菊风协议架构

Juphoon SIP Stack

发表日期: 2007-6-13 访问 http://www.juphoon.com 了解更多产品信息

菊风 SIP 协议栈简介

宁波菊风系统软件有限公司
http://www.juphoon.com
电话: +86-574-87287820
传真: +86-574-87304379
文件编号: 100-000-02-02

目 录

表格列表

图片列表

1. 简介

本文介绍了Juphoon SIP 协议栈产品的主要功能和使用方法，以便客户初步了解我们的产品。

1.1 目的

本文的目的是介绍SIP 协议栈的主要功能，如协议一致性、接口原语等，同时也部分介绍了协议栈的部分使用方式。最后是介绍了如何使用演示程序。

1.2 范围

本文只是对 SIP 协议栈的主要功能做了描述，对协议栈的使用方式也只粗略描述，详细的信息请参见 Juphoon SIP 的 Release Notes 和 User Manual。

1.3 术语定义

ZOS Zero Operating System

SIP Session Initiation Protocol

SDP Session Description Protocol

1.4 参考

[1] *RFC2327: Session Description Protocol*
[2] *RFC4566: Session Description Protocol*
[3] *RFC3261: Session Initiation Protocol*
[4] *RFC3262: Reliability of Provisional Response*
[5] *RFC3263: SIP: Locating SIP Servers*
[6] *RFC3264: An Offer-Answer Model with Session Description*
[7] *RFC3265: SIP – Specific Event Notification*
[8] *RFC3311: The SIP UPDATE method*
[9] *RFC3312: Integration of Resource Management and SIP*
[10] *RFC3313 Private SIP Extensions for Media Authorization*
[11] *RFC3323: A Private mechanism for the Session Initiation Protocol*
[12] *RFC3325: Private Extensions to the SIP for Asserted Identity within Trusted Networks*
[13] *RFC3326: The Reason Header*
[14] *RFC3329: Security Mechanism Agreement for the SIP*
[15] *RFC3372: SIP for Telephones (SIP-T) Context and Architectures*
[16] *RFC3428: SIP Extensions for Instant Messaging*
[17] *RFC3515: The REFER method*
[18] *RFC2806: URLs for Telephone Calls*
[19] *RFC2976: The SIP INFO Method*
[20] *RFC3204: MIME media types for ISUP and QSIG Objects*
[21] *RFC3581: An Extension to the SIP for Symmetric Response Routing*
[22] *RFC3903: Session Initiation Protocol (SIP) Extension for Event State Publication*
[23] *RFC3959: The Early Session Disposition Type for the SIP*
[24] *RFC3960: Early Media and Ringing Tone Generation in the SIP*

[25] *RFC3911: The Session Initiation Protocol (SIP) "Join" Header*
[26] *RFC3455: Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3GPP*
[27] *RFC3327: SIP Extension Header Field for Registering Non-Adjacent Contacts*
[28] *RFC3608: SIP Extension Header Field for Service Route Discovery During Registration*
[29] *RFC2368: The mailto URL scheme*
[30] *RFC4028: The SIP Session Timer*
[31] *draft-ietf-sip-events-01: SIP-Specific Event Notification*
[32] *RFC3515: The Refer Method*
[33] *RFC3891: The Session Initiation Protocol (SIP) "Replaces" Header*
[34] *draft-ietf-sip-callerprefs: Caller Preferences and Callee Capabilities for the Session Initiation Protocol (SIP)*
[35] *RFC3323: SIP Extensions for Network-Asserted Caller Identity and Privacy within Trusted Networks*
[36] *RFC3824: Using E.164 numbers with the Session Initiation Protocol (SIP)*
[37] *RFC4354: A Session Initiation Protocol (SIP) Event Package and Data Format for Various Settings in Support for the Push-to-Talk over Cellular (PoC) Service*
[38] *RFC4575:: A Session Initiation Protocol (SIP) Event Package for Conference State*
[39] *RFC3603: Private Session Initiation Protocol (SIP) Proxy-to-Proxy Extensions for Supporting the Packet Cable Distributed Call Signaling Architecture*
[40] *RFC3856: A Presence Event Package for the SIP*
[41] *RFC3857: A Watcher Information Event Template-Package fo SIP*
[42] *RFC4662: A SIP Event Notification Extension*

1.5 概述

Juphoon公司提供专业的协议软件，具有以下特点：

- 统一的清晰的代码架构
- 规范的接口
- 高度的可移植性
- 优秀的性能
- 良好的扩展性

用户只要仔细研读代码和运行协议软件，就能了解我们协议产品的高质量。向客户提供高品质的软件产品一直是我们的使命。

我们所有的协议软件产品都是严格遵照我们的协议规范来实现的。

Juphoon Software Architecture and Standard (JSAS)是菊风系统软件公司的软件开发框架，其内容包括了协议架构和协议规范。通过应用JSAS，使得菊风协议产品能够独立于特定的处理机、编译器和操作系统等应用环境，同时具备了良好的系统兼容性和客户产品整合能力。

在JSAS下，所有的协议软件都是采用相同的架构和规范编程，用户可通过比较演示程序中的SDP和SIP的ABNF编解码获得证实。因此，用户只要熟悉我们任何一种协议的设计和编码规范，就可以方便的维护Juphoon的其他协议软件，大大节省了客户的维护和开发成本。

2. 协议栈特性

SIP协议栈的主要功能是要满足协议标准的一致性，提供快速、可扩展、易维护的协议编解码，要支持基本传输处理，提供事务管理、Dialog 管理等功能。另外，目前Juphoon的SIP协议栈对于IMS所要求的各项特性已经支持得比较完备，可以作为IMS的核心SIP协议栈模块。

2.1 协议标准

SIP 协议栈支持的协议标准有：

标准名称	标准描述
RFC 3261	Session Initiation Protocol
RFC 3262	Reliability of Provisional Response
RFC 3263	Locating SIP Servers
RFC 3264	An Offer-Answer Model with Session Description
RFC 3265	Specific Event Notification
RFC 3311	The SIP UPDATE method
RFC 3312	Integration of Resource Management and SIP
RFC 3313	Private SIP Extensions for Media Authorization
RFC 3323	A Private mechanism for the SIP
RFC 3325	Private Extensions to the SIP for Asserted Identity within Trusted Networks
RFC 3326	The Reason Header
RFC 3329	Security Mechanism Agreement for the SIP
RFC 3372	SIP for Telephones (SIP-T): Context and Architectures
RFC 3428	SIP Extensions for Instant Messaging
RFC 3515	The REFER method
RFC 2806	URLs for Telephone Calls
RFC 2976	The SIP INFO Method
RFC 3204	MIME media types for ISUP and QSIG Objects
RFC 3581	An Extension to the SIP for Symmetric Response Routing
RFC 3903	Session Initiation Protocol (SIP) Extension for Event State Publication
RFC 3911	The Session Initiation Protocol (SIP) "Join" Header
RFC 3959	The Early Session Disposition Type for the SIP
RFC 3960	Early Media and Ringing Tone Generation in the SIP

