Internet Management: Status and Challenges

Jürgen Schönwälder

j.schoenwaelder@iu-bremen.de

International University Bremen

Campus Ring 1

28725 Bremen, Germany

http://www.faculty.iu-bremen.de/schoenw/

Copyright Notice

Copyright © 2004 Jürgen Schönwälder,

International University Bremen, Germany

All rights reserved.

No part of these sheets may be reproduced, stored in a retrieval system or transmitted in any form or by any means, without obtaining written permission of the author.

Tutorial Overview

1. IETF Management Standards ($\approx$ 60 min)
   (a) Management Standards
   (b) Working Groups and Activities
2. Monitoring with SNMP ($\approx$ 60 min)
   (a) Simple Network Management Protocol (SNMP)
   (b) SNMP Version 3 (SNMPv3)
   (c) Integrated Security Models
3. Configuration with NETCONF ($\approx$ 60 min)
   (a) XML Technologies
   (b) Revolutionary Research
   (c) Summary and Outlook
4. Discussion

Management Standards

Why 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)

$\Rightarrow$ FCAPS functional areas (very broad but widely accepted functional categorization)

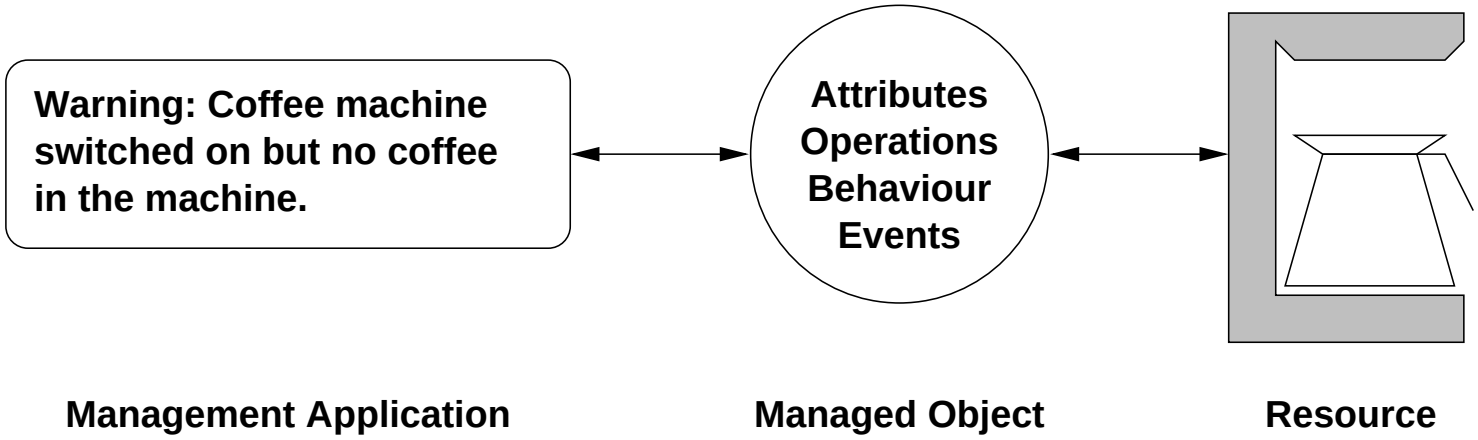
Why is Network Management Hard?

- Scalability is a key concern (millions of devices/users)
- Short technology life times (what happened to ATM?)
- Heterogenity requires standards-based solutions
- Lack of skilled persons

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

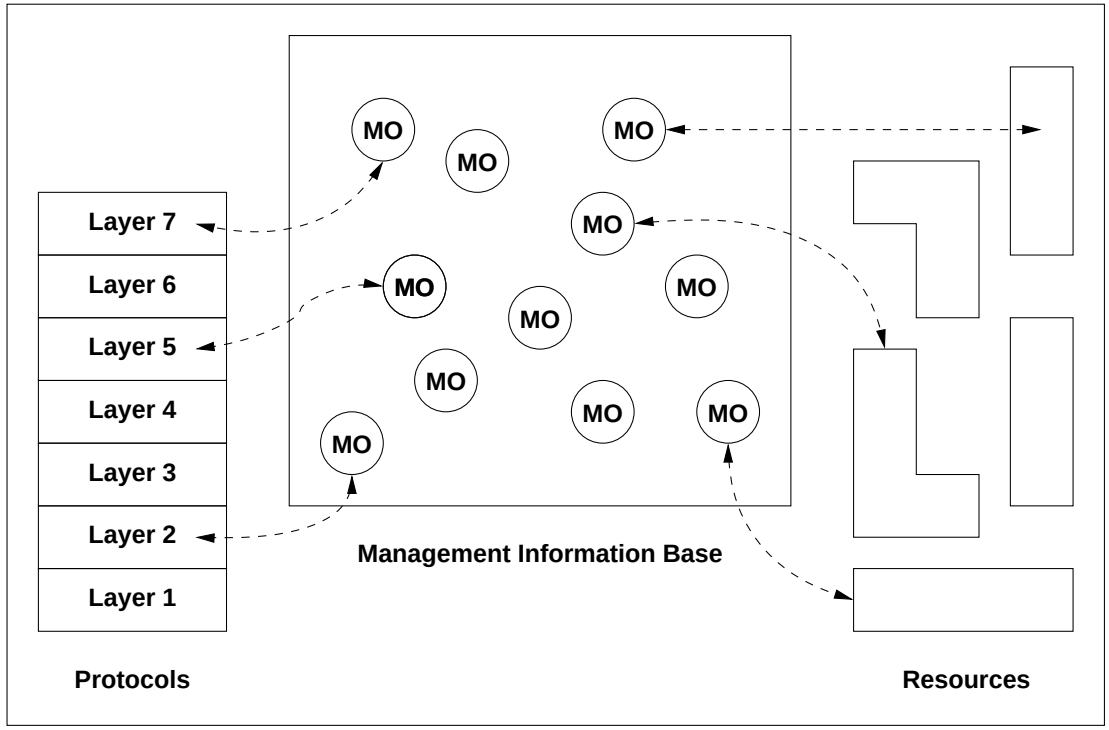
$\Longrightarrow$ However, network management terminology is often very different and sometimes somewhat confusing (especially for people with computer science background).

