IETF Approaches to Network Management

Concordia Summer Workshop on Services Engineering and Network Management

Montreal, 18-20 August 2003

Jürgen Schönwälder

Electrical Engineering and Computer Science
International University Bremen
P.O. Box 750 561
28725 Bremen, Germany

Phone: +49 421 200 3587
Email: j.schoenwaelder@iu-bremen.de
Web: http://www.eecs.iu-bremen.de/

Slides available at http://www.ibr.cs.tu-bs.de/users/schoenw/slides/ for downloading.

Copyright Notice

Outline of the Tutorial

1. Fundamental Concepts (30 min)
2. Internet Standard Management Framework (60 min)
3. Common Open Policy Service (30 min)
4. Evolution of the Internet Standard Management Framework (30 min)
5. Revolution of the Internet Standard Management Framework (30 min)
6. References & Online Resources

1. Fundamental Concepts

1.1 Principles and Scope of Network Management

1.2 Data- vs. Command- vs. Object-centric Approaches

1.3 Fundamental Primitives in Data-centric Approaches

1.4 Information Models vs. Data Models

1.5 Network Management Standards

1.1 Principles and Scope of Network Management

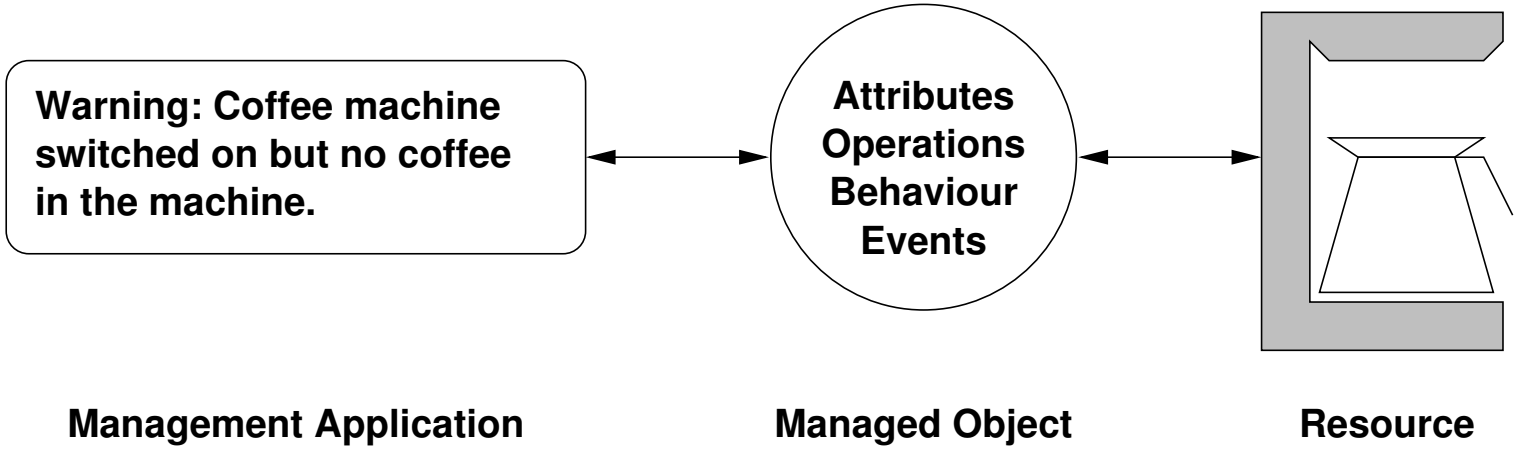
- Networks of non-trivial size need management:
  - Fault detection and isolation
  - Configuration generation and installation
  - Accounting data gathering
  - Performance monitoring and tuning
  - Security management (keys, access control)

- Network management is complicated:
  - Scalability is a key concern
  - Short technology life times
  - Heterogenity requires standards-based solutions
  - Lack of skilled persons

$\Longrightarrow$ But network management is not really fundamentally different from other control systems (e.g. systems that control robots in a vehicle fabric).

$\Longrightarrow$ But network management terminology is often very different (especially for people with computer science background).

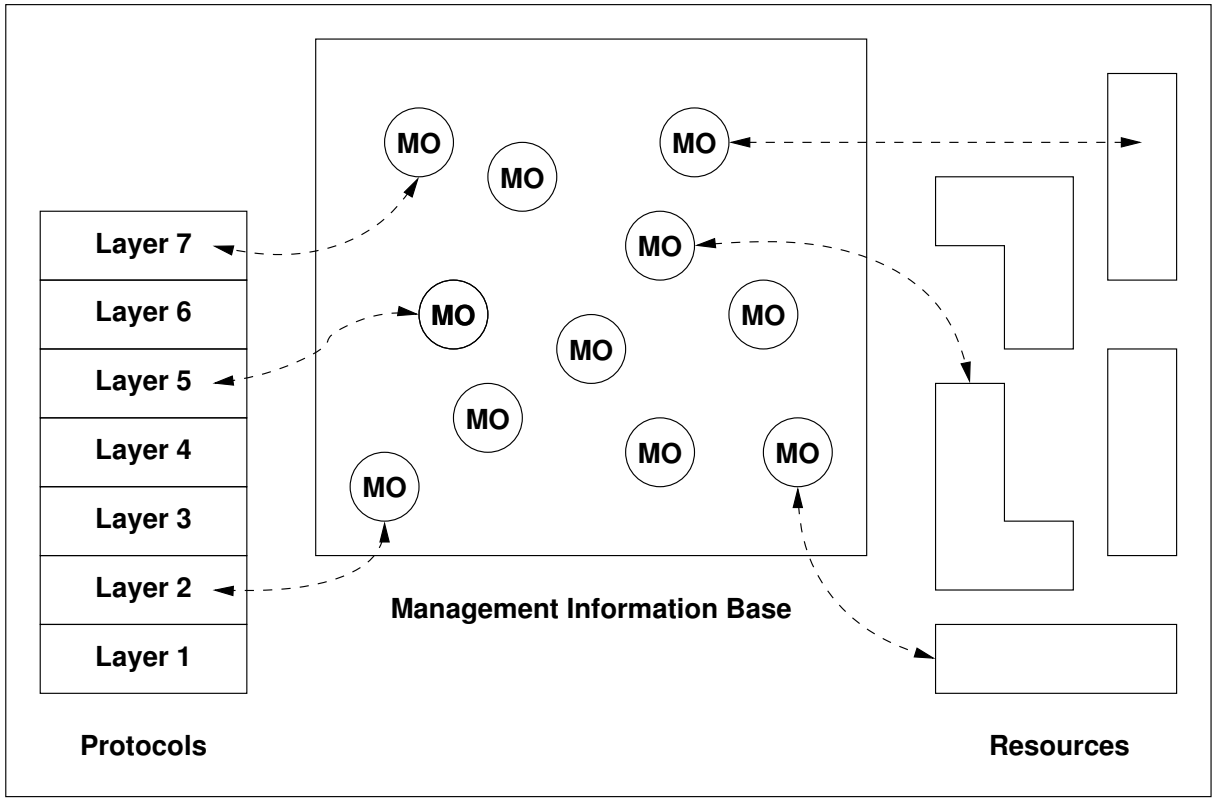
Abstraction of Managed Objects (MOs)



- A managed object is the abstracted view of a resource that presents its properties as seen by (and for the purpose of) management (ISO 7498-4).

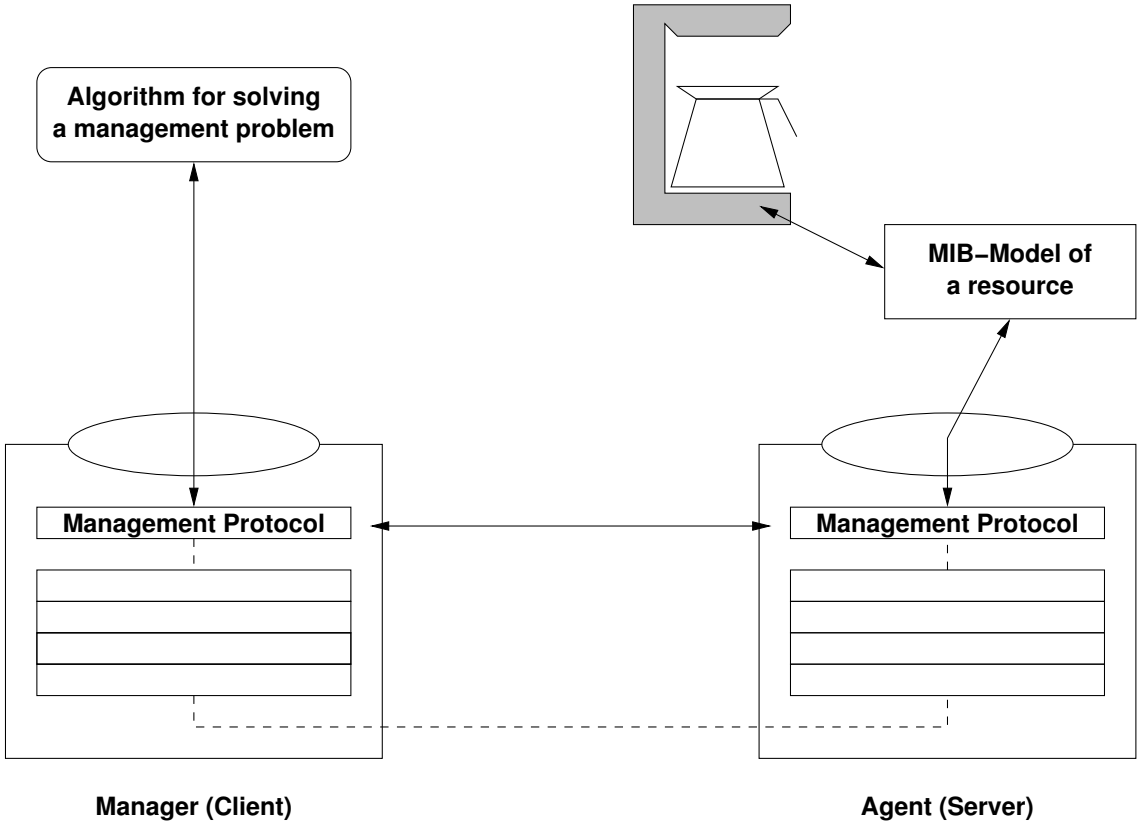
- The boundary of a managed object defines the level of details which are accessible for management systems.

Management Information Base (MIB)



- The set of managed objects within a system, together with their attributes, constitutes that system's management information base (ISO 7498-4).

Management Protocols



- Management protocols realize the access to MOs contained in a MIB.

1.2 Data- vs. Command- vs. Object-centric Approaches

- Data-centric approach:
  - The device is represented as a collection of data objects which represent all the properties and capabilities of a device.
  - The management protocol manipulates the data objects representing a device.
  - Example: Internet management (SNMP) approach

- Command-centric approach:
  - The device is considered to be a stateful black box.
  - A set of commands can be send to the device to (a) change the state of the device or (b) to retrieve data about the current state of (portions of) the device.
  - Examples: Command line interfaces of routers or switches

- Object-centric approach:
  - The device is represented as a collection of data objects with associated methods.
  - This is basically a combination of the data- and the command-centric approach.
  - Example: OSI management approach (CMIP)

1.3 Primitives in Data- and Object-centric Protocols

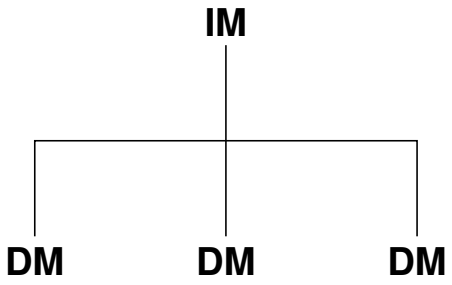
- From a very abstract viewpoint, the following set of essential management protocol primitives is needed for data-centric or object-centric management protocols:
  - `GET`, `SET`
  - `CREATE`, `DELETE`
  - `SEARCH` (or at the very least `ITERATE`)
  - `LOCK`, `UNLOCK`, `COMMIT`, `ROLLBACK`
  - `NOTIFY` someone about an asynchronous event
  - `EXECUTE` or `INVOKE` an operation or method
- Protocols that lack some of the primitives have proven to be problematic.
- The locking primitives are needed to support transactions across sets of devices.
- Command-centric protocols on the other hand usually have a very rich set of primitives (often hierarchically structured).

1.4 Information Models vs. Data Models (RFC 3444)

- Information Models (IMs):
  - IMs are used to model managed objects at a conceptual level, independent of any specific implementations or protocols used to transport the data.
  - The degree of specificity (or detail) of the abstractions defined in the IM depends on the modeling needs of its designers.
  - In order to make the overall design as clear as possible, an IM should hide all protocol and implementation details.
  - Another important characteristic of an IM is that it defines relationships between managed objects.

- Data Models (DMs):
  - DMs are defined at a lower level of abstraction and include many details.
  - They are intended for implementors and include protocol-specific constructs.

Relationship between Information Models and Data Models

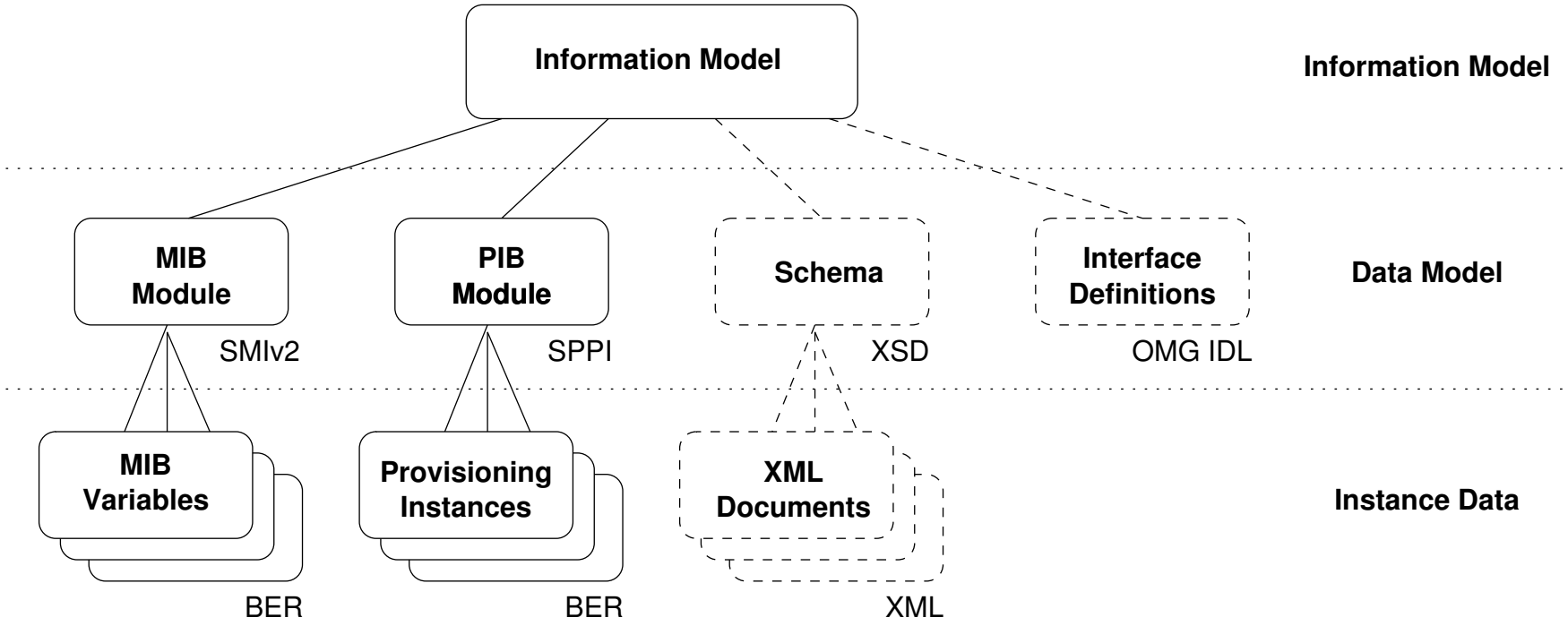


conceptual/abstract model for designers and operators

concrete/detailed model for implementors

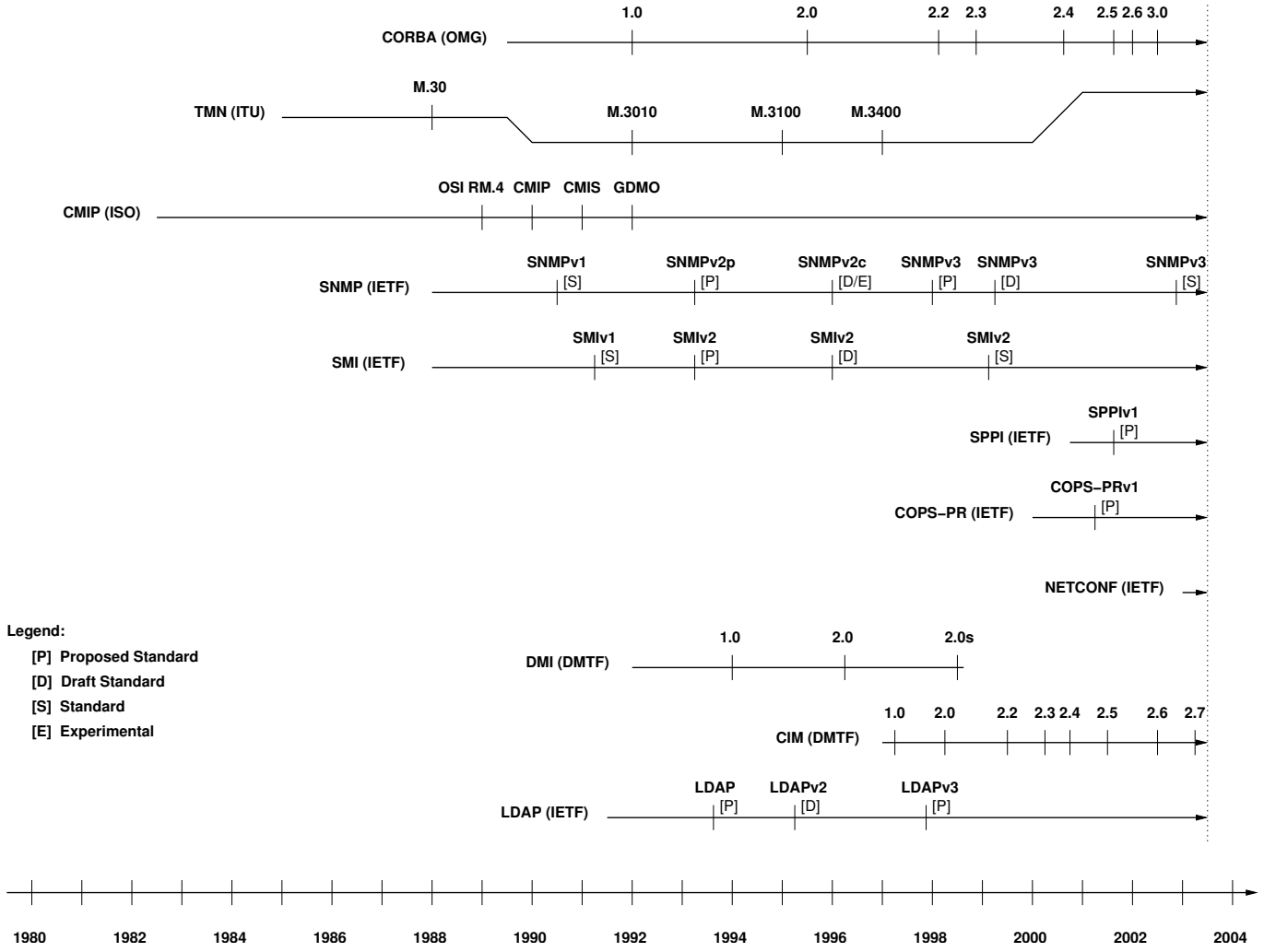
- Since conceptual models can be implemented in different ways, multiple DMs can be derived from a single IM.
- Although IMs and DMs serve different purposes, it is not always possible to precisely define what level of detail should be expressed in an IM and which one in a DM.
- Similarily, it is sometimes difficult to determine whether an abstraction belongs to an IM or a DM.
- IMs are often represented in Unified Modeling Language (UML) diagrams, but there are also informal IMs written in plain English language. DMs are often represented in data definition languages that are specific to the management protocol.

IMs and DMs in the Real World

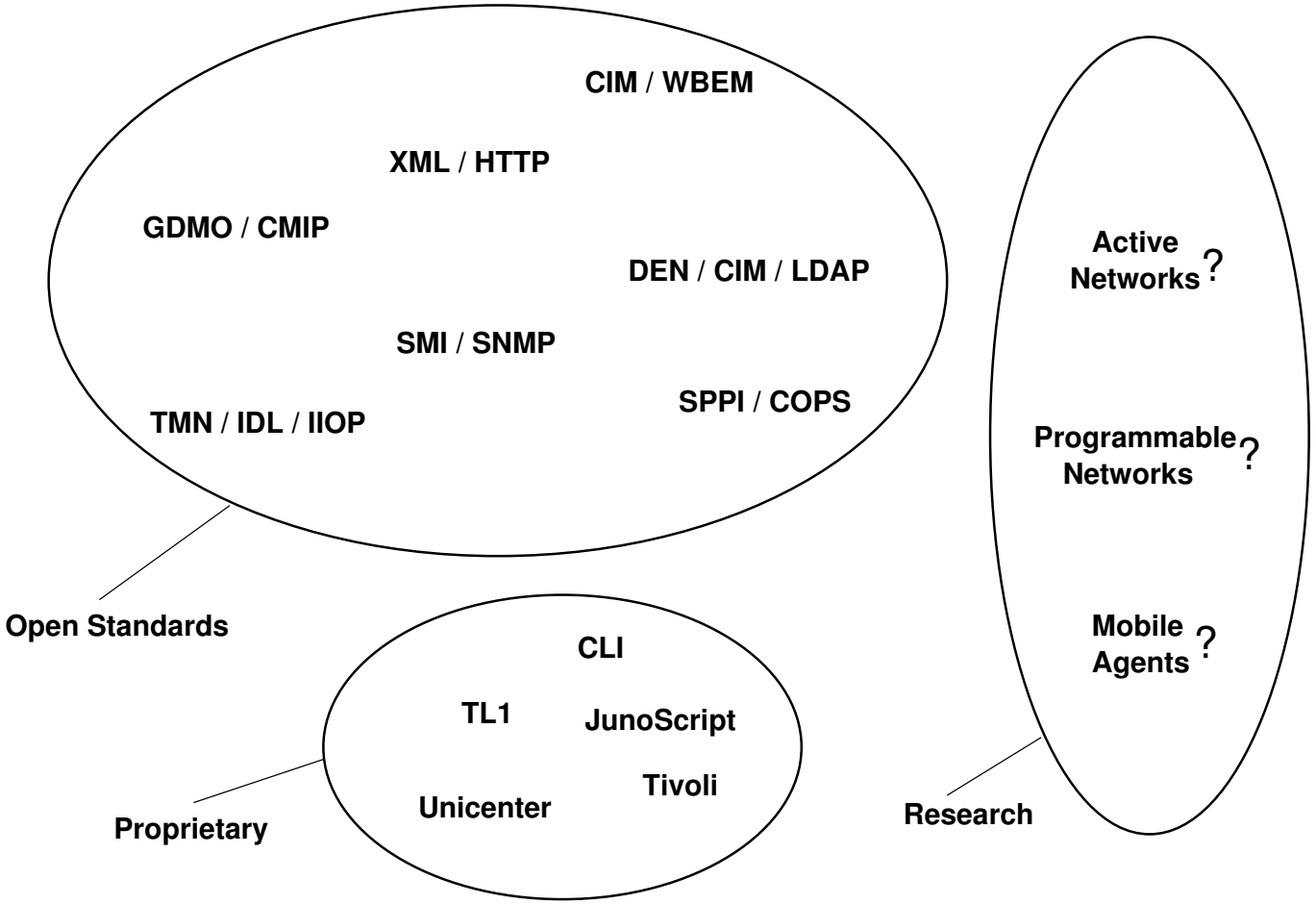


- The Architecture for Differentiated Services (RFC 2475) is an example for an informal definition of the DiffServ information model.
- The DiffServ MIB module (RFC 3289) and the DiffServ PIP module (RFC 3317) are examples of data models (hopefully) conforming to the DiffServ information model.

1.5 Network Management Standards



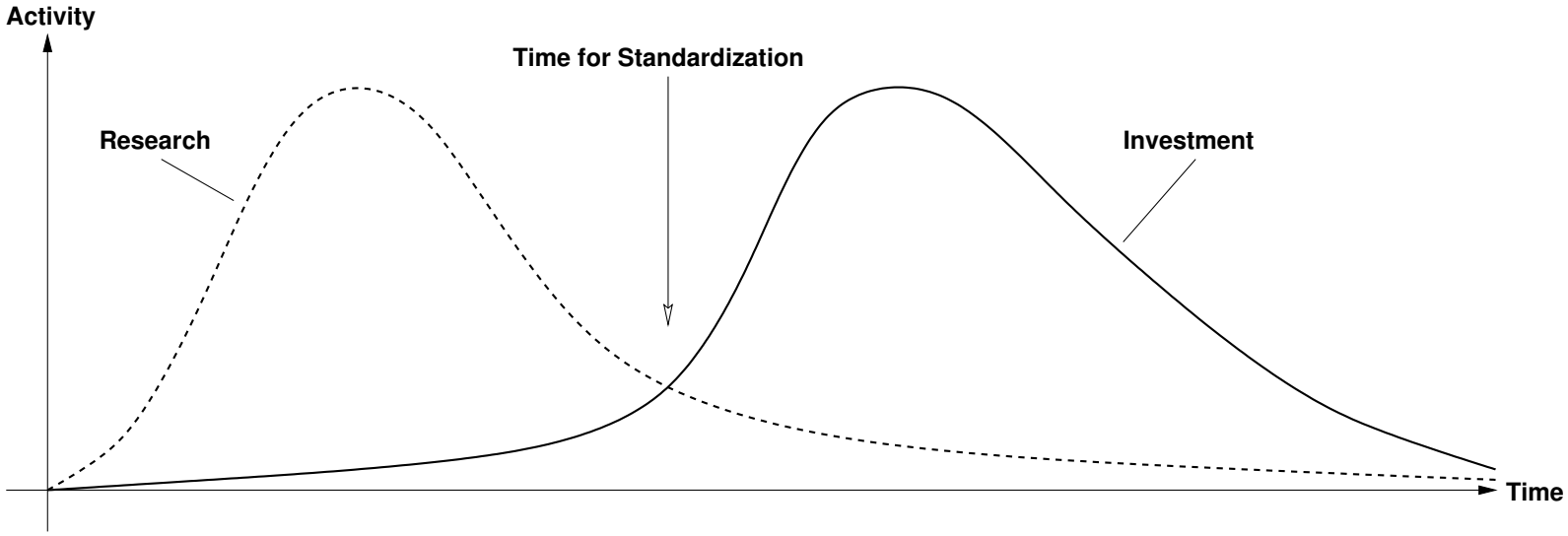
Network Management Technology Fragmentation



Implications

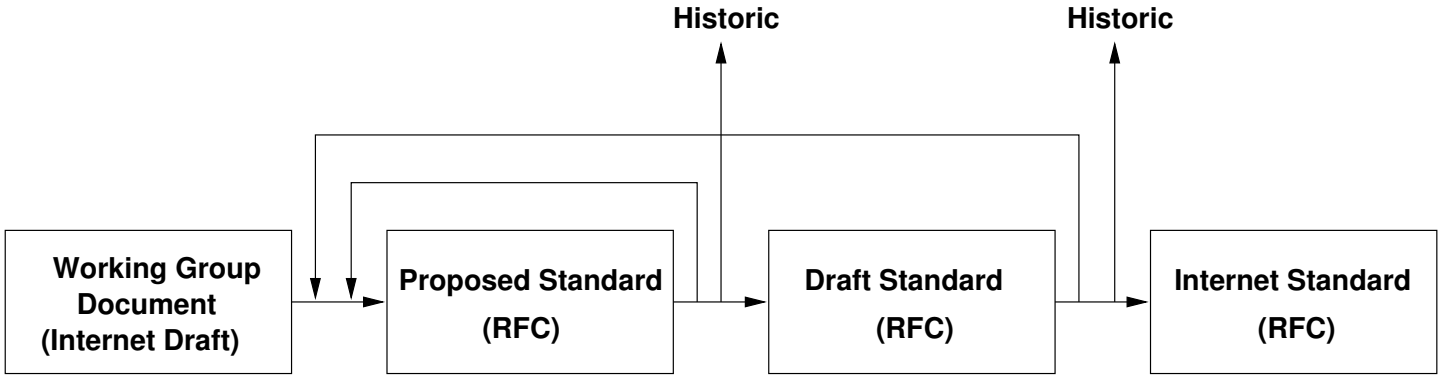
- Standards Organizations:
  - Duplication of efforts binds scare human resources.
- Network Device Vendors:
  - Customers force device vendors to support multiple management technologies.
  - Makes management interfaces unnecessarily complex and increases costs.
- Management Application Vendors:
  - Difficulties to choose from the various paradigms.
  - Integrated solutions are hard and thus expensive to realize.
- Network Operators:
  - Difficult to create heterogeneous networks with common management interfaces.
  - Fragmented technologies increase the costs to implement new services efficiently.

Theory of Standardization



- The success of a standard must be measured in terms of wide-spread deployment.
- Standards must allow vendors to differentiate their products.
- Successful standards can create new open markets.
- The timeliness of standards is a key factor for success.

IETF Standards Process (RFC 2026)



- Internet Drafts are working documents and can be changed or removed anytime.
- All Internet standards are published in a series called Request For Comments (RFC).
- Not all RFCs define standards (informational or experimental RFCs).
- The step from Proposed to Draft standard requires at least two independent and interoperable implementations from different code bases for all protocol features.
- The step from Draft to Internet standard requires significant implementation and operational experience.

Problems with Existing IETF Management Standards

- The requirements for Internet management technologies have changed during the last decade.
- Some fundamental design decisions must be revisited to better reflect today's realities.
- Working group members are mostly from network device vendors. Little participation of management application vendors and from the network operator community.
- Solutions tend to be too device specific or way too detailed for running real networks.
- Work on SNMP security took many many years to finally result in a stable and accepted SNMPv3 specification.
- Other urgently needed SNMP improvements were kept on hold during this time.
- Need to move to more mainstream technologies since network management remains a niche market.

More Thoughts on IETF Standardization

- Things to keep in mind while developing standards:
  1. KISS: Complex standards which require people with special skills will not survive.
  2. Timeliness: Standards need to address real problems in a timely manner.
  3. Concentration processes have given a few "big players" strong influence on the success of standards.

- Standardizing MIBs in order to establish an open market between device vendors and management software vendors does not work very well in some cases:
  1. Standardization takes too long.
  2. Consensus often on the lowest common denominator.
  3. Operationally important information often contained in proprietary MIB definitions.
  4. Implementation and resource costs hinder fast and wide-spread deployment.

$\Rightarrow$ The sheer number of standardization efforts and proposals sometimes seem to distract those who do the actual work from doing the actual work.

2. Internet Standard Management Framework

2.1 Overview and Evolution of SNMP Technology

2.2 Data Modeling Language SMIv2

2.3 Communication Protocol SNMPv3

2.4 Core Information Models and Data Models

2.5 Extensible and Programmable Agents

2.2 Structure of Management Information Version 2 (SMIv2)

2.2.1 Fundamental SMIv2 Concepts

2.2.2 SMI Naming (Object Type Identifier and Instance Identifier)

2.2.3 SMI Modules and Object Identities

2.2.4 Object Types and Notification Types

2.2.5 Textual Conventions

2.2.6 Conformance Statements

2.2.7 MIB Module Design, Implementation and Testing

2.2.1 Fundamental SMIv2 Concepts

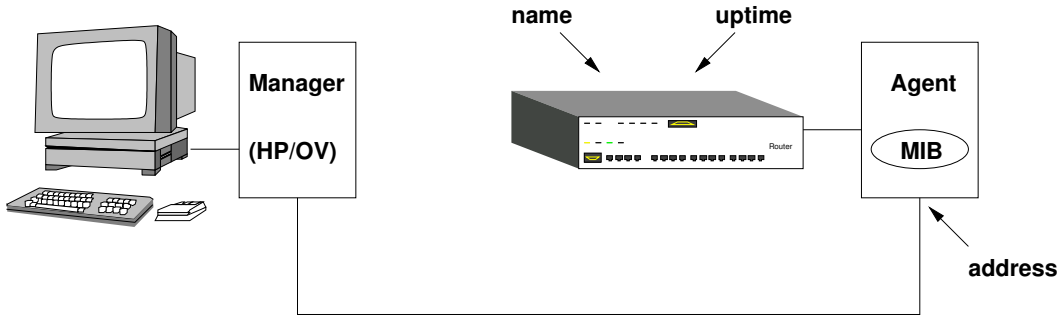
- The second version of the Structure of Management Information (SMIv2) [RFC 2578, RFC 2579, RFC 2580] is a data definition language which is used to define the syntax and semantics of management data models.
- The SMIv2 is based on an adapted subset of ASN.1 (1988).
- Management data is held in a collection of typed variables.
- All SMIv2 data types resolve to one of the three primitive ASN.1 types: INTEGER, OCTET STRING, and OBJECT IDENTIFIER.
- There are no complex data types (like classes, structures or unions).
- Variables either appear as scalars (exactly one instance) or as columns in a conceptual table (zero or more instances).
- Variables can be read and written using SNMP. The protocol allows to manipulate arbitrary lists of variables in one transaction.