RFC 3455	Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3GPP
RFC 3327	SIP Extension Header Field for Registering Non-Adjacent Contacts
RFC 3608	SIP Extension Header Field for Service Route Discovery During Registration
RFC 2368	The mailto URL scheme
RFC 2327	Session Description Protocol
RFC 4028	The SIP Session Timer
draft-ietf-sip-events-01	SIP-Specific Event Notification
RFC 3515	The Refer Method
RFC 3891	The Session Initiation Protocol (SIP) "Replaces" Header
draft-ietf-sip-callerprefs	Caller Preferences and Callee Capabilities for the Session Initiation Protocol (SIP)
RFC 3323	SIP Extensions for Network-Asserted Caller Identity and Privacy within Trusted Networks

表 2-1 SIP 协议标准支持列表

2.2 Method

SIP 协议栈支持的 Method 有：

- INVITE
- ACK
- OPTIONS
- CANCEL
- BYE
- REGISTER
- PRACK
- SUBSCRIBE
- NOTIFY
- INFO
- UPDATE
- MESSAGE
- REFER
- COMET
- PUBLISH

SIP 协议栈也支持其他扩展的 Method。

2.3 编解码

SIP 协议栈支持以下头域的编解码精确解析，并根据头域控制表格，选择头域部分编解码：

协议标准	支持的 SIP 头域
RFC 3261	1. Accept 2. Accept-Encoding 3. Accept-Language 4. Alert-Info 5. Allow 6. Authentication-Info 7. Authorization 8. Call-ID 9. Call-Info 10. Contact 11. Content-Disposition 12. Content-Encoding 13. Content-Language 14. Content-Length 15. Content-Type 16. CSeq 17. Date 18. Error-Info 19. Expires 20. From 21. In-Reply-To 22. Max-Forwards 23. MIME-Version 24. Min-Expires 25. Organization 26. Priority 27. Proxy-Authenticate 28. Proxy-Authorization 29. Proxy-Require 30. Record-Route 31. Reply-To 32. Require 33. Retry-After 34. Route 35. Server 36. Subject

	37. Supported 38. Timestamp 39. To 40. Unsupported 41. User-Agent 42. Via 43. Warning 44. WWW-Authenticate 45. extension-header
RFC 3262	46. RAck 47. RSeq
RFC 3265	48. Event 49. Allow-Events 50. Subscription-State
RFC 3515	51. Refer-To
draft-ietf-sip-events	52. Subscription-Expires
RFC 3892	53. Referred-By
RFC 3891	54. Replaces
RFC 4028	55. Session-Expires 56. Min-SE
RFC 3841	57. Request-Disposition 58. Accept-Contact 59. Reject-Contact
draft-ietf-sip-privacy	60. Anonymity 61. RPID-Privacy 62. Remote-Party-ID
RFC 3903	63. SIP-ETag 64. SIP-If-Match
RFC 3911	65. Join
RFC 3455 for 3GPP	66. P-Associated-URI 67. P-Called-Party-ID 68. P-Visited-Network-ID 69. P-Access-Network-Info 70. P-Charging-Addr 71. P-Charging-Vector

RFC 3327 for 3GPP	72. Path
RFC 3608 for 3GPP	73. Service-Route
RFC 3325	74. PAssertedID 75. PPreferredID
RFC 3313	76. P-Media-Authorization
RFC 3323	77. Privacy-hdr
RFC 3326	78. Reason
RFC 3329	79. security-client 80. security-server 81. security-verify
RFC 3603	82. P-DCS-Trace-Party-ID 83. P-DCS-OSPS 84. P-DCS-Billing-Info 85. P-DCS-LAES 86. P-DCS-Redirect
RFC 4244	87. History-Info
draft poc-p-headers	88. P-Alerting-Mode 89. P-Answer-State
RFC 4457	90. P-User-Database
RFC 4488	91. Refer-Sub

表 2-2 SIP 头域解析支持列表

SIP 协议栈也支持其他扩展的头域。

2.4 传输处理

目前的传输处理可同时处理 UDP 和 TCP，并且可以根据用户的实际传输协议（如SCTP, TLS, IPSec等）进行快速开发应用。

支持 RFC3261 协议规范中对传输请求时的规定：

在 MTU 值已知的情况下，当报文在比 MTU 小 200字节内，SIP 请求应通过 TCP 等可靠传输协议发送。

当 MTU 值未知的情况下，当报文大于 1300 字节，SIP 请求应通过 TCP 等可靠传输协议发送。

2.5 事务处理

SIP 协议栈的事务处理支持以下四种事务状态机：

- INVITE 的客户端事务

- 无 INVITE 的客户端事务
- 有 INVITE 的服务器端事务
- 无 INVITE 的服务器端事务

在事务处理状态机中，提供 User Agent 和 Proxy 的事务处理支持，因为这两者对事务处理的细节要求有所不同。

事务处理要跟对话（Dialog）层保持一定的界限，此二者的层间耦合度是松耦合的关系。

2.6 Dialog 处理

支持 RFC 3261(INVITE)，RFC 3265(SUBSCRIBE，NOTIFY)，RFC 3428（REFER）等协议规范中规定的 Dialog 创建方式。

支持 early dialog、establish dialog 和 multi dialog

2.7 URI Scheme

SIP 协议栈支持以下的 SIP URI scheme

- SIP
- SIPS
- IM
- TEL
- Absolute

SIP 协议栈也支持其他扩展 URI scheme。

2.8 MIME Type

SIP 协议栈支持以下的 Content Type：

- multipart/mixed
- application/sdp
- application/isup
- application/qsig
- text/plain
- message/sipfrag
- application/ simple-message-summary
- message/ cpim
- application/reginfo+xml
- application/watcherinfo+xml
- application/pidf+xml
- application/rlmi+xml
- application/simple-filter+xml

SIP 协议栈也支持其他扩展 MIME 类型。

2.9 自动头域产生

SIP 协议栈在呼叫流程中可以自动产生一些主要头域，这有助开发人员简化开发过程，快速实现业务流程。

SIP 协议栈可以自动产生 Call-ID, Cseq, Contact, Via, From, To, Content-Length 等关键头域，其中 From 和 To 头域在 Dialog 初创操作请求时（如 INVITE、SUBSCRIBE、REFER 等），用户需要手动填写，而在后续的操作中，协议栈会自动填写

