1. Load Balancing in OpenSIPS

The "load-balancing" module comes to provide traffic routing based on load. Shortly, when **OpenSIPS** routes calls to a set of destinations, it is able to keep the load status (as number of ongoing calls) of each destination and to choose to route to the less loaded destination (at that moment). **OpenSIPS** is aware of the capacity of each destination - it is preconfigured with the maximum load accepted by the destinations. To be more precise, when routing, **OpenSIPS** will consider the less loaded destination not the destination with the smallest number of ongoing calls, but the destination with the largest available slot.

Also, the "load-balancing" (LB) module is able to receive feedback from the destinations (if they are capable of). This mechanism is used for notifying **OpenSIPS** when the maximum capacity of a destination changed (like a GW with more or less E1 cards).

The "load-balancing" functionality comes to enhance the "dispatcher" one. The difference comes in having or not load information about the destinations where you are routing to:

- **Dispatcher has no load information** - it just blindly forwards calls to the destinations based on a probabilistic dispersion logic. It gets no feedback about the load of the destination (like how many calls that were sent actually were established or how many are still going).
- **Load-balancer is load driven** - LB routing logic is based primary on the load information. The LB module is using the DIALOG module in order to keep trace of the load (ongoing calls).

A Workshop on this topic was held at Amoocon (former AsteriskTag) in Rostock, 4-5 May 2009

---

2. Load Balancing - how it works

When looking at the LB implementation in **OpenSIPS**, we have 3 aspects:

2.1 Destination set

A destination is defined by its address (a SIP URI) and its description as capacity.

Form the LB module perspective, the **destinations are not homogeneous** - they are not alike; and not only from capacity point of view, but also from what kind of **services/resources** they offer. For example, you may have a set of Yate/Asterisk boxes for media-related services -some of them are doing transcoding, other voicemail or conference, other simple announcement , other PSTN termination. But you may have mixed boxes - one box may do PSTN and voicemail in the same time. So each destination from the set may offer a different set of services/resources.

So, for each destination, the LB module defines the offered resources, and for each resource, it defines the **capacity / maximum load** as number of

concurrent calls the destination can handle for that resource.

Example:

```
4 destinations/boxes in the LB set
  1) offers 30 channels for transcoding and 32 for PSTN
  2) offers 100 voicemail channels and 10 for transcoding
  3) offers 50 voicemail channels and 300 for conference
  4) offers 10 voicemail, 10 conference, 10 transcoding and 32 PSTN

This translated into the following setup:

+----+----------+------------------------+---------------------------------+
| id | group_id | dst_uri                | resources                       |
+----+----------+------------------------+---------------------------------+
|  1 |        1 | sip:yate1.mycluset.net | transc=30; pstn=32              |
|  2 |        1 | sip:yate2.mycluset.net | vm=100; transc=10               |
|  3 |        1 | sip:yate3.mycluset.net | vm=50; conf=300                 |
|  4 |        1 | sip:yate4.mycluset.net | vm=10;conf=10;transc=10;pstn=32 |
+----+----------+------------------------+---------------------------------+
```

For runtime, the LB module provides MI commands for:

- reloading the definition of destination sets
- changing the capacity for a resource for a destination

2.2 Invoking Load-balancing

Using the LB functionality is very simple - you just have to pass to the LB module what kind of resources the call requires.

The resource detection is done in the **OpenSIPS** routing script, based on whatever information is appropriated. For example, looking at the RURI (dialed number) you can see if the call must go to PSTN or if it a voicemail or conference number; also, by looking at the codecs advertised in the SDP, you can figure out if transcoding is or not also required.

```
 if (!load_balance("1","transc;pstn")) {
   sl_send_reply("500","Service full");
   exit;
 }
```

The first parameter of the function identifies the LB set to be used (see the group_id column in the above DB snapshot). Second parameter is list of the required resource for the call.

The **load_balance()** will automatically create the dialog state for the call (in order to monitor it) and will also allocate the requested resources for it (from the selected box).
The function will set as destination URI ($du) the address of the selected destination/box.

The resources will be automatically released when the call terminates.

The LB module provides an MI function that allows the admin to inspect the current load over the destinations.