Abstraction of Managed Objects (MOs)



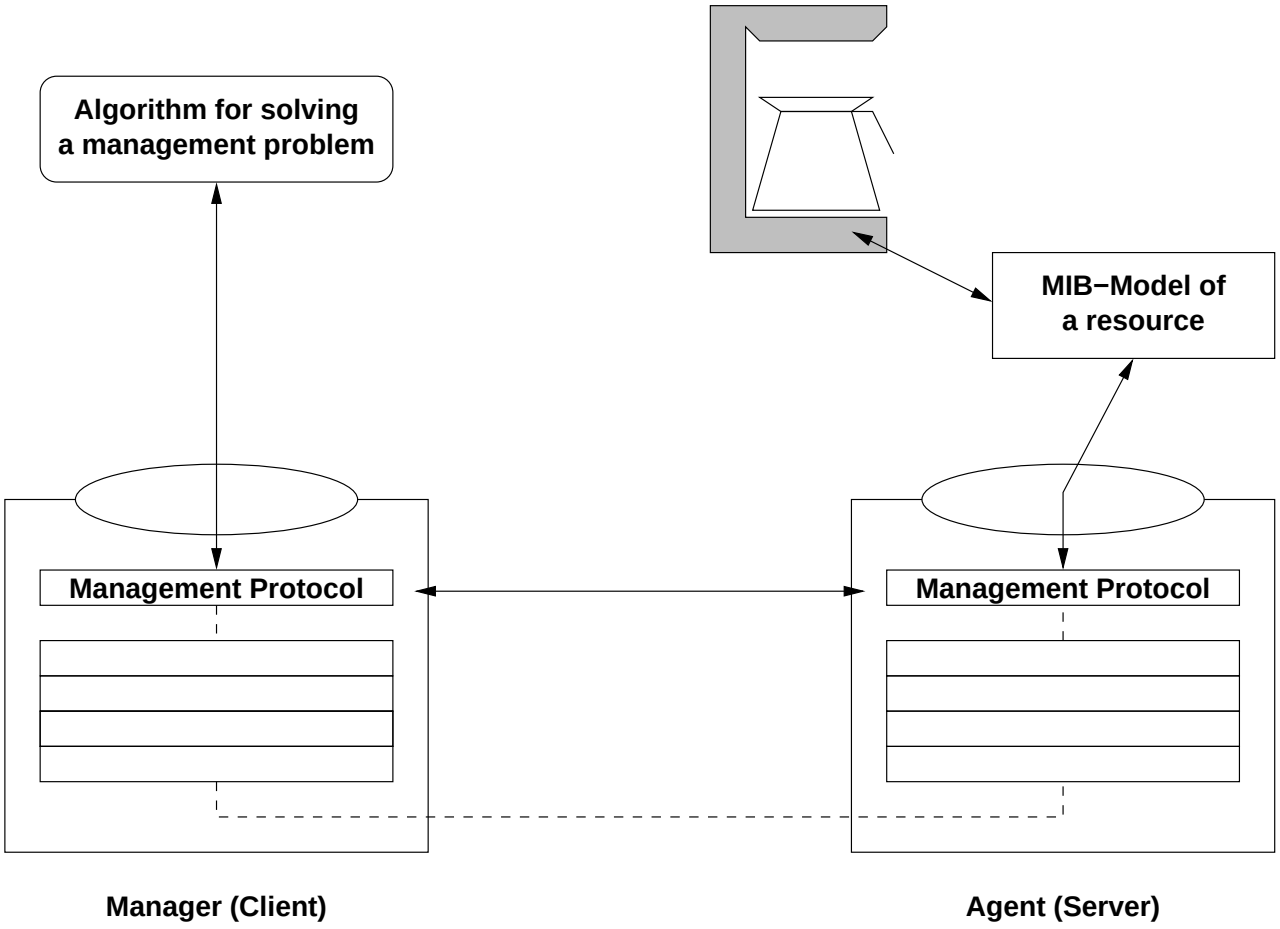
- A managed object is the abstracted view of a resource that presents its properties as seen by (and for the purpose of) management (ISO 7498-4).
- The boundary of a managed object defines the level of details which are accessible for management systems.

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.

Data-centric Approach

- The device is represented as a collection of data objects representing all the properties and capabilities of a device.
- The management protocol manipulates the data objects representing a device and its state.
- Manipulation of data objects might cause side effects.

$\Rightarrow$ Example: Internet management (SNMP)

Command-centric Approach

- The device is considered to be a stateful black box.
- A sequence of commands can be send to the device to
  a) change the state of the device or to
  b) retrieve data about the current state of the device (or portions thereof).
- Determining the right sequence of commands to bring a device into a certain state might not be trivial.

$\Rightarrow$ Example: Command line interfaces of routers or switches

Object-centric Approach

- The device is represented as a collection of data objects with associated methods.
- This can be seen as a combination of the data- and the command-centric approaches.
- Usually leads to object-oriented modeling and thus object-oriented approaches.
- A critical design decision is the *granularity* of the objects and the level of *interdependencies* between objects

$\Rightarrow$ Example: OSI management (CMIP), DMTF information models

Document-centric Approach

- The configuration and state of a device is represented as a structured document.
- Management operations are realized by manipulating the structured document.
- Allows to use general document processors for management purposes.
- Closely related to data-centric approaches.

$\Rightarrow$ Example: Most XML-based management approaches follow this model.

Essential Management Protocol Primitive

- From a very abstract viewpoint, the following set of management protocol primitives is essential for data-centric or object-centric management protocols:
  - `GET`, `SET`
  - `CREATE`, `DELETE`
  - `SEARCH` (or at the very least `ITERATE`)
  - `LOCK`, `UNLOCK`, `COMMIT`, `ROLLBACK`
  - `NOTIFY`
  - `EXECUTE` an operation or `INVOKE` a method
- Command-centric protocols usually have a very rich set of primitives (which are often hierarchically structured).

