Dialogic®

Application Note

Load Balancing on Dialogic® HMP Software using OpenSER-Based SIP Call Balancer

Executive Summary

This application note explains how to balance SIP calls using OpenSER – the Open Source SIP Server. OpenSER provides many features, but this application note focuses on use of OpenSER as a SIP call load-balancer. The ability to balance SIP calls is required for high-density systems and High Availability (HA) systems. This application note does not discuss how to achieve high-density and HA systems.

Table of Contents

Introduction

When using multiple IP-based media servers in an inbound or outbound call center environment, resources can be optimized by load balancing across multiple media server platforms. Load balancing allows multiple VoIP devices to operate in parallel, and provides fault tolerance in that if one server goes down, the load balancer can redistribute traffic over the remaining servers.

Environment

The hardware and software used in the test environment for the method described herein consisted of various software packages running on four servers as shown in Figure 1:

- **SIP call balancer** — Utilizes OpenSER on Linux
- **Bulk call generator** — Utilizes Dialogic® Host Media Processing Software Release 3.0 for Windows®
- **Inbound SIP call receiver** — Utilizes Dialogic HMP Software 3.0
- **Inbound SIP call receiver** — Utilizes Dialogic HMP Software Release 3.1LIN

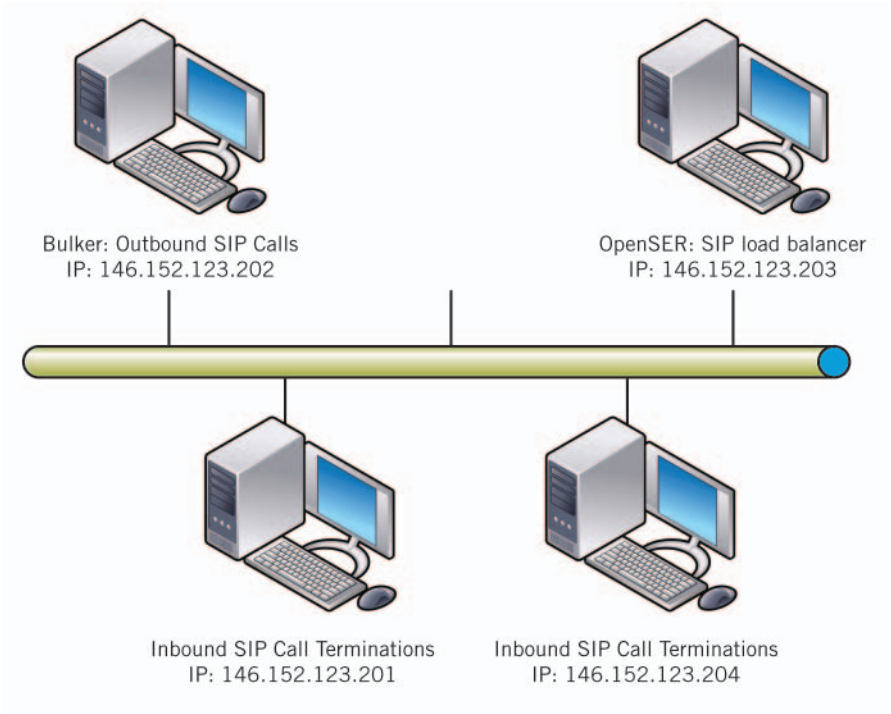

Figure 1. Network Topology

Call Flow

The basic SIP call flow is discussed here. The bulk call generator initiates calls using gc_basic_call demo. The INVITE is sent to OpenSER, which picks one of the two Inbound SIP Call Termination servers. OpenSER then forwards the INVITE to one of those servers. Figure 2 shows some sample call flows that were made by capturing call traffic on OpenSER using Tethereal and analyzing the captured traffic using Wireshark.

Note: Tethereal is a console-based tool, which unlike Wireshark, does not require a GUI environment. Thus, Tethereal is used to capture the packets into a file and then the traffic is analyzed using Wireshark (formerly known as Ethereal).

