Network Working Group
Internet-Draft
Expires: January 5, 2001

D. Raz
Lucent Technologies
J. Schoenwaelder
TU Braunschweig
B. Sugla
ISPSoft Inc.
July 7, 2000

An SNMP Application Level Gateway for Payload Address Translation
draft-ietf-nat-snmp-alg-05.txt

Status of this Memo

Copyright Notice

Abstract

This document describes the Application Level Gateway (ALG) for the Simple Network Management Protocol (SNMP) by which IP addresses in the payload of SNMP packets are statically mapped from one group to another. The SNMP ALG is a specific case of an Application Level Gateway as described in [15].

An SNMP ALG allows network management stations to manage multiple networks that use conflicting IP addresses. This can be important in environments where there is a need to use SNMP with NAT in order to

manage several potentially overlapping addressing realms.

This document includes a detailed description of the requirements and limitations for an implementation of an SNMP Application Level Gateway. It also discusses other approaches to exchange SNMP packets across conflicting addressing realms.

Table of Contents

1. Introduction

The need for IP address translation arises when a network's internal IP addresses cannot be used outside the network. Using basic network address translation allows local hosts on such private networks (addressing realms) to transparently access the external global Internet and enables access to selective local hosts from the outside. In particular it is not unlikely to have several addressing realms that are using the same private IPv4 address space within the same organization.

In many of these cases, there is a need to manage the local addressing realm from a manager site outside the domain. However, managing such a network presents unique problems and challenges. Most available management applications use SNMP (Simple Network Management Protocol) to retrieve information from the network elements. For example, a router may be queried by the management application about the addresses of its neighboring elements. This information is then sent by the router back to the management station as part of the payload of an SNMP packet. In order to retain consistency in the view as seen by the management station we need to be able to locate and translate IP address related information in the payload of such packets.

The SNMP Application Level Gateway for Payload Address Translation, or SNMP ALG, is a technique in which the payload of SNMP packets (PDUs) is scanned and IP address related information is translated if needed. In this context, an SNMP ALG can be an additional component in a NAT implementation, or it can be a separate entity, that may reside in the same gateway or even on a separate node. Note that in our context of management application all devices in the network are assumed to have a fixed IP address. Thus, SNMP ALG should only be combined with NAT that uses static address assignment for all the devices in the network.

A typical scenario where SNMP ALG is deployed as part of NAT is presented in figure Figure 1. A manager device is managing a remote stub, with translated IP addresses.

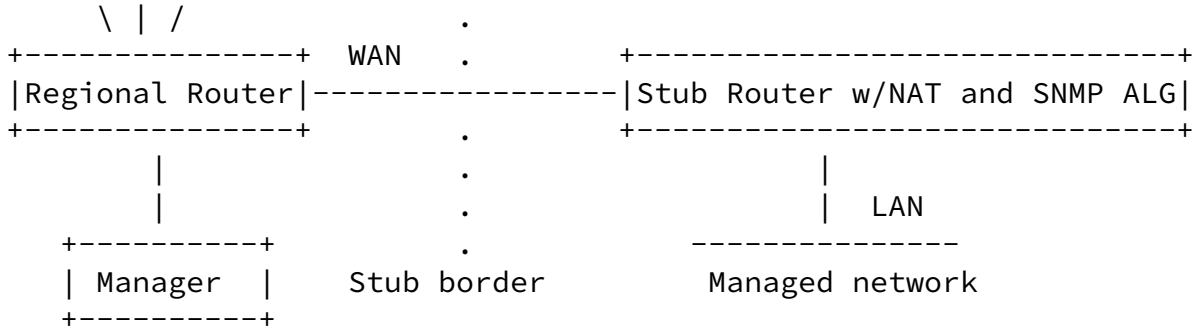
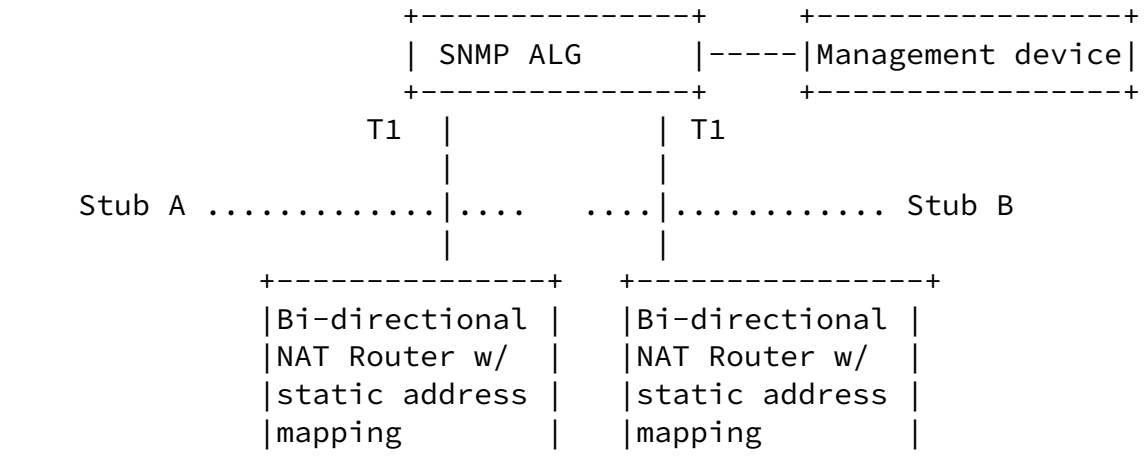


