

Table of Contents

1. Description.....	2
2. Test Results.....	2
2.1. Multiparty tests.....	2
2.1.1 Parallel forking test.....	2
2.1.2 Loop detection test.....	3
2.2. Presence test.....	4
2.2.1. Interaction with xcap server.....	4
2.2.2. Presence	4
2.2.3. RLS.....	4
2.3. Individual tests.....	5
2.1.1 Tested scenarios.....	5
2.1.2 Found issues.....	5
3. Conclusions.....	6

Authors:
Anca-Maria Vamanu
Bogdan-Andrei Iancu

1. Description

For the first time OpenSIPS/OpenSER participates to SIPit – SIP interoperability event, 13-17 October 2008, in Lannion, France.

The main goal was to test the implementations (code and specs) of a lot of features that were added across the years but never tested against other SIP implementation – a feature is not only good to have, but better good to work!

The tests focused on several hot topics like server discovery (SRV/NAPTR), Presence and transport protocols (UDP/TCP/TLS/SCTP).

2. Test Results

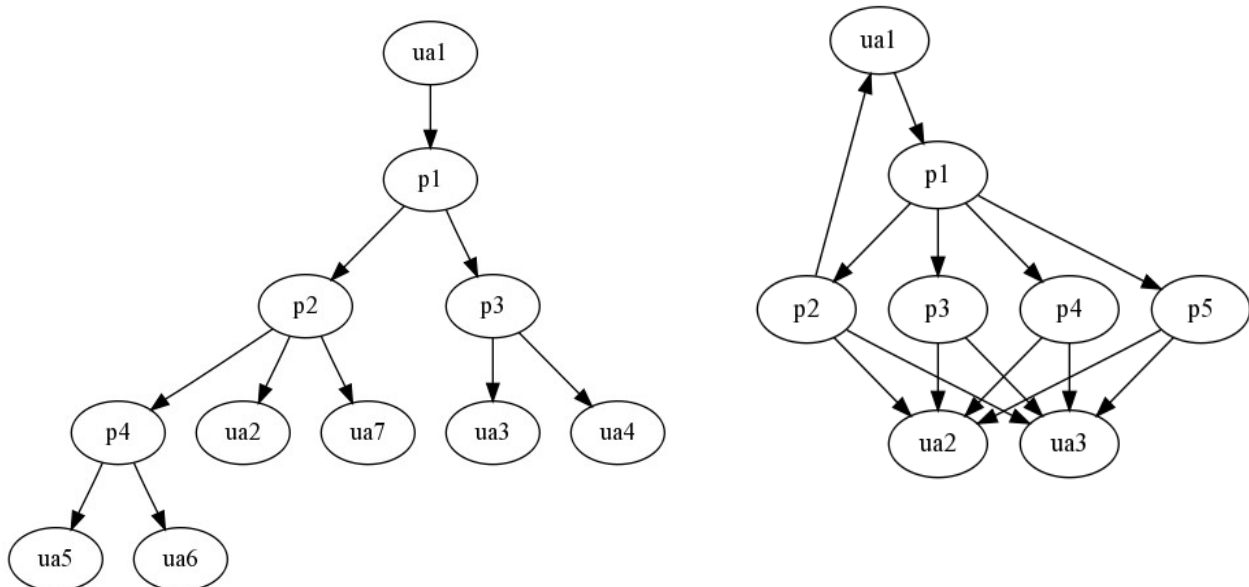
2.1. Multiparty tests

2.1.1 Parallel forking test

Parallel forking tree - 3 level of proxies with up to 3 branches from each proxy.

Participants – 4 proxies(including OpenSIPS)

Participants – 7 UACs



Test scenarios:

- placing a call through all the tree -> all UAC devices must ring and UAS to get ringing indication
- after placing a call, generate timeout in different points (levels) of the tree -> proxy canceling and timeout replies
- after placing a call, the UAS cancels the call -> cancel propagation through all the tree
- placing a call and call pick up (from different UAC)
- placing a call and multiple 200 OK (pick ups) from different UACs
- transport changing (UDP->TCP) in different nodes; forking with multiple protocols

Results:

- there were no issues identified with OpenSIPS implementation. All scenarios were successfully passed.
- Interoperability – during CANCEL tests, the 487 INVITE reply from a client was not matching the transaction, so no ACK was generated – still to investigate the trace to locate the problem.

Remarks:

- T1 timer is constant – RFC 3261 does not allow to be changed
- RFC 4320 / RFC 4321 – SIP's Non-INVITE Transaction – do not generate 408 if timeout.
- In stateful mode, if a 200 OK does not match a transaction -> drop it
- in stateful mode, if an ACK without RR does not match a transaction -> drop it

2.1.2 Loop detection test

Looping a call through a circular chain of servers.

Participants – 4 proxies (including OpenSIPS).

