1. SIPp概述
   1. SIPp简介

SIPp是一个测试SIP协议性能的工具软件。它包含了一些基本的用户代理工作流程（UAC和UAS：UAC负责发起SIP呼叫请求，UAS接收UAC的请求并负责对其做出响应），并可使用INVITE和BYE建立和释放多个呼叫。

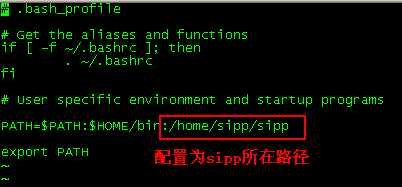
* 可以读取XML的场景文件，即描述任何性能测试的配置文件，在场景定义文件中可以使用正则表达式；
* 能动态显示测试运行的统计数据（呼叫速率、消息统计等）；
* 可以用来测试许多真实的SIP设备，也可以模仿上千个SIP代理呼叫你的SIP系统；

另外，SIPp可以用来模拟现场的SIP信令，以重现出现的故障；或者可以自定义SIP协议以测试终端对某些方面的容错或错误处理能力。

* 1. SIPp安装
     1. SIPp在Linux下安装

**安装步骤**：

1. SIPp安装到sipp账户下面（也可以安装到其他账户下），上传tar包到服务器，直接解压；
2. 执行编译脚本，进入sipp目录下，执行make；
3. 进入.bash\_profile配置环境变量，如下图所示，配置完成后执行. .bash\_profile后生效；



**启动方式**：

screen方式启动主被叫，登录sipp账户打开两个窗口，分别用于启动主叫和被叫，进入到sipp脚本目录下，执行如下命令：

1. 启动主叫：

Screen –S uac（screen的名字，可以随便命名，便于区分主被叫即可）

./caller.sh

启动成功，可以按ctrl+a+d退出screen（切到后台运行）

1. 启动被叫：

Screen –S uas

./callee.sh

启动成功，可以按ctrl+a+d退出screen（切到后台运行）

1. 再次进入screen，可以先执行screen -ls查看screen进程号，然后再执行screen -r 进程号，便可进入该screen，按ctrl+c后再输入exit，即可结束进程。
   * 1. SIPp在windows下安装

直接解压双击安装即可。

注意事项：

1. 安装前：在SIPp安装目录的根目录（如C:\)下安装cygwin或者直接在根目录下建立一个空的文件夹cygwin；
2. 安装完成后：配置环境变量，在系统变量“Path”的最后添加“;SIPp的安装目录”并保存；
3. 运行脚本时：脚本所在盘的根目录下必须包含usr文件夹，系统盘下一般都有，或者直接从其他机器上复制一份过来；
   1. SIPp使用

用SIPp做测试的时候需要准备五个文件：uac.xml，uas.xml，uac.sh，uas.sh，data.csv。（xml文件是必须的，后面三个根据情况可以使用命令或参数代替）

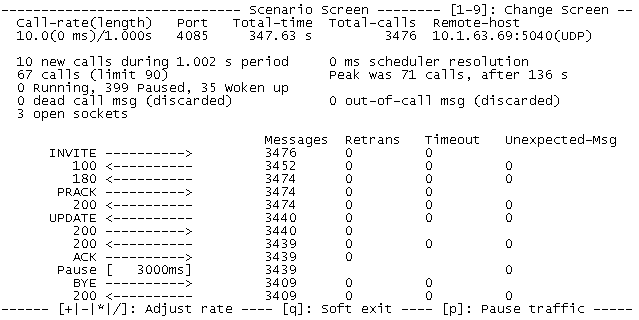
uac.xml（uas.xml）：根据实际需要编写的uac（uas）侧的sip信号流程；

uac.sh（uas.sh）：模拟主叫（被叫），调用sipp命令，并传入相应参数的批处理文件，也可不准备此文件直接输入sipp命令执行程序，但是写成文件执行更加方便可靠；

data.csv：用于uac.xml和uas.xml中需要引入的相应数据。

注：在Windows下运行时，通过usc.bat（uas.bat）代替.sh文件。

SIPp脚本运行示例：



1. SIPp核心配置文件编写

在编写xml脚本文件之前，需要先明确SIPp脚本模拟的对象、以及最终需要测试的对象，现以Volte彩铃业务为例说明，Volte彩铃业务框架如下图：



当需要测试业务平台，且不受核心网限制时，可以将主/被叫与核心网当成一个整体，通过抓取SIP前台服务器的包作为参考来编写uac.xml（uas.xml），而UAC（UAS）发送给SIP前台的消息应该是真实主（被）叫消息通过核心网后发给SIP前台的消息形式。

