Generic vs. Specific
Simple Network Management Tools

Jürgen Schönwälder
<schoenw@informatik.uni-osnabrueck.de>

University of Osnabrück
Germany
SNMP in a Nutshell

- The Simple Network Management Protocol (SNMP) is used to access and manipulate simple typed variables organized in conceptual tables or groups of scalars.
SNMP in a Nutshell

- The Simple Network Management Protocol (SNMP) is used to access and manipulate simple typed variables organized in conceptual tables or groups of scalars.

- The semantics of the variables are specified in MIB modules which are written in the SMI data definition language (Structure of Management Information).
SNMP in a Nutshell

- The Simple Network Management Protocol (SNMP) is used to access and manipulate simple typed variables organized in conceptual tables or groups of scalars.

- The semantics of the variables are specified in MIB modules which are written in the SMI data definition language (Structure of Management Information).

- Each variable is uniquely named by an OID value (a sequence of numbers defining a path in a global registration tree).
SNMP in a Nutshell

- The Simple Network Management Protocol (SNMP) is used to access and manipulate simple typed variables organized in conceptual tables or groups of scalars.

- The semantics of the variables are specified in MIB modules which are written in the SMI data definition language (Structure of Management Information).

- Each variable is uniquely named by an OID value (a sequence of numbers defining a path in a global registration tree).

- SNMP operates on a (lexicographically) ordered list of variables (varbind list). Each element consists of an OID identifying a variable and its value.
Due to the simplicity of the operations, people call SNMP—the peek/poke/trap protocol of the Internet or the Turing machine for network management.
Due to the simplicity of the operations, people call SNMP
- *the peek/poke/trap protocol of the Internet* or
- *the turing machine for network management.*
SNMP Architecture

Traditional Agent

- MIB Instrumentation
- Access Control Subsystem
  - View-based Access Control
  - Notification Originator
  - Proxy Forwarder

- PDU Dispatcher

- Message Processing Subsystem
  - v1MP
  - v2cMP
  - v3MP
  - other MP

- Security Subsystem
  - Community Security Model
  - User-based Security Model
  - Other Security Model

- Security Subsystem

- Communication Network

Traditional Manager

- Command Generator
- Notification Receiver
- Notification Originator

- Message Processing Subsystem
  - v1MP
  - v2cMP
  - v3MP
  - other MP

- Security Subsystem
  - Community Security Model
  - User-based Security Model
  - Other Security Model

- Transport Mappings

- Communication Network

UDP, IPX
SNMPv3 Message Encoding
Observations

- The protocol operations are simple while the protocol itself is everything else than simple.
- You need good tools/libraries to invoke the rather simplistic SNMP operations.
- Once you can invoke SNMP operations, you need to write meaningful management procedures to overcome the peek/poke/trap abstraction level.
- Since humans can't remember OIDs, you need tools/libraries which help to avoid dealing with OIDs.
Observations

- The protocol operations are simple while the protocol itself is everything else than simple.
Observations

- The protocol operations are simple while the protocol itself is everything else than simple.

- You need good tools/libraries to invoke the rather simplistic SNMP operations.
Observations

- The protocol operations are simple while the protocol itself is everything else than simple.

- You need good tools/libraries to invoke the rather simplistic SNMP operations.

- Once you can invoke SNMP operations, you need to write meaningful management procedures to overcome the peek/poke/trap abstraction level.
Observations

- The protocol operations are simple while the protocol itself is everything else than simple.

- You need good tools/libraries to invoke the rather simplistic SNMP operations.

- Once you can invoke SNMP operations, you need to write meaningful management procedures to overcome the peek/poke/trap abstraction level.

- Since humans can’t remember OIDs, you need tools/libraries which help to avoid dealing with OIDs.
What can be done?

Approach #1: Extend scripting languages with SNMP APIs to allow people to easily script their own useful management applications on top of the simplistic SNMP operations.

- Perl extensions (snmp-perl, snmp-session)
- Tcl extensions (Tnm)

Approach #2: Build compilers that generate C stubs from MIB specifications which are easier to program with to create specific management applications.

- SNMP Command Line Interface (scli)
What can be done?

⇒ Approach #1: Extend scripting languages with SNMP APIs to allow people to easily script their own useful management applications on top of the simplistic SNMP operations.

Perl extensions (snmp-perl, snmp-session)

Tcl extensions (Tnm)

⇒ Approach #2: Build compilers that generate C stubs from MIB specifications which are easier to program with to create specific management applications.

SNMP Command Line Interface (scli)
What can be done?