Participants – 3 proxies (we need only chain end points)

Test scenarios:

- inject the call in the proxy circle – loop detection based on Max-Forward header
- inject the call in the proxy circle – loop detection based on draft-ietf-sip-fork-loop-fix-07

Results:

- OpenSIPS/OpenSER does not implements draft-ietf-sip-fork-loop-fix-07
- minor issues with Max Forward processing for non-Invite requests (see remarks below).

Remarks:

- if the requests targets the proxy (no username part, like in OPTIONS), a Max-Forward 0 should not trigger Too Many Hops!

2.2. Presence test

2.2.1. Interaction with xcap server

Testing scenarios:

- OpenXCAP 1.0.4 : integrated xcap server mode
- non integrated server – HTTP queries (without authorization support).

The tests were successful and no problem found.

2.2.2. Presence

Testing scenarios:

- presence with privacy rules using XCAP server
- basic Subscribe/Notify without privacy rules

The tests were successful, and no functional problem was noticed. However, the logs revealed three small bugs which will be fixed in the near future:

- missing of a Notify for winfo when a user changes the presence privacy rules document
- Notify for winfo body will have the same id for records of subscriptions coming from the same uri
- sending Notifies for presence on callback for Notify for presence;winfo caused multiple Notifies being sent when more subscriptions for winfo exist.

2.2.3. RLS

Testing partner:

- **AG Projects** - SIP SIMPLE test client

Since RLS was not really tested as there was not a known client supporting it (but was modeled after an old version of Eyebeam that had this feature) , this part revealed some inadequates with the standard and some mis functionalities:

- using rls-services document to get list components, instead of resource-list document that was used before

- consider document names as valid per user and not globally accessible
- sending 200OK after sending Subscriptions to the contacts in the list causes retransmissions to occur

RLS is still work in progress and we are glad there is a client that we can test our server with. We will be doing some more testing and fixing in this part in the near future.

2.3. Individual tests

2.1.1 Tested scenarios

UAC devices:

- registration / unregistration
- authentication (registration and calling)
- NAPTR / SRV – server discovery
- UDP / TCP / TLS
- proxy changing the protocol
- Record Routing
- FROM changing
- changing SDP (forcing RTPproxy) and re-INVITES
- 100rel (Require header)
- multi SDP offer during parallel forking (183 Early Media and 200 OK).

UAS devices:

- NAPTR / SRV – server discovery
- UDP / TCP / TLS
- proxy changing the protocol
- Record Routing
- FROM changing

2.1.2 Found issues

Remarks:

- maybe OPTIONS (server target) in OSIPS should be done automatically (also the capabilities to be automatically discovered)
- SIPS support – it looks like there is a problem in the SIPS implementation in OSIPS – consulting Robert Sparks on the matter, here is correct

behavior: SIPS scheme must not be overwritten with any other scheme disregarding the number of hops or the local policy. SIPS must be preserved by all SIP entities and TLS used for delivering the calls. OSIPS does not implement any mechanism to ensure the SIPS preservation.

- OSIPS – to investigate – sip:name;transport=tls -> does the SIP resolver returns the 5061 port (as it should), if no SRV is available ?
- SCTP – kernel problems on linux – Kernel Panic when receiving a SCTP connection – to investigate the interoperability of the SCTP stacks in linux and BSD.
- OSIPS – even if the registered contact was not correct as it was registered via TCP without “transport” param, OSIPS was storing the TCP interface, but still using UDP to deliver the call – To fix – if the URI contains no explicit transport information, shouldn't the forced socket dictate the transport to be used?
- OSIPS – to investigate double routing – the 200 OK ACK failed to switch protocol from UDP to TCP.

3. Conclusions

The multiparty and individual tests showed a SIP world with a more robust implementations (UACs and proxies) – registration, transaction processing, authentication, call routing (RR and Route) are no longer an issue.

At transport level, TLS is already commonly supported (implemented and interops) by almost all SIP entities (phones and proxies) – this open the perspective of large scale usage of SIPS scheme – at this point, indeed, there is room left for more work – OpenSIPS/OpenSER and other implementation still have issues in properly implementing SIPS scheme.

SCTP is the hot topic when comes to transport – several SIP implementation already have SCTP, but unfortunately, the high dependency to the OS makes things complicated. Observed by our team and by the iptel/tekelec team (SER), the linux kernel implementation for SCTP still has some issues with the SCTP stack from other OSs...

A good progress was noticed at the DNS level – server discovery based on NAPTR/SRV lookups – almost all implementation supports NAPTR/SRV and most of them are doing lookup each time when a request is sent out (opposite to a single lookup at startup time).

For presence, looking at the number of implementations (server and client) and at the interoperability issues, it is true to say that presence is still in the early stage. From this point of view, this SIPIT was a excellent opportunity to align to specs and to strengths the OpenSIPS presence implementation. Huge progress

was done in the understanding of RLS (Resource Lists Server), especially that there are just a few other parties to test with.

On UAC side, it is strangely to see that many UACs are out there without properly supporting SIP forking – this in the particular case they have to deal with multiple SDP offers coming from different branches. Especially now, when more and more advanced features are build on top of Early Media feature (injecting media before the call, Ring Back Tones, etc).