如果协议用户已经指定了以上头域，协议栈将不会重新填写这些头域。

2.10 DNS 查询

支持 RFC 3261 和 RFC 3263 对 DNS 功能的规定。

支持(A，SRV和NAPTR等多种记录)，协议栈跟 DNS 客户端采用松耦合方式（系统消息）的通讯，同时作为DNS客户端需要提供记录缓存信息。

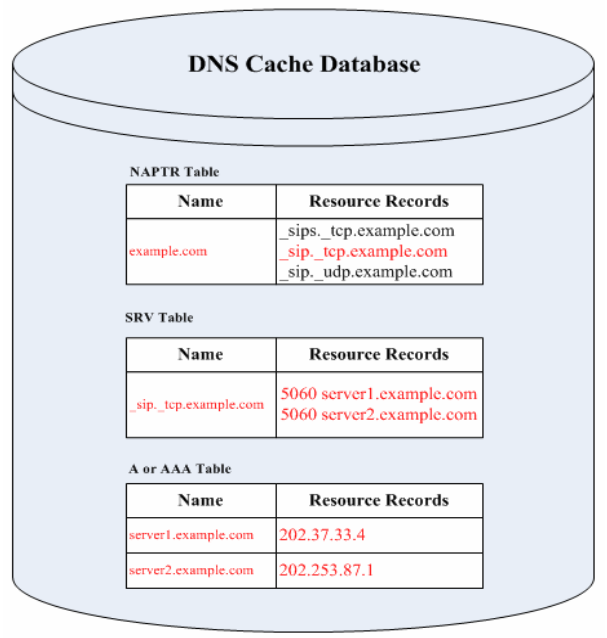

图 2-1 DNS数据查询过程和缓冲

2.11 Offer-Answer

SIP 协议栈支持携带 SDP 消息，支持 RFC3264 协议规范规定的 Offer Answer 模型。

SIP 协议栈通过支持 Alert-Info 头域和Offer Answer 模型，可以支持 RFC3960 中规定的 Early Media 功能。

2.12 Reliable Provisional Response

SIP 协议栈支持 RFC3262 协议规范中的 PRACK 方法来实现可靠临时响应。

SIP 协议栈支持 RSeq 和 Rack 头域，同时支持 100rel 的 option tag（参见枚举值 EN_SIP_OPT_TAG_100REL)。

2.13 Session Timer

SIP 协议栈支持通过 Re-INVITE 或 UPDATE 请求周期更新 SIP 会话的功能。

SIP 协议栈支持 RFC4028 规范中规定的 Session-Expire 和 Min-SE 头域，支持 timer 的 option tag（参见枚举值 EN_SIP_OPT_TAG_TMR)。

2.14 Sigcomp

Juphoon 提供独立的 Sigcomp 功能模块，可以跟 SIP 协议栈的传输层相结合，实现 SIP Signal Compress 功能。

SIP 协议栈支持 URI 参数：comp=sigcomp 以指示消息通过 Sigcomp 被压缩过了。

Juphoon Sigcomp 模块支持以下协议标准：

- RFC 3320(Signaling Compression)
- RFC 3321(SigComp - Extended Operations)
- RFC 3485(SIP and SDP static Dictionary for SigComp)
- RFC 3486(Compressing the SIP)
- RFC 4077 (A Negative Acknowledgement Mechanism for Signaling Compression)
- RFC 4464 (SigComp Users Guide)
- RFC 4465 (SigComp Torture Tests)

2.15 IPv6 支持

SIP 协议栈支持IPv6传输功能，并且已经在实际环境中进行了测试。用户只需配置协议栈参数即可控制 IPv4 和 IPv6 传输协议的选择。

2.16 SIP Body

SIP 协议栈支持在 SIP 消息中带不同的 SIP 消息体

- MIME 类型的消息
- 显示解析的 SDP 消息
- Mutlitpart 消息
- SIP Fragment 消息
- XML 消息

2.17 SDP 集成

SDP 协议栈是一个独立的模块，考虑很多 SIP 业务对 SDP 的关联性较大，SIP 协议栈默认集成了 SDP 消息的解析和编码功能。

SDP 协议栈支持精细的 SDP 报文解析，支持的协议标准有：

- RFC2327 SDP
- RFC4566 SDP
- RFC3266 Support for IPv6 in SDP
- RFC3605 RTCP attribute in SDP).txt
- RFC3556 SDP Bandwidth Modifiers for RTCP Bandwidth
- RFC3890 A Transport Independent Bandwidth Modifier for SDP
- RFC4145 TCP-Based Media Transport in the Session Description Protocol (SDP)
- RFC3312 Integration of Resource Management and SDP
- RFC3388 Grouping of Media Lines in SDP
- RFC2733 RTP Payload Format for Generic Forward Error Correction
- RFC2848 Extensions to SIP and SDP for PINT
- RFC2833 RTP Payload for DTMF Digits
- RFC3640 RTP Payload Format for Transport of MPEG-4 Elementary Streams

- RFC3950 Tag Image File Format Fax eXtended (TIFF-FX) - image/tiff-fx MIME Sub-type Registration
- OMA-TS-PoC-ControlPlane-V1_0_1-20061128-A

2.18 SIP Fragment

支持 RFC 3420 协议规范，即在 SIP Body 中带格式良好的 SIP 消息，一般在 RFFER 方法中用于传递相关请求的状态信息。

SIP Fragment 要求 Content-Type 支持 message/sipfrag, 同时在 SIP Body 中支持以下语法规定的 SIP 消息。

```
sipfrag = [ start-line ]
          *message-header
          [ CRLF [ message-body ] ]
```

2.19 Event Package

SIP 协议栈支持以下 Event Package：

- RFC3856 中的 “presence” 事件包
- RFC3515 中的 “refer” 事件包
- RFC4354 中的 “poc-settings” 事件包
- RFC4575 中的 “conference” 事件包
- RFC4235 中的 “dialog” 事件包
- RFC4730 中的 “kpml” 事件包
- RFC3842 中的 “message-summary” 事件包
- RFC3680 中的 “reg” 事件包
- draft-ietf-sip-xcap-config 中的 “ua-profile” 事件包

SIP 协议栈支持以下 Event Template：

- RFC3857 中的 “winfo” 事件模板

2.20 IM & Presence

SIP 协议栈支持 IM & presence 业务的标准有：

- RFC 3428 Extension for Instant Messaging
- RFC 3856 SIP Extensions for Presence
- RFC 3857 A Watcher Information Event Template-Package (WINFO)
- RFC4662 A SIP Event Notification Extension

2.21 3GPP & IMS

SIP 协议栈支持 3GPP 扩展的标准有：

- RFC3455 Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3GPP

- RFC3324 Short Term Requirements for Network Asserted Identity
- RFC3608 SIP Extension Header Field for Service Route Discovery During Registration
- RFC3325 Private Extensions to the SIP for Asserted Identity within Trusted Networks
- RFC3603 Private Session Initiation Protocol (SIP) Proxy-to-Proxy Extensions for Supporting the PacketCable Distributed Call Signaling Architecture
- RFC4354 A SIP Event Package and Data Format for Various Settings in Support for the Push-to-Talk over Cellular (PoC) Service
- RFC4083 3GPP Release 5 Requirements
- RFC4032 Update to SIP Preconditions Framework
- 支持 “+g.poc.talkburst“, “+g.poc.groupad“, “3GPP-GERAN“ 等参数

2.22 Quality of Service

SIP 协议栈支持 QoS 的标准有：

- RFC 3313 Private Extensions for Media Authorization

2.23 Security

SIP 协议栈支持 Security 的标准有：

- RFC3323 A Privacy mechanism for the Session Initiation Protocol
- RFC3329 Security Mechanism Agreement

2.24 NAT Traversal

SIP 协议栈支持NAT穿越的参数支持：

支持在 RFC 3261 协议规范中的 Via 头域中的“received”参数

支持在 RFC 3581 协议规范中的“rport”参数

3. 协议应用举例

3.1 SIP Phone (JPhone)

JPhone 是菊风公司推出的SIP Phone的解决方案,它包括了Fast ABNF、Object Fsm、RTimer等多项菊风公司创新协议开发技术。JPhone 可以实现独立于Call Server的各种传统和创新型的Value-Added业务，其应用网络可支持3G，NGN，P2P，INTERNET等。

JPhone方案支持在Windows上开发测试，实现相同的功能特性，在多种平台上体现多种多样的Look & Feel：

图 3-1 SIP Phone（JPhone）的多种表现形式

3.2 SIP IM & Presence

使用Juphoon SIP Stack顺从以下规范以支持实现基于SIP标准的即时通信和呈现业务的终端：

RFC3428, Session Initiation Protocol (SIP) Extension for Instant Messaging

RFC3856，A Presence Event Package for the Session Initiation Protocol (SIP)

