Implement
Session Initiation Protocol (SIP)
User Agent Prototype

Thomas Pang (ktpang@sfu.ca)
Peter Lee (mclees@sfu.ca)

March 29, 2001
Agenda

- SIP Introduction
- Project Scope
- High Level Design
- Limitations
- Future Work
- Reference
- Q&A
SIP Introduction (1/2)

- IETF Signaling protocol (RFC 2543) for establishing real-time calls and conferences over IP
- Work began in 1995
- Authors: Henning G. Schulzrinne, Jonathan D. Rosenberg
- Lightweight - six basic messages
Text-based internet protocol resembles HTTP and SMTP
Uses Session Description Protocol (SDP) for media description
Transport independent (UDP, TCP, SCTP)
Client-server protocol
Not intended to replace H.323
SIP Messages (1/3)

- Invite - invites a user to join a call
- Bye - terminates the call
- Options - requests information on the capabilities of a server
- Ack - confirms that a client has received a final response to an INVITE
- Register - map for address resolution and location lookup
- Cancel - ends pending request
SIP Messages (2/3)

Example: SIP Request

```
INVITE sip:UserB@there.com SIP/2.0
Via: SIP/2.0/UDP here.com:5060
From: BigGuy <sip:UserA@here.com>
To: LittleGuy <sip:UserB@there.com>
Call-ID: 12345601@here.com
CSeq: 1 INVITE
Contact: BigGuy <sip:UserA@here.com>
Content-Type: application/sdp
Content-Length: ...

v=0
o=UserA 2890844526 2890844526 IN IP4 client.here.com
s=Session SDP
c=IN IP4 100.101.102.103
t=3034423619 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```
Example: SIP Response

SIP/2.0 200 OK
Via: SIP/2.0/UDP here.com:5060
Record-Route: <sip:UserB@there.com;maddr=ss1.wcom.com>
From: BigGuy <sip:UserA@here.com>
To: LittleGuy <sip:UserB@there.com>;tag=314159
Call-ID: 12345601@here.com
CSeq: 1 INVITE
Contact: LittleGuy <sip:UserB@there.com>
Content-Type: application/sdp
Content-Length: ...

v=0
o=UserB 2890844527 2890844527 IN IP4 client.there.com
s=Session SDP
c=IN IP4 110.111.112.113
t=3034423619 0 m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
Protocol Components

- User Agent - end system that acts on behalf of someone who wants to participate in calls
  - User Agent Client (UAC)
  - User Agent Server (UAS)
  - peer-to-peer operation
- Network Servers: proxy, redirect
- Registrar
SIP Architecture (1/2)
SIP Architecture (2/2)
Basic Call Message Flow

tpang@lorentz.sfu.ca  mclee@hall.sfu.ca

INVITE

Ringing

200 OK

ACK

RTP is activated (two way)

BYE

200 OK

RTP is terminated!
Project Scope (1/2)

- SIP User Agent (both server and client)
- Supported methods: INVITE, BYE, ACK
- Supported response messages: 200 OK, 180 Ringing, 486 Busy, 400 Bad Request
- Data terminal to data terminal
- One session at a time
- Media type: SDP
The following IP call scenarios will be simulated:

- Normal call setup and release
- Busy call
- Call not answer
- Call holding
Software Components

- State Event Machine
- UDP Listen
- UDP Send
- Parser
- Composer
- Share Memory Manager
User Agent System Diagram
SIP User Agent Design

- Written in C under UNIX environment
- Single Process with 2 threads
  - State Event Machine
  - UDP Listen
- Both threads share the same database

![Diagram of SIP User Agent Process]

SIP User Agent Process

- SEM Thread
- UDP Listen Thread

Share Memory
State Event Machine

- 2 different types of events
  - User Input event
  - UDP receive event
- 7 States
  - IDLE, WAITING, CONNECTED, HOLDING, PAUSE, RESUMING, DISCON
## State Event Matrix

<table>
<thead>
<tr>
<th>Event</th>
<th>IDLE</th>
<th>WAITING</th>
<th>CONNECTED</th>
<th>HOLDING</th>
<th>PAUSE</th>
<th>RESUMING</th>
<th>DISCON</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI_INVITE</td>
<td>IDLE, 3</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
</tr>
<tr>
<td>UI_BYE</td>
<td>IDLE, 3</td>
<td>IDLE, 3</td>
<td>IDLE, 3</td>
<td>IDLE, 3</td>
<td>IDLE, 3</td>
<td>IDLE, 3</td>
<td>IDLE, 3</td>
</tr>
<tr>
<td>UI_HOLD</td>
<td>IDLE, 3</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
</tr>
<tr>
<td>UI_RESUME</td>
<td>IDLE, 3</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
</tr>
<tr>
<td>UI_ANSWER</td>
<td>IDLE, 3</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
</tr>
<tr>
<td>UI_CANCEL</td>
<td>IDLE, 3</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
<td>IDLE, 23</td>
</tr>
<tr>
<td>INVITE</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>BYE</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>HOLD</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>RESUME</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>RSP_OK</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>RSP_ACK</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>RSP_BUSY</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>RSP_RINGING</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
<tr>
<td>RSP_NOK</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
<td>IDLE, 4</td>
</tr>
</tbody>
</table>
Program Flow (1/4)

Start

User Login

Initialized SEM & UDP listen Threads

Waiting for UI or UDP event

Listen UDP port
Program Flow (2/4)

Listen to the user specify UDP port

UDP message receive

If previous message is not clear

Yes

Copy the message to the share memory & set UDP receive message flag

No
Program Flow (3/4)

Waiting for UI Invite or UDP Event

- Prompt user to enter callee information
- Store callee info & generate invite event to remote end

Call in progress

- Yes
  - Store caller info & Clear UDP receive message flag

- No
  - UDP invite event for user

UI invite event
**Program Flow (4/4)**

- **Waiting for UI Invite or UDP event**
  - yes
  - **Call in Progress**
    - **Wait for UI or UDP event**
      - **Decompose the Message & clear the UDP receive message flag**
    - **Prompt User request**
      - **SEM will process the event according to the current state**
  - No
    - **Go to next state**

- **Call in progress clear**
Test Environment

Normal Call: Invite, Answer, Hold, Release

faraday → hall

Busy Call

faraday → kevin
Conclusions

- Successfully implement SIP user agent prototype
- Demonstrate the basic call scenarios
- Gain experience in developing protocol stack & project planning
- SIP is lightweight simple protocol
Future Works

• Support rest of the basic methods: REGISTER, CANCEL, OPTION
• Implement transaction manager to allow multiple simultaneous SIP sessions
• Implement network servers: proxy and redirect
• Handle more call scenarios: call forwarding, call waiting, multi-party call
• Include SDP
• Traffic analysis
References

- RFC 2543bis-02, SIP: Session Initiation Protocol, IETF, November 24, 2000
- SIP Telephony Service Examples, IETF, November 2000
- SIP for Telephones (SIP-T): Context and Architectures, IETF, November 21, 2000