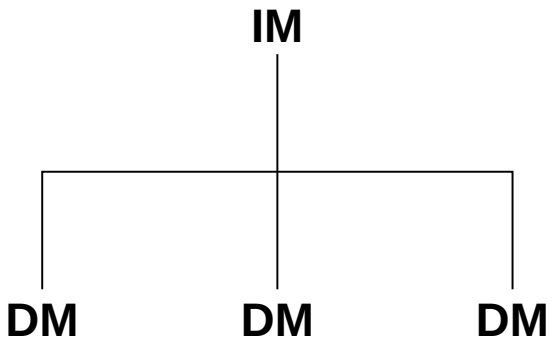
Information Models (RFC 3444)

- Information Models (IMs) are used to model managed objects at a conceptual level, independent of any specific 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, IMs should hide all protocol and implementation details.
- IMs focus on relationships between managed objects.
- IMs are often represented in Unified Modeling Language (UML) diagrams, but there are also informal IMs written in plain English language.

Data Models (RFC 3444)

- Data Models (DMs) are defined at a lower level of abstraction and include many details (compared to IMs).
- They are intended for implementors and include implementation- and protocol-specific constructs.
- DMs are often represented in formal data definition languages that are specific to the management protocol being used.

Information Models vs. Data Models

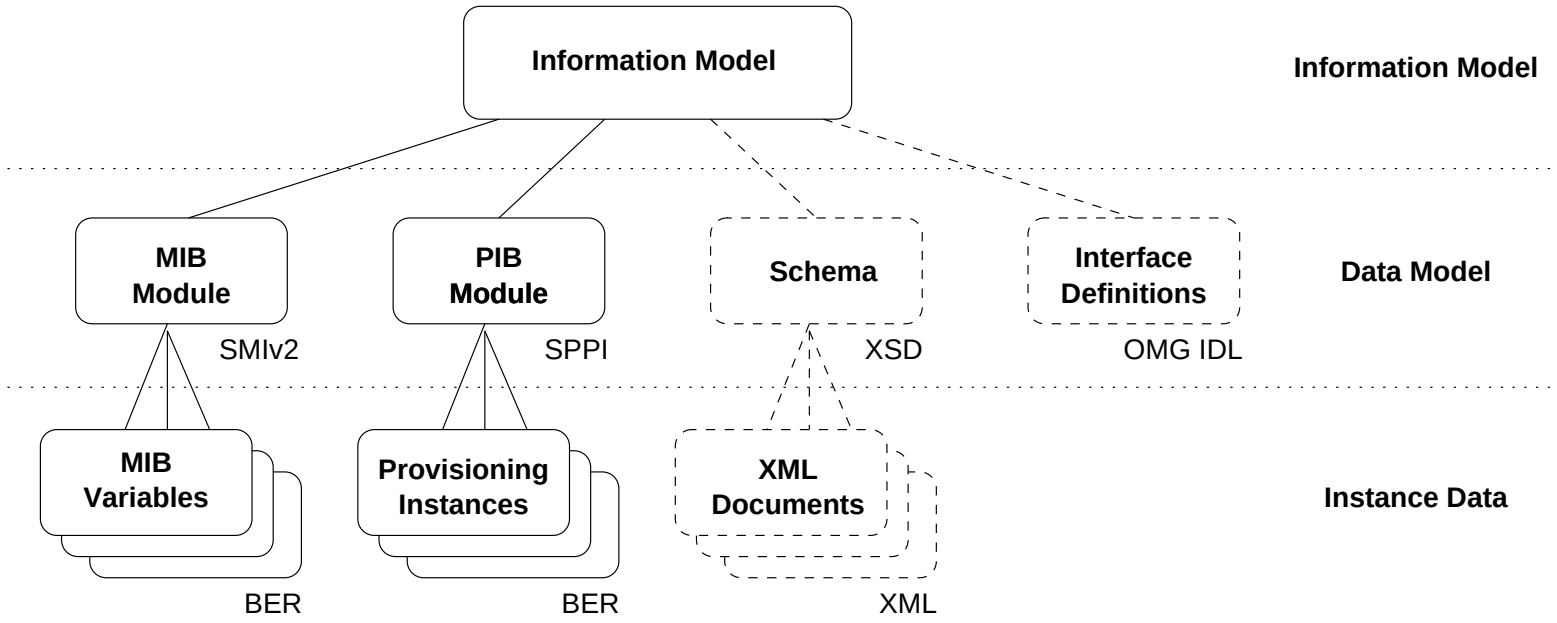


conceptual/abstract model for designers and operators

concrete/detailed model for implementors

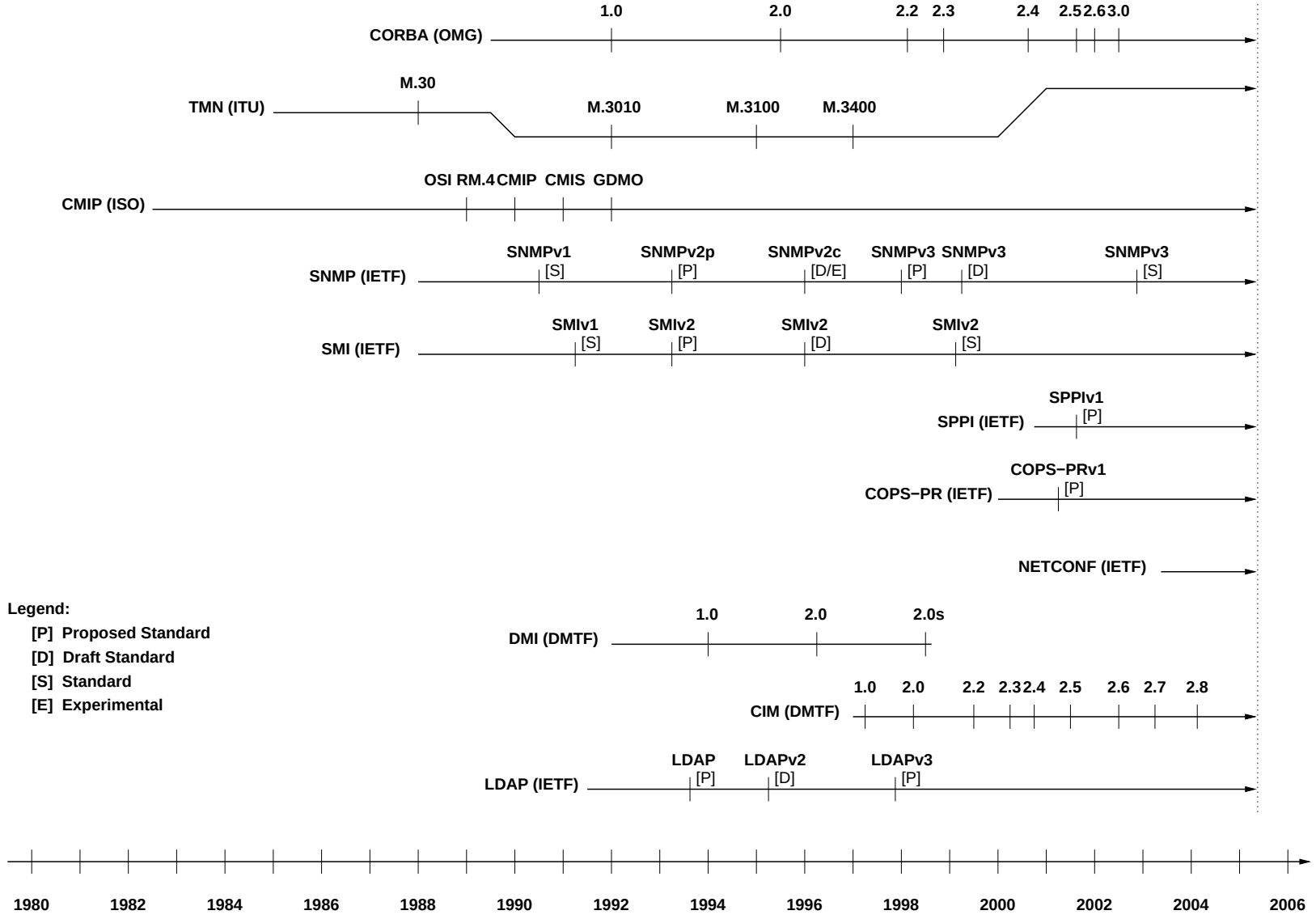
- Since conceptual models can be implemented in different ways, multiple DMs can be derived from a single IM.
- Although IMs and DMs serve different purposes, it is not always possible to decide which detail belongs to an IM and which detail belongs to a DM.
- Similarily, it is sometimes difficult to determine whether an abstraction belongs to an IM or a DM.

IMs and DMs in the Real World

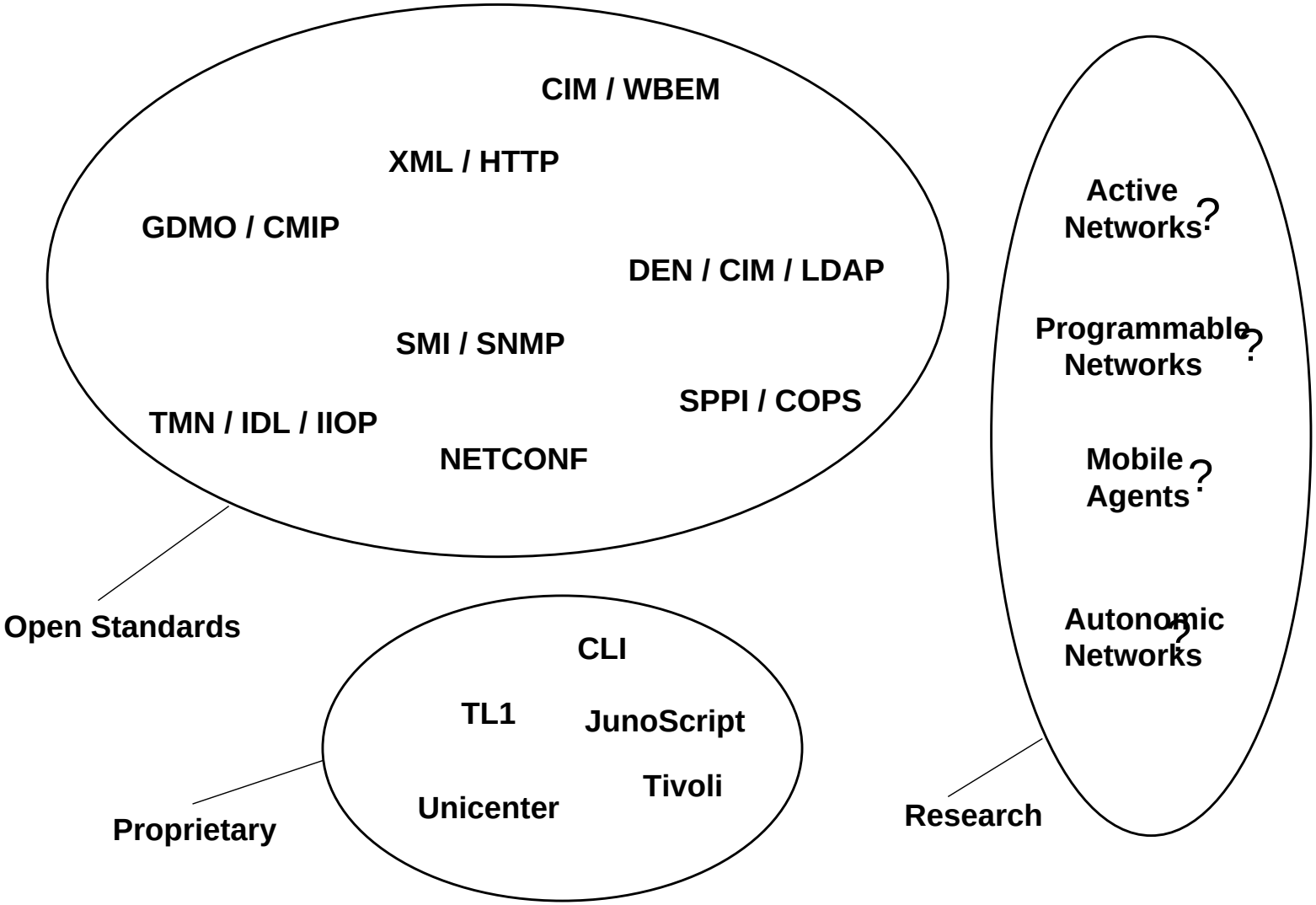


- The Architecture for Differentiated Services (RFC 2475) is an example for an informal definition of the DiffServ information model.
- The DiffServ MIB module (RFC 3289) and the DiffServ PIP module (RFC 3317) are examples of data models conforming to the DiffServ information model.