下面以Volte彩铃非Precondition流程为例：



将主/被叫与核心网当成一个整体，得出SIPp脚本编写所需的流程图为：



* 1. SIP消息命令

介绍一下如何创建自己的SIPp XML脚本。一个SIPp脚本总是以如下开头：

<?xml version="1.0" encoding="ISO-8859-1" ?>

<scenario name="Basic Sipstone UAC">

而且总是以下面语句结束：

</scenario>

开头和结尾很简单，至于中间的写法规则其实也不难，在SIPp的脚本文件中，有许多用于操作SIP消息的命令，以下是使用率比较高的命令详细列表：

|  |  |  |  |
| --- | --- | --- | --- |
| 命令 | 属性 | 描述 | 例子 |
| <**send>** | retrans | 只用于UDP传输：指定计数器T1值，在RFC3261，17.1.1.2章节有描述 | <send retrans="500">:  表示每500ms重传输该消息，可在没收到响应的情况下，在设定的时间之后重传。 |
| lost | 模拟丢包率 | <send lost="10">:  发送的丢包率为10% |
| **<recv>** | response | 指定SIPp期望收到的SIP消息代码，如1xx，2xx，3xx等 | <recv response="200">:  表示SIPp期望收到代码为200的消息 |
| request | 指定SIPp期望收到的SIP请求消息，如：INVITE，ACK，BYE，REGISTER，CANCEL，OPTIONS等。 | <recv request="ACK">:  表示SIPp期望收到ACK请求 |
| optional | 指定SIPp期望收到的消息是可选的，对端可以回这个期望的消息，也可以没有回这个期望的消息。 | <recv response="100" optional="true">:  表示SIPp期望收到代码为100的消息，但如果没有收到也没有关系。 |
| action | 指定当收到指定的消息时SIPp需要采取的动作 |  |
| lost | 模拟丢包率 | <recv lost="10"> |
| **<pause>** | milliseconds | 定义暂停的时间，单位为ms。当没有指定时间时，则使用命令行参数-d来指定。 | <pause milliseconds="5000"/>:  暂停脚本5秒钟 |
| **<sendCmd>** | <![CDATA[]]> | 用于SIP中第三方呼叫控制（3PCC）。CDATA必须包括Call-ID。 | <sendCmd>  <![CDATA[  Call-ID: [call\_id]  ……  ]]>  </sendCmd> |
| **<recvCmd>** | action | 当接收到命令时定义一个动作。 | <recvCmd>  <action  <ereg regexp="Content-Type:.\*"  search\_in="msg"  assign\_to="2"/>  </action>  </recvCmd> |
| **通用** | crlf | 在脚本视图中对应的位置处输出一个空行 | <send crlf="true"> |

有些命令在书写时是成对出现的，如<send> </send>是一对，有些命令在书些时不用成对出现，如<pause milliseconds="5000"/>。

* 1. SIPp脚本关键词

常用关键词具体可见下表"关键词列表"。

|  |  |  |
| --- | --- | --- |
| 关键词 | 默认值 | 说明 |
| [service] | service | 由参数-s传递，一般用来指定单个主被叫 |
| [remote\_ip] |  | 远端设备地址 |
| [remote\_port] | 5060 | 远端设备端口。可以在脚本中使用偏移量，如[remote\_port+3] |
| [transport] | UDP | 指定传输层协议，UDP/TCP，由参数-t决定 |
| [local\_ip] | 主机本地地址 | 可以由参数-i指定 |
| [local\_ip\_type] |  | ip版本 |
| [local\_port] | 由系统随机分配 | 可由-p指定，可以在脚本中使用偏移量，如[local\_port+3] |
| [len] |  | sdp长度，用于"Content-Length"头域，由sipp自动生成或者手动指定，可以添加偏移量，如[len+3] |
| [call\_number] |  | 呼叫索引，从1开始，每增加一个呼叫递增1 |
| [cseq] |  | 初始值为1，可以使用参数 -base\_cseq手动指定初始值。 |
| [media\_ip] |  | 本地媒体流ip，可以由-mi参数指定 |
| [media\_ip\_type] |  | 本地媒体流ip版本 |
| [media\_port] |  | 本地媒体流端口，可由-mp指定，可以设置偏移量[media\_port+3] |
| [last\_\*] |  | 此关键词用于从接收的上一个sip消息中提取指定头域（如果存在）的值。比如[last\_to]则表示从接收的上一个sip消息中提取To域的消息保存到[last\_to]中并应用。 |
| [branch] |  | 生成一个由(z9hG4bK) + call number + message索引组成的branch id到脚本中。 |
| [field0-n file=<filename> line=<number>] |  | 从外部文件csv加载值，file表示选择从命令行中指定的csv文件的一个文件作为外部文件，line定义选择的外部文件的起始行，field选择字段。 |