Figure 1: SNMP ALG in a NAT configuration

A similar scenario occurs when several subnetworks with private (and possibly conflicting) IP addresses are to be managed by the same management station. This scenario is presented in Figure 2.



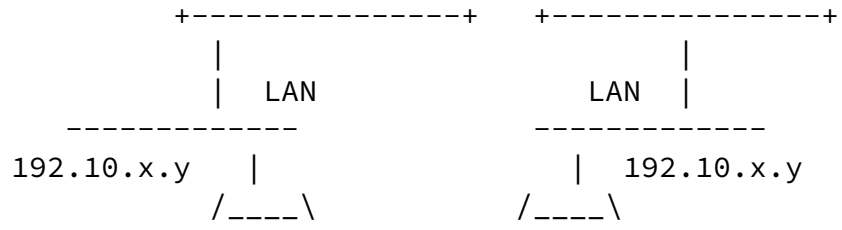


Figure 2: Using external SNMP ALG to manage two private networks

Since the devices in the managed network are monitored by the manager device they must obtain a fixed IP address. Therefore, the NAT used in this case must be a basic NAT with a static one to one mapping.

An SNMP ALG is required to scan all the payload of SNMP packets, to detect IP address related data, and to translate this data if needed. This is a much more computationally involved process than the bi-directional NAT, however they both use the same translation tables. In many cases the router may be unable to handle SNMP ALG and retain acceptable performance. In these cases it may be better to locate the SNMP ALG outside the router, as described in Figure 2.

2. Terminology and Concepts Used

In general we adapt the terminology defined in [15]. Our main concern are SNMP messages exchanged between SNMP engines. This document only discusses SNMP messages that are send over UDP, which is the preferred transport mapping for SNMP messages [5]. SNMP messages send over other transports can be handled in a similar way. Thus, the term SNMP packet is used throughout this document to refer to an SNMP message contained in an UDP packet.

SNMP messages contain SNMP PDUs (Protocol Data Units). An SNMP PDU defines the parameters for a specific SNMP protocol operation. The notion of flow is less relevant in this case, and hence we will focus on the information contained in a single SNMP packet.

There are currently three versions of SNMP. SNMP version 1 (SNMPv1) protocol is defined in STD 15, RFC 1157 [2]. The SNMP version 2c (SNMPv2c) protocol is defined in RFC 1901 [3], RFC 1905 [4] and RFC 1906 [5]. Finally, the SNMP version 3 (SNMPv3) protocol is defined in RFC 1905 [4], 1906 [5], RFC 2572 [10] and RFC 2574 [12]. See RFC 2570 [9] for a more detailed overview over the SNMP standards. In

the following, unless otherwise mentioned, we use the term SNMP in statements that are applicable to all three SNMP versions.

SNMP uses ASN.1 [13] to define the abstract syntax of the messages. The actual encoding of the messages is done by using the Basic Encoding Rules (BER) [14], which provide the transfer syntax .

We refer to packets that go from a management station to the network elements as "outgoing", and packets that go from the network elements to the management station as "incoming".

A basic SNMP ALG is an SNMP ALG implementation in which only IP address values encoded in the IpAddress type are translated. A basic SNMP ALG therefore does not need to be MIB aware.

An advanced SNMP ALG is an SNMP ALG implementation which is capable of handling and replacing IP address values encoded in well known IP address data types and instance identifiers derived from those data types. This implies that an advanced SNMP ALG is MIB aware.