SMIv2 Base Types

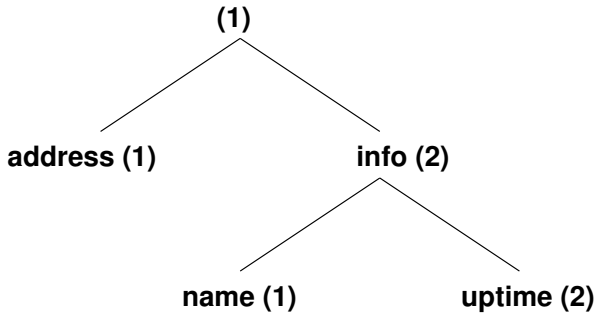
SMIv2 Type	Description
INTEGER	enumerations (signed integer)
OCTET STRING	sequence of arbitrary bytes
OBJECT IDENTIFIER	unique identifier
Integer32	signed integer (-2147483648..2147483647)
Unsigned32	unsigned integer (0..4294967295)
Gauge32	gauge (0..4294967295)
Counter32	counter (0..4294967295)
Counter64	counter (0..18446744073709551615)
TimeTicks	elapsed time in hundredths of a second
IpAddress	four byte IP version 4 address
Opaque	wrapper for arbitrary ASN.1 types
BITS	named bit sets

- The Opaque type is not available for general use.
- The IpAddress type is problematic wrt. IPv6.
- The BITS constructions is a pseudo-type from an ASN.1 point of view.

2.2.2 SMI Naming



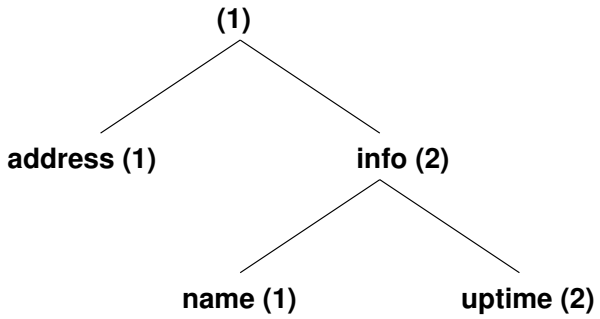
- Every variable type is registered in the ISO OID registration tree.
- The leaves of the OID tree represent so called object types.
- Intermediate nodes may be used to group related object types together.



Object Type Identifier versus Instance Identifier

- The registration of an object type assigns a globally unique name (OID).
- Concrete instances of the object type are identified by an instance identifier.
- The instance identifier is appended to the OID (name) of the object type.
- Scalars can only have one instance. The instance identifier is always "0".
- The instance identifier of columnar variables is derived from the key(s) of the underlying conceptual table.
- Humans usually refer to object types by the descriptor (e.g. "name").

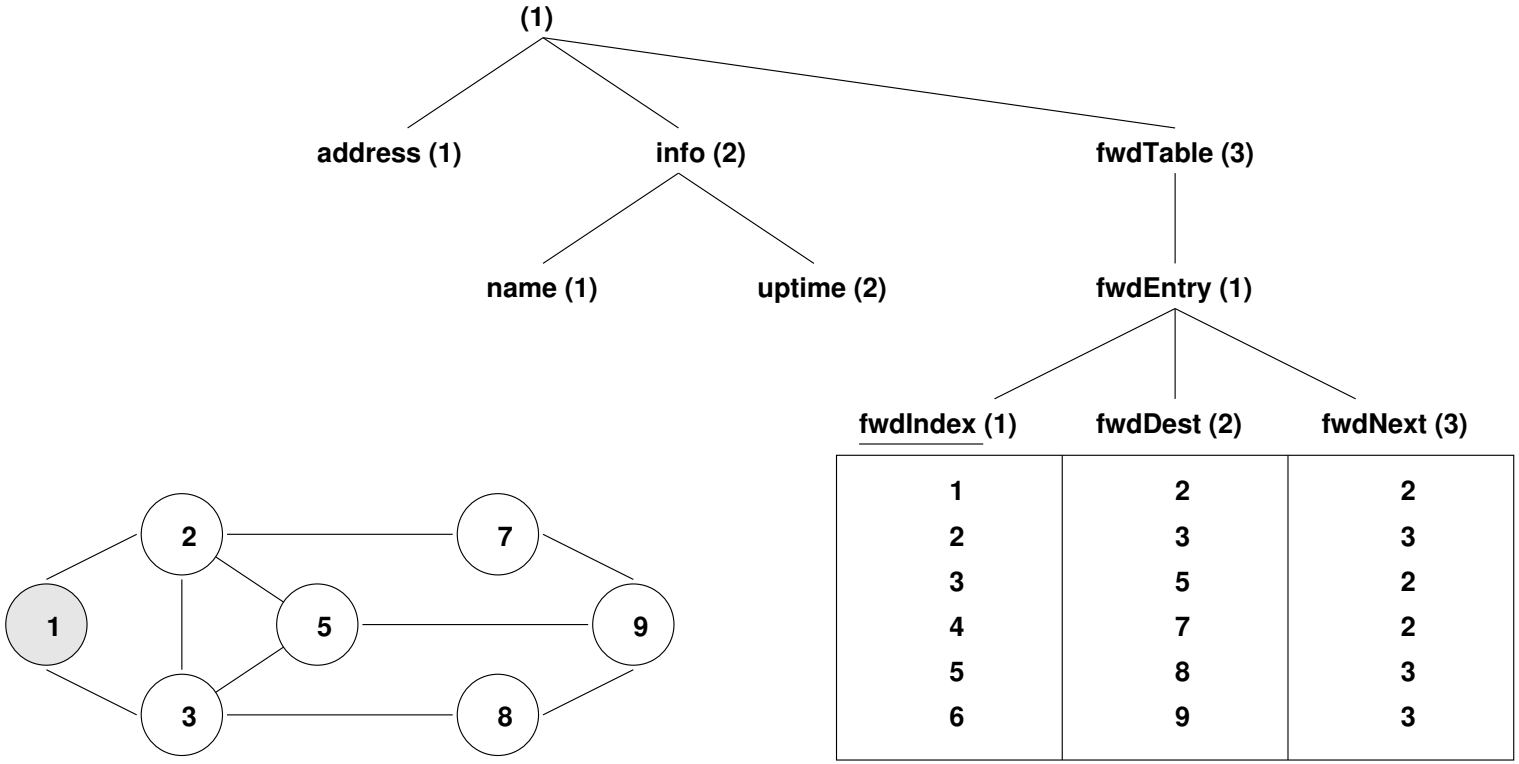
Scalar Object Type and Instance Identifiers



Object Identifier	Instance Identifier	Type	Value
1.1	0	IpAddress	10.1.2.1
1.2.1	0	OCTET STRING	"ACME Router"
1.2.2	0	TimeTicks	54321

- Descriptors are only used by humans — SNMP only operates on OID values.
- Descriptors must be unique within a module. Module names must be globally unique.

Columnar Object Type and Instance Identifiers



- The column fwdIndex uniquely identifies a row in the conceptual fwdTable.

Columnar Instance Identifier

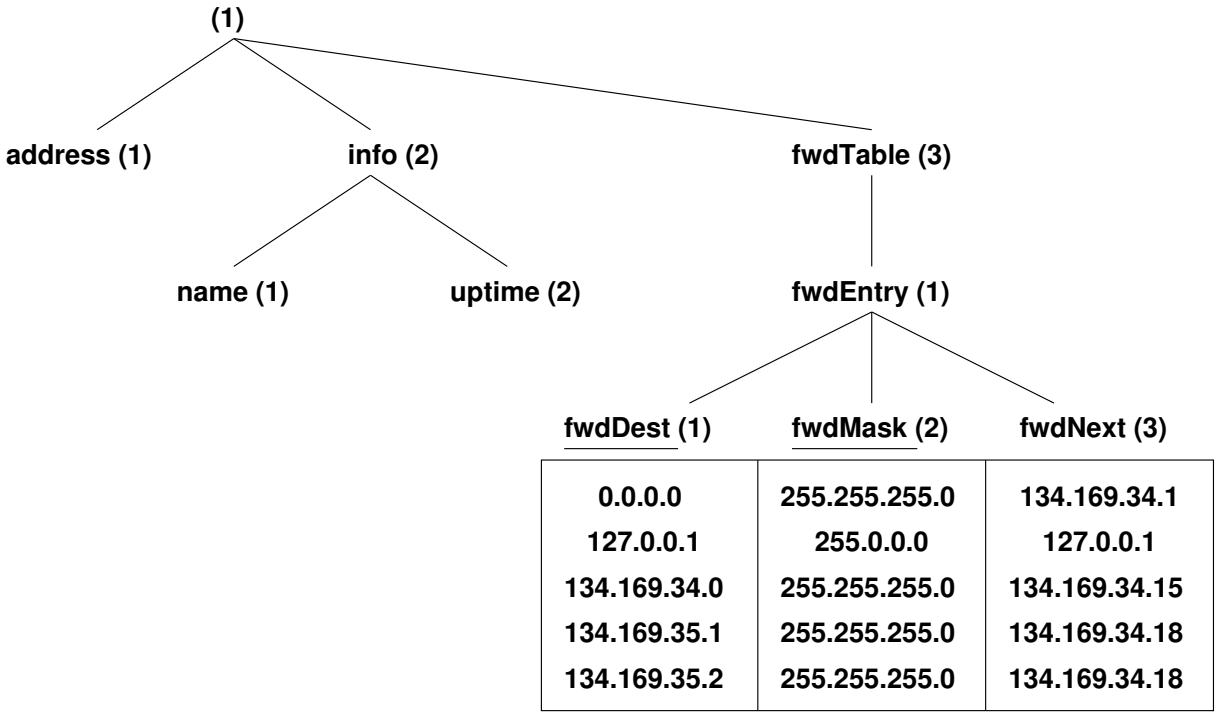
- Conceptual tables are constructed using two auxiliary nodes:
  - The first node represents the table (syntactically an ASN.1 SEQUENCE OF).
  - The descriptor of the first node always ends with "Table".
  - The second node represents a table row (syntactically an ASN.1 SEQUENCE).
  - The descriptor of the second node always ends with "Entry".

- The primary key of the table is called the index and is used to derive the instance identifier for a table row.

- The concatenation of the object type identifier and the instance identifier yields the unique object identifier to address a columnar variable.

- Examples: 1.3.1.1.1 $\Rightarrow$ 1 1.3.1.3.1 $\Rightarrow$ 2 1.3.1.2.4 $\Rightarrow$ 7 1.3.1.2.7 $\Rightarrow$ does not exist

Complex Columnar Instance Identifier



- The index into an IPv4 forwarding table is the combination of a destination address and an address mask.
- Example: 1.3.1.3.134.169.34.0.255.255.255.0 $\Rightarrow$ 134.169.34.15

Forming an Instance Identifier for Conceptual Tables

- Rules for forming the instance identifiers:

  1. integer-valued:
     a single sub-identifier taking the integer value
  2. string-valued, fixed-length:
     each octet of the string is encoded in a separate sub-identifier
  3. string-valued, variable-length:
     the first sub-identifier is the length followed by each octet of the string encoded in a separate sub-identifier
  4. object identifier-valued:
     the first sub-identifier is the length followed by each sub-identifier

- The IMPLIED keyword can be used to avoid the length sub-identifier in some cases (not recommended).

- Instance identifier can not be arbitrarily complex since OIDs are limited to 128 sub-identifiers.

2.2.3 MIB Modules and Object Identities

- Related SMI definitions are grouped into MIB modules.
- Every MIB module must have a unique name (usually uppercase).
- A MIB module corresponds to an ASN.1 module.
- Definitions can be imported from other MIB modules using an ASN.1 IMPORT.
- All SMIv2 macros and derived data types used in a MIB module must be imported.

```
ACME-ROUTER-MIB DEFINITIONS ::= BEGIN

IMPORT MODULE-IDENTITY, OBJECT-TYPE, Integer32, enterprises FROM SNMPv2-SMI;

...

END
```

SMIv2 Module Identity Macro (RFC 2578)

```
<Descriptor> MODULE-IDENTITY
        LAST-UPDATED <ExtUTCTime>
        ORGANIZATION <Text>
        CONTACT-INFO <Text>
        DESCRIPTION  <Text>
       [REVISION     <ExtUTCTime>
        DESCRIPTION  <Text>      ]*
       ::= <ObjectIdentifier>
```

- The MODULE-IDENTITY macro provides contact information and the revision history.
- The REVISION and DESCRIPTION clauses can repeat multiple times.
- ExtUTCTime contains a date in one of the formats "YYMMDDHHMMZ" (UTC) or "YYYYMMDDHHMMZ", e.g. "9502192015Z" or "199502192015Z".
- The OID of the MODULE-IDENTITY macro is often used as the top-level node for all definitions in a MIB module.

Module Identity Macro Example (RFC 2863)

```
IF-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, Counter32,
    Gauge32, Counter64, Integer32, TimeTicks, mib-2,              FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, DisplayString, PhysAddress, TruthValue,
    RowStatus, TimeStamp, AutonomousType, TestAndIncr             FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP           FROM SNMPv2-CONF
    snmpTraps                                                     FROM SNMPv2-MIB
    IANAifType                                                    FROM IANAifType-MIB;

ifMIB MODULE-IDENTITY
    LAST-UPDATED "200006140000Z"
    ORGANIZATION "IETF Interfaces MIB Working Group"
    CONTACT-INFO "Keith McCloghrie           Cisco Systems, Inc.
                  408-526-5260               170 West Tasman Drive
                  kzm@cisco.com              San Jose, CA 95134-1706, US"
    DESCRIPTION  "The MIB module to describe generic objects for network interface
                  sub-layers.  This MIB is an updated version of MIB-II's ifTable,
                  and incorporates the extensions defined in RFC 1229."
    REVISION     "200006140000Z"
    DESCRIPTION  "Clarifications agreed upon by the Interfaces MIB WG, and published as RFC 2863"
    REVISION     "199602282155Z"
    DESCRIPTION  "Revisions made by the Interfaces MIB WG, and published in RFC 2233."
    REVISION     "199311082155Z"
    DESCRIPTION  "Initial revision, published as part of RFC 1573."
    ::= { mib-2 31 }
...
END
```

SMIv2 Object Identity Macro (RFC 2578)

```
<Descriptor> OBJECT-IDENTITY
        STATUS       <Status>
        DESCRIPTION  <Text>
      [ REFERENCE    <Text>    ]
      ::= <ObjectIdentifier>
```

- The macro is used to define and register OBJECT IDENTIFIER values.
- The macro attaches a description to an intermediate MIB node.
- The STATUS clause defines whether the definition is "current", "deprecated" or "obsolete".
- The optional REFERENCE clause can point to related definitions.
- Used to define constants ("extensible enumerations") or to structure the OID tree.

Object Identity Macro Examples (RFC 2578, RFC 3417)

```
zeroDotZero    OBJECT-IDENTITY
    STATUS     current
    DESCRIPTION
            "A value used for null identifiers."
    ::= { 0 0 }
```

```
snmpUDPDomain  OBJECT-IDENTITY
    STATUS     current
    DESCRIPTION
            "The SNMPv2 over UDP transport domain.  The corresponding
            transport address is of type SnmpUDPAddress."
    ::= { snmpDomains 1 }
```

```
snmpIPXDomain  OBJECT-IDENTITY
    STATUS     current
    DESCRIPTION
            "The SNMPv2 over IPX transport domain.  The corresponding
            transport address is of type SnmpIPXAddress."
    ::= { snmpDomains 5 }
```

2.2.4 Object Types and Notification Types

- The heart of a MIB are the object types and conceptual tables.
- The object type macro is used to define object types and conceptual tables.
- Additional ASN.1 type definitions are needed for conceptual tables.
- Notification Types define events that can be reported to managers.
- Notifications have been used sparingly in the past since there were no standardized mechanisms to enable/disable them.
- Not carefully designed notification types can lead to notification storms.

SMIv2 Object Type Macro (RFC 2578)

```
<Descriptor> OBJECT-TYPE
        SYNTAX          <Syntax>
      [ UNITS           <Text>    ]
        MAX-ACCESS      <Access>
        STATUS          <Status>
        DESCRIPTION     <Text>
      [ REFERENCE       <Text>    ]
      [ INDEX           <Index>   ]
      [ AUGMENTS        <Index>   ]
      [ DEFVAL          <Value>   ]
      ::= <ObjectIdentifier>
```

- OBJECT-TYPE macros are used to define object types as well as conceptual tables.
- The INDEX or AUGMENTS clause is only allowed for conceptual row definitions.
- Conceptual row definitions must have either an INDEX clause or an AUGMENTS clause.

Object Type Macro Examples (RFC 3418)

```
sysDescr OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "A textual description of the entity.  This value should
                 include the full name and version identification of the
                 system's hardware type, software operating-system, and
                 networking software."
    ::= { system 1 }

sysUpTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "The time (in hundredths of a second) since the network
                 management portion of the system was last re-initialized."
    ::= { system 3 }

sysName OBJECT-TYPE
    SYNTAX      DisplayString (SIZE (0..255))
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION "An administratively-assigned name for this managed node.
                 By convention, this is the node's fully-qualified domain name.
                 If the name is unknown, the value is the zero-length string."
    ::= { system 5 }
```

```
sysORTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SysOREntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "The (conceptual) table listing the capabilities of the
                 local SNMP application acting as a command responder with
                 respect to various MIB modules.  SNMP entities having
                 dynamically-configurable support of MIB modules will have a
                 dynamically-varying number of conceptual rows."
    ::= { system 9 }

sysOREntry OBJECT-TYPE
    SYNTAX      SysOREntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "An entry (conceptual row) in the sysORTable."
    INDEX      { sysORIndex }
    ::= { sysORTable 1 }

SysOREntry ::= SEQUENCE {
    sysORIndex     INTEGER,
    sysORID        OBJECT IDENTIFIER,
    sysORDescr     DisplayString,
    sysORUpTime    TimeStamp
}
```

```
sysORIndex OBJECT-TYPE
    SYNTAX      INTEGER (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "The auxiliary variable used for identifying instances of
                 the columnar objects in the sysORTable."
    ::= { sysOREntry 1 }

sysORID OBJECT-TYPE
    SYNTAX      OBJECT IDENTIFIER
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "An authoritative identification of a capabilities statement with respect to various
                 MIB modules supported by the local SNMP application acting as a command responder."
    ::= { sysOREntry 2 }

sysORDescr OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "A textual description of the capabilities identified by the
                 corresponding instance of sysORID."
    ::= { sysOREntry 3 }

sysORUpTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "The value of sysUpTime at the time this conceptual row was last instanciated."
    ::= { sysOREntry 4 }
```

SMIv2 Notification Type Macro (RFC 2578)

```
<Descriptor> NOTATION-TYPE
        [ OBJECTS       <Objects> ]
          STATUS        <Status>
          DESCRIPTION   <Text>
        [ REFERENCE     <Text>    ]
        ::= <ObjectIdentifier>
```

- Registers a notification type with an object identifier.
- The next to last sub-identifier should be "0" for backwards compatibility with older SNMP versions.
- The OBJECTS clause defines the ordered sequence of MIB object types which are contained within every instance of the notification.
- The DESCRIPTION clause must describe which instances are to be included in a notification.

Notification Type Macro Examples (RFC 2863)

```
linkDown NOTIFICATION-TYPE
    OBJECTS     { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS      current
    DESCRIPTION "A linkDown trap signifies that the SNMP entity,
                 acting in an agent role, has detected that the
                 ifOperStatus object for one of its communication links
                 is about to enter the down state from some other state
                 (but not from the notPresent state).  This other state
                 is indicated by the included value of ifOperStatus."
    ::= { snmpTraps 3 }
```

```
linkUp NOTIFICATION-TYPE
    OBJECTS     { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS      current
    DESCRIPTION "A linkDown trap signifies that the SNMP entity,
                 acting in an agent role, has detected that the
                 ifOperStatus object for one of its communication links
                 left the down state and transitioned into some other
                 state (but not into the notPresent state).  This other
                 state is indicated by the included value of ifOperStatus."
    ::= { snmpTraps 4 }
```

2.2.5 Textual Conventions (RFC 2579)

- Textual conventions are used to derive new types from SMIv2 base types.
- New types may not be derived from other TEXTUAL-CONVENTIONs.
- The DISPLAY-HINT clause defines a mapping from the internal representation into a human readable format and back.
- The DISPLAY-HINT clause can only be used with an INTEGER or an OCTET STRING base type.
- A textual convention supports formal restrictions (e.g. ranges).
- A textual convention does not allow to define complex types (e.g. structures or unions).

SMIv2 Textual Conventions Macro (RFC 2579)

```
<TypeDescriptor> ::= TEXTUAL-CONVENTION
        [ DISPLAY-HINT  <Text>   ]
          STATUS        <Status>
          DESCRIPTION   <Text>
        [ REFERENCE     <Text>   ]
          SYNTAX        <Syntax>
```

- The DISPLAY-HINT clause defines a mapping from the internal representation into a human readable format and back.
- The DISPLAY-HINT clause can only be used with an INTEGER or an OCTET STRING base type.
- The SYNTAX clause defines the (restricted) SMIv2 base type.
- Additional semantics are defined in the DESCRIPTION clause.

DISPLAY-HINTS (INTEGER)

Format	Description
`d`	decimal value
`d-<number>`	decimal value with a decimal point
`b`	binary value
`o`	octal value
`x`	hexadecimal value

Examples:

- " `d` " for value 143 renders 143
- "`d-2`" for value 143 renders 1.43
- " `b` " for value 143 renders 100001111
- " `o` " for value 143 renders 217
- " `x` " for value 143 renders 8F

DISPLAY-HINTS (OCTET STRING)

`[<repeat>]<number><format>[<separator>][<terminator>]`

Field	Description
`<repeat>`	number of times to apply the remainder of the specification (∗ indicates the current octet, default 1)
`<number>`	number of octets converted according to the next field
`<format>`	display format (a ascii, d decimal, x hexadecimal, o octal, t UTF-8)
`<separator>`	separator character
`<terminator>`	terminating character if `<repeat>` and `<separator>` parts are present

Examples:

- "255a" for value "aBc" renders "aBc"
- "1x:" for value "aBc" renders "61:42:63"
- "0aH0ae0al0al0ao 1a" for value "World" renders "Hello World"

Textual Convention Macro Examples (RFC 3411, RFC 2579)

```
SnmpSecurityLevel ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION "A Level of Security at which SNMP messages can be sent or with
                 which operations are being processed; in particular, one of:

                   noAuthNoPriv - without authentication and without privacy,
                   authNoPriv   - with authentication but without privacy,
                   authPriv     - with authentication and with privacy.

                 These three values are ordered such that noAuthNoPriv is less
                 than authNoPriv and authNoPriv is less than authPriv."
    SYNTAX       INTEGER { noAuthNoPriv(1), authNoPriv(2), authPriv(3) }
```

```
MacAddress ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "1x:"
    STATUS       current
    DESCRIPTION  "Represents an 802 MAC address represented in the
                  'canonical' order defined by IEEE 802.1a, i.e., as if it
                  were transmitted least significant bit first, even though
                  802.5 (in contrast to other 802.x protocols) requires MAC
                  addresses to be transmitted most significant bit first."
    SYNTAX       OCTET STRING (SIZE (6))
```

```
DateAndTime ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "2d-1d-1d,1d:1d:1d.1d,1a1d:1d"
    STATUS       current
    DESCRIPTION  "A date-time specification.

                 field  octets  contents                  range
                 -----  ------  --------                  -----
                   1      1-2   year*                     0..65536
                   2       3    month                     1..12
                   3       4    day                       1..31
                   4       5    hour                      0..23
                   5       6    minutes                   0..59
                   6       7    seconds                   0..60
                                (use 60 for leap-second)
                   7       8    deci-seconds              0..9
                   8       9    direction from UTC        '+' / '-'
                   9      10    hours from UTC*           0..13
                  10      11    minutes from UTC          0..59

                 * Notes:   - the value of year is in network-byte order
                            - daylight saving time in New Zealand is +13

                 For example, Tuesday May 26, 1992 at 1:30:15 PM EDT would be displayed as:

                                  1992-5-26,13:30:15.0,-4:0

                 Note that if only local time is known, then timezone
                 information (fields 8-10) is not present."
    SYNTAX       OCTET STRING (SIZE (8 | 11))
```

2.2.6 Conformance Statements

- Conformance statements define implementation requirements.
- Object type and notification type definitions may be grouped together.
- Groups can be mandatory, optional or conditionally optional.
- The maximum access of an object type definition may be revised.
- The type associated with an object type may be refined.
- It is possible to have different type refinements for read and write operations.

SMIv2 Object Group Macro (RFC 2580)

```
<Descriptor> OBJECT-GROUP
        OBJECTS       <Objects>
        STATUS        <Status>
        DESCRIPTION   <Text>
      [ REFERENCE     <Text>      ]
      ::= <ObjectIdentifier>
```

- Defines a collection of related managed object types.
- Object groups can be used to define conformance levels.
- The status of the object group definition must conform to the status of the objects types listed in the OBJECTS clause.

SMIv2 Notification Group Macro (RFC 2580)

```
<Descriptor> NOTIFICATION-GROUP
        NOTIFICATIONS <Notifications>
        STATUS        <Status>
        DESCRIPTION   <Text>
       [REFERENCE     <Text>       ]
       ::= <ObjectIdentifier>
```

- Defines a collection of related notification types.
- Notification groups can be used to define conformance levels.
- The status of the notification group definition must conform to the status of the notifications listed in the NOTIFICATIONS clause.

Object Group and Notification Group Examples (RFC 3418)

```
systemGroup OBJECT-GROUP
    OBJECTS     { sysDescr, sysObjectID, sysUpTime,
                  sysContact, sysName, sysLocation,
                  sysServices,
                  sysORLastChange, sysORID,
                  sysORUpTime, sysORDescr }
    STATUS      current
    DESCRIPTION "The system group defines objects which are common
                 to all managed systems."
    ::= { snmpMIBGroups 6 }
```

```
snmpBasicNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS { coldStart, authenticationFailure }
    STATUS        current
    DESCRIPTION   "The basic notifications implemented by an SNMP
                   entity supporting command responder applications."
    ::= { snmpMIBGroups 7 }
```

SMIv2 Module Compliance Macro (RFC 2580)

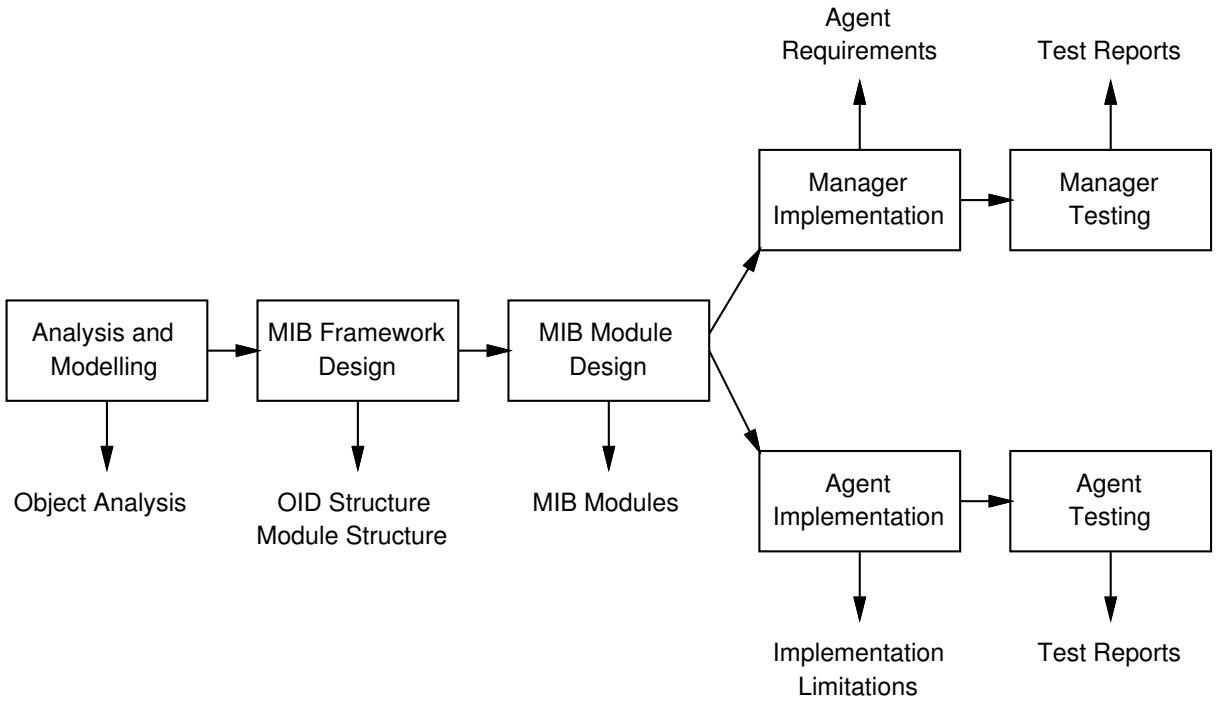
```
<Descriptor> MODULE-COMPLIANCE
        STATUS                  <Status>
        DESCRIPTION             <Text>
      [ REFERENCE               <Text>                ]
      [ MODULE                  <Name>
      [ MANDATORY-GROUPS        <Groups>              ]
      [ GROUP                   <ObjectIdentifier>
        DESCRIPTION             <Text>                ]*
      [ OBJECT                  <ObjectIdentifier>
        DESCRIPTION             <Text>                ]* ]*
      ::= <ObjectIdentifier>
```

- This is a simplified version!
- Specifies implementation requirements for a compliant implementation.
- Allows to express unconditional and conditional requirements.

Module Compliance Macro Examples (RFC 3418, RFC 3414)

```
snmpBasicComplianceRev2 MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION "The compliance statement for SNMP entities which
                 implement this MIB module."
    MODULE       -- this module
        MANDATORY-GROUPS { snmpGroup, snmpSetGroup, systemGroup, snmpBasicNotificationsGroup }
        GROUP       snmpCommunityGroup
        DESCRIPTION "This group is mandatory for SNMP entities which
                     support community-based authentication."
        GROUP       snmpWarmStartNotificationGroup
        DESCRIPTION "This group is mandatory for an SNMP entity which supports command responder
                     applications, and is able to reinitialize itself such that its configuration is unaltered."
    ::= { snmpMIBCompliances 3 }


usmMIBCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION "The compliance statement for SNMP engines which
                 implement the SNMP-USER-BASED-SM-MIB."
    MODULE       -- this module
        MANDATORY-GROUPS { usmMIBBasicGroup }
        OBJECT          usmUserAuthProtocol
        MIN-ACCESS      read-only
        DESCRIPTION     "Write access is not required."
        OBJECT          usmUserPrivProtocol
        MIN-ACCESS      read-only
        DESCRIPTION     "Write access is not required."
    ::= { usmMIBCompliances 1 }
```

2.2.7 MIB Module Design, Implementation and Testing



- Iterations are possible and useful before MIB definitions are released.
- MIB definitions that have been released can not be modified anymore.

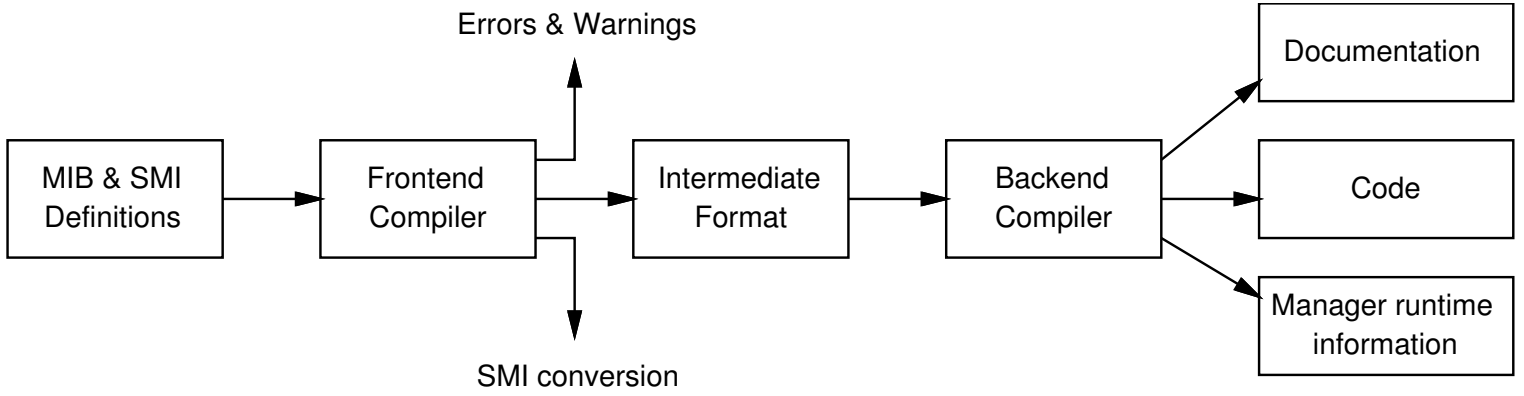
MIB Analysis and Design

- Components
  - Collections of logical or physical devices or services.
- Attributes
  - "Stable" attributes that describe the management aspects of a component.
- Relationships
  - Relationships between components (containment, logical order, ...)
  - Relationships between attributes of components.
  - Cardinality of components and attributes.
- State Attributes
  - Describe the desired or current state of a component.
  - Use state diagrams to document state transitions.
- Statistic Attributes
  - Statistic attributes document what a component has done in the past.
  - Statistics are bound to a lifetime which should be well-defined.

