ClueCon 2018
End2End test automation with
VoIP Patrol

Julien Chavanton
Lead software engineer - voice routing
www.flowroute.com

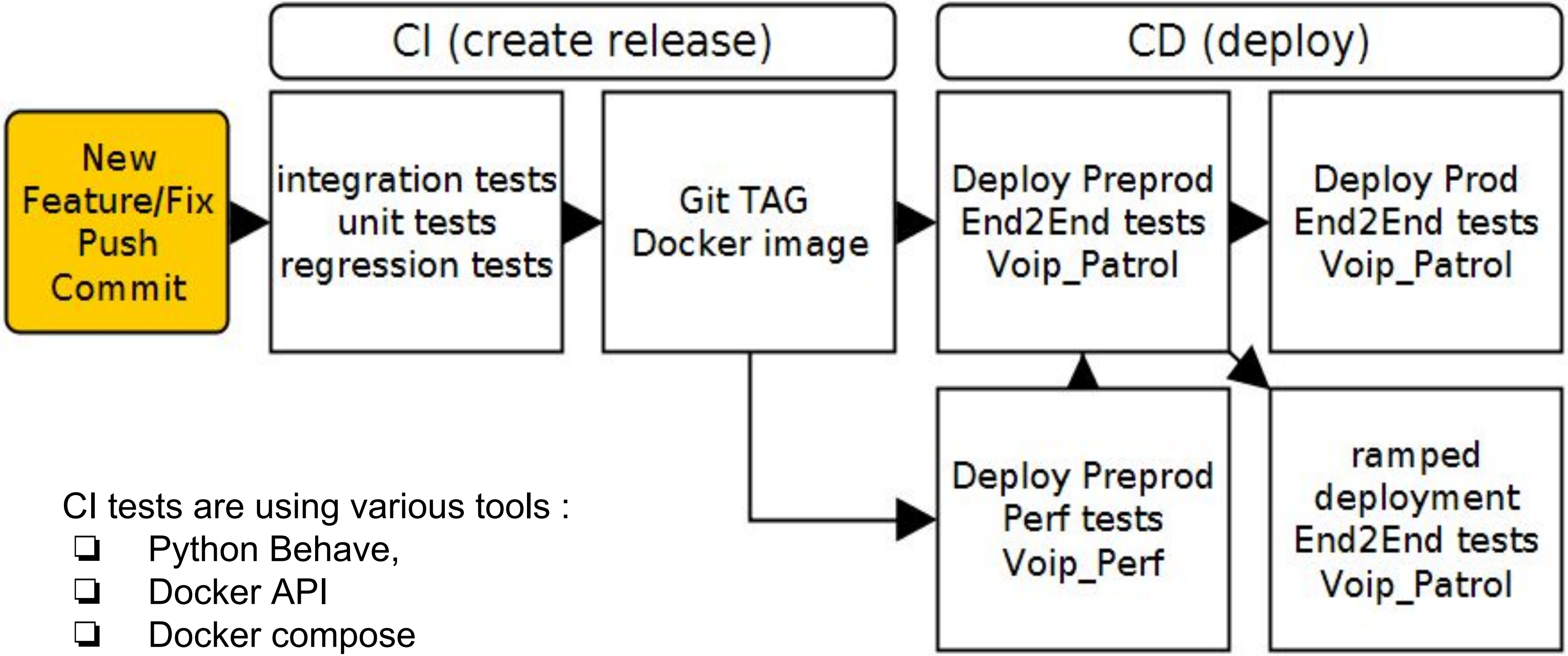
Presentation Summary

1 - Share some of the testing strategies we are using at Flowroute
2 - This presentation could serve as an initial documentation on how to use VoIP Patrol

I am always trying to find time to implement new features and since I do not have personal testers and I need to own everything I commit.