**脚本参数化：**

1. 需要sipp命令赋值的参数-p、-i、-s：[local\_ip]、[local\_port]、[remote\_ip]、[remote\_port]、[service]
2. sipp自动检测生成的参数：[call\_number]、[call\_id]、[cseq]、[len]、[branch]
   1. 创建客户端（UAC）脚本

UAC脚本以“send”命令开始，如下：

<send retrans="500">

**<![CDATA[**

INVITE sip:[service]@[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/[transport] [local\_ip]:[local\_port]

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag=[call\_number]

To: sut <sip:[service]@[remote\_ip]:[remote\_port]>

Call-ID: [call\_id]

Cseq: 1 INVITE

Contact: sip:sipp@[local\_ip]:[local\_port]

Max-Forwards: 70

Content-Type: application/sdp

Content-Length: [len]

v=0

o=user1 53655765 2353687637 IN IP[local\_ip\_type] [local\_ip]

s=-

t=0 0

c=IN IP[media\_ip\_type] [media\_ip]

m=audio [media\_port] RTP/AVP 0

a=rtpmap:0 PCMU/8000

**]]>**

</send>

在send命令内部，必须将待发送的sip消息括入"<![CDATA" 和 "]]>"中间，在这中间的所有内容将会被发送到远端系统。同时，在这个示例中包含一些特殊的关键词，比如：[service]，[remote\_ip]，这些关键词可以通过sipp命令参数来进行赋值，具体可见2.2节"关键词列表"。另外：加入retrans参数，可在没收到响应的情况下，在设定的时间之后重传，此例中为500毫秒。

**注意：在测试业务时，通过在头域中添加<Route：被叫侧地址>以保证呼叫会从业务路由到被叫，否则路由会通过Request URI确定下一跳地址。**

SIPp脚本中可以使用"recv"命令等待接收消息。如下：

<recv response="100" optional="true">

</recv>

<recv response="180" optional="true">

</recv>

<recv response="200">

</recv>

其中，100和180消息是可选接收的（optional），但200是强制接收的，**在一序列"recv"命令中，必须至少有一个消息是强制接收的**。

发送请求的时候不需要也不可能重新填写所有字段（比如说From字段不需要，因为一个dialog里的From字段都是相同的；而To字段是没办法自己填写，必需从上一个响应中引入，因为To-tag是远端加上的，本地并不知道），所以可以用[last\_字段名]的方式从上一个消息中取得。**通常From，To，Call-ID字段从上一个消息中取得**，例如：

<send retrans="3000">

<![CDATA[

PRACK sip:[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/[transport] [local\_ip]:[local\_port];branch=[branch]

CSeq: 2 PRACK

[last\_From:]

[last\_To:]

[last\_Call-ID:]

RAck: 1 1 INVITE

Max-Forwards: 70

Contact: <sip:sipp@[local\_ip]:[local\_port];transport=udp>

Content-Length: [len]

]]>

</send>

在SIPp主叫脚本中，可以通过插入一个暂停语句来模拟实际持续的通话时长，例如：在建立通话后插入语句<pause milliseconds="5000"/>为在通话状态维持5s。

暂停语句要灵活运用，还可以在被叫脚本180Ring后插入暂停来模拟振铃时长等。

* 1. 创建服务端（UAS）脚本

UAS脚本以"recv"命令开始，语法规则和可用命令跟客户端UAC脚本是一样的，不过在UAS脚本中会用到很多的[last\_\*]关键词。

UAS脚本会首先收到UAC脚本发送的invite消息：

<recv request="INVITE">

<action>

<ereg regexp=".\*" search\_in="hdr" header="Via:" check\_it="true" assign\_to="4" />