同样，该方案支持在Windows上开发测试，实现相同的功能特性，在多种平台上体现多种多样的Look & Feel。

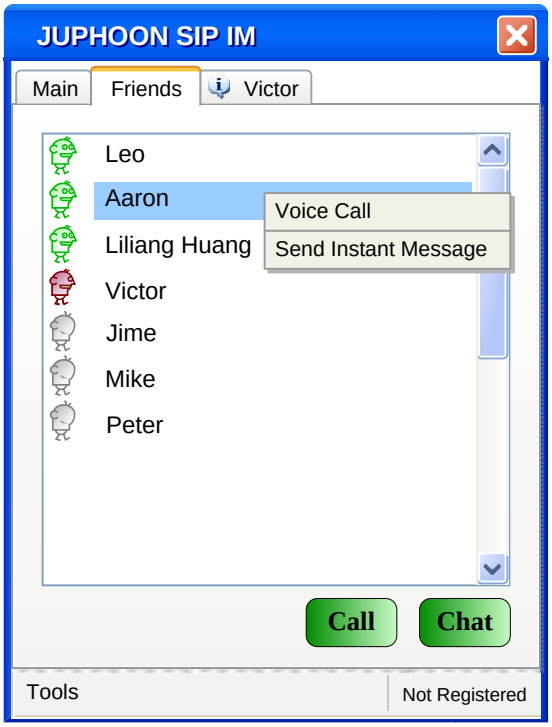

图 3-2 Juphoon SIP IM 的界面参考

3.3 SIP Server

Juphoon SIP Stack按照实现SIP Server的规格实现SIP Stack的各种特性，包括支持多个SUA端点，支持多路多任务并发处理。可支持的Server包括，SIP Call Server，SIP Proxy（state & stateless）等。

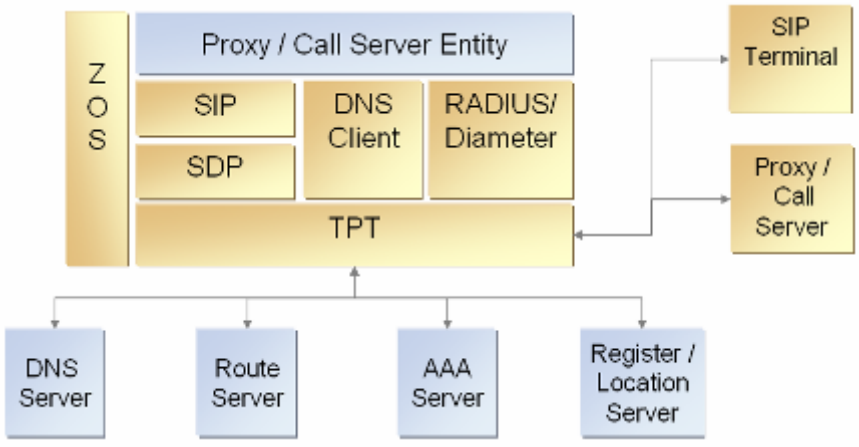

图 3-3 SIP Server的参考场景

3.4 IMS PoC client

Juphoon SIP Stack加上Juphoon PoC Client Framework支持开发IMS PoC UE，顺从IETF/OMA/3GPP标准，包括：

3GPP/3GPP2/IETF & OMA PoC 1.0 Compliant

IETF SIMPLE Compliant

IETF SIP (RFC3261, RFC3262, RFC3265...etc)

IETF SDP (RFC2327, RFC4566...etc)

IETF HTTP (RFC2616, OMA-TS-XDM_Core-V1_0_1, 3GPP TS 24.109 etc)

IETF XCAP (draft-ietf-simple-xcap-12, draft-ietf-simple-xcap-list-usage ... etc)

IETF XML (W3C XML 1.0, 1.1, DOM Level-1, 2, 3... etc)

IETF SigComp (RFC3320, RFC3321, RFC3485, RFC4464 ... etc)

IETF RTP/RTCP (RFC1889, RFC1890 RFC3267, RFC3711... etc)

功能包括：

- 一对一通话
- 三种群组通话（ad-hoc，pre-arranged 和 chat）
- 用户列表/组管理
- 用户注册
- PoC 会话管理（PoC Session）
- PoC 语音控制（Talk Burst）
- 用户呈现（Presence）
- 用户消息（Instant Personal Alert）
- PoC 设置管理（Application Setting， Account Setting）

图 3-4 PoC UE参考终端

4. 协议用户接口

SIP 协议栈提供用户接口原语，以方便用户进行协议应用开发。

原语分为以下4类：

- Request 请求
- Confirm 确认
- Indication 指示
- Response 响应

接口原语按照消息的作用分为以下几类：

- 会话相关消息，如 INVITE, re-INVITE，CANCLE, BYE, ACK 等消息参与会话建立、修改和删除
- 会话状态消息，如 REGISTER，INFO，OPTION，MESSAGE，NOTIFY 等在会话中使用，但不参与会话建立、修改和删除等过程
- 对话相关消息，如 SUBSCRIBE，REFER，NOTIFY 等消息会导致对话(Dialog)的建立和删除
- 呼叫独立消息，如 REGISTER，INFO，OPTION 等不属于某个会话的消息

