Network Working Group   J. Schoenwaelder, Ed.
Internet-Draft   Jacobs University
Intended status: Informational   H. Mukhtar
Expires: April 28, 2011   S. Joo
                                                                    ETRI
                                                                  K. Kim
                                                         Ajou University
                                                        October 25, 2010

SNMP Optimizations for Constrained Devices
draft-hamid-6lowpan-snmp-optimizations-03.txt

Abstract

   Simple Network Management Protocol (SNMP) is a widely deployed
   application protocol for network management and in particular network
   monitoring. This document describe the applicability of SNMP to
   constrained devices, e.g., nodes in Low-power and Lossy Networks. We
   discuss SNMP implementation techniques and we provide deployment
   considerations. Our discussion also covers the applicability of MIB
   modules to constrained devices.

Status of This Memo

   This Internet-Draft is submitted to IETF in full conformance with the
   provisions of BCP 78 and 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 April 28, 2011.

Copyright Notice

Table of Contents

1. **Introduction**

   The Simple Network Management Protocol (SNMP) is a datagram-oriented
   protocol operating in the application layer of the Internet protocol
   suite. The underlying framework consists of four basic components
   [RFC3410]:

   o several (typically many) managed nodes, each with an SNMP entity
      which provides remote access to management instrumentation
      (traditionally called an agent),

   o at least one SNMP entity with management applications (typically
      called a manager),

   o a management protocol used to convey management information
      between the SNMP entities, and

   o management information.

   The SNMP protocol is used to convey management information between
   SNMP entities such as managers and agents. SNMP is datagram-oriented
   and the implementations of SNMP can be very lightweight. The
   protocol is widely deployed for monitoring and troubleshooting
   purposes and it may fit constrained devices very well. The following
   features make SNMP suitable for constrained devices on Low-power and
   Lossy Networks (LLNs):

   o Protocol Maturity: SNMPv3 is a full IETF standard having a high
      degree of technical maturity with significant experiences in
      implementation and operation.

   o Data Naming: SNMP provides a hierarchical namespace utilizing
      object identifiers (OIDs) for data naming purposes. The data
      accessible via SNMP is described by Management Information Bases
      (MIB modules). These MIB modules can either be standardized or
      specific to certain enterprises.

   o Network Management: SNMP is widely used for network management and
      it is the Internet community's de facto network management and
      monitoring protocol. As a consequence, it makes sense to utilize
      SNMP also for the management and in particular monitoring of
      resource constrained networks. Network management is also stated
      as one of the goals in [RFC4919].

   o Data Retrieval: SNMP employs a trap-directed polling scheme in
      which data is being requested by a manager from the agents. In
      addition, SNMP supports a push model in which data is sent from
      agents to the managers without a prior request. Trap-directed

polling refers to a mode where polling is used with relatively
long polling intervals but agents can send notifications in order
to notify a manager of events that might require changes to the
polling strategy.

o Security: SNMPv3 can provide both message-level and transport-
   level security. SNMPv3 defines User based Security model (USM)
   [RFC3414] for message-driven security; and transport-based
   security model (TSM) [RFC5591] for transport-driven security. TSM
   makes it possible to use existing security protocols such as
   Transport Layer Security (TLS) [RFC5246] and the Datagram
   Transport Layer Security (DTLS) Protocol [RFC4347] with SNMPv3.
   The modular design of SNMPv3 also allows new security and access
   control protocols to be added to it.

o Access control: SNMP provides standard mechanisms to control
   access to information [RFC3415].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119].

2. SNMP Features and Overhead Considerations

This section first explains some less widely known SNMP concepts
before discussing message sizes.

2.1. SNMP Contexts

Each SNMP entity is composed of a single SNMP engine, which is
identified by an SNMP engine identifier. A context is a collection
of management information accessible by an SNMP entity. An SNMP
entity has access to one or more contexts where each context is
uniquely identified by its context name. In order to identify an
individual item of management information within a management domain,
the SNMP entity's context is identified first (using the
contextEngineID and contextName) and this is followed by the object
type and instance. For further details, see [RFC3411].

