Hochschule Ostfalia
Fakultät Elektrotechnik
Labor für Kommunikationssysteme

Ostfalia
Hochschule für angewandte
Wissenschaften

AskoziaPBX-Performancetests

Written by

Mark Stephan

Head:	Prof. Dr.-Ing. D. Wermser
Supervisor:	M. Iedema

13.07.2010

Inhaltsverzeichnis

Listings

1 Introduction

This subproject of AskoziaPBX was developed for executing performance- and stress-tests to different Askozia-boxes. It was written by Mark Stephan (mark.stephan@askozia.com) during a job as a student assistant for the Askozia-founder Michael Iedema (michael@askozia.com) in Spring/Summer 2010.

It is very important to understand the differences between the three test-types (twoway-calls, conference calls with fixed number of calls and conference calls with fixed number of rooms) as well as the two test-operations (one test with all calls and rooms and slow-and-steadily building-up tests). They are explained detailed in chapter 3 and the first paragraphs of chapters 4, 5 and 6.

1.1 Problem

The AskoziaPBX software can be downloaded as firmware image for embedded systems and as live cd. The live cd can be run on every normal computer, so the underlying hardware may have very different performance. E.g., the same software can handle three, 30 or 300 parallel twoway-calls, depending on the computer performance. So we had to develop an algorithm to find out how tough the current Askozia box is.

1.2 Features

The current testsuite supports the following features:

- completely automatic testing of one AskoziaPBX
- automatic configuration of the AskoziaPBX with the needed settings
- three different types of test:

  twoway-calls The testsuite establishes a variable number of A-to-B (or end-to-end) calls between the AskoziaPBX and the testsystem. This test simulates normal telephone calls between two persons.

  conference-tests with fixed number of calls The testsuite calls a variable number of different conference-rooms with a fixed number of users in each room.

  conference-tests with fixed number of rooms The testsuite calls a fix number of conference-rooms with a variable number of users in each rooms.
- monitoring of the CPU-load of the AskoziaPBX caused by the testcalls
- download of the recorded CPU-load data
- interpretation and creation of graphs of the testresults

- two different test-operations:

  one test with maximum number of users The testsuite establishes one test with the specified number of calls and rooms.

  slow-and-steadily building-up tests The testsuite starts with two users (users A and B in twoway-tests), one conference-room (conference-tests with fixed number of calls) and one user in all desired conference-rooms (conference-tests with fixed number of rooms) and executes this test completely (including register, invitation and deregister). Then, after a short break, the next test with one more user (or room) is executed. This procedure is repeated until the specified number of users (twoway), calls (conference-test with fixed rooms) or conference-rooms (conference-test with fixed calls) is reached.

1.3 Usage

The script can be called in the command line, e.g.:

Listing 1: Aufruf des Scripts

```
./PERF_TEST <options>
perl PERF_TEST <options>
```

`--local-ip=<IP>`
: The IP-adress of the testcomputer that executes the testscript. <IP> stands for the address of the device connected to the AskoziaPBX.
Standard: undefined (**must** be defined)
Example: `--local-ip=192.168.0.2`

`--askozia-ip=<IP>`
: The IP-address of the Askozia box that should be tested.
Standard: `10.10.10.1`
Example: `--askozia-ip=192.168.0.1`

`--askozia-port=<number>`
: The number of the webport of the AskoziaPBX
Standard: `80`
Example: `--askozia-port=80`

`--askozia-confpage=<string>`
: Name of the php-page for down-/uploading the XML-configfile
Standard: `system_backup.php`
Example: `--askozia-confpage=system_backup.php`

`--askozia-realm=<string>`
: Name of the authentication-realm of the AskoziaPBX.
Standard: `Web Server Authentication`
Example: `--askozia-realm='Web Server Authentication'`

`--askozia-user=<string>`
: Name of the root-user of the AskoziaPBX (necessary for executing commands using the webserver).
Standard: `admin`
Example: `--askozia-user=admin`

`--askozia-pw=<string>`
: Password for the root-user of the AskoziaPBX.
Standard: `askozia`
Example: `--askozia-pw=askozia`

`--save-sipp-log=<string>`
: The output generated by the testprogram "sipp" can be saved in a file for debugging. The path to the file where the output should be saved can be specified in the overgiven string and may be absolute or relative to the subdirectory `./results/<testname>/`. If not specified, the output is ignored.
Standard: undefined (output ignored)
Example: `--save-sipp-log=../sipp.log` or `--save-sipp-log=/tmp/sipp.log`

`--debug`
: Activates debug messages. Have a look at the option `--save-debug`.
Standard: undefined (no debug output)
Example: `--debug`