⇒ Approach #1: Extend scripting languages with SNMP APIs to allow people to easily script their own useful management applications on top of the simplistic SNMP operations.

- Perl extensions (snmp-perl, snmp-session)
- Tcl extensions (Tnm)
- ...
What can be done?

- Approach #1: Extend scripting languages with SNMP APIs to allow people to easily script their own useful management applications on top of the simplistic SNMP operations.
  
  - Perl extensions (*snmp-perl*, *snmp-session*)
  - Tcl extensions (*Tnm*)
  - ...

- Approach #2: Build compilers that generate C stubs from MIB specifications which are easier to program with to create specific management applications.
What can be done?

⇒ Approach #1: Extend scripting languages with SNMP APIs to allow people to easily script their own useful management applications on top of the simplistic SNMP operations.

- Perl extensions (snmp-perl, snmp-session)
- Tcl extensions (Tnm)
- ...

⇒ Approach #2: Build compilers that generate C stubs from MIB specifications which are easier to program with to create specific management applications.

- SNMP Command Line Interface (scli)
Tnm extension for Tcl

- Tnm provides a generic SNMP API for Tcl
- Written entirely in C for good performance
- Tightly integrated into the Tcl event mechanism
- Scripts can talk to many devices simultaneously
- Traffic shaping for smoothing bulky message streams
- Several (generic) applications exist on top of Tnm (tkined, sgmospy, sbrowser, ...)
- Used by several companies to drive test suites
- Available since 1994, relative few changes since 1999
package require Tnm 3.0

proc walkproc {s stat vbl} {
    if {$stat == "noError"} {
        set i [Tnm::mib unpack [Tnm::snmp oid $vbl 0]]
        set o [Tnm::snmp value $vbl 0]
        set a [Tnm::snmp value $vbl 1]
        puts "[\$s cget -address] $i $o $a"
    }
}

puts "ADDRESS \t ADDRESS \t INTERFACE \t OPER \t ADMIN"
foreach host $argv {
    set s [Tnm::snmp generator -address $host]
    $s walk {ifOperStatus ifAdminStatus} { walkproc %S %E "%V" } }
Tnm::snmp wait
exit
package require Tnm 3.0

proc checkproc {s stat vbl} {
    if {$stat == "noError"} {
        puts "[$s cget -address]\t[$s cget -community]\t$vbl"
    }
}

proc check {hosts community} {
    foreach h $hosts {
        set s [Tnm::snmp generator -address $h -community $community \
                -version SNMPv2c -window 100 -delay 5 -timeout 2]
        $s get sysDescr.0 { checkproc %S %E "%V"; %S destroy }
    }
}

while {! [eof stdin]} { check $argv [gets stdin] }
Tnm::snmp wait
exit
Limitations of the Tnm Approach

- Not everyone is interested in writing Tnm scripts
- Not everyone is interested to understand the sometimes subtle semantics of MIB variables
- Administrators/operators prefer specific tools rather than generic tools
- Low-level APIs cause scripts to be fragile
- Scripts tend to be site specific and not portable
- Maintenance of Tcl scripts is no fun
Limitations of the Tnm Approach

- Not everyone is interested in writing Tnm scripts
- Not everyone is interested to understand the sometimes subtle semantics of MIB variables
- Administrators/operators prefer specific tools rather than generic tools
- Low-level APIs cause scripts to be fragile
- Scripts tend to be site specific and not portable
- Maintenance of Tcl scripts is no fun

⇒ Create specific rather than generic tools
⇒ Build infrastructure to do this efficiently
SNMP Command Line Interface

- Command line interface with runs locally
- Uses standard SNMP interactions and MIBs
- Interworks with devices produced by different vendors
- Commands are structured in a hierarchy
- Related commands are logically grouped into modes
- Select objects using names and regular expressions
- Support simple short-term monitoring activities
- Command editing/history and command aliases
- Default output format is optimized for human readability
- XML output format optimized for machine readability
show entity containment

ENTITY CLASS CONTAINMENT
1 chassis 7206VXR chassis, Hw Serial#: 21275454, Hw Revision: D
2 module  I- NPE 300 Card, Hw Serial#: 21275454, Hw Revision: D
3 container I- Chassis Slot
4 module   I- I/O FastEthernet (TX-ISL)
5 port     I- DEC21140A
6 container I- Chassis Slot
7 module   I- 2 Port Fast Ethernet/ISL 100BaseTX Port Adapter
8 port     I- AmdFE
9 port     I- AmdFE
10 container I- Chassis Slot
11 module  I- POS Port Adapter (SM)
12 port    I- Packet over Sonet
13 container I- Chassis Slot
14 container I- Chassis Slot
15 module  I- ATM Lite Port Adaptor (SM)
16 port    I- TI1570 ATM
17 container I- Chassis Slot
18 container I- Chassis Slot

