SIPPING Working Group
Internet-Draft
Expires: May 28, 2007

J. Hautakorpi, Ed.
G. Camarillo
Ericsson
R. Penfield
Acme Packet
A. Hawrylyshen
Ditech Networks Inc.
M. Bhatia
NexTone Communications
November 24, 2006

Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments
draft-ietf-sipping-sbc-funcs-00.txt

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 SBCs 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 12
3.3.1. General Information and Requirements 12
3.3.2. Architectural Issues 12
3.3.3. Example 12
3.4. NAT Traversal 13
3.4.1. General Information and Requirements 13
3.4.2. Architectural Issues 14
3.4.3. Example 14
3.5. Access Control 15
3.5.1. General Information and Requirements 15
3.5.2. Architectural Issues 16
3.5.3. Example 16
3.6. Protocol Repair 17
3.6.1. General Information and Requirements 17
3.6.2. Architectural Issues 17
3.6.3. Examples 18
3.7. Media Encryption 19
3.7.1. General Information and Requirements 19
3.7.2. Architectural Issues 19
3.7.3. Example 19
4. Derived Requirements 20
5. Security Considerations 21
6. IANA Considerations 21
7. Acknowledgements 21
8. References 22
8.1. Normative References 22
8.2. Informational References 22
Authors' Addresses 23
Intellectual Property and Copyright Statements 24

1. Introduction

In the past few years there has been a rapid adoption of Session Initiation Protocol (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 Back-to-Back User Agents (B2BUAs.)

2. Background on SBCs

The term SBC is relatively non-specific, 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 can implement behavior which is equivalent to a "privacy service" (as described in[2]) 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.

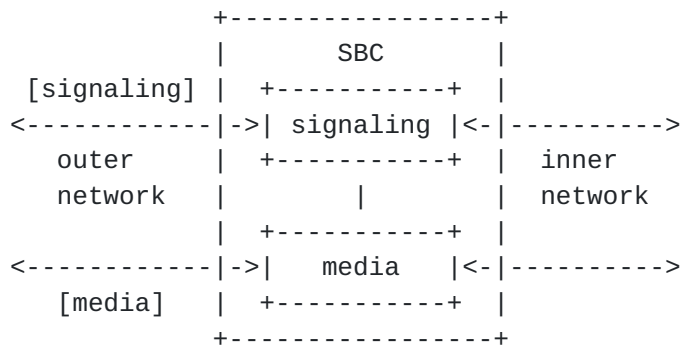

Figure 1: SBC architecture

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.

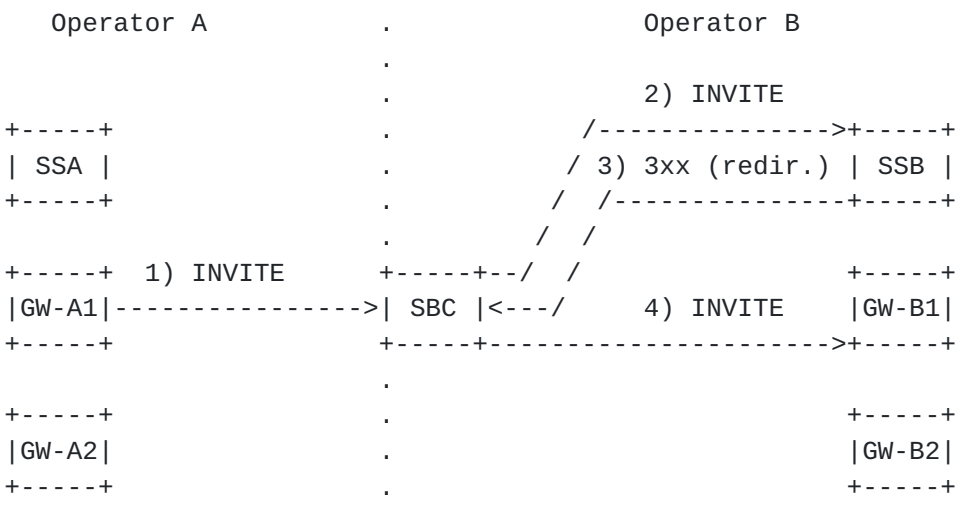

Figure 2: Peering with SBC

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 helps ensure that only media from sessions the SBC authorizes 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 network (inner network) to control access to the operator 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 operator network.