3. Problem Scope and Requirements

As mentioned before, in many cases, there is a need to manage a local addressing realm that is using NAT, from a manager site outside the realm. A particular important example is the case of network management service providers who provide network management services from a remote site. Such providers may have many customers, each using the same private address space. When all these addressing

realms are to be managed from a single management station address collision occurs. There are two straight forward ways to overcome the address collision. One can

1. reassign IP addresses to the different addressing realms, or
2. use static address NAT to hide the address collisions from the network management application.

The first solution is problematic as it requires both a potentially large set of IP addresses, and the reconfiguration of a large portion of the network. The problem with the second solution is that many network management applications are currently unaware of NAT, and because of the large investment needed in order to make them NAT aware are likely to remain so in the near future.

Hence, there is a need for a solution that is transparent to the network management application (but not to the user), and that does not require a general reconfiguration of a large portion of the network (i.e. the addressing realm). The SNMP ALG described in this memo is such a solution.

3.1 IP Addresses in SNMP Messages

SNMP messages can contain IP addresses in various places and formats. The following four categories have been identified:

1. IP version 4 addresses and masks stored in the IpAddress tagged ASN.1 data type which are not part of an instance identifier. An example is the ipAdEntNetMask object defined in the IP-MIB [6].
2. IP version 4 addresses contained in instance identifiers derived from index objects using the IpAddress data type. An example is the ipAdEntAddr index object of the IP-MIB [6].
3. IP addresses (any version) contained in OCTET STRINGS. Examples include addressMapNetworkAddress object of the RMON2-MIB [7], and IP addresses contained in OCTET STRINGS derived from well-known textual conventions (e.g. TAddress [5] or Ipv6Address [8] or InetAddress [19]).
4. IP addresses (any version) contained in instance identifiers derived from OCTET STRINGS. This may derived from well-known textual conventions (e.g. TAddress [5] or Ipv6Address [8] or InetAddress [19]) like the ipv6AddrAddress index object of the IPV6-MIB [8].

Textual conventions that can contain IP addresses can be further divided in NAT friendly and NAT unfriendly ones. A NAT friendly textual convention ensures that the encoding on the wire contains sufficient information that an advanced SNMP ALG which understands the textual convention and which has the necessary MIB knowledge can do a proper translation. An example of this type is the Ipv6Address textual convention.

A NAT unfriendly textual convention requires that an SNMP ALG, which understands the textual convention and which has the necessary MIB knowledge, has access to additional information in order to do a proper translation. Examples of this type are the TAddress and the InetAddress textual conventions which require that an additional varbind is present in an SNMP packet to determine what type of IP address a given value represents. Such a varbind may or may not be

present depending on the way a management applications retrieves data.

3.2 Requirements

An SNMP ALG should provide transparent IP address translation to management applications. An SNMP ALG must be compatible with the behavior of the SNMP protocol operations as defined by RFC 1157 [2] and RFC 1905 [4] and must not have negative impact on the security provided by the SNMP protocol. A fully transparent SNMP ALG must be able to translate all categories of IP addresses as described above, when provided with the specified OID's and the encoding details.

The SNMP ALG requires bi-directional NAT devices enroute, that support static address mapping for all nodes in the respective private realms. When there are multiple private realms supported by a single SNMP ALG, the external addresses assumed by each of the NAT devices must not collide with each other.

4. Translating IP Addresses in SNMP Packets

This section describes several ways to translate IP addresses in SNMP packets.

A general SNMP ALG must be capable to translate IP addresses in outgoing and incoming SNMP packets.

SNMP messages send over UDP may experience fragmentation at the IP layer. In an extreme case, fragmentation may cause an IP address type to be partitioned into two different fragments. In order to translate IP addresses in SNMP messages, the complete SNMP message must be available. As described in [18], fragments of UDP packets do not carry the destination/source port number with them. Hence, an SNMP ALG must reassemble IP packets which contain SNMP messages. The good news is, however, that usually SNMP agents are aware of the MTU, and that SNMP packets are usually relatively small. Some SNMP implementations also set the don't fragment (DF) bit in the IP header [1] to avoid fragmentation.

4.1 Basic SNMP Application Level Gateway

A basic SNMP ALG is an SNMP ALG implementation in which only IP

address values encoded in the IpAddress base type are translated. A basic SNMP ALG implementation parses an ASN.1/BER encoded SNMP packet looking for elements that are encoded using the IpAddress base type. Appendix A contains a more detailed description of the structure and encoding used by SNMP.

