Lausanne Minutes

Meeting purpose:

- Discuss how the Internet management framework should evolve to allow for Bulk and Easy Access to MIBs.
- Constraints:
  - Current MIB definitions, and therefore ASN1, are here to stay.
  - Further, if solutions are to stay relatively close to the original SNMP framework:
    - Reuse agent instrumentation as much as possible. In particular, for current agent instrumentation that use some form of X-agent technology, our target should be to keep any changes confined to the master agent portion, leaving sub-agents as they are.
    - Keep VACM
- Results: an article in The Simple Times.

Structure of meeting (the 5 options that were on the blackboard)

1. Extend SNMP
2. HTTP
3. New protocol
4. LDAP
5. FTP

Option 1: Extend SNMP

a) Use TCP

- TCP port 161 is already reserved for SNMP
- TCP will not replace UDP, but will be used in conjunction instead.
- Choice between TCP ad UDP will be made on a per-transaction basis: example: for a get operation use UDP, for a getbulk operation use TCP.
- Problem with long messages:
  - SNMP uses definite form of BER.
  - With this form, the length of the BER message should be coded at the beginning of the message.
  - Before the message can be transmitted, it must be stored in a buffer to determine the length -> you need large buffers of an unknown max size.
  - PER seems to solve this.
  - Unfortunately, SNMP v3 security re-introduces similar problems. As a result, we'll have to live with large buffers. (note: after the meeting on the mailing list consensus was that some form of multiple related messages of limited size was not so bad after all).

b) Add compression

- add an encryption scheme that in fact does compression. All agreed that this could be useful: it does the job, fits into the current SNMPv3 framework.
- Filtering: to reduce the amount of information that needs to be transferred by carefully selecting at the information source only that particular information that is actually of interest. Filtering in this sense would have the net effect that less information needs to be transferred over the network.
  - Although the initial filtering proposal was to limit the filtering functionality, some believed that we would wind up with rich filtering functionality.
  - The result may be filtering schemes comparable to those in OSI.
  - According to some, CPU and memory developments are such that this would not be a problem, others claimed that bandwidth developments go faster than CPU and memory developments; therefore such type of filtering will always be a problem.
  - Conclusion on filtering: there was some consensus that filtering functionality should be in an intermediate system, so not down in the actual agent/device, and not at the top level in the manager, but in a system (intermediate level manager) in between.
- Use OID compression.

- OIDs are now always 'absolute'. What would reduce the amount of information to be transmitted significantly would be the equivalent of a file-system change directory notion. This would result in relative OIDs, that is, the common prefix of a group of OIDs would be sent only once.

c) New PDU types

- Get and Set config PDUs.
  - Why would we want these PDUs? Two reasons were identified:
    - a configuration of a device can be large and complex. One therefore wants to make a backup of it, to be able to restore the configuration to the device in case of some failure or breakdown.
    - One might want to take a configuration of one device, clone it, and send it to a number of similar devices (cloning).
  - Does a configuration that is retrieved from a device have to have some defined structure? Or can it be opaque, only meaningful to the device that generated the configuration. A defined structure for the configuration would allow editing of a configuration.
  - Should a configuration be cloneable? This issue differs from the previous one.
  - How do you know what is part of the configuration and what is not?
  - Should it be possible to have multiple configurations? Can there be subsets?

- Get Table / Subtree PDU.
  - We first discussed the need for such kind of PDU. The problem with get-bulk is that we can not predict where the end of the table will be. As a result, we have to guess a value for max-repetitions. If the value for maxrep is to low, we will have too many interactions, which cause a long delay. If the value for maxrep is to high, we will retrieve many objects that do not belong to the table.
  - An example of the use of get-subtree: A manager wants all scriptmib script-table entries owned by foo. This is a subtree of the whole MIB, and could be retrieved by a get-subtree protocol operation.
  - Issue: These operations could or couldn't do filtering. Consensus in Lausanne was that it is probably best not to have filtering for a get-table operation between an agent and a mid-level manager.
  - This operation can give problems in current agent implementations that use X-agent technology. The problem is that the master agent has no notion of tables; table boundaries are only known to the sub agent. A single table can be spread over multiple subagents, each subagent implements some columns. A consequence of these facts can be that implementing this operation for an existing X-agent subagent can either be very costly or altogether impossible. It might be required to extend or change the x-agent protocol(s) in use today; this would destroy our targeted 'small effort first step' property.
  - Issue in retrieving tables: holes in tables. Columns of different length will be returned by this operation. How to deal with this. Rows can be reconstructed at the receiving manager by matching the least-significant portions of the OIDs of the returned values.
  - Issue: do we want/need to retrieve tables row-by-row or column-by-column? Or should we allow both? Some said that options are things you don't want in these specifications. Furthermore, retrieving tables row-by-row is in line with current practice of retrieving tables (using a series of get-next operations on a list of varbinds) and gives agent implementers better chances of returning consistent (pseudo atomic) table rows. Retrieving tables column-by-column might very well result in not even individual rows to be consistent.

Option 2: HTTP

a) data representation: mime, BER/PER, proprietary
   1) HTTP should specify what kind of information is being transferred -> new mime-type needed.
   2) Mime introduces overhead, so it should only be used as an envelope around a complete SNMP message, not as a per-OID or per-varbind wrapper. For the latter purpose, either XML, BER or PER, or a proprietary encoding should be used.
      Here was some discussion on XML, opinions were different. Some argued that XML introduces additional complexity but does not add anything PER/BER don't already have. Others argued that XML becomes a defacto technology for everything; DTMF e.g. used XML, and that management should be based on defacto techniques, not on proprietary ones. More discussion is needed on XML; XML is probably best situated between the mid level manager and the upper level manager.

b) security (mapping VACM, firewalls)
   1) VACM should not be broken by the use of HTTP.
   2) Currently, firewalls are already configured to allow HTTP web traffic to pass. By some this was considered to be an advantage, because the burden of configuring firewalls would be lessened by this feature.
   3) There was some discussion on TCP vs. UDP: a TCP based protocol was first considered to be the only option if you want to pass a firewall, later consensus was that currently firewall products exist that implement UDP relays, and thus both transports can in principle be used across firewalls.

c) Addressing

There wasn't sufficient available time to discuss HTTP and addressing, no clear conclusions on this topic.

Options 3), 4) and 5) were hardly discussed given the lack of available time. For Option 5), the use of FTP, it was observed that an approach using ftp together with SNMP is currently being defined in the ATM accounting world.

Overall conclusions (brief)

- The group as a whole advocates a two step approach.
  - Step one is to in the short term build some credit within the Internet community, by tackling some limited, short-scoped problem.
  -
  - Step two is to start working on longer term Internet management framework issues.
- We'll try to put the beam work under the umbrella of the IRTF, as a IRTF working group. Juergen will write up a draft charter for this.
- Jean-Phillipe will start working on a draft paper to be published in the next issue of The Simple Times.