



Understanding SIP Servers

© 2002 RADVISION Ltd.

NOTICE

© 2002 RADVISION Ltd. All intellectual property rights in this publication are owned by RADVISION Ltd. and are protected by United States copyright laws, other applicable copyright laws and international treaty provisions. RADVISION Ltd. retains all rights not expressly granted.

No part of this publication may be reproduced in any form whatsoever or used to make any derivative work without prior written approval by RADVISION Ltd.

No representation of warranties for fitness for any purpose other than what is specifically mentioned in this guide is made either by RADVISION Ltd. or its agents.

RADVISION Ltd. reserves the right to revise this publication and make changes without obligation to notify any person of such revisions or changes. RADVISION Ltd. may make improvements or changes in the product(s) and/or the program(s) described in this documentation at any time.

If there is any software on removable media described in this publication, it is furnished under a license agreement included with the product as a separate document. If you are unable to locate a copy, please contact RADVISION Ltd. and a copy will be provided to you.

Unless otherwise indicated, RADVISION registered trademarks are registered in the United States and other territories. All registered trademarks recognized.

For further information contact RADVISION or your local distributor or reseller.

SIP Server Toolkit version 1.0, August, 2002

Publication 1

<http://www.radvision.com>

CONTENTS

Introduction	1
What is a SIP Server?	1
Register Server	2
Redirect Server	3
Proxy Server	6
Stateful and Stateless Proxies	6
Request Validation	9
Address Resolution	11
Determining the Target-set	11
DNS Resolution	12
Stateful Message Forwarding	12
CANCEL	16
Record-Routing	17
Loose Routing	18
Recursion on 3xx Responses	19
Forking	20
Authentication	23
Loop detection and Max Forwards	24
Message Spiraling	25
Outbound Proxy	26
Policy	26
RADVISION SIP Server Toolkit	27
SIP Server Toolkit Package	28
Operating System Support	28
RADVISION Family of SIP Development Solutions	28

UNDERSTANDING SIP SERVERS

INTRODUCTION

SIP Servers are essential network elements that enable SIP endpoints to exchange messages, register user location, and seamlessly move between networks. SIP Servers enable network operators to install routing and security policies, authenticate users and manage user locations.

SIP Server applications may take many forms, but the SIP standard defines three general types of server functionality that apply to all—proxy, redirect and registrar servers. These standard functionalities can be used according to the needs of the specific implementation.

With the advance of SIP, server logic has become increasingly complex. SIP Servers need to deal with varying network topologies (such as public Internet networks, cellular networks, broadband residential networks), complex routing policies, security and SIP extensions. SIP Servers often need to handle high message/transaction rates and yield real-time performance and scalability, high throughput, and low delay. This chapter discusses the protocol aspects of SIP Server behavior and the usage of the RADVISION SIP Server Toolkit to address the challenges of effective SIP Server development.

WHAT IS A SIP SERVER?

The SIP baseline specification RFC3261 (previously RFC2543bis) divides SIP Server functionality into the following three parts:

- SIP Registrar Server—handles location registration messages.
- SIP Redirect Server—returns “contact this address” responses.
- SIP Proxy Server—forwards SIP requests and responses.

The authors of SIP have made this functional separation for ease of understanding, and not necessarily as a guideline for implementers. In other words, it is acceptable to have one “box” do more than just one type of server functionality. For example, a SIP Server may benefit from having a routing capability to forward some SIP messages as a proxy and redirect others as a redirect server. There are existing SIP Server implementations that have all three types of functionality in one product running in the same process space, while others prefer to distribute functionality, with different machines performing different server functions. In fact, the SIP standard does not prohibit a SIP user agent (endpoint) from having server capabilities, since user agents may redirect

requests or process registrations. It is important to understand that proxy, redirect and registrar are functionalities that may be put to use in any way that benefits the application.

The behavior of SIP Servers is covered in the RFC2543-bis and its subsequent RFC 3261 standard, but you can find more detailed examples of message flows in “SIP Call Flow Examples” in the document *draft-ietf-sipping-call-flows-00.txt*, found at the IETF site or one of its mirror sites.

Note The document version number may change as newer versions are published.

REGISTRAR SERVER

The SIP standard defines a registrar server as “a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles”. REGISTER requests are generated by clients in order to establish or remove a mapping between their externally known SIP address(es) and the address(es) they wish to be contacted at. The REGISTER request can also be used to retrieve all the existing mappings saved for a specific address.

The Registrar processes the REGISTER request for a specific set of domains. It uses a “location service”—an abstract location database—in order to store and retrieve location information. The location service may run on a remote machine and may be contacted using any appropriate protocol (such as LDAP). The SIP standard leaves this decision to the implementation. Some implementations may co-locate the location service and the registrar server on the same machine.

A registrar server may authenticate incoming REGISTER requests using the 401 (Unauthenticated) response.

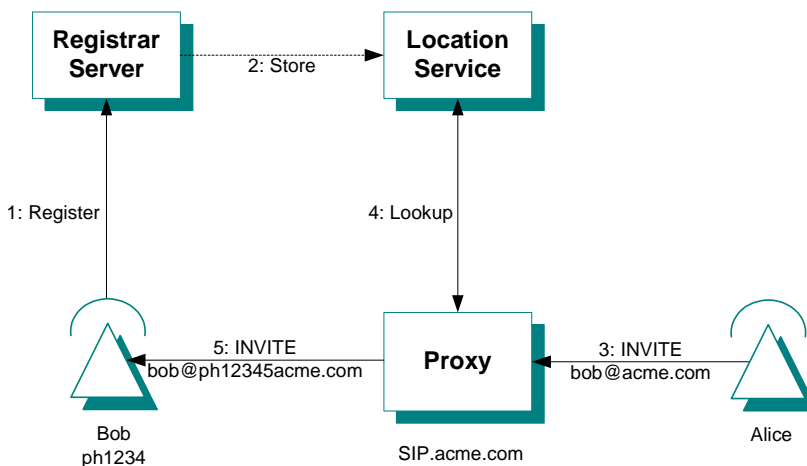


Figure 1-1 Registration Process

REDIRECT SERVER

Redirect server functionality is the simplest of the three functionalities. A redirect server receives SIP requests and responds with 3xx (redirection) responses, directing the client to contact an alternate set of SIP addresses. The alternate addresses are returned as Contact headers in the response message.

Table 1-1 shows the 3xx responses that are currently defined by SIP.

