Interoperability Testing for SIP

Jonathan Rosenberg
Bell Laboratories
jdrosen@bell-labs.com

Henning Schulzrinne
Columbia University
hgs@cs.columbia.edu

April 7, 1999

Abstract

This document describes a set of tests that can be performed by SIP equipment in order to verify interoperability and correctness of implementation.

Contents

1 Introduction

The tests described here are meant as a basic means for demonstrating interoperability among SIP [1] implementations.

2 Client to Client Basic

This suite test validates the ability of two user agents (either end systems or gateways) to correctly communicate.

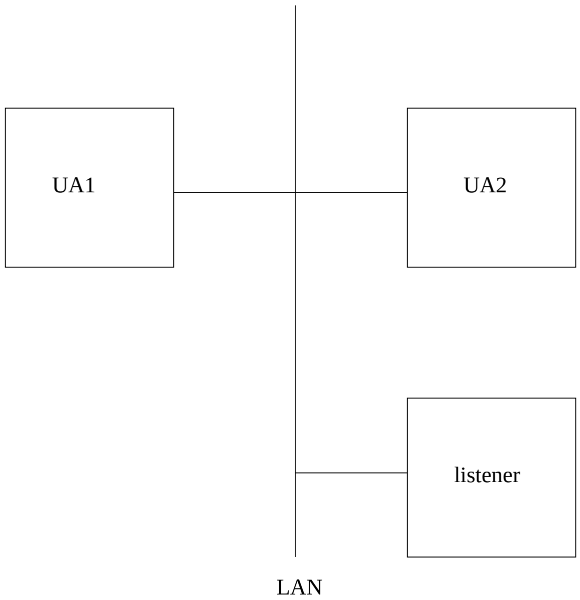

Figure 1: Client to Client Scenario

The setup for the test is shown in Figure 1. There are just two user agents, a caller and callee, communicating directly over a LAN. Another host, running a tcpdump utility (or something equivalent) listens and prints all messages exchanged between the SIP systems.

Both caller and callee SHOULD be capable of sending and receiving G.711 audio. The caller SHOULD indicate its desire to participate in a single media stream, with G.711 only, when setting up the call.

2.1 Initiation (UA1)

The caller initiates a call, causing a UDP INVITE to be sent to the callee. The callee should alert the user somehow, and MAY send a provisional response back to the caller. The callee accepts the call. This should cause a 200 OK to be sent back to the caller (again by UDP), which then causes an ACK to be sent to the callee. The SDP offered by the caller should indicate a single audio session, consisting of G.711 audio only. The SDP returned to the caller in the 200 OK should indicate acceptance of this codec.

The test is deemed *partially successful* if the SIP exchange completes correctly, and *fully successful* if media is successfully exchanged between the clients.

The packets sent on the network SHOULD be examined, to check for correct formatting. In particular, the following should be checked:

- Inclusion of a correctly formatted Via field in the request and mirroring of this in the response
- Inclusion of a correctly formatted To and From field in the request, and mirroring in the response (with possible addition of a tag, although its a MAY in this scenario)
- A Content-Type header indicating application/sdp
- Correct request line formatting
- Inclusion of a CSeq in the request, including the INVITE tag after the number
- Inclusion of a Call-ID in the request, and mirroring of this in the response.

2.2 Call Termination A to B (UA2)

The call initiation test is performed, as above. Then, the caller hangs up the call. This should cause a UDP BYE to be sent to the callee, and a 200 OK response generated automatically, sent to the caller, and accepted. Media should no longer be sent between the parties.

The test is deemed *partially successful* if the SIP exchange completes, and *fully successful* if media is no longer exchanged.

In addition, the following items should be checked:

- The Call-ID in the BYE is identical to that in the INVITE in the initiation.
- The CSeq numerical value in the BYE is larger than that of the INVITE
- The CSeq method tag is BYE in the BYE request
- The To and From fields mirror those in the original INVITE request
- The BYE request contains a Via field, mirrored in the response

2.3 Call Termination B to A (UA3)

The call initiation test is performed, as above. Then, the callee hangs up the call. This should cause a UDP BYE to be sent to the caller, and a 200 OK response generated automatically, sent to the callee, and accepted. Media should no longer be sent between the parties.

The test is deemed *partially successful* if the SIP exchange completes, and *fully successful* if media is no longer exchanged.

In addition, the following items should be checked:

- The Call-ID in the BYE is identical to that in the INVITE in the initiation.
- The CSeq method tag is BYE in the BYE request
- The To in the BYE field mirrors that of the From in the original INVITE
- The From field in the BYE mirrors that of the To field in the original INVITE
- The BYE request contains a Via field, mirrored in the response
- The request URI of the BYE is the same as the From field of the INVITE, or of the Contact in the INVITE, if present.

2.4 Call Rejection (UA4)

The basic call setup procedure of section 2 is used, except the callee rejects the call. This should cause either a 600 or 400 class response to be sent back to the caller. That, in turn, should trigger an ACK to be sent to the callee. The caller should be notified of the rejection in some way, and no media should be exchanged.

If the callee user agent allows multiple rejection types (486 Busy vs. 600 class), each should be tested in turn, to verify that each is understood by the UAC. In each case the UAC should at least send an ACK, indicate to the caller that the call was rejected, and not exchange media.

In addition, the following items should be checked:

- Call-ID is mirrored in the response
- To and From mirrored in the response, with possible addition of a tag
- CSeq mirrored in the response

2.5 Call Redirection I (UA5)

The basic call setup procedure of section 2 is used, except the callee redirects the call. This should cause a 300 class response to be sent back to the caller. That, in turn, should trigger an ACK to be sent to the callee. The caller should be notified of the redirection response, and no media should be exchanged.

In the first version of this test, the redirection response contains a single Contact header, indicating a SIP URL of some sort. This test can only be performed with UAS's which allow user agents to enter in a redirection address. This URL should be parsed by the UAC, and displayed to the user in some form.

2.6 Call Redirection II (UA6)

The basic redirect test of section 2.5 is peformed. However, the UAS enters in a redirect address that is a non SIP URL, such as a mailto or http URL. This should be parsed by the UAC, and displayed or used in some way.

2.7 Call Redirection III (UA7)

The basic redirect test of section 2.5 is performed. However, the UAS enters in multiple redirect addresses. These should appear in Contact header(s) in the redirection response. They should be parsed correctly by the UAC, and both displayed to the user in some fashion.