2.2. SNMP Proxies

The term 'proxy' in SNMP has a restrictive meaning. A proxy refers
to a proxy forwarder application which forwards SNMP messages to
other SNMP engines and forwards the response to such previously
forwarded messages back to SNMP engine from which the original
message was received [RFC3413]. The forwarding decision is based on
contexts and it is taken irrespectively of the management objects
being accessed. Thus, an SNMP proxy can be used to forward messages
from one transport to another, or to translate SNMP messages from one
version to another version.

The SNMP proxy cannot be used for translation of SNMP requests into
operations of a non-SNMP management protocol and it cannot be used
for supporting aggregated objects. Proxies depend on context
information and the forwarding of messages is independent of the
objects being accessed. To support aggregated objects, where the
value of one object depends upon multiple other remote items, special
MIB modules and sub-agent protocols are used instead of proxies.

2.3. SNMP Subagents

In order to support modular systems, SNMP agents often do not
implement all MIB objects internally. Instead, the SNMP agent is
delegating the access to the instrumentation to other processes,
called subagents. A special purpose protocol is used between the
SNMP agent and its subagents. The Agent Extensibility Protocol
(AgentX) is a standard subagent access protocol [RFC2741]

2.4. Maximum Message Sizes

An SNMPv3 message contains the msgMaxSize field, which is used to
communicate the maximum message size a sender is able to receive.
The response to a request should not exceed the maximum message size
of the requesting SNMP entity. The minimum required maximum message
size to implement is transport model specific. For SNMP over UDP,
the size is 484 octets.

2.5. SNMP Message Formats

SNMPv1 [RFC1157] is the first version of SNMP and it reached the IETF
full standard status in 1990. The protocol operation consisted of
Get and Get-Next, for data retrieval, Trap for event notification,
the Set for configuration. SNMPv1 security uses clear-text community
string authentication, which is easy to break. Access control is
provided with SNMP MIB views. SNMPv2c is an improvement over SNMPv1
which introduced new data retrieval and event notificaiton
operations, i.e., Get-Bulk and Inform. It also introduced improved
error handling for Set operations. SNMPv2c could only reach
Experimental status.

SNMPv3, STD 62, [RFC3411] [RFC3412] [RFC3413] [RFC3414] [RFC3415]
[RFC3416] [RFC3417] [RFC3418], supports all the aforementioned data
retrieval and configuration options of SNMPv1 and SNMPv2c. The
SNMPv3 framework is modular in order to enhance extensibility.
Moreover, SNMPv3 supports authentication and data integrity and an
additional privacy option for confidentiality. After SNMPv3 became a
full standard, SNMPv1 and SNMPv2c were declared Historic due to their
weak security features. However, SNMPv3 can coexist with the earlier
versions of SNMP [RFC3584].

2.6. SNMPv3 Security Overhead

SNMP security can be supported by two different approaches, i.e.,
message-driven security and transport-driven security. With message-
driven security, SNMP provides its own security where the security
parameters are passed within each SNMP message. On the other hand,
transport-driven security enables operators to leverage existing
secure transport protocols. Security is provided at the transport
layer, usually establishling a security session.

The User-based Security Model [RFC3414] is a shared secret scheme,
which provides message-driven security. Although it utilizes
existing mechanisms, it is designed to not depend on other security
infrastructures. As a consequence, it provides its own security
processing and has its own key management infrastructure. The
operator configures secrets (authentication and encryption keys) in

the SNMP engines. Messages can be authenticated, or authenticated
and encrypted.

The Transport Security Model (TSM) [RFC5591] enables operators to
leverage existing security infrastructures. TSM allows security to
be provided by an external secure transport protocol and as such
enables the use of existing security mechanisms, such as Transport
Layer Security (TLS) [RFC5246], Datagram Transport Layer Security
(DTLS) Protocol [RFC4347], and the Secure Shell (SSH) Protocol
[RFC4251].

In transport-driven protocols, DTLS, which is UDP based, can be
considered for constrained networks since it does not require TCP.
[RFC5953] details how DTLS can be used with SNMPv3/TSM. The DTLS
transport protocol involves an initial handshake to establish a
session. Upon successful session establishment, the security related
session parameters are cached in the client and the server for the
duration of the session instead of being sent in all messages.

