



Performance Benchmark Test for  
OpenSER and SIP Express Router

June 4, 2007

## Revision History

Revision	Date of Issue	Changes
0.1	Feb. 24, 2007	Initial Draft.
0.2	Feb. 28, 2007	Add failover devices.
0.3	Mar. 12, 2007	Add template for recording test results
0.31	Mar. 14, 2007	Modify test results template
0.32	Mar. 19, 2007	<ul style="list-style-type: none"> <li>• Change test bed</li> <li>• Add NexOSS</li> <li>• Add multiple OSP servers</li> <li>• Use one pair of SIPp devices</li> <li>• Add weights explanation</li> <li>• Client.xml</li> <li>• Add Ringing timer</li> <li>• Test results template</li> <li>• Add 404/408 estimations</li> </ul>
0.33	Mar. 23, 2007	<ul style="list-style-type: none"> <li>• Use OSP Server 3.0.1</li> <li>• Disable TCP/TLS</li> <li>• Asyn syslog</li> </ul>
0.34	Apr. 17, 2007	Add test results for OpenSER 1.1
0.35	Apr. 17, 2007	Minor edits for presentation
0.36	Apr. 19, 2007	Add test results for OpenSER 1.2
0.37	Apr. 21, 2007	Minor edits for presentation
0.38	Apr. 26, 2007	Add test results for SER 2.0
0.39	Apr. 30, 2007	Add charts and final edits for publishing
0.40	May 3, 2007	<ul style="list-style-type: none"> <li>• Add compile time options used for each release</li> <li>• Clarify the test call scenario</li> </ul>
0.41	June 4, 2007	Add test results for SER 2.0 compiled with debug turned off <ul style="list-style-type: none"> <li>• Without DBG_QM_MALLOC</li> <li>• With F_MALLOC compile time options</li> <li>• With memdbg=100 and server_signature=0 run time options</li> </ul> Add Appendix with details on non-default SER 2.0 compile options and configuration.

Copyright TransNexus, Inc.

# Contents

Revision History .....	2
Contents .....	3
1 Introduction.....	4
1.1 Summary of Results.....	4
1.1.1 CPU Utilization.....	4
1.1.2 Memory Usage.....	5
1.1.3 Post Dial Delay .....	5
1.1.4 Call Completion.....	6
2 Test Bed Diagram .....	7
2.1 Call Scenarios .....	7
2.2 Summary of Test Bed Devices .....	8
3 Test Bed Configuration.....	8
3.1 OSP Server.....	8
3.2 CentOS .....	9
3.3 Routing Info for OSP Server.....	9
3.4 OpenSER & SER .....	10
3.5 SIPp.....	11
3.5.1 Server End.....	11
3.5.2 Client End .....	13
3.5.3 Client End for SER 2.0 without Debug Option .....	15
4 Parameters.....	17
4.1 Transport Mode.....	17
4.2 Call Limit .....	17
4.3 Timer Resolution .....	17
4.4 Frequency.....	17
4.5 Call Numbers .....	18
4.6 Screen Flush Frequency.....	18
5 SIPp Scripts.....	18
5.1 Server End.....	18
5.2 Client End .....	18
6 Test Procedure .....	18
6.1 Start Test.....	18
6.2 During Test .....	18
6.3 After Test .....	19
7 Test Results.....	19
7.1 Actual Results .....	20
7.1.1 OpenSER V1.1.....	21
7.1.2 OpenSER V1.2.....	25
7.1.3 SER V2.0 .....	28
7.1.4 SER V2.0 without debug option .....	31
Appendix.....	34
SER 2.0 without Debug Configuration.....	34

# 1 Introduction

This document describes a benchmark test and performance results for the OpenSER and SIP Express Router SIP proxies. The purpose of the stress test is to understand the performance boundary of OpenSER and SIP Express Router in a production environment. To simulate a production environment, the SIP proxy queries an external OSP server for routing information on each call and then reports call detail records to the external OSP server. Five destinations are returned to the SIP proxy for each call in random order. Four of the five destinations simulate call failure scenarios so the SIP proxy must retry the call an average of two times before the call is completed. These tests were performed on a single core of an Intel Xeon 5140 2.33 GHz CPU.

Section 1 of this document provides a graphical comparison of the test results for OpenSER V1.1, OpenSER V1.2 and SER 2.0. Section 7.1 provides the raw data collected from each test.

## 1.1 Summary of Results

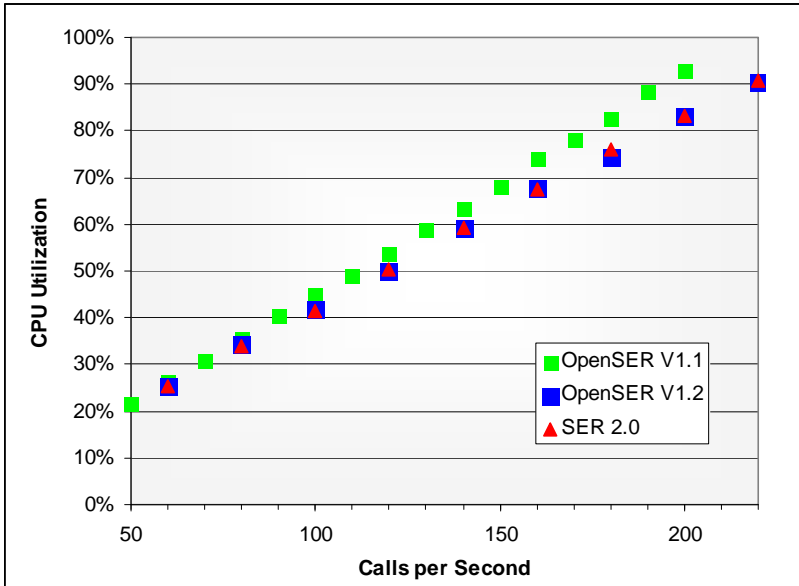
- The performance of OpenSER V1.2 and SER 2.0 are not materially different, however, there are two minor differences.
  - SER V2.0 requires less memory.
  - OpenSER V1.2 has less post dial delay.
- By all measures, OpenSER V1.2 is significantly better than OpenSER V1.1.
- For production operation (with call retries and CDR reporting), we suggest the following guideline for sizing server hardware for OpenSER V1.2 and SER V2.0:

One GHz of CPU processing capacity can manage 60 calls per second.

For example, a server with two, dual core, 3.0 GHz CPUs would effectively have twelve GHz of CPU processing capacity (2 CPUs \* 2 cores \* 3 GHz per CPU). This server, hosting either OpenSER V1.2 or SER 2.0, would be able to manage 720 calls per second at 60% CPU utilization.

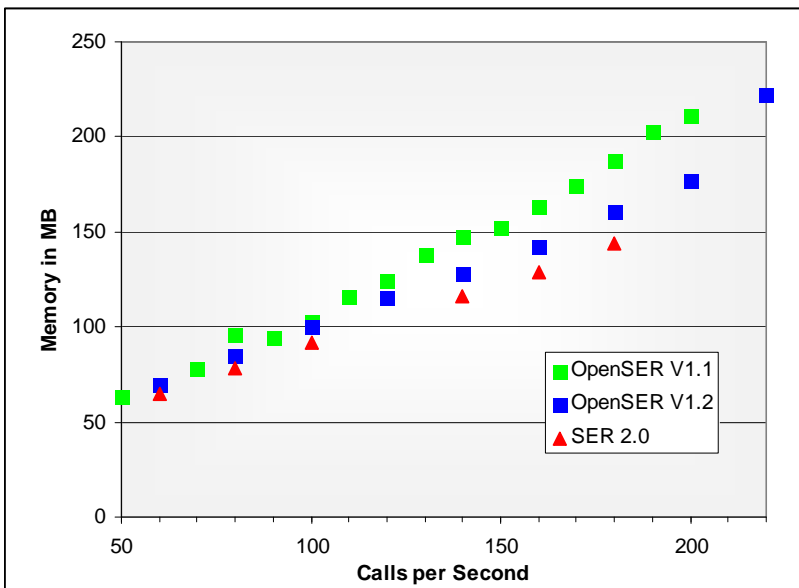
### 1.1.1 CPU Utilization

The following chart plots CPU utilization as a function of calls per second. The results for OpenSER V1.2 and SER 2.0 are identical. The performance of OpenSER V1.2 is 13.4% better than OpenSER V1.1.



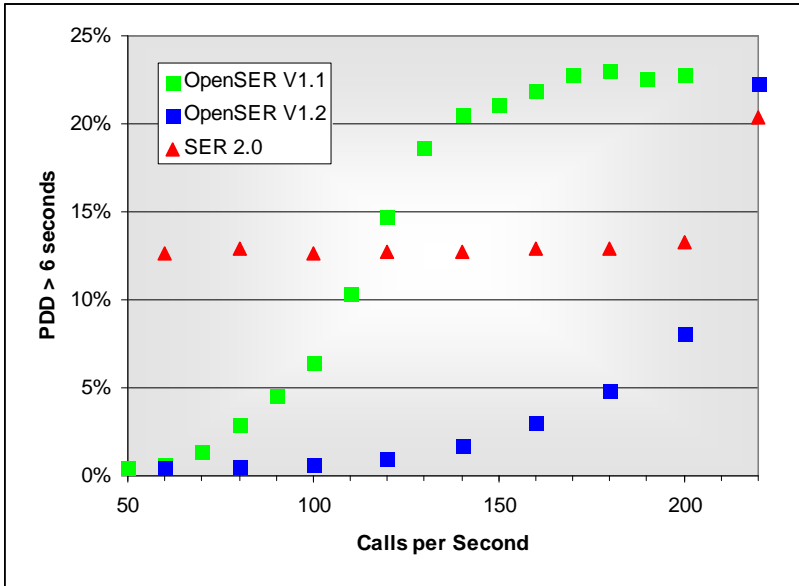
### 1.1.2 Memory Usage

SER requires less memory than OpenSER. However, shared memory is not a major resource requirement for either OpenSER or SER.



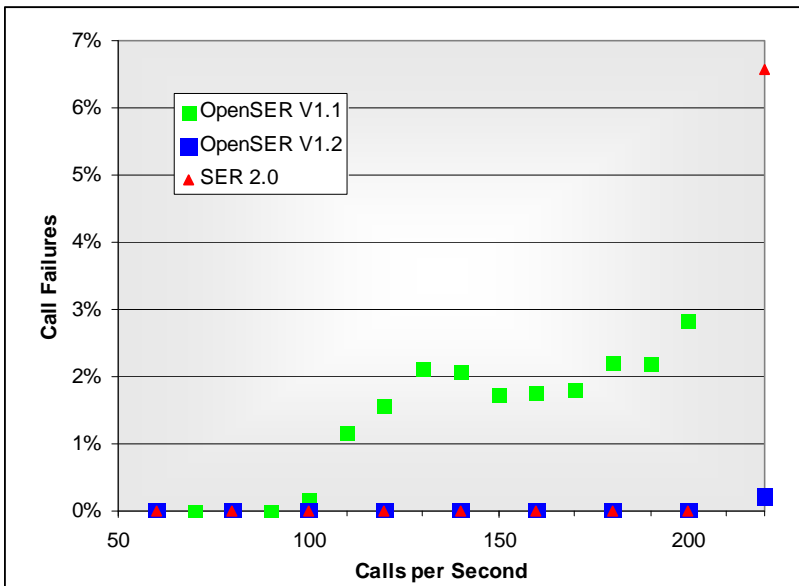
### 1.1.3 Post Dial Delay

The data on the following chart is an indirect indication of Post Dial Delay (PDD). The data presented is the percentage of calls in each test that experience a PDD greater than six seconds.