4.1 原语操作

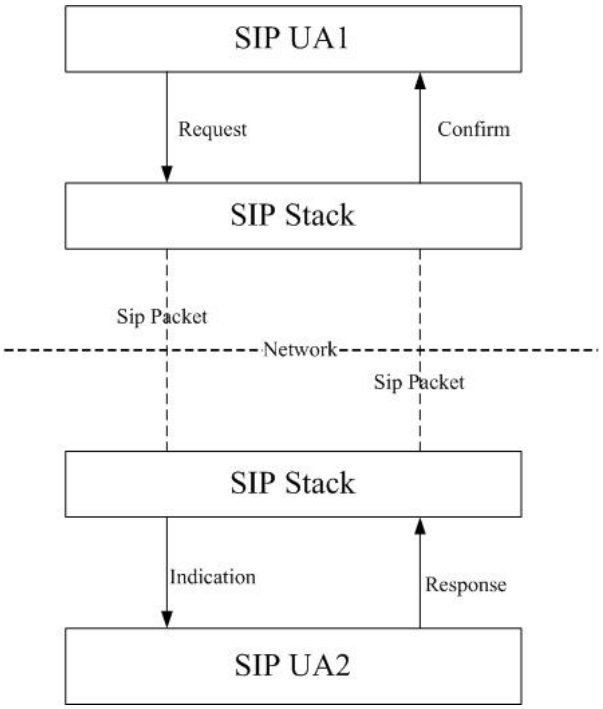

图 4-1 层间原语操作流程

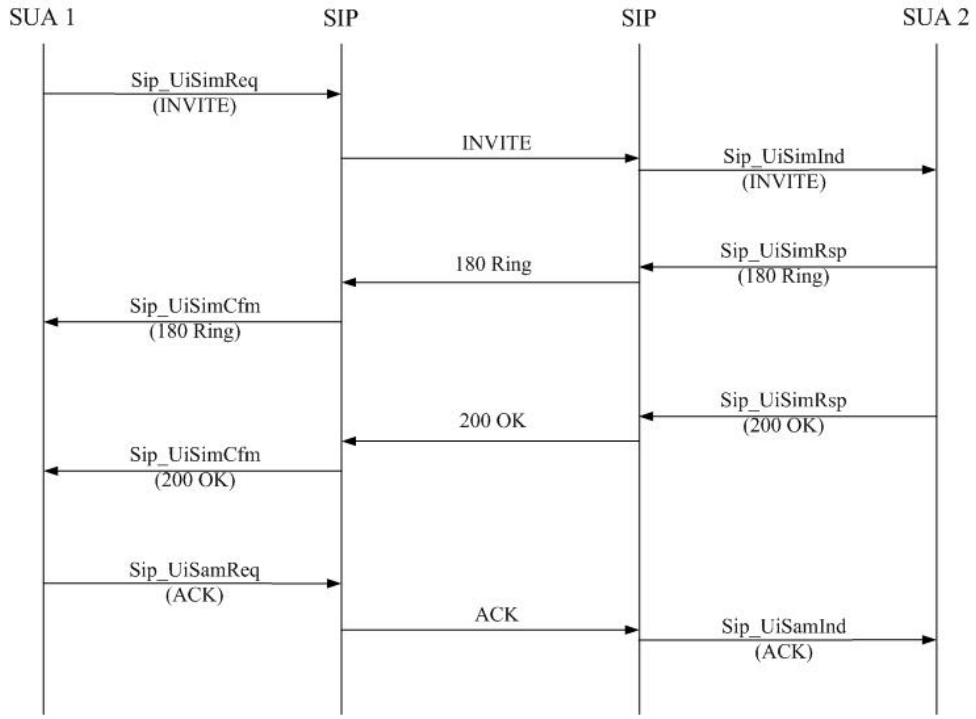

图 4-2 SIP正常流程原语操作图

4.2 接口原语

主要的用户应用接口原语有：

接口名称	接口描述	接口操作方向
Sip_UiSimReq	会话起始请求 (INVITE)	SUA → SIP
Sip_UiSimRsp	会话起始响应 (INVITE)	SUA → SIP
Sip_UiSsmReq	会话状态请求 (PRACK, INFO, UPDATE, OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SUA → SIP
Sip_UiSsmRsp	会话状态响应 (PRACK, INFO, UPDATE, OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SUA → SIP
Sip_UiSamReq	会话确认请求 (Ack)	SUA → SIP
Sip_UiScmReq	会话取消请求 (CANCEL)	SUA → SIP
Sip_UiSmmReq	会话修改请求 (Re-INVITE)	SUA → SIP
Sip_UiSmmRsp	会话修改响应 (Re-INVITE)	SUA → SIP
Sip_UiStmReq	会话释放请求 (BYE)	SUA → SIP
Sip_UiDamReq	对话相关消息请求 (SUBSCRIBE, REFER, NOTIFY)	SUA → SIP

Sip_UiDamRsp	对话相关消息响应 (SUBSCRIBE, REFER, NOTIFY)	SUA → SIP
Sip_UiCimReq	呼叫独立消息请求 (OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SUA → SIP
Sip_UiCimRsp	呼叫独立消息响应 (OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SUA → SIP

表 4-1 用户应用接口原语

4.3 会话事件

事件名称	事件描述	消息方向
EN_SIP_SESSE_ERR_IND	会话错误指示	SIP → SUA
EN_SIP_SESSE_SIM_CNF	会话起始确认 (INVITE)	SIP → SUA
EN_SIP_SESSE_SIM_IND	会话起始指示 (INVITE)	SIP → SUA
EN_SIP_SESSE_SSM_CNF	会话状态确认 (PRACK, INFO, UPDATE, OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SIP → SUA
EN_SIP_SESSE_SSM_IND	会话状态指示 (PRACK, INFO, UPDATE, OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SIP → SUA
EN_SIP_SESSE_SAM_IND	会话确认指示 (Ack)	SIP → SUA
EN_SIP_SESSE_SCM_IND	会话取消指示 (CANCEL)	SIP → SUA
EN_SIP_SESSE_SMM_CNF	会话修改确认 (Re-INVITE)	SIP → SUA
EN_SIP_SESSE_SMM_IND	会话修改指示 (Re-INVITE)	SIP → SUA
EN_SIP_SESSE_STM_IND	会话释放请求 (BYE)	SIP → SUA
EN_SIP_SESSE_DAM_CNF	对话相关消息确认 (SUBSCRIBE, REFER, NOTIFY)	SIP → SUA
EN_SIP_SESSE_DAM_IND	对话相关消息指示 (SUBSCRIBE, REFER, NOTIFY)	SIP → SUA
EN_SIP_SESSE_CIM_CNF	呼叫独立消息确认 (OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SIP → SUA
EN_SIP_SESSE_CIM_IND	呼叫独立消息指示 (OPTIONS, REGISTER, MESSAGE, COMET, PUBLISH)	SIP → SUA

4.4 消息接口

SIP 协议栈提供创建、删除、查找，拷贝，比较，复制头域等近 200 个 SIP 消息操作接口。

如：Sip_CreateMsgHdr，Sip_DeleteMsgHdr，Sip_FindMsgHdr等。

5. 功能扩展

SIP 协议栈具有非常便捷的编解码扩展能力支持，主要分为以下三类：

- 头域扩展
- 类型扩展
- 参数扩展

值得说明的是，用户可以对头域的编解码进行是否全解控制，此场景在Proxy应用尤其重要，用户可以只关心需要处理的头域，而对其他不重要的头域可以忽略解码，这将大大提高处理速度。

Juphoon 的软件产品可以通过应用 ZOS 操作系统服务平台，使得软件产品快速高效的与多种操作系统兼容。ZOS 即 Zero Operating System，由一组公共服务接口组成，如任务管理、内存管理、消息队列、数据缓冲区、计时器管理等。SIP 协议应该使用 ZOS 接口来实现协议栈和完成与其它模块间的通信处理。

ZOS平台支持Windows、Linux、VxWorks、Solaris等操作系统，如要支持Windows，在演示程序中只要设置 **ZPLATFORM=ZPLATFORM_WIN32** 宏即可。

6. 实现说明

SIP 协议栈主要代码由3部分组成，分别是ZOS操作系统服务平台、SDP协议栈和SIP 协议栈。

6.1 关联模块

SIP 协议栈主要关联模块有：

- ZOS
- ABNF
- SDP

6.2 测试说明

通过 SIPit Interoperability Testing 和 RFC4475 SIP Torture Testing 等协议测试

SIP 协议栈可以跟 Asterisk, SER, OpenSER (open SIP server), CISCO SIP server 等服务器对接，同时也可以直接与 SJPhone, EyeBeam(X-Lite), LinPhone 等 SIP 电话机对接。

6.3 代码体积

下表是ZOS、SDP、SIP、RTP等协议栈在不同编译环境的代码体积：

OS	Compiler	Binary Library
Windows	VC6.0	sip.lib 1810k, zos.lib 561k, sdp.lib 357k, rtp.lib 102k
Linux	GCC 3.2.2	libsip.a 722k libzos.a 264k, libsdp.a 127k, librtp.a 53k