- ❑ CI/CD strategy overview on the voice routing platform at Flowroute
- ❑ Objectives of VoIP-Patrol
- ❑ VoIP-Patrol libraries architecture overview (PJSUA2 and other libraries)
- ❑ OO architecture overview: accounts, calls, etc.
- ❑ Few usage examples: making and receiving calls, etc.
- ❑ Using VoIP-Patrol with docker
- ❑ Example of integration with Python docker API and test templating
- ❑ Future improvement, MOS and End2end audio quality testing

CI/CD and risk mitigation efforts



CI tests are using various tools :

- Python Behave,
- Docker API
- Docker compose
- SIPP, Kamailio, etc.

https://github.com/jchavanton/voip_perf

Why another tool ?

Why do we need another tool if we can use Freeswitch or other specialized test tools like SIPP ?

- Using VoIP-Patrol, you can test interoperability with another compliant SIP stack.
- SIPP is simple and clever however it is more a mock, it is not trying to be SIP compliant and media handling is not really supported.
- PJSUA2 is well documented :

  http://www.pjsip.org/docs/book-latest/PJSUA2Doc.pdf *(233 pages book was published in September 2017)*

  http://www.pjsip.org/release/0.5.4/PJSIP-Dev-Guide.pdf

Main goal : automate all manual tests you could do with a softphone

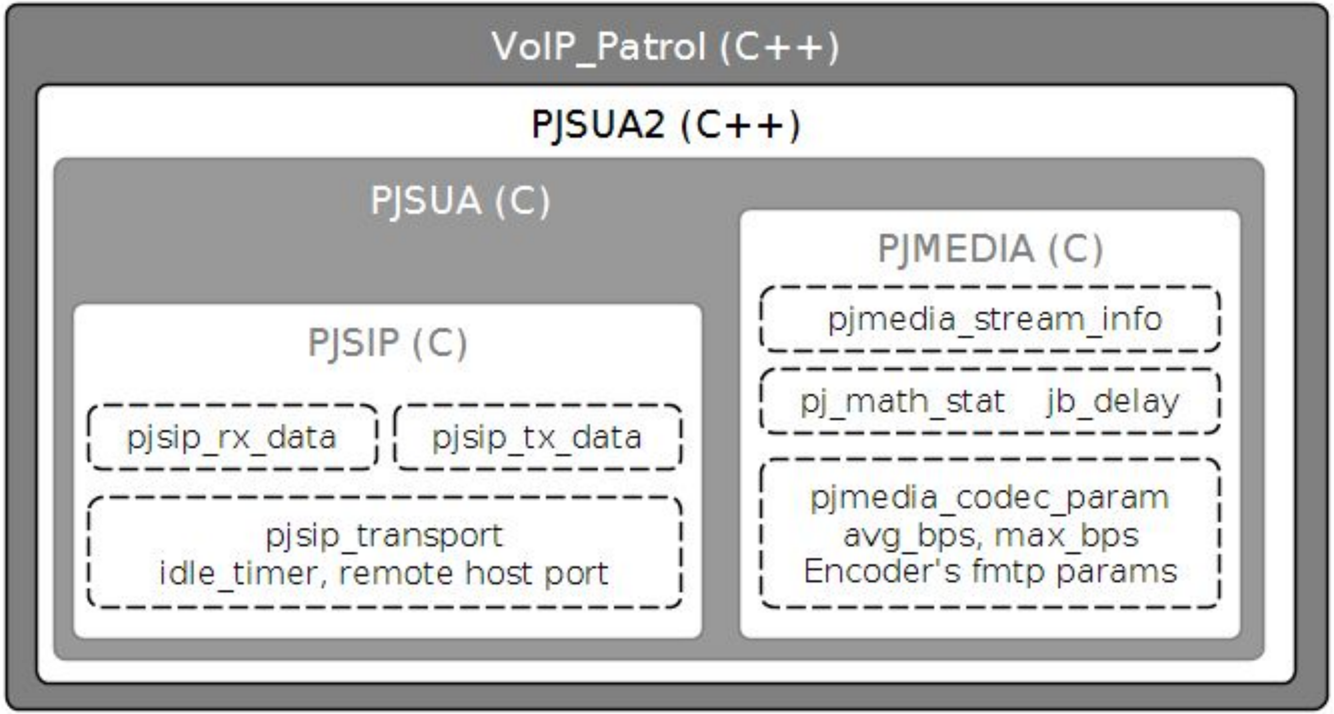
- Scenarios easy to configure in XML
  not sure where this idea came from … :) *
- Gathering various outputs relevant to test validation formatted in JSON
- Test interoperability and RFC compliance
- Easy to troubleshoot with verbose logging