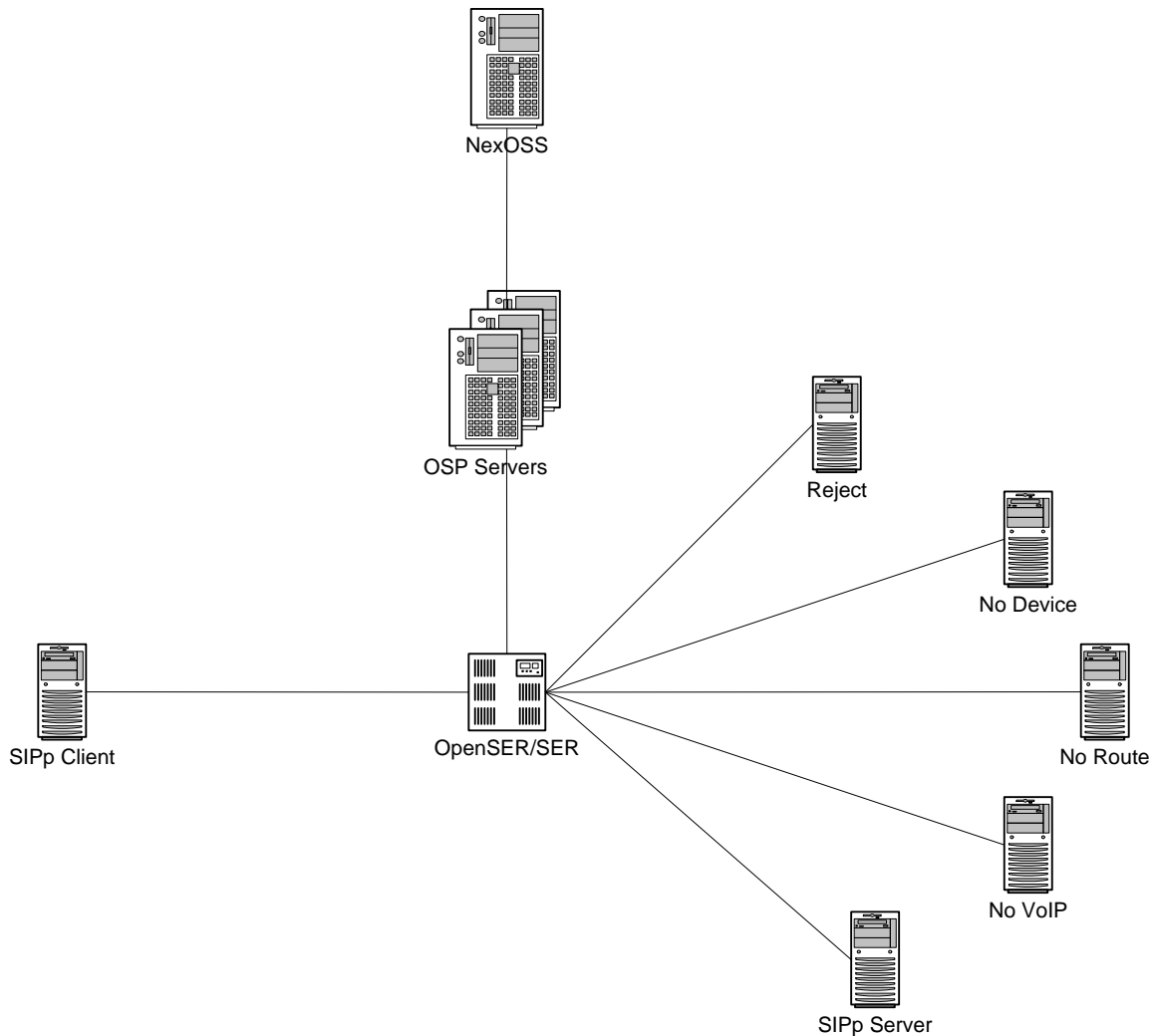


### 1.1.4 Call Completion

The following chart presents the percentage of calls which were not completed successfully for each test. Both OpenSER V1.2 and SER completed 100% of all calls successfully when CPU utilization was less than 90%



## 2 Test Bed Diagram



### 2.1 Call Scenarios

In the test bed diagram above the SIPp client generates all SIP messages for the test. The call scenario for the test is:

1. SIPp client sends a SIP INVITE to OpenSER/SER.
2. OpenSER/SER sends OSP AuthorizationRequest to an OSP server.
3. The OSP server returns five destinations in random order (Reject, No Device, No Route, No VoIP, SIPp Server). Four destinations test specific failover/retry scenarios. Only the SIPp server destination can complete the call and the other four destinations are configured to fail. Therefore, around
  - 20% of the calls are completed on the first try.
  - 20% of the calls fail on the first attempt and complete on the second attempt.
  - 20% of the calls fail on the first two attempts and complete on the third attempt.
  - 20% of the calls fail on the first three attempts and complete on the fourth attempt.
  - 20% of the calls fail on the first four attempts and complete on the fifth attempt.

4. When the call is finished, OpenSER/SER sends OSP UsageIndication messages (Call Detail Records) to the OSP server.

## 2.2 Summary of Test Bed Devices

- **NexOSS:** ultra60/172.16.4.133 NexOSS 3.04
- **OSP Servers:** t2000/172.16.4.75:1080/2080/3080/4080/5080/6080/8080/9080 OSP Server 3.0.y
- **Server hosting OpenSER and SER:** Dell Precision 490 server with two Intel Xeon 5140 dual core, 2.33 GHz, CPUs and 4 GB RAM. For this performance test, three CPU cores have been disabled. Only one CPU core was used for the test. The operating system is Centos4. Server IP address is 172.16.4.32.
- **Reject:** fedora2/172.16.4.56:5060. The Asterisk for voipuser test is running on fedora2. It will reject any number different than 4045266054. The TCCode for this destination should be 404.
- **No Device:** An IP address in 172.16.4.x network segment but no device uses this address, such as 172.16.4.100. This is a connection timeout destination. The TCCode for this destination should be 408.
- **No Route:** An IP address out of 172.16.4.x network segment, such as 1.1.1.1. This is a connection timeout destination. The TCCode for this destination should be 408.
- **No VoIP:** A PC in 172.16.4.x network segment without running SIP proxy or gateway, such as 172.16.4.1. This is a connection timeout destination. The TCCode for this destination should be 408.
- **SIPp Client:** fedora1/172.16.4.58. The port will be automatically selected by SIPp client.
- **SIPp Server:** fedora3/172.16.4.60:5060. The port must be 5060. OpenSER/SER does not support other port in recent release.

### Note:

1. In order to simulate product environment,
  - The OSP server and OpenSER/SER should be hosted on the different boxes.
  - SIPp and OpenSER/SER should be hosted on the different boxes.
  - The OSP server should return all 5 destinations for every AuthReq.
  - Multiple OSP server instances are used.
2. Since the limitation of OpenSER/SER and OSP server, the port of the destinations must be 5060, the default SIP port.

## 3 Test Bed Configuration

### 3.1 OSP Server

In order to use multiple OSP Server instances on one host, the ports of the OSP Server instances must be configured.

There are 3 places where the configurable ports must agree.

1. PORT\_BASE=N000 (in start\_osp\_server.sh)
  2. portbase=N000 (in xitami/defaults.cfg)
  3. port = N443 (in xitami/sslhtt.cfs)
- N should be 1, 2, 3, etc.

## 3.2 CentOS

The system under test has two dual core processors. In order to test only one core, edit “maxcpus=<n>” in /etc/grub.conf and then reboot. There are actually two such lines in grub.conf. We would edit both lines in the future to be on the safe side.

## 3.3 Routing Info for OSP Server

1. One Customer:
  - STRESS
1. Seven devices:
  - stress\_proxy 172.16.4.32
  - stress\_reject 172.16.4.56
  - stress\_nodevice 172.16.4.100
  - stress\_noroute 1.1.1.1
  - stress\_novoip 172.15.4.1
  - stress\_sippsrc 172.16.4.58
  - stress\_sippdst 172.16.4.60

**Note:** Except stress\_sippsrc may not terminate, all devices should be configured with OSP version 2.1.1, may terminate, is online, and has enrolled.

2. Six destinations:
  - dst\_stress: with 5 same weight devices: stress\_reject, stress\_nodevice, stress\_noroute, stress\_novoip, and stress\_sippdst. This destination is for returning destinations in random order.
  - dst\_stress\_reject: This destination is for returning destinations in certain order.
  - dst\_stress\_nodevice: This destination is for returning destinations in certain order.
  - dst\_stress\_noroute: This destination is for returning destinations in certain order.
  - dst\_stress\_novoip: This destination is for returning destinations in certain order.
  - dst\_stress\_sipp: This destination is for returning destinations in certain order.
3. Five route plans:
  - route\_sipp: for all numbers return only 1 dst\_stress\_sipp. This route plan is used for troubleshooting purpose.
  - route\_5\_sipp: for all numbers return 5 dst\_stress\_sipp. This route plan is used for troubleshooting purpose.
  - route\_sip\_reject: for all numbers return 4 dst\_stress\_rejects and 1 dst\_stress\_sipp in random order. This route plan is used for troubleshooting purpose.
  - route\_order: for all numbers return the destinations in dst\_stress\_reject, dst\_stress\_nodevice, dst\_stress\_noroute, dst\_stress\_novoip, and dst\_stress\_sipp order. This route plan is used for troubleshooting purpose.
  - route\_random: for all numbers return the destination dst\_stress. This route plan is used for the real tests.

1. Five products:
  - For number 111111111, use route plan route\_sipp. This number is used for troubleshooting purpose.
  - For number 222222222, use route plan route\_5\_sipp. This number is used for troubleshooting purpose.
  - For number 333333333, use route plan route\_sipp\_reject. This number is used for troubleshooting purpose.
  - For number 999999999, use route plan route\_order. This number is used for troubleshooting purpose.
  - For all other numbers, use route plan route\_random. These numbers are used for the real tests.
2. Dial plan 1 is used for customer STRESS for all numbers.

### 3.4 OpenSER & SER

OpenSER and SER should be compiled using default compile time options except the test for SER 2.0 without debug. The following parameters should be configured:

- OSP server IP addresses
- OSP server weights. According to our study, for the stress test, the weights should be small to generate the best load balancing results. For example, 90/10 is better than 900/100. Large weights are better for backup scenarios.
- OpenSER/SER local IP address, 172.16.4.32
- OSP key files
- Debug level. The default value is 2.
- Children. For the stress test, we use 64.
- TCP/TLS should be disabled by add “disable\_tcp=yes” and “listen=udp:172.16.4.32:5060”. It reduces the total number of processes.
- Change syn syslog to asyn syslog will reduce CPU usage for io wait.
- For SER 2.0 without debug option test
  - Comment out DBG\_QM\_MALLOC compile time option
  - Add F\_MALLOC compile time option
  - Add memdbg=100 and server\_signature=0 run time options

Compile time options for each OpenSER and SER build are provided below:

- OpenSER 1.1 compile time options

```
[root@centos4 sbin]# ./openser -V
version: openser 1.1.1-notls (i386/linux)
flags: STATS: Off, USE_IPV6, USE_TCP, DISABLE_NAGLE, USE_MCAST, SHM_MEM,
SHM_MMAP, PKG_MALLOC, F_MALLOC, FAST_LOCK-ADAPTIVE_WAIT
ADAPTIVE_WAIT_LOOPS=1024, MAX_RECV_BUFFER_SIZE 262144, MAX_LISTEN 16,
MAX_URI_SIZE 1024, BUF_SIZE 65535
poll method support: poll, epoll_lt, epoll_et, sigio_rt, select.
@(#) $Id: main.c 960 2006-07-04 17:25:54Z bogdan_iancu $
main.c compiled on 00:23:13 Apr 19 2007 with gcc 3.4.6
```