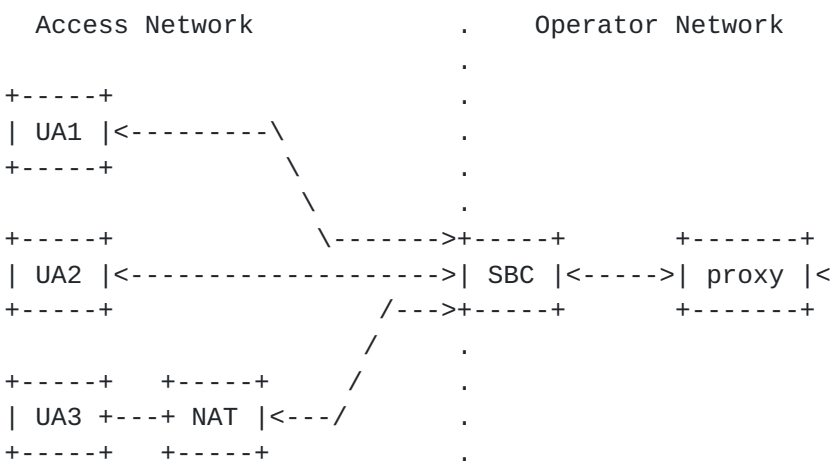

Figure 3: Access scenario with SBC

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 and/or authorization for registrations and outbound calls. The SBC modifies the REGISTER request so that subsequent requests to the registered address-of-record are 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 SBCs

This section lists those functions that are used in SBC deployments in current communication networks. Each subsection describes a particular function or feature, the operators' requirements for having it, explanation on any impact to the end-to-end SIP architecture, and a concrete implementation example. Each section also discusses potential concerns specific to that particular implementation technique. Suggestions for alternative implementation techniques that may be more architecturally compatible with SIP are outside the scope of this document.

All the examples given in this section are simplified; 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 [2]), 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 may 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 as opposed to 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 user in an end-to-end call may perceive this as a Man In The Middle (MitM) attack.

Modification of IP addresses in Unifor Resource Indetifiers (URIs) within SIP headers can lead to application failures if 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.[[Comment.1: Do we have a sane example of where this is a real problem? It sounds somewhat contrived to me, but I agree it is a theoretical concern - Alan.]][[Comment.2: I personally would like to include this text, although it might be more of a theoretical concern. - Jani]]

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.

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 shown in Figure 4.

```
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>
```

Figure 4: INVITE Request Prior to Topology Hiding

Then the SBC performs a topology hiding function. In this scenario, the SBC removes and stores all existing Record-Route headers, and then inserts a Record-Route header field with its own SIP URI. After the topology hiding function, the message could appear as shown in Figure 5.

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

Figure 5: INVITE Request After Topology Hiding

Like a regular proxy server that inserts a Record-Route entry, the SBC handles every single message of a given SIP dialog. If the SBC loses state (e.g., the SBC restarts for some reason), it may 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 may not be able to route responses to that request; depending on the information that was lost when the SBC restarted. [[Comment.3: There are techniques to mitigate this problem, not all SBCs suffer from this. Is this worth capturing in the text? [Alan]]][[Comment.4: No, not all suffer from this, but some do, so I believe we shouldn't remove this text. - Jani]]

This is only one example of topology hiding, in some cases, SBCs may modify other headers, including the Contact header field values.

3.2. Media Traffic Shaping

3.2.1. General Information and Requirements

Media traffic shaping is the act of controlling media traffic. Network operators may require this functionality in order to control

the traffic being carried on their network on behalf of their subscibers. Traffic shaping helps 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 where required to support audit or legal obligations.

Since the media path is independent of the signaling path, the media may not traverse through the operator's network unless the SBC modifies the session description. By modifying the session description the SBC can force the media to be sent through a media relay which may be co-located with the SBC.

Some operators do not want to reshape the traffic, but only to monitor it for collecting statistics and making sure that they are able to meet any business service level agreements with their subscribers and/or partners. The protocol 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 if 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 any state in other intermediate elements and the endpoints.

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.

3.2.2. Architectural Issues

Implementing traffic shaping in this manner 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).

In this application, the SBC may originate messages that the user may not be able to authenticate as coming from the dialog peer or the SIP Registrar/Proxy.

3.2.3. Example

Traffic shaping may be 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 user-agent and performs the identical operation with media traveling in the reverse direction.

CODEC restriction is an example application of traffic shaping. The SBC restricts the codec set negotiated in the offer/answer exchange [3] 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 or take other appropriate (configured) actions (e.g. alarming or reporting).

Consider the following example scenario: The SBC receives an INVITE request from the one network, which in this case is an access network. The received SIP message contains the SDP session descriptor shown in Figure 6.

