           scode = scc.getStatusCode();
           switch(scode)
           {
              case 401:
                 sh = new SipHeader("WWW-Authenticate",
                   scc.getHeader("WWW-Authenticate"));
                 realm = sh.getParameter("realm");
                 // here for example, prompt user for password for this realm
                 // set credentials to initiate re-REGISTER
```

```
                scc.setCredentials(username, password, realm);
                break;
            case 407:
                sh = new SipHeader("Proxy-Authenticate",
                  scc.getHeader("Proxy-Authenticate"));
                realm = sh.getParameter("realm");
                // here for example, prompt user for password for this realm
                // set credentials to initiate re-REGISTER
                scc.setCredentials(username, password, realm);
                break;
            case 200:
                // handle OK response
                handled = true;
            default:
handle other responses
                handled =
        }
    }
    scc.close();
 } catch(Exception ex) {
    // handle Exceptions
 }
}
```

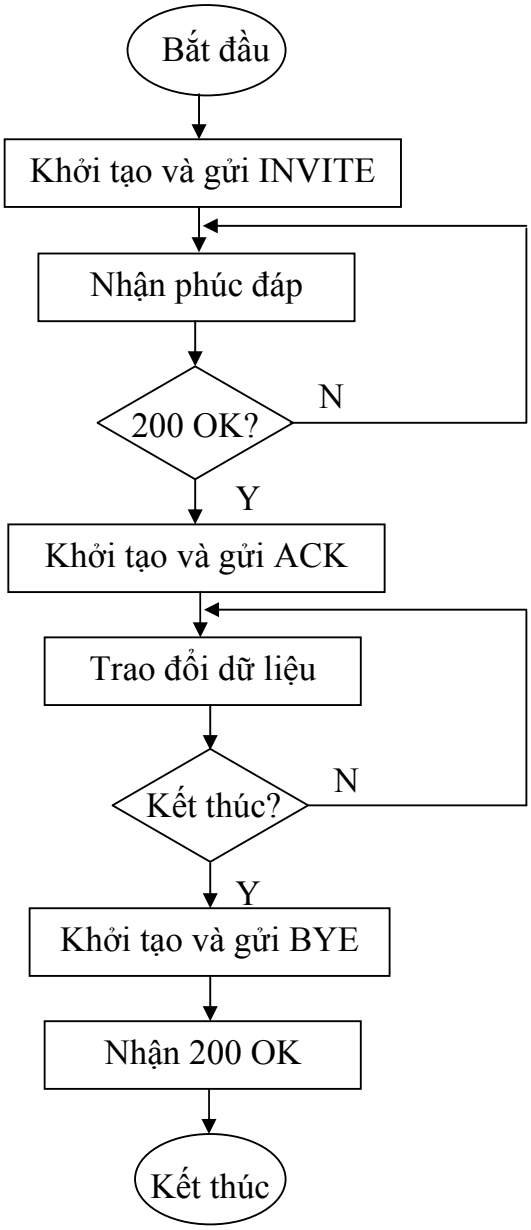
5.4. Gọi đi

Quy trình gọi đi như sau:

- Khởi tạo và gửi bản tin INVITE
- Nhận bản tin phúc đáp tạm thời 100, 180 cho đến khi nhận bản tin cuối cùng 200 OK.
- Khởi tạo và gửi bản tin ACK để thiết lập phiên.
- Trao đổi dữ liệu
- Khởi tạo và gửi bản tin BYE.
- Nhận bản tin 200 OK cho BYE, kết thúc phiên.



Hình 5.4. Quá trình thực hiện gọi đi

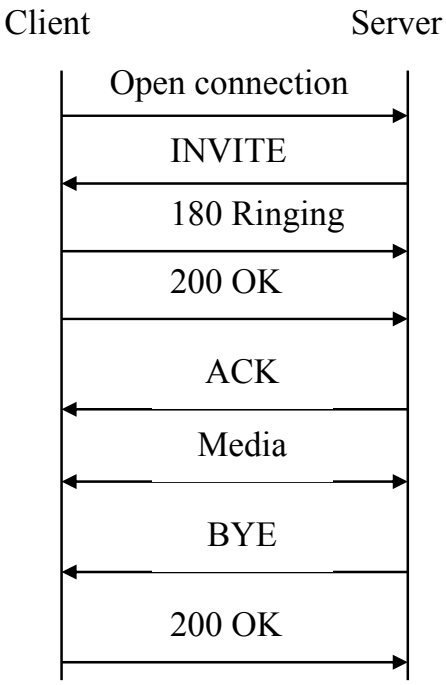


Hình 5.5 Thuật toán gọi đi

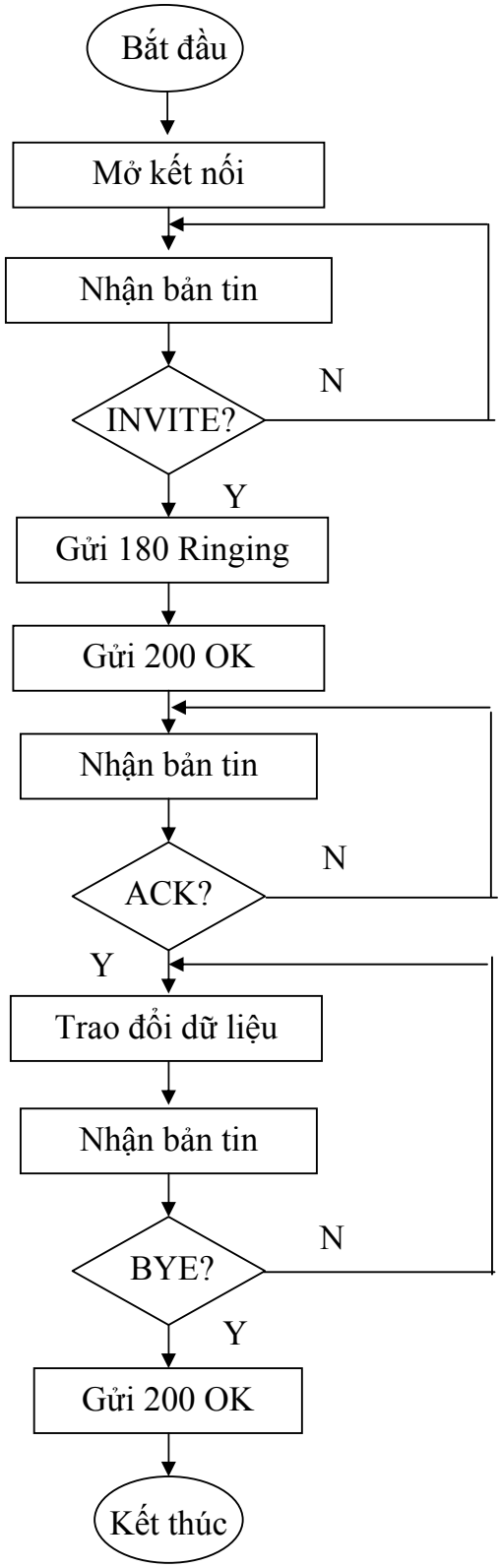
5.5. Chờ gọi đến và trả lời

Quy trình chờ gọi đến và trả lời như sau:

- Mở kết nối để chờ yêu cầu gửi đến.
- Nhận yêu cầu INVITE.
- Khởi tạo và gửi phúc đáp 180 và 200 OK.