- OpenSER 1.2 compile time options

```
[root@centos4 sbin]# ./openser -V
version: openser 1.2.0-notls (i386/linux)
flags: STATS: Off, USE_IPV6, USE_TCP, DISABLE_NAGLE, USE_MCAST, SHM_MEM,
SHM_MMAP, PKG_MALLOC, F_MALLOC, FAST_LOCK-ADAPTIVE_WAIT
ADAPTIVE_WAIT_LOOPS=1024, MAX_RECV_BUFFER_SIZE 262144, MAX_LISTEN 16,
```

```

MAX_URI_SIZE 1024, BUF_SIZE 65535
poll method support: poll, epoll_lt, epoll_et, sigio_rt, select.
svnrevision: 2:1886M
@(#) $Id: main.c 1746 2007-03-05 16:06:58Z miconda $
main.c compiled on 22:04:49 Apr 17 2007 with gcc 3.4.6

```

- **SER 2.0 compile time options**

```

[root@centos4 sbin]# ./ser -V
version: ser 2.0.0-rc1 (i386/linux)
flags: STATS: Off, USE_IPV6, USE_TCP, USE_TLS, TLS_HOOKS, DISABLE_NAGLE,
USE_MCAST, DNS_IP_HACK, SHM_MEM, SHM_MMAP, PKG_MALLOC, DBG_QM_MALLOC,
FAST_LOCK-ADAPTIVE_WAIT, USE_DNS_CACHE, USE_DNS_FAILOVER, USE_DST_BLACKLIST
ADAPTIVE_WAIT_LOOPS=1024, MAX_RECV_BUFFER_SIZE 262144, MAX_LISTEN 16,
MAX_URI_SIZE 1024, BUF_SIZE 65535
poll method support: poll, epoll_lt, epoll_et, sigio_rt, select.
@(#) $Id: main.c,v 1.228.2.2 2007/03/01 13:53:37 andrei Exp $
main.c compiled on 08:34:09 Apr 24 2007 with gcc 3.4.6

```

- **SER 2.0 without debug option compile time options**

```

[root@centos4 sbin]# ./ser -V
version: ser 2.0.0-rc2 (i386/linux)
flags: STATS: Off, USE_IPV6, USE_TCP, USE_TLS, TLS_HOOKS, DISABLE_NAGLE,
USE_MCAST, DNS_IP_HACK, SHM_MEM, SHM_MMAP, PKG_MALLOC, F_MALLOC, FAST_LOCK-
ADAPTIVE_WAIT, USE_DNS_CACHE, USE_DNS_FAILOVER, USE_DST_BLACKLIST
ADAPTIVE_WAIT_LOOPS=1024, MAX_RECV_BUFFER_SIZE 262144, MAX_LISTEN 16,
MAX_URI_SIZE 1024, BUF_SIZE 65535
poll method support: poll, epoll_lt, epoll_et, sigio_rt, select.
@(#) $Id: main.c,v 1.228.2.2 2007/03/01 13:53:37 andrei Exp $
main.c compiled on 20:42:41 May 29 2007 with gcc 3.4.6

```

## 3.5 SIPp

### 3.5.1 Server End

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!-- This program is free software; you can redistribute it and/or      -->
<!-- modify it under the terms of the GNU General Public License as    -->
<!-- published by the Free Software Foundation; either version 2 of the -->
<!-- License, or (at your option) any later version.                  -->
<!--                                                                    -->
<!-- This program is distributed in the hope that it will be useful,   -->
<!-- but WITHOUT ANY WARRANTY; without even the implied warranty of    -->
<!-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the    -->
<!-- GNU General Public License for more details.                      -->
<!--                                                                    -->
<!-- You should have received a copy of the GNU General Public License -->
<!-- along with this program; if not, write to the                    -->
<!-- Free Software Foundation, Inc.,                                    -->
<!-- 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA            -->
<!--                                                                    -->
<!--          Sipp default 'uas' scenario.                             -->
<!--                                                                    -->

<scenario name="Basic UAS responder">
  <!-- By adding rrs="true" (Record Route Sets), the route sets      -->
  <!-- are saved and used for following messages sent. Useful to test  -->
  <!-- against stateful SIP proxies/B2BUAs.                            -->
  <recv request="INVITE" crlf="true">
    </recv>

  <!-- The '[last_*]' keyword is replaced automatically by the        -->
  <!-- specified header if it was present in the last message received -->

```

```

<!-- (except if it was a retransmission). If the header was not      -->
<!-- present or if no message has been received, the '[last_*]'    -->
<!-- keyword is discarded, and all bytes until the end of the line -->
<!-- are also discarded.                                           -->
<!--                                                                 -->
<!-- If the specified header was present several times in the      -->
<!-- message, all occurrences are concatenated (CRLF seperated)   -->
<!-- to be used in place of the '[last_*]' keyword.                -->

<send>
  <![CDATA[

    SIP/2.0 180 Ringing
    [last_Via:]
    [last_Record-Route:]
    [last_From:]
    [last_To:];tag=[pid]SIPpTag01[call_number]
    [last_Call-ID:]
    [last_CSeq:]
    Contact: <sip:[local_ip]:[local_port];transport=[transport]>
    Content-Length: 0

  ]]>
</send>

<send retrans="500">
  <![CDATA[

    SIP/2.0 200 OK
    [last_Via:]
    [last_Record-Route:]
    [last_From:]
    [last_To:];tag=[pid]SIPpTag01[call_number]
    [last_Call-ID:]
    [last_CSeq:]
    Contact: <sip:[local_ip]:[local_port];transport=[transport]>
    Content-Type: application/sdp
    Content-Length: [len]

    v=0
    o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
    s=-
    c=IN IP[media_ip_type] [media_ip]
    t=0 0
    m=audio [media_port] RTP/AVP 0
    a=rtpmap:0 PCMU/8000

  ]]>
</send>

<recv request="ACK"
  optional="true"
  crlf="true">
</recv>

<recv request="BYE">
</recv>

<send>
  <![CDATA[

    SIP/2.0 200 OK
    [last_Via:]
    [last_From:]

```

```

    [last_To:]
    [last_Call-ID:]
    [last_CSeq:]
    Contact: <sip:[local_ip]:[local_port];transport=[transport]>
    Content-Length: 0

  ]]>
</send>

<!-- Keep the call open for a while in case the 200 is lost to be      -->
<!-- able to retransmit it if we receive the BYE again.              -->
<pause milliseconds="4000"/>

<!-- definition of the response time repartition table (unit is ms)  -->
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

<!-- definition of the call length repartition table (unit is ms)   -->
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>

```

### 3.5.2 Client End

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!-- This program is free software; you can redistribute it and/or    -->
<!-- modify it under the terms of the GNU General Public License as   -->
<!-- published by the Free Software Foundation; either version 2 of the -->
<!-- License, or (at your option) any later version.                  -->
<!--                                                                    -->
<!-- This program is distributed in the hope that it will be useful,   -->
<!-- but WITHOUT ANY WARRANTY; without even the implied warranty of   -->
<!-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the    -->
<!-- GNU General Public License for more details.                      -->
<!--                                                                    -->
<!-- You should have received a copy of the GNU General Public License -->
<!-- along with this program; if not, write to the                    -->
<!-- Free Software Foundation, Inc.,                                    -->
<!-- 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA           -->
<!--                                                                    -->
<!--          Sipp default 'uac' scenario.                              -->
<!--                                                                    -->

<scenario name="Basic Sipstone UAC">
  <!-- In client mode (sipp placing calls), the Call-ID MUST be      -->
  <!-- generated by sipp. To do so, use [call_id] keyword.          -->
  <send retrans="500" start_rtd="1">
    <![CDATA[

      INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp
<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[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
      Subject: Performance Test
      Content-Type: application/sdp
      Content-Length: [len]

    ]>

    v=0

```

```

    o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
    s=-
    c=IN IP[media_ip_type] [media_ip]
    t=0 0
    m=audio [media_port] RTP/AVP 0
    a=rtpmap:0 PCMU/8000

  ]]>
</send>

<recv response="100" rtd="1" start_rtd="2">
</recv>

<recv response="100" rtd="2" start_rtd="3" optional="true">
</recv>

<recv response="180" rtd="3" optional="true">
</recv>

<!-- By adding rrs="true" (Record Route Sets), the route sets      -->
<!-- are saved and used for following messages sent. Useful to test -->
<!-- against stateful SIP proxies/B2BUAs.                          -->
<recv response="200" rrs="true">
  <action>
    <ereg regexp="[1-9].*" search_in="hdr" header="Contact:" check_it="true"
assign_to="1" />
  </action>
</recv>

<!-- Packet lost can be simulated in any send/recv message by      -->
<!-- by adding the 'lost = "10"'. Value can be [1-100] percent.   -->
<send>
  <![CDATA[

    ACK sip:[$1] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    [routes]
    From: sipp
<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
    To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
    Call-ID: [call_id]
    CSeq: 1 ACK
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Subject: Performance Test
    Content-Length: 0

  ]]>
</send>

<!-- This delay can be customized by the -d command-line option    -->
<!-- or by adding a 'milliseconds = "value"' option here.         -->
<pause min="1000" max="5000"/>

<!-- The 'crlf' option inserts a blank line in the statistics report. -->
<send retrans="500">
  <![CDATA[

    BYE sip:[$1] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    [routes]
    From: sipp
<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
    To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
  ]]>
</send>

```

```

    Call-ID: [call_id]
    CSeq: 2 BYE
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Subject: Performance Test
    Content-Length: 0

  ]]>
</send>

<recv response="200" crlf="true">
</recv>

<label id="1"/>

  <!-- definition of the response time repartition table (unit is ms)  -->
  <ResponseTimeRepartition value="50, 100, 200, 500, 1000, 2000, 3000, 4000,
5000, 6000, 10000"/>

  <!-- definition of the call length repartition table (unit is ms)  -->
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>

```

### 3.5.3 Client End for SER 2.0 without Debug Option

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<!-- This program is free software; you can redistribute it and/or  -->
<!-- modify it under the terms of the GNU General Public License as  -->
<!-- published by the Free Software Foundation; either version 2 of the  -->
<!-- License, or (at your option) any later version.  -->
<!--  -->
<!-- This program is distributed in the hope that it will be useful,  -->
<!-- but WITHOUT ANY WARRANTY; without even the implied warranty of  -->
<!-- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  -->
<!-- GNU General Public License for more details.  -->
<!--  -->
<!-- You should have received a copy of the GNU General Public License  -->
<!-- along with this program; if not, write to the  -->
<!-- Free Software Foundation, Inc.,  -->
<!-- 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA  -->
<!--  -->
<!--           Sipp default 'uac' scenario.  -->
<!--  -->

<scenario name="Basic Sipstone UAC">
  <!-- In client mode (sipp placing calls), the Call-ID MUST be  -->
  <!-- generated by sipp. To do so, use [call_id] keyword.  -->
  <send retrans="500" start_rtd="1">
    <![CDATA[

      INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp
<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[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
      Subject: Performance Test
      Content-Type: application/sdp
      Content-Length: [len]

```

```

v=0
o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
s=-
c=IN IP[media_ip_type] [media_ip]
t=0 0
m=audio [media_port] RTP/AVP 0
a=rtpmap:0 PCMU/8000

  ]]>
</send>

<recv response="100" rtd="1" start_rtd="2">
</recv>

<recv response="100" rtd="2" start_rtd="3" optional="true">
</recv>

<recv response="180" rtd="3" optional="true">
</recv>

<!-- By adding rrs="true" (Record Route Sets), the route sets      -->
<!-- are saved and used for following messages sent. Useful to test -->
<!-- against stateful SIP proxies/B2BUAs.                          -->
<recv response="200" rrs="true">
  <action>
    <ereg regexp="[1-9].*" search_in="hdr" header="Contact:" check_it="true"
assign_to="1" />
  </action>
</recv>