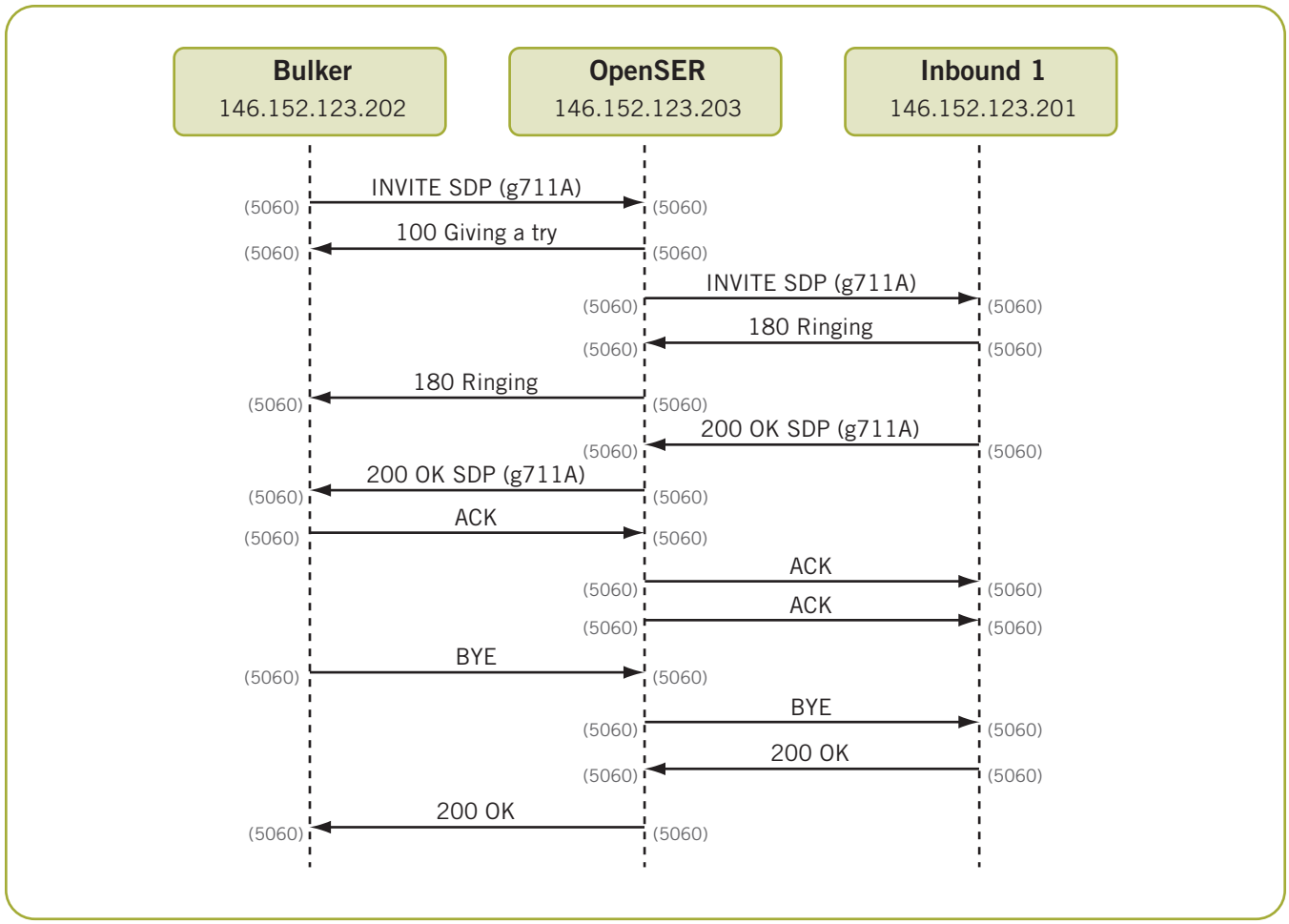

Figure 2. Sample Call Flows using Tethereal and Wireshark

OpenSER and Dispatcher Module Usage

OpenSER

OpenSER has built-in functionality. If features need to be added to OpenSER, they can be added via modules, which are actually shared libraries. For this test, the Dispatcher module was used (see the *References* section for downloading it). Modules are loaded via OpenSER configuration, and the OpenSER configuration file that was used is available for downloading (see the *For More Information* section). Lines of interest from that configuration include:

```
1.    loadmodule "dispatcher.so"
2.    modparam("dispatcher", "list_file",
      "/usr/local/etc/openser/dispatcher.list")
3.    modparam("dispatcher", "force_dst", 1)
4.    if (loose_route()) {
5.          t_relay();
6.    };
7.    ds_select_dst("1", "0");
8.    record_route();
9.    t_relay();
```

Line 1: Loads the Dispatcher module for use by OpenSER

Line 2: Location of configuration file used by Dispatcher module

Line 3: Forces overwriting of the destination address when that is already set

Lines 4 and 5: Messages within a dialog should take the path determined by record-routing

Line 7: Selects a destination address using Dispatcher module's "hash over called" method

Line 8: Packets that reach this line of the configuration file will get OpenSER added to the via list so that subsequent messages in the dialog go through OpenSER

Line 9: Uses stateful forwarding of messages

The lines pertaining to the Dispatcher module are described in the next section, *Dispatcher Module*.