2.8 TCP Invitation (UA8)

The basic test of section 2 is performed. However, the invitation (and the responses) should be sent using TCP. The test is deemed successful if the call is setup correctly, based on the same definitions in 2. In addition, the following should be checked:

- The Via field indicates SIP/2.0/TCP
- The response is sent to the same port the request came from. This can be checked by a tcpdump.
- The requests are not retransmitted

2.9 re-INVITES (UA9)

This test verifies the ability of a user agent to process re-INVITEs correctly. The basic call setup procedure of section 2 is performed. After the call is setup, UA1 changes some aspect of the call, such as indicating desire to use another codec. This should cause UA1 to generate a re-INVITE to UA2. If the change is acceptable to UA2, it should respond with a 200 OK, else a 400 class response. This test is only possible with a UAC which can change codecs mid-call, and make use of re-INVITEs. For best results, UA1 should elect to use a codec understood by UA2.

The test is deemed partially successful if the following occurs:

- UA1 generates a new INVITE and sends it to UA2
- The new INVITE has a higher CSeq number than the original INVITE
- The version number in the SDP has increased
- The SDP indicates a new codec in the m=audio line, with a new payload type number
- UA2 either accepts the request, and sends a 200 OK, or else rejects it with a 400 class response
- A 200 OK response from UA2 should mirror the new codec in the SDP
- UA1 sends an ACK to UA2

The test is fully successful if, for a 200 OK from UA2, media continues to be exchanged, and UA1 switches to the new codec in the media stream, and this media understood by UA2. For a 400 response from UA2, the test is successful if media continues to be exchanged using the old media type.

3 Registration

This test verifies interoperability for SIP registrations. The configuration is shown in Figure 2. A SIP user agent (UA1), capable of performing registrations, communicates with a SIP registrar. An additional user agent (UA2) is used to make calls to UA1. A tcpdump entity is running on the network to examine the messages being passed back and forth.

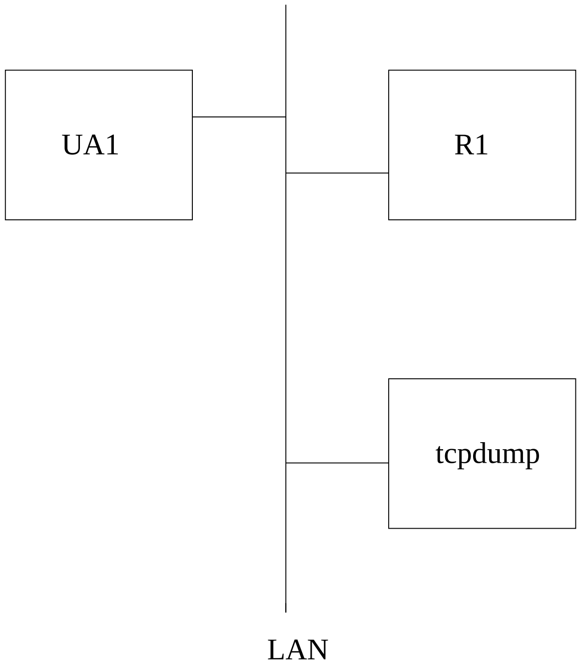

Figure 2: Registration Configuration

3.1 Basic Registration (REG1)

UA1 is statically configured with the address of the registrar. It sends a UDP REGISTER message to the registrar, containing at least one Contact address. The registrar receives and processes the registration, and sends back a 200 OK response. The test is deemed *partially successful* if the following occur:

- The REGISTER message contains a Call-ID field
- The REGISTER message contains a CSeq, with the method tag of REGISTER
- The REGISTER message contains a request-URI that contains only a host name, and no username
- The REGISTER message contains a To field with the name of the user on UA1
- The REGISTER message contains a From field with the name of the user on UA1
- The REGISTER message contains a Contact field, listing the URI where the UA is reachable
- The 200 OK mirrors the To, From, Call-ID, and CSeq fields from the request, with the possible addition of a tag in the To field.
- The 200 OK contains either an Expires header, or an expires Contact parameter, indicating an expiration time at a time later than the expiration sent in the request (if present), otherwise later than the current time.
- The 200 OK mirrors the registered URI in the Contact header
- No ACK is sent by UA1

To verify that the registration has been accepted by the registrar, UA2 sends an INVITE message to the registrar (it must support local proxies to do so), containing users' address (the address in the To field of the REGISTER) in the Request-URI and To field. The registrar, acting as either a proxy or redirect server, should proxy or redirect to the address registered (the address in the Contact header of the REGISTER message). The basic test is deemed *fully successful* if this occurs.

3.2 Registration Refresh (REG2)

The basic registration of section 3.1 is performed. However, the server is configured to expire the registration after a short period of time (less than one minute). This test is only possible if a server can be so configured. The 200 OK response for this REGISTER should be checked to see if it indicates the shortened registration interval.

Before the registration expires, the client should refresh the registration by sending another REGISTER message. The test is deemed successful if the registration is resent before expiration. The REGISTER message should also be checked for the following (beyond those checks indicated in 3.1):

- The CSeq in the second registration is higher than in the first
- The Call-ID in the second registration is the same as the first

3.3 Registration Expiration (REG3)

This tests the servers ability to expire registrations correctly.

The basic registration test of section 3.1 is performed. However, the server is configured to expire the registration after a short period of time (less than one minute). This test is only possible if a server can be so configured.

Once the registration is received, and a 200 OK has been sent and accepted, UA1 exits. It may or may not send another REGISTER to remove its previous registrations. After waiting a minute, UA2 initiates a call to UA1 using the same procedure specified in 3.1. Since the registration should have been expired, the registrar should send a 400 class response (preferrably 404 Not Found) to UA2. The test is deemed successful if this 400 class response is sent to UA2.

3.4 Multicast Registrations (REG4)

The basic registration test of section 3.1 is repeated. However, the REGISTER is sent to the sip multicast address (224.0.1.75). This test is possible only if UA1 can be configured to send its registrations to this address. The test is deemed successful based on the same criteria defined in section 3.1.

