Part 3: SNMP Protocol Operations

- Protocol operations
- SNMPv3 PDU
- Interactions:
  - Read requests: Get, GetNext, GetBulk Requests
  - SNMPv2 and SNMPv3 Exceptions
  - Write request: Set Request
  - Reports
  - Asynchronous requests: Traps and Informs

SNMP Protocol Operations

- SNMP is a request-response protocol
  - get, getNext, getBulk, set -> response (or report)
  - asynchronous trap
- SNMP uses connectionless transport service
- Management application implements desired level of reliability, and agents maintain no transaction state
- Standard mapping is onto UDP over IP
Transactions and Ports

- Manager (Any port, Port 162)
- Agent (Port 161, Any port)

Queries & Commands: Manager to Agent
Responses: Agent to Manager
Notifications: Agent to Manager

SNMPv3 PDU

PDU ::= SEQUENCE {
  request-id INTEGER,
  error-status INTEGER {
    noError(0), tooBig(1),
    noSuchName(2), ...
  },
  error-index INTEGER,
  variable-bindings VarBindList
}
SNMPv3 Variable Bindings

max-bindings INTEGER ::= 2147483647
VarBindList ::= SEQUENCE (SIZE (0..max-bindings)) OF VarBind
VarBind ::= SEQUENCE {
  name               ObjectName,
  CHOICE { value ObjectSyntax,
           unSpecified        NULL,
           noSuchObject[0]   IMPLICIT NULL,
           noSuchInstance[1] IMPLICIT NULL,
           endOfMibView[2]   IMPLICIT NULL
  }
}

SNMPv2 and SNMPv3 Operations

✦ We needed a bit more power (and complexity)
✦ The outcome of an operation is now:
  a response;
  a response with exceptions;
  a response with an error;
  an error report; or,
  a time-out.
SNMPv2 and SNMPv3
Typical Interaction

◆ Manager issues request with:
  ○ unique request-id;
  ○ zero-valued error-status/error-index; and,
  ○ zero or more variable bindings, each with unSpecified value

◆ Agent issues response with:
  ○ same request-id;
  ○ zero-valued error-status; and,
  ○ each variable binding with non-exceptional value

Get Operation Example

◆ Assume that you wish to retrieve some of the objects in the system group using the get operation:

```
# get localhost public sysDescr.0 sysObjectID.0 \
 sysUpTime.0 sysContact.0 sysName.0 sysLocation.0
```

sysDescr.0 = SNMPv3 agent from SNMP Research, Inc.
sysObjectID.0 = hpxx10EMANATEMasterAgent
sysUpTime.0 = 3952
sysContact.0 = SNMP Research +1 423 573 1434
sysName.0 = speedbump.snmp.com
sysLocation.0 = Knoxville, Tennessee USA
Introduction to the Internet Standard Management Framework

Observations on the Get Operation

- Each “variable” named entirely in the request using class and instance information.
- Only leaf objects retrieved
- Can have one or many objects in a single request
- Manager “knows” of the objects
- Values, exceptions returned, or error

GetNext Operation

- To retrieve some of the objects in the system group using the getNext operation:

  # getnext localhost public system sysObjectID \n  sysUpTime sysContact sysName sysLocation
  sysDescr.0 = SNMPv3 agent from SNMP Research, Inc.
  sysObjectID.0 = hpux10EMANATEMasterAgent
  sysUpTime.0 = 45274
  sysContact.0 = SNMP Research +1 423 573 1434
  sysName.0 = speedbump.snmp.com
  sysLocation.0 = Knoxville, Tennessee USA
Observations on the GetNext Operation

- Request had system, response had sysDescr.0
  Why?
- Leaf variable “lexicographically” after the object requested is returned
- Can be used to walk through the MIB at a managed system without a priori knowledge of the management information available

GetBulk Operator: Goals

- Minimize network interactions by allowing the agent to return large packets, while:
  - Still using a CL-mode transport service,
  - Leaving the manager in control of each interaction,
  - Not requiring state mechanisms in the agent, and
  - Using the same PDU format as every other operation!
### GetBulk Operator: Observations

- Management applications often need a timestamp with each interaction
- For example, an application might want to retrieve in each request:
  - once:
    - sysUpTime
  - many (N times):
    - ifInOctets, ifOutOctets, ifInErrors, ifOutErrors, ifInNUCastPkts, ifInUCastPkts, ifOutNUCastPkts, and ifOutUCastPkts

### GetBulk Operator: Observations (Continued)

- So, an ideal situation is to have an operation that retrieves:
  - one instance of some objects and
  - lots of instances of other objects
- This is exactly what the awesome getBulk operator does
GetBulk PDU

BulkPDU ::= SEQUENCE {
    request-id            Integer32,
    non-repeaters         INTEGER (0..max-bindings),
    max-repetitions       INTEGER (0..max-bindings),
    variable-bindings     VarBindList
}