`--save-debug=<string>`
: Because of the many debug messages, thus can be saved in a file. The string is the relative or absolute path to the file where the messages should be saved. The cwd for this file is `./results/<testname>/`, this means, if `debug.txt` is entered, the path to this file is `./results/<testname>/debug.txt`.
Standard: undefined (no saving)
Example: `--save-debug=../debug.txt` or `--save-debug=/tmp/debug.txt`

`--2way-calls=<number>`
: This is either the number of calls that should be established during one test with all users or the maximum number of twoway-calls that should be established during a slow-and-steadily building-up test, depending on the parameter `--2way-pause`. For more information, please have a look at chapter 4.
Standard: undefined (no twoway-tests)
Example: `--2way-calls=30`

`--2way-pause=<number>`
: This parameter defines the test-operation for twoway-tests: If not specified, there is

one test with all users (depending on the parameter `--2way-calls`. If defined, the test-operation is set to slow-and-steadily building-up tests. For more information, please have a look at chapter 3.
Standard: undefined (one test with all users)
Example: `--2way-pause=10`

`--conf-calls-room=<number>`
: This is either the number of users in each conference-room or the maximum number of users in each conference-room, depending on the test-operation. For a more understable explanation, please have a look at chapter 6.
Standard: undefined (no conference-tests with fixed number of rooms)
Example: `--conf-calls-room=10`

`--conf-rooms-room=<number>`
: This is the number of conference-rooms existing during the conference-test with fixed rooms. For a more understable explanation, please have a look at chapter 6.
Standard: `1`
Example: `--conf-rooms-room=1`

`--conf-pause-room=<number>`
: This parameter defines the seconds to wait between each single test. This pause helps the AskoziaPBX to calm down between the tests and avoids manipulating neighboured tests each other. Furthermore, it defines the test-operation: If not specified, there is only one single tests with all users and rooms; if specified, the test-operation is set to "slow-and-steadily building-up". For more information, please have a look at chapter 3.
Standard: undefined (one test with all users and rooms)
Example: `--conf-pause-room=10`

`--conf-calls-call=<number>`
: This is the number of users in all conference-rooms existing during the conference-test with fixed calls. For a more understable explanation, please have a look at chapter 5.
Standard: undefined (no conference-tests with fixed number of calls)
Example: `--conf-calls-call=3`

`--conf-rooms-call=<number>`
: This is either the number of conference-rooms existing during one test with all users and rooms or the maximum number of conference-rooms during a slow-and-steadily building-up test. For a more understable explanation, please have a look at chapter 5.
Standard: `1`
Example: `--conf-rooms-call=10`

`--conf-pause-call=<number>`
: This parameter defines the seconds to wait between each single test. This pause

helps the AskoziaPBX to calm down between the tests and avoids manipulating neighboured tests each other. Furthermore, it defines the test-operation: If not specified, there is only one single tests with all users and rooms; if specified, the test-operation is set to "slow-and-steadily building-up". For more information, please have a look at chapter 3
Standard: undefined (one test with all users and calls)
Example: `--conf-pause-call=10`

`--sipp-exe=<string>`
This parameter specifies the sipp-executable that should be used to execute the performance-tests. It is recommended to use the supplied version because it is self-compiled and does not crash if its parameter `-aa` is used contrary to the repositories-version. String is the path to the sipp-exe that should be used; it may be relative or absolute.
Standard: `./PERF_TESTS/sipp` or, if not existing, the result of `which sipp`
E.g. `--sipp-exe=../sipp` or `--sipp-exe=/tmp/sipp`

`--users-2way-file=<string>`
For executing twoway-tests with sipp, there has to be a so-called injection-file. This is a csv-file which is used by sipp for generating multiple calls automaticly. It is created by the script and contains all needed information (and not more) for sipp. It is possible to change the filepath and -name of this file by specifying this parameter.
Standard: `./results/<testname>/Users_twoway.csv`
E.g. `--users-2way-file=../Users_twoway.csv`
or `--users-2way-file=/tmp/Users_twoway.csv`

`--users-conf-room-file=<string>`
Please have a look at the parameter `--users-2way-file`. This parameter is the same only for conference tests with fixed number of rooms.
Standard: `./results/<testname>/Users_conf_room.csv`
E.g. `--users-conf-room-file=../Users_conf_room.csv`
or `--users-conf-room-file=/tmp/Users_conf_room.csv`

`--users-conf-call-file=<string>`
Please have a look at the parameter `--users-2way-file`. This parameter is the same only for conference tests with fixed number of calls.
Standard: `./results/<testname>/Users_conf_call.csv`
E.g. `--users-conf-call-file=../Users_conf_call.csv`
or `--users-conf-call-file=/tmp/Users_conf_call.csv`

`--reg-scen=<string>`
This parameter specifies the path to the Register-scenario used by sipp. The Register-scenario is needed for twoway-tests only. For more information, please have a look at chapter 4.