3.5 TCP Registration (REG5)

The basic registration test of section 3.1 is repeated. However, the REGISTER is sent using TCP instead of UDP. The test is possible only if UA1 supports TCP registrations. The test is deemed successful based on the same criteria defined in section 3.1, with the following additions:

- The Via field in the request should indicate SIP/2.0/TCP
- The Contact address in the registration should indicate TCP (although it may not if the UAC is willing to accept UDP invitations)
- The response should not be retransmitted
- The client should close the TCP connection after receiving the 200 OK

3.6 Multiple Registrations (REG6)

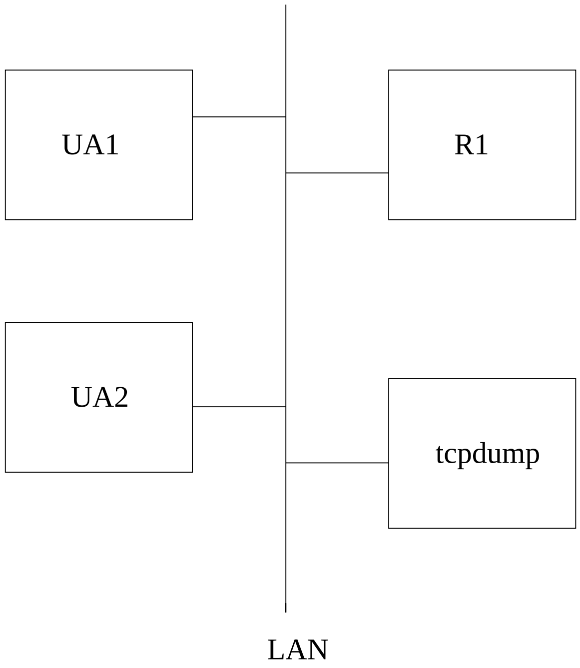

Figure 3: Multiple Registrations

This is a more complex registration test than that described in section 3.1. In this test, as show in Figure 3, there are three UA's - UA1, UA2, and UA3, along with a single server. UA1 and UA2 both send REGISTER messages (via UDP) to the registrar, for the same To address, but with different Contact headers (reflecting the fact that they are different UA's).

UA1 sends the REGISTER first, and after the 200 OK has been received, UA2 sends it registration. Once this 200 OK has been received by UA2, UA3 sends an INVITE to the registrar, with the To and Request-URI set to the same address in the To fields of both registrations. The registrar may act as either a proxy or redirect server. In the former case, it should fork the request to both user agents, and in the latter case, it should send a 300 class response containing both Contact addresses.

The test is deemed successful if the following occur:

- The 200 response to UA2's REGISTER contains two addresses in the Contact header, corresponding to UA1 and UA2's address
- The INVITE from UA3 is redirected with two Contact addresses, or the request is forked to both UA1 and UA2
- The expiration time for the UA1 address, sent in the 200 OK to UA2's registration, matches the expiration time sent to UA1 in response to UA1s REGISTER request.

4 Proxy Tests

This section contains a number of tests aimed at verifying interoperability between UA's and proxies. Note that all of these tests require a UAC to support local proxies.

4.1 Basic Stateful Proxy (P1)

The basic proxy configuration is shown in Figure 4. There are two user agents, UA1 and UA2, and a single proxy server P1. UA2's address is known to P1, either by a registration, or by static configuration. A tcpdump machine is listening on the network connections between UA1 and P1, and between P1 and UA2.

UA1 sends a UDP INVITE to P1. UA1 must be configured to use local proxies in order to do this. The INVITE contains the address which P1 is aware of, that is translated to the address of UA2. P1 then proxies the request to UA2. UA2 accepts the call. A 200 OK is sent to P1, and then to UA1. UA1 sends an ACK, either directly to UA2 (if UA2 inserted a Contact header into the response), or through P1 if it did not. The test is deemed partially successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE

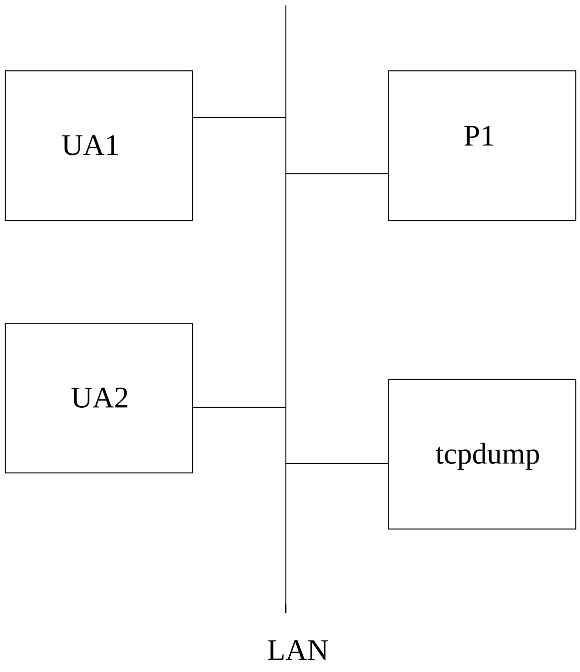

Figure 4: Basic Proxy Configuration

- The proxied INVITE from P1 to UA2 has an additional Via header, before the Via inserted by UA1, containing its own UDP address
- P1 sent a 100 response to UA1 upon receipt of the request
- P1 forwards the 200 OK response to UA1, removing its Via field from the response
- P1 never retransmits the 200 OK; it only forwards 200 OK's upstream that it receives from UA2
- If the ACK is sent through P1, P1 inserts a Via field into the ACK before forwarding it to UA2
- UA1 reports a call acceptance to the user

The test is deemed fully successful if media is also exchanged between UA1 and UA2.