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 10.16.64.4
c=IN IP4 10.16.64.4
m=audio 49230 RTP/AVP 96 98
a=rtpmap:96 L8/8000
a=rtpmap:98 L16/16000/2
```

Figure 6: Request Prior to Media Shaping

In this example, the SBC performs the media traffic shaping function by rewritting the 'm' line, and removing one 'a' line according to some (external) policy. Figure 7 shows the session description after the traffic shaping function.

```
v=0
o=mhandley 2890844526 2890842807 IN IP4 10.16.64.4
c=IN IP4 10.16.64.4
m=audio 49230 RTP/AVP 96
a=rtpmap:96 L8/8000
```

Figure 7: Request Body After Media Shaping

One problem with media traffic shaping is that the SBC needs to understand the session description protocol and all 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, SBCs in the network may need to be upgraded in conjunction with the endpoints. Certain extensions that do not

require active manipulation of the session descriptors to facilitate traffic shaping will be able to be deployed without upgrading existing SBCs, depending on the degree of transparency the SBC implementation affords. In cases requiring an SBC modification to support the new protocol features, the rate of service deployment may be affected. [[Comment.5: I do not think this will slow down innovation; innovation is a distinct phase of development and separable from operational network deployment. -Alan]][[Comment.6: I don't quite get what you are suggesting. If you want to change the text, go ahead. - Jani]]

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.

[[Comment.7: I have removed the network engineering concern; this is an unrealistic anti-Apple-Pie problem that could only arise through a fundamental bug in either configuration or SBC implementation. -Alan]][[Comment.8: Ok. - Jani]]

3.3.3. Example

Consider 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
```

Figure 8: Request Prior to Capabilities Match

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. Figure 9 shows the request after the capability mismatch adjusment.

```
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
```

Figure 9: Request After Capability Match

This message is then sent by the SBC to the onward IPv6 network.

3.4. NAT Traversal

3.4.1. General Information and Requirements

NAT traversal in this instance refers to the specific message modifications required to assist a user-agent in maintaining SIP and media connectivity when there is a NAT device located between the user-agent and the proxy/registrar and, most likely, any other user-agent.

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, and (iii) performing a load balancing function or reducing the load on other network equipment.

Also other measures, such as acting as a media relay by modifying SDP session descriptors (see Section 3.2), may be taken by SBC to ensure media connectivity.

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.

[[Comment.9: Is there something more specific to this function we can put here? This is very generic and a general limitation of SBC architectures.]][[Comment.10: If you want to add something, please do. - Jani]]

3.4.3. Example

Consider the following example scenario: The SBC resides between the UA and Registrar. Previously the UA has sent a REGISTER request to Registrar, and the SBC receives the registration response shown in Figure 10.

```
SIP/2.0 200 OK
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sip:bob@biloxi.example.com>;tag=34095828jh
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=3600
```

Figure 10: Response Prior to NAT Maintenance Function

When performing the NAT traversal function, the SBC may re-write the expiry time to coax the UA to re-register prior to the intermediating NAT deciding to close the pinhole. Figure 11 shows a possible modification of the response from Figure 10.

```
SIP/2.0 200 OK
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sip:bob@biloxi.example.com>;tag=34095828jh
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=60
```

Figure 11: Manipulated Response for NAT Traversal

Naturally also other measures need to be taken in order to enable the NAT traversal, but this example illustrates only one mechanism for preserving the SIP related NAT bindings.

3.5. Access Control

3.5.1. General Information and Requirements

Network operators may wish to control what kind of signaling and media traffic their network carries. There is strong motivation and a requirement to do access control on the edge of an operator's network. Access control can be based on, for example, IP addresses or SIP identities.

This function can be 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 though 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 might behave as a proxy server. If access control is applied to both the signaling and media, then the SBC behaves 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 and/or associated media relay devices.

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. SBCs may contact a policy server to determine whether sufficient bandwidth is available on a per-session basis.

3.5.2. Architectural Issues

Since the SBC needs to handle all SIP messages, this function has scalability implications. In addition, the SBC is a single point of failure from an architectural point of view. Although, in practice, many current SBCs have the capability to support redundant configuration, which prevents the loss of calls and/or sessions in the event of a failure on a single node.

[[Comment.11: I am tempted to remove this paragraph; this is a general architectural problem that is not truly specific to SBCs. A proxy configured into a SIP architecture that Record-Route'd requests would ALSO be a single point of failure. Provisioning a network to deal with the outage of a single element is just good design. -Alan]][[Comment.12: I agree that this is not specific only to SBCs, but is specific also to SBCs. I wouldn't like to remove this paragraph. - Jani]]

If access control is performed only on behalf of signaling, then the SBC is compatible with general SIP architectural principles, 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

Figure 12 shows a callflow where the SBC is providing both signaling and media access control.