Standard: `./PERF_TEST_FILES/Register.xml`
E.g. `--reg-scen=../Register.xml` or `--reg-scen=/tmp/Register.xml`

`--dereg-scen=<string>`
This parameter specifies the path to the Deregister-scenario used by sipp. The Deregister-scenario is needed for twoway-tests only. For more information, please have a look at chapter 4.
Standard: `./PERF_TEST_FILES/Deregister.xml`
E.g. `--dereg-scen=../Deregister.xml` or `--dereg-scen=/tmp/Deregister.xml`

`--acc-scen=<string>`
This parameter specifies the path to the Accept-scenario used by sipp. The Accept-scenario is needed for twoway-tests only. For more information, please have a look at chapter 4.
Standard: `./PERF_TEST_FILES/Accept.xml`
E.g. `--acc-scen=../Accept.xml` or `--acc-scen=/tmp/Accept.xml`

`--inv-scen=<string>`
This parameter specifies the path to the Invite-scneario used by sipp. The Invite-scenario is needed for every test-type. For more information, please have a look at the description of the different testtypes (chapters 4, 5 and 6).
Standard: `./PERF_TEST_Files/Invite.xml`
E.g. `--inv-scen=../Invite.xml` or `--inv-scen=/tmp/Invite.xml`

`--sip-src-port=<number>`
Port of the local computer (testcomputer) to communicate with Askozia. It is used for User A in twoway-tests and for all users in conference tests. During the tests, there was a softphone running in background for communication in the office. Because of this, sipp was not able to reserve the usual sip-port 5060.
Standard: `5061`
E.g. `--sip-src-port=5061`

`--sip-dst-port=<number>`
Port of the local computer (testcomputer) to communicate with Askozia, but this time only for User B in twoway-tests. The first sipp-process (User A) blocks one port for communication with AskoziaPBX, so the seconds sipp-process (User B) needs another one to talk to Askozia. This is necessary for twoway-tests only.
Standard: `5062`
E.g. `--sip-dst-port=5062`

`--rtp-src-port=<number>`
Port of the local computer for establishing RTP-streams between the local test-computer and AskoziaPBX. This one is used by User A of twoway-tests and by all users of conference tests. Sipp was not able to use the standard-port because of a softphone running on the testcomputer in background.

Standard: `6020`
E.g. `--rtp-src-port=6020`

`--rtp-dst-port=<number>`
Port of the local computer for establishing RTP-streams between the local test-computer and AskoziaPBX. This one is used by User B of twoway-tests only, so it not needed for conference testing.
Standard: `6030`
E.g. `--rtp-dst-port=6030`

`--testname=<string>`
This parameter helps to keep your results-directory uncluddered. All files of the current script call (all tests, debug files etc.) are saved in the subdir `./results/<testname>/`. If undefined, the files will be saved in the results-directory directly, so it will be messy soon.
Standard: undefined (direct saving of results in subdir `./results`)
E.g. `--testname=2010-01-01_1030`
(saving of results in subdir `./results/2010-01-01_1030/`)

`--restore=<string>`
After the test, the AskoziaPBX is strongly reconfigured. There are possibly hundreds of testusers and some new conference-rooms. To avoid cleaning up by hand, this parameter helps to reconfigure the box after the test. There are three possible values:

none The AskoziaPBX is not reconfigured.

old-config The AskoziaPBX is restored with the config existing before the test.

factory-defaults The AskoziaPBX is set to factory-defaults.

Standard: `old-config`
E.g. `--restore=none` or `--restore=factory-defaults`

`--help`
Displays a short help for using the testscript and exits immediatly.
Standard: undefined (no help shown)
E.g. `--help`

For <options>, there are many possibilities. Some are optional (like the name says), but some them you have to specify. It is necessary to specify your own IP (`--local-ip`) and one of the parameters `--2way-calls`, `--conf-calls-call` or `--conf-calls-room` because these three parameters define which test should be executed. If none is specified, the script does not know what to test.

You have to be root to execute this script because `sipp` reserves port for its connection to Askozia.