An IpAddress value can be identified easily by its tag value (0x40). Once an IpAddress has been detected, the SNMP ALG checks the translation table and decides whether the address should be translated. If the address needs translation, the 4 bytes representing the IPv4 address are replaced with the translated IPv4 address and the UDP checksum is adjusted. Section 4.3 describes an efficient algorithm to adjust the UDP checksum without recalculating it.

The basic SNMP ALG does not require knowledge of any MIBs since it relies on the ASN.1/BER encoding of SNMP packets. It is therefore easy to implement. A basic SNMP ALG does not change the overall messages size and hence it does not cause translated messages to be lost due to message size constraints.

However, a basic SNMP ALG is only able to translate IPv4 addresses in objects that use the IpAddress base type. Furthermore, a basic SNMP ALG is not capable to translate IP addresses in objects that are index components of conceptual tables. This is especially problematic on index components that are not accessible. Hence, the basic SNMP ALG is restricted to the first out of the four possible ways to represent IP addresses in SNMP messages (see Section 3.1).

4.2 Advanced SNMP Application Level Gateway

An advanced SNMP ALG is an SNMP ALG implementation which is capable of handling and replacing IP address values encoded in well known IP address data types and instance identifiers derived from those data types. Hence, an advanced SNMP ALG may be able to transparently map IP addresses that are in the format 1-4 as described in Section 3.1. This implies that an advanced SNMP ALG must be MIB aware.

An advanced SNMP ALG must maintain an OBJECT IDENTIFIER (OID) translation table in order to identify IP addresses that are not encoded in an IpAddress base type. The OID translation table needs to maintain information about the OIDs where translation may be needed. Furthermore, the translation table needs to keep information about instance identifiers for conceptual tables that contain IP addresses. Such an OID translation table may be populated offline by using a MIB compiler which loads the MIBs used within an addressing realm and searches for types, textual conventions and table indexes that may contain IP addresses.

The translation function scans the packet for these specific OIDs, checks the translation table and replaces the data if needed. Note that since OIDs do not have a fixed size this search is much more computationally consuming, and the lookup operation may be expensive.

The ability to translate IP addresses that are part of the index of a conceptual table is a required feature of an advanced SNMP ALG. IP addresses embedded in an instance identifier are ASN.1/BER encoded according to the OID encoding rules. For example, the OID for the 10.1.2.3 instance of the ipAdEntIfIndex object of the IP-MIB [6] is encoded as 06 0D 2B 06 01 02 01 04 14 01 02 0A 01 02 03. Replacing the embedded private IPv4 address with 135.180.140.202 leads to the OID 06 11 2B 06 01 02 01 04 14 01 02 81 07 81 34 81 0C 81 4A. This example shows that an advanced SNMP ALG may change the overall packet size since IP addresses embedded in an OID can change the size of the ASN.1/BER encoded OID.

Another effect of an advanced SNMP ALG is that it changes the lexicographic ordering of rows in conceptual tables as seen by the SNMP manager. This may have severe side-effects for management applications that use lexicographic ordering to retrieve only parts of a conceptual table. Many SNMP managers check lexicographic ordering to detect loops caused by broken agents. Such a manager will incorrectly report agents behind an advanced SNMP ALG as broken SNMP agents.

4.3 Packet Size and UDP Checksum

Changing an IpAddress value in an SNMP packet does not change the size of the SNMP packet. A basic SNMP ALG does therefore never change the size of the underlying UDP packet.

An advanced SNMP ALG may change the size of an SNMP packet since a different number of bytes may be needed to encode a different IP address. This is highly undesirable but unavoidable in the general case. A change of the SNMP packet size requires additional changes in the UDP and IP headers. Increasing packet sizes are especially problematic with SNMPv3. The SNMPv3 message header contains the msgMaxSize field so that agents can generate Response PDUs for GetBulkRequest PDUs that are close to the maximum message size the receiver can handle. An SNMP ALG which increases the size of an SNMP packet may have the effect that the Response PDU can not be

processed anymore. Thus, an advanced SNMP ALG may cause some SNMPv3 interactions to fail.