The minimum message size for SNMPv3 with USM (SNMPv3/USM) is 67
octets whereas the minimum message size for SNMPv3 with TSM (SNMPv3/
TSM) utilizing DTLS is 46 octets (59 octets if the DTLS header is
included). The minimum message size for the historic SNMPv1 message
format is 20 byte. The details of the calculation can be found in
Appendix A. TSM may involve additional session establishment costs
consisting of the initial handshake and the caching of transport
parameters. The tradeoff between the message size and session
overhead should be kept in mind while designing applications.

3. SNMP Agent Implementation Considerations

   This section covers SNMP agent implementation considerations for
   constrained devices.

3.1. Access Control

   The Local Configuration Datastore (LCD), which contains access rights
   and policies of an SNMP entity, need not be configured remotely. It
   is recommended to have permanent access control tables on the nodes.
   The implementers should keep the authorization tables as compact as
   possible to reduce the memory and code size overhead. Compact
   permanent authorization tables on the nodes can, for example, provide
   read-only and read-write access to the management instrumentation on
   the node at almost zero processing cost since the SNMP agents may not
   support instance level access control granularity to further reduce
   performance cost.

   A minimal View-based Access Control Model (VACM) implementation only
   provides a static view granting access to all MIB objects. The
   access rights are statically configured to either grant full read
   access or full read and write access. There is only support for the
   default context. Such a simplified implementation processes the
   isAccessAllowed() ASI [RFC3415] as follows:

   1) If the viewType is "write", the securityName is "w" (for any
       securityModel and any securityLevel), and the contextName is "",
       then grant access to the requested variable.

   2) Otherwise, if the viewType is either "read" or "notifiy", the
       securityName is "r" (for any securityModel and any
       securityLevel), and the contextName is "", then grant access to
       the requested variable.

   3) Otherwise, return an errorIndication (noAccessEntry) to the
       calling module.

   An implementation should provide the following MIB objects (note that
   all values are permanent):

```
vacmContextName."" = ""

vacmGroupName.0."r" = "r"
vacmGroupName.0."w" = "w"
vacmSecurityToGroupStorageType.0."r" = 5 (readOnly)
vacmSecurityToGroupStorageType.0."w" = 5 (readOnly)
vacmSecurityToGroupStatus.0."r" = 1 (active)
vacmSecurityToGroupStatus.0."w" = 1 (active)

vacmAccessContextMatch."r"."".0.1 = 1 (exact)
vacmAccessContextMatch."w"."".0.1 = 1 (exact)
vacmAccessReadViewName."r"."".0.1 = "a"
vacmAccessReadViewName."w"."".0.1 = "a"
vacmAccessWriteViewName."r"."".0.1 = "a"
vacmAccessWriteViewName."w"."".0.1 = "a"
vacmAccessNotifyViewName."r"."".0.1 = "a"
vacmAccessNotifyViewName."w"."".0.1 = "a"
vacmAccessStorageType."r"."".0.1 = 5 (readOnly)
vacmAccessStorageType."w"."".0.1 = 5 (readOnly)
vacmAccessStatus."r"."".0.1 = 1 (active)
vacmAccessStatus."w"."".0.1 = 1 (active)

vacmViewTreeFamilyMask."a".2.1.3 = ""
vacmViewTreeFamilyType."a".2.1.3 = 1 (included)
vacmViewTreeFamilyStorageType."a".2.1.3 = 5 (readOnly)
vacmViewTreeFamilyStatus."a".2.1.3 = 1 (active)
```

4. SNMP Manager Implementation Considerations

   This section covers SNMP manager implementation considerations for
   6LoWPAN.

4.1. Polling, Pushing, and Trap-directed Polling

   In Sensor networks, polling can be reactive or proactive. Data
   gathering or event reporting sensors may 'push' their information
   towards the managers or they may wait for a manager to 'pull' the
   information through a request.

   When the demand for data is relatively high, push mechanisms are
   deployed in order to save energy cost where the data flows from
   managed entities towards the managers. SNMP notifications are a
   realization the push based model in which data is sent to the manager
   without a prior request. Data can be reported periodically from the
   SNMP agent to the manager through SNMP notifications and the
   notifications can take the advantage of SNMP security and access
   control features to ensure the access to legitimate users along with
   confidentiality and integrity of the data. The SNMP Inform PDU
   requires a response back from the receiving manager and it can be
   used in applications in which reliability is important.