1.4 Dependencies

This script was developed under Linux Mint 8 Helena (`http://www.linuxmint.com`). It should be possible to execute it under windows, but it was not tested. The script needs the following packages including their own dependencies:

- Perl v5.10.0 (`http://www.perl.org`)
- gnuplot 4.2 patchlevel 5 (`http://www.gnuplot.info`)

2 Configuration of the AskoziaPBX

This paragraph is about the configuration of the AskoziaPBX. First of all, the used configfile from Askozia is downloaded because the user should not have to reconfigure the whole box including all accounts and the dialplan after every test. The target is to deliver the Askozia box just like it was issued. So, there are three necessary steps which are described in the next chapters.

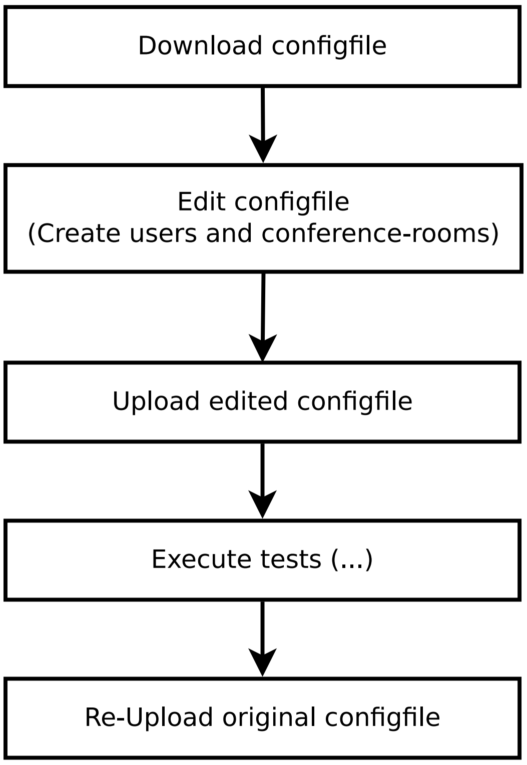

Abbildung 2.1: Process of editing the Askozia-configfile

2.1 Down-/Upload Configfile

For downloading the configfile, it is necessary to send a HTML POST request to Askozia. Or, to be more precise, it has to be sent to the qstat-page of the Askozia box. The useragent has to be authenticated as root and the content type must be "multipart/form-data". With this POST request, Askozia sends the output of the qstat page. For downloading the configfile, the parameter "Download" has to be set to any desired value – but it has to be set.

In the performance test script, the following perl code is used to send this POST request:

Listing 2: POST request for downloading configfile

```
use HTTP::Request::Common;
use LWP;
my $ua = new LWP::UserAgent;
$ua->credentials ("$ask_ip:$ask_port",
        $ask_realm, "$ask_user" => "$ask_pw");
my $res = $ua->request (POST
        "http://$ask_ip:$ask_port/$ask_conf_page",
        "Content-Type" => "multipart/form-data",
        Content => [ Download => "1"]);
```

After executing this request, the following dataflow has to be expected:

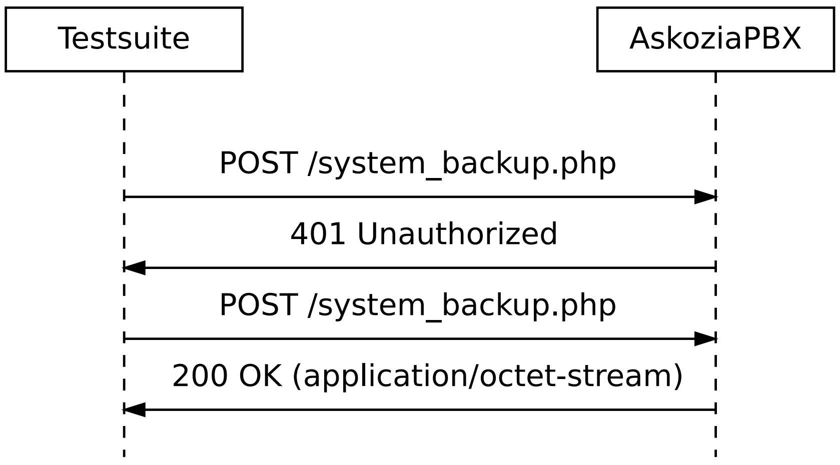

Abbildung 2.2: Dataflow of configfile download

The “200 OK” message sent by Askozia includes the configfile in xml format.

2.2 Users

2.3 Conference-Rooms

3 Test-Operations

3.1 One test with all calls and rooms

3.2 Slow-and-steadily building-up tests

4 Twoway-Calls

5 Conference-Calls with fixed number of calls

6 Conference-Calls with fixed number of rooms

7 Watchdog-Feature

A Appendix

A.1 Example Tests

A.2 XML-Scenarios

A.2.1 Register-Scenario

A.2.2 Deregister-Scenario

A.2.3 Invite-Scenario

A.2.4 Accept-Scenario