Modern Performance Testing with Open-Source Tools

Rob Day
Twitter: @day_rk
Email: rkd@rkd.me.uk

- Are you making VoIP software that needs to run at high loads?
- Are you running a VoIP service that might suffer bursts of traffic?
- Are you concerned about quality of service – call setup times and media jitter?

What this talk covers

- SIP performance testing with SIPp
- Diameter/MEGACO performance testing with Seagull
- Automation and integration into test suites
- JSIPp – my recent attempt to fix some of the problems with SIPp

What is this talk not about?

- Functional testing
- IMS performance benchmarking (ETSI TS 186 008)

SIPp

- Describe your call scenario in XML
- Run over 5,000 calls a second (18M calls/hour) per core
- Get success rates, response times, failure rates back out

Sample SIPp scenario

Activities Emacs ▾ Sat 29 Mar, 13:15:54

uac.xml – emacs@localhost.localdomain

```
<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">
    <![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: [service] <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"
        optional="true">
  </recv>

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

  <recv response="183" optional="true">
```

1:--- uac.xml   31% L40   (nXML Valid WS yas Fill)

Sample SIPp command-line output

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
6000.0(0 ms)/1.000s   5061      10.01 s        59188  127.0.0.1:5060(UDP)

  5947 new calls during 1.001 s period   0 ms scheduler resolution
  24 calls (limit 18000)                 Peak was 150 calls, after 7 s
  0 Running, 59174 Paused, 6429 Woken up
  15 dead call msg (discarded)           0 out-of-call msg (discarded)
  3 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
      INVITE ---------->         59176     65        0
         100 <----------         0         0         0         0
         180 <----------         59168     0         0         0
         183 <----------         0         0         0         0
         200 <----------  E-RTD1 59168     0         0         0
         ACK ---------->         59168     0
       Pause [      0ms]         59168                         0
         BYE ---------->         59164     83        0
         200 <----------         59164     0         0         0

------ [+|-|*|/]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause traffic -----