MIB Naming Conventions

- Related object type definitions are grouped into a subtree.
- Object type names start with a common prefix to identify the logical group (e.g. sysUpTime, sysServices).
- Counter object type names use the plural spelling (e.g. ifInOctets).
- Conceptual table names end with the suffix Table (e.g. ifTable).
- Conceptual row names end with the suffix Entry (e.g. ifEntry).
- All (conceptual table) definitions share a common prefix (e.g. ifType, ifDescr).

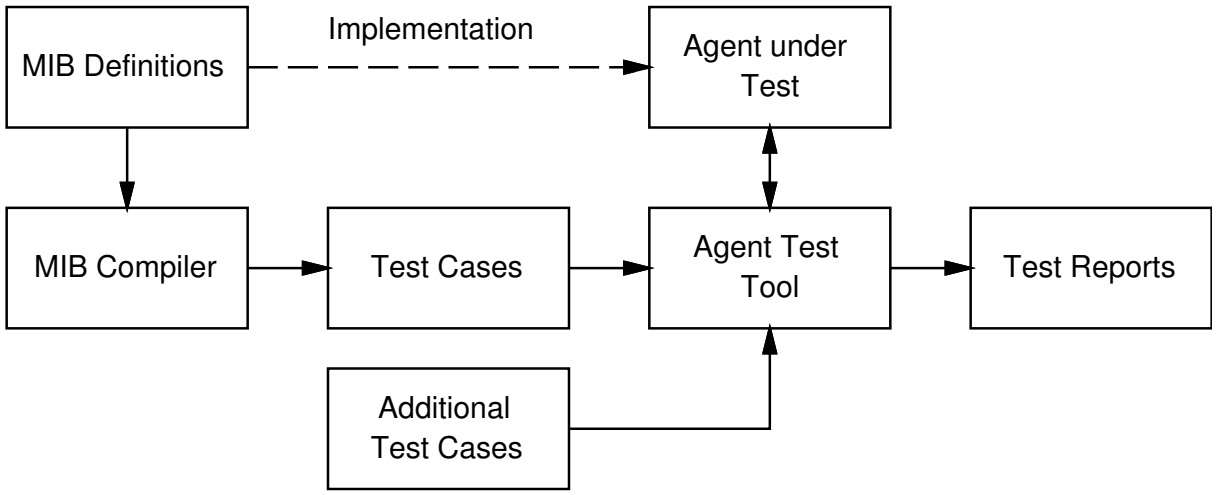
MIB Compiler



- Front-end compilers perform syntax checks and to some extend semantic checks.

- Back-end compilers can generate
  - documentation (HTML versions of MIB modules, graphical representations),
  - code to automate some steps of the implementation process,
  - test suites to test agent or manager implementation,
  - input for management applications that interpret MIB definitions at run time.

Testing MIB Implementations



- Automated regression tests help to ensure the quality of the MIB implementation.
- Many test cases can be generated automatically from the MIB definition.
- Additional test cases should be written by an independent engineer.

Typical MIB Implementation Problems

- Caching:
  - Access to management information might be expensive.
  - Solution: Try to cache some information in the access method to reduce the number of costly interactions.
  - Problem: How to define the lifetime of the cache?
- Atomic Sets:

  It is sometimes difficult to guarantee "as if simultaneous" semantics because:
  - Variable names and values can come in any order.
  - A single set may include multiple values for a single instance.
  - It is difficult to maintain state during a sequence of set operations.

2.3 Communication Protocol SNMPv3

2.3.1 Architectural Concepts

2.3.2 Protocol Operations

2.3.3 Message Format

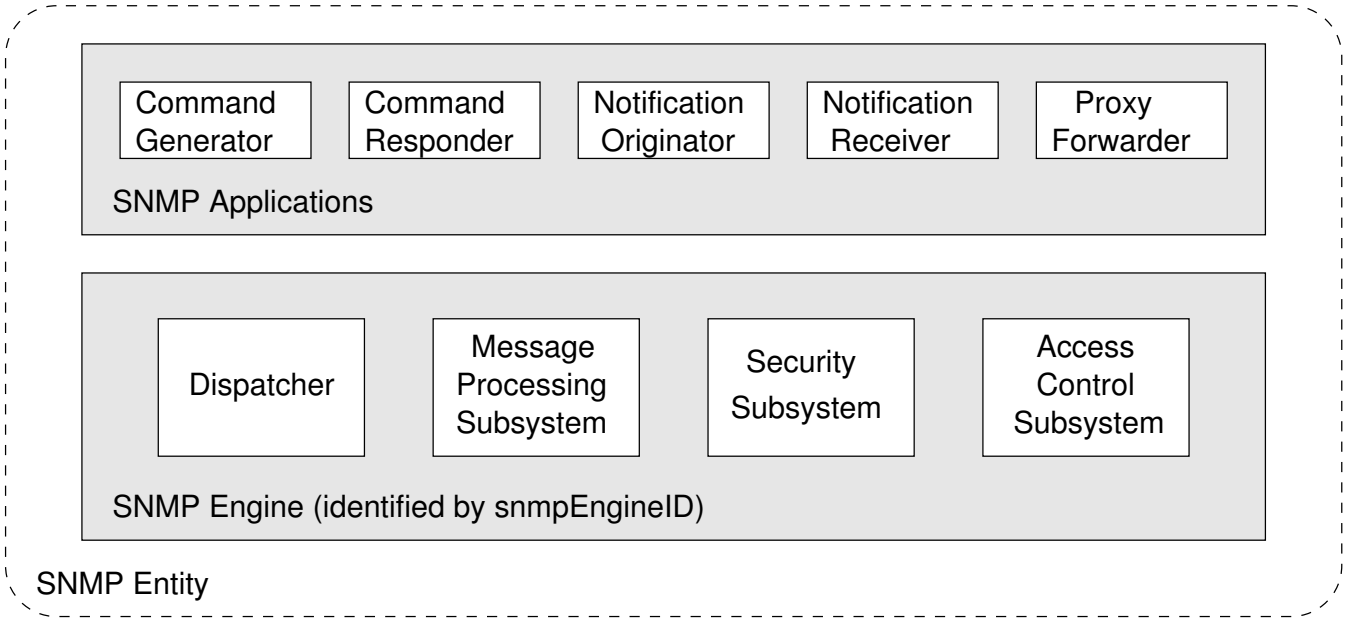
2.3.4 Authentication and Privacy

2.3.5 Authorization and Access Control

2.3.6 Remote Configuration

2.3.7 Status and Limitations

2.3.1 Architectural Concepts (RFC 3411)

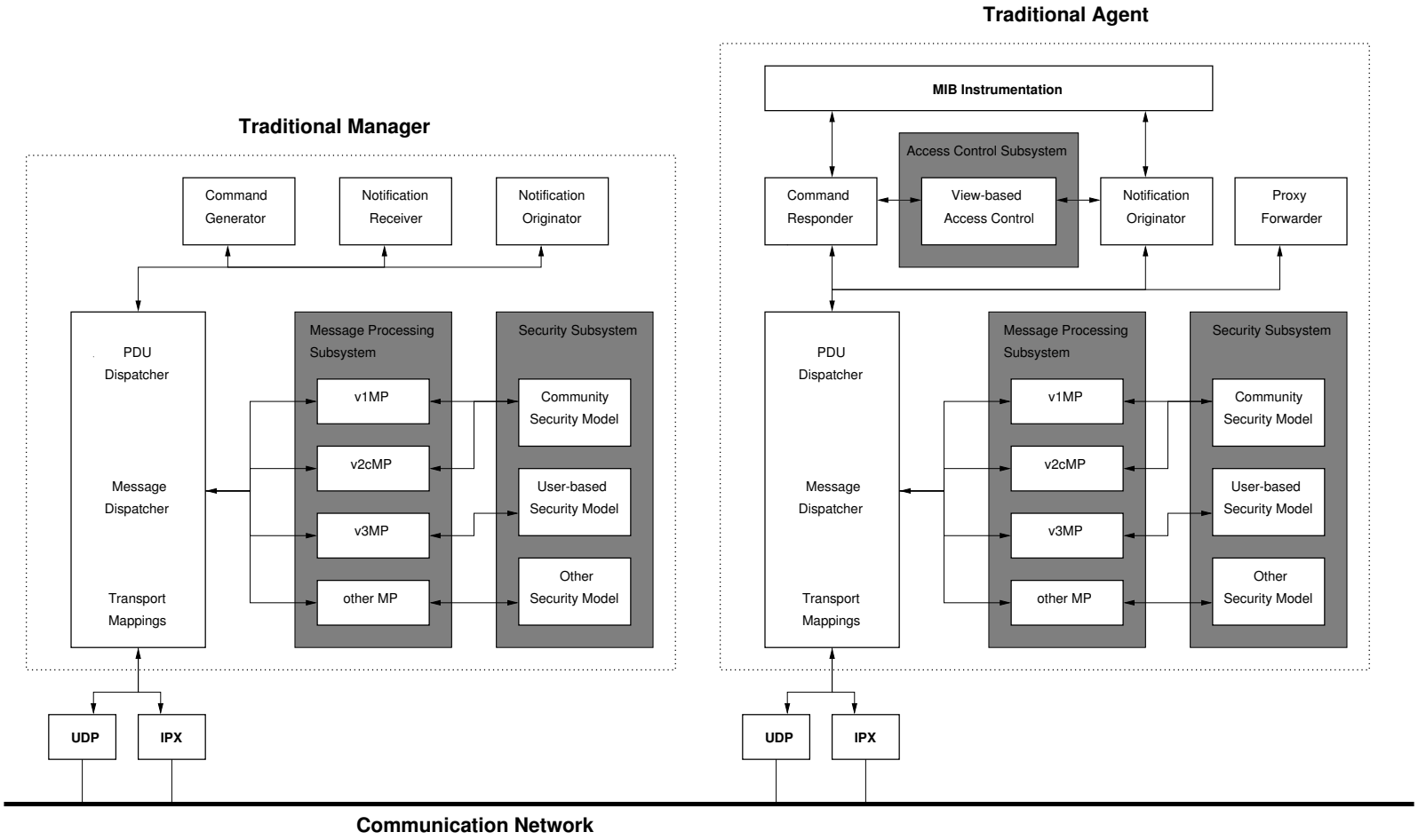


- Fine grained SNMP applications instead of coarse grained agents and managers.
- Exactly on engine per SNMP entity and exactly one dispatcher per SNMP engine.
- Every abstract subsystem may consist of one or more concrete models.
- Modularization enables incremental enhancements to SNMP.

SNMP Contexts

- An context is a collection of management information accessible by an SNMP entity.
  - An SNMP entity potentially has access to many contexts.
  - An item of management information may exist in more than one context.
- Within a management domain, a managed object is uniquely identified by:
  1. the identification of the engine within the SNMP entity (e.g. "xyz")
  2. the context name within the SNMP entity (e.g. "board1")
  3. the managed object type (e.g. "IF-MIB.ifDescr")
  4. the instance identifier (e.g. "1")

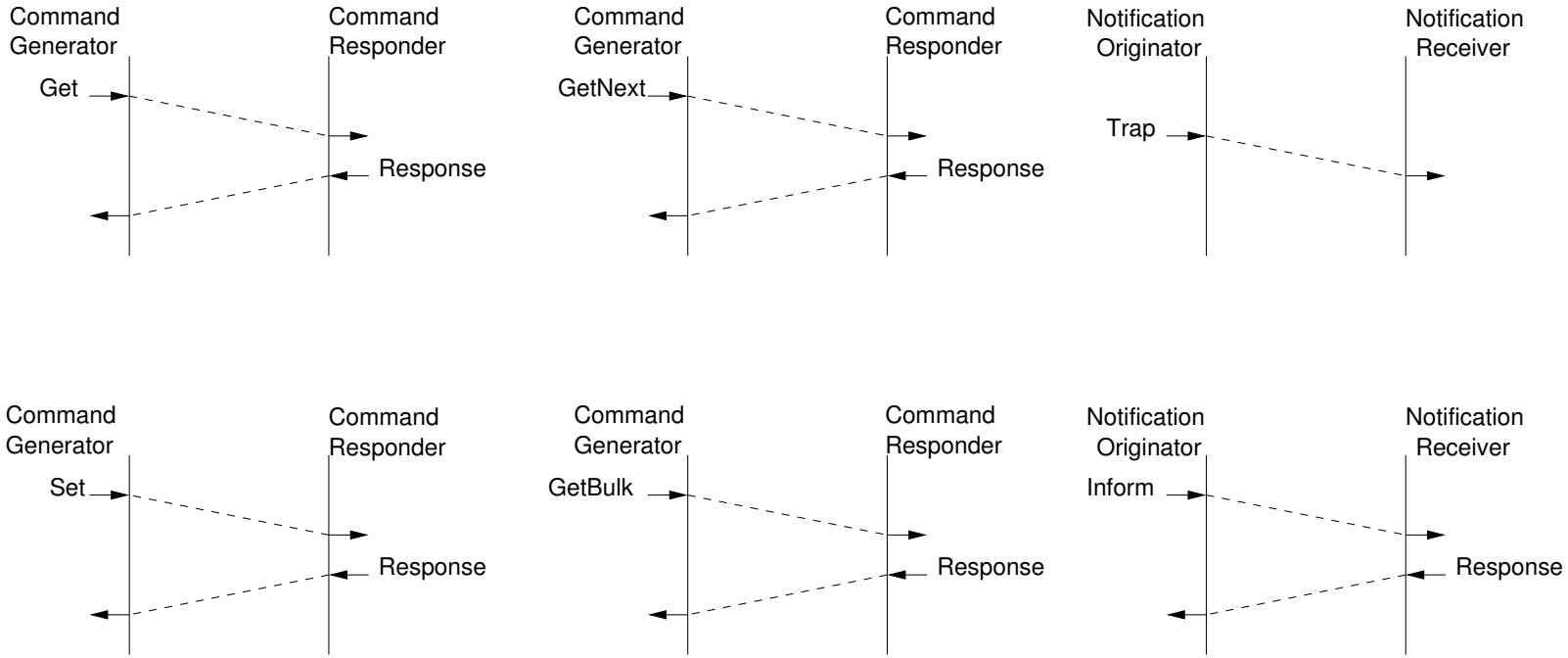
Manager and Agent in the SNMP Architecture



SNMPv3/USM Textual Conventions

- SnmpEngineID
  - Unique identification of an SNMP engine within a management domain.
- SnmpSecurityModel
  - Identification of a specific security model.
- SnmpMessageProcessingModel
  - Identification of a specific message processing model.
  - The message processing model is encoded in the `msgVersion`.
- SnmpSecurityLevel
  - The security level of a given message (`noAuthNoPriv`, `authNoPriv`, `authPriv`).
  - The security level is encoded in the `msgFlags`.
- KeyChange
  - Defines a cryptographic algorithm to change authentication or encryption keys.
  - Does not require encryption. An attacker can "drill forward" once a key is broken.

2.3.2 Protocol Operations (RFC 3416)



- An additional `Report` protocol operation is used internally for error notifications, engine discovery and clock synchronization.

Lexicographic Ordering

- Given are two vectors of natural numbers $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$ with $n \leq m$. We say that x is lexicographically less than y if and only if one of the following conditions is true:

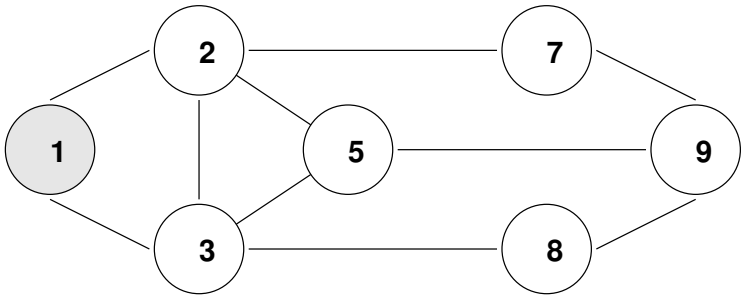
  (a) $x_j = y_j$ for $1 \leq j \leq k$ and $x_k < y_k$ with $k \leq n$ and $k \leq m$

  (b) $x_j = y_j$ for $1 \leq j \leq n$ and $n < m$

- All OIDs identifying instances can be lexicographically ordered.

- The SNMP protocol operates only on the lexicographically ordered list of MIB instances and not on the OID registration tree or on conceptual tables.

Example of a Simple Routing Table



Example for Lexicographic Ordering

- Lexicographic ordered list of MIB instanced for the example MIB with the simple routing table:

OID	name	value
1.1.0	address.0	10.1.2.1
1.2.1.0	name.0	"ACME Router"
1.2.2.0	uptime.0	54321
1.3.1.1.1	fwdIndex.1	1
1.3.1.1.2	fwdIndex.2	2
1.3.1.1.3	fwdIndex.3	3
1.3.1.1.4	fwdIndex.4	4
1.3.1.1.5	fwdIndex.5	5
1.3.1.1.6	fwdIndex.6	6
1.3.1.2.1	fwdDest.1	2
1.3.1.2.2	fwdDest.2	3

OID	name	value
1.3.1.2.3	fwdDest.3	5
1.3.1.2.4	fwdDest.4	7
1.3.1.2.5	fwdDest.5	8
1.3.1.2.6	fwdDest.6	9
1.3.1.3.1	fwdNext.1	2
1.3.1.3.2	fwdNext.2	3
1.3.1.3.3	fwdNext.3	2
1.3.1.3.4	fwdNext.4	2
1.3.1.3.5	fwdNext.5	3
1.3.1.3.6	fwdNext.6	3

- Conceptual table instances are ordered column by column not row by row.
- It is not necessary (and actually forbidden by SMIv2) to make index columns accessible.

Errors versus Exceptions

- Errors:
  - An error response signals the complete failure of the corresponding request.
  - An error response contains an error status and an error index.
  - Error responses contain no useful management information.
  - There is only a single error status and error index even if there are multiple errors.

- Exceptions:
  - A response can contain per variable binding exceptions.
  - One or more exceptions in a response are not considered to be an error condition of the corresponding request.
  - A response with exceptions still contains useful management information.

Protocol Operation Error Codes (RFC 3416)

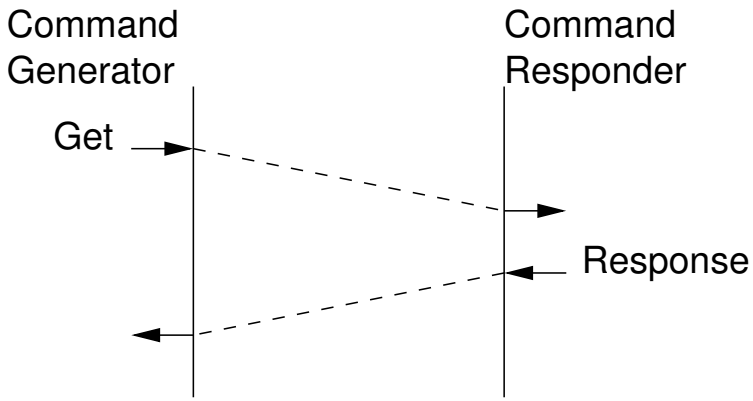
SNMPv3 Error Code	Get/GetNext/GetBulk	Set	Trap/Inform	SNMPv1 Error Code
noError(0)	X	X	X	noError(0)
tooBig(1)	X	X	X	tooBig(1)
noSuchName(2)				noSuchName(2)
badValue(3)				badValue(3)
readOnly(4)				readOnly(4)
genErr(5)	X	X	X	genErr(5)
noAccess(6)		X		noSuchName(2)
wrongType(7)		X		badValue(3)
wrongLength(8)		X		badValue(3)
wrongEncoding(9)		X		badValue(3)
wrongValue(10)		X		badValue(3)
noCreation(11)		X		noSuchName(2)
inconsistentValue(12)		X		badValue(3)
resourceUnavailable(13)		X		genErr(5)
commitFailed(14)		X		genErr(5)
undoFailed(15)		X		genErr(5)
authorizationError(16)	X	X	X	noSuchName(2)
notWritable(17)		X		noSuchName(2)
inconsistentName(18)		X		noSuchName(2)

Protocol Operation Exceptions (RFC 3416)

SNMPv3 Exception	Get	GetNext/GetBulk	SNMPv1 Error Status
noSuchObject	X		noSuchName(2)
noSuchInstance	X		noSuchName(2)
endOfMibView		X	noSuchName(2)

- Applications receiving response messages must check the error code and detect exceptions and deal with them gracefully.
- The `noSuchInstance` exceptions indicates that a particular instances does not exist, but that other instances of the object type can exist.
- The `noSuchObject` exception indicates that a certain object type is not available.
- This distinctions allows smart applications to adapt to the capabilities of a particular command responder implementation.

SNMPv3 Get Operation (RFC 3416)



- The `Get` operation is used to read one or more MIB variables.

- Possible error codes:
  - tooBig — The response is too big to return it to the manager.
  - genErr — Some other error happened during Get processing.

- Possible exceptions:
  - noSuchObject — An object type is not supported.
  - noSuchInstance — An object instance is not available.

Example SNMPv3 Get Operations

1. `Get(1.1.0)`
   `Response(noError@0, 1.1.0=10.1.2.1)`

2. `Get(1.2.0)`
   `Response(noError@0, 1.2.0=noSuchObject)`

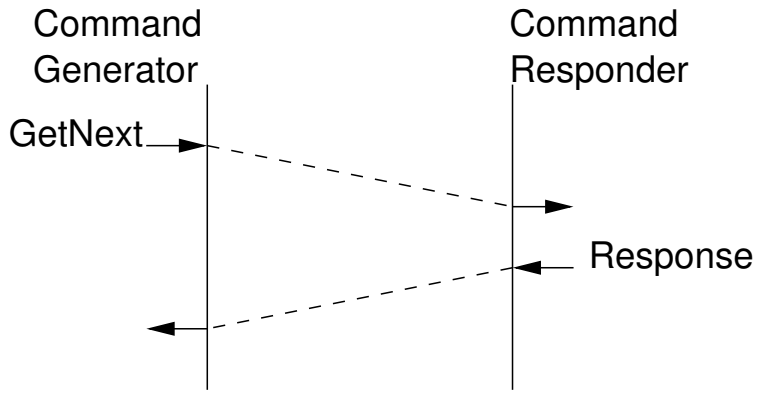
3. `Get(1.1.1)`
   `Response(noError@0, 1.1.1=noSuchInstance)`

4. `Get(1.1.0, 1.2.2.0)`
   `Response(noError@0, 1.1.0=10.1.2.1, 1.2.2.0=54321)`

5. `Get(1.3.1.1.4, 1.3.1.3.4)`
   `Response(noError@0, 1.3.1.1.4=4, 1.3.1.3.4=2)`

6. `Get(1.1.0, 1.1.1)`
   `Response(noError@0, 1.1.0=10.1.2.1, 1.1.1=noSuchInstance)`

SNMPv3 GetNext Operation (RFC 3416)



- The `GetNext` operation allows to retrieve the value of the next existing MIB instances in lexicographic order.
- Successive `GetNext` operations can be used to walk the MIB instances without prior knowledge about the MIB structure.
- Possible error codes:
  - tooBig — The response is too big to return it to the manager.
  - genErr — Some other error happened during GetNext processing.
- Possible exceptions:
  - endOfMibView — There is lexicographic next instance in the MIB view.

Example SNMPv3 GetNext Operations

1. `GetNext(1.1.0)`
   `Response(noError@0, 1.2.1.0="ACME Router")`

2. `GetNext(1.2.1.0)`
   `Response(noError@0, 1.2.2.0=54321)`

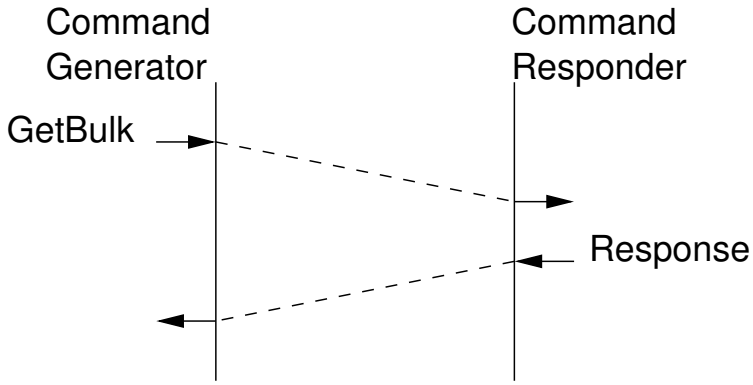
3. `GetNext(1.1)`
   `Response(noError@0, 1.1.0=10.1.2.1)`

4. `GetNext(1.3.1.1.1)`
   `Response(noError@0, 1.3.1.1.2=2)`

5. `GetNext(1.3.1.1.6)`
   `Response(noError@0, 1.3.1.2.1=2)`

6. `GetNext(1.3.1.1.1, 1.3.1.2.1, 1.3.1.3.1)`
   `Response(noError@0, 1.3.1.1.2=2, 1.3.1.2.2=3, 1.3.1.3.2=3)`

SNMPv3 GetBulk Operation (RFC 3416)



- The `GetBulk` operation is a generalization of the `GetNext` operation:
  - The first N elements (`non-repeaters`) of the varbind list will be processed similar to the `GetNext` operation.
  - The remaining R elements of the varbind list are repeatedly processed similar to the `GetNext` operation.
  - The parameter M (`max-repetitions`) defines the upper bound of repetitions.

- The `GetBulk` operation operates only on the lexicographically ordered list of MIB instances and does therefore not respect conceptual table boundaries.

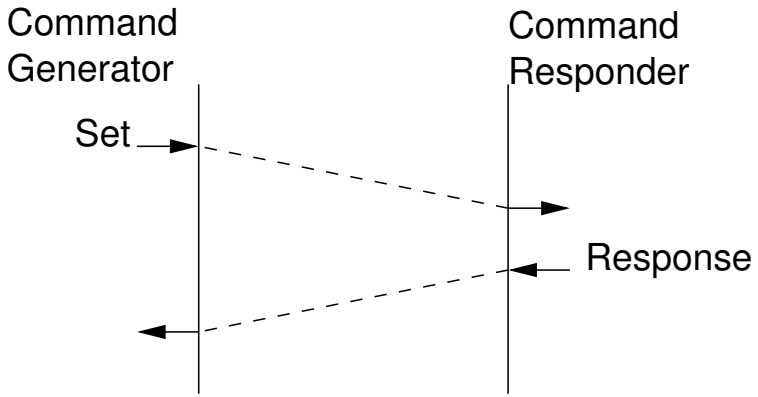
Example SNMPv3 GetBulk Operations

1. `GetBulk(non-repeaters=0, max-repetitions=4, 1.1)`
   `Response(noError@0, 1.1.0=10.1.2.1, 1.2.1.0="ACME Router",`
   `1.2.2.0=54321, 1.3.1.1.1=1)`

2. `GetBulk(non-repeaters=1, max-repetitions=2, 1.2.2,`
   `1.3.1.1, 1.3.1.2, 1.3.1.3)`
   `Response(noError@0, 1.2.2.0=54321,`
   `1.3.1.1.1=1, 1.3.1.2.1=2, 1.3.1.3.1=2,`
   `1.3.1.1.2=2, 1.3.1.2.2=3, 1.3.1.3.2=3)`

- The manager usually does not know how to choose a value for `max-repetitions`.
- Single-threaded agents can delay the processing of other requests during `GetBulk` processing so much that other managers start sending retries, which makes things even worse.

SNMPv3 Set Operation (RFC 3416)



- The `Set` operation allows to modify a set of MIB instances.
- The `Set` operation is atomic (either all instances are modified or none).
- Possible error codes:

`wrongValue`	`wrongEncoding`	`wrongType`
`wrongLength`	`inconsistentValue`	`noAccess`
`notWritable`	`noCreation`	`inconsistentName`
`resourceUnavailable`	`commitFailed`	`undoFailed`

- The error codes `readOnly` and `authorizationError` are not used.
- No support for object type specific error codes.

Example SNMPv3 Set Operations

1. `Set(1.2.1.0="Moo Router")`
   `Response(noError@0, 1.2.1.0="Moo Router")`

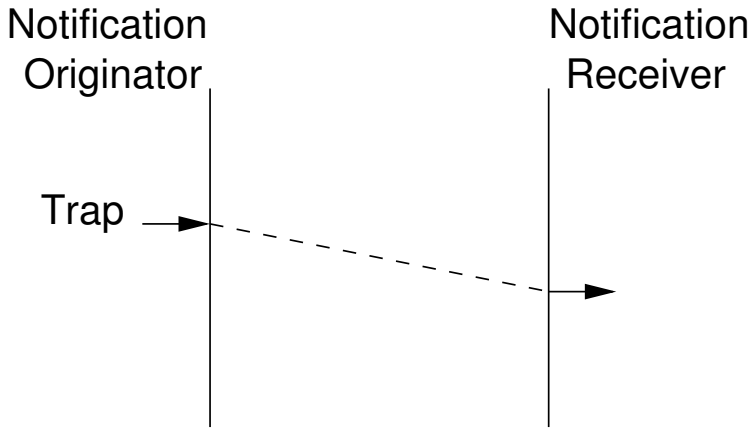
2. `Set(1.1.0="foo.bar.com")`
   `Response(badValue@1, 1.1.0="foo.bar.com")`

3. `Set(1.1.1=10.2.3.4)`
   `Response(noSuchName@1, 1.1.1=10.2.3.4)`

4. `Set(1.2.1.0="Moo Router", 1.1.0="foo.bar.com")`
   `Response(badValue@2, 1.2.1.0="Moo Router", 1.1.0="foo.bar.com")`

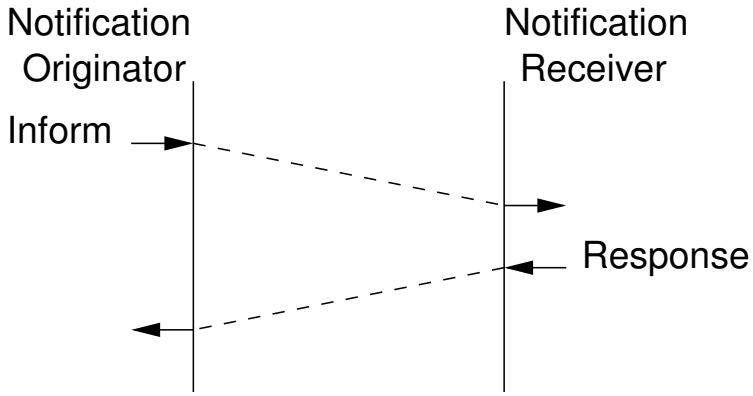
5. `Set(1.3.1.1.7=7, 1.3.1.2.7=2, 1.3.1.3.7=3)`
   `Response(noError@0, 1.3.1.1.7=7, 1.3.1.2.7=2, 1.3.1.3.7=3)`

SNMPv3 Trap Operation (RFC 3416)



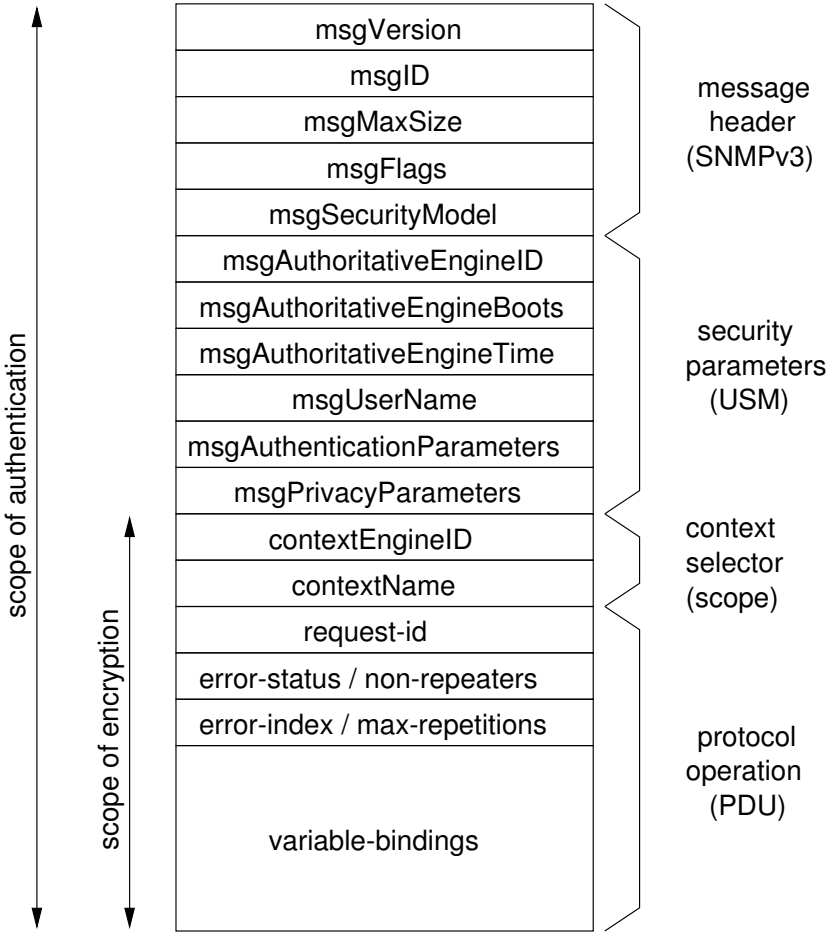
- The `Trap` operation is used to notify a manager of the occurance of an event.
- The `Trap` operation is unconfirmed: The sending agent does not know whether the trap was received and processed by a manager.
- Some operational conditions can result in trap storms (e.g. all systems rebooting after a power outage).
- All trap specific information in encoded in the varbind list. (`sysUpTime`, `snmpTrapOID`, `snmpTrapEnterprise`)

SNMPv3 Inform Operation (RFC 3416)



- The `Inform` operations is used as a confirmed trap (although it was not originally designed for that purpose).
- The contents of the varbind list of an `Inform` operation is similar to that of a `Trap` operation.
- The reception of an `Inform` operation is confirmed by a response.
- The confirmation does only indicate that the notification was delivered, not that the notification was actually understood and dealt with.