   The use of notifications is recommended for data flows from sensors
   to the manager and also for the scenarios where multiple nodes
   generate the same information.

4.2. Support for SNMP Proxies

   The SNMP proxy forwarder application resides on an intermediate SNMP
   entity (e.g. an SNMP entity on a management server or an edge router
   in case of 6LoWPAN). The proxy forwarder registers each context to
   which it wishes to forward messages. After the remote context is
   registered, the managers send messages to the proxy forwarder's
   engine with the context information of the remote host. The proxy
   forwarder forwards the message to the remote context. Upon reception
   of a response from the remote host, it forwards the response back to
   the manager.

   In 6LoWPAN networks proxies may be used to change message encoding,
   or they may be used to translate between SNMP versions, or they may
   be used to change the security domain at the 6LoWPAN side of the
   network.

5. SNMP Deployment Considerations

Following are a list of considerations for deployment of SNMP in
6LoWPANs.

5.1. Naming Issues

In order to reduce the message overhead, the managers are advised to
use short values for Engine Identifiers. The minimum length for an
Engine Identifier is 5 octets. The managers may generate and assign
the Engine identifiers using the 16-bit short address or the 64-bit
IEEE EUI-64 addresses of a node. Context name is an administratively
assigned octet string that names a context. In order to reduce the
message size overhead the length of the string should be kept short.
The default context is identified by a a zero-length context name.

5.2. SNMP Protocol Operations

SNMP supports four basic data retrieval operations i.e. GetRequest-
PDU, GetNextRequest-PDU, GetBulkRequest-PDU [RFC3416]. The
GetRequest-PDU is useful for retrieving well known scalar data,
whereas the GetNextRequest-PDU and GetBulkRequest-PDU operations are
particularly advantageous for retrieving dynamically changing tabular
data. The SNMPv2-Trap-PDU and InformRequest-PDU can be used for
push-based data retrieval, in which periodic or event-based
notifications are sent to the managers.

During the processing of a GetBulkRequest-PDU operation, the agent
can decide the number of objects to include in response. For
requesting objects the manager has to consider the underlying packet
size constraints. Also, the number of objects in the variable-
binding in request messages and max-repeaters field of GetBulk
operation should be selected keeping the constraint in mind.

5.3. Timeouts and Retransmissions

In 6LoWPANs, the SNMP message may be fragmented or may encounter more
latency because of underlying wireless link. The value of timeouts
should be adjusted on the manager side by considering the link
characteristics so that SNMP does not timeout between queries. In
some cases the number of retries may also be adjusted to cater for
link characteristics.

5.4. Polling Intervals

Similarly, in order to reduce the amount of polling, the polling
interval should be increased for less time critical data. 6LoWPANs
are energy constrained networks and excessive polling is not

recommended.

5.5. Caching Issues

Caching the important information can save the transmission cost e.g.
caching the snmpEngineID would save the traffic overhead of EngineID
discovery mechanisms. It is recommended that the EngineID should be
cached in order to reduce the transmission cost. In case of TSM,
caching the transport parameters can reduce the message sizes.

6. Applicable MIB Modules

   This section describes some MIB modules relevant for constrained
   devices and it provides guidelines for authors of MIB modules that
   can be used efficiently in constrained networks.

6.1. Applicable Standardized MIB Modules

   Below is a list of MIB modules that may be applicable to a
   constrained device:

   o The SNMPv2-MIB [RFC3418] MUST be implemented as it provides basic
      information about the SNMP agent and crucial objects that allow to
      detect continuities.

   o The IF-MIB [RFC2863] SHOULD be implemented in order to provide
      basic statistics about the network interfaces of the constrained
      device. [TODO: Define what is really essential from the IF-MIB.]

   o Devices supporting IPv4 or IPv6 SHOULD implement the IP-MIB
      [RFC4293]. [TODO: Define what is really essential from the IP-
      MIB.]

   o Devices supporting UDP SHOULD implement the UDP-MIB [RFC2013].
      [TODO: Define what is really essential from the UDP-MIB.]