(ciscobs.rz) scli >
monitor interface stats

Agent: ciscobs.rz:161 up 9 days 23:34:35 15:56:26
Descr: Cisco Internetwork Operating System Software IOS (tm) 7200 Software
IPv4: 6435 pps in 6408 pps out 6399 pps fwd 0 pps rasm 0 pps frag
UDP: 8 pps in 6 pps out
TCP: 0 sps in 0 sps out 0 con est 0 con aopn 0 con popn
Command: monitor interface stats

<table>
<thead>
<tr>
<th>INTERFACE</th>
<th>STATUS</th>
<th>I-BPS</th>
<th>O-BPS</th>
<th>I-PPS</th>
<th>O-PPS</th>
<th>I-ERR</th>
<th>O-ERR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 UUCN</td>
<td>1m</td>
<td>2m</td>
<td>3270</td>
<td>3152</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>FastEthernet0/0</td>
</tr>
<tr>
<td>2 UUCN</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>FastEthernet1/0</td>
</tr>
<tr>
<td>3 UUCN</td>
<td>10k</td>
<td>10k</td>
<td>50</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>FastEthernet1/1</td>
</tr>
<tr>
<td>4 UUCN</td>
<td>2m</td>
<td>1m</td>
<td>3197</td>
<td>3254</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>POS2/0</td>
</tr>
<tr>
<td>5 UDCN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ATM4/0</td>
</tr>
<tr>
<td>6 UD--</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>ATM4/0-atm layer</td>
</tr>
<tr>
<td>7 UD--</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>ATM4/0.0-atm subif</td>
</tr>
<tr>
<td>8 UDNN</td>
<td>0</td>
<td>------</td>
<td>0</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>ATM4/0-aal5 layer</td>
</tr>
<tr>
<td>9 UDNN</td>
<td>0</td>
<td>------</td>
<td>0</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>ATM4/0.0-aal5 layer</td>
</tr>
<tr>
<td>10 UUNN</td>
<td>0</td>
<td>2105</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Null0</td>
</tr>
<tr>
<td>11 UUNN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Loopback0</td>
</tr>
<tr>
<td>12 UUNN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Tunnel34</td>
</tr>
</tbody>
</table>
scli interface mode

set interface status <regexp> <status>
set interface alias <regexp> <string>
set interface notifications <regexp> <value>
set interface promiscuous <regexp> <bool>

show interface info [<regexp>]
show interface details [<regexp>]
show interface stack [<regexp>]
show interface stats [<regexp>]

monitor interface stats [<regexp>]

dump interface
scli nortel mode

create nortel bridge vlan <vlanid> <name>
delete nortel bridge vlan <regexp>

set nortel bridge vlan ports <regexp> <ports>
set nortel bridge vlan default <string> <ports>

show nortel bridge vlan info [<regexp>]
show nortel bridge vlan details [<regexp>]
show nortel bridge vlan ports

dump nortel bridge vlan
Configuring VLANs using scli and m4

```
delete nortel bridge vlan "^(134|ibr-)"  # regexps are cool :-)

create nortel bridge vlan 544 ibr-core
create nortel bridge vlan 545 ibr-cip
create nortel bridge vlan 546 ibr-test
create nortel bridge vlan 547 ibr-wlan

define(UP,'25,185')  # uplink ports
define(WLAN,'2,56')   # wireless vlan
define(CORE,'1,3-24,33-55,65-88')  # core vlan

include(vlan-all.scli)  # create the vlans

set nortel bridge vlan ports ibr-core UP,CORE
set nortel bridge vlan default ibr-core CORE
set nortel bridge vlan ports ibr-wlan UP,WLAN
set nortel bridge vlan default ibr-wlan UP,WLAN
```
Software Design Goals

Extensibility:
- Make it easy for programmers to add new features

Robustness:
- Ensure that errors are detected and handled gracefully

Efficiency:
- Short startup times for efficient usage in shell scripts

Portability:
- Tools should run on all major Unix platforms

Maintainability:
- Manual must be consistent with the implementation
Software Architecture

- scli command implementations
  - procedures
  - stubs
  - gsnmp
  - glib
  - scli interpreter core
  - libxml2
  - curses
  - readline
  - history

- operating system (Linux, Solaris, BSD, ...)