<ereg regexp=".\*" search\_in="hdr" header="CSeq:" check\_it="true" assign\_to="2" />

</action>

</recv>

**正则表达式使用说明**：在该示例中用到了正则表达式，当sipp的消息序列中带有PRACK时，UAS发送INVITE的200 OK时，某些字段（比如Via和Cseq）则不能使用[last\_字段名]方式从上一个收到的消息中引入，因为此时上一个消息是PRACK，而不是INVITE，所以需要先将INVITE的这两个字段保存下来供以后使用。上面的用法便是将INVITE的Via字段的值保存为数字1，在以后发送INVITE的200 OK的时候引用，通过Action来执行这一过程，Action的具体使用方法见2.5节。

在UAS脚本中会用到很多[last\_\*]关键词，例如：

<send>

<![CDATA[

SIP/2.0 180 Ringing

[last\_Via:]

[last\_From:]

[last\_To:];tag=[call\_number]

[last\_Call-ID:]

Require: 100rel

CSeq: 1 INVITE

Contact: <sip:[local\_ip]:[local\_port];transport=[transport]>

Content-Length: [len]

]]>

</send>

在这个脚本中是准备回复180消息，而且该180消息中的一些内容（Via、From、To、Call-ID）是从上一个接收的invite中提取出来的，但是Contact字段不能从上一个消息中引入。

**注意**：To字段从上一个消息中引入的时候，**需要添加To-tag**，call\_number为sipp自动生成的，在一个dialog中call\_number是相同的。从上一个消息引入相关字段的时候，如果上一个消息没有这个字段，则在本消息中也不会有。

在回复180Ring之后可以通过插入暂停语句来模拟真实振铃时长：

<pause milliseconds="3000"/>

**注意：被叫回复200OK摘机之前，最好暂停几秒模拟振铃时长，如果180之后立刻回复200OK，容易出现早摘机异常情况。**（该异常会导致主叫未收到180而直接收到200OK）

回复invite的200OK时，引用之前存储的变量：

<send >

<![CDATA[

SIP/2.0 200 OK

**Via:[$4]**

[last\_To:]

[last\_From:]

[last\_Call-ID:]

**CSeq:[$2]**

Contact: <sip:[local\_ip]:[local\_port];transport=[transport]>

Content-Type: application/sdp

Content-Length: [len]

v=0

o=- 197 197 IN IP4 [local\_ip]

s=SBC call

c=IN IP4 [media\_ip]

t=0 0

m=audio 28190 RTP/AVP 108 106 101 102 100 111 96

……

]]>

</send>

* 1. 动作（Actions）

在一个"recv"或者"recvCmd"命令中，可以执行一些动作，例如：

* 正则表达式（ereg）
* 给变量赋字符串值
* 记录日志（log）等等

现在主要讲一下正则表达式：

在SIPp中使用正则表达式可以实现如下功能：

* 提取SIP消息中的内容并存储到变量中以在后续中用到
* 检查SIP消息中的某些内容是否满足要求

下面是正则表达式动作的一些常用语法：

|  |  |  |
| --- | --- | --- |
| 关键词 | 默认值 | 说明 |
| regexp |  | 用于使用正则表达式匹配接收到的消息头或者消息体。".\*"用于代表所有字符串。 |
| search\_in | msg | 有四个值：  msg：匹配整个消息（匹配后可以再拆分）。  **hdr**：匹配消息头（消息头匹配后不能再拆分）。  body：匹配消息体。  var：匹配SIPp字符串变量。 |
| header |  | 匹配头域，仅在search\_in被设置为hdr时使用。 |
| check\_it | false | 设置为真时，如果不匹配则置此次呼叫为失败，不能同check\_it\_inverse同时使用。 |
| assign\_to |  | 将匹配的结果存储到指定单个变量或几个变量，使用[$n]引用变量，可以将变量[$n]的内容应用于sip消息或者用于编写sipp条件分支脚本。 |