<!-- Packet lost can be simulated in any send/recv message by     -->
<!-- by adding the 'lost = "10"'. Value can be [1-100] percent.   -->
<send>
  <![CDATA[

    ACK sip:[$1] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    [routes]
    From: sipp
<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
    To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
    Call-ID: [call_id]
    CSeq: 1 ACK
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Subject: Performance Test
    Content-Length: 0

  ]]>
</send>

<!-- This delay can be customized by the -d command-line option   -->
<!-- or by adding a 'milliseconds = "value"' option here.        -->
<pause min="1000" max="5000"/>

<!-- The 'crlf' option inserts a blank line in the statistics report. -->
<send retrans="500">
  <![CDATA[

    BYE sip:[$1] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    [routes]

```

```

    From: sipp
<sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
    To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
    Call-ID: [call_id]
    CSeq: 2 BYE
    Contact: sip:sipp@[local_ip]:[local_port]
    Max-Forwards: 70
    Subject: Performance Test
    Content-Length: 0

  ]]>
</send>

<recv response="200" crlf="true">
</recv>

<label id="1"/>

<!-- definition of the response time repartition table (unit is ms) -->
<ResponseTimeRepartition value="50, 100, 200, 500, 1100, 2100, 3100, 4100,
5100, 6100, 10000"/>

<!-- definition of the call length repartition table (unit is ms) -->
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>

```

## 4 Parameters

### 4.1 Transport Mode

SIPp support multiple transport modes. It can be set by command line, “*-t mode*”, from the client ends. The default setting is UDP one socket. To simulate product environments, multiple sockets may be used.

### 4.2 Call Limit

This parameter is used to control the maximum number of simultaneous calls. It can be set by command line, “*-l limit*”, from the client ends. It should be set to a large number, such as 5000, for heavy load.

### 4.3 Timer Resolution

This parameter can be set by command line, “*-timer\_resol ms*”, from the client ends. The default timer resolution is 200ms. We want a more precise scheduling, this value is set to 50ms.

### 4.4 Frequency

The call rate can be set by command line, “*-r rate (cps)*”, from the client ends. This parameter should be set according to two facts. First, the total SIP traffic load should reach a certain level. For example, the load should be more than 36,000 calls / hour. Another fact is the capability of the OSP servers. If the OSP servers cannot handle the heavy traffic load, more OSP servers should be used. The default rate is set to 10 calls / second.

## 4.5 Call Numbers

This parameter is used to stop the test when the total number of the calls reaches this parameter. It can be set by command line, “*-m number*”, from the client ends.

## 4.6 Screen Flush Frequency

This parameter is used to set the flush frequency of displaying test results on the screen. It can be set by command line, “*-f ms*”, from the client ends. In order to reduce the overhead, this value should be set to a large number, such as 30000.

# 5 SIPp Scripts

**Note:** All the SIPp server ends should be started before any SIPp client is started.

For SIPp client, we need to collect data on the following time intervals:

- Time from INVITE sent until first Trying message.
- Time from first Trying message to second Trying message.
- Time from second Trying message to RINGING message.

## 5.1 Server End

```
sipp -sf server.xml -nd -i 172.16.4.60
```

**Note:** The SIPp server end can be run at background using “*-bg*” command line parameter.

## 5.2 Client End