Table 1-1 *3xx Responses*

Response	Meaning
300 Multiple Choices	The address in the request was resolved to several choices, each with its own specific location, and the user (or user agent) can select a preferred communication end point and redirect its request to that location. This status response is appropriate if the callee can be reached at several different locations and the server cannot, or prefers not, to proxy the request Note: 301 and 302 responses can also contain multiple Contact address. The difference is that they convey a more specific reason for the redirection.
301 Moved Permanently	The user can no longer be found at the address specified in the Request-URI (the destination address in the request), and the requesting client should retry at the new address given by the Contact header field.
302 Moved Temporarily	The user is temporarily available at a different address(es). The duration of validity of these addresses may be expressed in the Contact header.
305 Use Proxy	The requested destination address must be accessed through the proxy specified in the Contact field.
380 Alternative Service	The call was not successful, but alternative services are possible. The alternative services are described in the message body of the response. The use of this response code is still not defined in SIP and is for future use.

In most cases, 301 and 302 responses are used by redirect servers. 300 can also be used, although it is more ambiguous to the calling client.

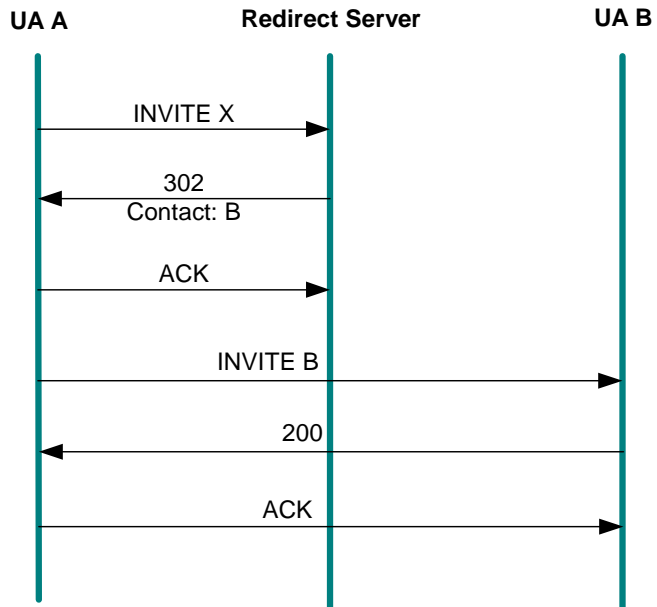


Figure 1-2 Request Redirection

The scenario in [Figure 1-2](#) illustrates a redirection scenario. Note that the second INVITE request is generated with the same dialog identifiers, Call-ID, and To and From headers as the first INVITE request, but with a different CSeq value.

Redirection allows servers to push back routing information for a request in a response to the client, thereby aiding in locating the target of the request, while taking themselves out of the loop of further messaging for this transaction. Redirect servers typically are not aware of the state of dialogs (calls, subscriptions), only of the state of the individual transactions they are handling, making them transaction-stateful elements. Redirection is designed as a simple and quick procedure, allowing for redirect servers to be highly scalable and to yield high-performance. Redirect servers are sometimes used as a load balancing devices.

Redirect servers may request user authentication with the use of the 407 response as explained below.

PROXY SERVER

The SIP standard defines SIP proxies as “elements that route SIP requests to user agent servers (UAS) and SIP responses to user agent clients (UAC). A request may traverse several proxies on its way to a UAS. Each will make routing decisions, modifying the request before forwarding it to the next element. Responses will route through the same set of proxies traversed by the request in the reverse order.”

It is useful to view Proxy Servers as SIP-level routers that forward SIP requests and responses. However SIP proxies employ routing logic that is typically more sophisticated than just automatically forwarding messages based on a routing table. The SIP standard allows proxies to perform actions such as validate requests, authenticate users, fork requests, resolve addresses, cancel pending calls, Record-Route and Loose-Route, and detect and handle loops. The versatility of SIP proxies allows the operator/system administrator to use the proxies for different purposes and in different locations in the network (such as edge proxy, core proxy and enterprise proxy). This versatility also allows for the creation of a variety of proxy policies, such as routing calls only for authenticated users that have no standing debt to the network service provider operating the proxy. Proxies can be placed at the network of the service provider or at the enterprise or SOHO premises. The 3GPP IMS architecture, for example, uses proxies known as Call State Control Functions (CSCF in 3GPP terminology) of different kinds for various purposes—as the first hop server that communicates with the handset; as the element that accesses location repositories and application servers and triggers services; and as the edge element that communicates with proxies in foreign networks.

A proxy server is designed to be mostly transparent to user agents. Proxy servers are allowed to change messages only in specific and limited ways. For example, a proxy is not allowed to modify the SDP body of an INVITE. Apart from a few exceptions, proxies cannot generate requests at their own initiative. Therefore a proxy cannot terminate an existing call by generating a BYE request.

STATEFUL AND STATELESS PROXIES

The SIP specification defines two types of SIP proxies:

- Stateful proxy
- Stateless proxy

STATELESS PROXY

A stateless proxy is a “simple message forwarder”, as described in the SIP standard. When receiving a request, the stateless proxy processes the request much like a stateful proxy, however the stateless proxy forwards the message in a stateless fashion—without saving any transaction context. This means that

once the message is forwarded the proxy “forgets” ever handling this message. Stateless forwarding allows for improved performance and scalability, but has some consequences:

- A stateless proxy cannot associate responses with forwarded requests because it retains no knowledge of the requests it has forwarded. Therefore, the proxy application cannot know if a transaction was successful or not.
- A stateless proxy cannot associate retransmissions of requests and responses with the previous instance of these messages. It processes retransmissions exactly as if this is the first copy of the message it received.

Note Stateless proxies need to use a routing algorithm that always routes copies of the same message to the same destination. If this rule is not enforced, a retransmission of an ACK message, for example, may be routed to a different user agent than the one that returned the 200 response.

- If the message is lost, the proxy will not retransmit it. Retransmission is the responsibility of stateful user agents or proxies

Because of their high-throughput capabilities, stateless proxies are often used at the core of carrier and service provider networks assisting in forwarding SIP messages on the network. Stateless proxies may also be used as load balancers.

Note that once a proxy decides to return a non-2xx response, the proxy cannot do this statelessly and has to retain a transaction state.

STATEFUL PROXY

When stateful, the proxy processes transactions rather than individual messages. The proxy manages two types of transactions—server transactions to receive requests and return responses, and client transactions to send requests and receive responses. An incoming request is processed by a server transaction and then forwarded downstream by one or more client transactions (there may be more than one in the case of parallel forking, for example). An incoming response is received by the matching client transaction and forwarded back to the server transaction. Associating between client and server transactions and managing the overall state of this request is the responsibility of the proxy core object. The proxy core object chooses the destination address(es) and instantiates one or

more client transaction objects accordingly. The proxy core object also collects the responses from the different client transactions and chooses the response(s) that will be sent upstream via the server transaction.