Network Management Standards



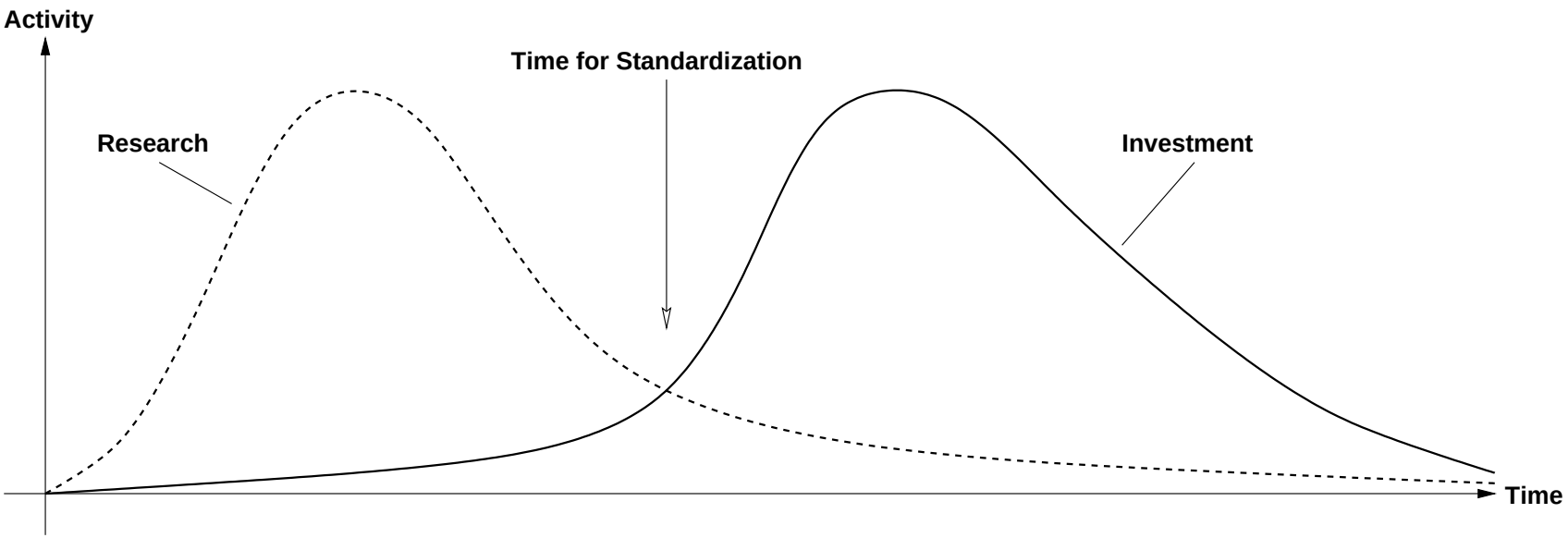
Management Technology Fragmentation



Implications

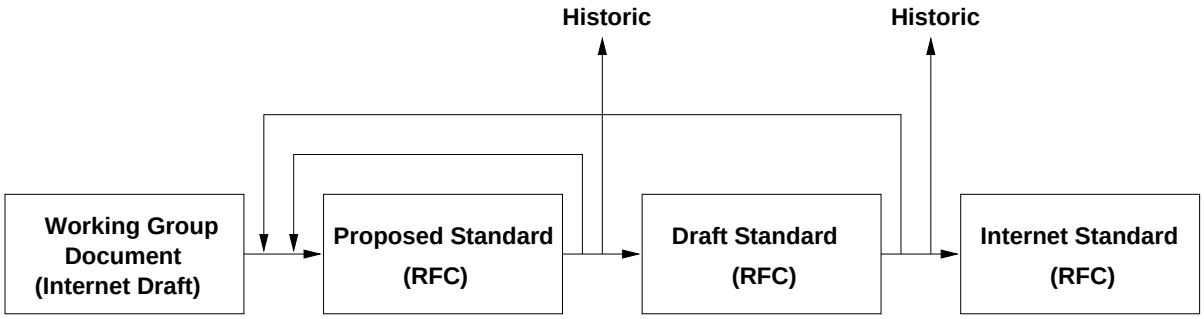
- Standards Organizations:
  - Duplication of efforts binds scare human resources.
- Network Device Vendors:
  - Customers force device vendors to support multiple management technologies, which makes devices unnecessarily complex and expensive.
- Management Application Vendors:
  - Integrated solutions are complex and thus expensive due to the many different interfaces.
- Network Operators:
  - Creating heterogeneous networks with common management interfaces is hard/expensive.
  - Increased costs and time for deploying new services.

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), but not all RFCs define standards (informational or experimental RFCs).
- The step from Proposed to Draft standard requires two independent and interoperable implementations from different code bases for all protocol features.
- The step from Draft to Standard requires significant implementation and operational experience.

IETF Management Standards

- The requirements for Internet management technologies have changed during the last decade.
- Some fundamental design decisions taken in the late 1980s must be revisited to better reflect today's realities.
- Working group members are dominated by network device vendors (solutions tend to be too device specific or way too detailed for real networks).
- Work on SNMP security took many many years to finally result in a stable SNMPv3 specification while other urgently needed SNMP improvements were kept on hold.
- Need to move to more mainstream technologies since network management remains a niche market.

IETF Standardization Principles

- KISS: Complex standards requiring people with special skills will not survive.
- Timeliness: Standards need to address real-world problems in a timely manner.
- Interoperability is more important than strict correctness. (Implementations should be liberal in what they accept and stringent in what they generate.)
- Protocols sometimes show effects when used on a larger scale that can not be observed on small scales.
- Concentration processes have given a few “big players” strong influence on the success of standards.

MIB Standardization Experience

- Standardizing MIBs in order to establish an open market between device vendors and management software vendors does not always work very well:
  1. Standardization takes too long.
  2. Consensus often on the lowest common denominator.
  3. Operationally important information often contained in proprietary MIB extensions.
  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.

