Extending SNMP to Support Bulk MIB Data Transfers

Presented by J. Schönwälder on behalf of the

Network Management Research Group (NMRG)

<URL:http://www.irtf.org/charters/management.html>

Summary:

1. SNMP over TCP transport mapping
2. SNMP lossless payload compression
3. `get-subtree` and linked `response` PDUs

SNMP over TCP Transport Mapping

Summary:

- The originator of a request/response transaction chooses the transport protocol for the entire transaction.
- Command responder listen on TCP port 161 while notification receiver listen on TCP port 162.
- An SNMP entity supporting the SNMP over TCP transport mapping MUST be capable of accepting messages that are at least 8192 octets in size.
- All SNMP entities can close TCP connections at any point in time.

Status:

- Discussion of <draft-irtf-nmrg-snmp-tcp-04.txt> on the SNMPv3 mailing list.
- Compiled a list of issues which are dealt with in the next revision.
- UCD SNMP and CMU Linux SNMP both support the SNMP over TCP mapping.

Lossless SNMP Payload Compression

Requirements:

- Compression must happen before encryption if compression is used together with encryption.
- SNMP payload compression should support multiple compression algorithms. This implies that SNMP engines must be able to negotiate the compression algorithm they are using.
- Instead of carrying compression algorithm identifier in every protocol message, it seems more effective to indicate compression algorithms in a MIB module.
- Each SNMP message is compressed and decompressed by itself without any relation to other SNMP messages ("stateless compression").
- The size of a compressed SNMP message must never exceed the size of the uncompressed SNMP message ("non-expansion policy").
- The abstract syntax of compressed SNMP messages should be defined using ASN.1.
- It is desirable to define common compression algorithms in order to achieve interoperability.

Lossless SNMP Payload Compression Options

#1: Treat compression as an SNMPv3 encryption algorithm

+ No change required to the SNMPv3 specifications.
- Support of N encryption algorithms and M compression algorithms results in $N \cdot M$ possible combinations.
- Compression requires authentication since there is no noAuthPriv security level.
+ Compression of the complete `ScopedPDU`.
- Does not work with older versions of SNMP (SNMPv1, SNMPv2c).

#2: Indicate compression in the `msgFlags`

- Not strictly compliant to the current SNMPv3 specifications.
+ Combination of M compression and N encryption algorithms possible without having to define $N \cdot M$ algorithms.
+ Compression can be used with or without encryption or authentication.
+ Compression of the complete `ScopedPDU`.
- Does not work with older versions of SNMP (SNMPv1, SNMPv2c).

Lossless SNMP Payload Compression Options

#3: Wrap compressed data in a new PDU type

- Introduce a new PDU type for compressed payloads.

```
CompressedPDU ::= [42] IMPLICIT OCTET STRING
                          -- contains a compressed PDU
```

- Strictly compliant SNMPv1 and SNMPv3 implementations will simply drop compressed PDUs.

\- Not strictly compliant to the current SNMPv3 specifications.

\+ Combination of M compression and N encryption algorithms possible without having to define $N \cdot M$ algorithms.

\+ Compression can be used with or without encryption or authentication.

\- Compression of the PDU rather than the `ScopedPDU`.

\+ Works with every version of SNMP.

\+ Allows to indicate the compression algorithm in the compressed PDU.

Lossless SNMP Payload Compression Algorithms

Deflate:

- Deflate (gzip) compression of the payload yields very good compression ratios.
- Requires noticeable CPU resources on the sending entity.
- Compression of small messages must be avoided.
- Difficult to build compressed messages that fill up the `msgMaxSize`.

ODC:

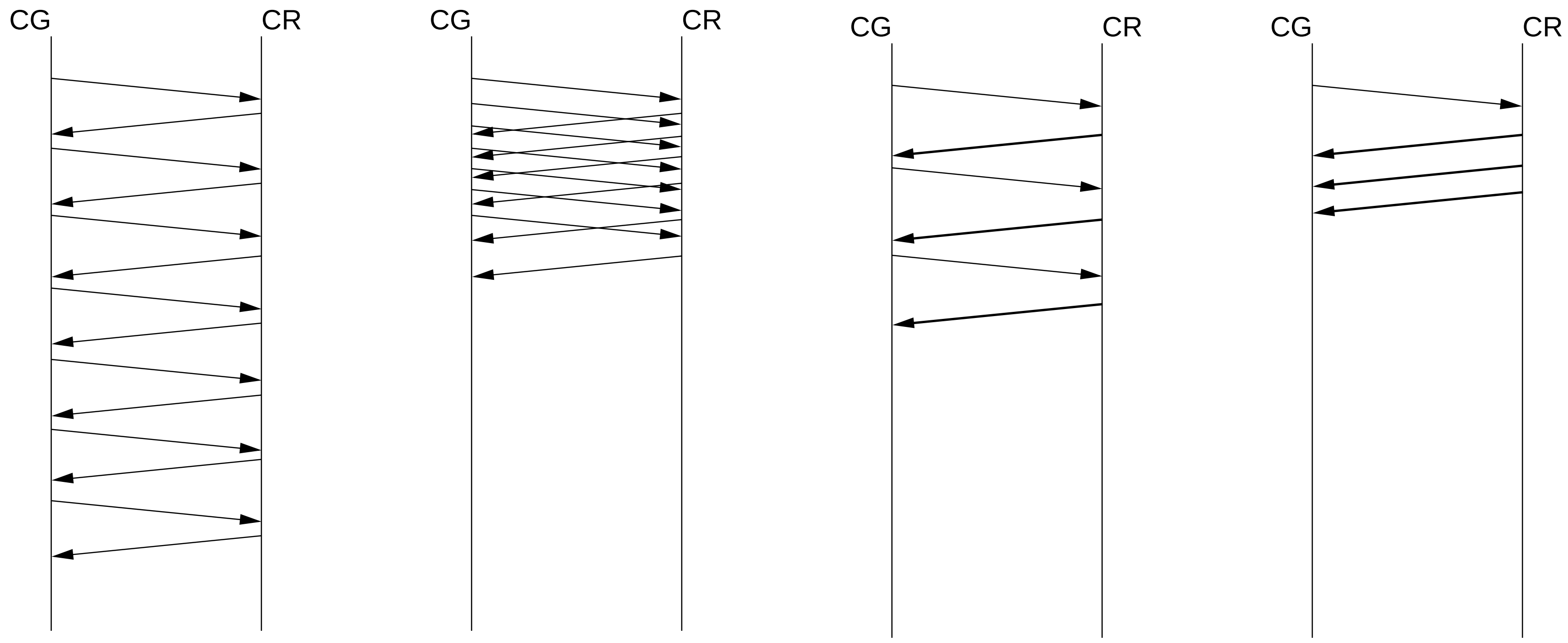
- The OID Delta Compression (ODC) algorithm has been designed to reduce the OID overhead inherent in SNMP messages.
- The general idea is to encode an OID as a delta to the previous OID.
- Algorithm can be efficiently integrated into the BER encoding functions.
- The achieved compression ratio is significant while the CPU cycles spend on the compression is almost irrelevant.

Lossless SNMP Payload Compression

Open Issues:

- The NMRG currently favors option #3 - but welcomes other opinions.
- Compression negotiation via a MIB module is easy for command generators.
- Notification originator applications need pre-configured information whether a target supports compression?
- The ODC mechanism can be extended into a specialized more effective encoding for SNMP payloads. But is this useful?

Get-Subtree PDU and Linked Responses



(a) GetNext (row by row) (b) GetNext (rows in parallel) (c) GetBulk (d) GetSubTree

Get-Subtree PDU and Linked Responses

Details:

- The structure of `get-subtree` PDUs is similar to `get-bulk` PDUs, except that there is no `max-repetitions` parameter.
- The OIDs in the variable-bindings list identify the roots of the subtrees to be retrieved.
- Specifying columns of a table allows to retrieve a projection of the table.
- Linked `response` PDUs are `response` PDUs where the `error-index` contains a sequence number if and only if the `error-status` is `noError`.
- The last linked `response` PDU has a negative sequence number.
- A reliable transport such as SNMP over TCP will handle any communication errors.
- An unreliable transport such as SNMP over UDP requires that the command generator requests any missing data with e.g. the `get-bulk` PDU.

Performance

Measurement Environment

- 233 MHz Pentium running Linux and Linux CMU SNMP package
- All measurements taken over the loopback interface (!)
- Retrieval of complete MIB tree (480 variables)
  (SNMPv2-MIB, IP-MIB, TCP-MIB, UDP-MIB, IF-MIB, HOST-RESOURCES-MIB)
- The time needed to load MIB definitions has been substracted.
- Measured each feature separate from the other features.

Results:

- ODC compression gives an average compression ratio 38%.
- DEFLATE compression gives an average compression ratio of 74%.
- Simple `get-next` walk takes on average 450 ms to complete.
- A `get-bulk` walk takes on average 200 ms to complete.
- A `get-subtree` walk takes on average 100 ms to complete.

Comments:

- More measurements needed to get comparable results. (SNMP Stones?)