举例：提取接收到的消息Via头，将提取的Via头分配给变量1，并在后续通过[$1]引用。

<recv request="INVITE" crlf="true">

<action>

<ereg regexp=".\*"

search\_in="hdr"

header="Via:"

check\_it="true"

assign\_to="1" />

</action>

</recv>

* 1. 关于CSeq，RAck，以及CANCEL，ACK的特殊性

对一个会话，UAC侧和UAS侧的CSeq都是按1递增的，不过CANCEL和ACK例外，它们这两个的CSeq和INVITE一致。例如：

**-->INVITE CSeq: 1 INVITE**

<--100 CSeq: 1 INVITE

<--183 CSeq: 1 INVITE

-->PRACK CSeq: 2 PRACK

<--200(PRACK) CSeq: 2 PRACK

**-->CANCEL CSeq: 1 CANCEL**

<--200(CANCEL) CSeq: 1 CANCEL

<--487 CSeq: 1 INVITE

**-->ACK CSeq: 1 ACK**

UAC侧和UAS侧的CSeq是单独的，没有关联性，只是对于Response来说，它的CSeq需要和它对应的Request一致。

再列出一个例子：

**-->INVITE CSeq: 1 INVITE**

**<--100 CSeq: 1 INVITE**

**<--183 CSeq: 1 INVITE**

-->PRACK CSeq: 2 PRACK

<--200(PRACK) CSeq: 2 PRACK

-->UPDATE CSeq: 3 UPDATE

<--200(UPDATE) CSeq: 3 UPDATE

**<--200(INVITE) CSeq: 1 INVITE**

-->ACK CSeq: 1 ACK

-->BYE CSeq: 4 BYE

<--200(BYE) CSeq: 4 BYE

PRACK中有RAck这个消息头，RAck的格式是：

RAck： RSeq CSeq Method

RSeq是对应的18X中的Rseq，CSeq是对应的18X中的CSeq，也就是INVITE的CSeq，Method是INVITE。

另外要注意的是，18X重发时，每次RSeq的值是按1递增。

例子：

-->INVITE CSeq: 1 INVITE

<--100 CSeq: 1 INVITE

**<--183 CSeq: 1 INVITE RSeq：100**

**-->PRACK CSeq: 2 PRACK RAck：100 1 INVITE**

<--200(PRACK) CSeq: 2 PRACK

<--183 CSeq: 1 INVITE RSeq：101

-->PRACK CSeq: 3 PRACK RAck：101 1 INVITE

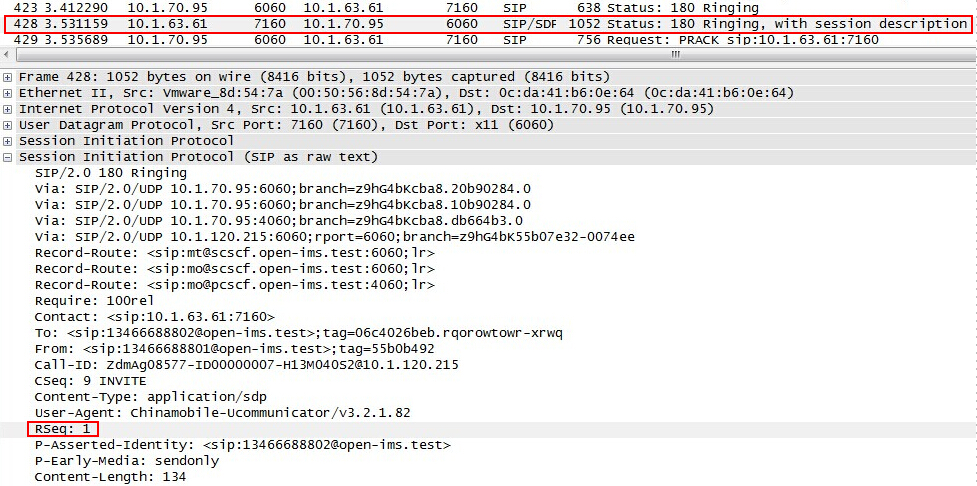
<--200(PRACK) CSeq: 3 PRACK

-->CANCEL CSeq: 1 CANCEL

<--200(CANCEL) CSeq: 1 CANCEL

<--487 CSeq: 1 INVITE

-->ACK CSeq: 1 ACK



* 1. 从外部CSV文件引入变量

SIPp可以在脚本运行命令行中使用"-inf 文件名"参数来引入变量到脚本中，例如性能测试时需要模拟不同的用户同时呼叫系统，需要通过从.csv文件中引入变量的形式来实现。

文件的第一行须申明变量的读取方式是顺序读取（**SEQUENTIAL**），还是随机读取（**RANDOM**），还是基于用户的方式读取（USER）。每一行对应一个呼叫，并使用";"分隔符分隔每一项数据，分开的项在脚本中作为变量名[filed0]、[field1]、……[fieldn]来引用。例如：