References

[1] H. G. Hegering, S. Abeck, and B. Neumair. Integrated Management of Networked Systems. Morgan Kaufmann, 1999.

[2] ISO. Information processing systems – Open System Interconnection – Basic Reference Model – Part 4: Management Framework. International Standard ISO/IEC 7498-4, ISO, 1989.

[3] S. Bradner. The Internet Standards Process – Revision 3. RFC 2026, Harvard University, October 1996.

[4] B. Carpenter. Architectural Principles of the Internet. RFC 1958, IAB, June 1996.

[5] A. Pras and J. Schönwälder. On the Difference between Information Models and Data Models. RFC 3444, University of Twente, University of Osnabrueck, January 2003.

[6] A. Pras. Network Management Architectures. PhD thesis, Centre for Telematics and Information Technology, Twente University, Enschede, 1995.

[7] J. Schönwälder, A. Pras, and J. P. Martin-Flatin. On the Future of Internet Management Technologies. IEEE Communications Magazine, 41(10):90–97, October 2003.

Working Groups and Activities

MIB Module Review Guidelines

- All MIB modules published by the IETF go through a review process (so called MIB doctors).
- The MIB Review Guidelines draft documents some of the SMI folklore and the CLRs (crappy little rules or consistency language rules) that are checked during MIB reviews.
- MIB module authors are encouraged to check their MIBs against these rules before publishing them or submitting them to the IESG.
- A subset of the rules that can be automatically checked has been added to the `smilint` MIB module checker of the `libsmi` package.
- Guidelines document is of high quality and relatively stable. Should go to the IESG during this year.

IPv6 Support and Management

- MIB modules under revision:
  - TCs for Internet Network Addresses (RFC 4001)
  - IP-MIB (approved, publication pending)
  - IP-FORWARD-MIB (approved, publication pending)
  - TCP-MIB (RFC 4022)
  - UDP-MIB (approved, publication pending)
  - TUNNEL-MIB (approved, publication pending)
- IPv6 MIB modules published in 1998 will be made historic.
- Several other MIB modules are being updated to support IPv6 (e.g., OSPF, Radius)

Entity MIB Evolution

- The ENTITY-MIB models physical entities (e.g., fans, sensors, cpus, ports, modules, chassis) that make up a device.
- Represents the containment hierarchy of physical entities
- Very essential MIB module (comparable to the IF-MIB)
- Improvements made during the last months:
  - 3rd revision of the ENTITY-MIB (approved, publication pending)
  - ENTITY-SENSOR-MIB extension for sensors (RFC 3433)
  - MIB module providing state objects for physical entities (IESG)

Distributed Management

- Definition of a generic alarm reporting mechanism, based on ITU work in this space (X.733).
- Definition of an alarm reporting control interface, again based on some ITU work in this space (M.3100 Amendment 3).
- Revision of the remote operations modules (bug fixes, minimum compliance for support cable industry)
- Documents:
  - Alarm MIB (RFC 3877)
  - Alarm Reporting Control MIB (RFC 3878)
  - Ping, Traceroute, Lookup MIB Revision (IESG)

Middlebox Management

- A middlebox is a network intermediate device (NAT, firewall) that needs to be configured in order to make applications work (“drilling holes into middleboxes”).
- Middlebox Communication Architecture and Framework (RFC 3303)
- Middlebox Communications Protocol Requirements (RFC 3304)
- Middlebox Communications Protocol Evaluation
- Middlebox Communications Protocol Semantics (RFC 3989)
- Middlebox Communications Protocol Managed Objects Analysis
- Middlebox Communication MIB Module (IESG)

IEEE 802 Management

- Bridge MIB has been revised and is waiting for publication.
- Rapid spanning tree MIB has been revised and waiting for approval.
- Future MIB work related to IEEE 802 standards will be done by the IEEE with initial MIB review support by the IETF.

ATM / MPLS / xDSL / Cable Modems

- Several working groups produce a stream of interface type specific MIB modules.
- Optical Interface Type (RFC 3591)
- SONET/SDH MIB Revision (RFC 3592)
- Supplemental ATM Interface MIB (RFC 3606)
- DS1 / E1 / DS2 / E2 Interface Type MIB (RFC 3895)
- DS3 / E3 Interface Type MIB (RFC 3896)
- Very High Speed Digital Subscriber Lines (VDSL) MIB (RFC 3728)
- VDSL SCM Line Coding MIB (RFC 4069)
- VDSL MCM Line Coding MIB (RFC 4070)
- Many MIB modules related to Cable Modems.
- Many MPLS MIB modules (probably need to get

Printer, Fiber Channel, iSCSI, Remote Monitoring

- Printer MIB (RFC 3805)
- Finisher MIB (RFC 3806)
- Fiber Channel MIB modules are being revised / extended
- iSCSI MIB modules are currently being defined
- RMON Overview Document (RFC 3577)
- Application Performance Measurement MIB (RFC 3729)
- Transport Performance Metrics MIB
- Synthetic Sources for Performance Monitoring Algorithms MIB
- Real-time Application Quality of Service Monitoring MIBs

Extensible Provisioning Protocol

- Application layer client-server protocol for the provisioning and management of objects stored in a shared central repository
- Target application area is automated interaction with registries
- Extensible Provisioning Protocol Features
  - XML based protocol (commands / responses)
  - Session management commands (login, logout)
  - Query commands (check, info, poll, transfer)
  - Object transform commands (create, delete, renew, transfer, update)
  - Mapping over TCP and BEEP defined

Internet Registry Information Service

- Application layer query response protocol to access information services provided by Internet registries.
- Replacement for the WHOIS protocol (RFC 3912)
- Recent documents:
  - IRIS Core Protocol (RFC 3981)
  - IRIS Domain Registry Type (RFC 3982)
  - IRIS over BEEP (RFC 3983)
- Support for additional registry types and transport mappings under development.

References

[1] C. M. Heard. Guidelines for Authors and Reviewers of MIB Documents. Internet Draft draft-ietf-ops-mib-review-guidelines-04.txt, Consultant, February 2005.

[2] A. Newton. Cross Registry Internet Service Protocol (CRISP) Requirements. RFC 3707, VeriSign, February 2004.