2.3.3 Message Format (RFC 3412, RFC 3414)



- `msgVersion` identifies the message processing model.
- `msgSecurityModel` identifies the security model.
- `contextEngineID` and `contextName` determine the context.
- The type of protocol operation (and version) is determined by the tag of the PDU.

Classes of Protocol Operations (RFC 3411)

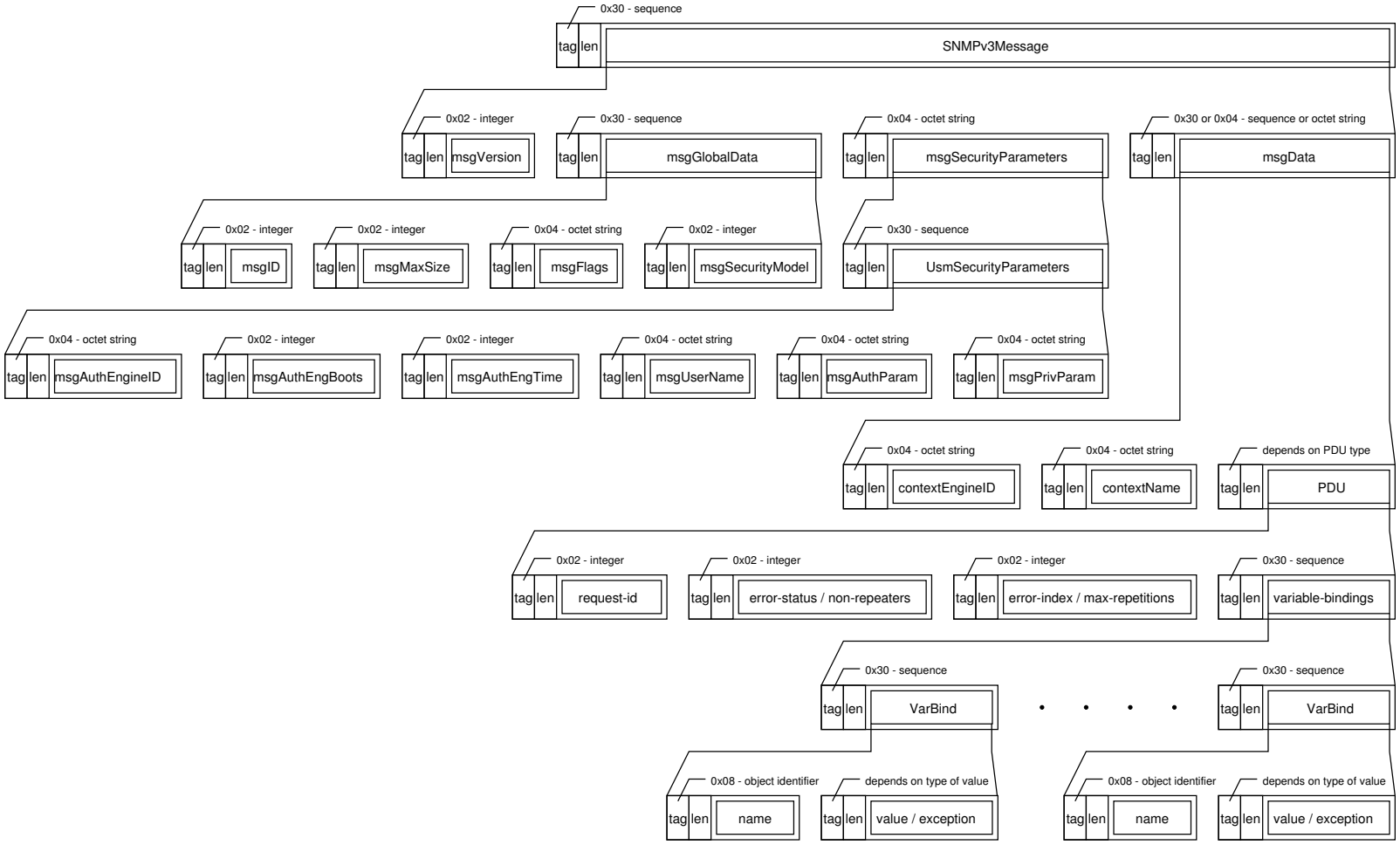
- The processing of a message depends on the class of the embedded protocol operation:

Class	Description
Read	PDUs that retrieve management information.
Write	PDUs which attempt to modify management information.
Response	PDUs which are sent in response to a request.
Notification	PDUs which transmit event notifications.
Internal	PDUs exchanged internally between SNMP engines.
Confirmed	PDUs which cause the receiver to send a response.
Unconfirmed	PDUs which are not acknowledged.

- The introduction of PDU classes enables the IETF to add new protocol operations without having to update the message processing specification.

- There is no explicit support in the message format to indicate the protocol operations supported/used by an SNMP engine.

Encoding of SNMPv3/USM Messages



Security Issues

- The following questions must be answered in order to decide whether an operation should be performed or not:

  1. Is the message specifying an operation authentic?
  2. Who requested the operation to be performed?
  3. What objects are accessed in the operation?
  4. What are the rights of the requester with regard to the objects of the operation?

- 1 and 2 are answered by message security mechanisms (authentication and privacy).
- 3 and 4 are answered by authorization mechanisms (access control).

2.3.4 Authentication and Privacy (RFC 3414)

- Protection against the following threads:
  1. Modification of Information
     (Unauthorized modification of in-transit SNMP messages.)
  2. Masquerade
     (Unauthorized users attempting to use the identity of authorized users.)
  3. Disclosure
     (Protection against eavesdropping on the exchanges between SNMP entities.)
  4. Message Stream Modification
     (Re-ordered, delayed or replayed messages to effect unauthorized operations.)

- No protection against:
  - Denial of Service
    (Denial of service attacks are usually indistinguishable from network failures.)
  - Traffic Analysis
    (No significant advantage afforded by protecting against traffic analysis.)

USM Security Services (RFC 3414)

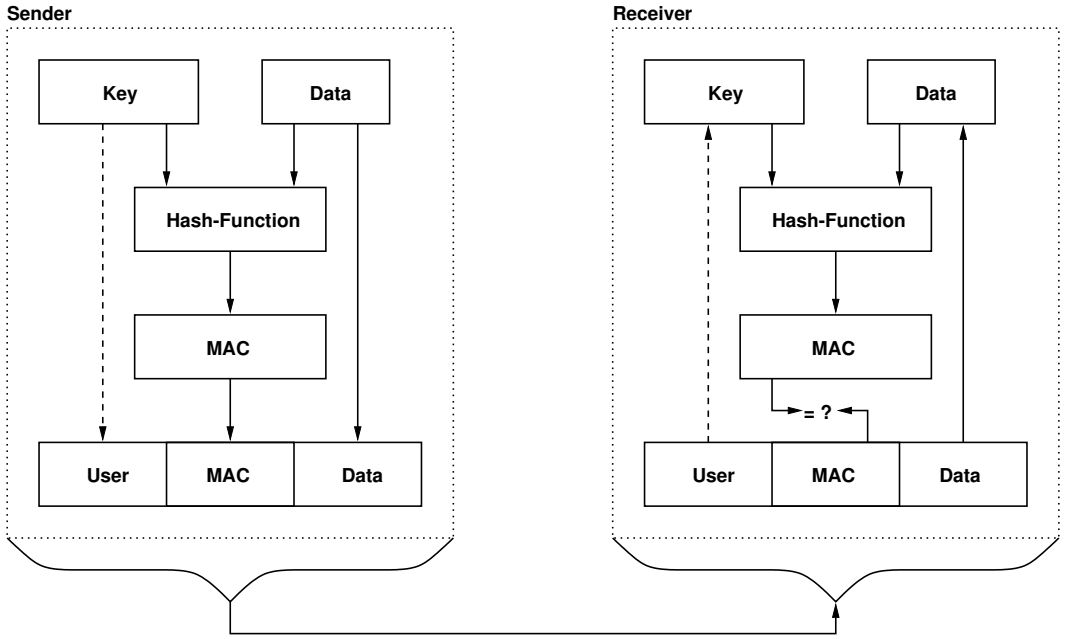
- Data Integrity
  - Data has not been altered or destroyed in an unauthorized manner.
  - Data sequences have not been altered to an extent greater than can occur non-maliciously.

- Data Origin Authentication
  - The claimed identity of the user on whose behalf received data was originated is corroborated.

- Data Confidentiality
  - Information is not made available or disclosed to unauthorized individuals, entities, or processes.

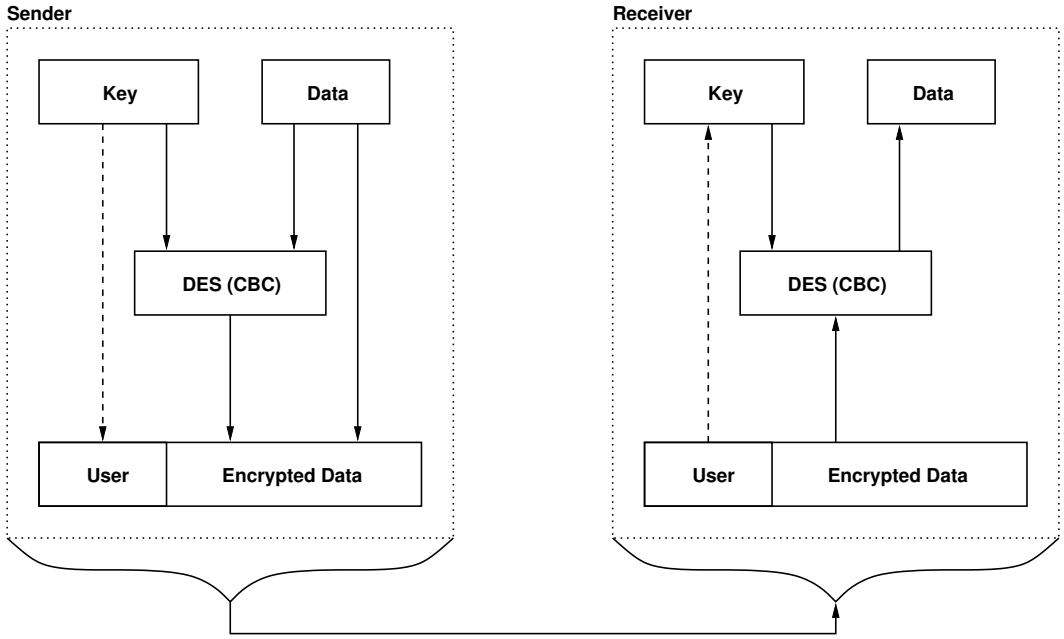
- Message Timeliness and Limited Replay Protection
  - A message whose generation time is outside of a time window is not accepted.
  - Message reordering is not dealt with and can occur in normal conditions too.

USM Data Integrity and Data Origin Authentication



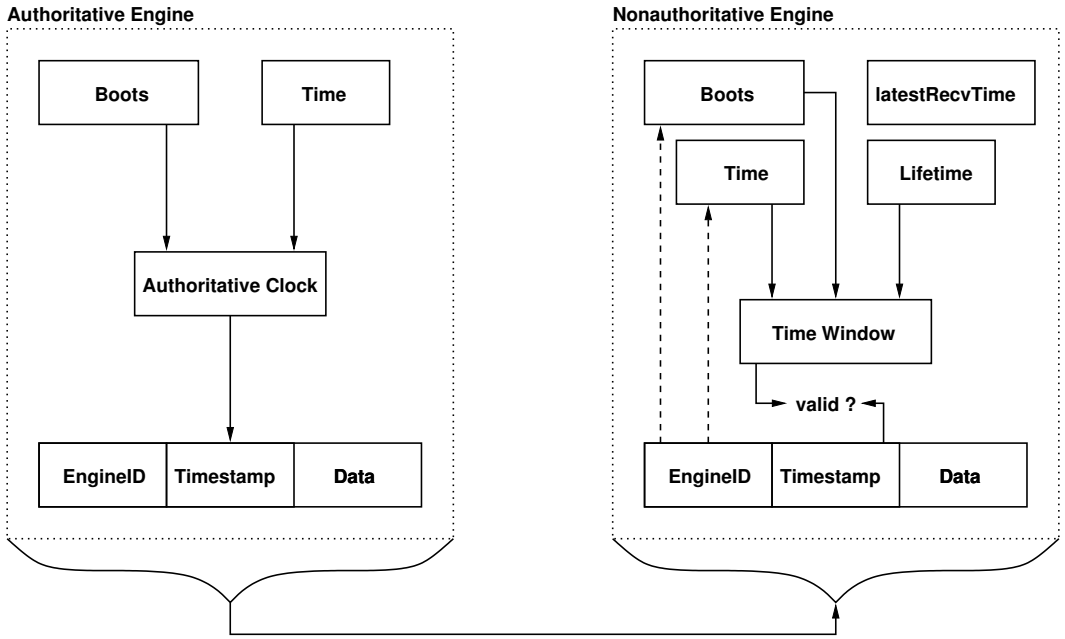
- Cryptographic strong hash functions generate a message authentication code (MAC).
- The MAC ensures integrity while the symmetric key provides for authentication.
- USM currently defines HMAC-MD5-96 and HMAC-SHA-96 (96 bits in the message).
- Other hash functions may be added in the future without changes to USM.

USM Data Confidentiality



- Optional encryption of the `ScopedPDU` using symmetric keys.
- USM currently defines CBC-DES (cipher-block-chaining mode).
- Other encryption functions may be added in the future without changes to USM.
- Encryption is relatively expensive and should only be used when needed.

USM Message Timeliness and Replay Protection



- A non-authoritative engine maintains a notion of the time at the authoritative engine.
- A non-authoritative engine also keeps track when the last authentic message was received from a given engine.
- A message is accepted and considered "fresh" if it falls within a time window.
- The maximum width of the time window is defined to be 300 seconds (5 minutes).

USM Passwords and Localized Keys

- Generating keys from passwords:
  - Algorithmic transformation of a human readable password into a cryptographic key.
  - Produce a string S of length $2^{20} = 1048576$ bytes by repeating the password as many times as necessary.
  - Compute the users key K_U using either $K_U = MD5(S)$ or $K_U = SHA(S)$.
  - Intended to slow down brute force password attacks.
  - No serious barrier for an attacker with a transformed dictionary.
- Generating localized keys:
  - Algorithmic transformation of the users key K_U and an engine identification E into a localized key.
  - For a given engine E, compute either $K_{UL} = MD5(K_U, E, K_U)$ or $K_{UL} = SHA(K_U, E, K_U)$.
  - Advantage: A compromised key does not give access to other SNMP engines.

USM Key Changes

- Key changes must be possible without encryption since encryption is optional.
- Key change procedure (initiator):
  1. Generate a random value r from a random number generator.
  2. Compute $d = MD5(K_{old}, r)$ or $d = SHA(K_{old}, r)$.
  3. Compute $\delta = d \oplus K_{new}$ and send (δ, r).
- Key change procedure (receiver):
  1. Compute $d = MD5(K_{old}, r)$ or $d = SHA(K_{old}, r)$.
  2. Compute $K_{new} = d \oplus \delta$.
- The receiver computes the correct new key since $d \oplus \delta = d \oplus (d \oplus K_{new}) = K_{new}$.
- An attacker who is able to catch all key updates can calculate the new keys once the old key is broken.

$\Rightarrow$ Use encryption for key changes if at all possible!

Authoritative Engine and Timeliness Checks

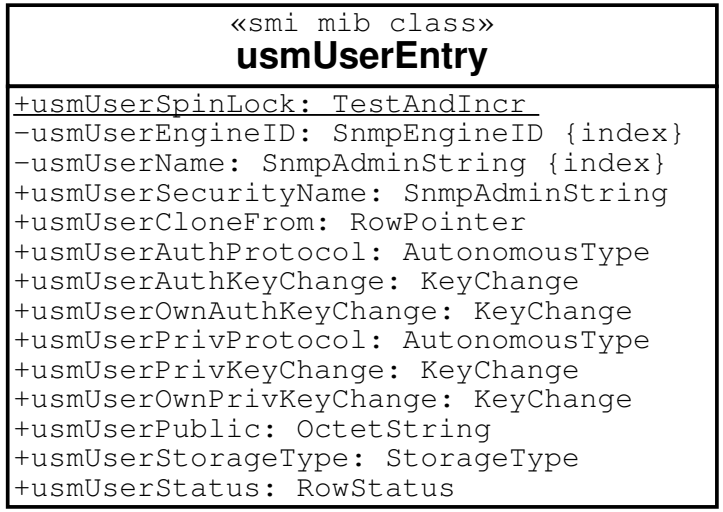
- Authoritative Engine:
  - Either the sender or the receiver of a message is designated the authoritative engine.
  - The receiver is authoritative if the message contains a confirmed class PDU.
  - The sender is authoritative if the message contains an unconfirmed class PDU.
- Timeliness check of an authoritative receiver:
  - A message is outside the time window if any of the following is true:
    1. `snmpEngineBoots` $= 2^{31} - 1$
    2. `msgAuthoritativeEngineBoots` $\neq$ `snmpEngineBoots`
    3. $abs($`msgAuthoritativeEngineTime` $-$ `snmpEngineTime`$) > 150$ seconds
- Timeliness check of a non-authoritative receiver:
  - A message is outside the time window if any of the following is true:
    1. `snmpEngineBoots` $= 2^{31} - 1$
    2. `msgAuthoritativeEngineBoots` $<$ `snmpEngineBoots`
    3. `msgAuthoritativeEngineBoots` $=$ `snmpEngineBoots` and `msgAuthoritativeEngineTime` $<$ `snmpEngineTime` -150

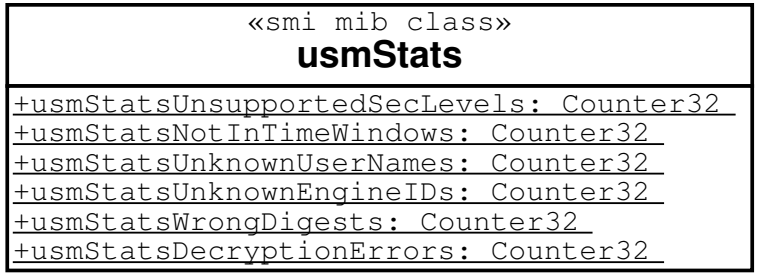
USM Clock Synchronization and Discovery

- An SNMP engine must remain loosely synchronized with other authoritative engines:
  - For each remote authoritative SNMP engine, an SNMP engine maintains: `snmpEngineBoots`, `snmpEngineTime` and `latestReceivedEngineTime`
  - Time synchronization only occurs if the message is authentic.
  - An update occurs, if at least one of the following conditions is true:
    1. `msgAuthoritativeEngineBoots` $>$ `snmpEngineBoots`
    2. `msgAuthoritativeEngineBoots` $=$ `snmpEngineBoots` and `msgAuthoritativeEngineTime` $>$ `latestReceivedEngineTime`

- Discovery and initial clock synchronization:
  - The engine identification is needed to compute localized keys and to keep clock information for authoritative engines.
  - An SNMP engine can learn the engine identification by sending a `noAuthNoPriv` request with a zero-length `msgAuthoritativeEngineID`.
  - The receiver returns a `Report` PDU with the real `msgAuthoritativeEngineID`.
  - Similarly, (initial) clock synchronization happens by sending an authentic request and receiving a `Report` PDU with the authoritative time.

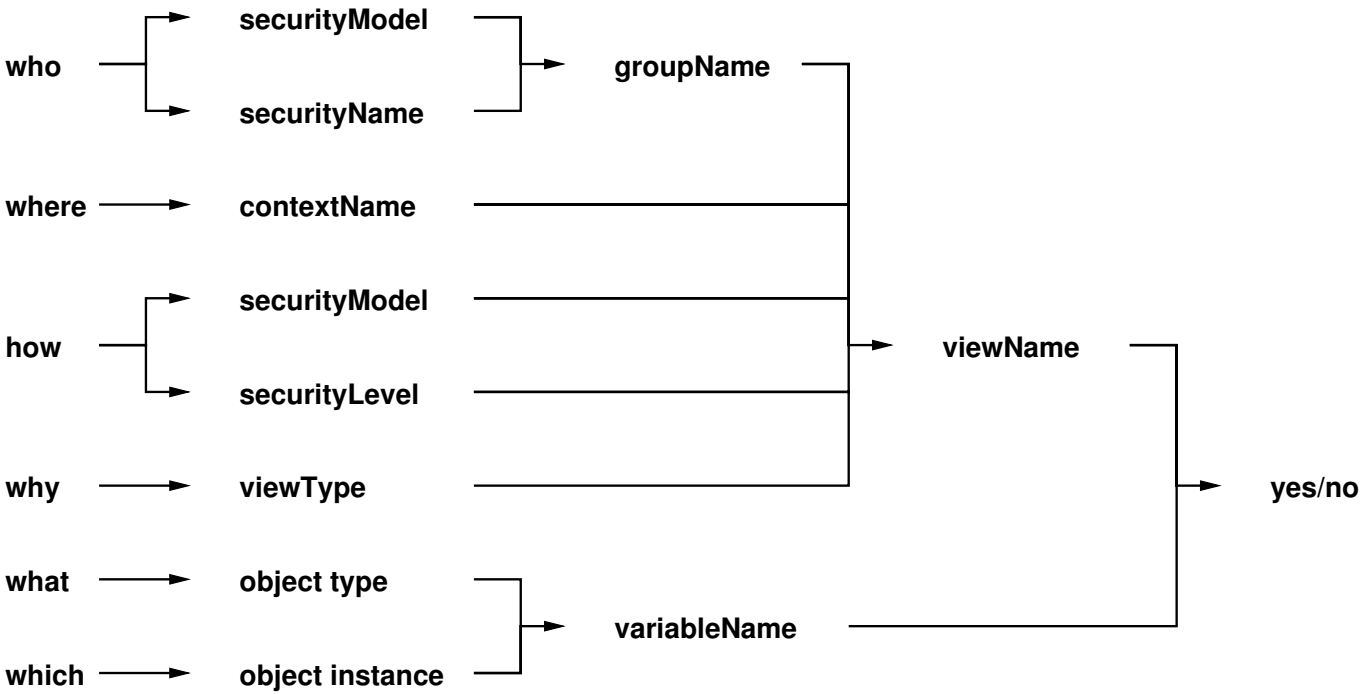
USM MIB (RFC 3414)





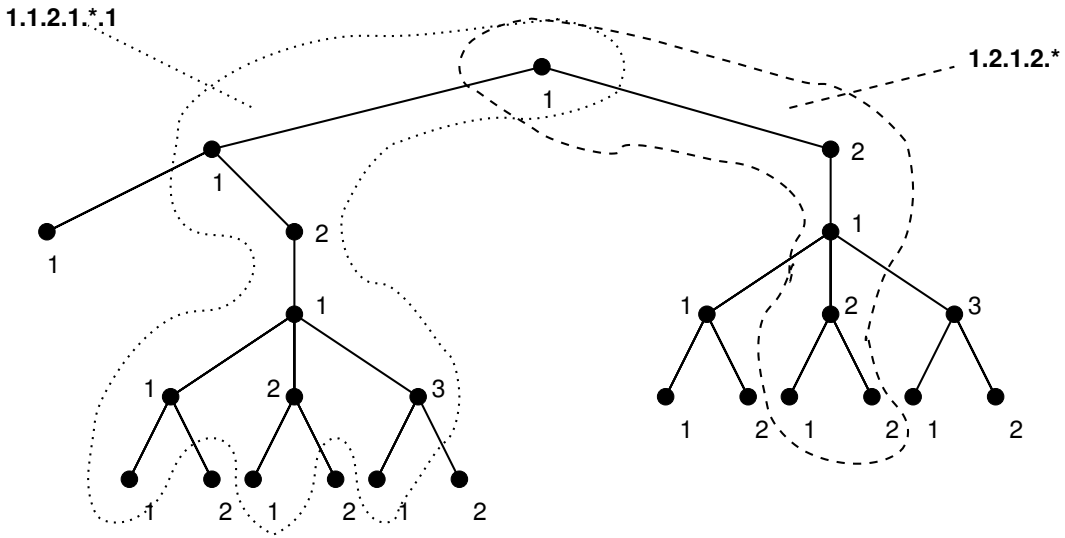
- The `usmUserTable` maps USM user names to `securityNames`.
- New entries may be created by cloning existing entries together with their keys.
- The `usmUserAuthKeyChange` and `usmUserPrivKeyChange` objects may be used by the security administrator to change the user's keys.
- The `usmUserOwnAuthKeyChange` and `usmUserOwnPrivKeyChange` objects may be used by the user to change his keys.

2.3.5 Authorization and Access Control (RFC 3415)



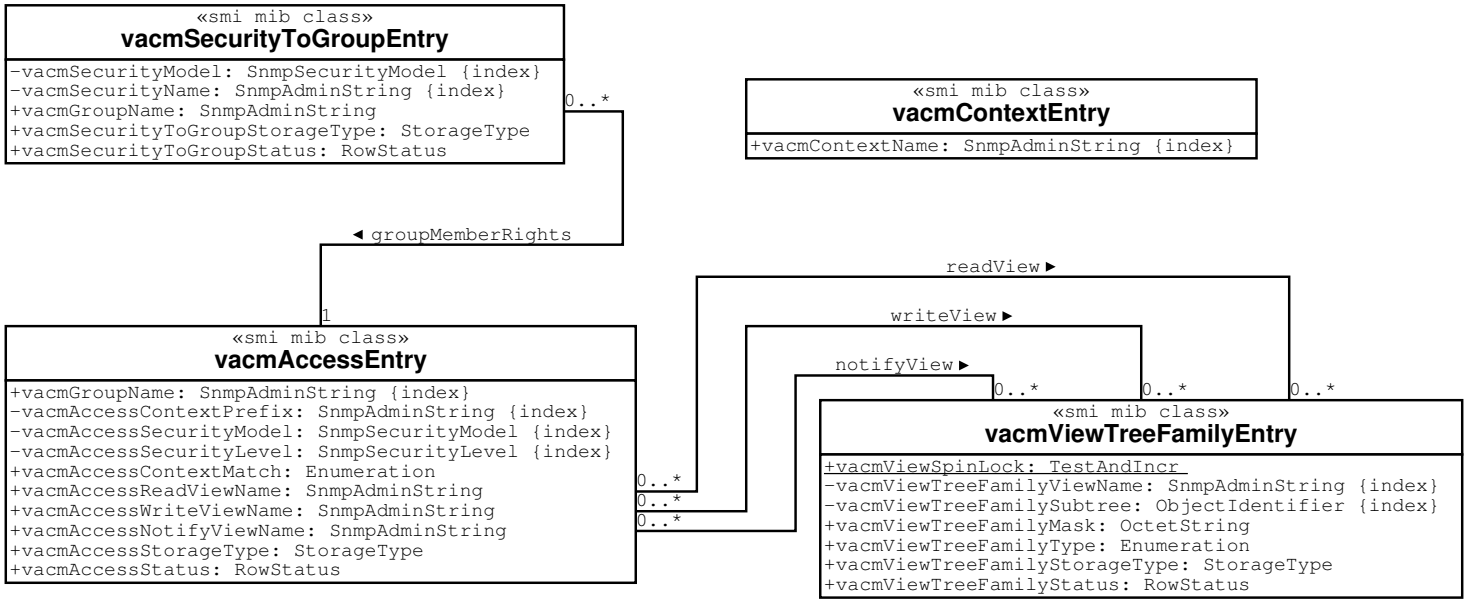
- Three different `securityLevels`: `noAuthNoPriv`, `authNoPriv`, `authPriv`
- A `securityName` is a security model independent name for a principal.

View-based Access Control (RFC 3415)



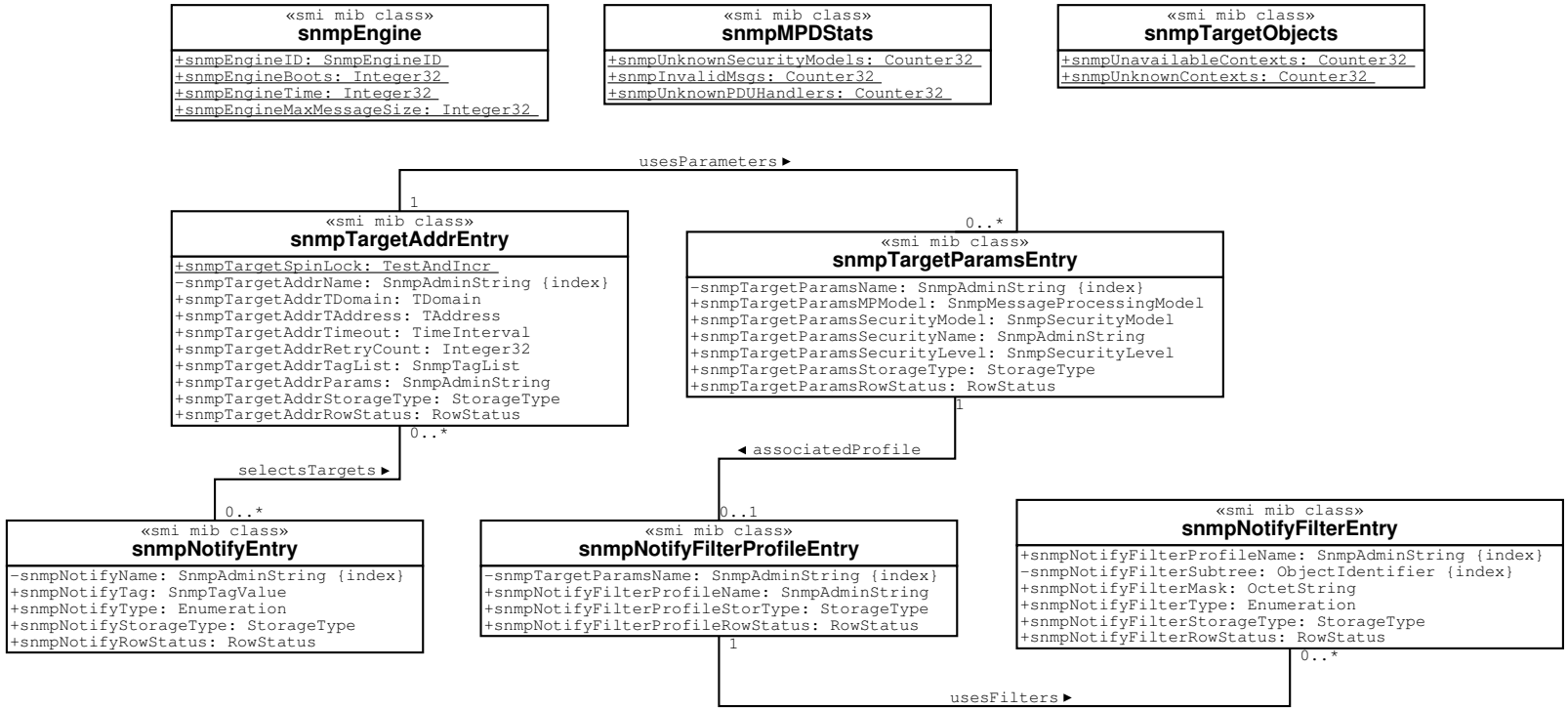
- A *view subtree* is a set of managed object instances with a common OID prefix.
- A *view tree family* is the combination of an OID prefix with a bit mask.
- A bit of the bit mask defines whether an OID prefix component is significant or not (wild-carding).
- A *view* is an ordered set of view tree families.
- Access control rights are defined by a *read view*, *write view* or *notify view*.

View-based Access Control MIB (RFC 3415)



- A security name (with a given security level) can not be a member of multiple groups.
- The `vacmViewTreeFamilyType` can be used to include or exclude a view tree family.
- The context table is kind of degenerated.