```
sipp -sf client.xml -m 120000 -r 200 -l 5000 -s 9876543210 -i 172.16.4.58 -nd -  
timer_resol 50 -pause_msg_ign -f 30000 -fd 30 172.16.4.32 -trace_stat
```

**Note:** “*-trace\_screen*” can be used to collect the test results.

# 6 Test Procedure

## 6.1 Start Test

- Restart SIPp server on fedora3
- Stop all OSP Server instances on t2000
- Erase all old CDRs
- Start all OSP Server instances
- Restart OpenSER/SER on centos4 with shared memory 256M
- Start SIPp client on fedora1

## 6.2 During Test

- Collect CPU usages for OpenSER/SER, OSP Server instances, SIPp client and server by using top command.

**Note:** In order to reduce the overhead, the top flush frequency should be set at least 10 sec.

## 6.3 After Test

- Collect statistics data from sressn or client\_<pid>\_screen.log if “*-trace\_screen*” is set.
- Use “*/sbin/ifconfig*” to collect traffic load data
- Use “*/usr/local/sbin/openserctl fifo get\_statistics shmem:*” to collect max used shared memory data for OpenSER. SER does not have this feature.
- Collect CDRs from all OSP Server instances and calculate the numbers of different CDRs.
- Run CDR pull on NexOSS to generate traffic report.

## 7 Test Results

The test results will be displayed on SIPp client end screens. The total number of calls, successful calls, and failed calls will be displayed.

The best condition is all calls complete without any unexpected message and without any message loses.

The Ringing message losing with call completing is acceptable when the traffic is heavy.

A low percentage call failing may be acceptable for very heavy traffic conditions.

### Note:

1. The numbers of different CDRs can be calculated using the command “*grep -P “\tTCCode\t” \*.cdr | wc -l*”
2. The NexOSS report may contain the calls for other tests.
3. OpenSER/SER configuration: PKG=1M; Shmem=128M (256 for SER 2.0); Disable TCP/TLS; Children=64; Debug=2; 10 minutes
4. OSP Server configuration: 8 instances on T2000
5. There are several data about the OpenSER/SER performance:
  - Number of attempts
  - SIPp report
  - OSP Server CDRs
  - NexOSS report

SIPp reported number of completed calls is most important. It means from the source side how many calls completed.

NexOSS report came from OSP Server CDRs. They should be totally same. There may be duplicated CDRs, only 10016 CDRs, caused by the re-sending BYE messages out of OpenSER/SER absorb timeout.

The relationship of the numbers should be:

- Number of internal CDRs == 5 \* number of attempts
- Number of SIPp completed calls <= number of 200 CDRs
- Number of SIPp completed calls <= number of 10016 CDRs

There is another reason that number of 10016 CDRs is larger than number of SIPp completed calls except the duplication. The 200 OK for the BYE messages may reach OpenSER/SER but did not reach the SIPp client.

## **7.1 Actual Results**

The following pages present raw data collected from tests with:

1. OpenSER V1.1
2. OpenSER V1.2
3. SER V2.0
4. SER V2.0 without debug

## 7.1.1 OpenSER V1.1

<b>Date Test Performed</b>	4/14/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>OpenSER Version</b>	1.1	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		50	60	70	80
<b>SIPp Traffic Report</b>	<b>Completed</b>	30000	36000	42000	48000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of OSP Server CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	30000	36000	42000	48000
	<b>Actual</b>	30000	36000	42000	48000
<b>10016 CDRs</b>	<b>Expected</b>	30000	36000	42000	48000
	<b>Actual</b>	30000	36000	42000	48000
<b>Internal CDRs</b>	<b>Expected</b>	150000	180000	210000	240000
	<b>Actual</b>	150000	180000	210000	240000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~15000	~18000	~21000	~24000
	<b>Actual</b>	14767	17373	20563	24092
<b>SIP 408 CDRs</b>	<b>Expected</b>	~45000	~54000	~63000	~72000
	<b>Actual</b>	44930	54807	63571	73542
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	30000	36000	42000	48000
	<b>Actual</b>	30000	36000	42000	48000
<b>SIP 404</b>	<b>Expected</b>	~15000	~18000	~21000	~24000
	<b>Actual</b>	14767	17373	20563	24092
<b>SIP 408</b>	<b>Expected</b>	~45000	~54000	~63000	~72000
	<b>Actual</b>	44930	54807	63571	73542
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>OpenSER Server</b>	<b>CPU idle</b>	78.4%	73.6%	69.1%	64.3%
	<b>IO wait</b>	0.2%	0.2%	0.1%	0.2%
	<b>Memory</b>	112M~175M	129M~199M	112M~190M	121M~217M
	<b>RX/TX</b>	247M/273M	365M/450M	341M/441M	489M/575M
	<b>Shmem</b>	18M	22M	26M	29M
<b>OSP Server</b>	<b>CPU idle</b>	92.4%	91.0%	89.2%	87.4%
	<b>xitami</b>	0.48%	0.55%	0.65%	.74%
<b>SIPp</b>	<b>Client</b>	8.6%	10.0%	12.0%	14.6%
	<b>Server</b>	6.6%	8.0%	9.8%	10.9%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	30000	36000	42000	48000
	<b>&lt; 100 ms</b>	0	0	0	0
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	29840	35549	41111	46003
	<b>&lt; 100 ms</b>	160	446	879	1941
	<b>&lt; 200 ms</b>	0	5	10	56
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	7761	8635	10590	11933
	<b>&lt; 2 s</b>	6867	8571	9446	10628
	<b>&lt; 3 s</b>	126	147	170	232
	<b>&lt; 4 s</b>	7618	9626	11138	12387
	<b>&lt; 5 s</b>	178	222	284	535
	<b>&lt; 6 s</b>	7302	8568	9780	10871
	<b>&gt; 10 s</b>	143	228	575	1386
	<b>&gt; 10 s</b>	0	0	0	0

<b>Date Test Performed</b>	4/14/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>OpenSER Version</b>	1.1	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		90	100	110	120
<b>SIPp Traffic Report</b>	<b>Completed</b>	54000	59900	65235	70877
	<b>Completed%</b>	100%	99.83%	98.84%	98.44%
<b>Analysis of OSP Server CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	54000	60000	66000	72000
	<b>Actual</b>	54000	59900	65244	70897
<b>10016 CDRs</b>	<b>Expected</b>	54000	60000	66000	72000
	<b>Actual</b>	54000	59930	65551	71332
<b>Internal CDRs</b>	<b>Expected</b>	270000	300000	330000	360000
	<b>Actual</b>	270000	300000	330000	360000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~27000	~30000	~33000	~36000
	<b>Actual</b>	27116	29861	32359	35852
<b>SIP 408 CDRs</b>	<b>Expected</b>	~81000	~90000	~99000	~108000
	<b>Actual</b>	82561	91212	101428	111848
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	54000	60000	66000	72000
	<b>Actual</b>	54000	59930	65551	71332
<b>SIP 404</b>	<b>Expected</b>	~27000	~30000	~33000	~36000
	<b>Actual</b>	27116	29861	32359	35852
<b>SIP 408</b>	<b>Expected</b>	~81000	~90000	~99000	~108000
	<b>Actual</b>	82561	91212	101428	111848
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>OpenSER Server</b>	<b>CPU idle</b>	59.6%	55.3%	51.0%	46.1%
	<b>IO wait</b>	0.1%	0.1%	0.2%	0.1%
	<b>Memory</b>	111M~205M	112M~215M	111M~227M	112M~237M
	<b>RX/TX</b>	476M/626M	539M/700M	637M/806M	683M/872M
	<b>Shmem</b>	33M	37M	43M	48M
<b>OSP Server</b>	<b>CPU idle</b>	86.1%	84.1%	82.8%	80.8%
	<b>xitami</b>	0.85%	0.95%	1.03%	1.15%
<b>SIPp</b>	<b>Client</b>	15.2%	17.6%	11.4%	12.6%
	<b>Server</b>	11.9%	13.4%	14.7%	16.7%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	53998	60000	65932	71639
	<b>&lt; 100 ms</b>	0	0	68	356
	<b>&lt; 200 ms</b>	0	0	0	4
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	2	0	0	1
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	50525	54636	58218	61106
	<b>&lt; 100 ms</b>	3331	4820	5891	7097
	<b>&lt; 200 ms</b>	143	543	1891	3793
	<b>&lt; 500 ms</b>	1	1	0	2
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	13436	15246	16614	18007
	<b>&lt; 2 s</b>	11884	12350	11263	9856
	<b>&lt; 3 s</b>	435	1164	3337	5362
	<b>&lt; 4 s</b>	13237	13608	12555	11368
	<b>&lt; 5 s</b>	1233	2425	5294	8141
	<b>&lt; 6 s</b>	11275	11148	9226	7304
	<b>&lt; 10 s</b>	2471	3897	6864	10628
<b>&gt; 10 s</b>	0	0	0	0	

<b>Date Test Performed</b>	4/14/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>OpenSER Version</b>	1.1	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		130	140	150	160
<b>SIPp Traffic Report</b>	<b>Completed</b>	76353	82269	88430	94313
	<b>Completed%</b>	97.89%	97.94%	98.26%	98.24%
<b>Analysis of OSP Server CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	78000	84000	90000	96000
	<b>Actual</b>	76445	82519	88776	94736
<b>10016 CDRs</b>	<b>Expected</b>	78000	84000	90000	96000
	<b>Actual</b>	77133	83585	89889	95968
<b>Internal CDRs</b>	<b>Expected</b>	390000	420000	450000	480000
	<b>Actual</b>	390000	420000	450000	480000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~39000	~42000	~45000	~48000
	<b>Actual</b>	39313	42240	45136	47843
<b>SIP 408 CDRs</b>	<b>Expected</b>	~117000	~126000	~135000	~144000
	<b>Actual</b>	122348	130555	138293	147792
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	78000	84000	90000	96000
	<b>Actual</b>	77133	83585	89899	95968
<b>SIP 404</b>	<b>Expected</b>	~39000	~42000	~45000	~48000
	<b>Actual</b>	39313	42240	45136	47843
<b>SIP 408</b>	<b>Expected</b>	~117000	~126000	~135000	~144000
	<b>Actual</b>	122348	130555	138293	147792
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>OpenSER Server</b>	<b>CPU idle</b>	41.1%	36.5%	31.9%	26.1%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	111M~249M	112M~259M	1690M~1842M	1732M~1895M
	<b>RX/TX</b>	730M/939M	766M/1020M	834M/1.1G	1.0G/1.2G
	<b>Shmem</b>	55M	59M	63M	69M
<b>OSP Server</b>	<b>CPU idle</b>	78.9%	77.5%	76.1%	73.7%
	<b>xitami</b>	1.23%	1.32%	1.44%	1.52%
<b>SIPp</b>	<b>Client</b>	14.6%	15.7%	21.8%	23.8%
	<b>Server</b>	17.5%	19.5%	20.6%	21.2%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	76976	81959	86778	91020
	<b>&lt; 100 ms</b>	926	1691	2235	3313
	<b>&lt; 200 ms</b>	59	162	517	820
	<b>&lt; 500 ms</b>	7	22	34	54
	<b>&gt;500 ms</b>	32	166	436	793
	<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	63659	65077	66723
<b>&lt; 100 ms</b>		8670	10180	11497	13719
<b>&lt; 200 ms</b>		5603	8515	11317	14597
<b>&lt; 500 ms</b>		15	115	283	661
<b>&gt; 500 ms</b>		0	0	0	0
<b>2nd 100 to Ringing</b>		<b>&lt; 1 s</b>	19361	21340	22777
	<b>&lt; 2 s</b>	8774	8459	8666	8672
	<b>&lt; 3 s</b>	6402	6547	7444	8023
	<b>&lt; 4 s</b>	10639	10020	9320	8117
	<b>&lt; 5 s</b>	10720	12680	14292	15966
	<b>&lt; 6 s</b>	5245	4446	4375	4072
	<b>&lt; 10 s</b>	14544	17249	19016	21039
	<b>&gt; 10 s</b>	0	0	0	0

<b>Date Test Performed</b>	4/14/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>OpenSER Version</b>	1.1	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		170	180	190	200
<b>SIPp Traffic Report</b>	<b>Completed</b>	100158	105608	111513	116606
	<b>Completed%</b>	98.19%	97.79%	97.82%	97.17%
<b>Analysis of OSP Server CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	102000	108000	114000	120000
	<b>Actual</b>	100714	106344	112510	117782
<b>10016 CDRs</b>	<b>Expected</b>	102000	108000	114000	120000
	<b>Actual</b>	102310	108363	115079	120437
<b>Internal CDRs</b>	<b>Expected</b>	510000	540000	570000	600000
	<b>Actual</b>	510000	540000	570000	600000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~51000	~54000	~57000	~60000
	<b>Actual</b>	51541	54356	57195	60545
<b>SIP 408 CDRs</b>	<b>Expected</b>	~153000	~162000	~171000	~180000
	<b>Actual</b>	157610	167136	176365	186792
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	1
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	102000	108000	114000	120000
	<b>Actual</b>	102310	108363	115079	120437
<b>SIP 404</b>	<b>Expected</b>	~51000	~54000	~57000	~60000
	<b>Actual</b>	51541	54356	57195	60545
<b>SIP 408</b>	<b>Expected</b>	~153000	~162000	~171000	~180000
	<b>Actual</b>	157610	167136	176365	186792
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	1
<b>System Utilization</b>					
<b>OpenSER Server</b>	<b>CPU idle</b>	22.0%	17.6%	11.6%	7.1%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	1776M~1950M	111M~299M	160M~363M	112M~323M
	<b>RX/TX</b>	1.0G/1.4G	1.0G/1.3G	1.1G/1.4G	1.2G/1.4G
	<b>Shmem</b>	74M	82M	91M	98M
<b>OSP Server</b>	<b>CPU idle</b>	72.0%	70.8%	69.5%	66.4%
	<b>xitami</b>	1.65%	1.72%	1.82%	1.93%
<b>SIPp</b>	<b>Client</b>	23.6%	30%	31.1%	32.4%
	<b>Server</b>	23.2%	24.8%	27.4%	29.4%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	94655	98182	101184	101706
	<b>&lt; 100 ms</b>	4367	4927	5517	7378
	<b>&lt; 200 ms</b>	1338	2139	2980	5099
	<b>&lt; 500 ms</b>	99	118	133	183
	<b>&gt; 500 ms</b>	1541	2634	4186	5634
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	65216	61701	54582	41049
	<b>&lt; 100 ms</b>	17174	21882	26712	29675
	<b>&lt; 200 ms</b>	17905	22005	29356	40771
	<b>&lt; 500 ms</b>	1484	2113	3088	8270
	<b>&gt; 500 ms</b>	0	0	0	23
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	25240	26696	27656	28993
	<b>&lt; 2 s</b>	8235	7123	6319	4133
	<b>&lt; 3 s</b>	8527	8857	9315	11022
	<b>&lt; 4 s</b>	7258	5926	5217	3585
	<b>&lt; 5 s</b>	16641	17723	18615	16098
	<b>&lt; 6 s</b>	4432	4044	3871	7092
	<b>&lt; 10 s</b>	23245	24802	25651	27328
<b>&gt; 10 s</b>	0	0	0	3	

## 7.1.2 OpenSER V1.2

<b>Date Test Performed</b>	4/18/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>OpenSER Version</b>	1.2	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		60	80	100	120
<b>SIPp Traffic Report</b>	<b>Completed</b>	36000	48000	60000	72000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of OSP Server CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>10016 CDRs</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>Internal CDRs</b>	<b>Expected</b>	180000	240000	300000	360000
	<b>Actual</b>	180000	240000	300000	360000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~18000	~24000	~30000	~36000
	<b>Actual</b>	17243	23785	29594	35106
<b>SIP 408 CDRs</b>	<b>Expected</b>	~54000	~72000	~90000	~108000
	<b>Actual</b>	55188	72823	90304	109011
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>SIP 404</b>	<b>Expected</b>	~18000	~24000	~30000	~36000
	<b>Actual</b>	17243	23785	29594	35106
<b>SIP 408</b>	<b>Expected</b>	~54000	~72000	~90000	~108000
	<b>Actual</b>	55188	72823	90304	109011
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>OpenSER Server</b>	<b>CPU idle</b>	74.5%	65.9%	58.1%	50.1%
	<b>IO wait</b>	0.2%	0.1%	0.1%	0.2%
	<b>Memory</b>	114M~184M	113M~198M	113M~213M	113M~228M
	<b>RX/TX</b>	330M/441M	410M/583M	536M/750M	716M/954
<b>OSP Server</b>	<b>Shmem</b>	23M	29M	36M	43M
	<b>CPU idle</b>	90.9%	87.8%	84.3%	81.0%
<b>SIPp</b>	<b>xitami</b>	0.55%	0.74%	0.95%	1.14%
	<b>Client</b>	5.0%	7.1%	9.2%	13.1%
	<b>Server</b>	8.1%	10.8%	13.6%	16.1%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	36000	48000	60000	72000
	<b>&lt; 100 ms</b>	0	0	0	0
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	35723	46553	56131	64556
	<b>&lt; 100 ms</b>	276	1412	3752	6641
	<b>&lt; 200 ms</b>	1	35	117	803
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	8470	11952	15392	17905
	<b>&lt; 2 s</b>	8606	10946	12928	15302
	<b>&lt; 3 s</b>	174	285	801	1699
	<b>&lt; 4 s</b>	9617	12533	15550	18122
	<b>&lt; 5 s</b>	214	323	473	1092
	<b>&lt; 6 s</b>	8734	11693	14394	16737
	<b>&lt; 10 s</b>	174	255	390	715
<b>&gt; 10 s</b>	0	0	0	0	

<b>Date Test Performed</b>	4/18/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>OpenSER Version</b>	1.2	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		140	160	180	200
<b>SIPp Traffic Report</b>	<b>Completed</b>	84000	96000	108000	120000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of OSP Server CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>10016 CDRs</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>Internal CDRs</b>	<b>Expected</b>	420000	480000	540000	600000
	<b>Actual</b>	420000	480000	540000	600000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~42000	~48000	~54000	~60000
	<b>Actual</b>	41493	47587	53462	59609
<b>SIP 408 CDRs</b>	<b>Expected</b>	~126000	~144000	~162000	~180000
	<b>Actual</b>	127051	145825	163477	181970
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>SIP 404</b>	<b>Expected</b>	~42000	~48000	~54000	~60000
	<b>Actual</b>	41493	47587	53462	59609
<b>SIP 408</b>	<b>Expected</b>	~126000	~144000	~162000	~180000
	<b>Actual</b>	127051	145825	163477	181970
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>OpenSER Server</b>	<b>CPU idle</b>	40.8%	32.2%	25.7%	16.7%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	113M~241M	114M~256M	113M~274M	113M~290M
	<b>RX/TX</b>	805M/1050M	988M/1187M	1070M/1383M	1179M/1585M