2.3 The LB logic

The logic used by the LB module to select the destination is:

1. gets the **destination set based on the group_id** (first parameter of the *load_balance()* function)
2. selects from the set only the **destinations that are able to provide the requested resources** (second parameter of the *load_balance()* function)
3. for the selected destinations, it **evaluated the current load** for each requested resource
4. the winning destination is the one with **the biggest value for the minimum available load** per resources.

```
Example:

4 destinations/boxes in the LB set
  1) offers 30 channels for transcoding and 32 for PSTN
  2) offers 100 voicemail channels and 10 for transcoding
  3) offers 50 voicemail channels and 300 for conference
  4) offers 10 voicemail, 10 conference, 10 transcoding and 32 PSTN

when calling load_balance("1","transc;pstn") ->

1) only boxes (1) and (4) will be selected at as they offer both transcoding and
pstn

2) evaluating the load  :
  (1) transcoding - 10 channels used; PSTN - 18 used
  (4) transcoding - 9 channels used; PSTN - 16 used

  evaluating available load (capacity-load) :
  (1) transcoding - 20 channels used; PSTN - 14 used
  (4) transcoding - 1 channels used; PSTN - 16 used

3) for each box, the minimum available load (through all resources)
  (1) 14 (PSTN)
  (2) 1 (transcoding)

4) final selected box in (1) as it has the the biggest (=14) available load for
the most loaded resource.
```

The selection algorithm tries to avoid the intensive usage of a resource per box.

---

3. Study Case: routing the media gateways

Here is the full configuration and script for performing LB between media peers.

3.1 Configuration

Let's consider the case previously described:

```
4 destinations/boxes in the LB set
  1) offers 30 channels for transcoding and 32 for PSTN
  2) offers 100 voicemail channels and 10 for transcoding
```

```
  3) offers 50 voicemail channels and 300 for conference
  4) offers 10 voicemail, 10 conference, 10 transcoding and 32 PSTN

This translated into the following setup:

+----+----------+------------------------+---------------------------------+
| id | group_id | dst_uri                | resources                       |
+----+----------+------------------------+---------------------------------+
|  1 |        1 | sip:yate1.mycluset.net | transc=30; pstn=32              |
|  2 |        1 | sip:yate2.mycluset.net | vm=100; transc=10               |
|  3 |        1 | sip:yate3.mycluset.net | vm=50; conf=300                 |
|  4 |        1 | sip:yate4.mycluset.net | vm=10;conf=10;transc=10;pstn=32 |
+----+----------+------------------------+---------------------------------+
```

3.2 OpenSIPS Scripting

```
debug=1
memlog=1

fork=yes
children=2
log_stderror=no
log_facility=LOG_LOCAL0

disable_tcp=yes
disable_dns_blacklist = yes

auto_aliases=no

check_via=no
dns=off
rev_dns=off

listen=udp:xxx.xxx.xxx.xxx:5060



loadmodule "modules/maxfwd/maxfwd.so"
loadmodule "modules/sl/sl.so"
loadmodule "modules/db_mysql/db_mysql.so"
loadmodule "modules/tm/tm.so"
loadmodule "modules/xlog/xlog.so"
loadmodule "modules/uri/uri.so"
loadmodule "modules/rr/rr.so"
loadmodule "modules/dialog/dialog.so"
loadmodule "modules/mi_fifo/mi_fifo.so"
loadmodule "modules/mi_xmlrpc/mi_xmlrpc.so"
loadmodule "modules/signaling/signaling.so"
loadmodule "modules/textops/textops.so"
loadmodule "modules/load_balancer/load_balancer.so"



modparam("mi_fifo", "fifo_name", "/tmp/opensips_fifo")

modparam("dialog", "dlg_flag", 13)
modparam("dialog", "db_mode", 1)
modparam("dialog", "db_url", "mysql://opensips:opensipsrw@localhost/opensips")

modparam("rr","enable_double_rr",1)
```

```
modparam("rr","append_fromtag",1)

modparam("load_balancer",
"db_url","mysql://opensips:opensipsrw@localhost/opensips")


route{
    if (!mf_process_maxfwd_header("3")) {
        sl_send_reply("483","looping");
        exit;
    }


    if (!has_totag()) {
        # initial request
        record_route();
    } else {
        # sequential request -> obey Route indication
        loose_route();
        t_relay();
        exit;
    }

    # handle cancel and re-transmissions
    if ( is_method("CANCEL") ) {
        if ( t_check_trans() )
            t_relay();
        exit;
    }


    # from now on we have only the initial requests
    if (!is_method("INVITE")) {
        send_reply("405","Method Not Allowed");
        exit;
    }

    # detect resources and do balancing
    if ($rU=~"^1") {
        # looks like a Conference call
        load_balance("1","conf");
    } else if ($rU=~"^2") {
        # looks like a VoiceMail call
        load_balance("1","vm");
    } else {
        # PSTN call, but the GWs supports only G711
        # for calls without G711, transcoding will be used on the GW
        if ( !search_body("G711") ) {
            load_balance("1","transc;pstn");
        } else {
            load_balance("1","pstn");
        }
    }

    # LB function returns negative if no suitable destination (for requested
resources) is found,
    # or if all destinations are full
    if ($retcode<0) {
       sl_send_reply("500","Service full");
       exit;
    }
```

```
    xlog("Selected destination is: $du\n");

    # send it out
    if (!t_relay()) {
        sl_reply_error();
    }
}
```