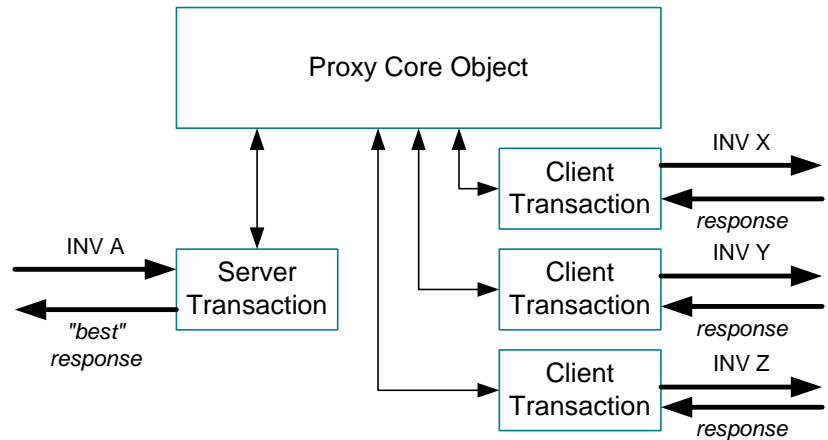


Figure 1-3 Stateful Proxy Model

A stateful proxy is aware of the state of transactions and message history, and can therefore perform better-informed processing on incoming messages. For example, a stateful proxy can identify a retransmission of an incoming message and forward the message only in situations that require retransmission forwarding, whereas a stateless proxy cannot identify retransmissions and has to forward every message it encounters. A stateful proxy can also generate retransmissions in cases of message loss. In addition, a stateful proxy can locally process incoming CANCEL requests and generate CANCEL requests as needed. Forking is also more natural for a stateful proxy (especially sequential forking).

Statefulness, however, has the following drawbacks:

- **Memory consumption**

The stateful proxy retains more memory per processed message than a stateless proxy, and for a longer duration. This has a negative impact on the maximal capacity of the proxy and limits the number of concurrent calls/transactions it can handle.

Certain code optimizations which are SIP-specific can compensate for this and bring memory consumption to the level required by high-capacity proxies.

- **Throughput**

A stateful proxy has to spend more CPU cycles on message processing—mapping messages to transactions, managing transaction state machines, processing transaction timers and associating client and server transactions. This extra processing reduces proxy capacity in terms of performance (maximal number of processed requests per second). For these reasons, developing a high-performance stateful proxy has proved to be non-trivial, and requires special optimizations in the design of the proxy. The ability to customize and to fine tune the proxy through flexible configuration is also crucial for achieving high performance.

- **Implementation complexity**

A stateful proxy does more than just forward requests. Certain logic needs to be employed in order to deal with actions such as parallel forking (for example, choosing the best response), CANCEL, recursion on 3xx responses, and handling ACK for 2xx responses. The evolution of the SIP standard introduced numerous “hard-to-deal-with” special cases for proxies that require the use of a number of special techniques, such as storing hashed context information in certain request message fields and later retrieving the information from responses. All of these factors make the implementation of the proxy non-trivial and add special cases that have to be tested.

- **Underlying SIP Stack complexity**

A proxy requires certain flexibilities from the underlying SIP Stack that a user agent does not. This is especially true for the transport and transaction layers. This requires the SIP Stack to be more modular and export more layers of API than a user agent-oriented stack.

REQUEST VALIDATION

Before routing a request, a SIP Server (proxy or redirect) needs to validate the request to make sure it can actually proceed with processing this message. The message has to pass the following validity checks:

- **Reasonable syntax check**

The request must be well-formed enough to be handled by the server. However, this applies only to specific fields in the message which the server must process. All other parts should not be checked or fixed by the proxy.

- **URI scheme check**

The URI scheme (for example, “ftp” in ftp:radvision.com) must be URI scheme the proxy understands and knows how to route. If not, the proxy must return a 416 (Unsupported URI Scheme) response.

- **Max-Forwards check**

Max-Forwards is a message field that indicates how many more hops the message is allowed to traverse. Each proxy that handles the message decrements this number by one (similar to the TTL field in certain protocols). If the message contains a Max-Forwards value of zero, the proxy must return a 483 (Too many hops) response. This mechanism allows preventing a message from going into an endless loop between a set of proxies.

- **Loop Detection (Optional)**

A proxy may check for loops by executing a loop detection algorithm on the Via list contained in the message. The proxy checks that it did not previously handle this message. If it did previously handle the message, it verifies that the message contains different values in the fields that influence routing decisions (such as Request-URI, From and To). If the proxy identifies a loop condition, it rejects the message with a 482 response code (Loop Detected). This check is now optional because the Max-Forwards field has become mandatory.

- **Proxy-Require**

The client may indicate certain SIP extensions in the Proxy-Require fields that the proxy must support in order to successfully handle this request. The proxy must inspect this field and verify that it supports all the extensions listed in the field.

- **Authentication**

If the SIP Server determines it has to authenticate the originator of the message, it has to make sure the message contains credentials that authenticate the user. If the message does not contain credentials or the credentials failed to authenticate the user, the proxy may return a 407 response containing a challenge. The authentication procedure is explained in more detail in *Authentication* on page 23.

ADDRESS RESOLUTION

Once a proxy has validated an incoming request and decided to forward it, it must determine the destination(s) to which the message is to be forwarded before sending the messages. The proxy does two types of address resolution:

- Determining the target-set—the proxy resolves the request SIP destination address (Request URI) to a set of SIP addresses that are mapped to it in some way.
- DNS resolution—the proxy resolves each of the SIP destination addresses to a transport address of the form:
{transport_protocol, IP address, port}

Note A SIP request may be forwarded to more than one destination address. An example of such a case is a user that is simultaneously registered at several locations. In this case, the proxy may decide to fork the request either sequentially or in parallel.

DETERMINING THE TARGET-SET

The first process in address resolution, known as obtaining a target-set in the SIP specification, results in producing a set of SIP addresses. Essentially this stage maps from SIP address to SIP addresses. (Some addresses in the target-set may express explicit transport addresses by using the format sip:user@ip_address;transport=xyz.)

A target-set is obtained in one of two ways:

- **Predefined target-set**

This is the simpler case, where the destination address of the request (Request-URI) is such that the proxy must automatically forward to the destination address without trying to resolve to other addresses. One such case is where the request-URI is in a domain for which the SIP Server is not responsible. For example, a proxy sip:proxy1.acme.com which

is responsible for the domain acme.com receiving a request for sip:bob@example.com must proxy the request to sip:bob@example.com.

- **Target-set determined by proxy**

If the target-set is not dictated by the message, the proxy may employ whatever mechanism it may wish to determine the target-set. Some options are:

- Accessing a location service updated by a SIP registrar
- Reading from a database
- Consulting a presence server
- Using other protocols
- Performing algorithmic substitutions on the destination address

While the Request-URI is an important factor in determining the target set, the proxy may also choose to route based on other message fields, or on external parameters, such as time of day, network and server load.