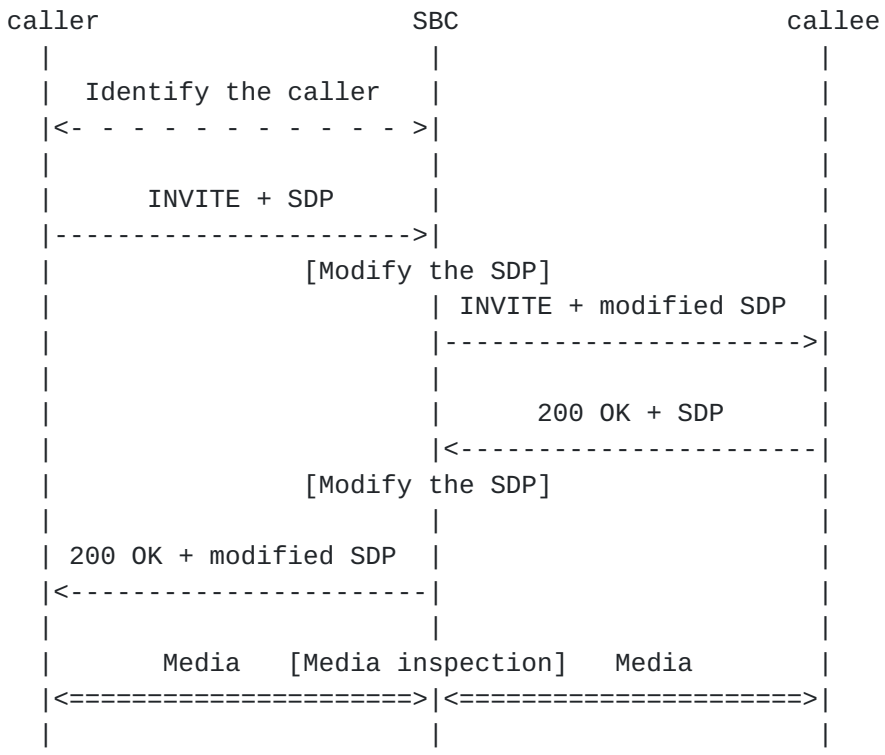

Figure 12: Example Access Callflow

In this scenario, the SBC first identifies the caller, so it can determine whether or not to give signaling access for the caller. Some SBCs may rely on the proxy to authenticate the user-agent placing the call. After authentication, 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 may wish 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 being compatible with SIP architectural principles, 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.

3.6.3. Examples

The SBC can, for example, receive the an INVITE message from a not-fully-standard client as illustrated in Figure 13.

```
INVITE sip:callee@sbchost.example.com
Via: SIP/2.0/UDP u1.example.com:5060;lr=1
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
```

Figure 13: Request from non-standard client

If the SBC does protocol repair, it can re-write the request line (in this case the UAC did not put the target in the URI and instead put the SBC hostname as the host portion). It could also remove excess white spaces to make the SIP message more human readable.

Some other examples of "protocol repair" that have 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 fmtp 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' or 'lr=1' 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 may 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 a 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 the same architectural issues as explained in Section 3.2.

3.7.3. Example

Figure 14 shows an example where the SBC is performing media encryption related functions.

