Requirements from SIP (Session Initiation Protocol) Session Border
Control Deployments
draft-camarillo-sipping-sbc-funcs-03.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 7, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This documents describes functions implemented in Session Initiation
Protocol (SIP) intermediaries known as Session Border Controllers (SBCs). Although the goal of this document is to describe all the functions of SBCs, a special focus is given to those practices that are viewed to be in conflict with SIP architectural principles. It also explores the underlying requirements of network operators that have led to the use of these functions and practices in order to identify protocol requirements and determine whether those requirements are satisfied by existing specifications or additional standards work is required.
# Table of Contents

1. Introduction .............................................. 4  
2. Background on SBCs ....................................... 4  
   2.1. Peering Scenario .................................... 5  
   2.2. Access Scenario ..................................... 6  
3. Functions of SBC ......................................... 7  
   3.1. Topology Hiding ...................................... 7  
      3.1.1. General Information and Requirements ............ 7  
      3.1.2. Architectural Issues .......................... 8  
      3.1.3. Example ....................................... 8  
   3.2. Media Traffic Shaping ............................... 9  
      3.2.1. General Information and Requirements ............ 9  
      3.2.2. Architectural Issues .......................... 10  
      3.2.3. Example ....................................... 10  
   3.3. Fixing Capability Mismatches ....................... 11  
      3.3.1. General Information and Requirements ............ 11  
      3.3.2. Architectural Issues .......................... 11  
      3.3.3. Example ....................................... 12  
   3.4. NAT Traversal ....................................... 13  
      3.4.1. General Information and Requirements ............ 13  
      3.4.2. Architectural Issues .......................... 13  
      3.4.3. Example ....................................... 13  
   3.5. Access Control ....................................... 14  
      3.5.1. General Information and Requirements ............ 14  
      3.5.2. Architectural Issues .......................... 15  
      3.5.3. Example ....................................... 15  
   3.6. Protocol Repair ..................................... 16  
      3.6.1. General Information and Requirements ............ 16  
      3.6.2. Architectural Issues .......................... 16  
      3.6.3. Examples ....................................... 16  
   3.7. Media Encryption .................................... 17  
      3.7.1. General Information and Requirements ............ 17  
      3.7.2. Architectural Issues .......................... 17  
      3.7.3. Example ....................................... 17  
4. Derived Requirements (TODO) ............................ 18  
5. Open Issues ............................................. 18  
6. Security Considerations ................................. 19  
7. IANA Considerations ..................................... 19  
8. Acknowledgements ....................................... 19  
9. References ............................................... 19  
   9.1. Normative References ................................ 19  
   9.2. Informational References ........................... 19  
Authors’ Addresses ....................................... 20  
Intellectual Property and Copyright Statements .......... 21
1. Introduction

In the past few years there has been a rapid adoption of SIP [1] and deployment of SIP-based communications networks. This has often out-paced the development and implementation of protocol specifications to meet network operator requirements. This has led to the development of proprietary solutions. Often these proprietary solutions are implemented in network intermediaries known in the marketplace as Session Border Controllers (SBCs) because they typically are deployed at the border between two networks. The reason for this is that network policies are typically enforced at the edge of the network.

Even though many SBCs currently break things like end-to-end security and can impact feature negotiations, there is clearly a market for them. Network operators need many of the features current SBCs provide and many times there are no standard mechanisms available to provide them in a better way. This document describes the most common functions of current SBCs and the reasons that network operators require them. It also describes the architectural issues with these functions. Although this document focuses on functions common to SBCs, many of the issues raised apply to other types of B2BUAs.

2. Background on SBCs

The term SBC is pretty vague, since it is not standardized or defined anywhere. Nodes that may be referred to as SBCs but do not implement SIP are outside the scope of this document.

SBCs usually sit between two service provider networks in a peering environment, or between an access network and a backbone network to provide service to residential and/or enterprise customers. They provide a variety of functions to enable or enhance session-based multi-media services (e.g., Voice over IP). These functions include: a) perimeter defense (access control, topology hiding, DoS prevention, and detection); b) functionality not available in the endpoints (NAT traversal, protocol interworking or repair); and c) network management (traffic monitoring, shaping, and QoS). Some of these functions may also get integrated into other SIP elements (like pre-paid platforms, 3GPP P-CSCF, 3GPP I-CSCF etc).