*Disruptor (Network impairment test app) (initial idea by Dragos Oncea!)

flowroute

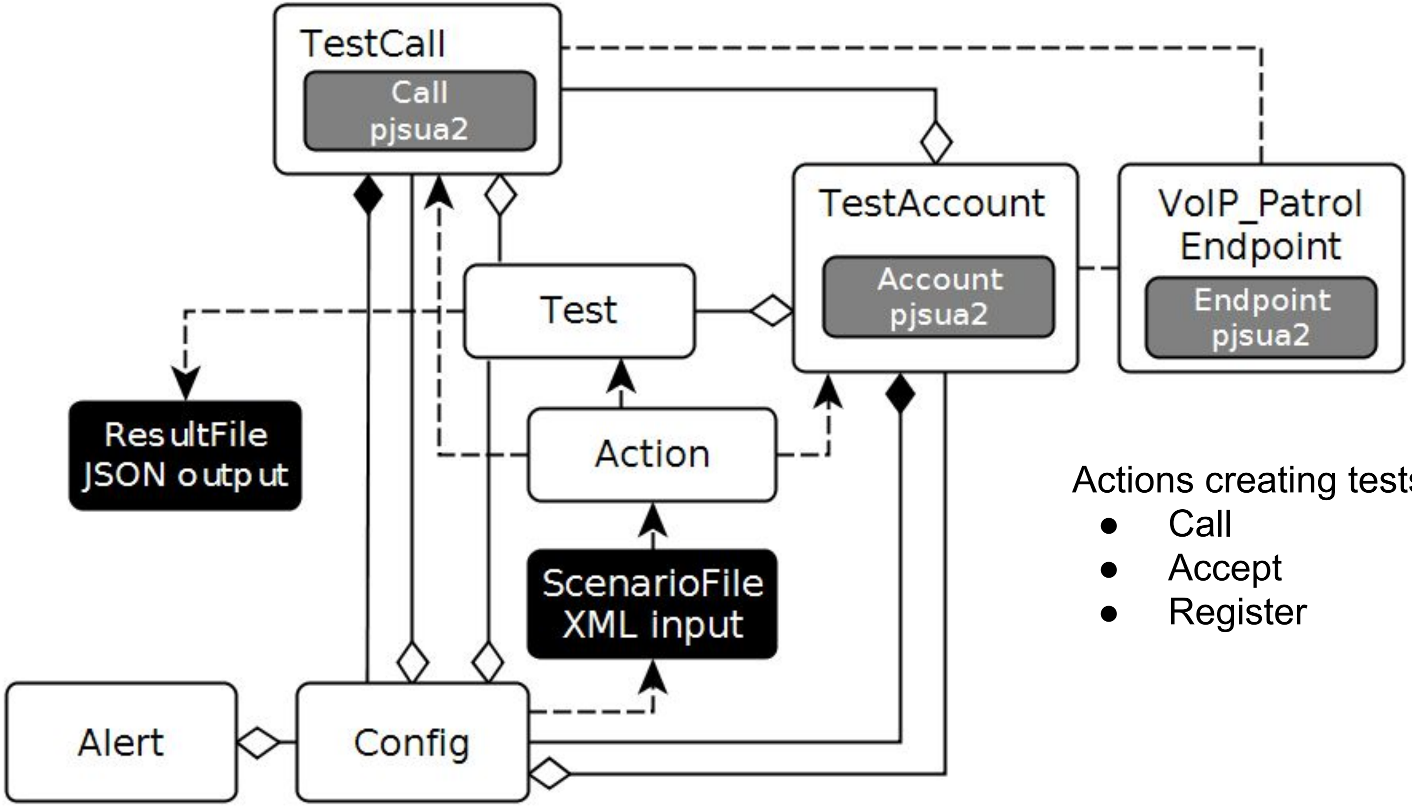
Architecture of libraries

Providing low level access to valuable information



Architecture classes and components

Accounts, Calls, transports, etc.