   o Devices supporting IPv6 over 802.15.4 (6LoWPAN) SHOULD implement
      the LOWPAN-MIB. [TODO: There is no LOWPAN-MIB yet.]

   o Devices supporting the RPL routing protocol SHOULD implement the
      RPL-MIB. [TODO: There is no RPL-MIB yet.]

   o Devices supporting sensors MAY implement the ENTITY-SENSOR-MIB
      [RFC3433], which defines objects for reading physical sensors
      (e.g., the current value of the sensor, the operational status of
      a sensor, or the data units precision associated with a sensor).
      The ENTITY-SENSOR-MIB depends on the ENTITY-MIB [RFC4133]. [TODO:
      Define what is really essential from the ENTITY-MIB.]

6.2. MIB Design Guidelines for Low Overhead

   When defining MIB modules, the MIB designers should avoid using long
   OIDs by avoiding unnecessary data hierarchies. Moreover, complex
   indexing schemes should be avoided in order to keep the overhead
   resulting from instance identifiers as small as possible.

[7](). Conclusion

SNMP can be very useful protocol for constrained devices with
significant implementation and operational experiences. The SNMP
standards allow for memory and CPU efficient implementations. The
utilization of secure transports such as DTLS can reduce the overhead
of message-based security mechanisms.

8. IANA Consideration

   TBD

9. Security Considerations

TBD

[10](#). References

[10.1](#). Normative References

[RFC1157]   Case, J., Fedor, M., Schoffstall, M., and J. Davin,
            "Simple Network Management Protocol (SNMP)", STD 15,
            RFC 1157, May 1990.

[RFC2013]   McCloghrie, K., "SNMPv2 Management Information Base for
            the User Datagram Protocol using SMIv2", RFC 2013,
            November 1996.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2863]   McCloghrie, K. and F. Kastenholz, "The Interfaces Group
            MIB", RFC 2863, June 2000.

[RFC3411]   Harrington, D., Presuhn, R., and B. Wijnen, "An
            Architecture for Describing Simple Network Management
            Protocol (SNMP) Management Frameworks", STD 62, RFC 3411,
            December 2002.

[RFC3412]   Case, J., Harrington, D., Presuhn, R., and B. Wijnen,
            "Message Processing and Dispatching for the Simple Network
            Management Protocol (SNMP)", STD 62, RFC 3412,
            December 2002.

[RFC3413]   Levi, D., Meyer, P., and B. Stewart, "Simple Network
            Management Protocol (SNMP) Applications", STD 62,
            RFC 3413, December 2002.

[RFC3414]   Blumenthal, U. and B. Wijnen, "User-based Security Model
            (USM) for version 3 of the Simple Network Management
            Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC3415]   Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based
            Access Control Model (VACM) for the Simple Network
            Management Protocol (SNMP)", STD 62, RFC 3415,
            December 2002.

[RFC3416]   Presuhn, R., "Version 2 of the Protocol Operations for the
            Simple Network Management Protocol (SNMP)", STD 62,
            RFC 3416, December 2002.

[RFC3417]   Presuhn, R., "Transport Mappings for the Simple Network
            Management Protocol (SNMP)", STD 62, RFC 3417,
            December 2002.

   [RFC3418] Presuhn, R., "Management Information Base (MIB) for the
              Simple Network Management Protocol (SNMP)", STD 62,
              RFC 3418, December 2002.

   [RFC3433] Bierman, A., Romascanu, D., and K. Norseth, "Entity Sensor
              Management Information Base", RFC 3433, December 2002.

   [RFC4133] Bierman, A. and K. McCloghrie, "Entity MIB (Version 3)",
              RFC 4133, August 2005.

   [RFC4293] Routhier, S., "Management Information Base for the
              Internet Protocol (IP)", RFC 4293, April 2006.

   [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model
              for the Simple Network Management Protocol (SNMP)",
              RFC 5591, June 2009.

   [RFC5953] Hardaker, W., "Transport Layer Security (TLS) Transport
              Model for the Simple Network Management Protocol (SNMP)",
              RFC 5953, August 2010.

10.2. Informative References

   [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart,
              "Introduction and Applicability Statements for Internet-
              Standard Management Framework", RFC 3410, December 2002.