Last Error: Dead call 44798-12562@127.0.0.1 (successful), received 'SIP/...
```

Sample SIPp log files

ecrease Indent

C	D	E	F	G	H	
	ElapsedTime(P)	ElapsedTime(C)	TargetRate	CallRate(P)	CallRate(C)	Inc
32791396098517.543279	00:00:00	00:00:00	6000	0	0	
48881396098518.544888	00:00:01	00:00:01	6000	5923.08	5917.17	
54531396098519.546453	00:00:01	00:00:02	6000	5911.09	5911.18	
34321396098520.548432	00:00:01	00:00:03	6000	5865.13	5893.88	
95911396098521.549591	00:00:01	00:00:04	6000	5932	5901.92	
99341396098522.549934	00:00:01	00:00:05	6000	5883	5898.14	
12461396098523.551246	00:00:01	00:00:06	6000	5906.09	5898.49	
36841396098524.553684	00:00:01	00:00:07	6000	5925.15	5902.3	
48771396098525.554877	00:00:01	00:00:08	6000	5961.04	5909.64	
50631396098526.556063	00:00:01	00:00:09	6000	5887.11	5906.48	
59251396098527.556925	00:00:01	00:00:10	6000	5947	5910.53	
38671396098528.558867	00:00:01	00:00:11	6000	5954.05	5913.94	
00001396098529.560000	00:00:01	00:00:12	6000	5994.01	5920.12	
77881396098529.817788	00:00:00	00:00:12	6000	2124.51	5840.65	
32421396098529.818242	00:00:00	00:00:12	6000	0	5840.18	

Sample SIPp log files

	BI	BJ	BK	BL	BM	BN	
(C)	ResponseTime1(P)	ResponseTime1(C)	ResponseTime1StDev(P)	ResponseTime1StDev(C)	CallLength(P)	CallLength(C)	CallLengt
0	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:
0	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:002000	00:00:00:002000	00:00:00:
0	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:002000	00:00:00:002000	00:00:00:
0	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:002000	00:00:00:002000	00:00:00:
0	00:00:00:001000	00:00:00:000000	00:00:00:026000	00:00:00:013000	00:00:00:006000	00:00:00:003000	00:00:00:
0	00:00:00:001000	00:00:00:000000	00:00:00:000000	00:00:00:012000	00:00:00:002000	00:00:00:003000	00:00:00:
0	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:011000	00:00:00:002000	00:00:00:003000	00:00:00:
0	00:00:00:000000	00:00:00:000000	00:00:00:000000	00:00:00:010000	00:00:00:002000	00:00:00:003000	00:00:00:
0	00:00:00:000000	00:00:00:000000	00:00:00:001000	00:00:00:009000	00:00:00:002000	00:00:00:003000	00:00:00:
0	00:00:00:004000	00:00:00:001000	00:00:00:045000	00:00:00:017000	00:00:00:012000	00:00:00:004000	00:00:00:
0	00:00:00:001000	00:00:00:001000	00:00:00:000000	00:00:00:016000	00:00:00:002000	00:00:00:004000	00:00:00:
0	00:00:00:001000	00:00:00:001000	00:00:00:000000	00:00:00:015000	00:00:00:003000	00:00:00:004000	00:00:00:
0	00:00:00:001000	00:00:00:001000	00:00:00:001000	00:00:00:015000	00:00:00:003000	00:00:00:004000	00:00:00:
0	00:00:00:048000	00:00:00:001000	00:00:00:146000	00:00:00:020000	00:00:00:084000	00:00:00:004000	00:00:00:
0	00:00:00:000000	00:00:00:001000	00:00:00:000000	00:00:00:020000	00:00:00:000000	00:00:00:004000	00:00:00:

Limitations

- RTP support
- Scenario model can be inflexible
- Log file parsing can be tricky (and requires spare disk space)

Seagull

- Multi-protocol test tool from same team as SIPp
- (I don't maintain this)
- Similar principles – scenario defined in an XML file
- Diameter and H.248 are probably the most interesting

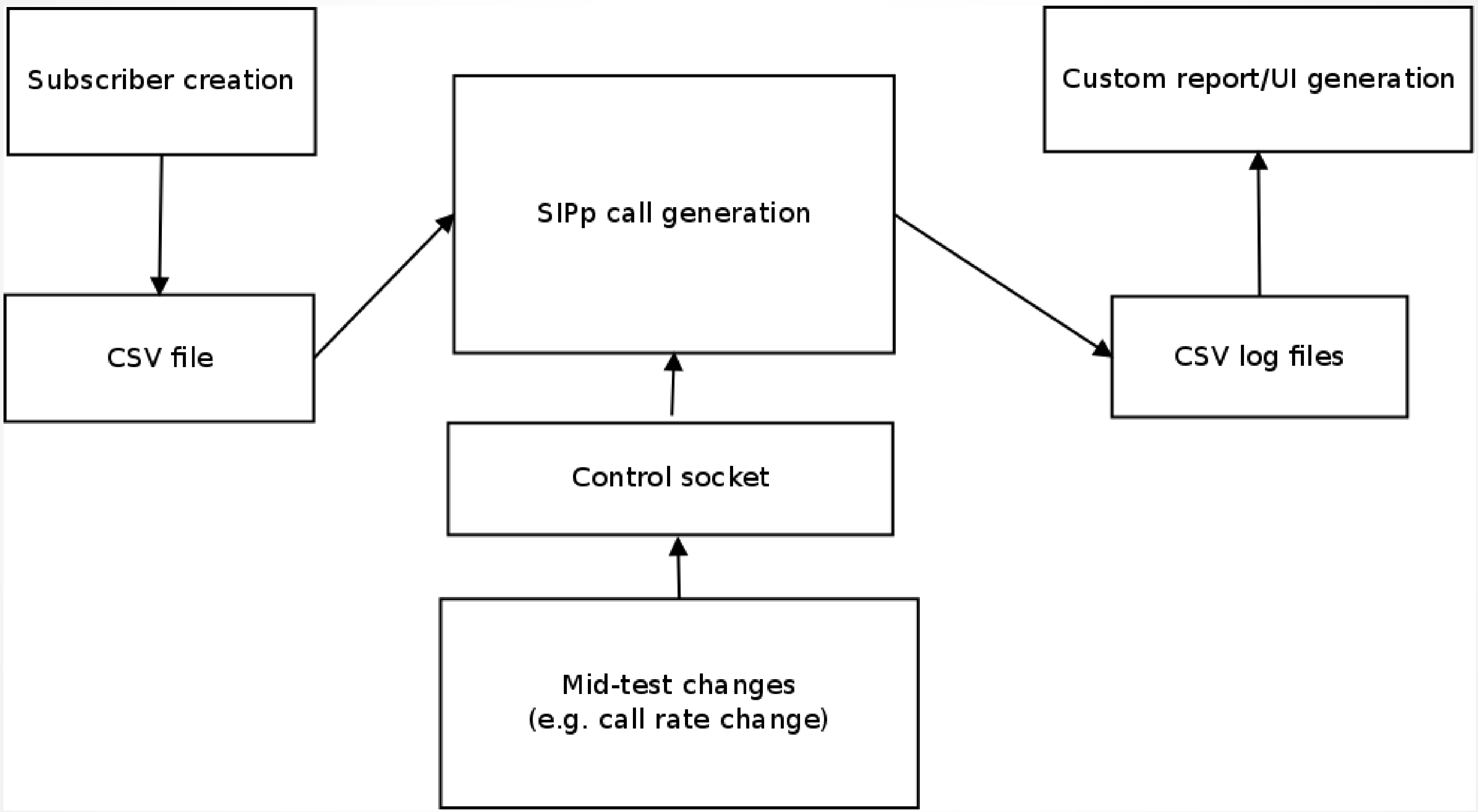
```
  <send channel="trans-ip-v4">
    <command name="CER">
      <avp name="Origin-Host" value="seagull.ims.hpintelco.org"> </avp>
      <avp name="Origin-Realm" value="ims.hpintelco.org"> </avp>
      <avp name="Host-IP-Address" value="0x00010a03fc5e"> </avp>
      <avp name="Vendor-Id" value="11"> </avp>
      <avp name="Product-Name" value="HP Cx Interface"> </avp>
</init>
```