DNS RESOLUTION

Before forwarding a message, the proxy must resolve the message to concrete transport addresses which it can use in sending the message. Proxies, as other SIP entities, use the DNS mechanism described in RFC 3263 (SIP: Locating SIP Servers). Essentially, this is an algorithm that selectively uses SRV, NAPTR, A and AAAA DNS queries to map a given SIP address to a prioritized set of transport addresses of the form:

{transport_protocol, IP_address, port}

Using this advanced DNS scheme is recommended, since it allows building highly available, load-balanced SIP networks with the possibility of dynamic adjustment through DNS tables.

STATEFUL MESSAGE FORWARDING

The message flows below illustrate the way a stateful SIP proxy forwards different types of messages. The term “downstream” means in the direction of the server (request direction); “upstream” means in the direction of the client (response direction). For the sake of simplicity, the scenarios show only non-forking proxies that Record-Route.

NON-INVITE REQUEST

The scenario in [Figure 1-4](#) shows Non-INVITE Request-Response message flow through multiple proxies.

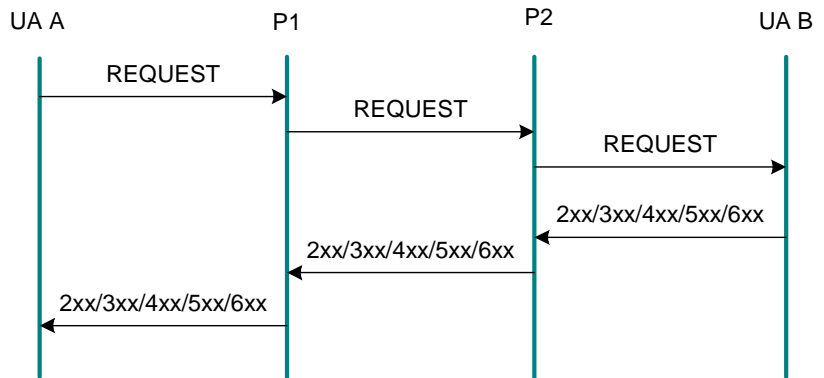


Figure 1-4 Non-INVITE Request-Response Message Flow

In the case of non-INVITE requests, such as BYE and REGISTER, proxy functionality is to forward requests and responses as they arrive. The proxy processes all final responses (2xx-6xx) the same way. Retransmitted requests are not forwarded by the proxy, but the proxy may retransmit requests based on its own retransmission timers. Retransmitted responses are forwarded by the proxy.

INVITE-ACK

When processing an INVITE request, a proxy typically responds with a 100 (Trying) response to stop INVITE retransmissions at the previous hop. All received 1xx (provisional) responses except 100 are forwarded to the previous hop. If the proxy does not receive a 100, it may retransmit the INVITE request as necessary.

The scenario in [Figure 1-5](#) shows INVITE-non-2xx-ACK message flow through multiple proxies

Proxy Server

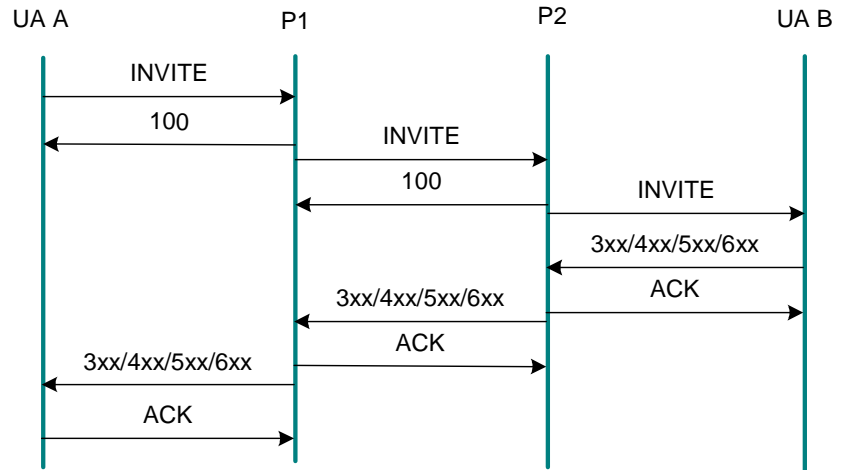


Figure 1-5 *INVITE-non-2xx-ACK Message Flow*

If a non-2xx response is received, the proxy generates the ACK request and forwards the response upstream. Upon reception of a retransmission of the response, the proxy will retransmit the ACK request.

INVITE-200-ACK

The scenario in [Figure 1-6](#) shows INVITE-200-ACK message flow through multiple proxies.

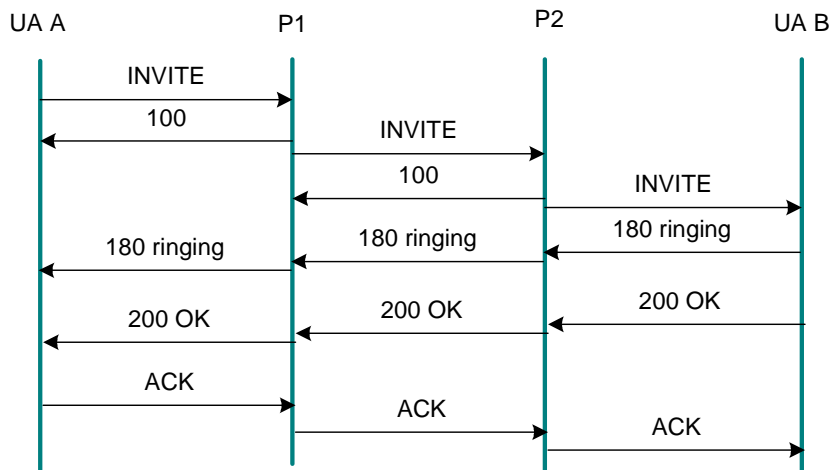


Figure 1-6 INVITE-200-ACK Message flow

A 2xx response to an INVITE request presents a special case. For purposes of call-setup robustness, it was decided that reliability (retransmissions) of 200 and ACK messages is to be handled end-to-end, rather than hop-by-hop. This means that upon receiving a 2xx response for an INVITE, the proxy forwards the message (and possibly any retransmission that follows) in a stateless fashion. It does not change state in any of its transactions and it does not generate an ACK request. Only the calling user agent (UA A in the figure above) is allowed to ACK a 2xx response. The proxies forward the ACK (and possible retransmission) also in a stateless fashion. If either the 2xx or the ACK messages are lost along the way, it is the responsibility of the callee (UA B in the drawing) to retransmit the 2xx until it receives an ACK. This procedure assures that a call is established (and media can start flowing) only when both user agents have completed the handshake.