表 6-1 软件代码体积

7. 应用开发

SIP 协议栈的应用主要是基于 SIP Session Event 和 原语接口来达到应用的目的，下面以发送 INVITE 请求和响应，ACK 请求为举例，相关的原语为 Sip_UiSimReq, Sip_UiSimRsp, Sip_UiSamReq。

7.1 会话事件初始化

```
/* sip ua session event from call info init */
ZINT Sua_SessEvntInit(ST_SIP_SESS_EVNT *pstSessEvnt,
            ST_SUA_MSG_EVNT *pstMsgEvnt)
{
   ST_SUA_FLOW *pstFlow;
   ST_SUA_REG *pstReg;

   pstSessEvnt->ucEvntType = EN_SIP_SESSE_UNKNOWN;
   pstSessEvnt->ucEvntOwner = EN_SIP_EOT_SUA;
   pstSessEvnt->ucMsgType = EN_SIP_EMT_NON;
   pstSessEvnt->ucMethodType = 0;
   pstSessEvnt->dwStatusCode = 0;
   pstSessEvnt->pstMsg = ZNULL;
   pstSessEvnt->pstMethod = ZNULL;
   pstSessEvnt->pstEvntPkg = ZNULL;
   pstSessEvnt->stTptAddr.ucType = EN_SIP_TPT_SERV_UNKNOWN;

   /* check call is exist */
   if (pstMsgEvnt->pstCall)
   {
      pstFlow = pstMsgEvnt->pstFlow;
      SIP_CPY_TPT_ADDR(&pstSessEvnt->stTptAddr, &pstFlow->stTptAddr);
      pstSessEvnt->dwSessUserId = pstMsgEvnt->pstCall->dwCallId;
      pstSessEvnt->dwDlgUserId = pstFlow->dwFlowId;
      pstSessEvnt->dwSessId = pstMsgEvnt->pstCall->dwSipSessId;
      pstSessEvnt->dwDlgId = pstFlow->dwSipDlgId;
      pstSessEvnt->dwTransId = pstMsgEvnt->pstTrans->dwSipTransId;
```

```
    }
    else /* without call, then must be register */
    {
        pstReg = pstMsgEvnt->pstReg;
        SIP_CPY_TPT_ADDR(&pstSessEvnt->stTptAddr, &pstReg->stTptAddr);
        pstSessEvnt->dwSessUserId = pstReg->dwRegId;
        pstSessEvnt->dwDlgUserId = ZMAXULONG;
        pstSessEvnt->dwSessId = pstReg->dwSipSessId;
        pstSessEvnt->dwDlgId = ZMAXULONG;
        pstSessEvnt->dwTransId = ZMAXULONG;
    }

    return ZOK;
}
```

7.2 发送 INVITE 请求

```
/* sip ua send INVITE message */
ZINT Sua_SipSendInvite(ST_SUA_MSG_EVNT *pstMsgEvnt)
{
    ST_SIP_SESS_EVNT stSessEvnt;
    ST_SUA_CP_EVNT *pstCpEvnt;
    ST_SUA_FLOW *pstFlow;
    ST_SIP_MSG *pstMsg;

    /* get call and flow */
    pstCpEvnt = pstMsgEvnt->pstCpEvnt;
    pstFlow = pstMsgEvnt->pstFlow;

    /* session event init */
    Sua_SessEvntInit(&stSessEvnt, pstMsgEvnt);

    /* generate invite message */
    if (Sua_GenSipMsg(pstCpEvnt, &pstMsg) != ZOK)
        return ZFAILED;
```

```
/* set session event info */
stSessEvnt.ucMsgType = EN_SIP_EMT_REQ;
stSessEvnt.ucMethodType = EN_SIP_METHOD_INVITE;
stSessEvnt.pstMsg = pstMsg;
pstMsg->ucReqPres = ZTRUE;

/* add Allow header */
if (Sua_AddAllow(pstMsg) != ZOK)
{
   SUA_MSG_DELETE(pstMsg);
   return ZFAILED;
}

/* add Supported header */
if (Sua_AddSupported(pstMsg, g_stSuaCfg.dwSuptFlag) != ZOK)
{
   SUA_MSG_DELETE(pstMsg);
   return ZFAILED;
}

/* add user uri */
if (Sua_AddUsrUri(pstMsg, EN_SIP_METHOD_INVITE, pstFlow) != ZOK)
{
   SUA_MSG_DELETE(pstMsg);
   return ZFAILED;
}

/* add session description */
if (Sua_AddSipBody(pstMsg, pstFlow->pstBody) != ZOK)
{
   SUA_MSG_DELETE(pstMsg);
   return ZFAILED;
}

/* send sim request */
if (Sip_UiSimReq(&g_stSuaModMgr.stLmMsp, &stSessEvnt) != ZOK)
{
```

```
      SUA_MSG_DELETE(pstMsg);
      return ZFAILED;
   }

   return ZOK;
}
```

7.3 发送 INVITE 响应

```
/* sip ua send INVITE response message */
ZINT Sua_SipSendInviteRsp(ST_SUA_MSG_EVNT *pstMsgEvnt,
            ZULONG dwStatCode)
{
   ST_SIP_SESS_EVNT stSessEvnt;
   ST_SUA_CP_EVNT *pstCpEvnt;
   ST_SUA_FLOW *pstFlow;
   ST_SIP_MSG *pstMsg;

   /* get call flow from message event */
   pstCpEvnt = pstMsgEvnt->pstCpEvnt;
   pstFlow = pstMsgEvnt->pstFlow;

   /* session event init */
   Sua_SessEvntInit(&stSessEvnt, pstMsgEvnt);

   /* generate invite message */
   if (Sua_GenSipMsg(pstCpEvnt, &pstMsg) != ZOK)
      return ZFAILED;

   /* set sesion event info */
   stSessEvnt.ucMsgType = EN_SIP_EMT_RSP;
   stSessEvnt.ucMethodType = EN_SIP_METHOD_INVITE;
   stSessEvnt.dwStatusCode = dwStatCode;
   stSessEvnt.pstMsg = pstMsg;
   pstMsg->ucReqPres = ZFALSE;

   /* add mixer contact header */
```

```
   if (pstCpEvnt != ZNULL)
   {
      /* add session description */
      if (Sua_AddSipBody(pstMsg, pstCpEvnt->pstBody) != ZOK)
      {
         SUA_MSG_DELETE(pstMsg);
         return ZFAILED;
      }
   }

   /* send sim response */
   if (Sip_UiSimRsp(&g_stSuaModMgr.stLmMsp, &stSessEvnt) != ZOK)
   {
      SUA_MSG_DELETE(pstMsg);
      return ZFAILED;
   }

   return ZOK;
}
```

7.4 发送 ACK 请求

```
/* sip ua send ACK message */
ZINT Sua_SipSendAck(ST_SUA_MSG_EVNT *pstMsgEvnt)
{
   ST_SIP_SESS_EVNT stSessEvnt;
   ST_SUA_CP_EVNT *pstCpEvnt;

   /* get call and flow */
   pstCpEvnt = pstMsgEvnt->pstCpEvnt;

   /* session event init */
   Sua_SessEvntInit(&stSessEvnt, pstMsgEvnt);

   /* set base info */
   stSessEvnt.ucMsgType = EN_SIP_EMT_REQ;
   stSessEvnt.ucMethodType = EN_SIP_METHOD_ACK;
```

```
   /* send sam request */
   if (Sip_UiSamReq(&g_stSuaModMgr.stLmMsp, &stSessEvnt) != ZOK)
      return ZFAILED;

   return ZOK;
}
```