   [RFC2741] Daniele, M., Wijnen, B., Ellison, M., and D. Francisco,
              "Agent Extensibility (AgentX) Protocol Version 1",
              RFC 2741, January 2000.

   [RFC3584] Frye, R., Levi, D., Routhier, S., and B. Wijnen,
              "Coexistence between Version 1, Version 2, and Version 3
              of the Internet-standard Network Management Framework",
              BCP 74, RFC 3584, August 2003.

   [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6
              over Low-Power Wireless Personal Area Networks (6LoWPANs):
              Overview, Assumptions, Problem Statement, and Goals",
              RFC 4919, August 2007.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246, August 2008.

   [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security", RFC 4347, April 2006.

   [RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH)

Protocol Architecture", RFC 4251, January 2006.

Appendix A. Calculation of Minimum Message Sizes

   A simple way to estimate the size (in octets) of an SNMP variable
   binding is the following formula (where |OID| denotes the number of
   subidentifier of an OID):

             sizeof(VarBind) = (2 + |OID|) + (2 + 2)

   The assumption here is that every OID subidentifier encodes into a
   single octet. An additional octet is needed for the OID tag and the
   OID length. Since most values are 32-bit numbers, we calculate one
   octet for the value tag, one octet for the value length, and 2 octets
   on average for the value itself. While the BER encoding of 32-bit
   unsigned numbers may require 5 octets, in general small numbers tend
   to dominate due to their usage in enumerations or many error counters
   staying close to zero. For sysUpTime.0 (1.3.6.1.2.1.1.3.0), we
   calculate 15 octets as the typical varbind encoding size of
   sysUpTime.0.

   For the PDU sequence [RFC3416], we calculate the following:

     PDU   2 octets
       request-id   3 octets
       error-status   3 octets
       error-index   3 octets
       variable-bindings   2 octets
                                     ----
                                    13 octets

   A PDU carrying a sysUpTime.0 varbind thus requires about 13+15 = 28
   octets.

   For the ScopedPDU sequence used by SNMPv3 [RFC3412], we calculate the
   following:

     ScopedPDU   2 octets
       contextEngineID   7 octets
       contextName   2 octets
       PDU   13 octets
                                     ----
                                    24 octets

   A scoped PDU carrying a sysUpTime.0 varbind thus requires about 24+15
   = 39 octets.

   For the HeaderData sequence used by SNMPv3 [RFC3412], we calculate
   the following:

```
      HeaderData                          2 octets
        msgID                             3 octets
        msgMaxSize                        4 octets
        msgFlags                          3 octets
        msgSecurityModel                  3 octets
                                          ---------
                                         15 octets
```

A.1. SNMPv3/USM Minimum Message Size

The minimum size of an SNMPv3/USM message can be calculated as
follows:

```
      SNMPv3Message (USM)                 2 octets
        msgVersion                        3 octets
        msgGlobalData (HeaderData)       15 octets
        msgSecurityParameters            24 octets  (UsmSecurityParameters)
        msgData (ScopedPDU)              24 octets
                                          ---------
                                         67 octets


      UsmSecurityParameters               2 octets
        msgAuthoritativeEngineID          7 octets
        msgAuthoritativeEngineBoots       3 octets
        msgAuthoritativeEngineTime        3 octets
        msgUserName                       3 octets
        msgAuthenticationParameters       2 octets
        msgPrivacyParameters              2 octets
                                          ---------
                                         22 octets
```

A complete SNMPv3/USM message to retrieve sysUpTime.0 therefore
requires 67+15 = 82 octets.

A.2. SNMPv3/TSM Minimum Message Size

The minimum size of an SNMPv3/TSM message can be calculated as
follows:

```
     SNMPv3Message (TSM)               2 octets
       msgVersion                      3 octets
       msgGlobalData (HeaderData)     15 octets
       msgSecurityParameters           2 octets   (TsmSecurityParameters)
       msgData (ScopedPDU)            24 octets
                                      ---------
                                      46 octets

     TsmSecurityParameters             2 octets
                                      ---------
                                       2 octets
```

A complete SNMPv3/TSM message to retrieve sysUpTime.0 therefore
requires 46+15 = 61 octets. If the secure transport used by SNMPv3/
TSM is DTLS, then the encoded message is wrapped in a DTLS record,
which adds the following number of octets:

```
     type                            1 octets
     version                         2 octets
     epoch                           2 octets
     sequence_number                 6 octets
     length                          2 octets
                                    ---------
                                    13 octets
```

The size of the resulting DTLS record is 61 + 13 = 74 octets.