In both cases, the UDP checksum must be adjusted when making an IP address translation. We can use the algorithm from [18], but a small modification must be introduced as the modified bytes may start on

an odd position. The C code shown in Figure 3 adjusts the checksum to a replacement of one byte in an odd or even position.

```
void checksumbyte(unsigned char *chksum, unsigned char *optr,
unsigned char *nptr, int odd)
/* assuming: unsigned char is 8 bits, long is 32 bits,
   we replace one byte by one byte in an odd position.
  - chksum points to the chksum in the packet
  - optr points to the old byte in the packet
  - nptr points to the new byte in the packet
  - odd is 1 if the byte is in an odd position 0 otherwise
*/
{  long x, old, new;
   x=chksum[0]*256+chksum[1];
   x=~x & 0xFFFF;
   if (odd) old=optr[0]*256; else old=optr[0];
   x-=old & 0xFFFF;
   if (x<=0) { x--; x&=0xFFFF; }
   if (odd) new=nptr[0]*256; else new=nptr[0];
   x+=new & 0xFFFF;
   if (x & 0x10000) { x++; x&=0xFFFF; }
   x=~x & 0xFFFF;
   chksum[0]=x/256; chksum[1]=x & 0xFF;
}
```

5. Limitations and Alternate Solutions

Making SNMP ALGs completely transparent to all management applications is not an achievable task. The basic SNMP ALG described in Section 4.1 only translates IP addresses encoded in the IpAddress base type. Such an SNMP ALG achieves only very limited transparency since IP addresses are frequently used as part of an index into a conceptual table. A management application will therefore see both the translated as well as the original address, which can lead to confusion and erroneous behavior of management applications. However, a certain class of management applications like e.g.

network discovery tools may work pretty well across NATs with a basic SNMP ALG in place.

An advanced SNMP ALG described in Section 4.2 achieves better transparency. However, an advanced SNMP ALG can only claim to be transparent for the set of data types (textual conventions) understood by the advanced SNMP ALG implementation and for a given set of MIB modules. The price paid for better transparency is additional complexity, potentially increased SNMP packet sizes and mixed up lexicographic ordering. Especially with SNMPv3, there is an opportunity that communication fails due to increased packet sizes. Management applications that rely on lexicographic ordering will show erroneous behaviour.

Both, basic and advanced SNMP ALGs, introduce problems when using SNMPv3 security features. The SNMPv3 authentication mechanism protects the whole SNMP message against modifications while the SNMPv3 privacy mechanism protects the payload of SNMPv3 messages against unauthorized access. Thus, an SNMP ALG must have access to all localized keys in use in order to modify SNMPv3 messages without invalidating them. Futhermore, the SNMP ALG must track any key changes in order to function. More details on the security implications of using SNMP ALGs can be found in Section 6.

Finally, an SNMP ALG only deals with SNMP traffic and does not modify the payload of any other protocol. However, management systems usually use a set of protocols to manage a network. In particular the telnet protocol is often used to configure or troubleshoot managed devices. Hence, a management system and the human network operator must generally be aware that a network address translation is occurring, even in the presence of an SNMP ALG.

A possible alternative to SNMP ALGs are SNMP proxies, as defined in RFC 2573 [11]. An SNMP proxy forwarder application forwards SNMP messages to other SNMP engines according to the context, and irrespective of the specific managed object types being accessed. The proxy forwarder also forwards the response to such previously forwarded messages back to the SNMP engine from which the original message was received. Such a proxy forwarder can be used in a NAT environment to address SNMP engines with conflicting IP addresses. (Just replace the box SNMP ALG with a box labelled SNMP PROXY in Figure 2.) The deployment of SNMP proxys has the advantage that

different security levels can be used inside and outside of the conflicting addressing realms.

The proxy solution, which is structurally preferable, requires that the management application is aware of the proxy situation. Furthermore, management applications have to use internal data structures for network elements that allow for conflicting IP addresses since conflicting IP addresses are not translated by the SNMP proxy. Deployment of proxies may also involve the need to reconfigure network elements and management stations to direct their traffic (notifications and requests) to the proxy forwarder.

6. Security Considerations

SNMPv1 and SNMPv2c have very week security services based on community strings. All management information is sent in cleartext without encryption and/or authentication. In such an environment, SNMP messages can be modified by any intermediate node and management application are not able to verify the integrity of SNMP messages. Furthermore, an SNMP ALG does not need to have knowledge

of the community strings in order to translate embedded IP addresses. Thus, deployment of SNMP ALGs in an SNMPv1/SNMPv2c environment introduces no additional security problems.

SNMPv3 supports three security levels: no authentication and no encryption (noAuth/noPriv), authentication and no encryption (auth/noPriv), and authentication and encryption (auth/priv). SNMPv3 messages without authentication and encryption (noAuth/noPriv) are send in cleartext. In such a case the usage of SNMP ALGs introduces no additional security problems.