7.5 会话事件转化

SIP User Agent(SUA) 任务在收到 SIP 协议栈消息后，首先通过 Sua_CpEvntTypeInit 对 SIP 协议栈的会话事件进行内部状态机事件的转化，然后直接驱动呼叫处理状态机。

下面是 SUA 中函数 Sua_CpEvntTypeInit 进行 SIP 事件转换的部分例子：

```
/* sip ua cp event type init */
ZCHAR * Sua_CpEvntTypeInit(ST_SUA_MSG_EVNT *pstEvnt)
{
   ZULONG dwStatusCode, dwEvntType;
   ST_SIP_SESS_EVNT *pstSessEvnt;
   ST_SUA_CP_EVNT *pstCpEvnt;
   ST_SUA_TMR *pstTmr;
   ZUCHAR ucMethodType;

   if (pstEvnt->ucEvntOwner == EN_SUA_EOT_SM)
   {
      pstSessEvnt = pstEvnt->pstSessEvnt;
      dwEvntType = pstSessEvnt->ucEvntType;
      ucMethodType = pstSessEvnt->ucMethodType;
      dwStatusCode = pstSessEvnt->dwStatusCode;
```

```
switch (dwEvntType)
{
case EN_SIP_SESSE_ERR_IND:
   pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_ERR;
   return "ERR IND";
case EN_SIP_SESSE_SIM_IND:
   pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_IVT;
   return "SIM IND";
case EN_SIP_SESSE_SIM_CNF:
   if (dwStatusCode > 100 && dwStatusCode <= 199)
   {
      pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_IVT_1XX;
      return "SIM CNF 1XX";
   }
   else if (dwStatusCode >= 200 && dwStatusCode <= 299)
   {
      pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_IVT_2XX;
      return "SIM CNF 2XX";
   }
   else if (dwStatusCode >= 300 && dwStatusCode <= 699)
   {
      pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_IVT_3XX;
      return "SIM CNF 3456XX";
   }
   break;
case EN_SIP_SESSE_SAM_IND:
   pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_ACK;
   return "SAM IND";
case EN_SIP_SESSE_SCM_IND:
   pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_CANCEL;
   return "SCM IND";
case EN_SIP_SESSE_STM_IND:
   pstEvnt->ucEvntType = EN_SUA_CPE_MINOR_SM_RECV_BYE;
   return "STM IND";
default:
   return ZNULL;
}
```

```
    }
    return ZNULL;
}
```

7.6 SIP 消息接口应用

SIP 协议栈提供创建、设置等操作头域的接口，下面就是创建一个 CSeq 头域的例子：

```
    ST_SIP_HDR_CSEQ *pstCseq;

    /* create cseq header */
    pstCseq = Sip_CreateMsgHdr(pstEvnt->pstMsg, EN_SIP_HDR_CSEQ);
    pstCseq->dwCseqVal = 100;
    pstCseq->stMethod.ucPres = ZTRUE;
    pstCseq->stMethod.ucType = EN_SIP_METHOD_INVITE;
```

8. 性能演示

目前的演示程序主要是测试协议的编解码能力和其相应的性能。演示程序分为Debug和Release两种版本，以方便用户（或与第三方SIP协议）进行性能比较。

为了让测试用户便于关注测试本身，演示程序删减了很多头域说明，如sip_abnf_decode.h只提供了对Sip_DecodeMsg的函数声明，只提供一个运行测试程序的小型集合。

此外，演示程序不提供头域解码控制，对目前所支持的头域实行全部解码处理，因此用户在比较性能的时候要注意这一点。而对于消息中的SDP协议报文，我们也对其所有的域进行全解处理的。

最后，本演示程序不是本公司SIP 协议栈的优化版，所有性能数据不是最佳的性能统计结果。而且目前版本下SIP协议是完全按照RFC3261的语法定义实现，如果消息解析不通过，请首先检查测试消息的语法正确性。

演示程序是 Visual C++6.0 的工程，用户可以通过修改工程设置选择 DEBUG 或 RELEASE 版本。

8.1 启动说明

演示程序的启动过程在 tsip_msg.c 中的main函数中，

通过注释代码中的Tsip_FmsgRun、Tsip_MmsgRun、Tsip_MmsgPerformRun(30000)等函数就可以选择测试的内容。

在main函数中主要有以下启动步骤：

- ZOS 系统配置，执行 Zos_SysCfgInit
- 启动 ZOS 系统，Zos_SysInit
- 初始化 SDP 的 ABNF 部分，Sdp_AbnfInit
- 初始化 SIP 的 ABNF 部分，Sip_AbnfInit
- 运行测试程序
- 结束 ZOS 系统
- 退出演示程序

```
ZINT main()
{
    Zos_SysCfgInit();

    /* system init */
    if (Zos_SysInit() != ZOK)
    {
        return -1;
    }

    /* sdp abnf init */
    if (Sdp_AbnfInit() != ZOK)
    {
        Zos_SysDestroy();
        return -1;
    }

    /* abnf init */
    if (Sip_AbnfInit() != ZOK)
    {
        Sdp_AbnfDestroy();
        Zos_SysDestroy();
        return -1;
    }

    /* run file message test */
    Tsip_FmsgRun();

    /* run memory message test
    Tsip_MmsgRun(); */

    /* run memory message perfrom test
    Tsip_MmsgPerformRun(30000); */

    getchar();

   /* system destroy */
    Zos_SysDestroy();

    return 0;
}
```

8.2 内存消息

在演示程序中，我们提供了14种测试报文，分别有6种请求报文、7种响应报文和1个带多个消息体的报文，而且报文中也带有SDP报文。

用户可以添加一块消息，修改代码在tsip_mmsg.c中，如：

```
ZCHAR *m_acTsipMemMsgRequest1 =
{
    "ACK sip:009198400000@218.115.159.20:5060;user=phone
SIP/2.0\r\n"
    "Via: SIP/2.0/UDP 218.115.159.156:5060;"
         "branch=z9hG4bK8ca5821c3fdf4cb3ea2b6b921fb68949\r\n"
    "From:
<sip:312345678@218.115.159.106:5060>;tag=003500521121\r\n"
    "To: <sip:009198400000@218.115.159.86>;tag=1434385583\r\n"
    "Call-ID: 851982_10244-1@218.115.159.156\r\n"
    "CSeq: 2 ACK\r\n"
    "Max-Forwards: 68\r\n"
    "Content-Length: 0\r\n\r\n"
};
```

修改测试报文配置管理记录：

```
/* sipit test files list */
ZCHAR **m_appcTsipMemMsgs[] =
{
    &m_acTsipMemMsgRequest1,
    &m_acTsipMemMsgRequest2,
    &m_acTsipMemMsgRequest3,
    &m_acTsipMemMsgRequest4,
    …
    &m_acTsipMemMsgOtherMsg1,
    ZNULL
};
```

Tsip_MmsgRun是文件消息测试的运行函数。

用户可以在ZINT Tsip_Mmsg(ZCHAR *pcMemMsg) 函数中修改代码，以选择是否支持解码后执行编码工作，同时也可以按照情况打印测试报文。

8.3 文件消息

在演示程序中提供了SIP interoperability test events测试报文，位于演示程序所在目录的msg/sipit目录中。

SIP interoperability test events are gatherings of SIP implementors to test interoperability of their creations.