SEQUENTIAL

Sarah;sipphone32

Bob;sipphone12

Fred;sipphone94

该文件中的数据行会被按顺序读取，第一个呼叫第一行，第二个呼叫第二行。在脚本中的任何地方只要出现了关键词[field0]，根据第几个呼叫决定，这个关键词就会被替换为Sarah或者Bob或者Fred，[field1]也是类似。如果达到了文件末尾则再重新开始，一直循环，文件的大小没有限制。

SIPp脚本使用示例：



另外，可以使用参数不从第一行开始，例如从第二行开始：[field0 line=1]

还可以使用不止一个外部文件来引入变量，比如你要做一个测试主叫号码是按顺序的但是被叫是随机的时候，你就可以用一个第一行为顺序的uac.csv文件和一个第一行为随机的uas.csv文件来实现。

INVITE sip:[field0 file="uas.csv"] SIP/2.0

From: sipp user <[**field0 file="uac.csv"**]>

To: sut user <[**field0 file="uas.csv"**]>

* 1. SDP会话描述协议中的参数整理

该章节内容了解即可。

v=（protocol version） //v=0

o=（owner/creator and session identifier） //o=<用户名><会话id><版本><网络类型><地址类型><地址>

s=（session name） //会话名

c=\*（connection information） //c=<网络类型><地址信息><连接地址>

a=\*（zero or more session attribute lines） //a=<属性>、或a=<属性>:<值>

时间描述：

t=（time the session is active） //<开始时间><结束时间>，单位秒，十进制NTP

媒体描述：

m=（media name and transport address） //m=<媒体><端口><传送><格式列表>

注：v，o，s，t，m为必须的,其他项为可选。

如果SDP语法分析器不能识别某一类型(Type)，则整个描述丢失；

如果“a=”的某属性值不理解,则予以丢失

整个协议区分大小写

“=”两侧不允许有空格

所有均格式为<type>=<value>

**SDP各type的详细解释**：

* 协议版本：v=SDP版本目前为0，没有子版本
* 会话源：

o=<用户名>用户在发起主机上登录名,如果主机不支持用户标识的概念,则为“-”

<会话id>一般为数字串，其分配由创建工具决定，建议用网络时间协议(NTP)时戳，以确保唯一性。

<版本>该会话公告的版本，建议用NTP时戳。

<网络类型>为文本串“IN”

<地址类型>“IP4”/“IP6”

<地址>

* 会话名：s=ISO10646字符表示的会话名
* 连接数据：

c=<网络类型>为文本串“IN”

<地址信息>“IP4”/“IP6”

<连接地址>

* 媒体描述：

m=<媒体>有5种类型：音频/视频/应用/数据(不向用户显示的)/控制

<端口>媒体流发往传输层的端口。取决于c=行规定的网络类型和接下来的传送层协议：对UDP为1024-65535。

<传送层协议>与c=行的地址类型有关，大多为RTP/AVP。

<格式列表>对音/视频，就是音/视频应用文档中规定媒体净荷类型。

* 属性：a=（一个m=行可有多个a=行）

1. SIPp运行脚本编写
   1. 运行脚本

下面是目前常用的启动脚本的命令。

**主叫uac脚本：**

sipp -sf ./uac-np.xml -inf call-np.csv 10.1.63.69:5040 -i 10.1.70.88 -p 4085 -r 1 -m 100 -trace\_err