4.2 Proxy with Rejecting UAS (P2)

The same test as in section 4.1 is performed. However, instead of accepting the call, UA2 rejects the call, with either a 400, 500, or 600 class response. P1 should send an ACK to UA2, and then forward the rejection to UA1. UA1 should then send an ACK to P1. The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- The proxied INVITE from P1 to UA2 has an additional Via header, before the Via inserted by UA1, containing its own UDP address
- P1 sent a 100 response to UA1 upon receipt of the request
- UA2 sends a rejection response to P1
- The rejection response has a tag inserted in the To field
- P1 sends an ACK to UA2
- The ACK from P1 to UA2 contains only a single Via field, that of P1
- P1 forwards the response to UA1

- The response from P1 to UA1 contains a single Via field, that of UA1
- UA1 sends an ACK to P1
- P1 does not forward the ACK from UA1 towards UA2
- UA1 reports a call rejection to the user

4.3 Parallel Forking Proxy (P3)

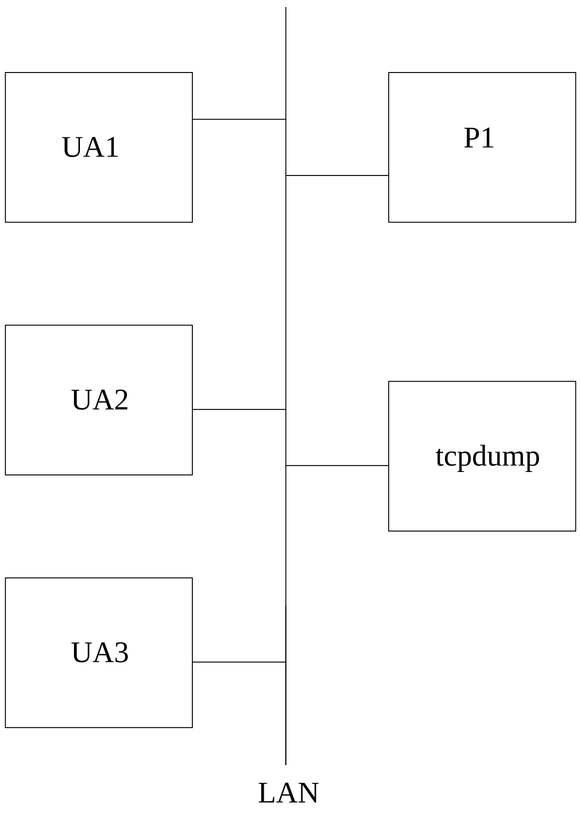

Figure 5: Forking proxy Configuration

This test configuration is shown in Figure 5. There are three user agents, UA1, UA2 and UA3, and a single proxy, P1. P1 knows that some address X translates to the address of UA2 and UA3. This can be done either through registrations (see section 3.6) or by static configuration of the proxy server. P1 is configured to act as a forking proxy, using parallel searches.

UA1 sends an INVITE for address X (listed in the Request URI and To field) to P1. P1 forks this request, sending it to UA2 and UA3 simultaneously. UA2 rejects the call, sending a 400,500 (but not 600) class response. Five seconds later, UA3 accepts the call, sending a 200 OK to P1. P1 then forwards this request upstream to UA1. UA1 then sends an ACK, either directly to UA3, or to P1 first. P1 then forwards this ACK to UA3.

The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- The proxied INVITE from P1 to UA2 and UA3 has an additional Via header, before the Via inserted by UA1, containing its own UDP address
- The requests from P1 to UA2 and UA3 each contain a branch parameter in the Via field, and the value of this parameter is different for each request
- P1 sent a 100 response to UA1 upon receipt of the request
- UA2 sent a 400 or 500 response to P1
- P1 did not forward the 400 or 500 reponse from UA2 towards UA1
- P1 sent an ACK to UA2
- The ACK from P1 to UA2 contained a Via field with a single address, that of P1
- UA3 sends a 200 OK to P1, containing a tag in the To field
- P1 forwards the 200 OK towards UA1, after removing its Via field
- UA1 sends an ACK, either to P1 or directly to UA3
- P1 does not retransmit the 200 OK itself, but forwards any retransmissions from UA3 to UA1

4.4 Parallel Forking Proxy with CANCEL (P4)

The forking test of section 4.3 is repeated. However, instead of rejecting the call with a 400 or 500 class response, UA2 rejects the call with a 600 class response. This should cause P1 to send a CANCEL to UA3. UA3 should respond with a 200 OK to the CANCEL, and should stop alerting the user. P1 should then forward the 600 response to UA1, which responds with an ACK. The test assumes that CANCEL is supported by P1 and UA3, and that this feature is enabled, as this is an optional feature of the specification.

The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- The proxied INVITE from P1 to UA2 and UA3 has an additional Via header, before the Via inserted by UA1, containing its own UDP address
- The requests from P1 to UA2 and UA3 each contain a branch parameter in the Via field, and the value of this parameter is different for each request
- P1 sent a 100 response to UA1 upon receipt of the request
- UA2 sent a 600 response to P1
- P1 sends an ACK to UA2, and the ACK contains a single Via field, containing its own address, and this ACK causes response retransmissions from UA2 to cease
- P1 sends a CANCEL to UA3, and the CANCEL contains a single Via field, containing its own address
- The CANCEL contains the same CSeq number as the INVITE, but with the method tag set to CANCEL
- UA3 sends a 200 OK to P1, mirroring the To, From, CSeq, and Call-ID fields from the CANCEL. This 200 OK is accepted by P1, and causes the CANCEL request to no longer be retransmitted by P1
- UA3 ceases alerting the user
- P1 ceases retransmitting the INVITE request to UA3
- P1 forwards the 600 response to UA1, after removing its Via field
- UA1 accepts the 600 response, and notifies the user of call rejection.
- UA1 sends an ACK request to P1, which is accepted by P1, causing the 600 response to cease being retransmitted by P1

4.5 Parallel Forking Proxy with multiple 200 OK (P5)

This test is similar to the test described in section 4.3. However, both UA2 and UA3 accept the call. P1 does not send a CANCEL at any time. This causes UA1 to receive two 200 OK's, which should result in one of two situations. Either (1) UA1 sends an ACK to both requests, establishing two call legs, or (2) UA1 accepts one 200 OK with an ACK, and sends a BYE in response to the other 200 OK, establishing a single call leg. This test can only be performed if (1) P1 can be configured not to send a CANCEL upon receiving a 200 OK, (2) UA1 handles multiple 200 OK's.

The test is deemed partially successful if:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- The proxied INVITE from P1 to UA2 and UA3 has an additional Via header, before the Via inserted by UA1, containing its own UDP address
- The requests from P1 to UA2 and UA3 each contain a branch parameter in the Via field, and the value of this parameter is different for each request
- P1 sent a 100 response to UA1 upon receipt of the request
- UA2 sent a 200 response to P1, containing a tag in the To field
- P1 did not send an ACK for the 200 response from UA2
- P1 forwarded the 200 OK upstream to UA1
- UA3 sent a 200 response to P1, containing a tag in the To field
- P1 did not send an ACK for the 200 response from UA3
- P1 forwarded the 200 OK upstream to UA1
- UA1 sent an ACK in response to at least one of the 200 OK's
- UA1 sends an ACK or a BYE in response to the other 200 OK
- At least one call leg is established

The test is fully successful if media is exchanged with one of UA2 or UA3, or both if UA1 accepts both 200 OK's.