2.3.6 Remote Configuration (RFC 3413)



- SNMPv3 defines several MIB modules for remote configuration of SNMP entities.
- The example shows the structure of MIB tables for notification forwarding and notification filtering and general SNMP engine statistics.

2.3.7 Status and Limitations

- Several implementations and products are available, among them:

ACMACE*COMM	IBM
SNMP++v3 Project	InterWorking Labs
BMC Software	MG-SOFT Corporation
Cisco Systems	MultiPort Corporation
ISI/Epilogue	SNMP Research
Gambit Communications	TU Braunschweig
Halcyon	UC Davis
IBM Research	University of Quebec

- Visit the SNMPv3 Web page for up-to-date information.
  <URL:http://www.ibr.cs.tu-bs.de/ietf/snmpv3/>

- Some technology domains (cable modem industry) require SNMPv3 support.

- Deployment happens much slower than originally expected.

Some Known Limitations of SNMPv3

- Missing extensibility for new base data types (e.g. Unsigned64).
- Missing extensibility for new protocol operations (e.g. `GetSubtree`).
- Limited flexibility in VACM grouping rules.
- Asymmetries between notification filtering and VACM filtering.
- Strength of USM security (DES versus AES, key change procedure).
- Initial key assignment problematic (lack of a Diffie-Helman exchange).
- Manual configuration is an error prone and time consuming task. Lack of integration in deployed AAA systems.
- Remote configuration and key management requires nontrivial applications.
- Unnecessary complexity and misleading names in the message format definition.
- Unlimited degrees of freedom in complex write operations on tables are likely to cause interoperability problems.

2.4 Core Information Models and Data Models

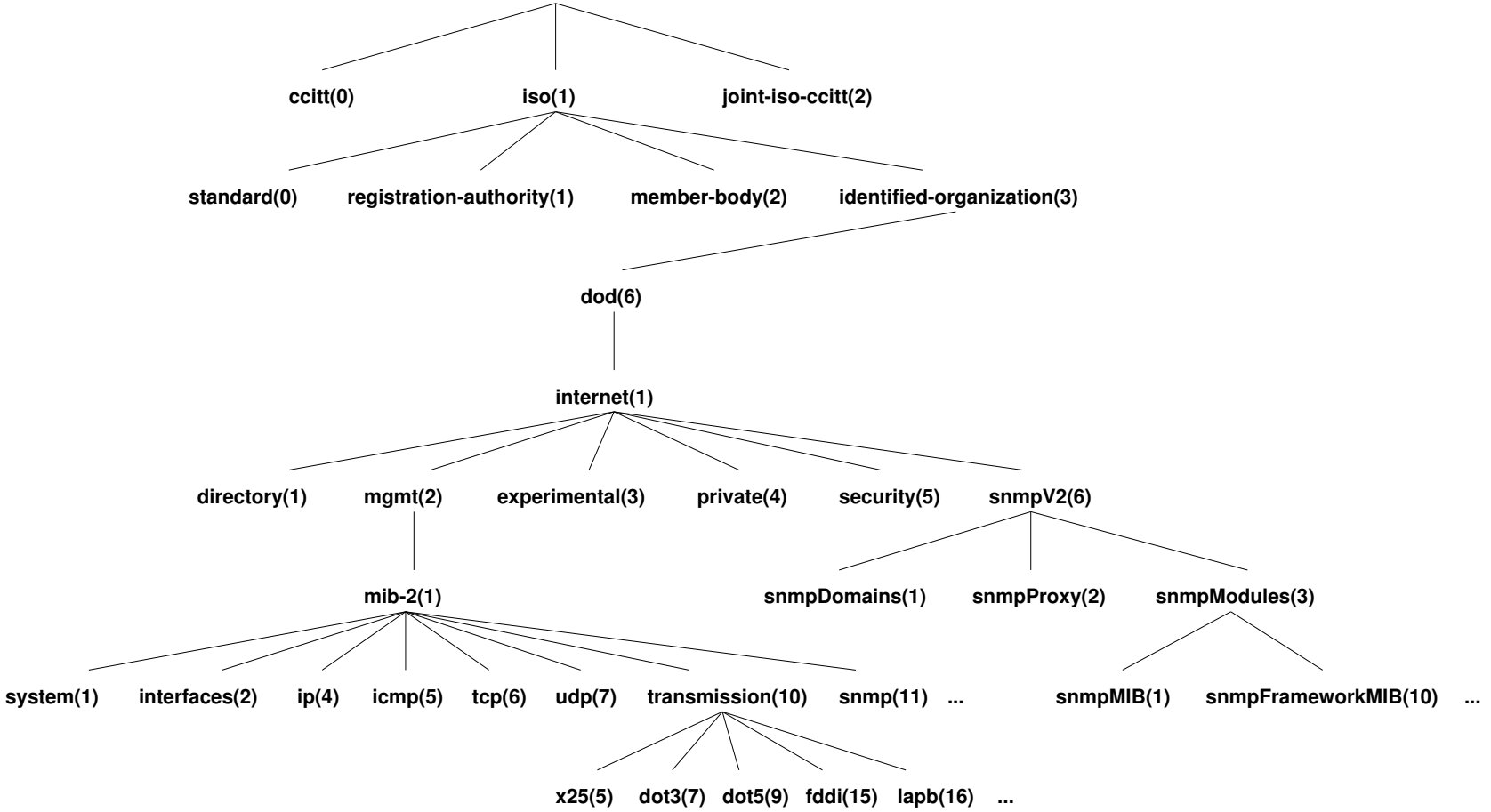
2.4.1 OID Registration Tree

2.4.2 Physical and Logical Network Interfaces

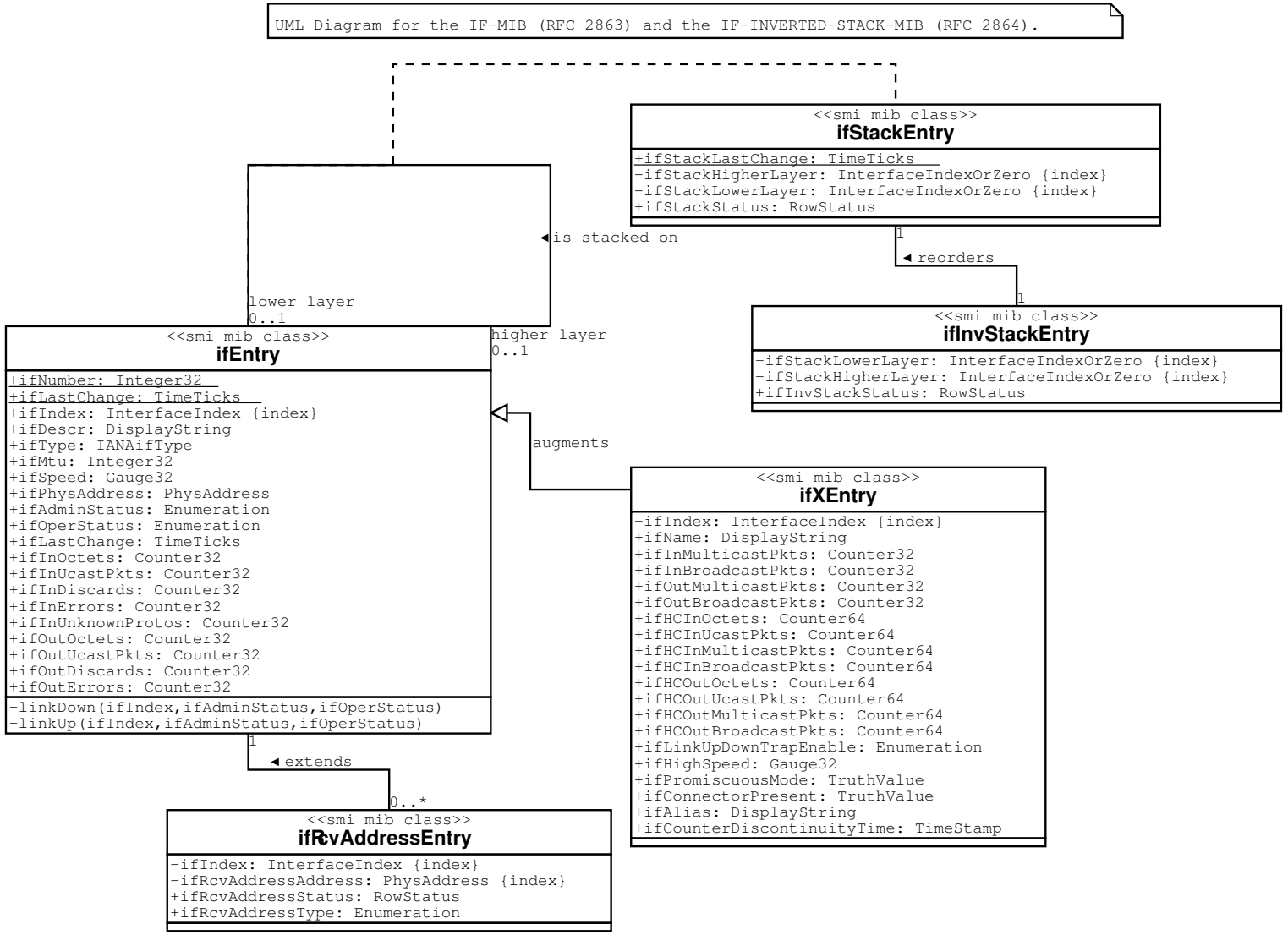
2.4.3 Physical and Logical Entities

2.4.4 Internet Protocol (IPv4)

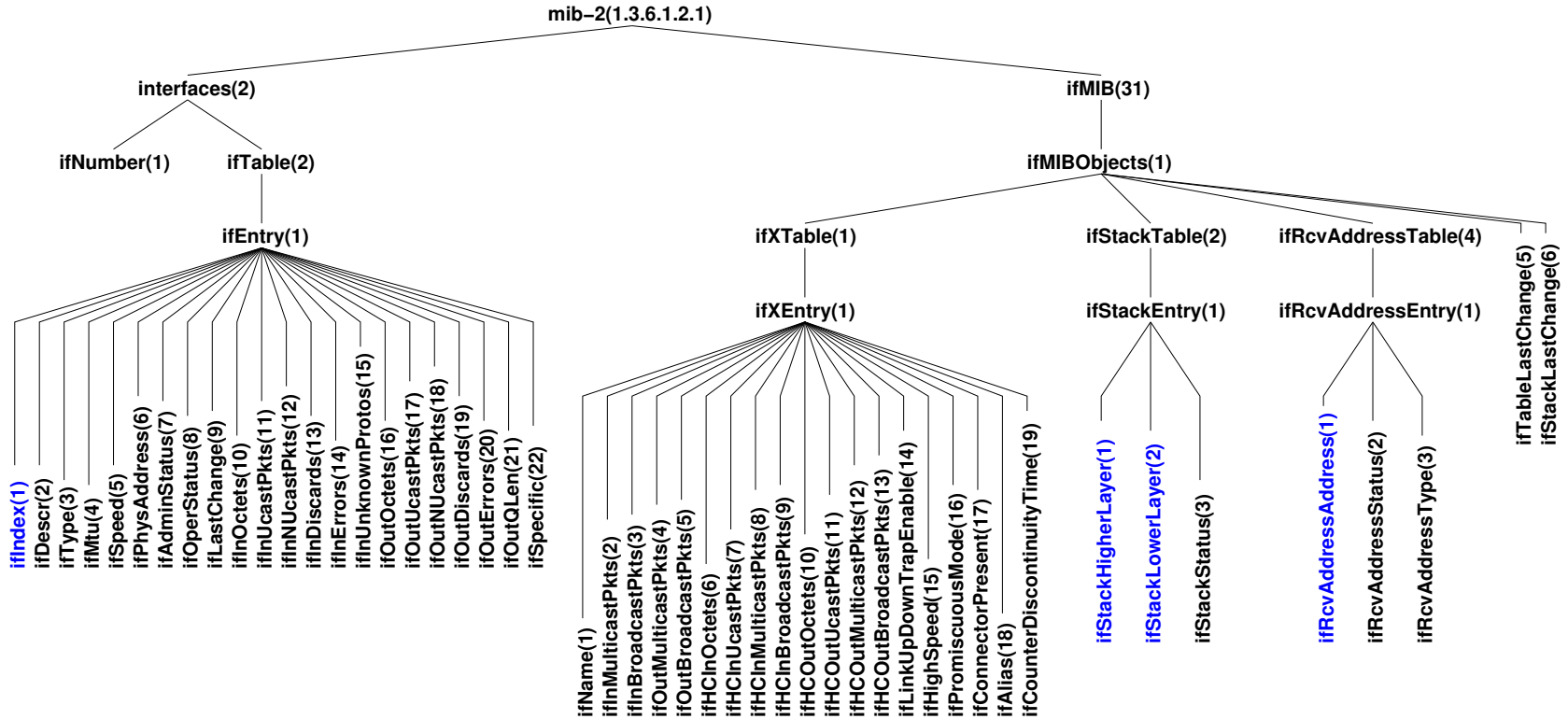
2.4.1 OID Registration Tree



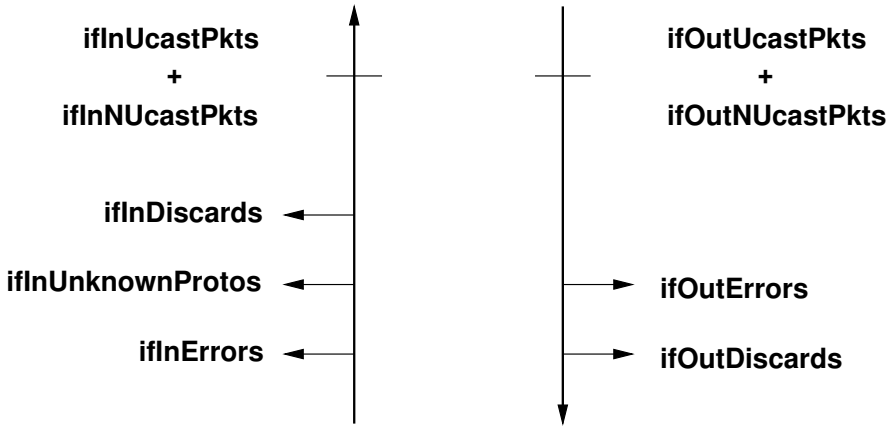
2.4.2 Physical and Logical Network Interfaces



OID Tree Structure of the IF-MIB (RFC 2863)

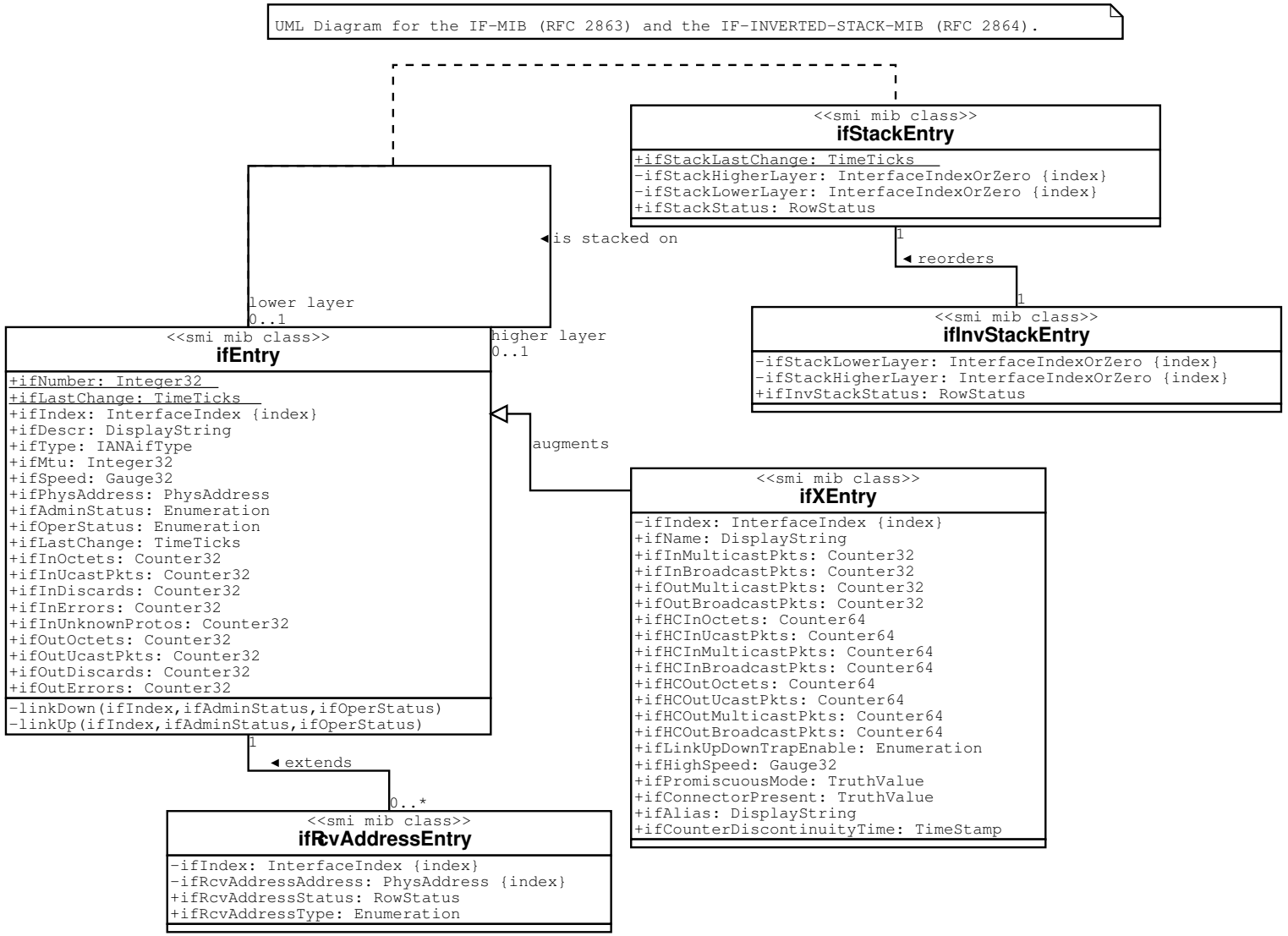


Case Diagram of the ifTable of the IF-MIB

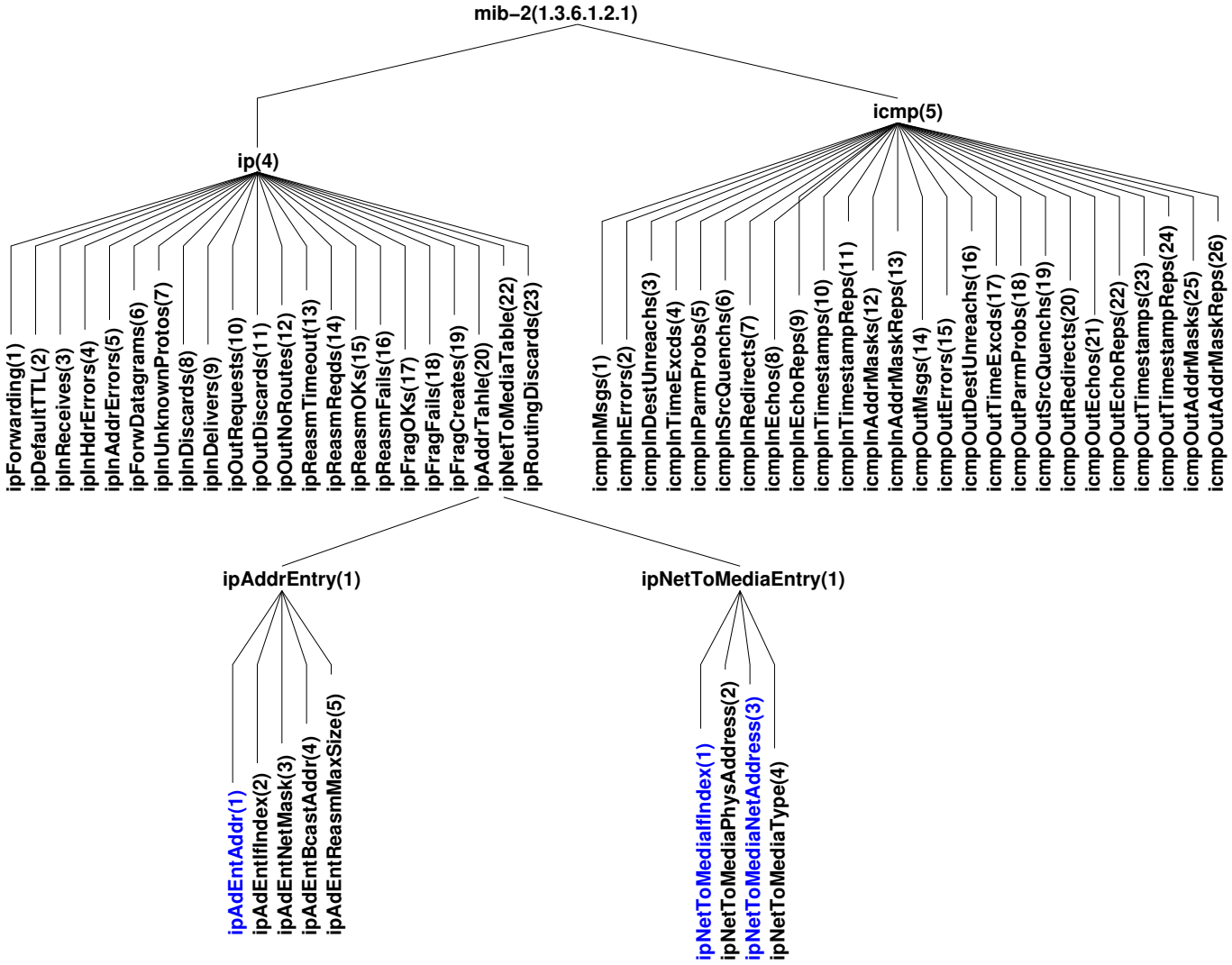


- Case diagrams are used to explain relationships between counters:
  - The total number of packets delivered to the next higher layer is:
    ifInUcastPkts + ifInNUcastPkts
  - The total number of received packets from the interface is:
    (ifInUcastPkts + ifInNUcastPkts) + ifInDiscards + ifInUnknownProtos + ifInErrors
  - The total number of send packets over the interface is:
    (ifOutUcastPkts + ifOutNUcastPkts) - ifOutErrors - ifOutDiscards

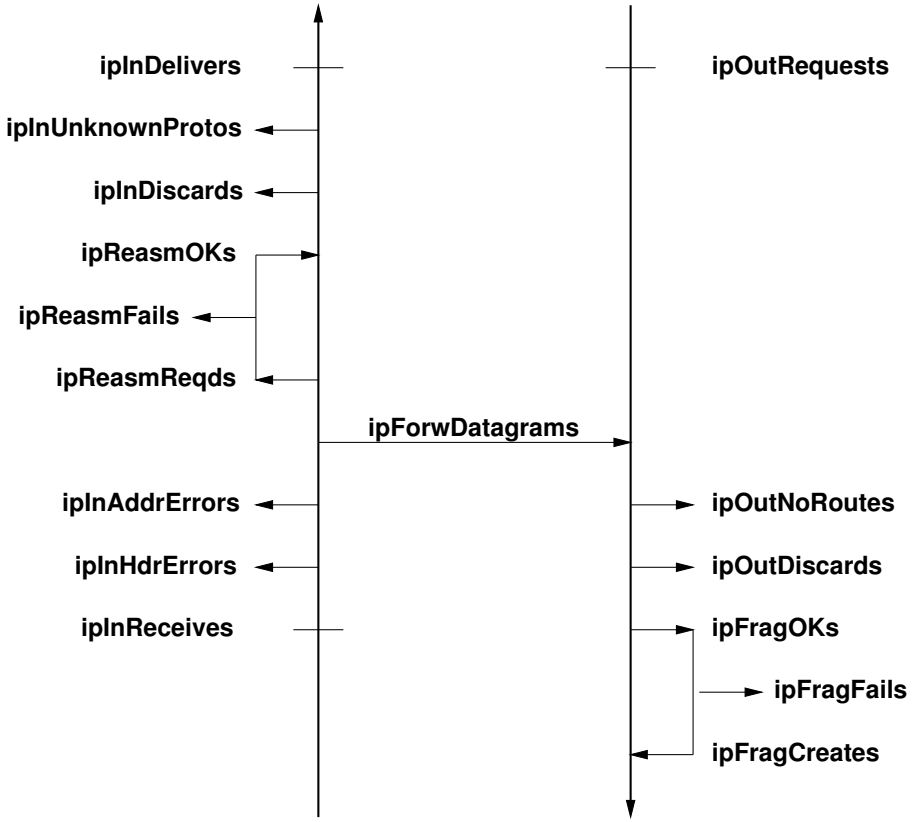
2.4.3 Physical and Logical Entities



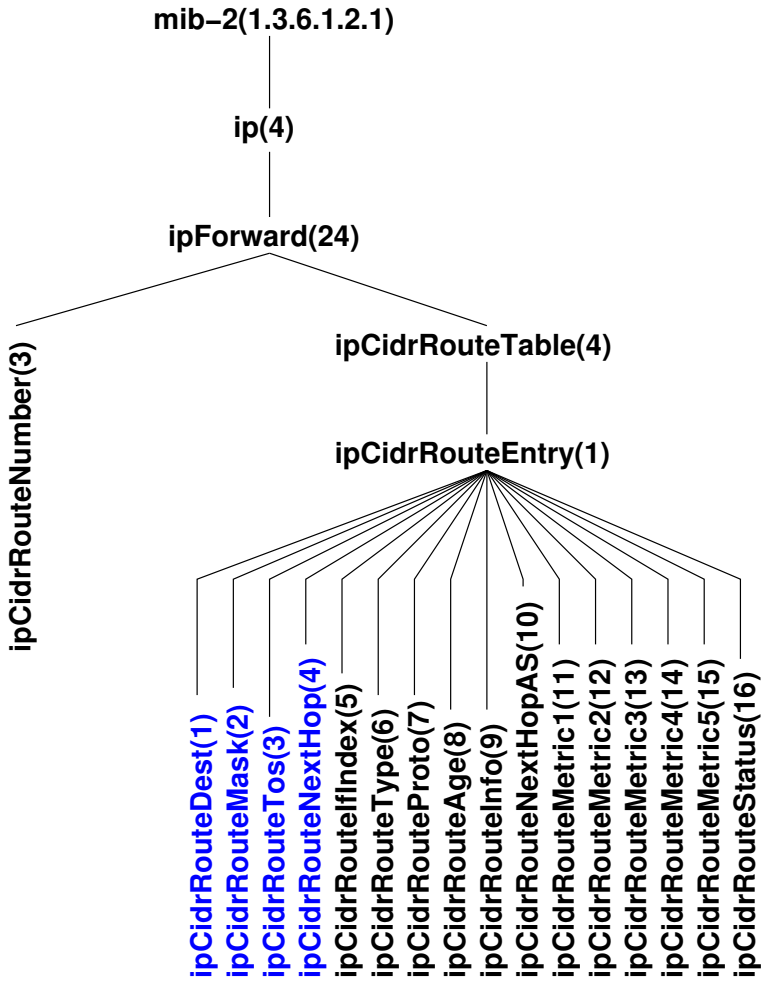
2.4.4 IP/ICMP Version 4 MIB Module (RFC 2011)



Case Diagram of the ip Group



IP Forwarding MIB Module (RFC 2096)



$\Longrightarrow$ These IP layer MIBs will soon be replaced by MIBs that support IPv4 and IPv6.

2.5. Extensible and Programmable Agents

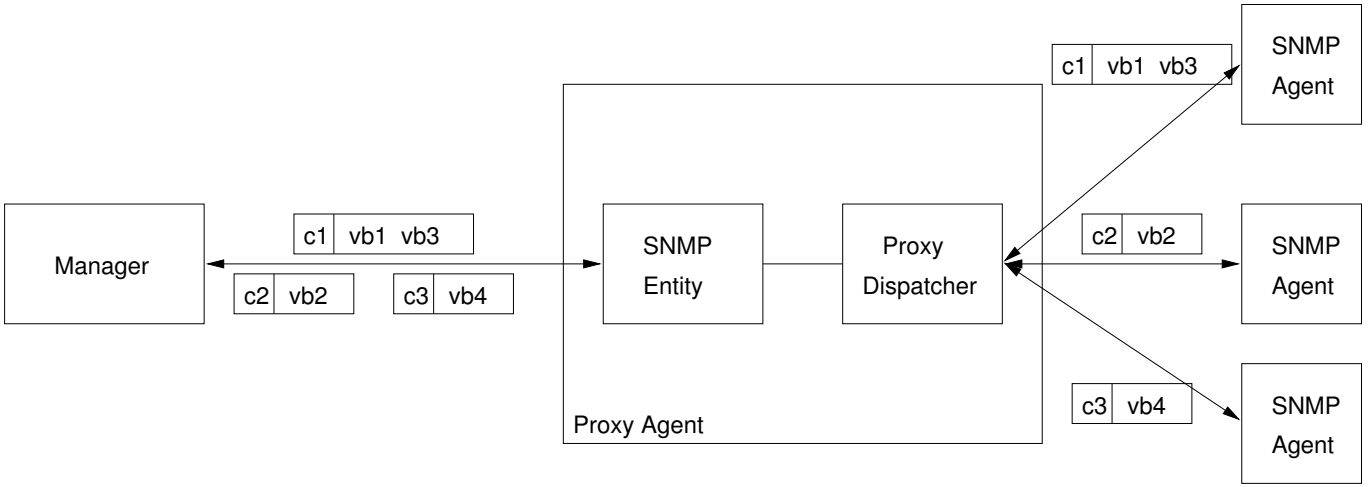
2.5.1 Proxy Agents

2.5.2 Extensible Agents

2.5.3 Agent Extensibility Protocol Version 1 (RFC 2741)

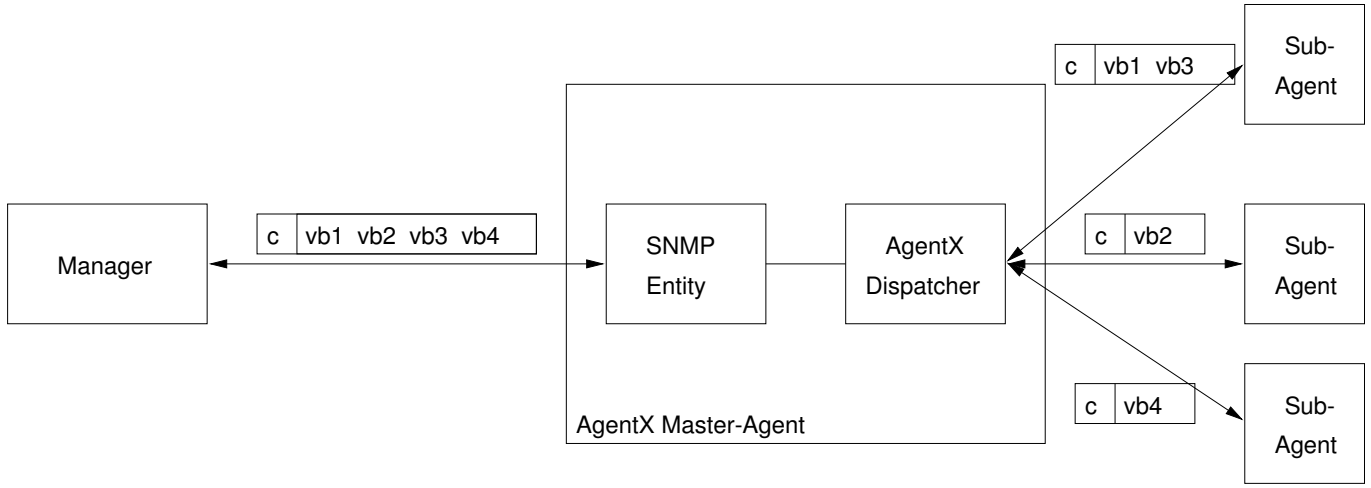
2.5.4 Progammable Agents (RFC 3165)

2.5.1 Proxy Agents



- Proxies provide access to SNMP engines that are not directly accessible (e.g. Firewalls).
- Management applications must address the target agent through the proxy agent (no transparency).
- Proxies can translate between different protocol versions.
- Proxies are fully specified in the SNMPv3 specifications.
- Limited efficiency since all parties must implement/use a full SNMP protocol entity.

2.5.2 Extensible Agents



- Separation of the SNMP protocol engine from the MIB instrumentation.
- MIB module implementations can be added dynamically.
- Management applications have transparent access to "distributed agents."
- An extensibility protocol manages communication between master and sub-agents.
- Requires relatively complex operations on the master agent side in order to realize SNMP lexicographic ordering and access control efficiently.