[3] A. Newton and M. Sanz. IRIS: The Internet Registry Information Service (IRIS) Core Protocol. RFC 3981, VeriSign, DENIC, January 2005.

[4] A. Newton and M. Sanz. IRIS: A Domain Registry (dreg) Type for the Internet Registry Information Service (IRIS). RFC 3982, VeriSign, DENIC, January 2005.

[5] S. Hollenbeck. Extensible Provisioning Protocol (EPP). RFC 3730, VeriSign, March 2004.

[6] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Domain Name Mapping. RFC 3731, VeriSign, March 2004.

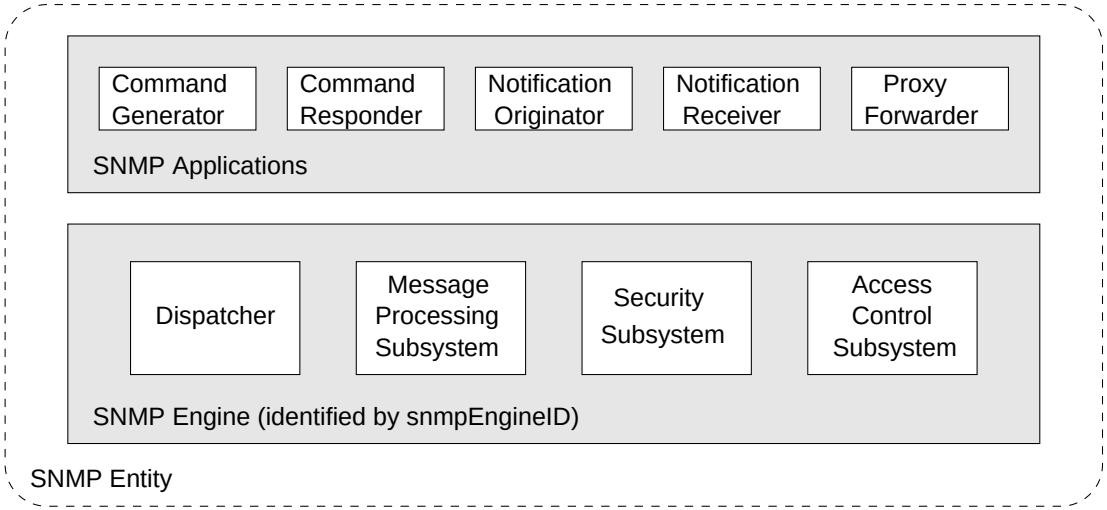
[7] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Transport Over TCP. RFC 3734, VeriSign, March 2004.

SNMP Version 3

SNMP Version 3

- → Architectural Concepts
- → Protocol Operations
- → Message Format
- → Authentication and Privacy
- → Authorization and Access Control
- → Remote Configuration
- → Status and Limitations

Architectural Concepts (RFC 3411)

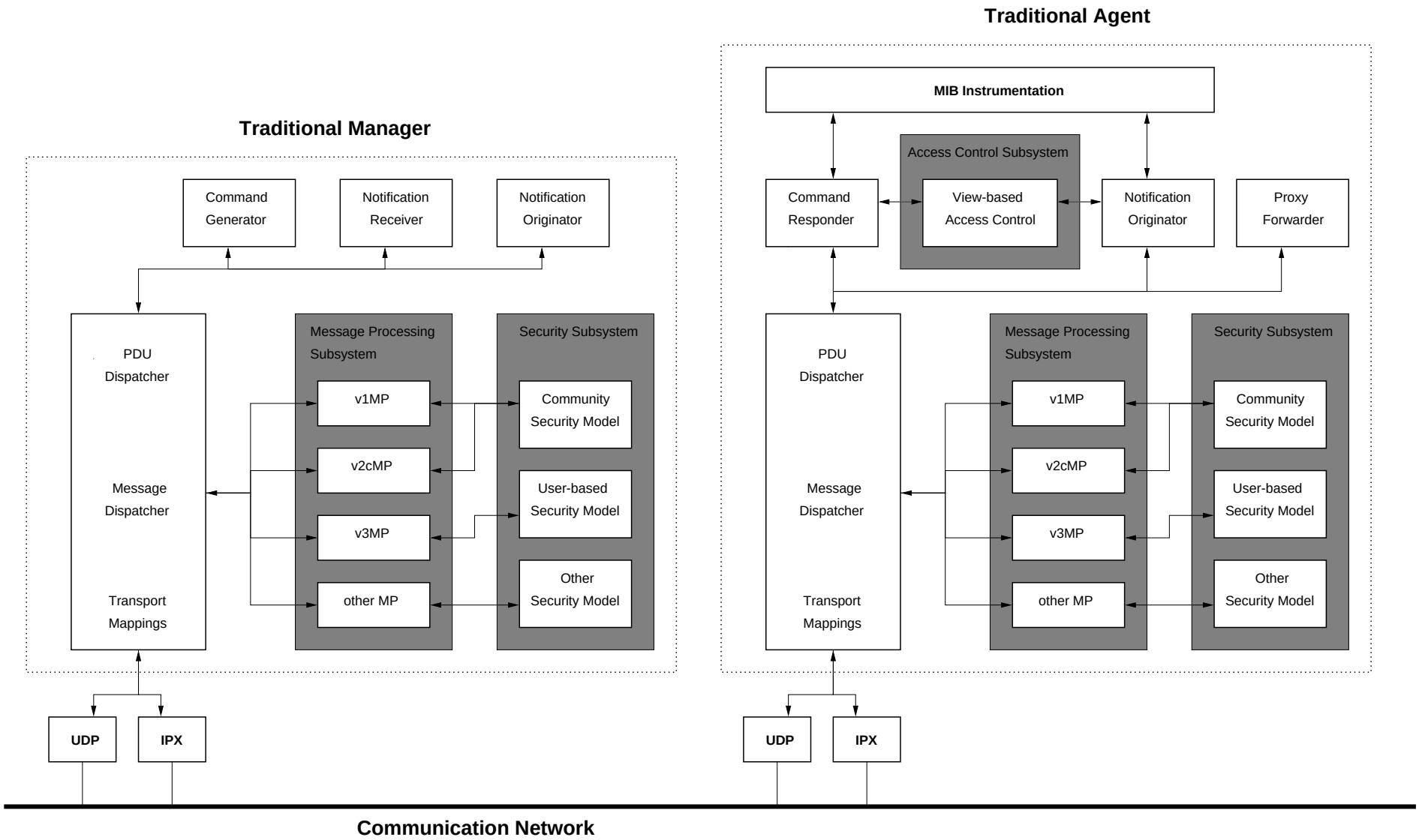


- Fine grained SNMP applications instead of coarse grained agents and managers.
- Exactly on engine per SNMP entity and exactly one dispatcher per SNMP engine.
- Every abstract subsystem may have of one or more concrete models.
- Modularization enables incremental enhancements.