SANE 2002 – p.20
Stub Code Generator

- Stub functions for retrieving and/or modifying
  - complete conceptual tables
  - rows of conceptual tables
  - groups of scalars

- Stubs hide all low-level SNMP details such as
  - OID naming / (un-)packing of instance identifiers
  - automatic type and range checking

- Generated data structures force programmers to validate data members (pointers) before using them

- Implemented as part of the libsmi package
Printer-MIB::prtConsoleLightEntry

typedef struct {
    gint32 hrDeviceIndex;
    gint32 prtConsoleLightIndex;
    gint32 *prtConsoleOnTime;
    gint32 *prtConsoleOffTime;
    gint32 *prtConsoleColor;
    guchar *prtConsoleDescription;
    gsize _prtConsoleDescriptionLength;
} printer_mib_prtConsoleLightEntry_t;

extern void
printer_mib_get_prtConsoleLightTable(GSnmpSession *s,
    printer_mib_prtConsoleLightEntry_t ***prtConsoleLightEntry,
    gint mask);

extern void
printer_mib_free_prtConsoleLightTable(
    printer_mib_prtConsoleLightEntry_t **prtConsoleLightEntry);

/* ... more stub prototypes deleted ... */
static int
show_printer_console_lights(scli_interp_t *interp, int argc, char **argv)
{
    printer_mib_prtConsoleLightEntry_t **lightTable;
    int i, width = 12;

    if (argc > 1) return SCLI_SYNTAX;

    printer_mib_get_prtConsoleLightTable(interp->peer, &lightTable, 0);
    if (interp->peer->error_status) return SCLI_SNMP;
    if (lightTable)
    {
        for (i = 0; lightTable[i]; i++)
        {
            if (lightTable[i]->_prtConsoleDescriptionLength > width)
                width = lightTable[i]->_prtConsoleDescriptionLength;
        }
        g_string_sprintfa(interp->header, "PRINTER LIGHT %-*s STATUS COLOR",
            width, "DESCRIPTION");
        for (i = 0; lightTable[i]; i++)
        {
            fmt_printer_console_light(interp->result, lightTable[i], width);
        }
    }
    if (lightTable) printer_mib_free_prtConsoleLightTable(lightTable);
    return SCLI_OK;
}
static void
fmt_printer_console_light(GString *s, printer_mib_prtConsoleLightEntry_t *lightEntry, int width)
{
  const char *state = "off", *e;

  g_string_sprintfa(s, "%6d ", lightEntry->hrDeviceIndex);
  g_string_sprintfa(s, "%4d ", lightEntry->prtConsoleLightIndex);

  if (lightEntry->prtConsoleDescription) {
    g_string_sprintfa(s, "%-.*s ", width,
                     (int) lightEntry->_prtConsoleDescriptionLength,
                     lightEntry->prtConsoleDescription);
  } else {
    g_string_sprintfa(s, "%*s", width, "");
  }

  if (*lightEntry->prtConsoleOnTime && !*lightEntry->prtConsoleOffTime) {
    state = "on";
  } else if (!*lightEntry->prtConsoleOnTime && *lightEntry->prtConsoleOffTime) {
    state = "off";
  } else if (*lightEntry->prtConsoleOnTime && *lightEntry->prtConsoleOffTime) {
    state = "blink";
  }
  g_string_sprintfa(s, " %-*s ", 5, state);

  e = fmt_enum(printer_mib_enums_prtConsoleColor, lightEntry->prtConsoleColor);
  g_string_sprintfa(s, "%s\n", e ? e : "");
}
void scli_init_printer_mode(scli_interp_t * interp)
{
    static scli_cmd_t cmds[] = {
        { "show printer console lights", NULL,
            "The `show printer console lights' command shows the current
            "status of the lights attached to the printer. The command
            "generates a table with the following columns:
            "\n            "   PRINTER   logical printer number\n            "   LIGHT     number identifying the light/led\n            "   DESCRIPTION description of the light/led\n            "   STATUS    current status (on, off, blink)\n            "   COLOR     current color of the light",
            SCLI_CMD_FLAG_NEED_PEER, NULL, NULL, show_printer_console_lights },
        { NULL, NULL, NULL, 0, NULL, NULL, NULL } }
    };

    static scli_mode_t printer_mode = {
        "printer",
        "The scli printer mode is based on the Printer-MIB as\n        "published in RFC 1759.",
        cmds
    };

    scli_register_mode(interp, &printer_mode);
}
Try it yourself!

Software:

Papers:
- Tcl Extensions for Network Management Applications, 3rd Usenix Tcl/Tk Workshop, Toronto, 1995
- Married with Tcl, 1st European Tcl/Tk User Meeting, June 2000