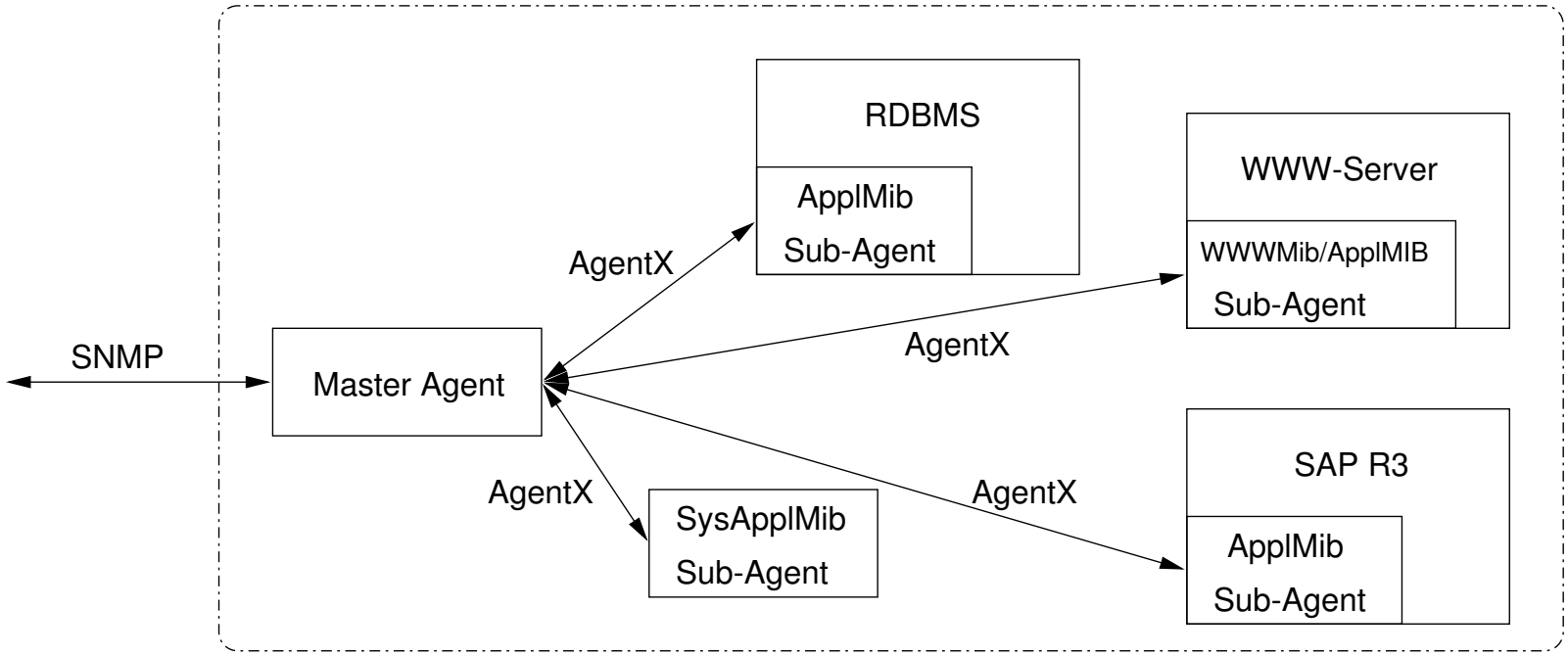
History of Extensible Agent Technology

- SNMP MUX Protocol (SMUX) circa 1991 (RFC 1227)
- SNMP Distributed Protocol Interface (DPI) circa 1991–1994 (RFC 1592)
- Digital's extensible SNMP agent (eSNMP) circa 1995
- SNMP Research's Emanate
- Several other proprietary protocols, often derived from SMUX
- Agent Extensibility Protocol (AgentX) circa 1998 (RFC 2257)

2.5.3 Agent Extensibility Protocol Version 1 (RFC 2741)

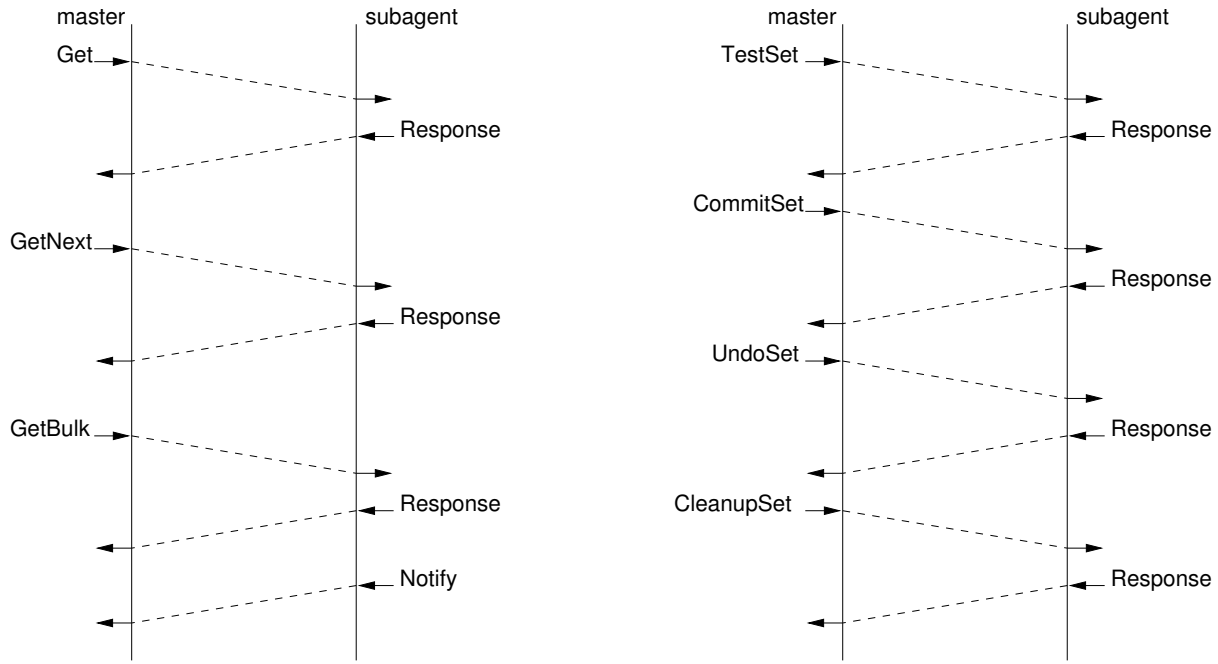
- Open standard for the implementation of extensible sub-agents.
- Based on experiences with several similar non-standard protocols.
- Core technology for modular networking devices.
- Required for portable system and application management agents.
- The AgentX master agent is MIB ignorant and SNMP omniscient.
- The AgentX sub-agent is SNMP ignorant and MIB omniscient.
- AgentX supports sub-agent integration through index allocation.
- Efficient AgentX message formats and encodings.

Example: Application Management with AgentX



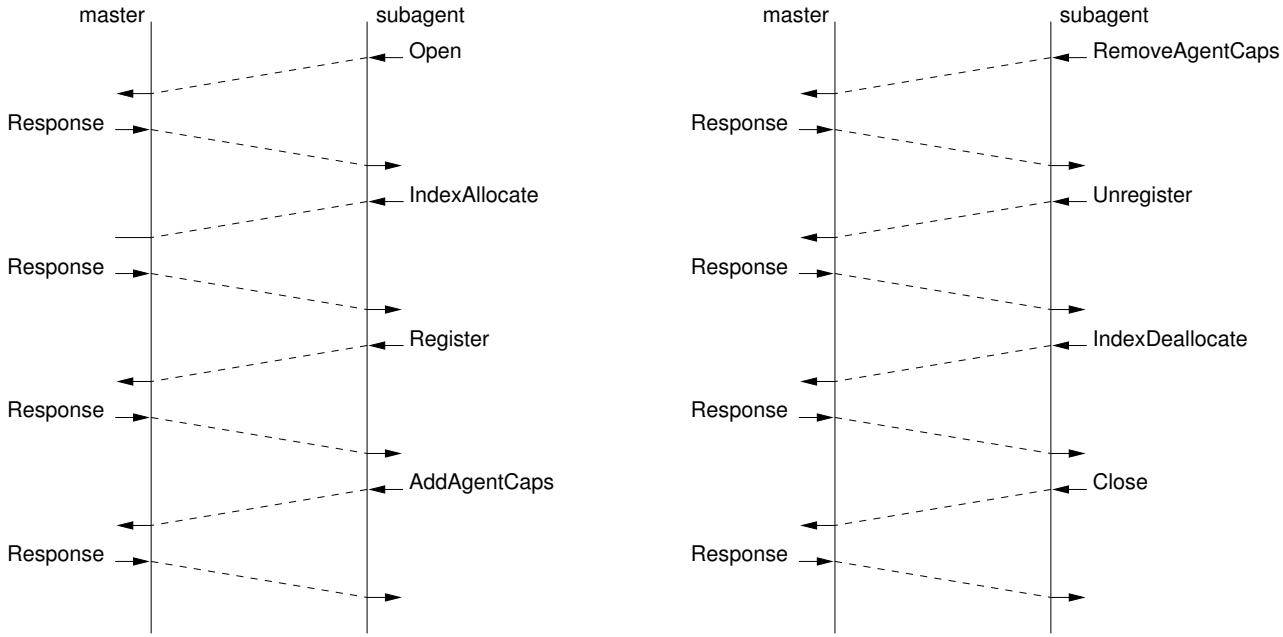
- Application management MIBs require instrumentation in the applications.
- AgentX provides the infrastructure for implementing application management MIBs.

AgentX Protocol Operations (RFC 2257)



- SNMP Get, GetNext, GetBulk PDUs are mapped to similar AgentX PDUs.
- AgentX employs test-commit-undo-cleanup phases to achieve "as if simultaneous" semantics for SNMP write operations.
- The AgentX Notify operation may be mapped to SNMP Trap or Inform messages.

Administrative AgentX Protocol Operations (RFC 2257)



- Administrative protocol operations are used to open and close AgentX sessions, allocate and deallocate index values, register and unregister MIB regions exported by sub-agents, and add and remove agent capabilities (`sysOrTable`).

Transport Mappings and Encodings

- AgentX defines the following transport mappings:
  - AgentX over TCP (well known port number 705)
  - UNIX domain sockets (well known socket endpoint /var/agentx/master)
- Additional transport mappings (e.g. via shared memory) can be added.
- AgentX does not use BER encodings for efficiency reasons.
- AgentX encodings provide natural alignment with respect to the start of a PDU.
- AgentX sub-agents may use the native byte ordering.
- AgentX encodes well-known object identifier prefixes efficiently.
- OCTET STRING value are padded to align the encoded value to 4 byte boundaries.
- AgentX introduces search ranges which consist of two OID values.

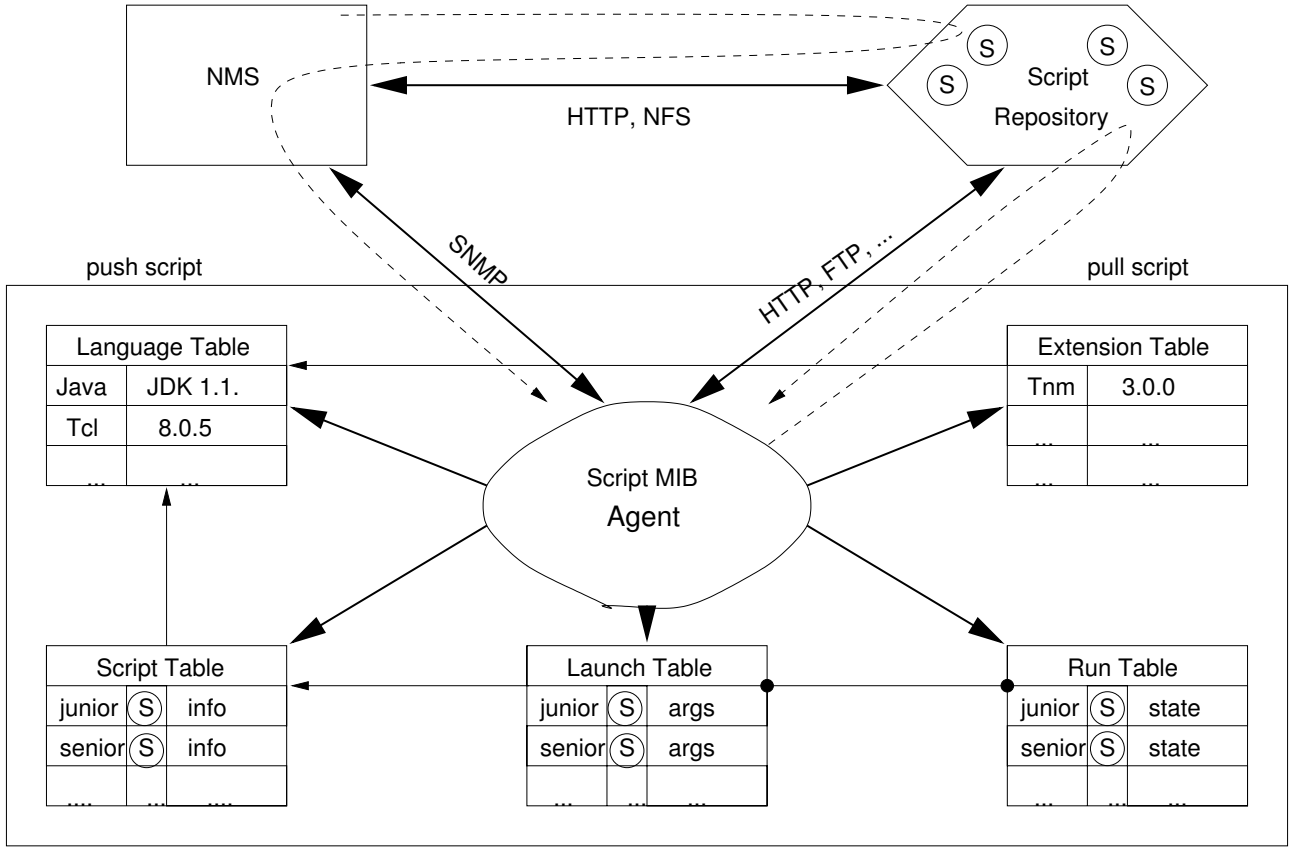
OID Registration and Scoping

- Sub-agents may register single instances:
  (e.g. 1.3.6.1.2.1.25.1.2.0 = HOST-RESOURCES-MIB.hrSystemDate.0)
- Sub-agents may register OID regions:
  (e.g. 1.3.6.1.2.1.2.2.1.[1-22].7 = IF-MIB.ifIndex.7 – IF-MIB.ifSpecific.7)
- Only a single sub-agent can be "authoritative" for a particular OID region.
- Priority values are used to identify the authoritative sub-agent if regions overlap.
- Search ranges may be used during GetNext/GetBulk processing (scopes).
- Retrieving the next instance from an "authoritative" sub-agent may result in a loop over MIB subtrees.

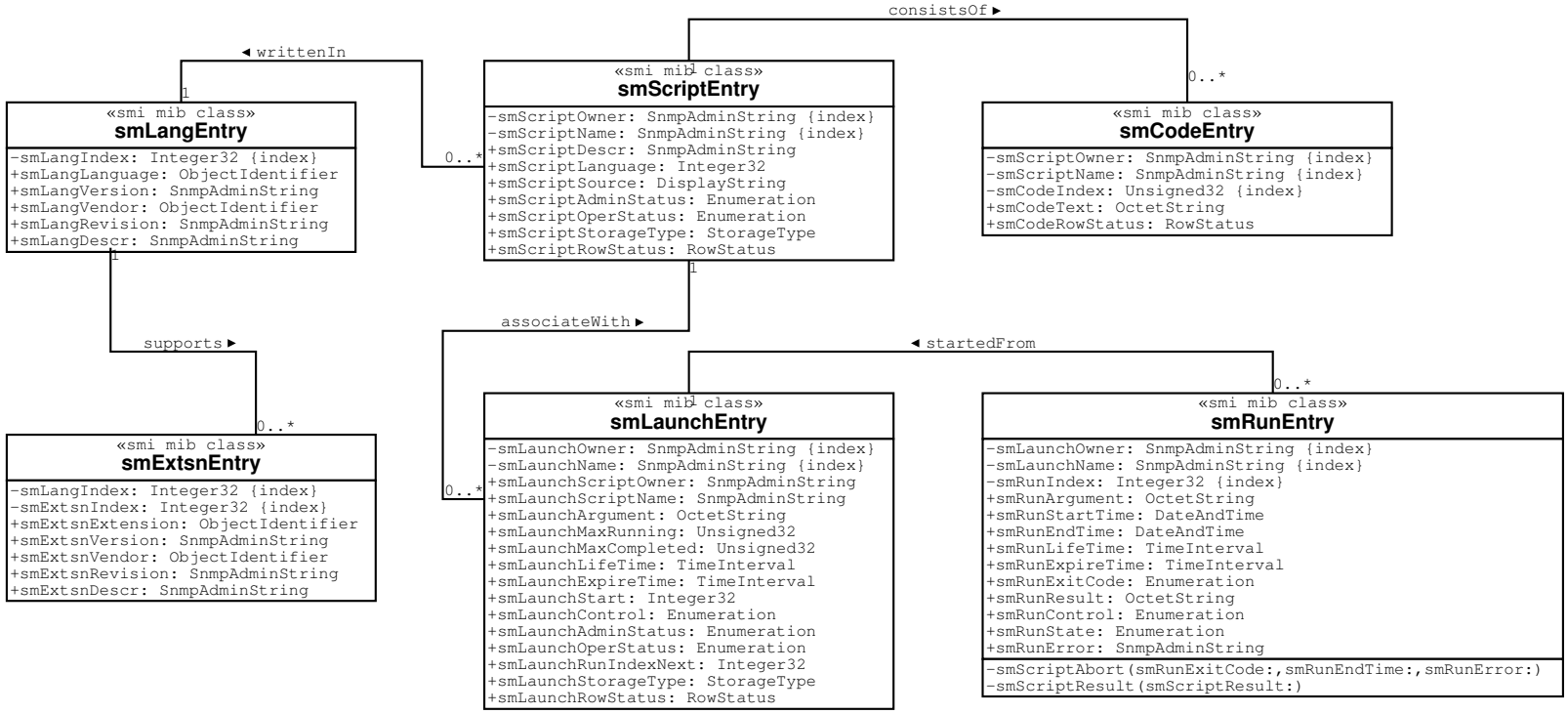
Index Allocation

- A shared MIB table contains rows that are implemented by different sub-agents.
- Index values may be allocated to support sub-agents implementing shared tables.
- A sub-agent may request the allocation of:
  - a specific index value
  - an index value that is not currently allocated
  - an index value that has never been allocated
- The master agent maintains a database of index objects (OIDs), and, for each index, the values that have been allocated for it.
- Example:
  - A sub-agent wants to export a row in the IF-MIB.ifTable.
  - It requests an allocation of a value for ifIndex that has never been allocated.
  - The master agent returns the value 7.
  - It now attempts to register row 7 of the IF-MIB.ifTable (1.3.6.1.2.1.2.2.1.[1-22].7).

2.5.4 Progammable Agents (RFC 3165)

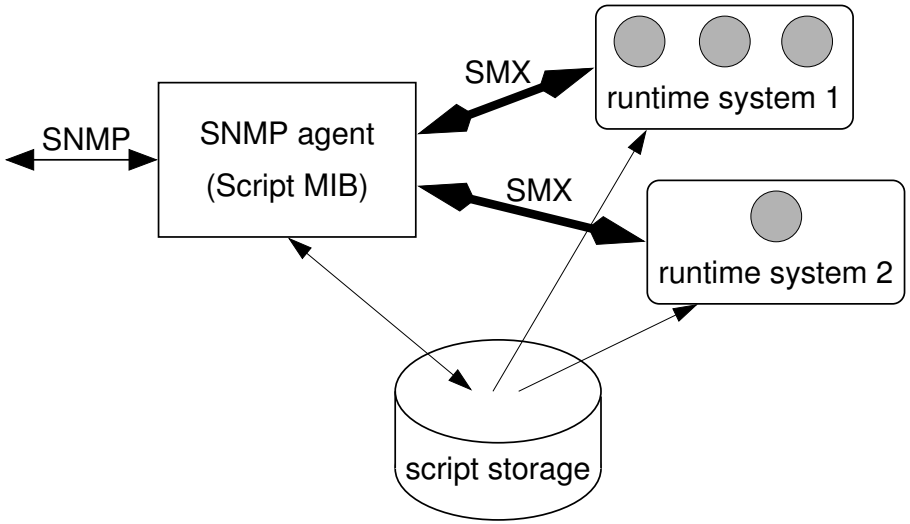


Structure of the Script MIB (RFC 3165)



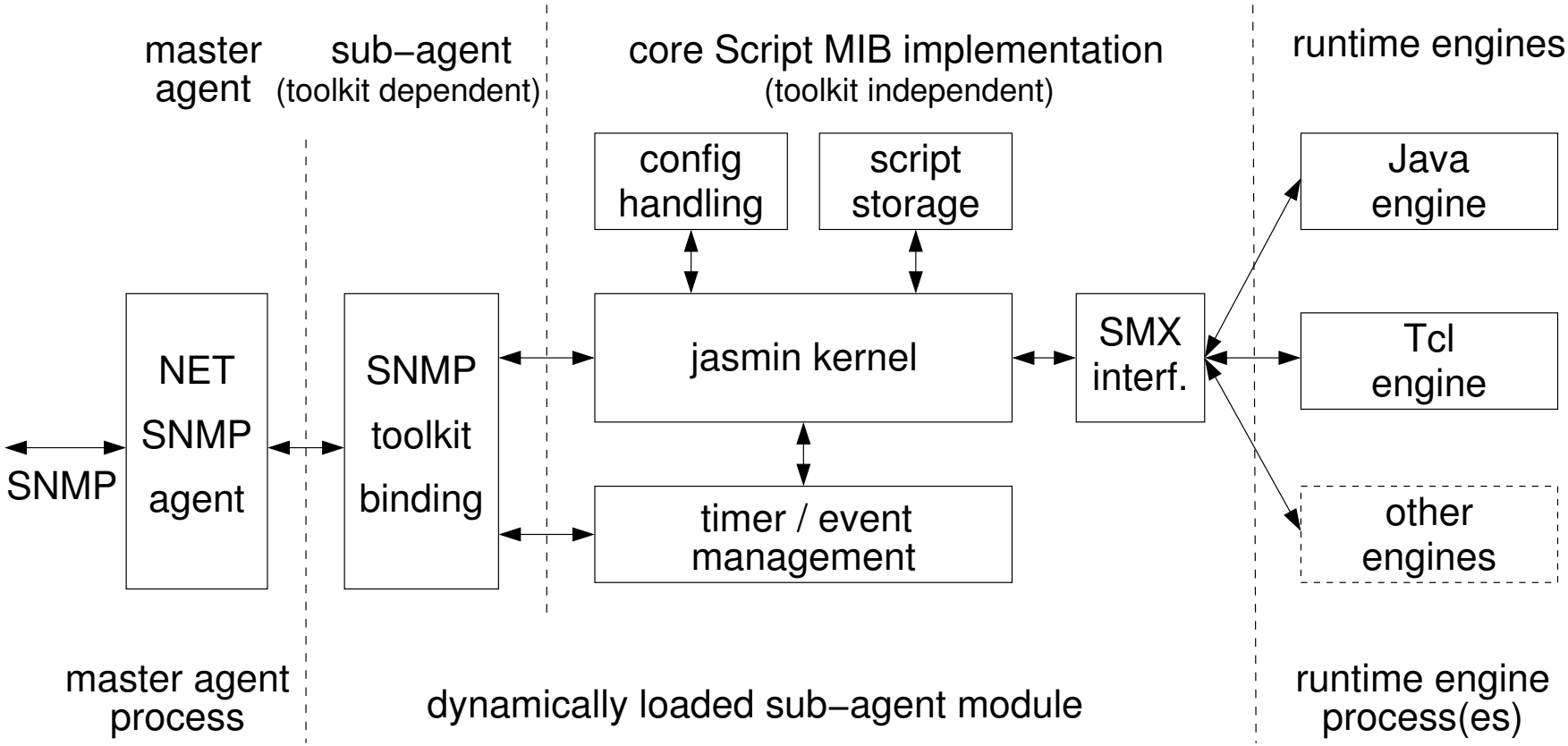
- The `smCodeTable` is optional and not required for conformance.
- The `smLangTable` and `smExtsnTable` are read-only.

Script MIB Extensibility Protocol (RFC 3179)

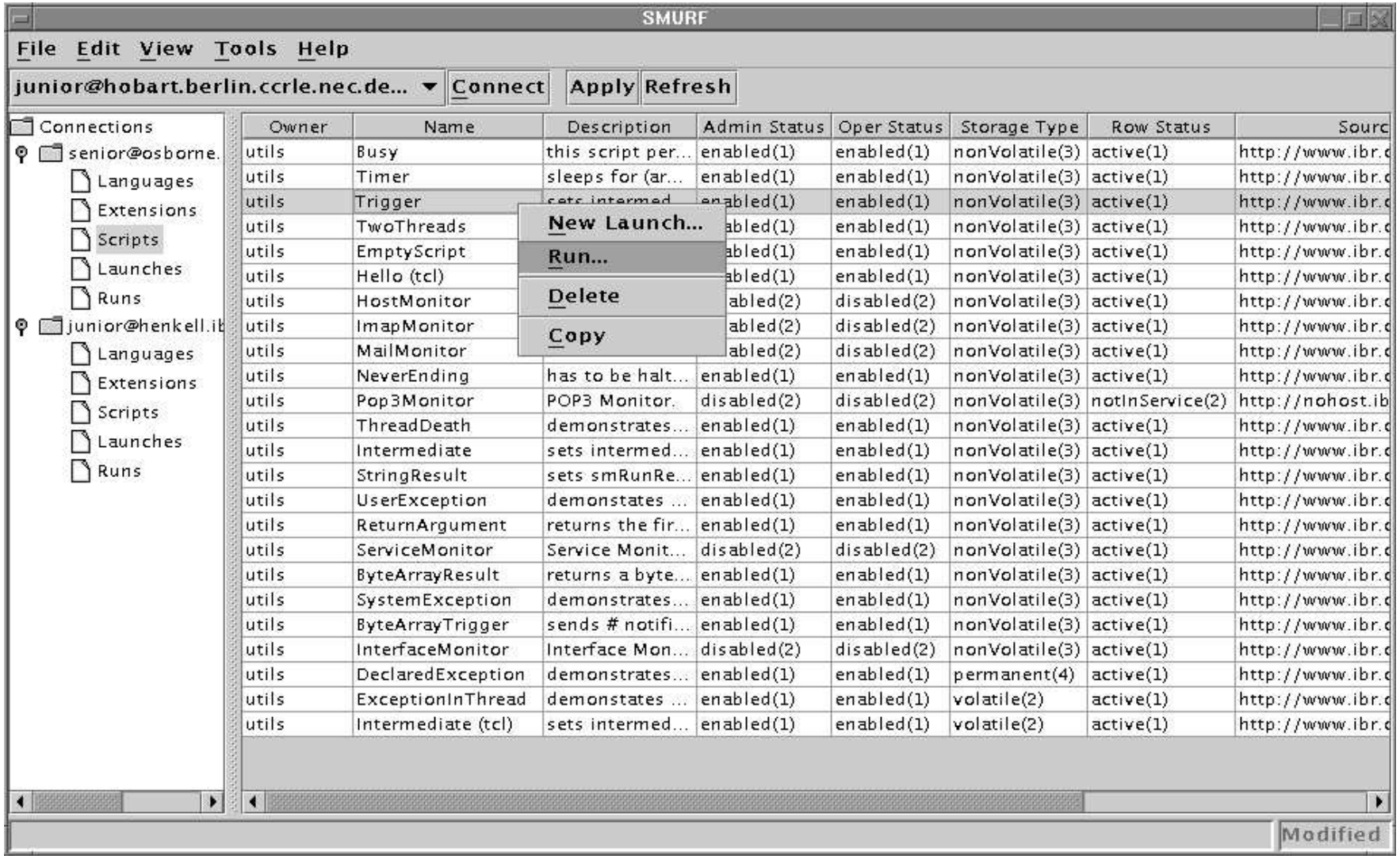


- Separates language specific runtime systems from the runtime system independent Script MIB implementation.
- Supports multiple languages and runtimes with different security profiles.
- Simple protocol running over a local TCP connection.
- Connection establishment procedure verifies a security cookie.
- Uses the local file system to pass executable code to runtime systems.

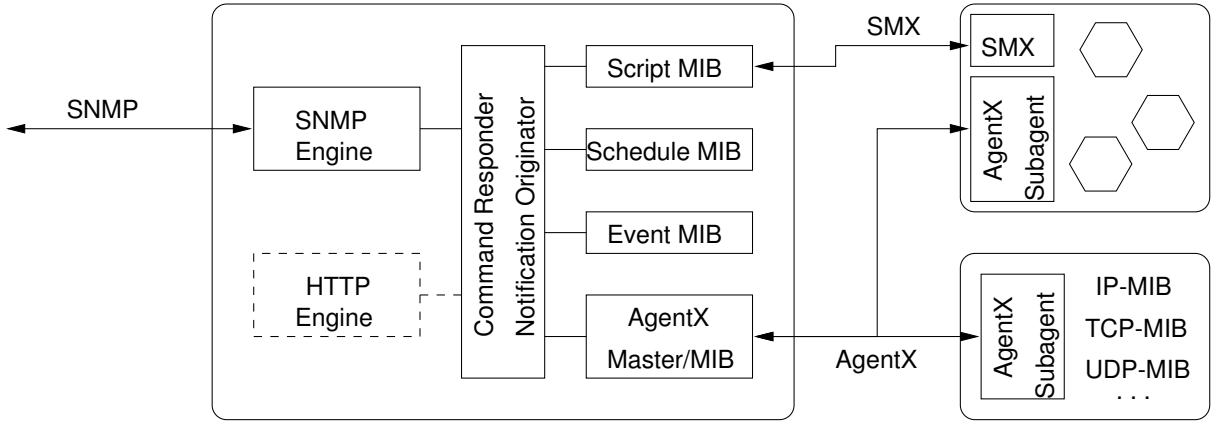
Jasmin Reference Implementation



SMURF – Java Management Application



Secure, Extensible and Programmable SNMP Agents



- SNMPv3 provides message security and authorization services.
- AgentX provides the services for dynamic agent extensions.
- Distributed management MIBs realize the control infrastructure.

$\Rightarrow$ No documented "manager extensibility" protocol yet for allowing "MIBlets" to access other SNMP capable devices via the existing SNMP engine.

$\Rightarrow$ Jasmin prototype implementation being worked on at the Technical University of Braunschweig and NEC C&C Research in Berlin.