CANCEL

The scenario in *Figure 1-7* shows CANCEL processing.

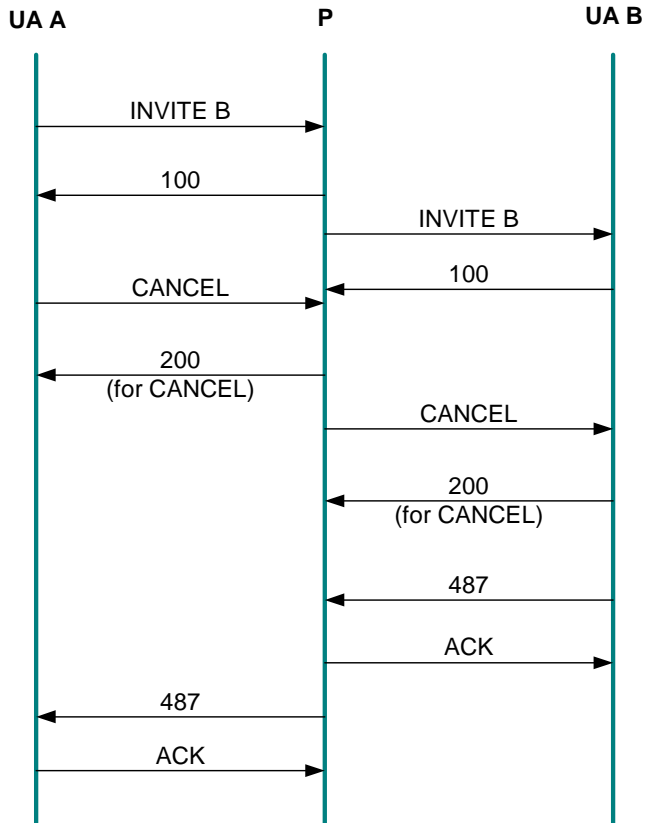


Figure 1-7 CANCEL Processing

A stateful proxy may choose to generate a CANCEL request for any pending INVITE request it has previously forwarded (subject to the rules of CANCEL as defined by the SIP standard). This capability is put to use in parallel forking as explained later in *Parallel Forking* on page 20.

A proxy receiving a CANCEL request must try and match it to an existing INVITE context (proxy core object) and cancel any pending client transactions associated with this INVITE, as illustrated in *Figure 1-7* above. If an INVITE context is not found, the proxy must statelessly forward the CANCEL request (it may have statelessly forwarded the associated INVITE previously).

RECORD-ROUTING

The scenario in *Figure 1-8* shows message proxying without Record-Routing.

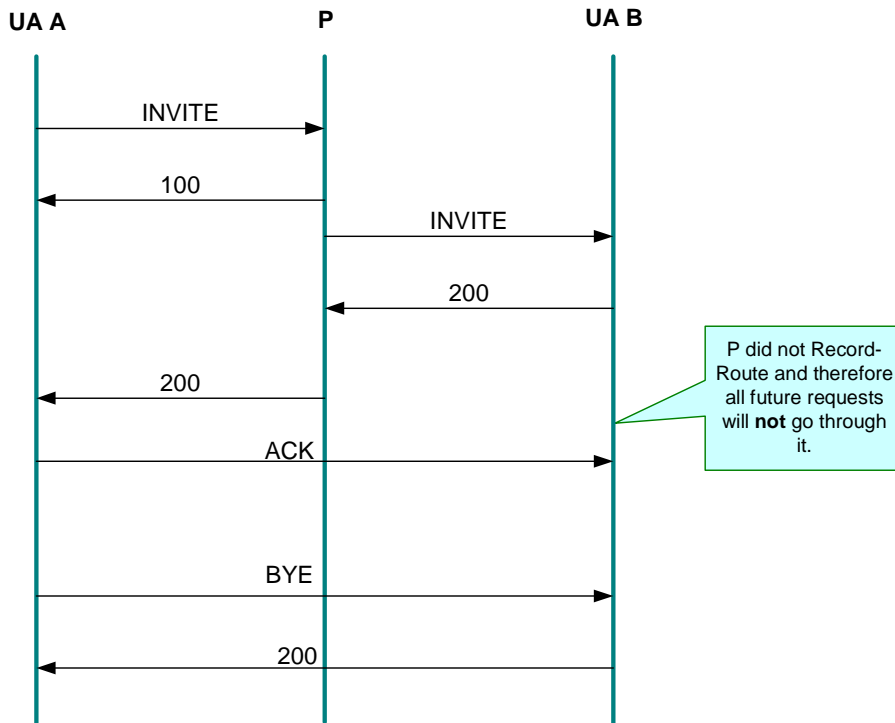


Figure 1-8 Message Proxying without Record-Routing

Record-Routing is a SIP mechanism that allows SIP proxies to request being in the signaling path of all future requests that belong to this dialog. A proxy Record-Routes by entering the Record-Route header into the initial dialog-establishing request (currently, INVITE and SUBSCRIBE). The UAS and UAC build route-lists based on the Record-Route headers they find in the request and send each subsequent request with the route list as a set of Route headers.

A proxy that does not Record-Route an INVITE message should not expect to receive any of the following requests sent as part of the call-leg, including the ACK request (as shown in *Figure 1-8*). Similarly, a proxy that does not Record-Route a SUBSCRIBE request should not expect to see any of the NOTIFY

requests sent in the context of the subscription. Exceptions to these rules may arise from the use of outbound proxy or from loose routing. For more information, see *Loose Routing* on page 18 and *Outbound Proxy* on page 26.

Proxies should normally Record-Route only requests that set up a dialog (currently INVITE and SUBSCRIBE). However, a proxy may add a Record-Route header to any SIP request if it so wishes. SIP user agents do not change their route-lists based on Record-Route headers in requests other than the initial INVITE or SUBSCRIBE. It is recommended that proxies avoid adding Record-Route headers to every request for reasons of processing time and message size. Selective Record-Routing is highly important because it allows proxies to keep track of some dialogs for their entire duration, while assisting others only in their initial setup phase and then removing themselves from loop. This helps proxies avoid spending resources on routing requests for dialogs that are of no interest to them.

The Record-Routing mechanism was enhanced and fine-tuned numerous times during the evolution of SIP in order to cope with various difficult cases. Today a Record-Routing proxy is required to implement the following functionality:

- Route information pre-processing
- Route information post-processing
- Rewriting Record-Route headers in responses (For example, the case of a proxy that is routing between different domain names.)
- Loose Routing (see below)

LOOSE ROUTING

A loose routing proxy (also known as Loose Router) is one that follows the procedures defined for Record-Routing in RFC2543bis-08 and later. These procedures allow a proxy to introduce more hops into the route-list regardless of the final destination of the message. This enables a proxy to route a message through a pre-defined set of other proxies before reaching its final destination. Loose routing was added to the SIP specification due to requirements set by wireless 3G standard bodies, such as the 3GPP, in order to allow for “intelligent” routing of messages between visited and home networks.