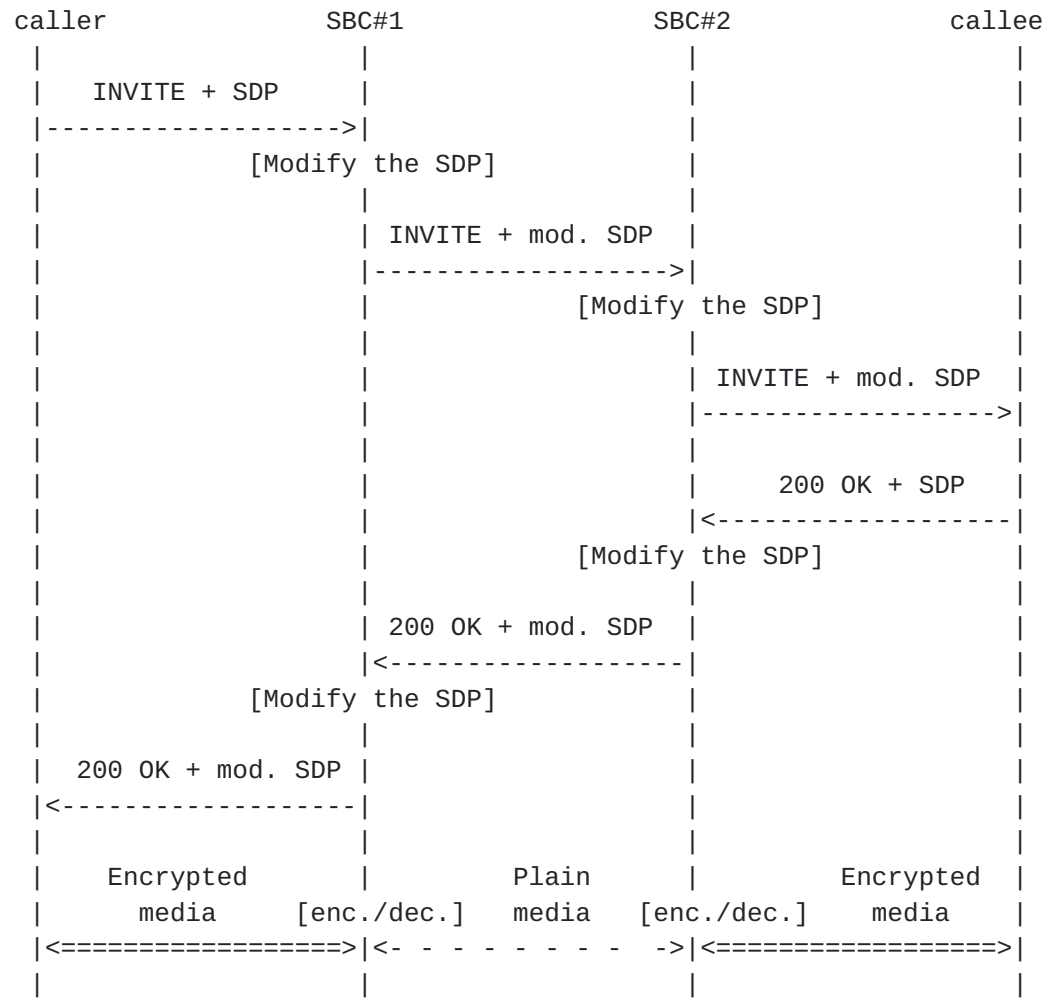

Figure 14: Media Encryption Example

First the UAC sends an INVITE request , 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 UAS replies with a 200 OK response, the SBCs inject themselves in the returning media path. After signaling the media start flowing, and both SBCs are performing media encryption and decryption.

4. Derived Requirements

Some of the functions listed in this document are more SIP-unfriendly than others. This list requirements that are derived from the functions that break the principles of SIP in one way or the other. The derived requirements are:

Req-1: There should be a SIP-friendly way to hide network topology information. Currently this is done e.g., by stripping and replacing header fields, which is against the principles of SIP.

Req-2: There should be a SIP-friendly way to direct media traffic through intermediaries. Currently this is done e.g., by modifying session descriptors, which is against the principles of SIP.

Req-3: There should be a SIP-friendly way to fix capability mismatches in SIP messages. Currently this is done by modifying SIP messages, which violates e.g., end-to-end security.

All the above-mentioned requirements are such that they do not have an existing solution today. Thus, future work is needed in order to develop solutions to these requirements.

5. 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.

[[Comment.13: I wonder if it is worth classifying the specific type of security problems and assembling them here. The remainder of this document can then refer to the specific problem a given operational activity has given today's typically implementation mechanisms. [Alan]]][[Comment.14: My gut feeling is that it would require a lot of work. If you want to do it, go ahead, but at least I don't have the time to do it. - Jani]]

6. IANA Considerations

This document has no IANA considerations.

7. 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.

8. References

8.1. Normative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Peterson, J., "A Privacy Mechanism for the Session Initiation Protocol (SIP)", RFC 3323, November 2002.

[3] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

8.2. Informational References

[4] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

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

Robert F. Penfield
Acme Packet
71 Third Avenue
Burlington, MA 01803
US

Email: bpenfield@acmepacket.com

Alan Hawrylyshen
Ditech Networks Inc.
Suite 200, 1167 Kensington Cres NW
Calgary, Alberta T2N 1X7
Canada

Email: ahawrylyshen@ditechnetworks.com

Medhavi Bhatia
NexTone Communications
101 Orchard Ridge Drive
Gaithersburg, MD 20878
US

Email: mbhatia@nextone.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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