SIP-based SBCs typically handle both signaling and media and implement behavior which is equivalent to a "privacy service" (as described in [3]) performing both Header Privacy and Session Privacy. SBCs often modify certain SIP headers and message bodies that proxies are not allowed to modify. Consequently, they are, by definition,
B2BUAs (Back-to-Back User Agents). The transparency of these B2BUAs varies depending on the functions they perform. For example, some SBCs modify the session description carried in the message and insert a Record-Route entry. Other SBCs replace the value of the Contact header field with the SBCs address, and generate a new Call-ID and new To and From tags.

![SBC architecture](image)

Figure 1: SBC architecture

Figure 1 shows the logical architecture of an SBC, which includes a signaling and a media component. In this document, the terms outer and inner network are used for describing these two networks.

### 2.1. Peering Scenario

A typical peering scenario involves two network operators who exchange traffic with each other. For example, in a toll bypass application, a gateway in operator A’s network sends an INVITE that is routed to the softswitch (proxy) in operator B’s network. The proxy responds with a redirect (3xx) message back to the originating gateway that points to the appropriate terminating gateway in operator B’s network. The originating gateway then sends the INVITE to the terminating gateway.
Figure 2 illustrates the peering arrangement with a SBC where Operator A is the outer network, and Operator B is the inner network. Operator B uses the SBC to control access to its network, protect its gateways and softswitches from unauthorized use and DoS attacks, and monitor the signaling and media traffic. It also simplifies network management by minimizing the number ACL (Access Control List) entries in the gateways. The gateways do not need to be exposed to the peer network, and they can restrict access (both media and signaling) to the SBCs. The SBC guarantees that only media from valid sessions will reach the gateway.

### 2.2. Access Scenario

In an access scenario, presented in Figure 3, the SBC is placed at the border between the access network (outer network) and the operator’s backbone network (inner network) to control access to the backbone network, protect its components (media servers, application servers, gateways, etc.) from unauthorized use and DoS attacks, and monitor the signaling and media traffic. Also, as a part of access control, since the SBC is call stateful, it can prevent over subscription of the access links. Endpoints are configured with the SBC as their outbound proxy address. The SBC routes requests to one or more proxies in the backbone.
Figure 3: Access scenario with SBC

Some endpoints may be behind enterprise or residential NATs. In cases where the access network is a private network, the SBC is the NAT for all traffic. The proxy usually does authentication/authorization for registrations and outbound calls. The SBC does modify the REGISTER request so that subsequent requests to the registered address-of-record is routed to the SBC. This is done either with a Path header, or by modifying the Contact to point at the SBC.

3. Functions of SBC

This section lists those functions that are used in SBC deployments in current communication networks. Each subsection describes a particular function or feature, operators’ requirements for having it, explanation on why it affects the SIP end-to-end model, and a concrete example from its implementation. Each section also discusses potential concerns specific to that particular way of implementing it. Providing suggestions for alternative, more SIP-friendly ways of implementing each of the functions is outside the scope of this document.

All the examples given in this section are somewhat simplified situations from the reality. Only the relevant header lines from SIP and SDP [4] messages are displayed.

3.1. Topology Hiding

3.1.1. General Information and Requirements

Topology hiding consists of limiting the amount of topology
information given to external parties. Operators have a requirement for this functionality because they do not want the IP addresses of their equipment (proxies, gateways, application servers, etc) to be exposed to outside parties. This may be because they do not want to expose their equipment to DoS (Denial of Service) attacks, they may use other carriers for certain traffic and do not want their customers to be aware of it or they may want to hide their internal network architecture from competitors or partners. In some environments, the operator’s customers may wish to hide the addresses of their equipment or the SIP messages may contain private, non-routable addresses.

The most common form of topology hiding is the application of header privacy (see Section 5.1 of [3]), which involves stripping Via and Record-Route headers and replacing the Contact header. However, in deployments which use IP addresses instead of domain names in headers that cannot be removed (e.g. From and To headers), the SBC must replace these IP addresses with its own IP address or domain name.

3.1.2. Architectural Issues

This functionality is based on a hop-by-hop trust model v/s an end-to-end trust model. The messages are modified without subscriber consent and could potentially modify or remove information about the user’s privacy, security requirements and higher layer applications which are communicating end-to-end using SIP. Either users in an end-to-end call may perceive this as a MitM (Man-in-the-Middle) attack.