RECURSION ON 3XX RESPONSES

The scenario in [Figure 1-9](#) shows recursion on 3xx Response

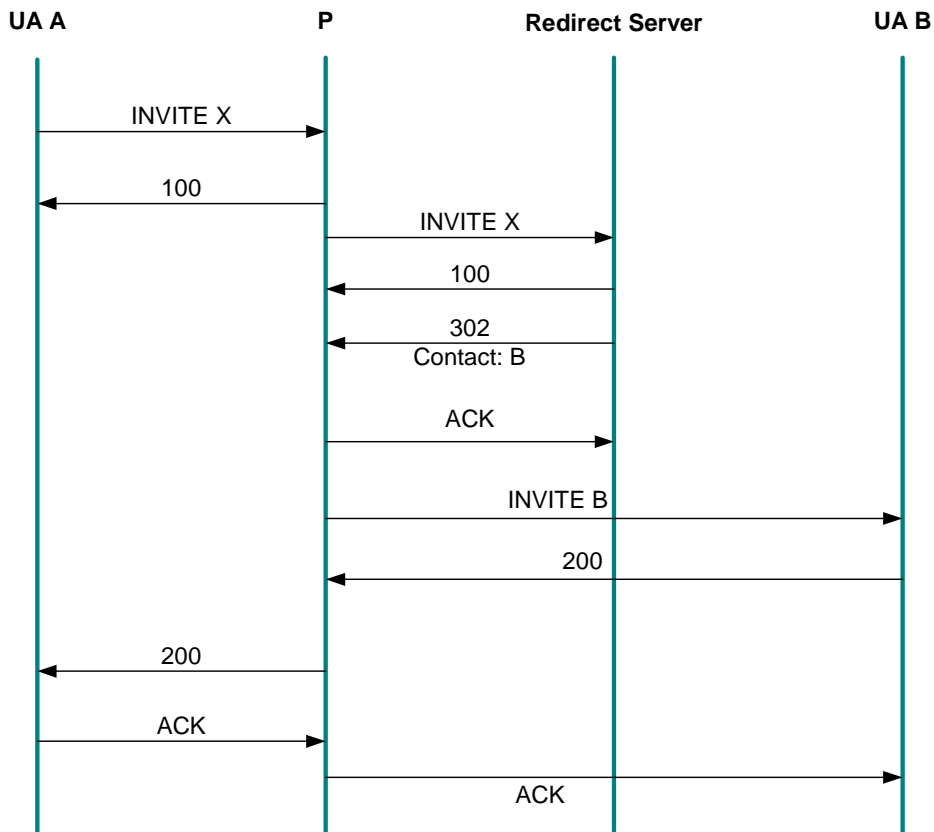


Figure 1-9 Recursion on 3xx Response

A proxy, upon receiving a 3xx (Redirection) response, may choose to add the contact address(es) provided in the 3xx response to the target set and possibly generate a new copy of the request to this address(es), as illustrated in [Figure 1-9](#). This process is called recursion on 3xx responses. A proxy may do recursion on 3xx responses only if the Request-URI of the original request indicates an address for which the proxy is responsible.

FORKING

After processing an incoming request and building a target-set for it, the proxy may choose to forward the request to multiple addresses. This process is called forking and a proxy that supports it is called a forking proxy. Forking allows the implementation of features such as simultaneously searching (ringing) for a user in multiple user agent devices (desktop phone and cellular phone, for example), finding the first available agent in a call-center, and Forward-on-Busy.

A proxy may choose to fork in several ways:

- Parallel forking—the proxy forwards copies of the request to multiple destinations simultaneously.
- Sequential forking—the proxy forwards the request to one target address at a time, waiting for a final response (failure) before moving on to the next address.
- Mixed forking—the proxy may choose to forward requests to some target addresses in parallel while doing sequential forwarding for others.

PARALLEL FORKING

The scenario in *Figure 1-10* (below) shows parallel forking.

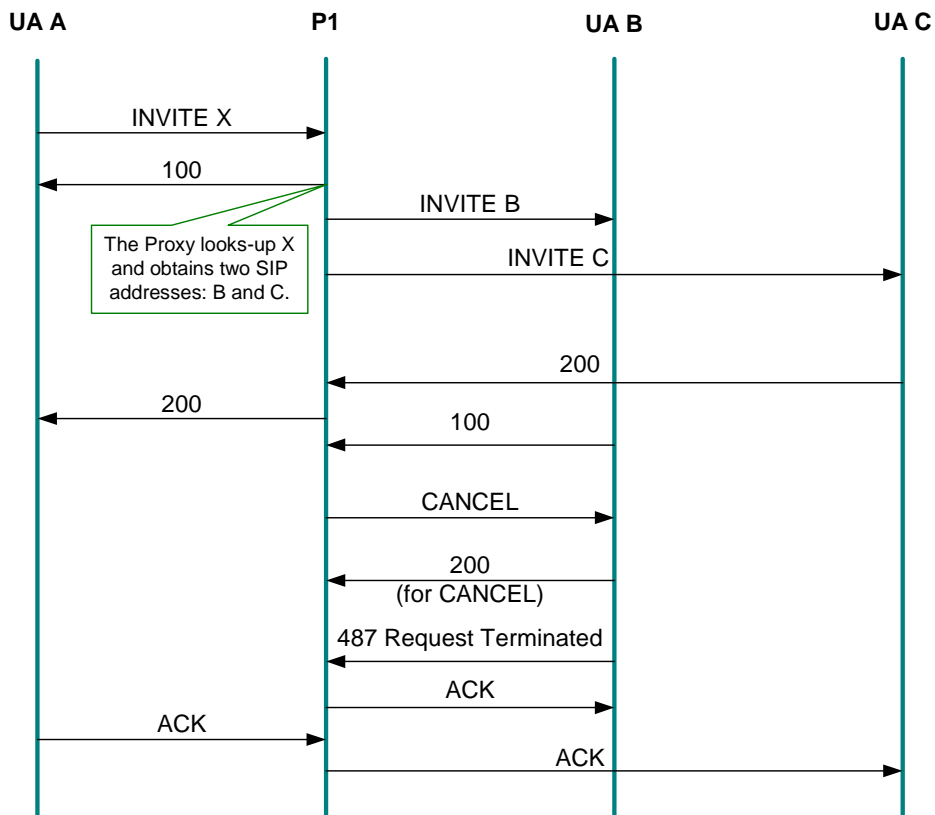


Figure 1-10 Parallel Forking

Parallel forking, as illustrated in [Figure 1-10](#), is a more time-effective way to search for a user. This is because parallel forking attempts to reach the user in multiple locations simultaneously. However, parallel forking introduces additional complexity into the work of the proxy. A parallel forking proxy has to handle multiple concurrent client transactions and possibly collect multiple final responses from them. The proxy has to choose the “best” final response and forward it upstream. Picking the best response is done according to an algorithm provided in the SIP standard. Some responses have higher preference than others, and some responses should never be forwarded upstream. For example, a 200 (OK) response is “better” than a 404 (Not Found) response. Upon receiving

Proxy Server

a 2xx (Success) or 6xx (Global Failure) response at one of the client transactions, the proxy has to cancel the remainder of the pending requests (using CANCEL) and forward the final response upstream.