Actions creating tests :

- Call
- Accept
- Register

Example: making a test call

```xml
<config>
  <actions>
    <action type="call" label="us-east-va"
            transport="tls"
            expected_cause_code="200"
            caller="15147371787@noreply.com"
            callee="12012665228@target.com"
            max_duration="20" hangup="16"
            username="VP_ENV_USERNAME"
            password="VP_ENV_PASSWORD"
            realm="target.com"
            rtp_stats
    />
    <action type="wait" complete/>
  </actions>
</config>
```

XML scenario


```json
{
  "2": {
    "label": "us-east-va",
    "start": "17-07-2018 00:00:05",
    "end": "17-07-2018 00:00:24",
    "action": "call",
    "from": "15147371787",
    "to": "12012665228",
    "result": "PASS",
    "expected_cause_code": 200,
    "cause_code": 200,
    "reason": "Normal call clearing",
    "callid": "7iYDFukJr-9BOLOmWg.7fZyHZeZUAwao",
    "transport": "TLS",
    "peer_socket": "34.226.136.32:5061",
    "duration": 16,
    "expected_duration": 0,
    "max_duration": 20,
    "hangup_duration": 16,
    "rtp_stats": {
      "rtt": 0,
      "Tx": {
        "jitter_avg": 0,
        "jitter_max": 0,
        "pkt": 816,
        "kbytes": 127,
        "loss": 0,
        "discard": 0,
        "mos_lq": 4.5
      },
      "Rx": {
        "jitter_avg": 0,
        "jitter_max": 0,
        "pkt": 813,
        "kbytes": 127,
        "loss": 0,
        "discard": 0,
        "mos_lq": 4.5
      }
    }
  }
}
```


JSON output

flowroute

Example: TLS test server

```
./voip_patrol \
   --port 5060 \ # TLS port 5061 +1
   --conf "xml/tls_server.xml" \
   --tls-calist "tls/ca_list.pem" \
   --tls-privkey "tls/key.pem" \
   --tls-cert "tls/certificate.pem" \
   --tls-verify-server \
```

```
<config>
  <actions>
    <action type="accept"
            account="default"
            <!-- default is the
                 "catch all" account -->
            hangup="5"
            play="voice_ref_files/f.wav"
            code="200" reason="YES"
    />
    <!-- wait for new incoming calls
    forever and generate test results -->
    <action type="wait" ms="-1"/>
  </actions>
</config>
```

Example: making tests calls with wait_until

```xml
<config>
  <actions>
    <action type="call" label="call#1"
            transport="udp"
            wait_until="CONFIRMED"
            expected_cause_code="200"
            caller="15148888888@noreply.com"
            callee="12011111111@target.com"
    />
    <action type="wait"/> <!-- wait until -->
    <action type="call" label="call#2"
            transport="udp"
            wait_until="CONFIRMED"
            expected_cause_code="200"
            caller="15147777777@noreply.com"
            callee="12012222222@target.com"
    />
    <action type="wait" complete/>
  </actions>
</config>
```

Call States
NULL : Before INVITE is sent or received
CALLING : After INVITE is sent
INCOMING : After INVITE is received.
EARLY : After response with To tag.
CONNECTING : After 2xx is sent/received.
CONFIRMED : After ACK is sent/received.
DISCONNECTED

Scenario execution is sequential and non-blocking.