3. Common Open Policy Service (COPS)

3.1 Outsourcing versus Provisioning

3.2 COPS Protocol Overview

3.3 COPS for RSVP

3.4 COPS for Provisioning

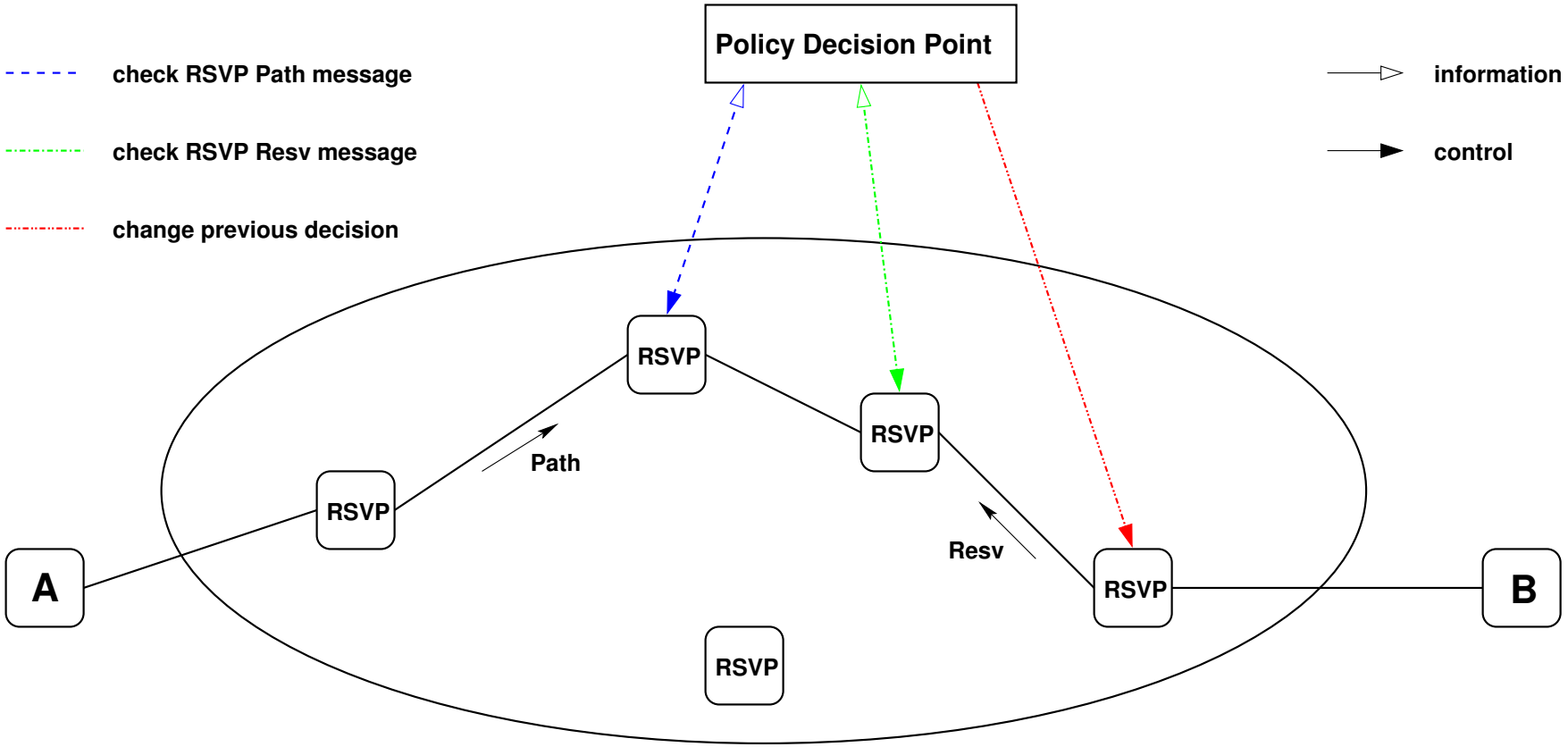
3.5 SPPI versus SMIv2

3.6 COPS-PR versus SNMP

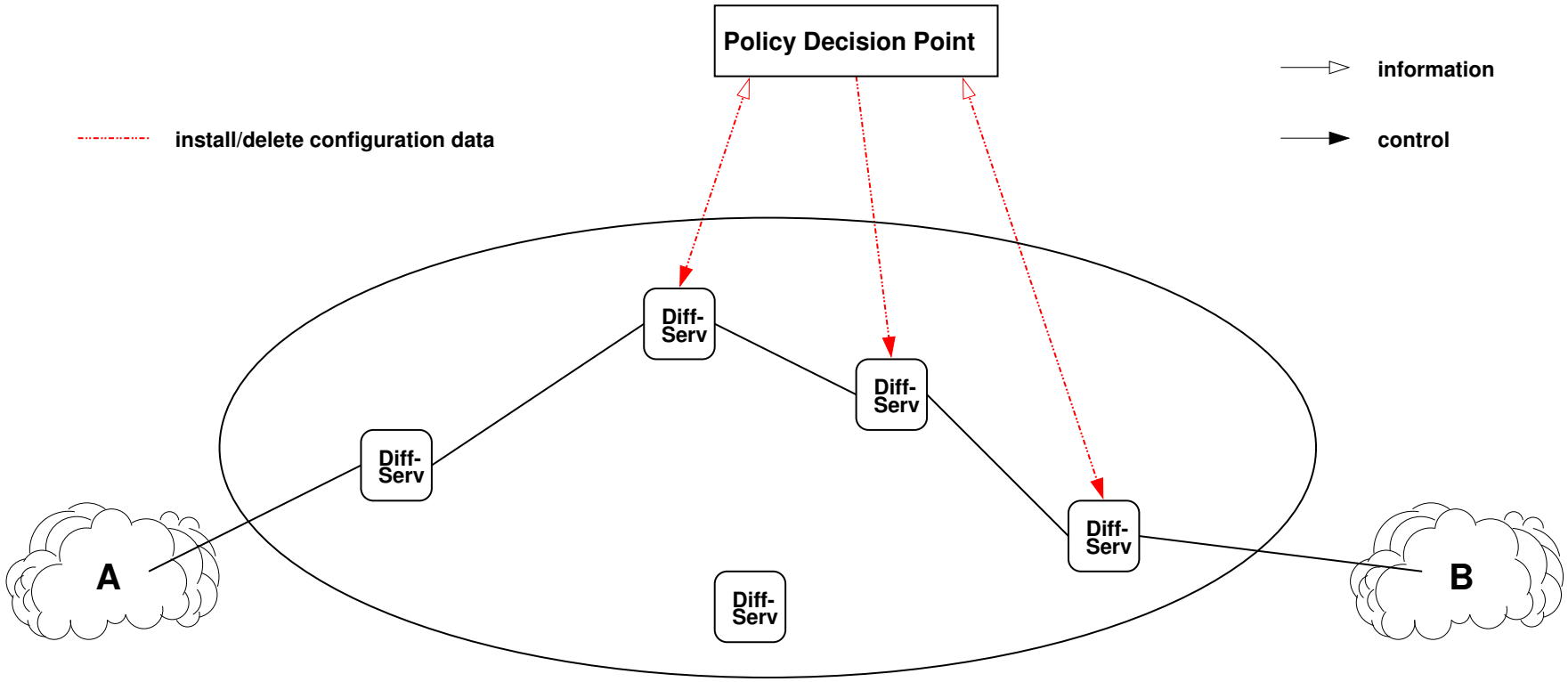
3.1 Outsourcing versus Provisioning

- Outsourcing Model:
  - Decisions are made event-driven during the signaling phase.
  - Additional asynchronous decisions from the policy-based management system.
  - Only applicable where scalability is not a big problem.
- Provisioning Model:
  - Provisioning of all necessary configuration information to enforce policies locally.
  - Provisioning information is defined in a Policy Information Base (PIB).
  - PIBs are defined using the Structure of Policy Provisioning Information (SPPI).
  - No real-time policy interactions and thus highly scalable.

Outsourcing Model (RSVP)



Provisioning Model (DiffServ)



3.2 COPS Protocol Overview (RFC 2748)

- COPS is a client/server protocol between a policy decision point (PDP) and policy enforcement points (PEPs).
- COPS messages are transmitted over a persistent TCP connections (port 3288).
- COPS messages contain sequences of COPS objects.
- Extensibility through self-identifying COPS objects.
- PDPs and PEPs can share state as long as the underlying TCP connection exists.
- The PEP is responsible to establish the connection to the PDP.
- Optional message level security for authentication, replay protection, and message integrity through integrity objects.
- Key management is not considered by the COPS specifications.
- IPsec or TLS (SSL) may be used to further secure (encrypt) the communication channel between PEPs and PDPs.

COPS Operation Types (RFC 2748)

Operation	Description	Direction
REQ	Request	PEP → PDP
DEC	Decision	PDP → PEP
RPT	Report State	PEP → PDP
DRQ	Delete Request State	PEP → PDP
SSQ	Synchronize State Request	PDP → PEP
SSC	Synchronize State Complete	PEP → PDP
OPN	Client-Open	PEP → PDP
CAT	Client-Accept	PDP → PEP
CC	Client-Close	PEP → PDP, PDP → PEP
KA	Keep-Alive	PEP → PDP, PDP → PEP

3.3 COPS for RSVP (RFC 2749)

- All objects received in an RSVP message are encapsulated inside the COPS Client Specific Information Object (ClientSI) send from the PEP to the PDP.

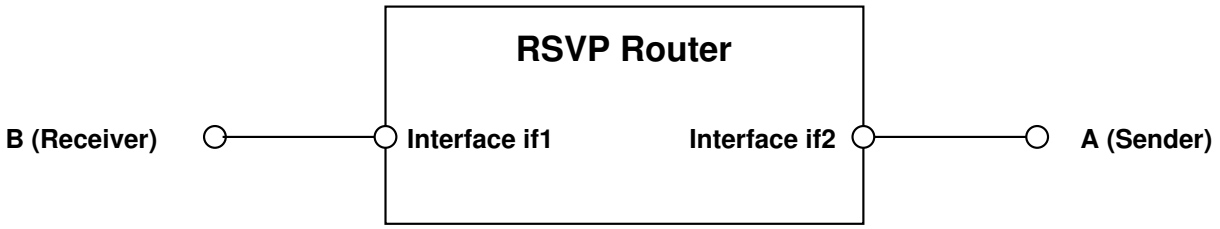
- The PEP and PDP share RSVP state.

- Install decision command:
  - Accept/Allow/Admit an RSVP message or local resource allocation.

- Remove decision command:
  - Deny/Reject/Remove an RSVP message or local resource allocation.

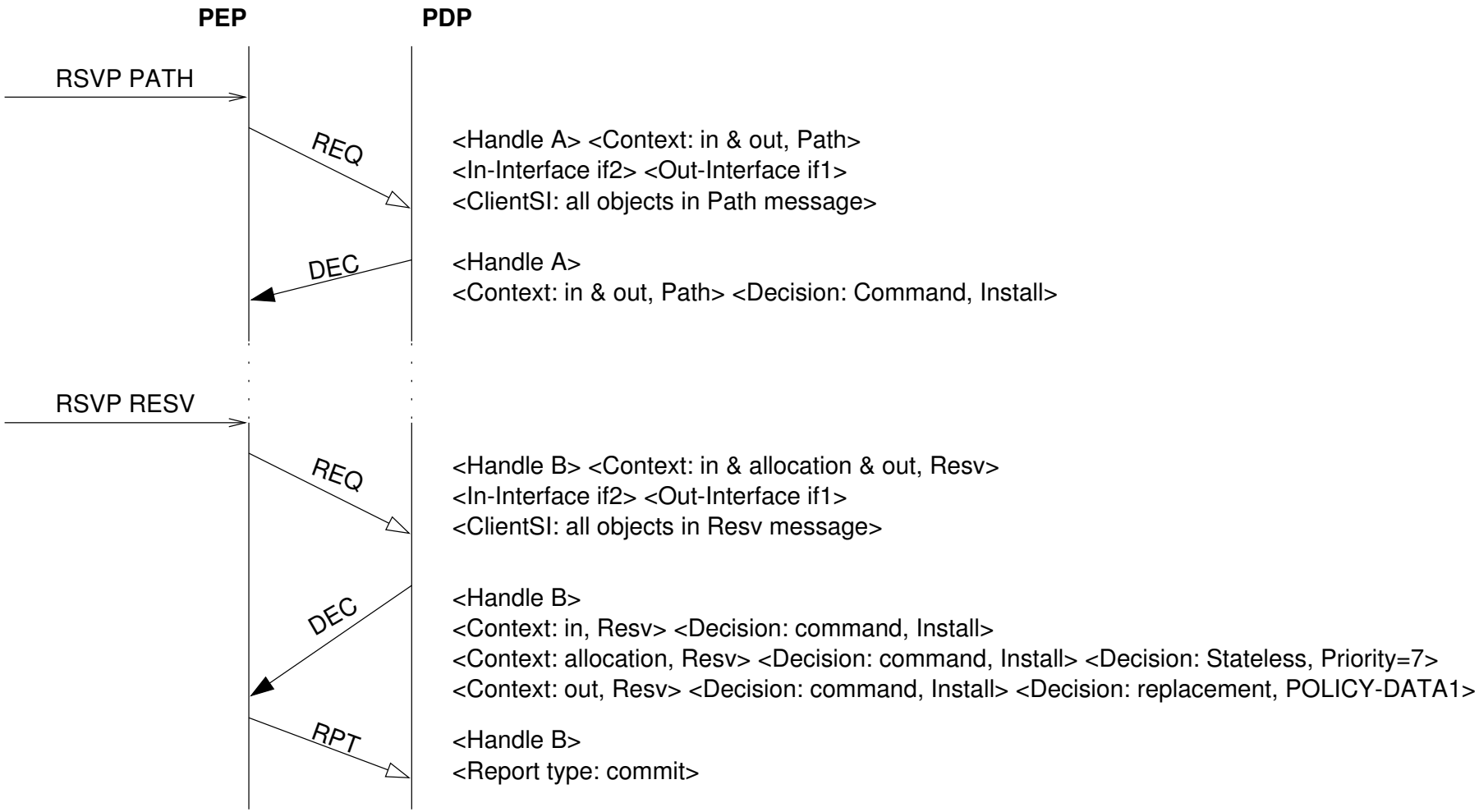
- PEP may cache decisions in order to use them for a given time interval while the connection between the PEP and its PDP is lost.

COPS-RSVP Protocol Example

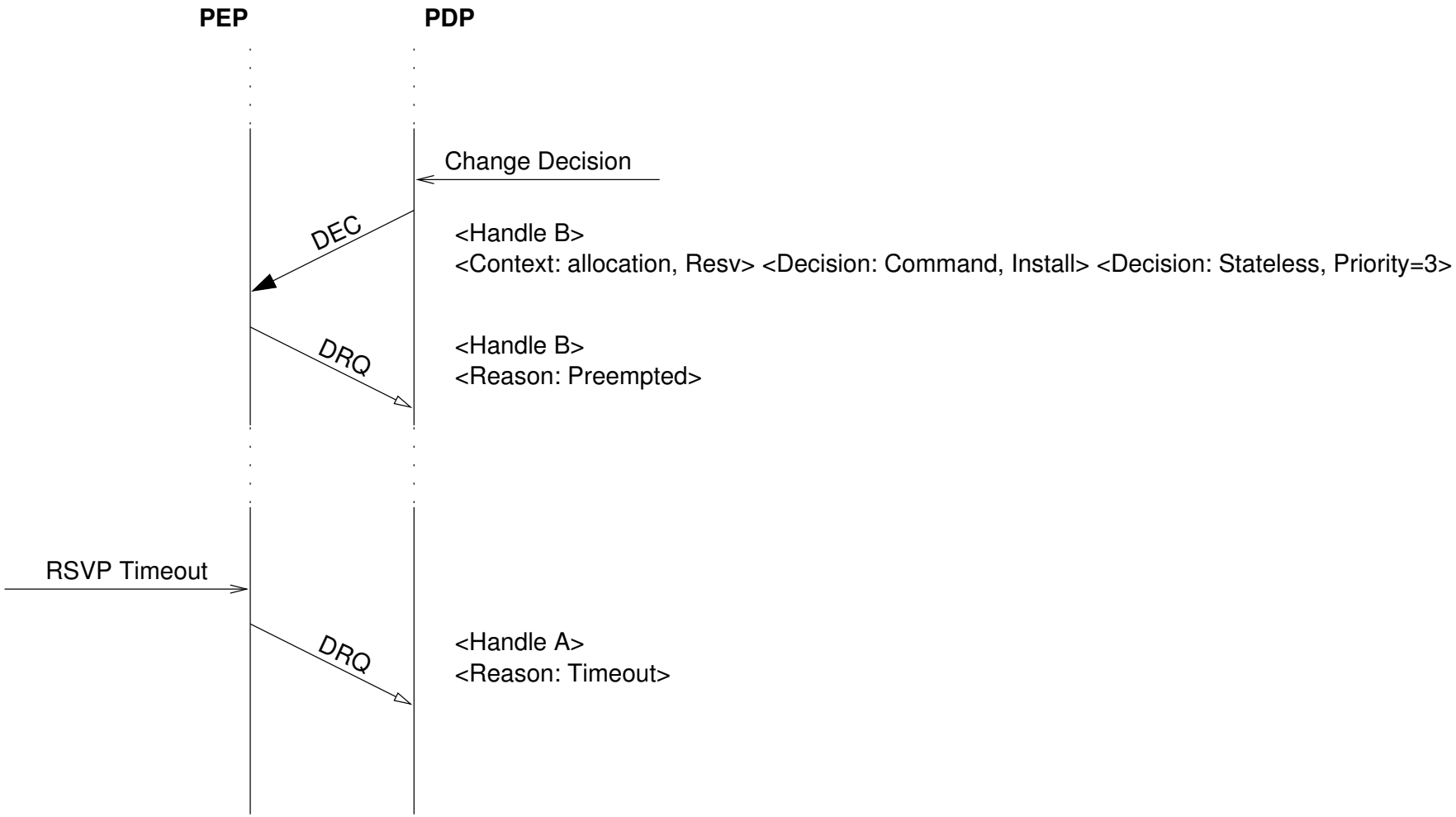


- The PEP router has two network interfaces (if1, if2).
- Sender A sends to receiver B.
- COPS RSVP is used to control the unicast RSVP flow between A and B.

COPS-RSVP Protocol Example



COPS-RSVP Protocol Example



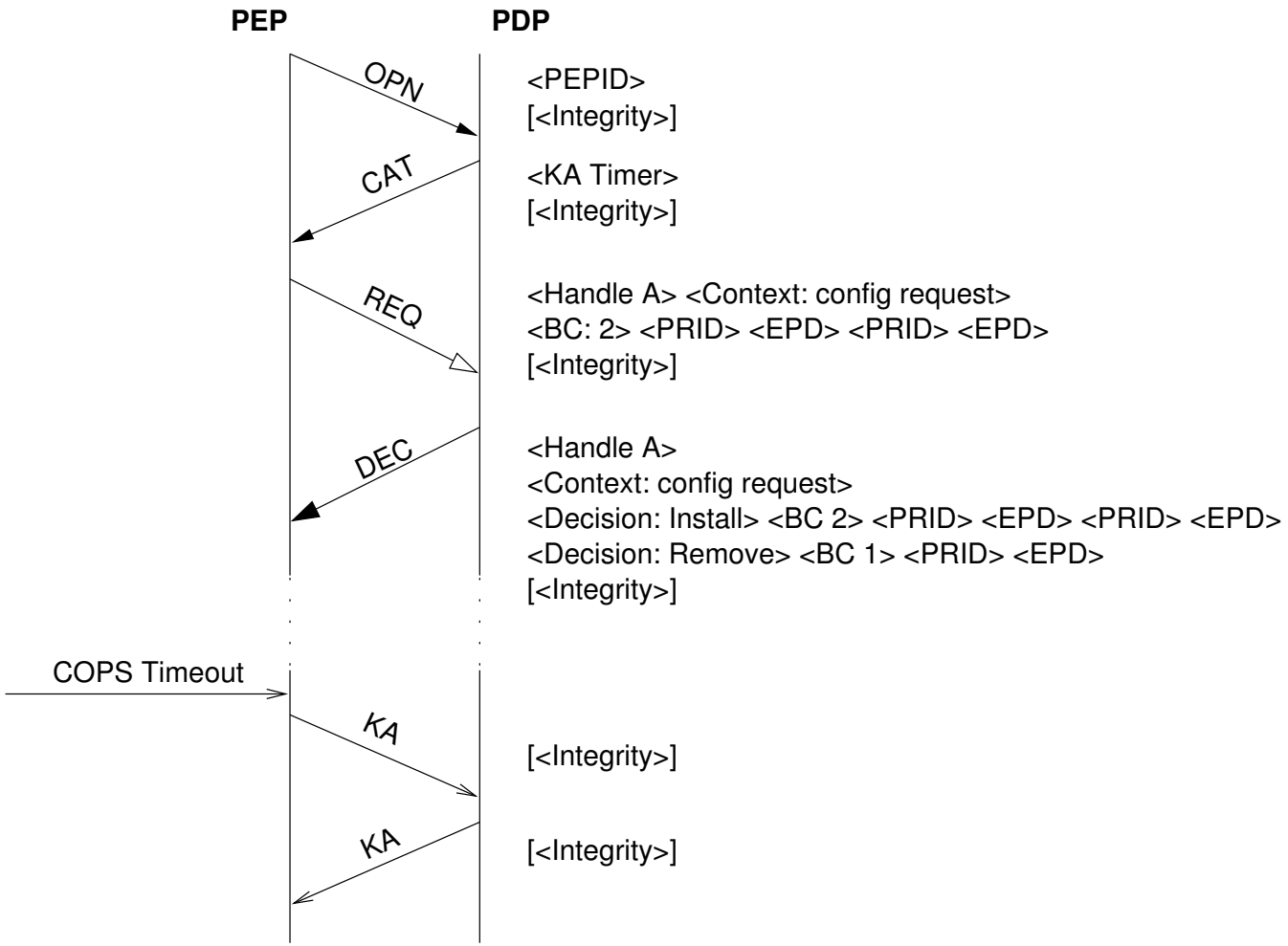
3.4 COPS for Provisioning

- Provisioning of policy configurations (a set of policy class instances) at PEPs.
- COPS-PR specific terminology (inconsistent with other WGs):
  - Policy Rule Class (PRC):
    An ordered set of attributes. PRCs are defined in PIB modules and registered in the Object Identifier tree.
  - Policy Rule Instance (PRI):
    An instantiation of a PRC.
  - Policy Rule Instance Identifier (PRID):
    A positive integer which identifies a PRI of a given PRC.
  - Encoded Policy Instance Data (EPD):
    BER encoded representation of a PRI.
- COPS-PR allows to install and remove EPD on a PEP.

Policy Information Base (PIB)

- A Policy Information Base (PIB) defines a set PRCs for use with COPS-PR.
- PIB modules are written using the Structure of Policy Provisioning Information (SPPI).
- The SPPI is an adapted superset of the SNMP's SMIv2.
- Hooks in the SPPI can be used to generate MIBs from PIBs for usage with SNMP.

COPS-PR Protocol Example



3.5 SPPI versus SMIv2

- SPPI supports 64 bit base types.
- SPPI does not support scalars.
- SPPI tables are always indexed by a single positive integer.
- SPPI defines table specific install error codes.
- SPPI provides mechanisms to express simple table relationships.
- SPPI introduces the `EXTENDS` keyword in addition to `AUGMENTS`.
- SPPI defines access clauses for table entries and not for each individual columns.

3.6 COPS-PR versus SNMP

- Common Open Policy Service for Provisioning (COPS-PR):
  - Reliable, efficient transport over TCP supports large message sizes.
  - Operates on PRIs (table rows) instead of individual attributes.
  - Operations to install/remove PRIs (table rows).
  - Compact encoding of of PRIs (table rows) increases bandwidth efficiency.
  - A COPS-PR connection gives the PDP complete control over the PEP.
- Simple Network Management Protocol (SNMP):
  - Unreliable transport over UDP with small message sizes.
  - Operates on lists of simple variables.
  - Supports multiple managers that simultaneously access an agent.
  - Access control can be exercised on the managed element.
  - Complex mechanism to create/delete table rows (RowStatus).
  - Unlimited complexity of quasi atomic SNMP set operations.
  - Poor efficiency due to OID naming overhead in SNMP PDUs.

4. Evolution of the Internet Management Framework

4.1 Next Generation Structure of Management Information (SMIng)

4.2 SNMP over TCP

4.3 SNMP Payload Compression

4.4 Extended SNMP Protocol Operations

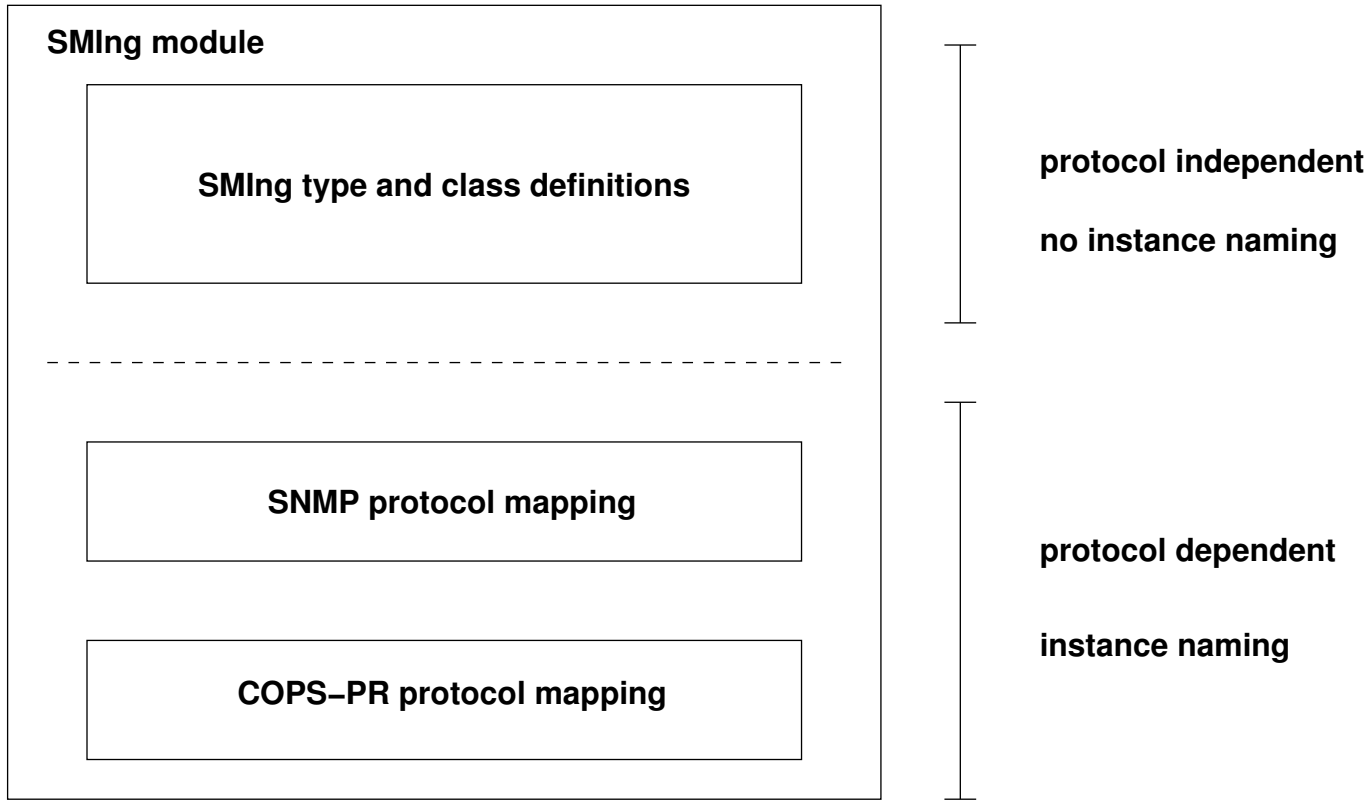
4.5 Session-Based SNMP Security Model

4.1 Next Generation Structure of Management Information

- Problems:
  - SMIv2 misses some important base types such as 64 bit numbers.
  - SMIv2 lacks reusable composite data types.
  - SMIv2 syntax depends on ASN.1 and is generally not well understood and implemented correctly.
  - SMIv2 parsers are difficult to write due to a lack of a well defined grammar.
  - SMIv2 is not extensible.
  - Desirable to use the same data definitions with SNMP and COPS-PR.

- Solution:
  - A next generation data modeling language called SMI (SMIng).
  - Detailed objectives have been documented in RFC 3216.
  - Soon to be published as Experimental RFCs.

SMIng Module Structure



- Reusable type and class definitions are seperated from protocol specific mappings.
- Abstraction of instance naming is the most difficult problem to solve.

SMIng Syntax

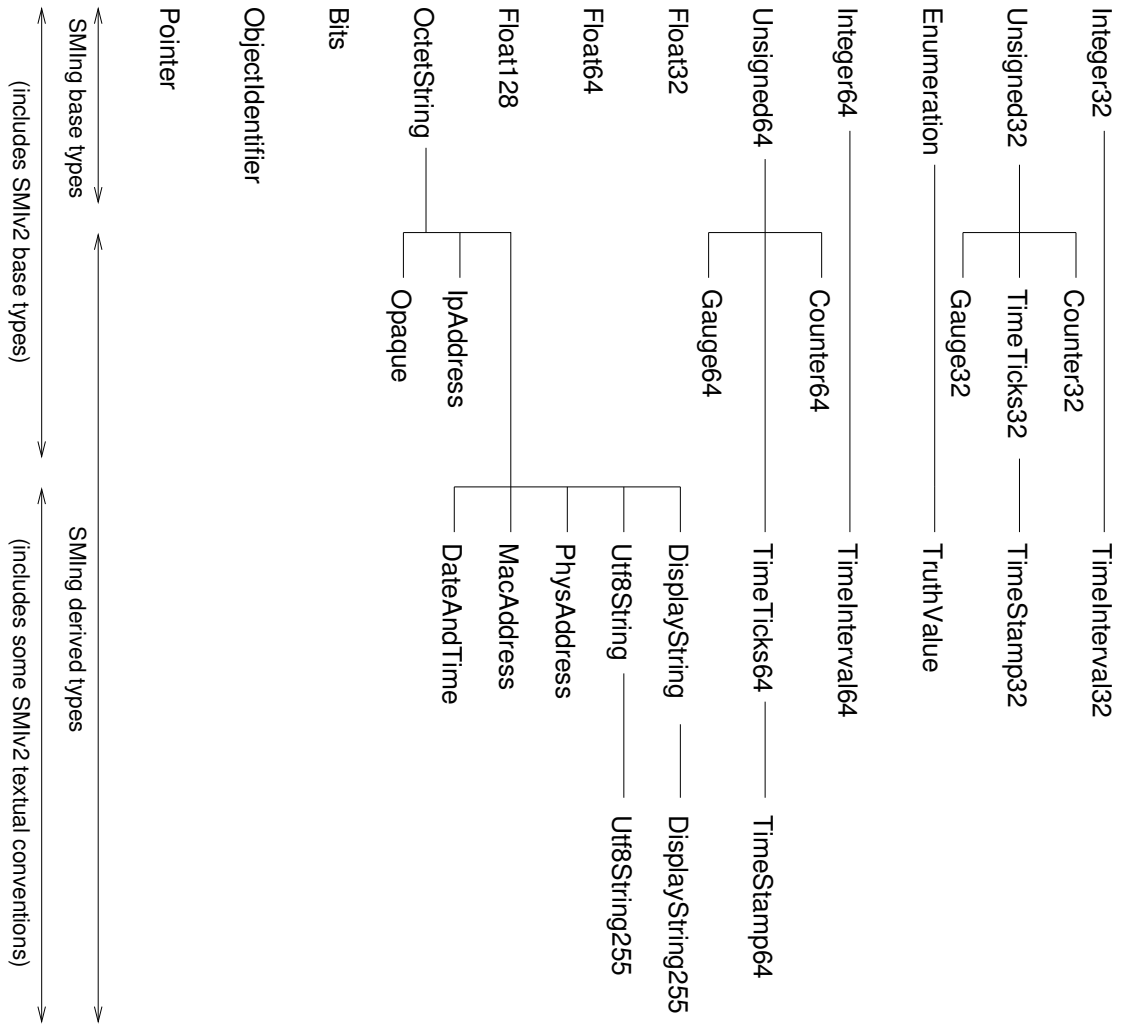
- Programmer friendly syntax:
  - look and feel similar to Java, C, C++, ...
  - consistent structure of statements (easier to memorize)

- Easy to implement and efficient to parse:
  - consistent syntactic structure simplifies grammar
  - no forward references (except in cases where they are unavoidable)
  - statement seperators help to recover from parse errors
  - complete grammar specified in ABNF (RFC 2234)

- Language extensibility:
  - declaration of new statements, parsers skip unknown statements

$\Longrightarrow$ Not everyone in the IETF agrees that the ASN.1 syntax is ugly...

SMIng Base Types and Core Derived Types



SMIng Attributes and Classes

- Classes encapsulate a set of attributes.
- Attributes have an associated type which can be
  - a base type, or
  - a derived type, or
  - a class (composite type).
- Classes can have associated events.
- Every event in SMIng is associated with a class.
- Events can be mapped to notification messages in protocol mappings.
- Methods are not supported, but might be added in the future.

SNMP Protocol Mapping

- Defines how SMIng base data types are mapped to SNMP data types.
- Uses Opaque wrapping to support new base types.
- Complex composite types are flattened and mapped to table rows or groups of scalars.
- OID names are assigned in mapping statements.
- SNMP specific derived types (e.g. RowStatus) are defined in a mapping module.

SMIng Example

```
class BasicInOutErrStats {
    attribute inOctets  { type Counter32;  ... };
    attribute inErrors  { type Counter32;  ... };
    attribute outOctets { type Counter32;  ... };
    attribute outErrors { type Counter32;  ... };
    ...
};

class Interface {
    attribute index { type InterfaceIndex; ... };
    attribute stats { type BasicInOutErrStats; ... };
    ...
};

snmp {
    table ifTable {
        oid    interfaces.2;
        index  (ifIndex);
        object ifIndex     { implements Interface.index; ... };
        object ifInOctets  { implements Interface.stats.inOctets; ... };
        object ifInErrors  { implements Interface.stats.inErrors; ... };
        object ifOutOctets { implements Interface.stats.outOctets; ... };
        object ifOutErrors { implements Interface.stats.outErrors; ... };
        ...
    };
    ...
};
```

4.2 SNMP over TCP (RFC 3430)

- Motivation:
  - Support larger message sizes to improve bulk transfers.
  - Support session-based security mechanisms.
  - No vehicle to turn unconfirmed operations into confirmed operations.
- Specification:
  - Optional transport mapping (UDP still has to be available).
  - Originator of a request-response transaction chooses the transport protocol for the entire transaction.
  - Framing relies on ASN.1/BER message length information.
  - Implementations must provide buffers to reassemble fragmented messages.
  - Piggybacking of TCP ACKs important!

4.3 SNMP Payload Compression

- Motivation:
  - Lossless compression of SNMP payloads with minimal processing overhead.
  - Improve encoding efficiency to ship more data in SNMP messages.

- Requirements:
  - Compression must happen before encryption.
  - 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 must be defined using ASN.1 to ensure that compressed SNMP messages have a valid ASN.1/BER encoding.
  - SNMP payload compression should be able to support multiple compression algorithms. It is desirable to define common compression algorithms in order to achieve interoperability.

OID Delta Compression (ODC)

- Reduce the OID overhead inherent in SNMP messages by encoding OIDs of variable names as deltas to the previous OID
- The deltas are expressed by a combination of the following primitives:
  1. Substitution of a single sub-identifier at a certain position
  2. Substitution of ranges of sub-identifiers at a given start position
  3. Truncation and enlargement of the OID

- Algorithm:
  1. Loop through the SNMP PDU until you find an OID name value pair (varbind).
  2. If it is the first varbind, make a copy of the OID, pass it to the output buffer and continue with the next varbind.
  3. Otherwise, compute the delta to the last OID and BER encode it into the CompOID value.
  4. If the CompOID representation is larger than the OID, pass the OID to the output buffer, else pass the CompOID to the output buffer.
  5. Update the last OID and goto step two if there are additional varbinds.