说明：

-trace\_err 追踪错误消息在<scenario file name>\_<pid>\_errors.log；

-sf 指定加载的xml文件名；

-inf 引入变量；

-i 设置本地地址；10.1.63.69:5040为对端地址；

-p设置本地端口；

-m 100 当100个呼叫处理完后停止；

-r 设置并发数（每秒的呼叫量）；

-trace\_msg在<scenario file name>\_<pid>\_messages.log显示发送和接受的SIP消息。

**被叫uas脚本：**

sipp -sf uas-np.xml 10.1.63.69:5040 -i 10.1.70.88 -p 4088 -m 100 -trace\_err

说明：参数说明同上。

* 1. 运行命令参数

下面的命令参数为平时进行测试时，运行脚本经常使用到的参数：

|  |  |
| --- | --- |
| 参数 | 含义 |
| -sf | 加载一个交互的xml场景文件，根据需要模拟的呼叫流程编写。 |
| -p | 设置本地端口号。默认值，系统尝试寻找一个空闲的端口从5060开始。 |
| -r | 设置并发数（每秒呼叫量）这个值可以在测试时通过按‘+’‘-’‘\*’‘/’进行更改，默认值为10。 |
| -rp | 设置呼叫速率的周期，默认是1000毫秒。例如-r 7 –rp 2000表示2秒中7个呼叫。 |
| -m | 设置本次最大呼叫个数，当‘calls’呼叫处理完停止测试并退出。 |
| -l | 最大同时保持呼叫量，默认值为3\*caps值\*呼叫时长，当因种种原因导致现存呼叫总数达到此值时，SIPp将停止产生新的呼叫，等待现存呼叫总数低于此值时才继续产生呼叫。 |
| -i | 设置本地ip地址，用于指定‘Contact:’‘Via:’和‘From:’的地址。 |
| -inf | 在呼叫过程中，从一个外部CSV文件引入值到脚本中去。文件的第一行表明数据的读取顺序。 |
| -trace\_err | 追踪所有的错误消息在<scenario file name>\_<pid>\_errors.log中 |
| -trace\_msg | 将发送和接收的sip消息保存在<scenario file name>\_<pid>\_messages.log中 |

1. 遇到的问题及解决方法
   1. 业务平台找不到UAS

原因：UAC脚本中的invite消息头为INVITE sip:[field1]@ [remote\_ip]:[remote\_port] SIP/2.0，其中[remote\_ip]:[remote\_port]为远端地址和端口，即sipserver的对外接口，而INVITE消息中又不包含Route头，导致业务平台找不到UAS

解决：在UAC的INVITE消息中添加Route头标明UAS地址，或者将上述[remote\_ip]:[remote\_port]修改为UAS地址

* 1. Sip前台回复400 Bad Reuest

原因：sipserver日志提示错误“Content Length is 1 bytes larger than body!”，sdp长度计算错误，即消息头中的Content-Length不正确

解决：将Content-Length设置为变量，由sipp自动生成，Content-Length: [len]

* 1. Sip前台拒绝200OK消息

原因：主叫发给被叫的invite消息携带sdp，但被叫回复的200OK消息缺少sdp，sipserver认为不符合sdp offer/answer流程，所以拒绝该200消息

解决：严格按照业务流程图，或者抓包检查每个消息的正确性，添加对应的sdp

* 1. UAC收到不期望的100消息

原因：主叫发起invite后，在等待响应的过程中为了防止重发，业务可能会回复100临时响应消息

解决：在UAC脚本invite消息之后添加以下语句

<recv response="100" optional="true">

</recv>

* 1. UAC等待180时却直接收到200OK

原因：UAS回复180振铃消息后立即回复200摘机消息，业务平台还未处理180消息就收到200，会丢弃180直接将200OK发送给UAC，造成早摘机异常

解决：在UAS回复200OK摘机消息之前添加以下语句：<pause milliseconds="3000"/>

* 1. 解决MS随时可能收到的OPTIONS消息

现象：当运行SIPp模拟的MS时，MS不停收到来自业务平台的OPTIONS消息，且该消息的个数，以及发送时间都不确定，导致脚本报错

解决：在MS脚本的最开始添加以下语句，业务平台发送OPTIONS消息后得不到响应会不停重发，由于消息是可选型，所以不会报错：

<recv request="OPTIONS" optional="true">

</recv>

* 1. 当流程正确时，被叫却偶尔会收到不期望的BYE

原因：由于UAS的运行命令行中存在-r、-l等参数，且xml脚本中存在通话模拟暂停语句引起的，呼叫的并发量以及暂停由UAC脚本控制即可，UAS为被动的接收消息

解决：删除UAS运行命令行中的-r、-l参数，删除UAS的xml文件中的通话模拟暂停语句

1. 参考资料

SIPp:

<http://blog.csdn.net/hanruikai/article/details/8024924>

SIPp使用手册：

<http://blog.163.com/hlz_2599/blog/static/142378474201412674153674/>

关于CSeq以及CANCEL，ACK的特殊性：

<http://blog.csdn.net/dingpeng1978/article/details/4407820>

SDP会话描述协议中的参数整理：

<http://blog.csdn.net/xu_fu/article/details/7560720>