4.6 Loop Detection (P6)

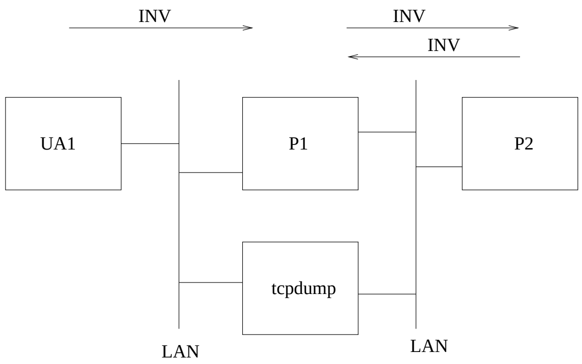

Figure 6: Loop Detection Scenario

This test verifies the ability of a proxy server to perform loop detection. The configuration is shown in Figure 6. There is a single user agent, UA1, and two proxies, P1 and P2. UA1 sends an INVITE directly to P1, containing the address X in both the Request URI and To field. P1 is configured (by means of registrations or otherwise), with a mapping from address X to address Y, corresponding to an address served by P2. The request is therefore forwarded to P2. P2 is configured with a mapping from address Y to address Z, corresponding to an address served by P1. It will either return a loop detected error to P1, or proxy the request, in which case P1 should return a loop detected error to P2.

This test is only possible if proxies P1 and P2 can be configured with mappings which don't require a DNS lookup to resolve the next hop server, but rather can be done statically.

The test is successful if the following occur:

- P1 receives a 482 error response from P2
- P1 sends an ACK to P2, containing a single Via field with its own address
- UA1 receives a 482 error response from P1
- UA1 sends an ACK to P1

4.7 Record Route (P7)

This test examines the ability of UA's and proxies to correctly process packets with Record Route and Route headers. The configuration is identical to the basic proxy test of section 4.1. However, P1 is configured to insert a Record-Route header into the request before proxying it to UA2. UA2 accepts the call, and the 200 OK is forwarded to P1 and then to UA1, with the Record-Route header. When UA1 sends an ACK, it should insert a Route header into the ACK, and send it to P1, which forwards it to UA2. Then, the call is established. UA1 then hangs up by sending a BYE. This BYE should also contain a Route header, and be sent to P1, which should forward it to UA2, terminating the call.

The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- The proxied INVITE from P1 to UA2 has an additional Via header, before the Via inserted by UA1, containing its own UDP address
- The proxied INVITE from P1 to UA2 has a Record-Route header, containing the Request-URI along with its own IP address.
- UA2 sends a 200 OK to P1, containing the Record-Route header mirrored from the INVITE
- P1 forwards the 200 OK to UA1
- UA1 sends an ACK to P1, containing a Route header. The first address in the Route header is the original request URI, plus the IP address of P1 in the maddr tag. If UA2 inserted a Contact header into the 200 OK response, this address is also present as the second Route header in the ACK
- P1 forwards the ACK to UA2, removing the top Route header
- At a later point, UA1 sends a BYE to hang up the call. This BYE contains the same Route headers as the ACK, and is sent to P1
- P1 removes the top Route header and forwards the BYE to UA2
- UA2 sends a 200 OK to P1
- P1 sends a 200 OK to UA1

5 Redirect Server

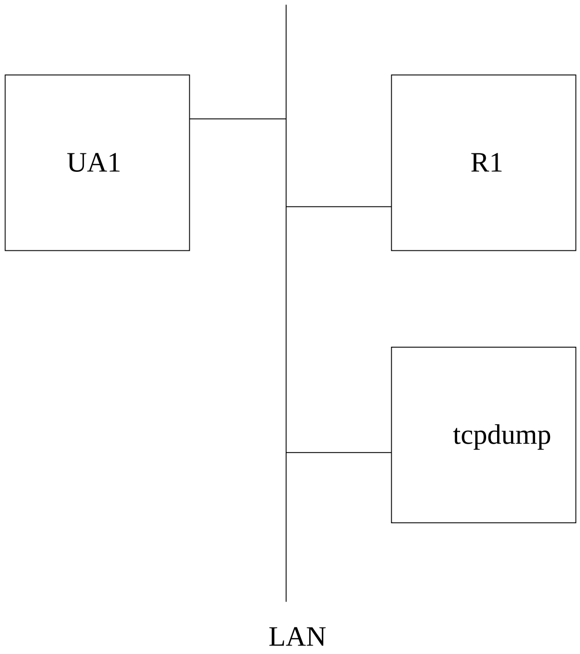

Figure 7: Redirect Configuration

These tests verify basic interoperability between a user agent and a redirect server. The configuration is shown in Figure 7. There is a single user agent, UA1, and a single redirect server, R1.

5.1 Basic Redirection (RED1)

In this test, R1 is aware of at least two addresses mappings, Y1 and Y2, for address X. UA1 sends an INVITE to R1, indicating address X in both the Request URI and To fields. R1 redirects by sending a 300 class response containing Contact headers with Y1 and Y2. R1 can be configured with this mapping through any means, such as by a registration or a static configuration.

The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE

- R1 generates a 300 class response, mirroring the To, Call-ID, CSeq, Via, and From fields. It may insert a tag in the To field
- The 300 class response contains a Contact header with Y1 and Y2 listed
- UA1 sends an ACK to R1, which is accepted by R1, causing response retransmissions to cease.

5.2 Advanced Redirection (RED2)

In this test, R1 redirects to a non-SIP URL; in particular, it contains an http URL. This test is only possible with a redirect server capable of placing non-SIP URL's in redirected responses. All UA's should at least recognize that the URL is not a SIP URL, and do something reasonable (not doing anything is reasonable, but crashing is not).

The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- R1 generates a 300 class response, mirroring the To, Call-ID, CSeq, Via, and From fields. It may insert a tag in the To field
- The 300 class response contains a Contact header with an http URL listed
- UA1 sends an ACK to R1, which is accepted by R1, causing response retransmissions to cease.