	<b>Shmem</b>	51M	58M	66M	74M
<b>OSP Server</b>	<b>CPU idle</b>	77.5%	74.5%	71.1%	67.0%
	<b>xitami</b>	1.35%	1.52%	1.74%	1.92%
<b>SIPp</b>	<b>Client</b>	14.8%	18.9%	21.5%	27.4%
	<b>Server</b>	19.1%	22.5%	24.7%	28.3%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	83998	95992	107793	118660
	<b>&lt; 100 ms</b>	0	8	203	1204
	<b>&lt; 200 ms</b>	0	0	4	136
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	2	0	0	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	71345	75819	76045	66971
	<b>&lt; 100 ms</b>	9372	12275	18167	27994
	<b>&lt; 200 ms</b>	3273	7894	13695	24202
	<b>&lt; 500 ms</b>	5	12	93	805
	<b>&gt; 500 ms</b>	1	0	0	18
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	21360	24029	27070	30072
	<b>&lt; 2 s</b>	16916	17066	16717	17414
	<b>&lt; 3 s</b>	2082	3951	7095	9643
	<b>&lt; 4 s</b>	20549	23003	24345	23658
	<b>&lt; 5 s</b>	1805	2662	3963	7435
	<b>&lt; 6 s</b>	18905	20215	20470	18713
	<b>&lt; 10 s</b>	1442	2923	5214	9762
<b>&gt; 10 s</b>	0	0	0	0	

<b>Date Test Performed</b>	4/18/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used
<b>OpenSER Version</b>	1.2	<b>Host Memory</b>	4 GB
<b>Test Rate</b>	220		
<b>SIPp Traffic Report</b>	<b>Completed</b>	131724	
	<b>Completed%</b>	99.79%	
<b>Analysis of OSP Server CDRs</b>			
<b>200 CDRs</b>	<b>Expected</b>	132000	
	<b>Actual</b>	131878	
<b>10016 CDRs</b>	<b>Expected</b>	132000	
	<b>Actual</b>	132299	
<b>Internal CDRs</b>	<b>Expected</b>	660000	
	<b>Actual</b>	660025	
<b>SIP 404 CDRs</b>	<b>Expected</b>	~66000	
	<b>Actual</b>	65471	
<b>SIP 408 CDRs</b>	<b>Expected</b>	~198000	
	<b>Actual</b>	200703	
<b>All Other Error CDRs</b>	<b>Expected</b>	0	
	<b>Actual</b>	0	
<b>NexOSS Traffic Report</b>			
<b>16 Normal Call Clearing</b>	<b>Expected</b>	132000	
	<b>Actual</b>	132299	
<b>SIP 404</b>	<b>Expected</b>	~66000	
	<b>Actual</b>	65471	
<b>SIP 408</b>	<b>Expected</b>	~198000	
	<b>Actual</b>	200703	
<b>All Other</b>	<b>Expected</b>	0	
	<b>Actual</b>	0	
<b>System Utilization</b>			
<b>OpenSER Server</b>	<b>CPU idle</b>	9.5%	
	<b>IO wait</b>	0.1%	
	<b>Memory</b>	113M~335M	
	<b>RX/TX</b>	1279M/1685M	
	<b>Shmem</b>	103M	
<b>OSP Server</b>	<b>CPU idle</b>	63.4%	
	<b>xitami</b>	2.08%	
<b>SIPp</b>	<b>Client</b>	29.8%	
	<b>Server</b>	31.2%	
<b>SIPp Average Response Time Repartition</b>			
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	94507	
	<b>&lt; 100 ms</b>	16731	
	<b>&lt; 200 ms</b>	14336	
	<b>&lt; 500 ms</b>	3410	
	<b>&gt; 500 ms</b>	3016	
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	13335	
	<b>&lt; 100 ms</b>	20993	
	<b>&lt; 200 ms</b>	49137	
	<b>&lt; 500 ms</b>	40298	
	<b>&gt; 500 ms</b>	237	
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	31859	
	<b>&lt; 2 s</b>	8594	
	<b>&lt; 3 s</b>	14847	
	<b>&lt; 4 s</b>	4777	
	<b>&lt; 5 s</b>	21483	
	<b>&lt; 6 s</b>	6640	
	<b>&lt; 10 s</b>	29449	
	<b>&gt; 10 s</b>	0	

### 7.1.3 SER V2.0

<b>Date Test Performed</b>	4/25/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>SER Version</b>	2.0	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		60	80	100	120
<b>SIPp Traffic Report</b>	<b>Completed</b>	36000	48000	60000	72000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of NexSRS CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>10016 CDRs</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>Internal CDRs</b>	<b>Expected</b>	180000	240000	300000	360000
	<b>Actual</b>	180000	240000	300000	360000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~18000	~24000	~30000	~36000
	<b>Actual</b>	17170	24082	29718	35367
<b>SIP 408 CDRs</b>	<b>Expected</b>	~54000	~72000	~90000	~108000
	<b>Actual</b>	55019	72947	90581	108885
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>SIP 404</b>	<b>Expected</b>	~18000	~24000	~30000	~36000
	<b>Actual</b>	17170	24082	29718	35367
<b>SIP 408</b>	<b>Expected</b>	~54000	~72000	~90000	~108000
	<b>Actual</b>	55019	72947	90581	108885
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>SER Server</b>	<b>CPU idle</b>	74.6%	66.1%	58.4%	49.7%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	115M~180M	115M~193M	114M~206M	114M~
	<b>RX/TX</b>	352.5M/520.8M	446.6M/687.5M	590.2M/869.7M	670.6M/951.7M
<b>NexSRS</b>	<b>CPU idle</b>	91.2%	88.4%	85.0%	81.6%
	<b>xitami</b>	0.56%	0.74%	0.94%	1.13%
<b>SIPp</b>	<b>Client</b>	5.2%	6.8%	9.8%	11.9%
	<b>Server</b>	8.0%	11.0%	13.8%	16.1%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	36000	47999	60000	72000
	<b>&lt; 100 ms</b>	0	0	0	0
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	1	0	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	35976	47956	59889	71722
	<b>&lt; 100 ms</b>	24	42	111	278
	<b>&lt; 200 ms</b>	0	1	0	0
	<b>&lt; 500 ms</b>	0	1	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	8714	12156	15334	18197
	<b>&lt; 2 s</b>	6553	8317	10448	12700
	<b>&lt; 3 s</b>	2063	2572	3175	3943
	<b>&lt; 4 s</b>	5781	7588	9515	11617
	<b>&lt; 5 s</b>	3826	5210	6652	7599
	<b>&lt; 6 s</b>	4509	5950	7263	8732
	<b>&lt; 10 s</b>	4553	6199	7609	9191
<b>&gt; 10 s</b>	0	0	0	0	

<b>Date Test Performed</b>	4/25/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>SER Version</b>	2.0	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		140	160	180	200
<b>SIPp Traffic Report</b>	<b>Completed</b>	84000	96000	108000	120000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of NexSRS CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>10016 CDRs</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>Internal CDRs</b>	<b>Expected</b>	420000	480000	540000	600000
	<b>Actual</b>	420000	480000	540000	600000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~42000	~48000	~54000	~60000
	<b>Actual</b>	41426	47607	53382	59089
<b>SIP 408 CDRs</b>	<b>Expected</b>	~126000	~144000	~162000	~180000
	<b>Actual</b>	126456	145314	163754	181341
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>SIP 404</b>	<b>Expected</b>	~42000	~48000	~54000	~60000
	<b>Actual</b>	41426	47607	53382	59089
<b>SIP 408</b>	<b>Expected</b>	~126000	~144000	~162000	~180000
	<b>Actual</b>	126456	145314	163754	181341
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>SER Server</b>	<b>CPU idle</b>	40.6%	32.4%	24.1%	16.9%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	114M~230M	114M~243M	114M~258M	114M~
	<b>RX/TX</b>	843.7M/1184.6M	931.8M/1254.7M	977.0M/1484.6M	1174.3M/1684M
<b>NexSRS</b>	<b>CPU idle</b>	78.6%	75.5%	69.1%	67.9%
	<b>xitami</b>	1.34%	1.53%	1.93%	1.94%
<b>SIPp</b>	<b>Client</b>	14.5%	16.7%	19.8%	23.0%
	<b>Server</b>	19.3%	21.8%	24.9%	27.6%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	84000	95999	107998	120000
	<b>&lt; 100 ms</b>	0	0	0	0
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	1	2	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	83294	93841	94535	96605
	<b>&lt; 100 ms</b>	705	2156	13186	22507
	<b>&lt; 200 ms</b>	1	3	240	888
	<b>&lt; 500 ms</b>	0	0	38	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>	21492	24056	27153	30087
	<b>&lt; 2 s</b>	14458	16457	17048	19116
	<b>&lt; 3 s</b>	4725	6006	7864	8965
	<b>&lt; 4 s</b>	13779	15538	16366	17270
	<b>&lt; 5 s</b>	8870	9946	12423	14748
	<b>&lt; 6 s</b>	9963	11481	13070	13693
	<b>&lt; 10 s</b>	10655	12410	13926	15944
	<b>&gt; 10 s</b>	0	0	0	0

<b>Date Test Performed</b>	4/25/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used
<b>SER Version</b>	2.0	<b>Host Memory</b>	4 GB
<b>Test Rate</b>			220
<b>SIPp Traffic Report</b>	<b>Completed</b>		123323
	<b>Completed%</b>		93.43%
<b>Analysis of NexSRS CDRs</b>			
<b>200 CDRs</b>	<b>Expected</b>		132000
	<b>Actual</b>		127301
<b>10016 CDRs</b>	<b>Expected</b>		132000
	<b>Actual</b>		138004
<b>Internal CDRs</b>	<b>Expected</b>		660000
	<b>Actual</b>		647050
<b>SIP 404 CDRs</b>	<b>Expected</b>		~66000
	<b>Actual</b>		58530
<b>SIP 408 CDRs</b>	<b>Expected</b>		~198000
	<b>Actual</b>		207944
<b>All Other Error CDRs</b>	<b>Expected</b>		0
	<b>Actual</b>		0
<b>NexOSS Traffic Report</b>			
<b>16 Normal Call Clearing</b>	<b>Expected</b>		132000
	<b>Actual</b>		138004
<b>SIP 404</b>	<b>Expected</b>		~66000
	<b>Actual</b>		58530
<b>SIP 408</b>	<b>Expected</b>		~198000
	<b>Actual</b>		207944
<b>All Other</b>	<b>Expected</b>		0
	<b>Actual</b>		0
<b>System Utilization</b>			
<b>SER Server</b>	<b>CPU idle</b>		9.3%
	<b>IO wait</b>		0.1%
	<b>Memory</b>		114M~311M
	<b>RX/TX</b>		1573M/1884M
<b>NexSRS</b>	<b>CPU idle</b>		63.3%
	<b>xitami</b>		2.10%
<b>SIPp</b>	<b>Client</b>		29.3%
	<b>Server</b>		37.1%
<b>SIPp Average Response Time Repartition</b>			
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>		26708
	<b>&lt; 100 ms</b>		22578
	<b>&lt; 200 ms</b>		26292
	<b>&lt; 500 ms</b>		15328
	<b>&gt; 500 ms</b>		38502
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>		7572
	<b>&lt; 100 ms</b>		9527
	<b>&lt; 200 ms</b>		35763
	<b>&lt; 500 ms</b>		72721
	<b>&gt; 500 ms</b>		1674
<b>2nd 100 to Ringing</b>	<b>&lt; 1 s</b>		28117
	<b>&lt; 2 s</b>		4255
	<b>&lt; 3 s</b>		18879
	<b>&lt; 4 s</b>		5841
	<b>&lt; 5 s</b>		21164
	<b>&lt; 6 s</b>		5987
	<b>&lt; 10 s</b>		26825
	<b>&gt; 10 s</b>		70

## 7.1.4 SER V2.0 without debug option

<b>Date Test Performed</b>	5/29/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz-1 core used		
<b>SER Version</b>	2.0	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		60	80	100	120
<b>SIPp Traffic Report</b>	<b>Completed</b>	36000	48000	60000	72000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of NexSRS CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>10016 CDRs</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>Internal CDRs</b>	<b>Expected</b>	180000	240000	300000	360000
	<b>Actual</b>	180000	240000	300000	360000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~18000	~24000	~30000	~36000
	<b>Actual</b>	17542	23743	29579	35483
<b>SIP 408 CDRs</b>	<b>Expected</b>	~54000	~72000	~90000	~108000
	<b>Actual</b>	54282	73019	89757	108805
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	36000	48000	60000	72000
	<b>Actual</b>	36000	48000	60000	72000
<b>SIP 404</b>	<b>Expected</b>	~18000	~24000	~30000	~36000
	<b>Actual</b>	17542	23743	29579	35483
<b>SIP 408</b>	<b>Expected</b>	~54000	~72000	~90000	~108000
	<b>Actual</b>	54282	73019	89757	108805
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>SER Server</b>	<b>CPU idle</b>	74.6%	66.8%	58.8%	50.7%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	115M~177M	114M~190M	114M~202M	114M~214M
	<b>RX/TX</b>	347.6M/506.1M	474.6M/683.9M	554.6M/820.6M	687.4M/953.5M
<b>NexSRS</b>	<b>CPU idle</b>	91.1%	87.7%	85.0%	81.0%
	<b>xitami</b>	0.56%	0.75%	0.95%	1.14%
<b>SIPp</b>	<b>Client</b>	6.0%	8.1%	11.2%	14.6%
	<b>Server</b>	8.7%	10.7%	13.9%	16.8%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	36000	48000	60000	72000
	<b>&lt; 100 ms</b>	0	0	0	0
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	35981	47945	59884	71687
	<b>&lt; 100 ms</b>	19	55	116	313
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt; 0.1 s</b>	9015	12051	15456	18245
	<b>&lt; 1.1 s</b>	0	0	0	0
	<b>&lt; 2.1 s</b>	8502	10943	13864	16612
	<b>&lt; 3.1 s</b>	0	0	0	0
	<b>&lt; 4.1 s</b>	9668	12930	16140	19212
	<b>&lt; 5.1 s</b>	0	0	0	0
	<b>&lt; 6.1 s</b>	8814	12066	14534	17914
	<b>&gt; 10 s</b>	0	0	0	0

<b>Date Test Performed</b>	5/29/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp		
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used		
<b>SER Version</b>	2.0	<b>Host Memory</b>	4 GB		
<b>Test Rate</b>		140	160	180	200
<b>SIPp Traffic Report</b>	<b>Completed</b>	84000	96000	108000	120000
	<b>Completed%</b>	100%	100%	100%	100%
<b>Analysis of NexSRS CDRs</b>					
<b>200 CDRs</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>10016 CDRs</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>Internal CDRs</b>	<b>Expected</b>	420000	480000	540000	600000
	<b>Actual</b>	420000	480000	540000	600000
<b>SIP 404 CDRs</b>	<b>Expected</b>	~42000	~48000	~54000	~60000
	<b>Actual</b>	41596	47490	53532	59435
<b>SIP 408 CDRs</b>	<b>Expected</b>	~126000	~144000	~162000	~180000
	<b>Actual</b>	126404	145147	164303	182014
<b>All Other Error CDRs</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>NexOSS Traffic Report</b>					
<b>16 Normal Call Clearing</b>	<b>Expected</b>	84000	96000	108000	120000
	<b>Actual</b>	84000	96000	108000	120000
<b>SIP 404</b>	<b>Expected</b>	~42000	~48000	~54000	~60000
	<b>Actual</b>	41596	47490	53532	59435
<b>SIP 408</b>	<b>Expected</b>	~126000	~144000	~162000	~180000
	<b>Actual</b>	126404	145147	164303	182014
<b>All Other</b>	<b>Expected</b>	0	0	0	0
	<b>Actual</b>	0	0	0	0
<b>System Utilization</b>					
<b>SER Server</b>	<b>CPU idle</b>	42.0%	33.8%	25.4%	18.3%
	<b>IO wait</b>	0.1%	0.1%	0.1%	0.1%
	<b>Memory</b>	114M~228M	113M~240M	114M~255M	115M~269M
	<b>RX/TX</b>	847.8M/1085.0M	969.7M/1284.8M	975.3M/1483.9M	1178.7M/1684.4M
<b>NexSRS</b>	<b>CPU idle</b>	77.8%	74.7%	70.7%	67.3%
	<b>xitami</b>	1.34%	1.55%	1.74%	1.95%
<b>SIPp</b>	<b>Client</b>	17.3%	20.7%	24.1%	27.2%
	<b>Server</b>	19.2%	22.6%	25.9%	27.6%
<b>SIPp Average Response Time Repartition</b>					
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	84000	96000	108000	120000
	<b>&lt; 100 ms</b>	0	0	0	0
	<b>&lt; 200 ms</b>	0	0	0	0
	<b>&lt; 500 ms</b>	0	0	0	0
	<b>&gt; 500 ms</b>	0	0	0	0
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	83066	93229	99559	94926
	<b>&lt; 100 ms</b>	930	2759	8309	24007
	<b>&lt; 200 ms</b>	4	12	129	1066
	<b>&lt; 500 ms</b>	0	0	3	1
	<b>&gt; 500 ms</b>	0	0	0	0
<b>2nd 100 to Ringing</b>	<b>&lt;0.1 s</b>	21428	24533	26919	30118
	<b>&lt; 1.1 s</b>	0	0	0	0
	<b>&lt; 2.1 s</b>	19328	21947	24810	27751
	<b>&lt; 3.1 s</b>	0	0	77	0
	<b>&lt; 4.1 s</b>	22603	25284	28848	31907
	<b>&lt; 5.1 s</b>	0	0	190	0
	<b>&lt; 6.1 s</b>	20597	24157	26793	30033
	<b>&lt; 10 s</b>	0	0	247	2
<b>&gt; 10 s</b>	0	0	0	0	

<b>Date Test Performed</b>	5/29/07	<b>Host OS</b>	CentOS 4.4 / 2.6.9-42.ELsmp
<b>Test Performed by</b>	Di-Shi	<b>Host CPU</b>	Intel Xeon 5140 @ 2.33 GHz - 1 core used
<b>SER Version</b>	2.0	<b>Host Memory</b>	4 GB
<b>Test Rate</b>		220	
<b>SIPp Traffic Report</b>	<b>Completed</b>	117814	
	<b>Completed%</b>	89.25%	
<b>Analysis of NexSRS CDRs</b>			
<b>200 CDRs</b>	<b>Expected</b>	132000	
	<b>Actual</b>	125439	
<b>10016 CDRs</b>	<b>Expected</b>	132000	
	<b>Actual</b>	141721	
<b>Internal CDRs</b>	<b>Expected</b>	660000	
	<b>Actual</b>	640985	
<b>SIP 404 CDRs</b>	<b>Expected</b>	~66000	
	<b>Actual</b>	46367	
<b>SIP 408 CDRs</b>	<b>Expected</b>	~198000	
	<b>Actual</b>	219427	
<b>All Other Error CDRs</b>	<b>Expected</b>	0	
	<b>Actual</b>	436	
<b>NexOSS Traffic Report</b>			
<b>16 Normal Call Clearing</b>	<b>Expected</b>	132000	
	<b>Actual</b>	141721	
<b>SIP 404</b>	<b>Expected</b>	~66000	
	<b>Actual</b>	46367	
<b>SIP 408</b>	<b>Expected</b>	~198000	
	<b>Actual</b>	219427	
<b>All Other</b>	<b>Expected</b>	0	
	<b>Actual</b>	436	
<b>System Utilization</b>			
<b>SER Server</b>	<b>CPU idle</b>	10.9%	
	<b>IO wait</b>	0.1%	
	<b>Memory</b>	113M~313M	
	<b>RX/TX</b>	1579.2M/1985.4M	
<b>NexSRS</b>	<b>CPU idle</b>	62.4%	
	<b>xitami</b>	2.14%	
<b>SIPp</b>	<b>Client</b>	36.5%	
	<b>Server</b>	38.0%	
<b>SIPp Average Response Time Repartition</b>			
<b>Invite to 1st 100 message</b>	<b>&lt; 50 ms</b>	21828	
	<b>&lt; 100 ms</b>	23306	
	<b>&lt; 200 ms</b>	26645	
	<b>&lt; 500 ms</b>	16327	
	<b>&gt; 500 ms</b>	40079	
<b>1st 100 to 2nd 100 message</b>	<b>&lt; 50 ms</b>	5832	
	<b>&lt; 100 ms</b>	8877	
	<b>&lt; 200 ms</b>	35793	
	<b>&lt; 500 ms</b>	71075	
	<b>&gt; 500 ms</b>	1773	
<b>2nd 100 to Ringing</b>	<b>&lt; 0.1 s</b>	12470	
	<b>&lt; 1.1 s</b>	13586	
	<b>&lt; 2.1 s</b>	9418	
	<b>&lt; 3.1 s</b>	12212	
	<b>&lt; 4.1 s</b>	9066	
	<b>&lt; 5.1 s</b>	15533	
	<b>&lt; 6.1 s</b>	8156	
	<b>&lt; 10 s</b>	25191	
<b>&gt; 10 s</b>	244		

## Appendix

The initial performance tests compare OpenSER V1.1, V1.2 and SER 2.0 all compiled using default values. To serve requests from the SER community, the performance tests for SER 2.0 were re-run after compiling SER 2.0 with debug turned off. The overall performance improvement for SER 2.0 with debug turned off was approximately 3%. Details on the non-default SER 2.0 configuration are provided here.

### ***SER 2.0 without Debug Configuration***