A.3. SNMPv1/SNMPv2c Minimum Message Size

The minimum size of an SNMPv3/TSM message can be calculated as
follows (assuming a one character community string):

```
     SNMPv1Message                    2 octets
       version                        3 octets
       community                      3 octets
       data (PDU)                    13 octets
                                     ---------
                                     21 octets
```

A complete SNMPv3/TSM message to retrieve sysUpTime.0 therefore
requires 21+15 = 36 octets. Note, however, that SNMPv1/SNMPv2c does
not provide security nor does it provide direct support for proxying.

Appendix B. Implementation and Deployment Models

   There are four fundamentally different implementation / deployment
   models for SNMPv3 in constrained networks.

B.1. SNMP End-to-End Model

   The SNMP manager talks SNMPv3 end-to-end to the 6LoWPAN nodes. In
   this model, existing management tools can be reused and only a few
   adaptations may be needed by specifying suitable deployment
   parameters through an applicability statement.

        Manager <--> 6LoWPAN
                                 SNMPv3   nodes

   The characteristics of this solution can be summarized as follows:

   + Straightforward access to individual 6LoWPAN nodes

   + Reuse of existing deployed SNMP-based tools

   o End-to-end security and end-to-end key management

   - Message size and potential fragmentation issues

   - 6LoWPAN nodes must run an SNMP engine

   - Trap-directed polling nature of SNMP has high energy costs

B.2. SNMP Proxy Model

   The SNMP manager talks SNMPv3 to an SNMP proxy residing on a 6LoWPAN
   edge router (ER). Existing management tools (as long as they are
   proxy aware, which is not generally true) can be reused.

        Manager <--------> SNMP Proxy <-----------------> 6LoWPAN
                 SNMPv3   (6LoWPAN ER)   SNMPv3   nodes

   The characteristics of this solution can be summarized as follows:

   + Alternate transport encoding can reduce message sizes

   o Indirect access to individual 6LoWPAN nodes

   o Reuse of existing SNMP-based tools supporting proxies

   o Two security domains, different key management schemes

   - 6LoWPAN nodes must run an SNMP engine

   - Trap-directed polling nature of SNMP has high energy costs

B.3. SNMP Subagent Model

   The SNMP manager talks SNMPv3 to an extensible SNMP agent residing on
   the 6LoWPAN edge router. This agent uses a subagent protocol (e.g.,
   AgentX [RFC2741]). The current standard subagent protocol is not
   necessarily suitable for 6LoWPAN networks since it assumes a reliable
   stream-oriented transport and an adaptation of a subagent protocol
   may be required.

```
     Manager <-------->  SNMP Agent  <----------------> 6LoWPAN
              SNMPv3   (6LoWPAN ER)  SubAgent Protocol   nodes
```

   The characteristics of this solution can be summarized as follows:

   + Alternate transport encoding can reduce message sizes

   o Indirect access to individual 6LoWPAN nodes

   o Reuse of existing SNMP-based tools supporting proxies

   o Two security domains, different key management schemes

   + 6LoWPAN nodes must run an SNMP subagent

   - Trap-directed polling nature of SNMP has high energy costs

B.4. SNMP Data-Fusion Model

   The SNMP manager talks SNMPv3 to an SNMP agent residing on the
   6LoWPAN edge router. This agent uses a different protocol (e.g., a
   protocol such as CoAP) to retrieve information from the 6LoWPAN
   network. In the ideal case, the protocol supports caching and in
   network data aggregation.

```
     Manager <-------->  SNMP Agent  <----------------> 6LoWPAN
              SNMPv3   (6LoWPAN ER)  CoAP Data Fusion    nodes
```

   The characteristics of this solution can be summarized as follows:

   + Indirect access to individual 6LoWPAN nodes

   + Leveraging a cache-aware data fusion protocol

   + SNMP agent acting as a cache, no expensive polling

   o Reuse of existing SNMP-based tools supporting contexts

   o Two security domains, different key management schemes

Appendix C. **Example: Contiki SNMP**

   Contiki-SNMP is an SNMP implementation for the Contiki operating
   system, designed to run on Atmel Raven boards (8-bit microcontroller
   running at 20 MHz with 16K of RAM and 128K of Flash). Contiki-SNMP
   supports SNMP messages up to 484 octets length. The currently
   supported message types are Get, GetNext, and Set. The currently
   supported message versions are SNMPv1 and SNMPv3/USM. The
   implementation provides an API to define and configure managed
   objects (MIB variables). The USM implementation supports HMAC-MD5-96
   and CFB128-AES-128.

   If both SNMPv1 and SNMPv3 are enabled, the code uses 31220 octets of
   ROM (around 24% of the available ROM) plus 235 octets of statically
   allocated RAM. With only SNMPv1 enabled, the code uses 8860 octets
   of ROM (around 7% of the available ROM) plus 43 bytes of statically
   allocated RAM. Leveraging the AES hardware support of the 802.15.4
   transceiver will significantly reduce the footprint of the SNMPv3
   option.

   The heap usage is not more than 910 octets for processing an SNMPv1
   message. About 16 octets are used for each managed object
   implemented. If a managed object is of a string-based type,
   additional heap storage space is used to store the value.

   The maximum observed stack usage is show in Table 1.

```
         +---------+---------------+-----------------+
         | Version | Security level | Max. stack size |
         +---------+---------------+-----------------+
         | SNMPv1  |       -       |      688 octets |
         |         |               |                 |
         | SNMPv3  |  noAuthNoPriv |      708 octets |
         |         |               |                 |
         | SNMPv3  |   authNoPriv  |     1140 octets |
         |         |               |                 |
         | SNMPv3  |    authPriv   |     1144 octets |
         +---------+---------------+-----------------+
```

                  Table 1: Maximum observed stack usage

   For SNMPv3/USM noAuthNoPriv messages and SNMPv1 messages, the round-
   trip latency is dominated by the data transfer tim of the 802.15.4
   radio. For SNMPv3/USM authPriv messages, the processing time is
   almost the same as the data transmission delay. The authNoPriv
   security level is slightly faster.

Appendix D. Change Log

D.1. Changes from -02 to -03

Broadened the scope of the document to discuss SNMP on constrained devices, not limited to 6LoWPAN networks.

1. Added a data fusion protocol scenario.

2. Reorganization of the text.

3. Reorganization of Section 2.4.

4. Addition of Appendix C.

5. Added details about minimal VACM implementation.

6. Started a discussion of relevant MIB modules.

D.2. Changes from -01 to -02

The draft now covers applicability of SNMPv3 for 6LoWPANs. The focus of the draft is shifted towards supporting SNMPv3 'as is' in 6LoWPANs.

1. Added SNMP Agent Implementation Considerations for 6LoWPANs.

2. Added SNMP Manager Implementation Considerations for 6LoWPANs.

3. Added the Deployment Considerations for 6LoWPANs.

4. Added the Applicable MIB modules for 6LoWPANs.

5. Moved SNMP Deployment Models to Appendix.

6. Removed the section on Packet Compression.

Authors' Addresses

    Juergen Schoenwaelder (editor)
    Jacobs University Bremen
    Campus Ring 1
    Bremen 28725
    Germany

    Phone: +49 421 200-3587
    EMail: j.schoenwaelder@jacobs-university.de

    Hamid Mukhtar
    ETRI
    USN Research Division, ETRI, 161 Gajeong-dong, Yuseong-gu
    Daejeon 305-350
    KOREA

    Phone: +82 42 860 5435
    EMail: hamid@etri.re.kr

    Seong-Soon Joo
    ETRI
    USN Research Division, ETRI, 161 Gajeong-dong, Yuseong-gu
    Daejeon 305-350
    KOREA

    Phone: +82 42 860 6333
    EMail: ssjoo@etri.re.kr

    Kim, Ki Hyung
    Ajou University
    San 5 Wonchun-dong, Yeongtong-gu
    Suwon-si, Gyeonggi-do 442-749
    KOREA

    Phone: +82 31 219 2433
    EMail: kkim86@ajou.ac.kr