- Chờ nhận yêu cầu ACK :phiên được khởi tạo.
- Trao đổi dữ liệu.
- Chờ nhận yêu cầu BYE: phiên kết thúc.
- Gửi phúc đáp 200 OK



Hình 5.6. Quá trình thực hiện trả lời cuộc gọi đến



Hình 5.7 Thuật toán trả lời cuộc gọi đến

5.6. Tạo project đóng gói chương trình

Sau khi viết xong mã nguồn, sử dụng J2ME Wireless Toolkit để lập dự án đóng gói chương trình. Trước hết ta khởi động chương trình KToolbar. Đặt tên dự án là "SIP". Sau đó copy các file nguồn vào thư mục \SIP\src\. Tiến hành biên dịch và tiền xác minh các file nguồn này bằng cách chọn Project → Package → Create Package. Trong quá trình dịch nếu có lỗi thì phải sửa lỗi ở các file nguồn. Sau khi dịch và tiền xác minh xong bộ công cụ sẽ tạo ra các file JAD, JAR và lưu trữ vào thư mục \SIP\bin\.

5.7. Mô phỏng

Sau khi đóng gói chương trình thì chạy trương trình mô phỏng trên máy tính. Hình mô phỏng như ở hình 5.8:

Hình 5.8. Mô phỏng điện thoại di động

KẾT LUẬN

Luận văn đã thực hiện được những vấn đề sau:

1. Tìm hiểu về J2ME, SIP, tìm được các điểm mạnh, điểm yếu, so sánh với các công nghệ tương tự.
2. Ứng dụng xây dựng một chương trình có tính năng:
   a. Đăng nhập vào mạng SIP.
   b. Thực hiện khởi tạo một phiên để gọi đến một thiết bị SIP khác.
   c. Trả lời khi có một cuộc gọi từ thiết bị SIP khác đến.
3. Phân tích, đánh giá, tổng hợp tạo ra một qui trình xây dựng các ứng dụng dựa trên J2ME và các giao thức truyền thông.

Hướng nghiên cứu tiếp theo là mở rộng framework về giao thức truyền tải thời gian thực (RTP- Real-time Transport Protocol) và thực hiện RTP bằng J2ME.

Một lần nữa em xin chân thành cảm ơn thầy giáo TS Hà Quốc Trung tận tình hướng dẫn em trong quá trình thực hiện luận văn. Đồng thời em cũng xin cảm ơn bạn bè, đồng nghiệp đã hỗ trợ em trong quá trình thực hiện luận văn này. Nếu có gì thiếu sót em rất mong được các thầy cô giáo và các bạn đồng nghiệp chỉ bảo.