Bonus points go to a SIP UA which starts a browser and sends it to the URL listed.

5.3 Redirection with Bodies (RED3)

In this test, a redirect server sends a redirection with no Contact headers, but with a body. The body contains text which is meant to be displayed to the user. This test can only be performed with redirect servers that can insert bodies into responses. All UA's should be able to accept a redirection with a body. They are not required to do anything with it, however.

The test is deemed successful if the following occur:

- The INVITE from UA1 contains a Call-ID, To, Via, From, and CSeq fields. The CSeq field has a method tag of INVITE
- R1 generates a 300 class response, mirroring the To, Call-ID, CSeq, Via, and From fields. It may insert a tag in the To field
- The 300 class response contains no Contact header
- The 300 class response contains a body
- The 300 class response has a Content-Type of text/plain and a valid and correct Content-Length field
- UA1 sends an ACK to R1, which is accepted by R1, causing response retransmissions to cease.

Bonus points go to a UA which causes a browser to display the content.

6 OPTIONS (OPT1)

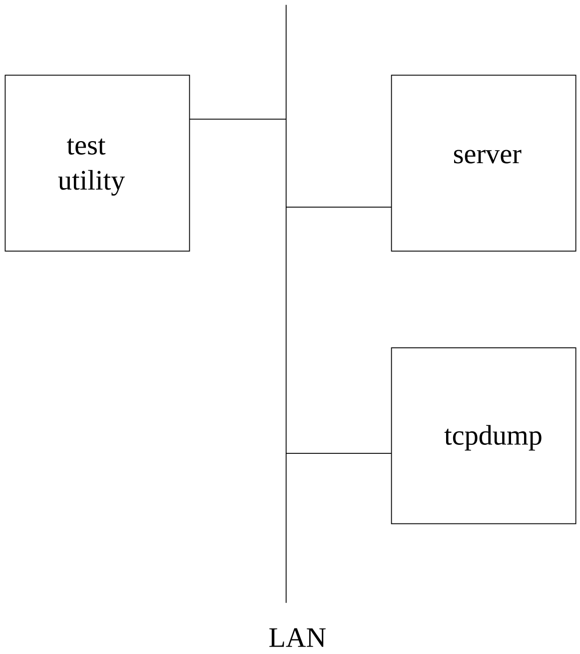

Figure 8: Test Instrument Configuration

This test verifies the ability of a proxy and UA to correctly handle an OPTIONS request. Since we expect few UA's can be forced to generate an OPTIONS request, we use a testing tool to create the OPTIONS message. The configuration is shown in Figure 8. A test utility sends an OPTIONS request to proxy P1. The address used in the To field and Request-URI is X. Proxy P1 has a translation of address X to address Y, corresponding the UA2. P1 therefore proxies the request to UA2. UA2 should respond with a 200 OK, which is forwarded to P1 and then from P1 to the test instrument.

The test is deemed successful if the following occur:

- P1 proxies the OPTIONS message to UA2, after inserting a Via field containing its own address
- UA2 responds with a 200 OK
- The 200 OK contains an Allow header listing the methods understood by UA2
- P1 forwards the 200 OK to the test instrument

Extra credit goes to a UA which does the following additional things:

- The 200 OK contains an SDP body listing the codecs supported by UA2
- The 200 OK contains an Accept listing the body types supported by UA2
- The 200 OK contains an Accept-Encoding listing the encoding types supported by UA2
- The 200 OK contains an Accept-Language listing the languages supported by UA2

7 Parser Correctness

These tests verify correctness parsers in servers. They are all based on the configuration in Figure 8. A test utility, capable of sending SIP messages, sends a request to a specific IP address, corresponding to a server (proxy, redirect, or user agent). A tcpdump utility listens on the network connection and prints all messages sent.

The tests complement the parsing which must be functional in the previous sections. They test features which normally cannot be controlled from a user interface, and therefore are difficult to check without an explicit test instrument.

Each test is successful if the message is accepted by the server. This can be determined based on the type server:

Proxy Server The request is proxied

Redirect Server A redirect response is received

User Agent Server The user is alerted

The test is unsuccessful if the server crashes, returns an error, or fails to take the above action. Furthermore, the proxied or redirected message must also be properly formatted.

7.1 name-addr form (PAR1)

SIP allows for a name-addr form in To, From, and Contact, which looks like the following:

```
From: Jonathan Rosenberg <sip:jdrosen@bell-labs.com>
```

or

```
From: sip:jdrosen@bell-labs.com
```

To test this, the test utility sends the following request to the SIP server:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE
Content-Length: 0
```

A proxy which proxies this request should forward it with the full name-addr form in tact. A redirect should mirror the To and From fields.

7.2 line-folding (PAR2)

Line folding improves readability of messages. Long lines can be folded by adding LWS after a CRLF. This indicates to the parser that the next line is not a new header. This characteristic can be tested by sending the following message:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee
 <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE
Content-Length: 0
```

Since the To field is folded, a parser which cannot perform line folding will reject the message as it will think the To field is missing.

7.3 LWS (PAR3)

Extra LWS can be inserted in a variety of places in the messages. Typically, it is allowed between almost all tokens. The following message tests this:

```
INVITE sip:user@domain SIP/2.0
Via: SIP / 2.0 / UDP host.test.com  ;   branch=0   ,
  SIP/2.0/TCP   host2.test2.com    ; branch=1
From   :   Caller  <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0    INVITE
Content-Length: 0
```

7.4 case insensitivity (PAR4)

SIP message headers should be parsed with case-insensitivity. The following message checks this:

```
INVITE sip:user@domain SIP/2.0
VIA: SIP/2.0/UDP host.test.com
```

```
from: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
CaLl-Id: 86888aab87@10.0.1.1
CSEq: 0 INVITE
COnTeNt-length: 0
```

7.5 Compact Form (PAR5)

SIP headers can be sent using compact form, which requires less space. The following message tests this:

```
INVITE sip:user@domain SIP/2.0
v: SIP/2.0/UDP host.test.com
From: sip:caller@domain.com
t: sip:callee@domain
i: 86888aab87@10.0.1.1
CSeq: 0 INVITE
l: 0
```