```
debug=2          # debug level (cmd line: -ddddddddd)
fork=yes
log_stderror=no # (cmd line: -E)
sip_warning=no

/* Uncomment these lines to enter debugging mode
# fork=no
# log_stderror=yes
*/

check_via=no    # (cmd. line: -v)
dns=no          # (cmd. line: -r)
rev_dns=no     # (cmd. line: -R)
disable_tcp=yes
listen=udp:172.16.4.32:5060
# port=5060
children=64

# use_dst_blacklist=on
memdbg=100
server_signature=0

# ----- module loading -----
loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/textops.so"
loadmodule "/usr/local/lib/ser/modules/ctl.so"
loadmodule "/usr/local/lib/ser/modules/usrloc.so"
loadmodule "/usr/local/lib/ser/modules/registrar.so"
loadmodule "/usr/local/lib/ser/modules/avpops.so"
loadmodule "/usr/local/lib/ser/modules/fifo.so"
loadmodule "/usr/local/lib/ser/modules/options.so"
loadmodule "/usr/local/lib/ser/modules/xlog.so"
# Load OSP module
loadmodule "/usr/local/lib/ser/modules/osp.so"

# ----- setting module-specific parameters -----

#
# PEERING PARAMETERS:
# =====
# This section contains OSP parameters that users may need to configure for
# multi-lateral
# peering. (spl_uri must be configured.) Additional detail on OSP Module
# parameters and
# functions is provided in the "OSP Module for Secure, Multi-Lateral Peering"
# document
# located at:http://developer.berlios.de/docman/?group\_id=3799
#
# Configure Peering Servers:
# =====
```

```

# SER can be configured to query two peering servers for routing information
and peering
# authorization tokens using the sp1_uri and sp2_uri parameters. A
configuration for sp1_uri
# is required, configuring sp2_uri is optional. The peering server address
should be
# configured as a standard URL beginning with either http:// or https://
followed by the
# domain name of the OSP server or the IP address enclosed in brackets. The
domain name
# or IP address should be followed by the peering server TCP port number and
uniform
# resource identifier. Below are example configurations.
#
# modparam("osp", "sp1_uri", "http://osptestserver.transnexus.com:1080/osp")
# modparam("osp", "sp2_uri", "https://[1.2.3.4]:1443/osp")
modparam("osp", "sp1_uri", "http://172.16.4.75:1080/osp")
modparam("osp", "sp2_uri", "http://172.16.4.75:2080/osp")
modparam("osp", "sp3_uri", "http://172.16.4.75:3080/osp")
modparam("osp", "sp4_uri", "http://172.16.4.75:4080/osp")
modparam("osp", "sp5_uri", "http://172.16.4.75:5080/osp")
modparam("osp", "sp6_uri", "http://172.16.4.75:6080/osp")
modparam("osp", "sp7_uri", "http://172.16.4.75:8080/osp")
modparam("osp", "sp8_uri", "http://172.16.4.75:9080/osp")

modparam ("osp", "sp1_weight", 10)
modparam ("osp", "sp2_weight", 10)
modparam ("osp", "sp3_weight", 10)
modparam ("osp", "sp4_weight", 10)
modparam ("osp", "sp5_weight", 10)
modparam ("osp", "sp6_weight", 10)
modparam ("osp", "sp7_weight", 10)
modparam ("osp", "sp8_weight", 10)