Modification of IP addresses in URIs in SIP headers can lead to application failures when these URIs are communicated to other SIP servers outside the current dialog. These URIs could appear in a REFER request or in the body of NOTIFY request as part of an event package. If these messages traverse the same SBC, it has the opportunity to restore the original IP address. On the other hand, if the REFER or NOTIFY message returns to the original network through a different SBC that does not have access to the address mapping, the recipient of the message will not see the original address. This may cause the application function to fail.

3.1.3. Example

The current way of implementing topology hiding consists of having an SBC act as a B2BUA (Back-to-Back User Agents) and remove all traces of topology information (e.g., Record-Route and Via entries) from outgoing messages.

Like a regular proxy server that inserts a Record-Route entry, the
SBC handles every single message of a given SIP dialog. However, unlike the proxy server, if the SBC loses state (e.g., the SBC restarts for some reason), it will not be able to route messages properly. For example, if the SBC removes Via entries from a request and then restarts losing state, the SBC will not be able to route responses to that request.

Let us imagine the following example scenario: The SBC (p4.domain.example.com) is receiving an INVITE request from the inner network, which in this case is an operator network. The received SIP message is:

```
INVITE sip:callee@u2.domain.example.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p3.middle.example.com>
Record-Route: <sip:p2.example.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

Then the SBC performs a topology hiding function. In this imagined situation the SBC removes and stores all existing Record-Route headers, and then insert a Record-Route header field with its own SIP-URI. After the topology hiding function, the message looks like:

```
INVITE sip:callee@u2.domain.example.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p4.domain.example.com;lr>
```

This is only one example scenario from topology hiding, and SBCs can, in some cases, modify other headers as well, like SIP Contact etc.

### 3.2. Media Traffic Shaping

#### 3.2.1. General Information and Requirements

Media traffic shaping is the act of controlling media traffic. Operators require this functionality, because they want to control the traffic they carry on their network. Traffic shaping helps them create different kinds of billing models (e.g., video telephony can be priced differently than voice-only calls). Additionally, traffic shaping can be used to implement intercept capabilities (e.g., lawful intercept).

Since the path of the media through the network is independent of the path of the signaling, the media may not traverse the operator’s network unless the SBC modifies the session description to force the media to be sent thru the SBC.
Some operators do not actually want to reshape the traffic, but only to monitor it for collecting statistics and making sure that they are able to meet any business level agreements with their subscribers and/or partners. However, the SIP techniques needed for monitoring media traffic are the same as for reshaping media traffic.

SBCs on the media path are also capable of dealing with the "lost BYE" issue when either endpoint dies in the middle of the session. The SBC can detect that the media has stopped flowing and issue a BYE to the both sides to cleanup the state in the network.

One possible form of media traffic shaping is that SBCs terminate media streams and SIP dialogs by generating BYE requests. This kind of procedure can take place e.g., in a situation where subscriber runs out of credits. In this scenario, the SBC may be perceived as originating messages which the user may not be able to authenticate as coming from the dialog peer or the SIP Registrar/Proxy.

3.2.2. Architectural Issues

The current way of implementing traffic shaping requires the SBC to access and modify the session descriptions (i.e., offers and answers) exchanged between the user agents. Consequently, this approach does not work if user agents encrypt or integrity-protect their message bodies end-to-end. Again, messages are modified without subscriber consent, and user agents do not have any way to distinguish the SBC actions from an attack by a MitM (Man-in-the-Middle).

3.2.3. Example

Currently, traffic shaping is performed in the following way. The SBC behaves as a B2BUA and inserts itself, or some other entity under the operator’s control, in the media path. In practice, the SBC modifies the session descriptions carried in the SIP messages. As a result, the SBC receives media from one user agent and relays it to the other in both directions.

An example of traffic shaping is codec restriction. The SBC restricts the codec set negotiated in offer/answer [2] exchange between the user agents. After modifying the session descriptions, the SBC can check whether or not the media stream corresponds to what was negotiated in the offer/answer exchange. If it differs, the SBC has the ability to terminate the media stream.

Let us imagine the following example scenario: The SBC is receiving an INVITE request from the outer network, which in this case is an access network. The received SIP message contains the following SDP session descriptor:
Then the SBC performs a media traffic shaping function. In this imagined situation the SBC rewrites the ’m’ line, and removes one ’a’ line according to some policy. After the traffic shaping function, the session descriptor looks like:

```
v=0
c=IN IP4 126.16.64.4
m=audio 49230 RTP/AVP 96
a=rtpmap:96 L8/8000
```

One problem of media traffic shaping is that the SBC needs to understand the session description protocol and all the extensions used by the user agents. This means that in order to use a new extension (e.g., an extension to implement a new service) or a new session description protocol, it is not enough with upgrading the user agents; SBCs in the network need also to be upgraded. This fact may slow down service innovation.

### 3.3. Fixing Capability Mismatches

#### 3.3.1. General Information and Requirements

SBCs fixing capability mismatches enable communications between user agents with different capabilities, SIP profiles or extensions. For example, user agents on networks which implement different SIP Profiles (for example 3GPP or Packet Cable etc) or those that support different IP versions, different codecs, or that are in different address realms. Operators have a requirement and a strong motivation for performing capability mismatch fixing, so that they can provide transparent communication across different domains. In some cases different SIP extensions or methods to implement the same SIP application (like monitoring session liveness, call history/diversion etc) may also be interworked through the SBC.

#### 3.3.2. Architectural Issues

SBCs fixing capability mismatches insert a media element in the media path using the procedures described in Section 3.2. Therefore, these SBCs have the same concerns as SBCs performing traffic shaping: the
SBC modifies SIP messages without explicit consent from any of the user agents. This may break end-to-end security and application extensions negotiation.

Additionally, if the network is not engineered properly, an SBC may make the wrong assumption about the capabilities of the user agents. When this happens, user agents with compatible capabilities may end up communicating via the SBC instead of doing it directly between them (e.g., the SBC assumes that a dual-stack user agent only supports IPv6).

3.3.3. Example

Let us imagine the following example scenario where the inner network is an access network using IPv4 and the outer network is using IPv6. The SBC receives an INVITE request with a session description from the access network:

```
INVITE sip:callee@ipv6.domain.example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4
Contact: sip:caller@u1.example.com
v=0
o=mhandley 2890844526 2890842807 IN IP4 192.0.2.4
c=IN IP4 192.0.2.4
m=audio 49230 RTP/AVP 96
a=rtpmap:96 L8/8000
```

Then the SBC performs a capability mismatch fixing function. In this imagined situation the SBC inserts Record-Route and Via headers, and rewrites the ‘c’ line from the sessions descriptor. After the capability mismatch fixing function, the message look like:

```
INVITE sip:callee@ipv6.domain.com SIP/2.0
Record-Route: <sip:[2001:620:8:801:201:2ff:fe94:8e10];lr>
Via: SIP/2.0/UDP sip:[2001:620:8:801:201:2ff:fe94:8e10]
Via: SIP/2.0/UDP 192.0.2.4
Contact: sip:caller@u1.example.com
v=0
o=mhandley 2890844526 2890842807 IN IP4 192.0.2.4
c=IN IP6 2001:620:8:801:201:2ff:fe94:8e10
m=audio 49230 RTP/AVP 96
a=rtpmap:96 L8/8000
```

Now the SBC sends the modified message to the outer IPv6 network.
3.4. NAT Traversal

3.4.1. General Information and Requirements

An SBC performing a NAT (Network Address Translator) traversal function for a user agent behind a NAT sits between the user agent and the registrar of the domain. NATs are widely deployed in various access networks today, so operators have a requirement to support it. When the registrar receives a REGISTER request from the user agent and responds with a 200 (OK) response, the SBC modifies such a response decreasing the validity of the registration (i.e., the registration expires sooner). This forces the user agent to send a new REGISTER to refresh the registration sooner that it would have done on receiving the original response from the registrar. The REGISTER requests sent by the user agent refresh the binding of the NAT before the binding expires.

Note that the SBC does not need to relay all the REGISTER requests received from the user agent to the registrar. The SBC can generate responses to REGISTER requests received before the registration is about to expire at the registrar. Moreover, the SBC needs to deregister the user agent if this fails to refresh its registration in time, even if the registration at the registrar would still be valid.

Operators implement this functionality in an SBC instead of in the registrar for several reasons: (i) preventing packets from unregistered users to prevent chances of DoS attack, (ii) prioritization and/or re-routing of traffic (based on user or service, like E911) as it enters the network. (iii) performing a load balancing function or reducing the load on other network equipment.

3.4.2. Architectural Issues

This approach to NAT traversal does not work when end-to-end confidentiality or integrity-protection is used. The SBC would be seen as a MitM modifying the messages between the user agent and the registrar.

3.4.3. Example

Let us imagine the following example scenario: The SBC resides between the UA and Registrar. Previously the UA has sent a REGISTER request to Registrar, and then the SBC is going to relay the following SIP message to UA:
Then the SBC performs a traversal function. In this imagined situation the SBC rewrites the 'expires' parameter on the Contact header field. After the NAT traversal function, the message looks like:

```
SIP/2.0 200 OK
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlf1
To: Bob <sip:bob@biloxi.example.com>;tag=34095828jh
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=60
```

Naturally also other measures need to be taken in order to enable the NAT traversal, but this example illustrated only the mechanism on how NAT bindings can be kept alive.

### 3.5. Access Control

#### 3.5.1. General Information and Requirements

It is pretty self evident that operators want to control what kind of signaling and media traffic their network carries. So, they have a strong motivation and a requirement to do access control on the edge of their network. Access control can be based on, for example, IP addresses or SIP identities.

This function is implemented by protecting the inner network with firewalls and configuring them so that they only accept SIP traffic from the SBC. This way, all the SIP traffic entering the inner network needs to be routed through the SBC, which only routes messages from authorized parties or traffic that meets a specific policy that is expressed in the SBC administratively.

Access control can be applied either only to the signaling, or to both the signaling and media. If it is applied only to the signaling, then the SBC behaves as a proxy server. Therefore, it does not break any SIP architectural principle. If access control is applied to both the signaling and media, then the SBC behaves as in a similar manner as explained in Section 3.2. A key part of media-layer access control is that only media for authorized sessions is allowed to pass through the SBC. In any case, since the SBC needs to handle every single message, this function has a scalability
implications. In addition, the SBC is a single point of failure from the architectural point of view. Although, in practice, many current SBCs have redundant configuration, which prevents the loss of calls/sessions in the event of a failure.

In environments where there is limited bandwidth on the access links, the SBC can compute the potential bandwidth usage by examining the codecs present in SDP offers and answers. With this information, the SBC can reject sessions before the available bandwidth is exhausted to allow existing sessions to maintain acceptable quality of service. Otherwise, the link could become over subscribed and all sessions would experience a deterioration in quality of service. Some SBCs can contact a policy server to determine whether sufficient bandwidth is available.

3.5.2. Architectural Issues

If access control is performed only on behalf of signaling, then the SBC is not SIP friendly, but if it is performed for signaling and for media, then there are similar problems as described in Section 3.2.2.

3.5.3. Example

There could be a following scenario, where SBC is performing access control for signaling and media:

<table>
<thead>
<tr>
<th>caller</th>
<th>SBC</th>
<th>callee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify the caller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;- - - - - - - - - - &gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVITE + SDP</td>
<td>[Modify the SDP]</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>INVITE + modified SDP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------------------------------</td>
<td></td>
</tr>
<tr>
<td>[Modify the SDP]</td>
<td>200 OK + SDP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>----------------------------------------</td>
<td></td>
</tr>
<tr>
<td>200 OK + modified SDP</td>
<td>[Modify the SDP]</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Media</td>
<td>[Media inspection]</td>
<td>Media</td>
</tr>
<tr>
<td>&lt;==========================================</td>
<td>==========================================</td>
<td></td>
</tr>
</tbody>
</table>
In this scenario, the SBC first identifies the caller, so it can determine whether or not to give signaling access for the caller. After that, the SBC modifies the session descriptors in INVITE and 200 OK messages in a way that the media is going to flow through SBC itself. When the media starts flowing, the SBC can inspect whether the callee and caller use the codec(s) that they had previously agreed on.

3.6. Protocol Repair

3.6.1. General Information and Requirements

SBC are also used to repair protocol messages generated by not-fully-standard clients. Operators have a requirement to support protocol repair, if they want to support as many client as possible. It is noteworthy, that this function affects only the signaling component of SBC, and that protocol repair function is not the same as protocol conversion.

3.6.2. Architectural Issues

In most cases, this function can be seen as SIP friendly, and it does not violate the end-to-end model of SIP. The SBC repairing protocol messages behaves as a proxy server that is liberal in what it accepts and strict in what it sends. In principle, such an SBC does not break any architectural principle of SIP.

3.6.3. Examples

The SBC can, for example, receive the following INVITE message from a not-fully-standard client:

```
INVITE
Via: SIP/2.0/UDP u1.example.com:5060
From: Caller <sip:caller@one.example.com>
To: Callee <sip:callee@two.example.com>
Call-ID: 18293281@u1.example.com
CSeq: 1 INVITE
Contact: sip:caller@u1.example.com
```

If the SBC does protocol repair, it can for example try to write the request line based on To header field, and it also can remove excess white spaces to make the SIP message more human readable.

Some other examples of "protocol repair" that have actually been implemented in commercially available SBCs include:
Changing Content-Disposition from "signal" to "session". This was required for a user agent which sent an incorrect Content-Disposition header.

Addition of userinfo to a Contact URI when none was present. This was required for a softswitch/proxy that would reject requests if the Contact URI had no user part.

Addition of a to-tag to provisional or error responses.

Re-ordering of Contact header values in a REGISTER response. This was required for a user agent that would take the expires value from the first Contact header value without matching it against its Contact value.

Correction of SDP syntax where the user agent used "annexb=" in the fmt attribute instead of the proper "annexb:".

Correction of signaling errors (convert BYE to CANCEL) for termination of early sessions.

Repair of header parameters in ‘archaic’ or incorrect formats. Some older stacks assume that parameters are always of the form NAME=VALUE. For those elements, it is necessary to convert ‘lr-true’ to ‘lr’ in order to interoperate with several commercially available stacks and proxies.

3.7. Media Encryption

3.7.1. General Information and Requirements

SBCs are used to perform media encryption / decryption at the edge of the network. This is the case when media encryption is used only on the access network (outer network) side and the media is carried unencrypted in the inner network. Operators can have an obligation to provide the ability to do legal interception, while they still want to give their customers the ability to encrypt media in the access network. This leads to a situation where operators have a requirement to perform media encryption function.

3.7.2. Architectural Issues

While performing media encryption function, SBCs need to be able to inject either themselves, or some other entity to the media path. Due to this, the SBCs have same architectural issues as explained in Section 3.2.

3.7.3. Example

There could be a following scenario, where SBC is performing media encryption:
<table>
<thead>
<tr>
<th>caller</th>
<th>SBC#1</th>
<th>SBC#2</th>
<th>callee</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE + SDP</td>
<td>---------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Modify the SDP]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INVITE + mod. SDP</td>
<td>---------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Modify the SDP]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>INVITE + mod. SDP</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Modify the SDP]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200 OK + mod. SDP</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Modify the SDP]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200 OK + mod. SDP</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Modify the SDP]</td>
<td></td>
</tr>
<tr>
<td>Encrypted media</td>
<td>Plain media</td>
<td>Encrypted media</td>
<td></td>
</tr>
<tr>
<td>[enc./dec.]</td>
<td>[enc./dec.]</td>
<td>[enc./dec.]</td>
<td></td>
</tr>
</tbody>
</table>

First the caller send INVITE, and the first SBC modifies the session descriptor in a way, that it injects itself to the media path. The same happens in the second SBC. Then the caller replies with 200 OK, and when the SBCs receive it, they inject themselves also to the returning media path. After signaling the media start flowing, and both SBCs are performing media encryption and decryption.

4. Derived Requirements (TODO)

TODO: enumerate protocol requirements based on network operator requirements and identify which are satisfied by existing work and which may require new work.

5. Open Issues

Several domains and IP addresses in the examples within this document still require NIT checking and changing to be in line with best
practices for examples.

6. Security Considerations

Many of the functions this document describes have important security and privacy implications. If the IETF decides to develop standard mechanisms to address those functions, security and privacy-related aspects will need to be taken into consideration.

7. IANA Considerations

This document has no IANA considerations.

8. Acknowledgements

The ad-hoc meeting about SBCs, held on Nov 9th 2004 at Washington DC during the 61st IETF meeting, provided valuable input to this document. Special thanks goes also to Sridhar Ramachandran, Gaurav Kulshreshtha, and to Rakendu Devdhar.

9. References

9.1. Normative References


9.2. Informational References

Authors’ Addresses

Jani Hautakorpi (editor)
Ericsson
Hirsalantie 11
Jorvas 02420
Finland
Email: Jani.Hautakorpi@ericsson.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland
Email: Gonzalo.Camarillo@ericsson.com

Medhavi Bhatia
NexTone Communications
101 Orchard Ridge Drive
Gaithersburg, MD 20878
US
Email: mbhatia@nextone.com

Robert F. Penfield
Acme Packet
71 Third Avenue
Burlington, MA 01803
US
Email: bpenfield@acmepacket.com

Alan Hawrylyshen
Ditech Communications Corporation
Suite 200, 1167 Kensington Cres NW
Calgary, Alberta T2N 1X7
Canada
Email: ahawrylyshen@ditechcom.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.