However, the usage of SNMP ALG introduces new problems when SNMPv3 authentication and optionally encryption is used. First, SNMPv3 messages with authentication and optionally encryption (auth/noPriv and auth/priv) can only be processed by an SNMP ALG which supports the corresponding cryptographic algorithms and which has access to the keys in use. Furthermore, as keys may be updated, the SNMP ALG must have a mechanism that tracks key changes (either by analyzing the key change interactions or by propagating key changes by other mechanisms). Second, the computational complexity of processing SNMP messages may increase dramatically. The message has to be decrypted before the translation takes place. If any translation is done the

hash signature used to authenticate the message and to protect its integrity must be recomputed.

In general, key exchange protocols are complicated and designing an SNMP ALG which maintains the keys for a set of SNMP engines is a non-trivial task. The User-based Security Model for SNMPv3 [12] defines a mechanism which takes a password and generates localized keys for every SNMP engine. The localized keys have the property that a compromised single localized key does not automatically give an attacker access to other SNMP engines, even if the key for other SNMP engines is derived from the same password.

An SNMP ALG implementation which maintains lists of (localized) keys is a potential target to attack the security of all the systems which use these keys. An SNMP ALG implementation which maintains passwords in order to generate localized keys is a potential target to attack the security of all systems that use the same password. Hence, an SNMP ALG implementation must be properly secured so that people who are not authorized to access keys or passwords can not access them.

Finally, SNMP ALGs do not allow a network operator to use different security levels on both sides of the NAT. Using a secure SNMP version outside of a private addressing realm while the private addressing realm runs an unsecured version of SNMP may be highly desirable in many scenarios, e.g. management outsourcing scenarios. The deployment of SNMPv3 proxies instead of SNMP ALGs should be

considered in these cases since SNMP proxies can be configured to use different security levels and parameters on both sides of the proxies.

7. Summary and Recommendations

Several approaches to address SNMP agents across NAT devices have been discussed in this memo.

1. Basic SNMP ALGs as described in Section 4.1 provide very limited transparency since they only translate IPv4 addresses encoded in the IpAddress base type. They are fast and efficient and may be sufficient to execute simple management applications (e.g. topology discovery applications) in a NAT environment. However, other management applications are likely to fail due to the limited transparency provided by a basic SNMP ALG. Basic SNMP

ALGs are problematic in a secure SNMP environment since they need to maintain lists of of keys or passwords in order to function.

2. Advanced SNMP ALGs as described in Section 4.2 provide better transparency. They can be transparent for the set of data types they understand and for a given set of MIB modules. However, an advanced SNMP ALG is much more complex and less efficiency than a basic SNMP ALG. An advanced SNMP ALG may break the lexicographic ordering when IP addresses are used to index conceptual tables and it may change the SNMP packet sizes. Especially with SNMPv3, there is an opportunity that communication fails due to increased message sizes. Advanced SNMP ALGs are problematic in a secure SNMP environment, since they need to maintain lists of of keys or passwords in order to function.
3. SNMP proxies as described in RFC 2573 [11] allow management applications to access SNMP agents with conflicting IP addresses. No address translation is performed on the SNMP payload by an SNMP proxy forwarder. Hence, management applications must be able to deal with network elements that have conflicting IP addresses. This solution requires that management applications are aware of the proxy situation. Deployment of proxies may also involve the need to reconfigure network elements and management stations to direct their traffic (notifications and requests) to the proxy forwarder. SNMP proxies have the advantage that they allow to use different security levels inside and outside of a given addressing realm.

It is recommended that network operators who need to manage networks in a NAT environment make a careful analysis before deploying a solution. In particular, it must be analyzed whether the management applications will work with the transparency and the side-effects provided by SNMP ALGs. Furthermore, it should be researched whether the management applications are able to deal with conflicting IP

addresses for network devices. Finally, the additional complexity introduced to the over all management system by using SNMP ALGs must be compared to the complexity introduced by the structurally preferable SNMP proxy forwarders.

8. Current Implementations

A basic SNMP ALG as described in Section 4.1 was implemented for

SNMPv1 at Bell-Labs, running on a Solaris Machine. The solution described in Figure 2, where SNMP ALG was combined with the NAT implementation of Lucent's PortMaster3, was deployed successfully in a large network management service organization.

9. Acknowledgments

We thank Pyda Srisuresh, for the support, encouragement, and advice throughout the work on this document. We also thank Brett A. Denison for his contribution to the work that led to this document. Additional useful comments have been made by members of the NAT working group.

References

[1] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[2] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "A Simple Network Management Protocol (SNMP)", STD 15, RFC 1157, May 1990.

[3] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.

[4] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[5] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[6] McCloghrie, K., "SNMPv2 Management Information Base for the Internet Protocol using SMIv2", RFC 2011, November 1996.