We can use “wait” command with previously set “wait_until” params
to control parallel execution.

flowroute

Email reporting

```
<config>
  <actions>
    <action type="alert"
     email="jchavanton+vp@gmail.com"
     email_from="test@voip-patrol.org"
     smtp_host="smtp://gmail-smtp-in.l.google.com:25"
    />

    <!-- add test actions here ...  -->
    <action type="wait" complete/>
  </actions>
</config>
```

label	start/end	type	result	cause code	reason	duration	from	to
PROD	22-07-2018 15:58:04 22-07-2018 15:58:11	call[2]transport[TLS] peer socket[147.75.60.160:5061] LNHhTtXSFL4yOJFdhw2NZ1O82mOFd4sJ	PASS	200|200	Normal call clearing	expected max hangup connect 0 20 10 5	5147371787	120620928
PROD	22-07-2018 15:58:04 22-07-2018 15:58:11	call[3]transport[TLS] peer socket[34.210.91.112:5061] o2vTEON4Pe-8QGs5EgQ8iN93VS4eR9zl	PASS	200|200	Normal call clearing	expected max hangup connect 0 20 10 5	5147371787	120620928
PROD	22-07-2018 15:58:04 22-07-2018 15:58:12	call[1]transport[TLS] peer socket[34.226.36.32:5061] LUDX7gzZFyz6JFcJV5JVRsz686jyQyl-	PASS	200|200	Normal call clearing	expected max hangup connect 0 20 10 5	5147371787	120620928

flowroute

VoIP Patrol in Docker, files included in the repo

```
voip_patrol/docker$ tree
.
├── build.sh          # docker build command example
├── Dockerfile        # docker build file for Linux Alpine
└── voip_patrol.sh    # docker run example starting
```

```
FROM alpine:3.8

RUN echo "building VoIP Patrol" && pak update \
    && apk add git cmake g++ cmake make curl-dev alsa-lib-dev \
    && mkdir /git && cd /git && git clone https://github.com/jchavanton/voip_patrol.git \
    && cd voip_patrol && git checkout master \
    && git submodule update --init \
    && cp include/config_site.h  pjsua/pjlib/include/pj/config_site.h \
    && cd pjsua && ./configure && make dep && make && make install \
    && cd .. && cmake CMakeLists.txt && make
```

flowroute

Python Jinja2 Templating example

```
from jinja2 import Environment

XML_SCENARIO = """
<?xml version="1.0"?><config>
  <actions>
   <action type="register" label="{{ hostname }}: REGISTER"
           transport="udp" expected_cause_code="200"
           username="{{ username }}" password="{{ password }}"
           realm="proxy.domain.com" registrar="{{ ip }}" />
   <action type="wait"/> <!-- with register/account test, this will return at
                              transaction completion -->
   <action type="accept" label="{{ hostname }}: UDP"
           account="120620934645" max_duration="4" hangup="2"/>
   <!-- add a call action here -->
   <action type="wait" complete/>
  </actions>
</config>
"""

def create_sipproxy_scenario(username, password, hostname, ip, xml_fn):
    xml_file = open(xml_fn,'w')

xml_file.write(Environment().from_string(XML_SCENARIO).render(username=username,
password=password, hostname=hostname, ip=ip))
```

Python Docker API example

```
# run voip_patrol scenario
container.exec_run("./voip_patrol/voip_patrol"
      " --conf /vp/scenarios/{}"
      " -l /vp/logs/voip_patrol.log"
      " -o /vp/logs/{}".format(voip_patrol_scenario_fn, result_fn))

# check voip_patrol results
json_obj = json.load(
    open("{}/logs/{}".format(voip_patrol_root_dir,result_fn)))
for result in json_obj:
    if json_obj[result]["result"] != "PASS":
        CommonLog.error(
             "end2end test failed: {}".format(json_obj[result]))
        sys.exit(0)
    CommonLog.info("test[{}][{}]passed!".format(
         json_obj[result]["action"],
         json_obj[result]["label"]))
```

flowroute

End2End audio quality testing ?