Dispatcher Module

The Dispatcher module implements a dispatcher for destination addresses. It computes hashes over parts of the request and selects an address from a destination set. The selected address is used then as outbound proxy. The module can be used as a stateless load balancer, having no guarantee of fair distribution. [Mierla]

The Dispatcher module is loaded during the OpenSER startup based on configuration, which is similar to that previously discussed in the *OpenSER* section.

The Dispatcher module uses a simple configuration file. The tested configuration file contents are as follows:

```
# HMP Media Servers
1 sip:146.152.123.201:5060
1 sip:146.152.123.204:5060
```

The "1" is a set ID. One or more hosts are in a set. This example has two hosts. Line 7 of the OpenSER configuration file indicates that set "1" — the first parameter of ds_select_dst() — is used for selecting destination hosts.

The Dispatcher module can use the following methods to determine the destination IP address selection:

- hash over callid
- hash over from uri
- hash over to uri
- hash over request-uri
- round-robin (next destination)

For the particular setup in this application note, it was decided that the "hash over callid" was the most suitable method. The round-robin method requires more configuration, since OpenSER does not track call state. The uri-based hash methods may be suitable for some network topologies where the source and destination addresses are random. It is likely that "to uri" and "request-uri" hash methods will not be suitable, however, since those hosts are not random — that is, the service provider has a finite set of those hosts and they do not change often. Callids are globally unique identifiers; thus, "hash over callid" is a good candidate for selection of destination IP address for distributing calls. Using "hash over callid" over a large amount of calls should converge on even distribution, and the test results, discussed in the *Call Distribution Statistics* section, appear to show that being the case.

Line 7 of the OpenSER configuration file indicates that destination selection is based on "hash over callid" by using a "0" parameter.

Call Distribution Statistics

The tables below show call statistics from a test run using the setup explained in the *OpenSER and Dispatcher Module Usage* section. The Dispatcher module in the tested setup uses SIP callid as input to a hash function to select the destination SIP call termination server. The HMP PC2 was running gc_basic_call_demo with two inbound SIP channels and HMP PC3 was running with five inbound SIP channels.

Call Count			
	Outbound Calls	Inbound Calls	
Channel	HMP PC1	HMP PC2	HMP PC3
1	7519	7237	3068
2	7457	7262	3067
3	7441	–	3064
4	7390	–	3054
5	–	–	3055
Total/server	29807	14499	15308
Total	29807	29807	

Calls by Percentage			
	Outbound Calls	Inbound Calls	
Channel	HMP PC1	HMP PC2	HMP PC3
1	25.23	49.91	20.04
2	25.02	50.09	20.04
3	24.96	–	20.02
4	24.79	–	19.95
5	–	–	19.96
Total/server	100.00	100.00	100.00

Configuration Files

Two configuration files discussed in the *OpenSER and Dispatcher Module Usage* section are available as downloads with this application note (see the *For More Information* section). They are called openser.cfg and dispatcher.list.

Considerations

The configuration tested uses more inbound channels than outbound channels. There are seven inbound channels and four outbound channels. As the number of outbound channels approaches the number of inbound channels, some challenges can be considered.

If the hash-based method is used in the Dispatcher module, it is likely that the number of calls being sent to a User Agent Server (UAS) will be greater than the number of channels available in that UAS. Therefore, some sort of call accounting would need to be done to make OpenSER's routing more intelligent. An example of a SIP dialog with call-overflow is shown in Figure 3.

One possibility is to have OpenSER serially fork calls to UASs. For this to occur, the first call can be sent to a hash-based UAS. When OpenSER receives the "486 Busy Here" from that UAS, it forks calls to other UASs.

Another possibility is to have OpenSER not use the Dispatcher module, but instead use a database. The database would store the call states of all calls and provide a destination address based on UAS availability.

These methods can increase the complexity of the OpenSER setup and are outside the scope of this document, although they are being considered for a future application note.