GetBulk Example

getBulk {
    request-id = 1335
    non-repeaters = 1
    max-repetitions = 2
    variable-bindings = {
        sysUpTime,
        ifType,
        ifMtu,
        ifSpeed,
        ifInErrors,
        ifOutErrors
    }
}
Response to getBulk

sysUpTime.0 = 17 hours, 36 minutes, 32.19 seconds
(6339219 timeticks)
ifType.1 = ethernet-csmacd(6)
ifMtu.1 = 1500
ifSpeed.1 = 10485760
ifInErrors.1 = 40165
ifOutErrors.1 = 133
ifType.2 = other(1)
ifMtu.2 = 1500
ifSpeed.2 = 0
ifInErrors.2 = 40165
ifOutErrors.2 = 133

SNMPv2 and SNMPv3 Exceptional Response

◆ Agent's response has zero-valued error-status and, at least one exceptional value:
  ○ noSuchObject;
  ○ noSuchInstance; or,
  ○ endOfMibView
◆ Any non-exceptional values are “good” ones
◆ Exceptions only for read operations
Exception Example

◆ If you issue the request:

```bash
# get localhost User sysUpTime.0 hrSystemDate.0 \ ifType.3
```

◆ What does this response tell you?

- `sysUpTime.0 = 36889`
- `hrSystemDate.0 = NO_SUCH_OBJECT_EXCEPTION`
- `ifType.3 = NO_SUCH_INSTANCE_EXCEPTION`

Set Operation

◆ “as if simultaneous” with a *conceptual* two-phase commit

◆ Pass 1: check validity of each variable-binding access and creation/write check, syntax check, consistency check, resource locking. If any variable-binding invalid, set operation fails

◆ Pass 2: “commit” change to each variable-binding
Set Operation Example

Assume that you wish to change the value of an object:

```bash
# set localhost something sysContact.0 "David Partain" \ 
sysLocation.0 "Mjardevi Science Park, Linkoping"
```

sysContact.0 = David Partain
sysLocation.0 = Mjardevi Science Park, Linkoping

```bash
# getnext localhost public sysContact sysLocation
```

sysContact.0 = David Partain
sysLocation.0 = Mjardevi Science Park, Linkoping

Observations on the Set Operation

- Manager names the instance of the object to be altered
- Manager understands object syntax (in these two cases, both are printable OCTET STRINGs)
- All or nothing: “as if simultaneous”
SNMPv2 and SNMPv3 Error Response: Sets

- If a set operation fails:
  - noAccess, notWritable . . .
  - wrongType, wrongLength, wrongEncoding, wrongValue . . .
  - noCreation . . .
  - inconsistentName, inconsistentValue,
  - resourceUnavailable,
  - commitFailed, undoFailed
  and, of course, genErr

Set Error Examples

- # set localhost user1 sysUpTime.0 123
  - Error code in packet - NOT_WRITABLE_ERROR
  - VarBind 1

- # set localhost user2 sysLocation.0 "Sweden"
  - Error code in packet - AUTHORIZATION_ERROR
  - VarBind 1
SNMPv2 and SNMPv3 Time-outs

◆ A lot of ambiguity here:
  ○ network dropped request;
  ○ the agent isn't running;
  ○ agent dropped request;
  ○ network dropped response; or,
  ○ the time-out was too short
◆ Security will introduce a whole new set of failure modes

SNMPv2 and SNMPv3 Time-outs (Continued)

◆ Solutions:
  ○ instrumentation to count silent drops
  ○ tooBig responses now as small as possible
  ○ possible inclusion of reportPDU
Report PDU

- Basic idea in SNMPv3:
  *If a protocol entity is going to discard a request without issuing a response, it will instead issue a Report that includes a VarBind representing the error condition.*
- Rule: reports are never sent about reports

SNMP Traps and Informs

- SNMP defines a few “generic traps”
  - coldStart, warmStart, linkDown, linkUp, authenticationFailure, egpNeighborLoss
- New ones can also be defined (using the NOTIFICATION-TYPE macro) in private MIBs
- Agent chooses (and can be configured) whether to send as confirmed or unconfirmed traps

David Partain (david.partain@ericsson.com)
SNMPv2 and SNMPv3 TRAP

- The trap PDU uses the same format as every other operation
- First two variable bindings are special:
  - sysUpTime.0: old time-stamp field
  - snmpTrapOID.0: trap OID

SNMPv2 and SNMPv3 TRAP (Continued)

SNMPv2-Trap-PDU ::= [7] IMPLICIT SEQUENCE

{ request-id INTEGER,
  error-status INTEGER,  -- always zero
  error-index INTEGER,  -- always zero
  variable-bindings VarBindList }
**Trap Destinations**

- **SNMPv1**: Destination configuration is implementation dependent
- **SNMPv2c**: Destination “is determined in an implementation-dependent fashion”
- **SNMPv3**: destinations configurable via standards-based MIB objects

---

**Trap Generation: Example**

Suppose the agent is generating a linkUp trap on interface #7:
- trap OID is `{ snmpTraps 4 }`
- one variable binding, `ifIndex.7`
- destination(s) determined locally

```plaintext
snmpV2-trap [request-id = 1] {
  {sysUpTime.0,   6},
  {snmpTrapOID.0, linkUp}
  {ifIndex.7,   7}
}
```
Inform PDU

- Generation determined locally
- Same format as other PDUs
- First two VarBinds are special (as for trap)
- Recipient acknowledges - MIB control over retransmission
- Some fear it will be abused as “there is no such thing as a reliable trap”