When parallel forking, a proxy also may have to do aggregation of challenges if it receives multiple 401 (Authentication Required) or 407 (Proxy Authentication Required) responses. In this case, the proxy must collect the challenges from all responses and forward them upstream.

A parallel forking proxy may have to do redirection response aggregation if multiple 3xx responses are received.

SEQUENTIAL FORKING

The scenario in [Figure 1-11](#) shows sequential forking.

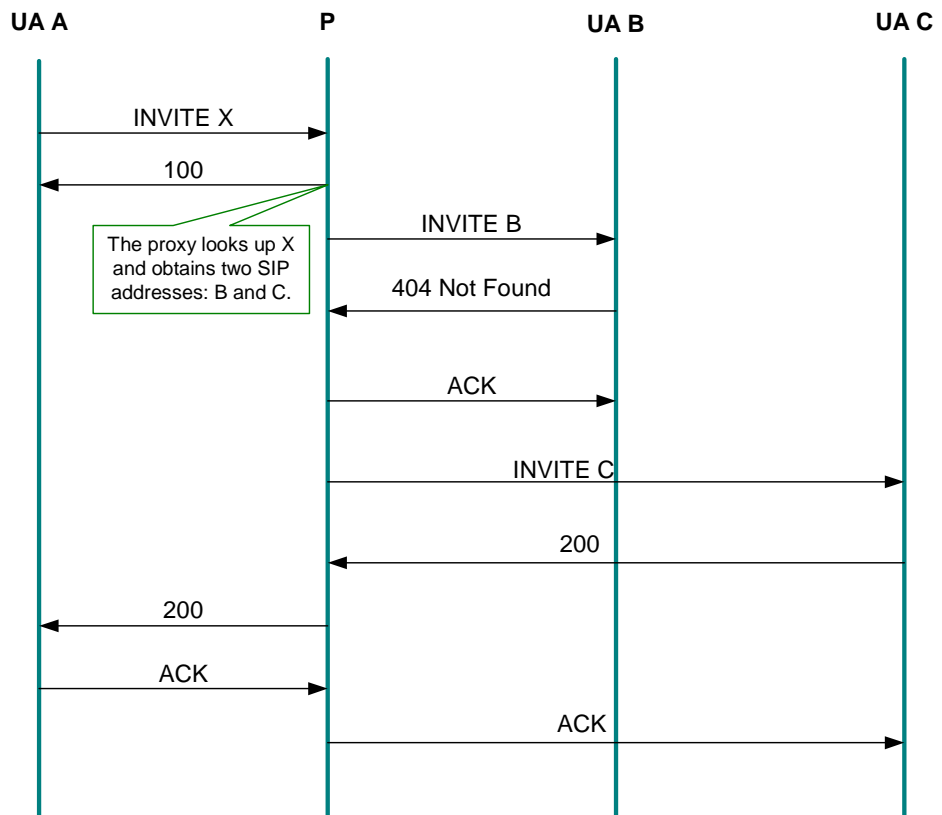


Figure 1-11 Sequential Forking

AUTHENTICATION

The scenario in *Figure 1-12* shows authentication.

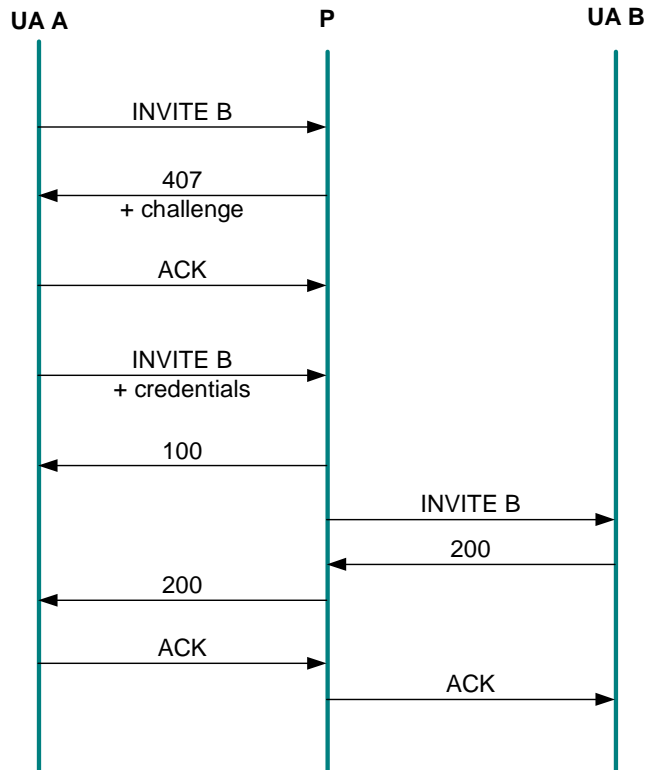


Figure 1-12 Authentication

When a UAC sends a request to a proxy server, the proxy server may decide to authenticate the originator before the request is processed. The proxy can challenge the originator by returning a 407 response (Proxy Authentication Required) with a Proxy-Authenticate header containing the challenge. The client can re-send the request with a Proxy-Authorization header which provides the credentials that match the challenge. A client may provide the credentials also before being challenged in order to avoid the delay and extra processing of the 407 response (the credentials may be built according to cached challenges). Both challenge and credentials are built using a cryptographic hash so that certain values, such as password, are not sent in the clear.

Authentication by proxy is useful for the following:

Proxy Server

- Verifying that the originator of the request is indeed an authorized user entitled to receive services (avoiding service theft)
- Asserting that certain message fields were not altered by a third party

SIP authentication also allows for multi-level authentication by different proxies along the signaling path.

LOOP DETECTION AND MAX FORWARDS

Figure 1-13 shows an example of an illegal loop.