7.6 Multiple UDP requests per packet (PAR6)

SIP allows multiple UDP requests in a single packet. This is verified with the following message. A proxy should respond by proxying two separate requests. A redirect server should respond with two separate redirects, and a UAS should act as if it received two separate call requests:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE
Content-Length: 0


INVITE sip:user2@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee 2 <sip:callee2@domain>
Call-ID: 088bbcfb87@10.0.1.1
CSeq: 1 INVITE
```

```
Content-Length: 10

0123456789
```

7.7 Missing Content-Length in UDP (PAR7)

A server should correctly process a UDP request with a missing Content-Length. In fact, a proxy should fill in the value and send it in the outgoing request. The following request tests this:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE

01233456789
```

7.8 Unknown header fields (PAR8)

A server should ignore header fields it does not understand. This request contains extra headers:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
NewHeader: newvalue
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE
Content-Length: 0
```

8 UDP Reliability

This section verifies that the reliability mechanisms for UDP are implemented correctly.

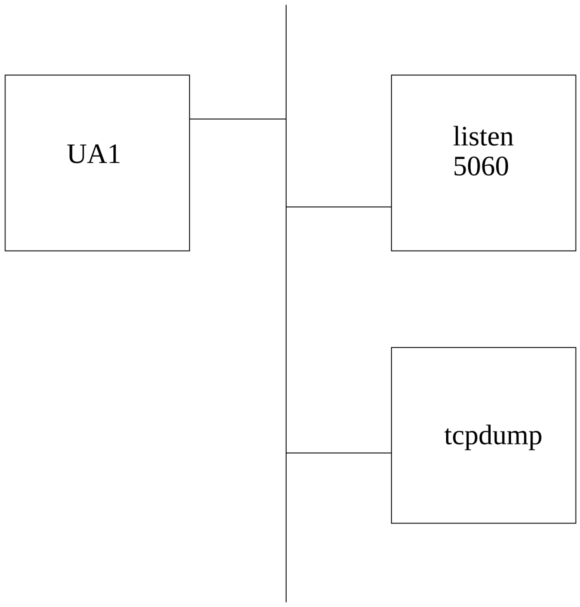

Figure 9: INVITE reliability configuration

8.1 INVITE request retransmission (REL1)

In this configuration, as shown in Figure 9, a single UA sends a request to an address where a test utility is listening on the SIP port. This implies that no ICMP messages are sent, but no responses are generated. The UA should retransmit the request at intervals with an exponential back-off, eventually timing out. The intervals should start at 500ms, and double each time until a total of seven packets have been sent. The test is deemed a success if the following occur:

- The UAC sends the INVITE seven times
- The interval between the INVITE requests starts at 500ms, and doubles after each transmission
- The UAC reports an error to the user after transmitting the request seven times

8.2 BYE request retransmission (REL2)

This configuration is identical to that of section 2. The call is setup between UA1 and UA2. Mid-call, UA2 is terminated. UA2 should not send any messages when terminated; this may require a non-graceful termination. In its place, a test utility is set to listen on port 5060 on the same machine UA2 was running. At this point, UA1 attempts to hang up the call. This should cause a BYE to be sent to UA2. The BYE should be retransmitted periodically. The test is deemed successful if the following occur:

- UA1 sends a BYE to UA2
- The BYE is retransmitted eleven times
- The interval between retransmissions starts at 500ms, and the doubles to 1s, to 2s, and to 4s, and then remains at 4s

8.3 INVITE/100 retransmission (REL3)

This configuration is identical to that of section 2. It verifies that a UAC will not retransmit an INVITE at all once a provisional response has been received. UA1 initiates a call to UA2. However, UA2 never answers the call. UA2 should send a provisional response to UA1 upon receiving the INVITE, causing request retransmissions to cease. The test is deemed successful if the following occur:

- UA2 sends a provisional response to UA1
- UA1 never retransmits the request after receiving the provisional response

8.4 INVITE response retransmission (REL4)

This configuration is identical to that of section 2. It verifies that a UAS retransmits a response to an INVITE with an exponential backoff in the retransmit interval. To perform this test, UA1 sends an INVITE to UA2. UA2 then sends a provisional response to UA1. At this point, UA1 is terminated. In its place, a test utility that listens on 5060 (or whatever port was indicated in the Via header in the request). Then, UA2 accepts the call. The 200 OK should be retransmitted by UA2. The test is deemed successful if the following occur:

- UA2 sends a 200 OK response

- The 200 OK response is retransmitted 7 times
- The interval between response retransmissions starts at 500ms and doubles after every transmission

9 Security

This section checks interoperability of the basic and digest security mechanisms.

9.1 Basic for REGISTER (SEC1)

The configuration for this test is shown in Figure 2. A single user agent, UA1, is configured to use registrar R1 for its registrations. Upon startup (or at some later point), it sends a REGISTER message to R1. R1 is configured to require basic authentication before accepting registrations. It therefore returns a 401 Unauthorized response to UA1. UA1 should prompt the user for a name and password. UA1 then resubmits the request with these credentials. They should be accepted by R1, and a 200 OK response returned. R1 must be preconfigured with the name and password in order for this test to work.

The test is deemed successful if the following occur:

- UA1 sends a REGISTER to R1
- R1 responds to the REGISTER with a 401 response
- The 401 response contains a WWW-Authenticate header, containing the scheme basic and a realm
- UA1 prompts the user for a name and password, and indicates the realm in the dialog
- UA1 resubmits the REGISTER request, with an incremented CSeq value and an Authorization header
- R1 accepts the REGISTER request, and returns a 200 OK

9.2 Digest for REGISTER (SEC2)

The configuration for this test is identical to 9.1. However, instead of basic authentication, digest authentication is done. The test is deemed successful if the following occur:

- UA1 sends a REGISTER to R1
- R1 responds to the REGISTER with a 401 response
- The 401 response contains a WWW-Authenticate header, containing the scheme digest, realm, and a nonce
- UA1 prompts the user for a name and password, and indicates the realm in the dialog
- UA1 resubmits the REGISTER request, with an incremented CSeq value and an Authorization header
- R1 accepts the REGISTER request, and returns a 200 OK

9.3 Basic Proxy Authorization for INVITE (SEC3)

This tests interoperability of the basic proxy authorization mechanism. The configuration is identical to that of section 4.1. However, when UA1 sends an INVITE to P1, P1 requires proxy authorization. It should return a 407 Proxy Authorization Required response to UA1. UA1 should prompt the user for a name and password, and resubmit the request. The request should be accepted, and proxied to UA2, causing the call to be setup.

This test requires P1 to be configured to use basic authentication for proxied requests, and to know the name and password of UA1.

The test is deemed successful if the following occur:

- UA1 sends an INVITE to P1
- P1 sends a 407 response to UA1
- The 407 response contains a Proxy-Authenticate header, indicating the scheme basic and a realm
- UA1 prompts the user for a name and password, and displays the realm to the user
- UA1 resubmits the INVITE request, increasing the CSeq number, and including a Proxy-Authorization header indicating basic and containing its credentials
- P1 accepts the INVITE request, and proxies it to UA2
- UA2 accepts the call and sends a 200 OK to P1

- P1 forwards the 200 OK to UA1
- UA1 sends an ACK

9.4 Digest Proxy Authorization for INVITE (SEC4)

This tests interoperability of the digest proxy authorization mechanism. The configuration is identical to that of section 4.1. However, when UA1 sends an INVITE to P1, P1 requires proxy authorization. It should return a 407 Proxy Authorization Required response to UA1. UA1 should prompt the user for a name and password, and resubmit the request. The request should be accepted, and proxied to UA2, causing the call to be setup.

This test requires P1 to be configured to use digest authentication for proxied requests, and to know the name and password of UA1.

The test is deemed successful if the following occur:

- UA1 sends an INVITE to P1
- P1 sends a 407 response to UA1
- The 407 response contains a Proxy-Authenticate header, indicating the scheme digest, a realm, and a nonce
- UA1 prompts the user for a name and password, and displays the realm to the user
- UA1 resubmits the INVITE request, increasing the CSeq number, and including a Proxy-Authorization header indicating digest and containing its credentials
- P1 accepts the INVITE request, and proxies it to UA2
- UA2 accepts the call and sends a 200 OK to P1
- P1 forwards the 200 OK to UA1
- UA1 sends an ACK

10 Miscellaneous

The following verifies some miscellaneous functions in SIP servers.

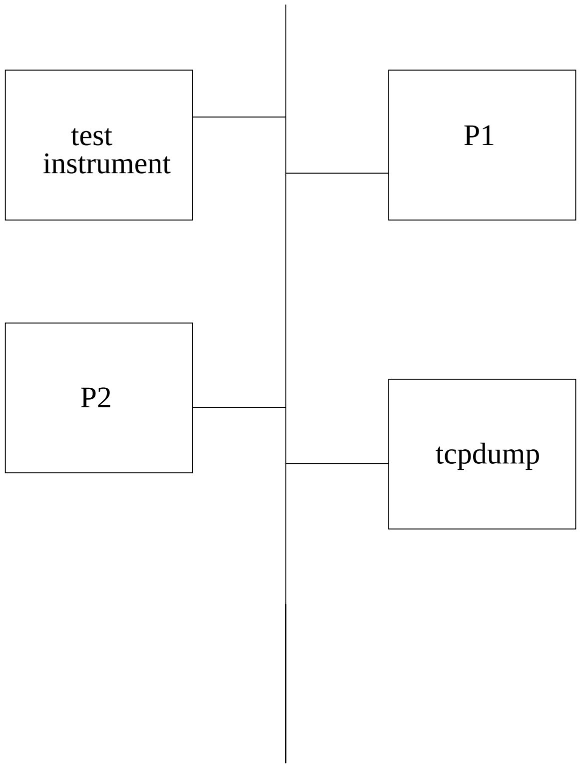

Figure 10: Max-Forwards Configuration

10.1 Max-Forwards Expiration (MISC1)

A proxy server is supposed to decrement the Max-Forwards value, when present, in a request. Should the value become zero, it is supposed to return an error. This function tests this. The configuration is shown in Figure 10. There is a single test instrument, capable of sending an INVITE request. It is sent to P1. The request contains an address X in the Request-URI and To field, which are known to P1, and cause the request to be proxied to P2. The request is sent with a Max-Forwards value of 1. This causes it to be decremented at P1 to 0, and then at P2, should cause an error to be returned.

The message that should be used by the test instrument is:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE
Max-Forwards: 1
Content-Length: 0
```