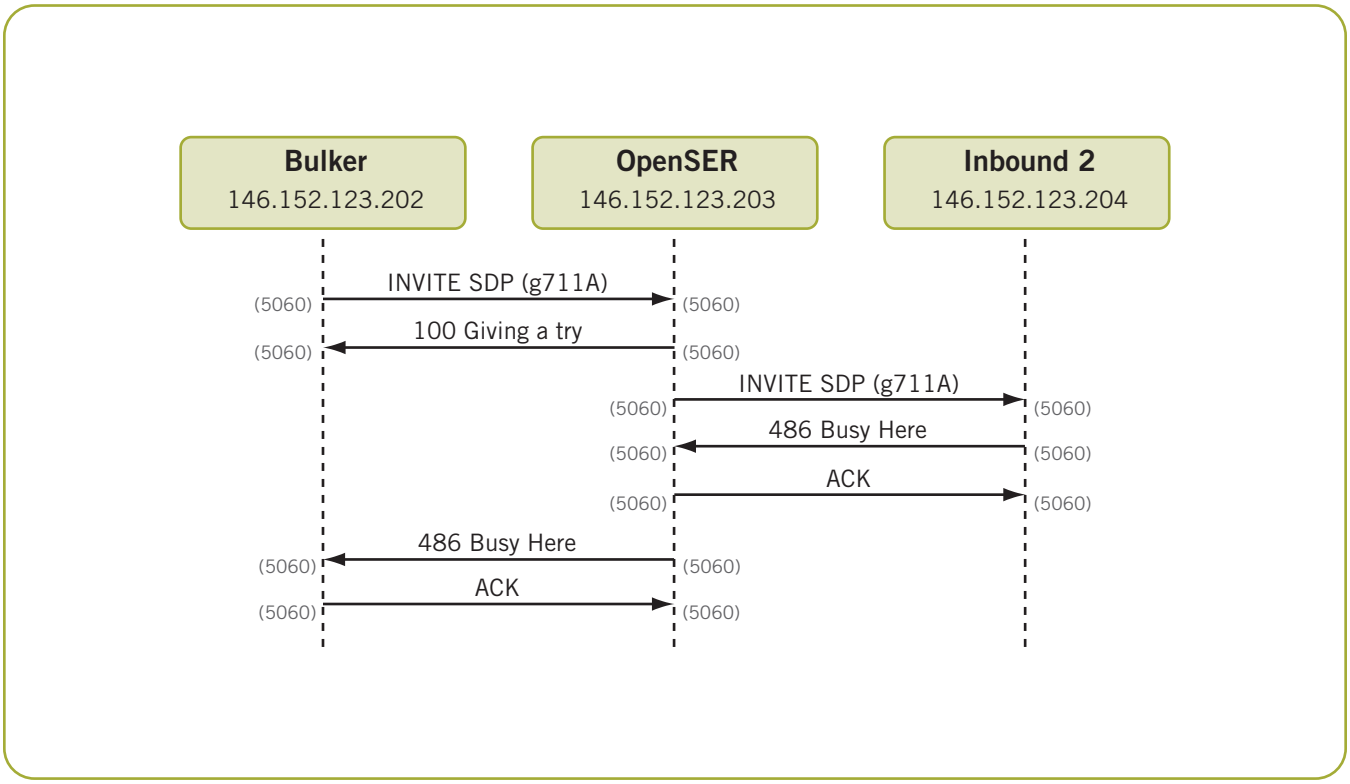

Figure 3. Example of a SIP Dialog with Call-Overflow

Downloads

Both OpenSER and Dispatch module are available on the web (see the *For More Information* section).

The version of OpenSER used is 1.2.1-notls:

```
# openser -V

version: openser 1.2.1-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.

svnrevision: unknown

@(#) $Id: main.c 1827 2007-03-12 15:22:53Z bogdan_iancu $

main.c compiled on 17:15:59 May 24 2007 with gcc 3.2.3
```

References

[Mierla] Daniel-Constantin Mierla, Editor, *DISPATCHER Module, 2005* Voice-System.RO is available at http://www.openser.org/docs/modules/1.1.x/dispatcher.html

For More Information

A Zip file containing the configuration files can be downloaded at http://www.dialogic.com/goto/?10854

OpenSER is available at http://openser.org/pub/openser/

Dispatcher module is available at http://www.openser.org/docs/modules/1.1.x/dispatcher.html

Dialogic® Host Media Processing Software — http://www.dialogic.com/products/ip_enabled/hmp_software.htm

Dialogic

To learn more about Dialogic® products, go to **www.dialogic.com**.

Dialogic Corporation
9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9

02/08 10833-01