4.4 Extended SNMP Protocol Operations

- Additional protocol operations can substantially improve SNMP's capabilities:
  - `GetConfig` and `SetConfig` to read and write configuration settings.
  - `CallRequest` and `CallResponse` to invoke operations.
  - `GetTable` to retrieve complete tables with filtering and OID suppression.
  - `Create` and `Delete` to address the complexity of the `RowStatus` mechanism.
  - Object-oriented PDUs with transaction support.

$\Longrightarrow$ There is no agreement which primitives are needed in which order and complexity.

4.5 Session-Based SNMP Security Model

- Problem:
  - The SNMP USM security model and VACM access control model are self-contained (following the original SNMP design goals) and do not integrate well into deployed authentication/authorization infrastructures.
  - Operators prefer to keep the number of authentication/authorization systems that must be managed to a minimum.

- Session-based security model (SSM):
  - Uses established security protocols, such as TLS/SSL or SSH.
  - SSM binds session to a connection, requiring SNMP over TCP.
  - Must map security protocol parameters into the SNMP parameters `securityName` and `securityLevel`.
  - Improved efficiency compared to USM under normal network conditions due to shared state about the security session.
  - Authentication should integrate with technologies such as Radius or SASL.

$\Longrightarrow$ Details still need to be worked out - no specification yet.

5. Revolution of the Internet Management Framework

5.1 XML Technologies Overview

5.2 NetConf Proposal based on JunoScript

5.3 NetConf Proposal using Web Services

5.4 SMI Translations and SNMP Gateways

5.1 XML Technologies Overview

XML The eXtensible Markup Language is a standard markup language that allows application s to exchange structured documents.

XSD The XML Schema Definition language offers facilities for describing the structure and constraining the contents of XML documents.

XSL The eXtensible Stylesheet Language is a family of recommendations for defining XML document transformation and presentation.

XSLT The eXtensible Stylesheet Language Transformations is a language for transforming XML documents into other XML documents.

XPATH The XML Path Language is a language for addressing parts of an XML document.

XQUERY The XML Query Language is a query langauge to extract data from XML documents.

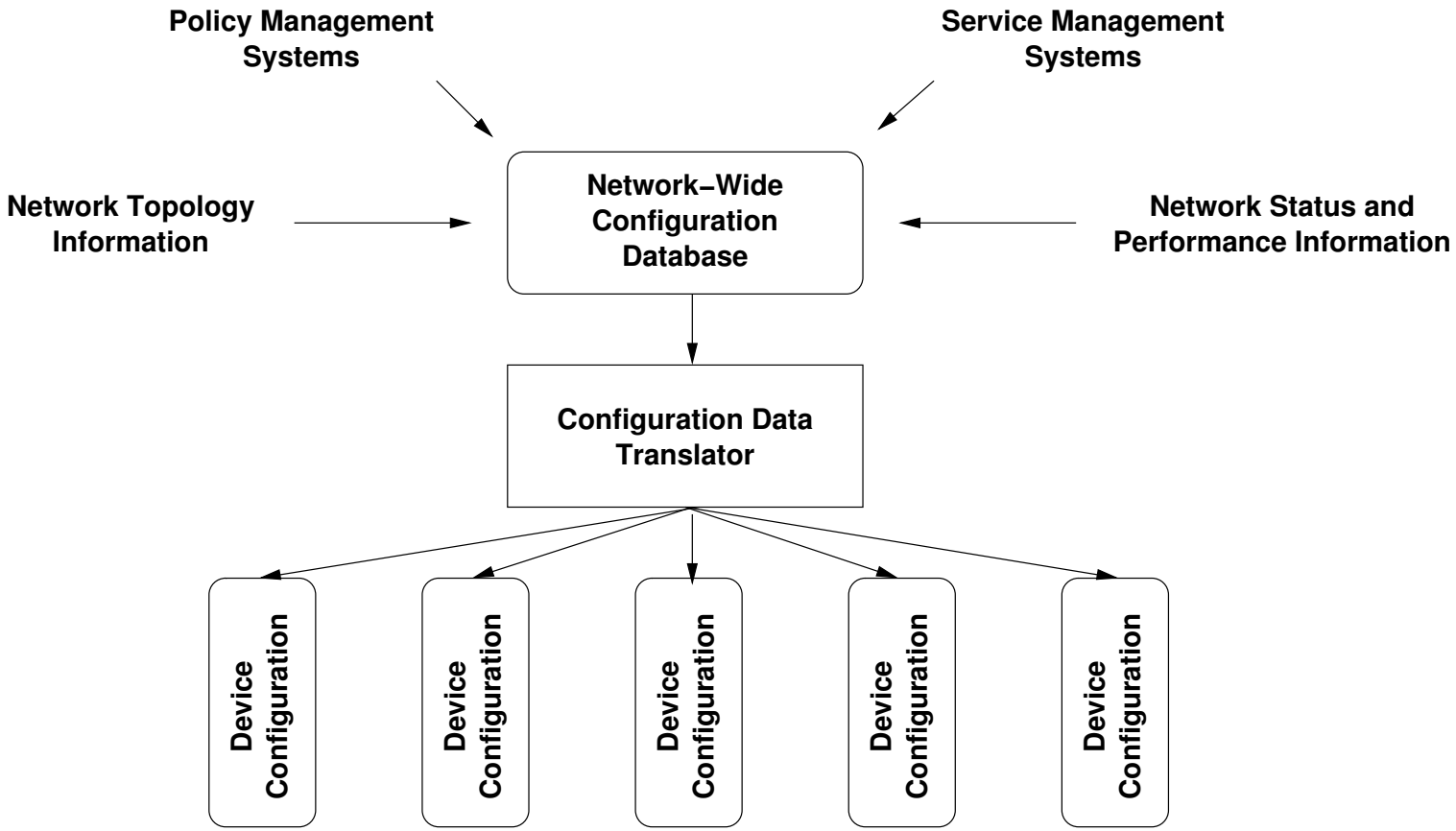
SOAP The Simple Object Access Protocol is for exchanging XML encoded messages.

WSDL Web Services Description Language is a language to describe the behavior of collections of XML encoded messages.

DOM The Document Object Model is a way to represent XML documents in memory.

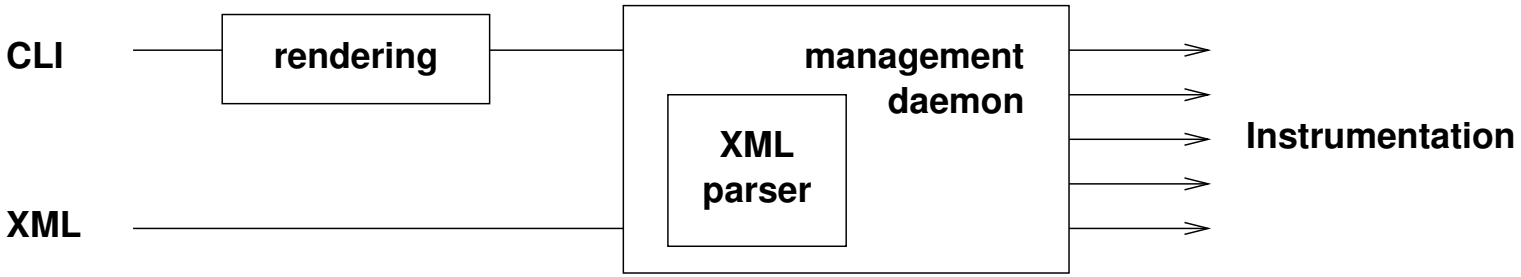
SAX SAX is an event-driven API to parse and access XML documents.

Network-Wide Configuration Management Model



$\Longrightarrow$ Treating configurations as documents leads naturally to the application of XML.

5.2 NetConf Proposal based on JunoScript



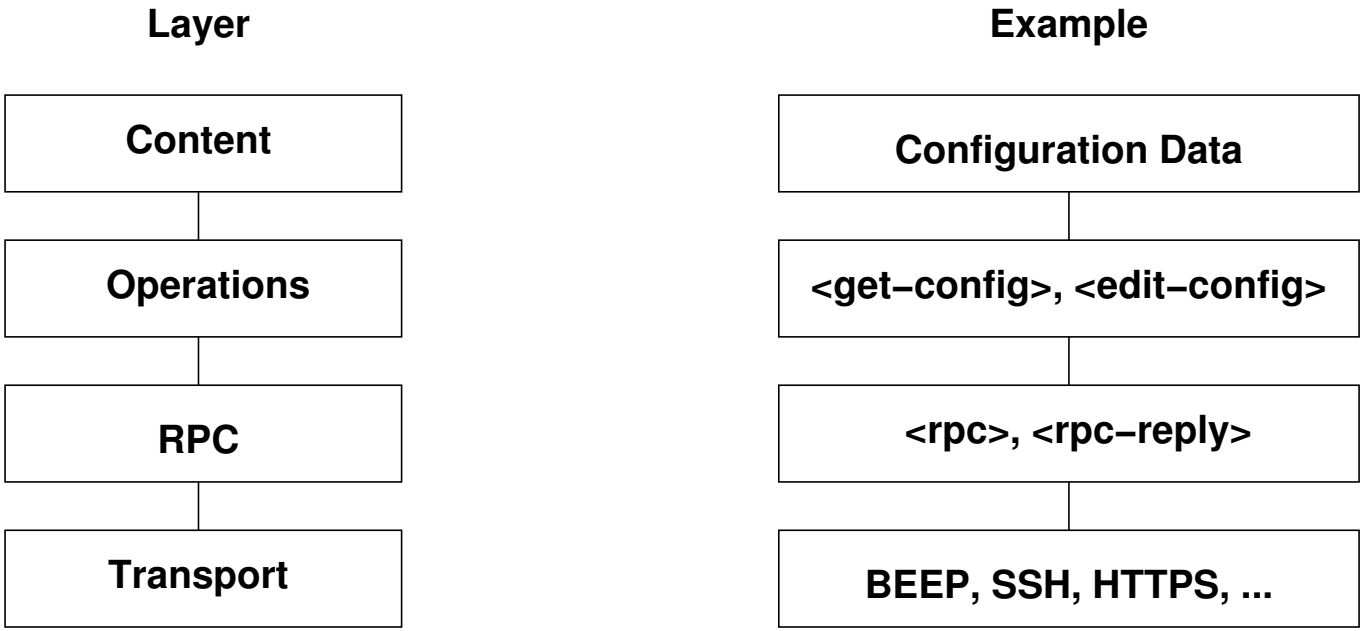
- Juniper Networks developed JunoScript as a programmatic interface to (and within) their router products.
- JunoScript uses XML for data representation and the protocol messages.
- The Juniper command line interface is based internally on JunoScript.
- JunoScript is a simple RPC protocol running of Telnet or SSH.
- JunoScript provides the primitives to build robust network-wide configuration management systems (timed confirmed commits).

Example JunoScript RPC Interaction

```
<rpc>
  <get-interface-information>
    <statistics/>
  </get-interface-information>
</rpc>

<rpc-reply>
  <interface-information>
    <InOctets>123456</InOctets>
    <InErrors>789</InErrors>
    <OutOctets>654321</OutOctets>
    <OutErrors>0</OutErrors>
  </interface-information>
</rpc-reply>
```

- All RPC interactions over a single connection form together a single XML document.
- Filtering is based on simple subtree selection.

NetConf Layering Model



- Security has to be provided by the transport layer.
- The operations layer provides the primitives to handle configurations.
- The content layer is currently not subject to standardization.

NetConf Operations (under discussion)

- `get-config(source, filter, format)`
  Retrieve all or part of a specified configuration from a source in a given format.
- `edit-config(target, format, options, config)`
  Load all or part of a specified configuration to the specified target configuration.
- `copy-config(source, target, format)`
  Create or replace an entire configuration with the contents of another configuration.
- `delete-config(target)`
  Delete a configuration datastore.
- `get-state(filter, format)`
  Retrieve device state information.
- validate(source)
  Validate the contents of the specified configuration.
- lock(source)
  Lock a configuration source.
- unlock(config)
  Unlock a configuration source.
- commit(confirmed, confirmed-timeout)
  Commit the candidate configuration as the device's new current configuration.

NetConf over BEEP Request Example (under discussion)

```
MSG 1 42 . 24 291
Content-Type: text/xml; charset=utf-8
<rpc message-id="105" xmlns="http://ietf.org/xmlconf/1.0/base">
  <get-config>
    <source>
      <running/>
    </source>
    <config xmlns="http://example.com/schema/1.2/config">
      <users/>
    </config>
    <format>xml</format>
  </get-config>
</rpc>
END
```

- BEEP (RFC 3080) is a generic application protocol kernel for connection-oriented, asynchronous interactions.
- Supports exchange styles MSG/RPY, MSG/ERR, MSG/ANS, multiple channels, application layer framing and fragmentation.
- Integrates into SASL (RFC 2222) and TLS (RFC 2246) for security.

NetConf over BEEP Request Example (cont.)

```
RPY 1 42 . 24 705
<rpc-reply message-id="105" xmlns="http://ietf.org/xmlconf/1.0/base">
  <config xmlns="http://example.com/schema/1.2/config">
    <users>
      <user>
        <name>root</name>
        <type>superuser</type>
        <full-name>Charlie Root</full-name>
      </user>
      <user>
        <name>fred</name>
        <type>admin</type>
        <full-name>Fred Flintstone</full-name>
      </user>
      <user>
        <name>barney</name>
        <type>admin</type>
        <full-name>Barney Rubble</full-name>
      </user>
    </users>
  </config>
</rpc-reply>
END
```

5.3 NetConf Proposal using Web Services

- Instead of inventing a special purpose RPC protocol, use Web Service standards.

- Pros:
  - more developers available
  - more tools available
  - better integration with IT infrastructure

- Cons:
  - base technology not under control of the IETF
  - unneeded complexity
  - interoperability problems (immature technology)
  - HTTP is not a good match for a generic application protocol kernel

- Ongoing debate in the IETF how to weight the arguments.

NetConf WSDL Definition Example

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:tns="http://ietf.org/netconf/1.0/soap"
             xmlns:xb="http://ietf.org/netconf/1.0/base"
             targetNamespace="http://ietf.org/netconf/1.0/soap"
             name="http://ietf.org/netconf/1.0/soap">

  <import namespace="http://ietf.org/netconf/1.0/base" location="base.xsd"/>

  <message name="rpcRequest">
    <part name="in" element="xb:rpc"/>
  </message>
  <message name="rpcResponse">
    <part name="out" element="xb:rpc-reply"/>
  </message>

  <portType name="rpcPortType">
    <operation name="rpc">
      <input message="tns:rpcRequest"/>
      <output message="tns:rpcResponse"/>
    </operation>
  </portType>
```

NetConf WSDL Definition Example (cont.)

```
  <binding name="rpcBinding" type="tns:rpcPortType">
    <SOAP:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="rpc">
      <SOAP:operation/>
      <input>
        <SOAP:body use="literal" namespace="http://ietf.org/netconf/1.0/base"/>
      </input>
      <output>
        <SOAP:body use="literal" namespace="http://ietf.org/netconf/1.0/base"/>
      </output>
    </operation>
  </binding>
</definitions>
```

- SOAP binding based on a hypothetical location for the NETCONF schema.
- Warning: This example mapping does not provide security!

NetConf WSDL Service Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xs="http://ietf.org/netconf/1.0/soap"
             targetNamespace="urn:myNetconfService"
             name="myNetconfService.wsdl">

  <import namespace="http://ietf.org/netconf/1.0/soap" location="soap.wsdl"/>

  <service name="netconf">
    <port name="rpcPort" binding="xs:rpcBinding">
      <SOAP:address location="http://localhost:8080/netconf"/>
    </port>
  </service>
</definitions>
```

- Service definition based on a hypothetical location for the NETCONF/SOAP WSDL definitions.

NetConf over SOAP/HTTP Example

```
POST /netconf HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, text/*
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: "netconfsession:123"
Content-Length: 470

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <rpc id="101" xmlns="http://ietf.org/netconf/1.0/base">
      <get-config>
        <source>
          <running/>
        </source>
        <config xmlns="http://example.com/schema/1.2/config">
          <users/>
        </config>
        <format>xml</format>
      </get-config>
    </rpc>
  </soapenv:Body>
</soapenv:Envelope>
```

NetConf over SOAP/HTTP Example (cont.)

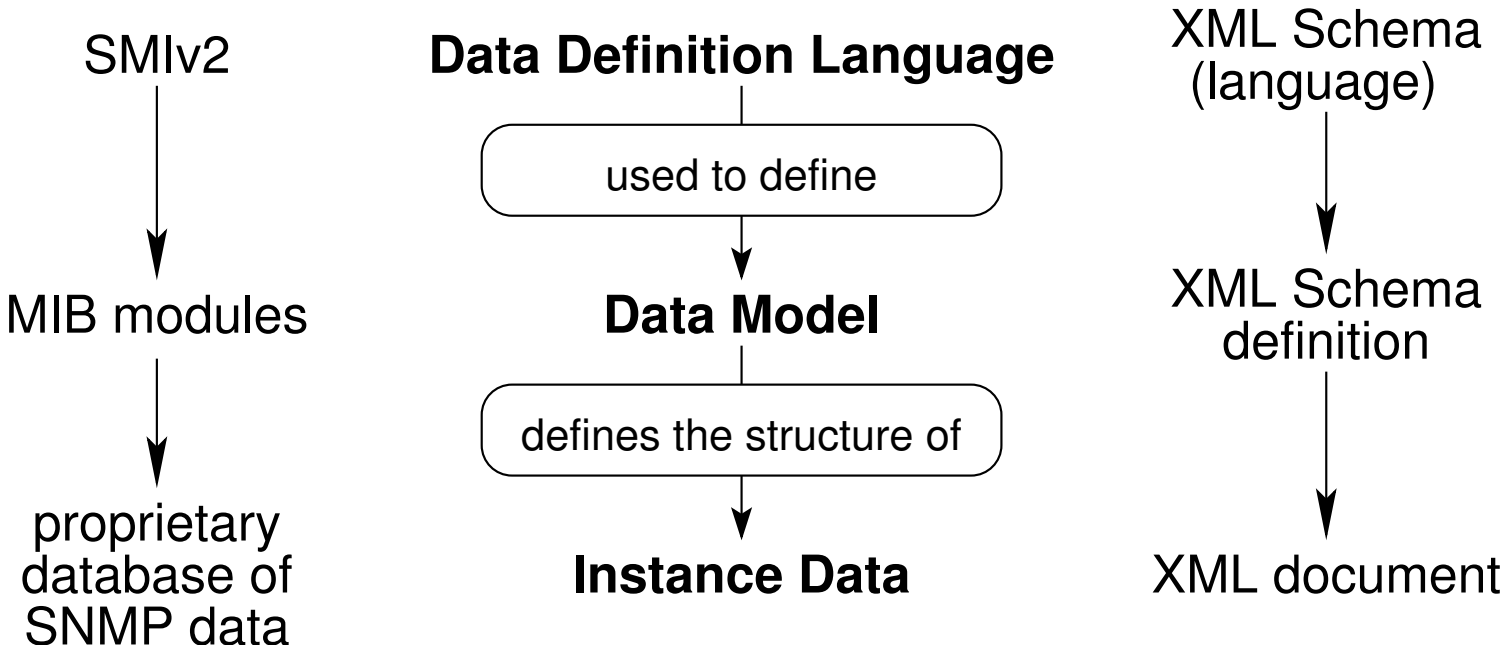
```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <rpc-reply id="101" xmlns="http://ietf.org/netconf/1.0/base">
      <config xmlns="http://example.com/schema/1.2/config">
        <users>
          <user>
            <name>root</name>
            <type>superuser</type>
          </user>
          <user>
            <name>fred</name>
            <type>admin</type>
          </user>
          <user>
            <name>barney</name>
            <type>admin</type>
          </user>
        </users>
      </config>
    </rpc-reply>
  </soapenv:Body>
</soapenv:Envelope>
```

NetConf Open Issues

- Transport: BEEP?, SSH?, SOAP/HTTP[S]?, SOAP/BEEP?, ...
- Filtering: ad-hoc subtree?, XPATH?, XQUERY?, XPATH light?, ...
- Protocol: primitives? locking mandatory? XML usage?
- Modeling: XML Schema?, RELAXng?, SMIng?, ...
- Integration: SNMP?, CLI?, ...

$\Longrightarrow$ NetConf interim meeting September 2003, design team just formed

5.4 SMI Translations and SNMP Gateways



- No standard format for storing / exchanging instance data in the SNMP world.
- Automatically translate MIB modules into XML schemas to save investments.
- Such translations should follow the "XML style" as close as possible.

Multiple "Contexts" per XML Document

A single document may contain data

- of multiple engines (agents)
  - @ipaddr
  - @hostname
  - @port
- of multiple per-agent contexts
  - @context or
  - @community
- of multiple points in time
  - @time

```
<?xml version="1.0"?>
<snmp-data [...]>
  <context
      ipaddr="134.169.246.1"
      hostname="ciscobs.rz.tu-bs.de"
      port="161"
      community="public"
      time="2003-03-10T10:31:16Z">
    [...context data...]
  </context>

  <context [...]>
    [...context data...]
  </context>

  [...]
</snmp-data>
```

No "deep" Element Nesting

- 1st level element <snmp-data>
  independent root element
  (not bound to a specific MIB, agent, or point in time)
- 2nd level elements <context>
- 3rd level elements e.g. <system>, <ifEntry ifIndex="1">
  - groups of scalar elements
  - table rows, identified through index attributes
- 4th level elements e.g. <sysContact>, <ifInOctets>
  - scalar elements
  - columnar elements (also of table augmentations)
- deeper level elements
  only for "table-in-table" relationships, hence quite rare

$\Longrightarrow$ Note: The element nesting is not based on the OID tree.

Using XML Namespaces to Identify Modules

- Each MIB will be compiled to a separate XML Schema that defines an according namespace:

```
<xsd:schema
    targetNamespace="http://example.com/IF-MIB" [...]>
  [...]
</xsd:schema>
```

- Imports from MIB modules are translated to imports of namespaces:

```
<xsd:schema [...]
    xmlns:SNMPv2-MIB="http://example.com/SNMPv2-MIB" [...]>
  [...]
  <xsd:import
    namespace="http://example.com/SNMPv2-MIB" [...]/>
  [...]
</xsd:schema>
```

- Elements can be named uniquely with namespace prefixes:

```
<IF-MIB:interfaces>
  <IF-MIB:ifNumber>7</IF-MIB:ifNumber>
</IF-MIB:interfaces>
```

Value Representations and Schema Definitions

- numeric values

  XML: display hints applied, represented in decimal digits
  Schema: range restrictions (<minInclusive>, <maxInclusive>)
  display hints (<fractionDigits>)

- octet strings with display hints

  XML: represented as strings conforming to display hints,
  Schema: DISPLAY-HINTs converted to <pattern> reg-exp's (as good as possible)

- octet strings without display hints

  XML: represented as sequences of 2-digit hex values
  Schema: based on the hexBinary type

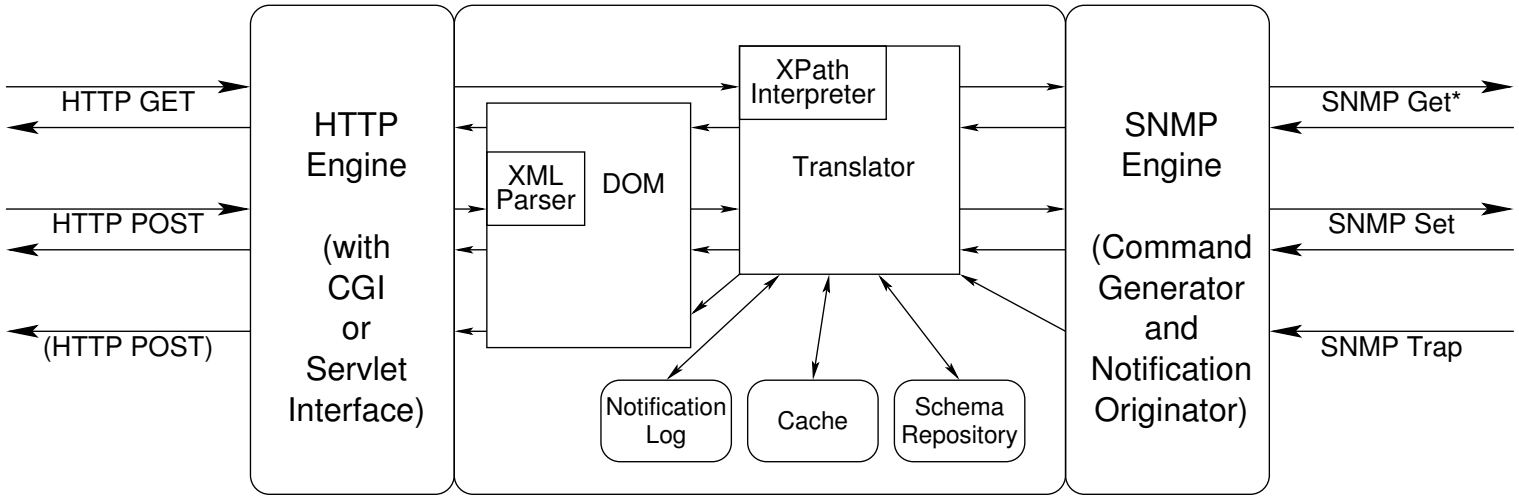
- enumeration and bit set values

  XML: represented as (sequences of) labels
  Schema: (<list>s of) <enumeration> values

Example XML Document

```
<snmp-data "xmlns=http://example.com/TCP-MIB">
  <context ipaddr="134.169.34.81" hostname="tom.example.com"
           port="161" community="public" time="2003-03-17T11:07:53Z">
    <TCP-MIB:tcp>
      <TCP-MIB:tcpRtoAlgorithm>other</TCP-MIB:tcpRtoAlgorithm>
      <TCP-MIB:tcpRtoMin>0</TCP-MIB:tcpRtoMin>
      [...]
    </TCP-MIB:tcp>
    <TCP-MIB:tcpConnEntry
        tcpConnLocalAddress="0.0.0.0" tcpConnLocalPort="9"
        tcpConnRemAddress="0.0.0.0" tcpConnRemPort="0">
      <TCP-MIB:tcpConnState>listen</TCP-MIB:tcpConnState>
    </TCP-MIB:tcpConnEntry>
    <TCP-MIB:tcpConnEntry
        tcpConnLocalAddress="134.0.0.0" tcpConnLocalPort="42077"
        tcpConnRemAddress="134.0.0.0" tcpConnRemPort="6010">
      <TCP-MIB:tcpConnState>established</TCP-MIB:tcpConnState>
    </TCP-MIB:tcpConnEntry>
  </context>
</snmp-data>
```

SNMP/XML Gateway Prototype (TU Braunschweig)



Example: Retrieve the descriptions of the interfaces at talisker.ibr.cs.tu-bs.de
that are currently in operation and transmitted or received at least one packet:

```
$ lynx -dump 'http://www.ibr.cs.tu-bs.de/snmp-xml-gw?\
  get=/snmp-data/context[@hostname="talisker.ibr.cs.tu-bs.de"]\
  /ifEntry[ifOperStatus="up" and (ifOutOctets > 0 or ifInOctets > 0)]\
  /ifDescr'
```

6. References & Online Resources

6.1 Request for Comments (RFCs)

6.2 Books and Articles

6.3 Online Information

6.1 Request for Comments (RFCs)

- J. Case, R. Mundy, D. Partain, B. Stewart: Introduction and Applicability Statements for Internet Standard Management Framework, RFC 3410, December 2002
- D. Harrington, R. Presuhn, B. Wijnen: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, STD 62, RFC 3411, December 2002
- J. Case, D. Harrington, R. Presuhn, B. Wijnen: Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), STD 62, RFC 3412, December 2002
- D. Levi, P. Meyer, B. Stewart: Simple Network Management Protocol (SNMP) Applications, STD 62, RFC 3413, December 2002
- U. Blumenthal, B. Wijnen: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), STD 62, RFC 3414, December 2002
- B. Wijnen, R. Presuhn, K. McCloghrie: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), STD 62, RFC 3415, December 2002
- R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser: Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP), STD 62, RFC 3416, December 2002
- R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser: Transport Mappings for the Simple Network Management Protocol (SNMP), STD 62, RFC 3417, December 2002
- R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser: Management Information Base for the Simple Network Management Protocol (SNMP), STD 62, RFC 3418, December 2002
- R. Frye, D. Levi, S. Routhier, B. Wijnen: Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework, RFC 2576, March 2000

Request for Comments (RFCs)

- K. McCloghrie, D. Perkins, J. Schönwälder, J. Case, M. Rose, S. Waldbusser: Structure of Management Information Version 2 (SMIv2), STD 58, RFC 2578, April 1999
- K. McCloghrie, D. Perkins, J. Schönwälder, J. Case, M. Rose, S. Waldbusser: Textual Conventions for SMIv2, STD 58, RFC 2579, April 1999
- K. McCloghrie, D. Perkins, J. Schönwälder, J. Case, M. Rose, S. Waldbusser: Conformance Statements for SMIv2, STD 58, RFC 2580, April 1999
- M. Daniele, B. Wijnen, M. Ellison, D. Francisco: Agent Extensibility (AgentX) Protocol Version 1, RFC 2741, January 2000
- L. Heintz, S. Gudur, M. Ellison: Definitions of Managed Objects for Extensible SNMP Agents, RFC 2742, January 2000
- J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry: The COPS (Common Open Policy Service) Protocol, RFC 2748, January 2000
- J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, A. Sastry: COPS usage for RSVP, RFC 2749, January 2000
- K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, A. Smith: COPS Usage for Policy Provisioning (COPS-PR), RFC 3084, March 2001
- K. McCloghrie, M. Fine, K. Chan, S. Hahn, R. Sahita, A. Smith, F. Reichmeyer: Structure of Policy Provisioning Information (SPPI), RFC 3159, August 2001

Request for Comments (RFCs)

- D. Levi, J. Schönwälder: Definitions of Managed Objects for the Delegation of Management Scripts, RFC 3165, August 2001.
- J. Schönwälder, J. Quittek: Script MIB Extensibility Protocol Version 1.1, RFC 3179, October 2001
- M. Daniele, B. Haberman, S. Routhier, J. Schönwälder: Textual Conventions for Internet Network Addresses, RFC 3291, May 2002
- M. Daniele, J. Schönwälder: Textual Conventions for Transport Addresses, RFC 3419, December 2002
- J. Schönwälder: Simple Network Management Protocol Over Transmission Control Protocol Transport Mapping, RFC 3430, December 2002
- A. Pras, J. Schönwälder: On the Difference between Information Models and Data Models, RFC 3444, January 2003
- J. Schönwälder: Overview of the 2002 IAB Network Management Workshop, RFC 3535, May 2003.
- M. Rose: The Blocks Extensible Exchange Protocol Core, RFC 3080, March 2001
- E. O'Tuathail, M. Rose: Using the Simple Object Access Protocol (SOAP) in Blocks Extensible Exchange Protocol (BEEP), RFC 3288, June 2002

6.2 Books and Articles

- W. Stallings: SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Addison-Wesley, 1999
- D. Zeltserman: A Practical Guide to SNMPv3 and Network Management, Prentice Hall, 1999
- D. Perkins: RMON: Remote Monitoring of SNMP-Managed LANs, Prentice Hall, 1999
- D. Perkins, E. McGinnis: Understanding SNMP MIBs, Prentice Hall, 1997
- M. Rose: The Simple Book – An Introduction to Internet Management, Revised Second Edition, Prentice Hall, 1996
- M. Rose, K. McCloghrie: How to Manage Your Network Using SNMP, Prentice Hall, 1995
- J.P. Martin-Flatin: Web-based Management of IP Networks and Systems, Wiley, 2002
- The SimpleTimes, Special Issue on Agent Extensibility, SimpleTimes 4(2), April 1996
- The SimpleTimes, Special Issue on SNMP Version 3, SimpleTimes 5(1), December 1997
- M. White, S. Gudur: An Overview of the AgentX Protocol, SimpleTimes 6(1), April 1998
- J. Schönwälder, A. Pras, J.P. Martin-Flatin: On the Future of Internet Management Technologies, IEEE Communications Magazin, to appear, 2003.
- F. Strauß, T. Klie: Towards XML Oriented Internet Management, Proc. of IM 2003, Kluwer, 2003.

6.3 Online Resources

- Internet Engineering Task Force (IETF) `http://www.ietf.org/`
- IETF OPS Area `http://www.ops.ietf.org/`
- Internet Research Task Force (IRTF) `http://www.irtf.org/`
- Network Management Research Group (NMRG) `http://www.irtf.org/`
- The SimpleTimes `http://www.simple-times.org/`
- The Simple Web `http://wwwsnmp.cs.utwente.nl/`
- Tutorial Slides `http://www.ibr.cs.tu-bs.de/~schoenw/slides/`