[7] Waldbusser, S., "Remote Network Monitoring Management Information Base Version 2 using SMIv2", RFC 2021, January 1997.

[8] Haskin, D. and S. Onishi, "Management Information Base for IP Version 6: Textual Conventions and General Group", RFC 2465, December 1998.

[9] Case, J., Mundy, R., Partain, D. and B. Stewart, "Introduction

to Version 3 of the Internet-standard Network Management Framework", RFC 2570, April 1999.

[10] Case, J., Harrington, D., Presuhn, R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.

[11] Levi, D., Meyer, P. and B. Stewart, "SNMP Applications", RFC 2573, April 1999.

[12] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, April 1999.

[13] ISO, , "Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)", International Standard 8824, December 1987.

[14] ISO, , "Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)", International Standard 8825, December 1987.

[15] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[16] Miller, M., "Managing Internetworks with SNMP", MT Books, 1997.

[17] Perkins, D. and E. McGinnis, "Understanding SNMP MIBs", Prentice Hall, ISBN 0-13-437708-7, 1997.

[18] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", ID draft-ietf-nat-traditional-04.txt, April 2000.

[19] Daniele, M., Haberman, B., Routhier, S. and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 2851, June 2000.

Authors' Addresses

Danny Raz
Lucent Technologies
101 Crawfords Corner Rd
Holmdel, NJ 07733-3030
USA

Phone: +1 732 949-6712
Fax: +1 732 949-0399
EMail: raz@lucent.com
URI: http://www.bell-labs.com/

Juergen Schoenwaelder
TU Braunschweig
Bueltenweg 74/75
38106 Braunschweig
Germany

Phone: +49 531 391-3266
Fax: +49 531 391-5936
EMail: schoenw@ibr.cs.tu-bs.de
URI: http://www.ibr.cs.tu-bs.de/

Binay Sugla
ISPSoft Inc.
106 Apple Street
Tinton Falls, NJ 07724
USA

Phone: +1 732 936-1763
EMail: sugla@ispsoft.com
URI: http://www.ispsoft.com/

Appendix A. Description of the Encoding of SNMP Packets

SNMP packets use the ASN.1/BER encoding. We will not cover the full details of this encoding in this document. These details can be found in the International Standards ISO-8824 [13] and ISO-8825 [14]. A good description of ASN.1/BER can be found in the book "Managing Internetworks with SNMP", by M. A. Miller [16], or in Appendix A of the book "Understanding SNMP MIBs", by D. Perkins, and E. McGinnis [17].

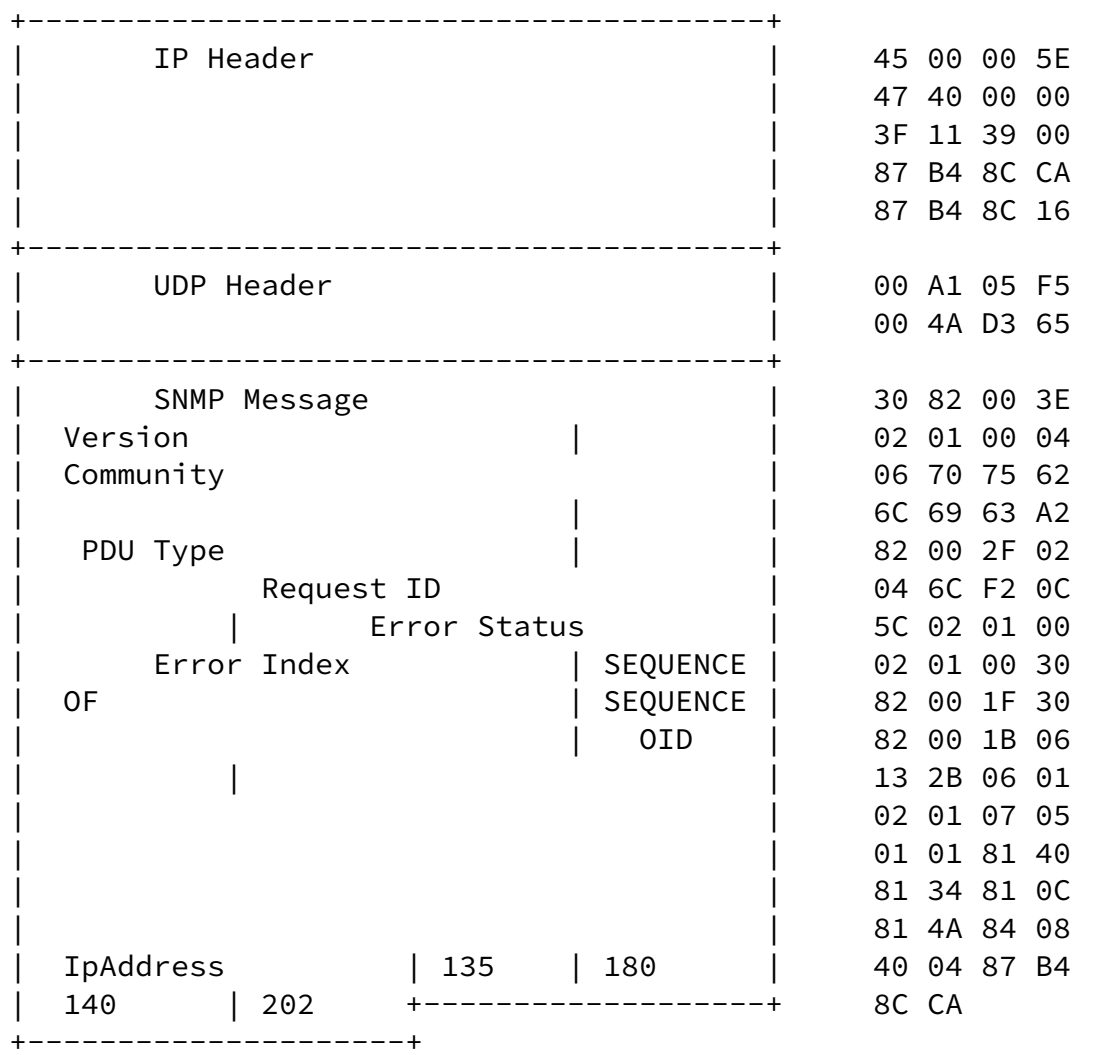
Each variable that is referred to in an SNMP packet is uniquely identified by an OID (Object Identifier), usually written as a