```
<receive channel="trans-ip-v4">
    <action>
      <stop-timer></stop-timer>
    </action>
    <command name="SAA">
    </command>
  </receive>
```

```
<send channel="channel-1">
  <action>
    <inc-counter name="transaction-counter"></inc-counter>
    <set-value name="transaction-id"
        format="$(transaction-counter)"></set-value>
  </action>
  <message>
    <!-- header -->
    <![CDATA[!/1 [16.16.88.188\]:55554
            T=18571]] >
    <!-- body -->
    <![CDATA[C=${A=${M{TS{SI=iv,BF=off},
            ST=1{O{MO=sr,RV=off,RG=off},
            R{m=audio 49152 RTP/AVP 3 97 98 8 0 101
              c=IN IP4 16.16.214.175
              a=rtpmap:3 GSM/8000
              a=rtpmap:101 telephone-event/8000
              a=fmtp:101 0-11,16
             }}}}}}]] >
  </message>
```

Integration

- CSV injection
- Control sockets:
  - `http://host:port/seagull/command/ramp&value=n&duration=d`
  - `echo '*' >/dev/udp/127.0.0.1/8888`
- Log file parsing
  - Success rates
  - Response times
  - SIP error codes
- Running commands
  - SIPp's `<exec>` action

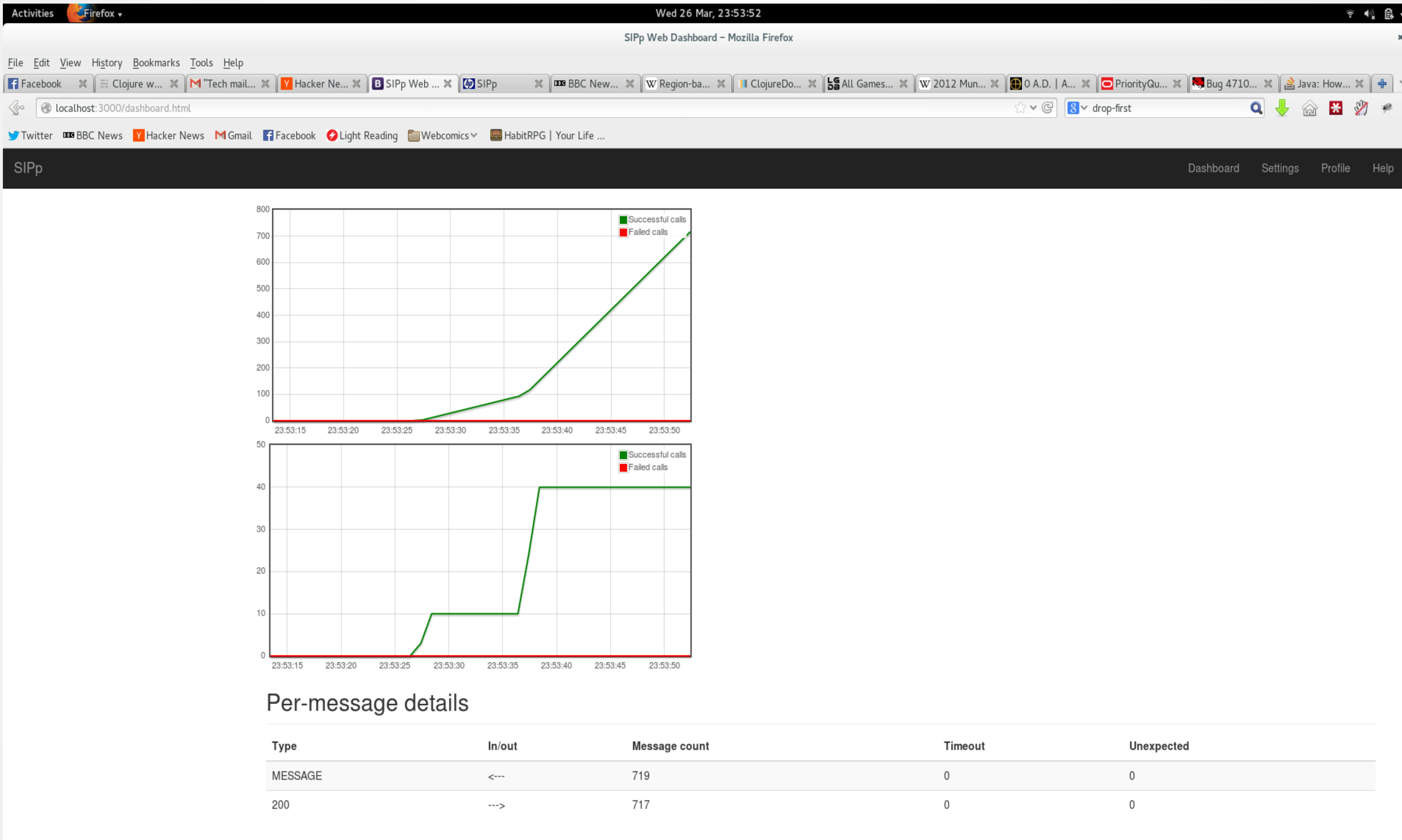

Subscriber creation
CSV file
SIPp call generation
Control socket
Mid-test changes
(e.g. call rate change)
CSV log files
Custom report/UI generation

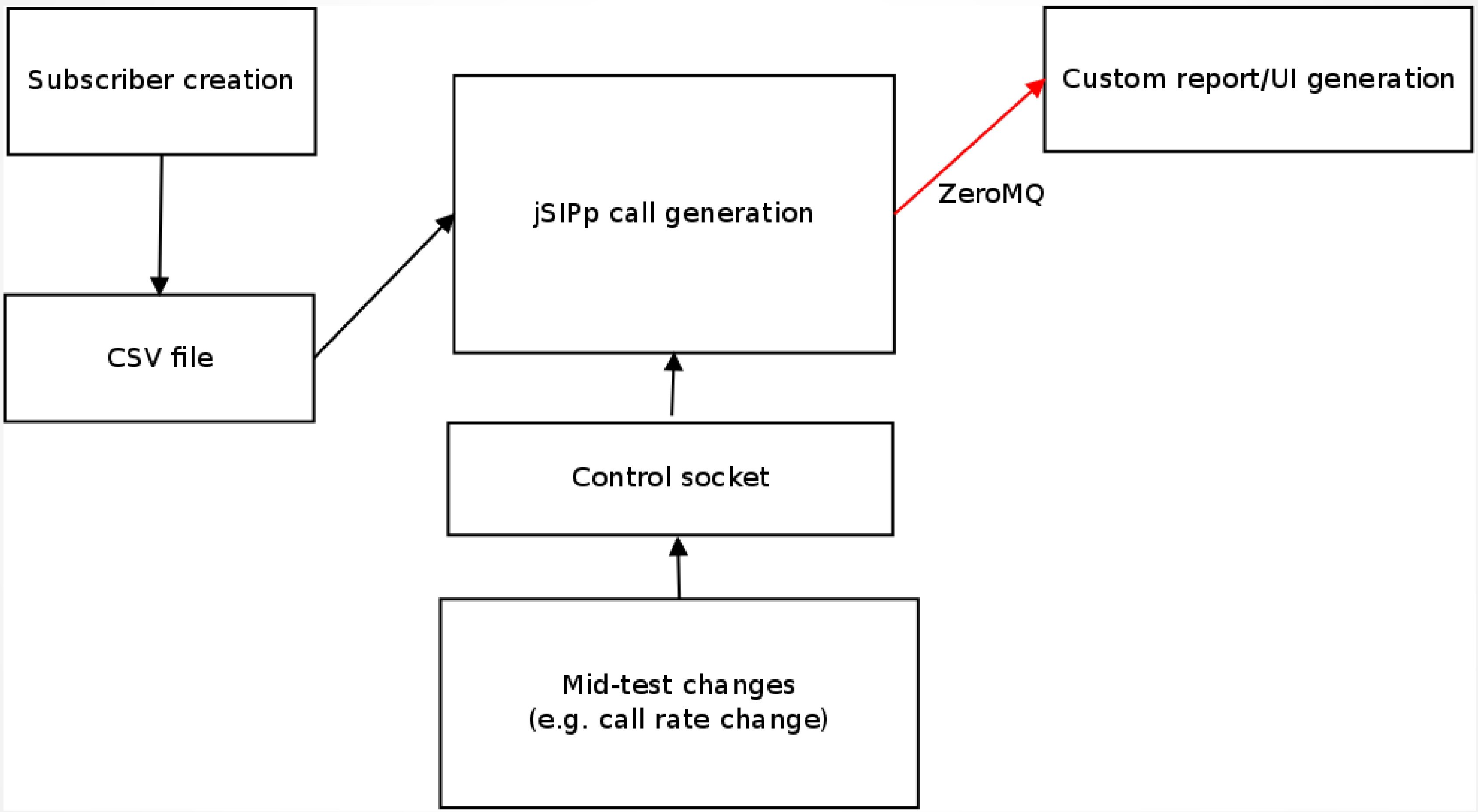
jSIPp

- Recent rewrite of SIPp in Java
- Use of Java and building on OSS libraries mean 90% reduction in codebase size
- More flexible, easier to add features
- Uses the same XML files and gets similar performance