#
# SER IP Address
# =====
# device_ip is a recommended parameter that explicitly defines the IP address
of SER in
# a peering request message (as SourceAlternate type=transport). The IP
address must
# be in brackets as shown in the example below.
#
# modparam("osp", "device_ip", "[1.1.1.1]")
modparam("osp", "device_ip", "[172.16.4.32]")

#
# Peering Token Validation
# =====
# When SER receives a SIP INVITE with a peering token, the OSP Module will
validate the token to
# determine whether or not the call has been authorized by a peering server.
Peering tokens may,
# or may not, be digitally signed. This parameter defines if SER will validate
signed or unsigned
# tokens or both. The values for "token format" are defined below. The default
value is 2.
#
# 0 - Validate only signed tokens. Calls with valid signed tokens are allowed.
# 1 - Validate only unsigned tokens. Calls with valid unsigned tokens are
allowed.
# 2 - Validate both signed and unsigned tokens are allowed. Calls with valid
tokens are allowed.
#

```

```

# modparam("osp", "token_format", 2)

#
# Crypto files from Peering Server Enrollment
# =====
# These parameters identify crypto files used for validating peering
authorization tokens
# and establishing a secure channel between SER and a peering server using SSL.
The files are
# generated using the 'Enroll' utility from the OSP toolkit. By default, the
proxy will look
# for pkey.pem, localcert.pem, and cacert_0.pem in the default configuration
directory.
# The default config directory is set at compile time using CFG_DIR and
defaults to
# /usr/local/etc/ser/. The files may be copied to the expected file location
or the
# parameters below may be changed.
#
# If the default CFG_DIR value was used at compile time, the files will be
loaded from:
# modparam("osp", "private_key", "/usr/local/etc/ser/pkey.pem")
# modparam("osp", "local_certificate", "/usr/local/etc/ser/localcert.pem")
# modparam("osp", "ca_certificates", "/usr/local/etc/ser/cacert_0.pem")

#
# Use Remote-Party-ID for calling number
# =====
# This parameter is used to tell OSP module if the calling number should be
obtained from RPID header.
# The default value is 1.
#
# 0 - OSP module will use the calling number in From header.
# 1 - OSP module will use the calling number in RPID header if a RPID header
exists.
#
# modparam("osp", "use_rpid_for_calling_number", 1)

# listen on the "standard" fifo for backward compatibility
modparam("ctl", "fifo", "fifo:/tmp/ser_fifo")

# -- usrloc params --
modparam("usrloc", "db_mode", 0)

# -- rr params --
avpflags dialog_cookie;

# add value to ;lr param to make some broken UAs happy
modparam("rr", "enable_full_lr", 1)

# enable append_fromtag, request's from-tag is appended to record-route;
# that's useful for understanding whether subsequent requests (such as BYE)
come from
# caller (route's from-tag==BYE's from-tag) or callee (route's from-tag==BYE's
to-tag)
modparam("rr", "append_fromtag", 1)

# -- tm params --
# Timer which hits if no final reply for a request or ACK for a
# negative INVITE reply arrives (in seconds). For example - UA server is off-
line.
# In other words, if the proxy does not receive a response to an Invite before
this

```

```

# timer expires, the proxy will retry the call and send an Invite to the next
VoIP
# destination in the routing list.
modparam("tm", "fr_timer", 2000)

# Timer which hits if no final reply for an INVITE arrives after
# a provisional message was received (in seconds).
# For example - user is not picking up the phone
modparam("tm", "fr_inv_timer", 30000)

# ----- request routing logic -----

# main routing logic
route{
    log(3, "----ROUTE: Route IN\n");

    # initial sanity checks
    if (!mf_process_maxfwd_header("10")) {
        xlog("L_WARN", "----ROUTE: Too many hops, $rm from '%fu' to '%tu'\n");
        sl_send_reply("483", "Too Many Hops");
        break;
    };

    if (msg:len >= max_len) {
        xlog("L_WARN", "----ROUTE: Message too big, $rm from '%fu' to '%tu'\n");
        sl_send_reply("513", "Message too big");
        break;
    };

    # we record-route all messages -- to make sure that
    # subsequent messages will go through our proxy; that's
    # particularly good if upstream and downstream entities
    # use different transport protocol
    if (method!="INVITE") {
        record_route();
    }

    # loose-route processing
    if(loose_route()) {
        if (method=="INVITE") {
            log(2, "----ROUTE: Relay re-INVITE\n");
            # send it out now; use stateful forwarding as it works reliably
even for UDP2TCP
            if (!t_relay()) {
                sl_reply_error();
            }
            break;
        } else if (method=="ACK") {
            log(2, "----ROUTE: Relay ACK\n");
            # send it out now; use stateful forwarding as it works reliably
even for UDP2TCP
            if (!t_relay()) {
                sl_reply_error();
            }
            break;
        }
    }
}

    if (method=="REGISTER") {
        log(2, "----ROUTE: Processing REGISTER\n");

        # Stop retransmission
        sl_send_reply("100", "Trying");
    }
}

```

```

    if (uri==myself) {
        log(2, "----ROUTE: Registered\n");
        save("location");
    } else {
        log(2, "----ROUTE: Register from outside domain rejected\n");
        sl_send_reply("488", "Unknown domain");
    }
} else if (method=="INVITE") {
    log(2, "----ROUTE: Processing INVITE\n");

    # Stop retransmission
    sl_send_reply("100", "Trying");

    if (t_lookup_request()) {
        log(2, "----ROUTE: Duplicated INVITE\n");
        break;
    }

    # Authentication
    log(3, "OSP authorization validation logic\n");

    # This function looks for OSP peering token in the message. It will
fail
    # if the token is not present
    if (checkospheader()) {
        log(3, "With OSP token, validate it\n");

        # The function validates OSP tokens. It will fail
        # if the token is not valid or has expired
        if (validateospheader()) {
            # Authorization is valid. The proxy can now use its own
database of
            # registered users for routing information.
            # The proxy could also issue another OSP peering authorization
and
            # routing request by calling route(1) function.
            log(3, "OSP authorization valid\n");

            # Remove the OSP peering token from the received message
            # Otherwise it will be forwarded on to the next hop
            remove_hf("P-OSP-Auth-Token");
        } else {
            log(3, "OSP authorization invalid\n");
            sl_send_reply("401", "Unauthorized");
            break;
        }
    };
} else {
    log(3, "Without OSP token, apply different authentication
strategy\n");
    log(3, "Go ahead, everyone is welcomed\n");

    # # Implement authentication strategy here or simply add the
tokens
    # # statements below to block all invites without OSP peering
    # sl_send_reply("401", "Unauthorized");
    # break;
}

log(2, "----ROUTE: Authentication passed\n");

# Routing
if (lookup("location")) {
    log(2, "----ROUTE: Registered user, forward the message\n");

```

```

        append_hf("P-hint: usrloc\r\n");
        record_route();
        t_relay();
    } else {
        log(2, "----ROUTE: Unregistered user, use OSP to get routing\n");
        route(2);
    }
} else if (method=="ACK") {
    log(2, "----ROUTE: Processing ACK\n");
    log(2, "----ROUTE: Not to relay ACK\n");
} else if (method=="BYE") {
    log(2, "----ROUTE: Processing BYE\n");

    if (t_lookup_request()) {
        log(2, "----ROUTE: Duplicated BYE\n");
        break;
    }

    # NOTE - don't t_relay before reporting usage
    if (!reportospusage()) {
        xlog("L_WARN", "----ROUTE: BYE without OSP information, from '%fu'
to '%tu'\n");
    }

    t_relay();
} else if (method=="CANCEL") {
    log(2, "----ROUTE: Processing CANCEL\n");
    t_relay();
} else if ((method=="OPTIONS") && (uri==myself)) {
    log(2, "----ROUTE: Processing OPTIONS\n");
    options_reply();
} else if (method=="PRACK") {
    log(2, "----ROUTE: Processing PRACK\n");
    t_relay();
} else if (method=="INFO") {
    log(2, "----ROUTE: Processing INFO\n");
    t_relay();
} else if (method=="UPDATE") {
    log(2, "----ROUTE: Processing UPDATE\n");
    t_relay();
} else {
    xlog("L_WARN", "----ROUTE: Unsupported message, $rm from '%fu' to
'%tu'\n");
    sl_send_reply("500", "Unsupported message");
}

log(3, "----ROUTE: Route OUT\n");
}

# OSP Authorization and Routing
route[2] {
    log(3, "OSP authorization and routing logic\n");

    # Is request to a phone number?
    # A phone number consists of digits (0 through 9)
    # and can begin with +
    # if (uri=~"sip:[+,0-9][0-9]*@") {
        # Requesting OSP peering routing and authorization
        # The request may fail if:
        # o OSP peering servers are not available
        # o Authentication failed
        # o There is no route to destination or the route is blocked
        log(3, "Requesting OSP authorization and routing\n");
    }
}

```

```

    if (requestosprouting()) {
        log(3, "Response received\n");

        setavpflag("fr._osp_orig_cookie_", dialog_cookie);
        setavpflag("fr._osp_term_cookie_", dialog_cookie);
        record_route();

        # Now we have 3 options.
        # o route(3) - sends a redirect to all available routes
        # o route(4) - fork off to all available routes
        # o route(5) in conjunction with failure_route(1) - sequentially
tries all routes

        # route(3);
        # route(4);
        route(5);

    } else {
        log(3, "OSP Authorization failed\n");
        sl_send_reply("503", "Service not available - No OSP routes");
    }
# } else {
#     log(3, "Wrong phone number\n");
#     sl_send_reply("401", "Not a phone number");
# }
}

route[3] {
    log(3, "Prepare all routes and redirect\n");

    if (prepareallosproutes()) {
        sl_send_reply("300", "Redirect");
    } else {
        log(3, "Failed to prepare all routes\n");
        sl_send_reply("500", "Internal Server Error");
    }
}

route[4] {
    log(3, "Prepare all routes and fork-off\n");

    if (prepareallosproutes()) {
        t_relay();
    } else {
        log(3, "Failed to prepare all routes\n");
        sl_send_reply("500", "Internal Server Error");
    }
}

route[5] {
    log(3, "Try the 1st route\n");

    if (prepareospfirstroute()) {
        t_on_branch("1");

        t_on_failure("1");

        t_relay();
    } else {
        log(3, "Could not use the 1st route\n");

```

```

        sl_send_reply("500", "Internal Server Error");
    }
}

failure_route[1] {
    log(3, "Try the next route\n");

    if (t_check_status("487")) {
        log(3, "Call canceled (status 487)\n");
        break;
    }

    if (t_check_status("486")) {
        log(3, "User busy (status 486)\n");
        break;
    }

    # tm's t_local_replied has not been implemented in SER yet.
    # if (t_check_status("408")) {
    #     if (!t_local_replied("last")) {
    #         log(3, "User unavailable (status 408)\n");
    #         break;
    #     }
    # }

    if (prepareospnextroute()) {
        t_on_branch("1");

        t_on_failure("1");

        t_relay();
    } else {
        log(3, "No more routes\n");
        t_reply("503", "Service not available - No more OSP routes");
    }
}

branch_route[1] {
    log(3, "Prepare route specific OSP information\n");
    appendospheaders();
}

```