sequence of numbers separated by dots (e.g. 1.3.6.1.2.1.1.3.0). Each variable also has an associated base type (this is not very accurate but good enough for this level of description). One possible base

type is the IpAddress type. The base type of each variable and its OID are conveyed by the ASN.1/BER encoding. Note that it is possible to associate additional type information with a variable by using textual conventions. The additional type semantics provided by textual conventions are not conveyed by the ASN.1/BER encoding.

When a value of a variable is needed by a manager it sends a get-request PDU with the OID of that variable, and a NULL value. The managed element then responds by sending a get-response PDU that contains the same OID, the base type of the variable, and the current value. Figure 4 shows an example of real data contained in an SNMPv1 GetResponse PDU.

The first 20 bytes contain the IPv4 4 header. The next 8 bytes contain the UDP header. The last two bytes of the UDP header contain the UDP checksum (D3 65). The next four bytes 30 82 00 3E are the beginning of the SNMP message: 30 is SEQUENCE, and 82 00 3E is the length of the data in the SEQUENCE in bytes (62). The data in the SEQUENCE is the version (02 01 00) and the community string (04 06 70 75 62 6C 69 63). The last element in the SEQUENCE of the SNMPv1 message is the SNMP PDU.



The SNMP PDU itself is a tagged SEQUENCE: A2 indicates a GetResponse PDU and 82 00 2F is the length of the data in the GetResponse PDU in

bytes (47). The data in the GetResponse PDU is the request-id (02 04 6C F2 0C 5C), the error-status (02 01 00), and the error-index (02 01 00). Now follow the variables which contain the real payload: A SEQUENCE OF of length 31 (30 82 00 1F) containing a SEQUENCE of length 27 (30 82 00 1B). In it, the first object is an OID of length 19 (06 13) with the value 1.3.6.1.2.1.7.5.1.1.192.180.140.202.520. The last 6 bytes 40 04 87 B4 8C CA represent an IpAddress: 40 is the identification of the base type IpAddress, 04 is the length, and the next four bytes are the IP address value (135.180.140.202).

The example also shows an IP address embedded in an OID. The OID prefix resolves to the udpLocalAddress of the UDP-MIB, which is indexed by the udpLocalAddress 192.180.140.202 (81 40 81 34 81 0C 81 4A) and the udpLocalPort 520 (84 08). The SNMP packet actually shows an internal contradiction caused by a basic SNMP ALG since the udpLocalAddress encoded in the OID (192.180.140.202) is not equal to the value of the udpLocalAddress object instance (135.180.140.202).

Full Copyright Statement

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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