详细信息参见：http://www.cs.columbia.edu/sip/sipit/

在线报文见：http://www.cs.columbia.edu/sip/sipit/testmsg.html

说明的是：sipit中的个别报文有严重的语法错误，个别报文我们做了修订，在最终版本中我们会考虑适当的支持这些语法错误的消息解析。

测试代码在tsip_fmsg.c中：

```
/* sipit test files list */
ZCHAR *m_apcSipitFiles[] =
{
   "../../msg/sipit/test1",
   "../../msg/sipit/test2",
   …
   "../../msg/sipit/test40",
   "../../msg/sipit/test41",
   "../../msg/sipit/test42",
   ZNULL
};
```

Tsip_FmsgRun是文件消息测试的运行函数。
用户可以在ZINT Tsip_Fmsg(FILE *pstFile) 函数中修改代码，以选择是否支持解码后执行编码工作，同时也可以按照情况打印测试报文。

8.4 性能分析

在演示程序中，我们提供了SIP协议报文性能测试的程序，主要方式就是把某个消息反复的进行编解码。

如在tsip_mmsg.c 中定义了测试报文：

```
ZCHAR *m_acTsipMemMsgPerformInvite1 =
{
   "INVITE sip:172.19.19.61:5063 SIP/2.0\r\n"
   "To: sip:172.19.19.61:5063\r\n"
   "From: sip:caller@university.edu\r\n"
   "Call-ID: 0ha0isndaksdj@10.0.0.1\r\n"
   "CSeq: 8 INVITE\r\n"
   "Via: SIP/2.0/UDP 172.19.19.61:5060\r\n"
   "Content-Type: application/sdp\r\n"
   "Content-Length: 162\r\n"
   "\r\n"
   "v=0\r\n"
   "o=mhandley 29739 7272939 IN IP4 126.5.4.3\r\n"
   "s=-\r\n"
   "c=IN IP4 135.180.130.88\r\n"
   "t=0 0\r\n"
   "m=audio 49210 RTP/AVP 0 12\r\n"
   "m=video 3227 RTP/AVP 31\r\n"
   "a=rtpmap:31 LPC/8000\r\n\r\n"
};
```

然后通过调用 ZINT Tsip_MmsgPerformRun(ZINT iTimes)，输入测试次数，就可以运行。

```
/* sip test run memory message perfrom test */
ZINT Tsip_MmsgPerformRun(ZINT iTimes)
{
    struct timeb stStart, stEnd;
    ZINT i, iSecond, iMicroSecond, iErrorNum = 0;

    /* get start time */
    ftime(&stStart);

    for (i = 0; i < iTimes; i++)
    {
        if (Tsip_Mmsg(m_acTsipMemMsgPerformInvite1) != ZOK)
        {
            iErrorNum++;
            Zos_Printf("test fail message no %d.\r\n", i);
        }
    }

    /* get end time */
    ftime(&stEnd);

    iSecond = stEnd.time - stStart.time;
    iMicroSecond = stEnd.millitm - stStart.millitm;
    if (iMicroSecond < 0)
    {
        iSecond--;
        iMicroSecond += 1000;
    }

    Zos_Printf("sip perform test times: %d cost time: %ds:%dms\r\n",
              iTimes, iSecond, iMicroSecond);

    return ZOK;
}
```

同样的SIP消息，使用了同样的硬件设备 (Celeron mobile 1.3G, 384 RAM)，且Juphoon的SIP协议栈被设置成详细解析消息的全部头域和SDP消息体。

使用13个测试用例: INVITE、180、183、200、ACK、BYE、INFO、PRACK 、 INVITE(ISUP)、200(BYE)、200(INFO）、200(PRACK1）、 200(PRACK2）

分别通过**Juphoon SIP** ABNF Codec和**osip** ABNF Codec重复进行3万次操作（Codec DV，Debug版本编解码; Decode DV，Debug版本解码; Codec RV，Release版本编解码; Decode RV，Release版本解码），下图是两者的运行时间的对比。

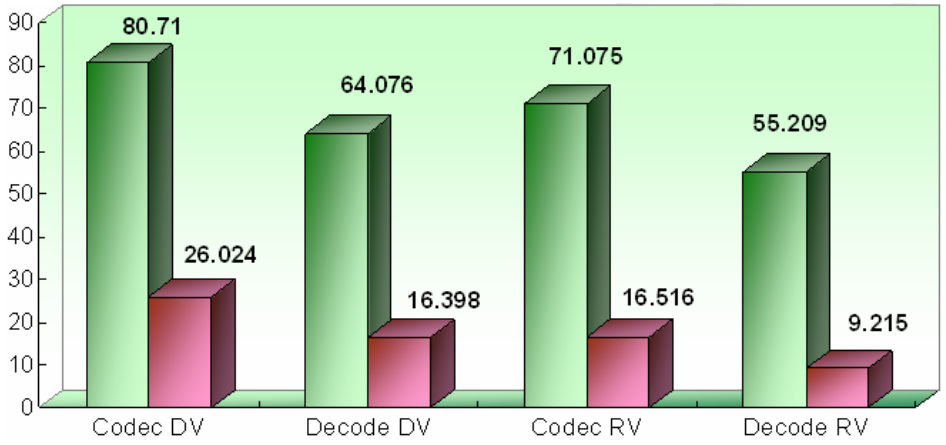

图 8-1 Juphoon SIP 同 osip 的编解码性能对比

9. 其他相关产品

Juphoon公司还提供其他软件产品，有操作系统服务平台、协议栈、解析库和编译器、协议测试工具等，详细信息请查看 http://www.juphoon.com。

9.1 操作系统服务平台

ZOS(Zero Operating System)是菊风公司的操作系统服务平台，提供了支持多种操作系统环境的统一抽象接口操作，使得软件产品能够独立于特定的处理机、编译器和操作系统等应用环境。此外，ZOS增强了系统服务功能，提供任务管理、消息队列、计时器管理、内存管理、数据缓冲区管理、日志管理，抽象了很多协议相关的服务功能接口，如ABNF、ASN.1编解码库。

9.2 协议软件

Juphoon 公司的主要协议软件有 MGCP, H.248, H.323, RTP/RTCP, HTTP, RTSP, DNS, TBCP, XCAP, STUN, SigComp, RADIUS 等 VoIP 和 3G 应用协议。

Juphoon 公司在协议栈的基础上提供业务应用软件，比如SIP User Agent，RTSP User Agent 等应用软件。同时结合多个协议栈和业务软件，Juphoon 公司提供专业的通信中间件，比如 SIP Phone Framework, IMS PoC Client Framework 等中间件。

9.3 编译器

Juphoon 公司的主要编译工具有 ABNF Parser, ASN.1 Parser, XML Parser, ABNF Compiler，Sigcomp ASM Compiler 等软件。

9.4 测试工具

Juphoon 公司提供的测试工具 有：

- ABNF2TEST 根据 ABNF 语法自动产生复杂和大量的测试用例的工具
- ABNF2HTML 根据 ABNF 语法产生其他语言的工具
- Call Engine 专门用于协议的呼叫流程测试
- CUNIT 用于 C 语言的单元测试软件
- ASM2BCODE 根据汇编代码产生 Sigcomp byte code