jSIPp – the major change so far

- Stats are now published over ZeroMQ, a lightweight messaging protocol
- Every successful call, unexpected message, every timeout – all with timestamps
- A platform for writing test infrastructure


Activities
Firefox
Wed 26 Mar, 23:53:52
SIPp Web Dashboard – Mozilla Firefox
localhost:3000/dashboard.html
SIPp
Dashboard
Settings
Profile
Help
Successful calls
Failed calls
23:53:15
23:53:20
23:53:25
23:53:30
23:53:35
23:53:40
23:53:45
23:53:50
Per-message details
Type | In/out | Message count | Timeout | Unexpected
MESSAGE | <--- | 719 | 0 | 0
200 | ---> | 717 | 0 | 0


Subscriber creation
CSV file
jSIPp call generation
Custom report/UI generation
ZeroMQ
Control socket
Mid-test changes
(e.g. call rate change)

Why Java?

- **<u>The existing C++ code isn't going away</u>**
- Manual memory management = yikes!
- Speed
- Good SIP/RTP parsers already available
- The future: easy JRuby/Jython/Groovy scripting based on the SIPp core

jSIPp - coming up

- Better RTP testing – including getting RTCP stats out for jitter analysis
- More flexible scenarios – including registering then receiving a call
- Support for SIP-over-WebSocket performance testing
- I'm open to suggestions!

References

- SIPp
  - Docs: http://sipp.sourceforge.net/doc/reference.html
  - Mailing list: sipp-users@lists.sourceforge.net
- Seagull
  - Docs: http://gull.sourceforge.net/doc/
  - Mailing list: gull-users@lists.sourceforge.net
- jSIPp
  - Github page: https://github.com/rkday/jsipp