The test is deemed successful if the following occur:

- The message is received by P1
- P1 forwards the message to P2, after decrementing the value of Max-Forwards by 1, and adding a Via field
- P2 receives the message, and generates a 483 response, and sends it to P1
- P1 sends the 483 response to the test instrument

10.2 Requires (MISC2)

The Require header is used to allow new features to be added to SIP. A UAS which receives a request with a Require header containing an extension it does not understand, is supposed to return an error, and list those extensions it does understand. The test configuration is shown in Figure 8. There is a test instrument and a single UAS, UA1. The test instrument sends the following request to UA1:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
Require: foo
CSeq: 0 INVITE
Content-Length: 0
```

The test is deemed successful if the following occur:

- The message is received by UA1
- UA1 responds with a 420 response
- The 420 response contains an Unsupported header, indicating "foo" as unsupported

10.3 Proxy-Requires (MISC3)

The Proxy-Requires header is used for a similar purpose as the Requires header, but is for proxies. The test is nearly identical to that of section 10.2. However, the server is a proxy server instead of a user agent server. The following message should be sent from the test instrument:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
Proxy-Require: foo
CSeq: 0 INVITE
Content-Length: 0
```

The test is deemed successful if the following occur:

- The message is received by P1
- UA1 responds with a 420 response
- The 420 response contains an Unsupported header, indicating "foo" as unsupported

10.4 Non SDP Payloads (MISC4)

A UAS which does not understand a body is supposed to return an error to the client indicating such. The configuration for testing this is shown in Figure 8. The test instrument sends the following message to UA1:

```
INVITE sip:user@domain SIP/2.0
Via: SIP/2.0/UDP host.test.com
From: Caller <sip:caller@domain.com>
To: Callee <sip:callee@domain>
Call-ID: 86888aab87@10.0.1.1
CSeq: 0 INVITE
Content-Length: 10
Content-Type: text/newformat

0123456789
```

The test is deemed successful if the following occur:

- The message is received by UA1
- UA1 responds with a 415 message, containing an Accept, Accept-Encoding, and Accept-Language header

References

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments (Proposed Standard) 2543, Internet Engineering Task Force, Mar. 1999.