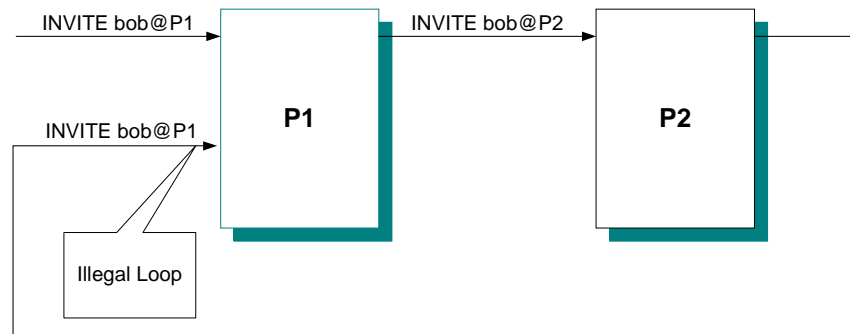


Figure 1-13 *Illegal Loop*

A loop is a situation where a request that arrives at a proxy is forwarded, and later arrives back at the same proxy. An illegal loop occurs if the second time it arrives, the message has the same values in fields that affect the routing decision (Request-URI and any other fields the proxy may take into account). In this case the proxy will forward the request the same way it did the first time and every other proxy along the loop path may do the same. This causes an error condition where the message is looped through a set of proxies for an undefined number of times, consuming network and proxy resources, but never reaching its final destination.

The SIP specification handles loops in two ways:

- Max Forwards (mandatory)
- Loop detection (optional)

MAX FORWARDS

The Max-Forwards header field serves to limit the number of hops a request can transit on the way to its destination. It consists of an integer that is decremented by one at each hop. If the Max-Forwards value reaches 0 before the request reaches its destination, it will be rejected with a 483 (Too Many Hops) error response. The default initial value for Max-Forwards is 70.

This solution requires minimal processing by proxies and UACs, but has the disadvantage of stopping the loop only after the message has been forwarded enough times to exhaust the Max-Forwards value (70 hops by default).

LOOP DETECTION

A proxy can optionally check for loops by employing a special loop detection algorithm. The algorithm affects the way the proxy builds the Via-branch field and mandates the proxy to do certain validations of the Via list in incoming requests. Loop detection requires some extra processing per message, but guarantees immediate detection of the loop as the proxy receives the message for the second time.

MESSAGE SPIRALING

Figure 1-14 shows an example of message spiraling (Legal Loops).

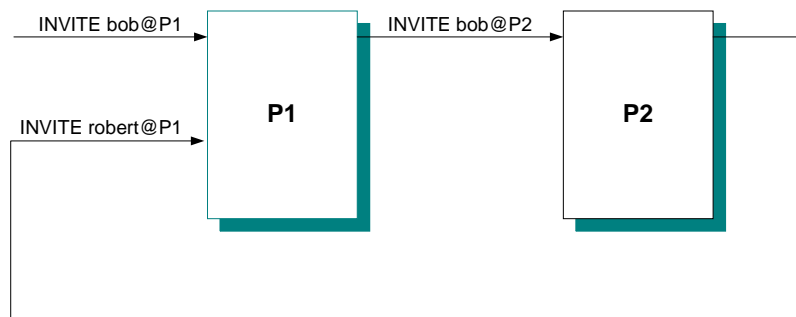


Figure 1-14 Spiral Example

A spiral is a SIP request that is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but with a different set of values in the fields that affect the routing decision. The proxy will route the spiraled request to different target addresses the first and second time it processes it, therefore an endless loop is not created. A spiral is not an error condition, unlike a loop. Identifying spirals and telling them apart from loops presents a challenge for loop-detecting proxies and requires the use of special techniques.

OUTBOUND PROXY

Outbound proxy is a proxy that receives requests from a client regardless of the destination of the messages (Request-URI) the client is sending. Simple clients may choose to send every outbound message via an outbound proxy. Typically, a user agent is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols.

Outbound proxies are important because they allow the creation of simple user agents that do not have to be concerned with making routing decisions and DNS queries. This is especially important in wireless devices which typically have limited capabilities and resources but may roam to foreign networks.

A proxy that is suitable for use as an outbound proxy is one that is willing to accept requests even though they are not intended for its domain and that do not belong to a dialog that it has Record-Routed.

POLICY

As explained above, SIP Servers have a wide array of possible ways to handle incoming messages. A SIP Server has the freedom to make the following decisions:

- Where to route the request and based on which location information source (location service/database/presence info/ other)
- Proxy/Redirect/Reject
- Stateful/Stateless forwarding
- Record-routing
- Forking—parallel/sequential/none
- Authentication
- Loop detection

These decisions can be made based on many factors, such as:

- Message destination address (Request-URI)
- Domain(s) managed by this server
- Type of request (method)
- Other message fields: From, To, Date, Priority ...
- External parameters, such as time of day

In this document, the collective set of rules that govern proxy routing decision making is called Server Policy. Server Policies are typically set by the service providers or administrators that install and configure the SIP Servers.

Because of the large number of input variables and output decisions that are available for Server Policies, SIP Servers are highly versatile and flexible elements. This is of key importance for deployment of SIP-based networks.

Server Policy can be as complex or as simple as required by network topology, user profiles, traffic load, security risk levels, and other factors specific to the environment in which the server operates.

Below are examples of some simple policies that may be employed by proxies:

- Forward all requests targeted to domain acme.com based on location service records. Redirect all other requests to sip:routing_proxy.sp.com
- Authenticate all incoming INVITE messages except those intended for 911 emergency services
- Statefully forward (and fork if necessary) requests with destination address sip:bob_smith@acme.com, unless they are from sip:alice_smith@worldcom.com, in which case forward to sip:bob_smith@voicemail.acme.com.
- Switch from stateful to stateless forwarding modes when system resources are low.

RADVISION SIP SERVER TOOLKIT

The RADVISION SIP Server Toolkit is a comprehensive framework for the development of any type of SIP Server application. The Toolkit includes an implementation of fully-standard server functionality that is controllable through multi-level APIs. SIP Server developers can easily and seamlessly integrate the SIP Server library into their application and customize it according to project needs, thus adding SIP Server capabilities to their implementation quickly and effectively.

The SIP Server Toolkit was designed to specifically address the following requirements.

- Standards compliance
- Interoperability
- Robustness
- Extensibility
- Flexibility and Customization
- Performance and Capacity
- Maintainability
- Portability
- Minimizing Development Time and Cost

The SIP Server Toolkit is available for all major operating systems.

**RADVISION
FAMILY OF SIP
DEVELOPMENT
SOLUTIONS**

The RADVISION SIP Server Toolkit is part of a complete family of SIP technology products RADVISION provides for SIP developers.

The other products include:

- RADVISION SIP Toolkit—RADVISION's award winning development tool including SIP, SDP, RTP/RTCP Stacks.
- RADVISION ProLabTM SIP Test Manager—a complete SIP lab testing solution.
- RADVISION Media Device Framework—a framework for the development of IP phones and residential gateways.