- At the moment we can get a MOS score based on RTP transmission, not using burst loss density and not considering the impact of jitter on the remote endpoint.
(PJSUA does support RTCP-XR, at least partially)

- PESQ and POLQA can not be integrated into an Open source software.
Even if it was just for evaluation purpose …
I did not find any Open source alternative.
(Temporal distortion caused by jitter buffer, expand and accelerate)

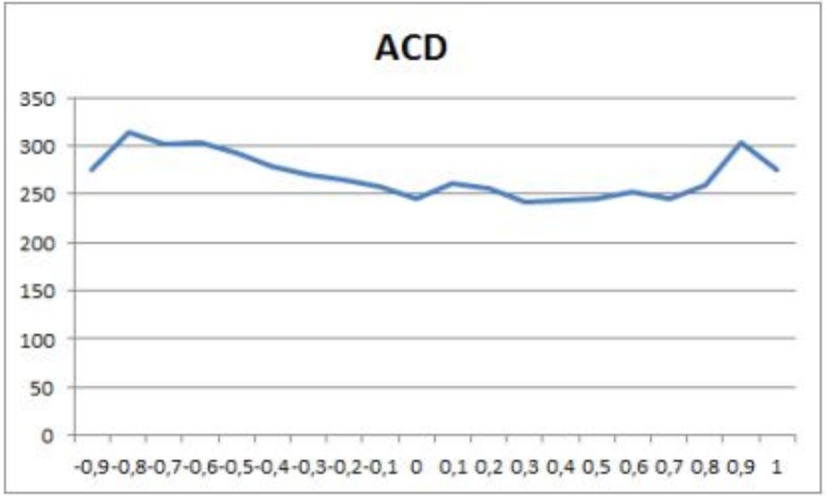
- One option would be to use FFT Fast fourier transform cross-correlation.
To measure the “degradation” from the reference signal in a coefficient of correlation.

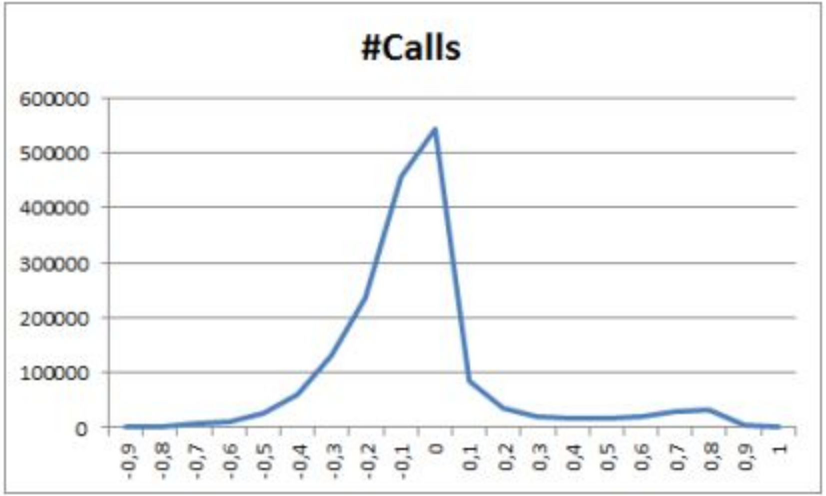
*I did experiment echo detection on Android, doing ring back tone detection with a goertzel algorithm and a coefficient of correlation.

https://github.com/jchavanton/ms2_ring_back_tone_echo_detector

flowroute

Echo detected during ringback tone and Average Call Duration based calls on Android





The first view over the data shows that the very large majority of terminals experience a very small (less than 0.1) or even negative correlation of what they detect with the ring back tone. The negative correlation is a known phenomenon that occurs when the echo canceler cancels so well the ringback tone that even the noise is canceled. Another important remark is that the extreme cases for the RBT correlation (close to -0.8 or 0.8) are not associated with a drop of the ACD. We even experience the opposite phenomenon: ACD is equal or higher is theses cases.

Thanks to Thierry Pochon and Jerome Galtier for the statistical analysis

flowroute

Thanks for listening !

https://github.com/jchavanton/voip_patrol