SNMP Contexts

- An context is a collection of management information accessible by an SNMP entity.
  - SNMP entities may have access to multiple contexts.
  - Identical 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., “800007e580e16e1566696f7440”)
  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



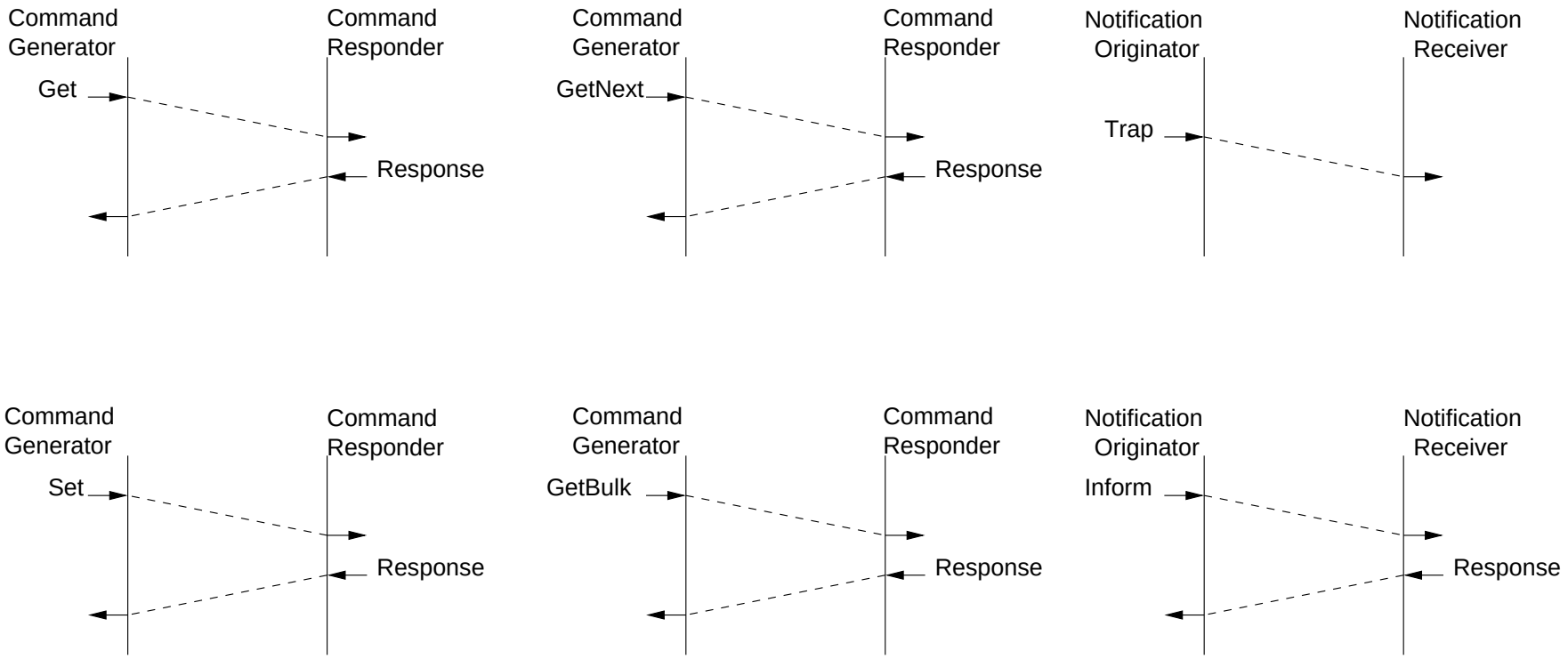
SNMPv3/USM Textual Conventions

- `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`.

SNMPv3/USM Textual Conventions

- `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.

Protocol Operations (RFC 3416)



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

Lexicographic Ordering

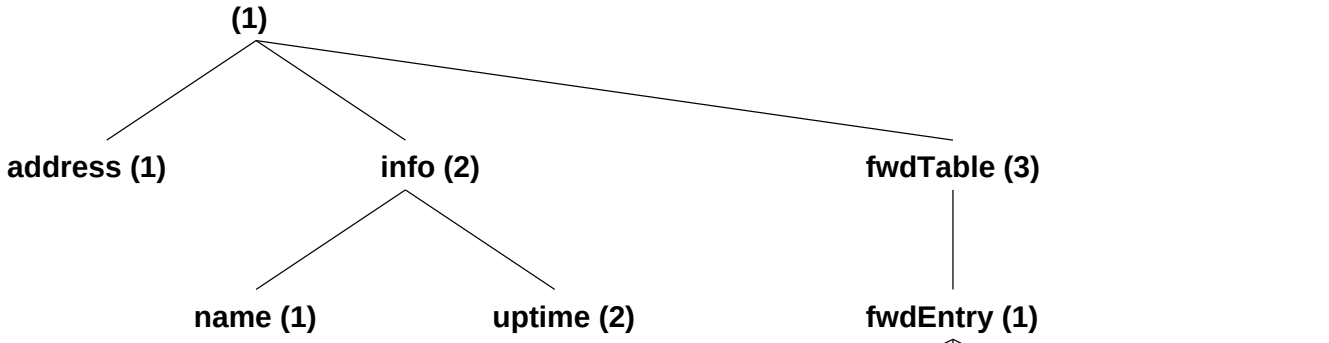
- Given are two vectors of natural numbers $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$ with $n \leq m$. We say that x is lexicographically less than y if and only if one of the following conditions is true:

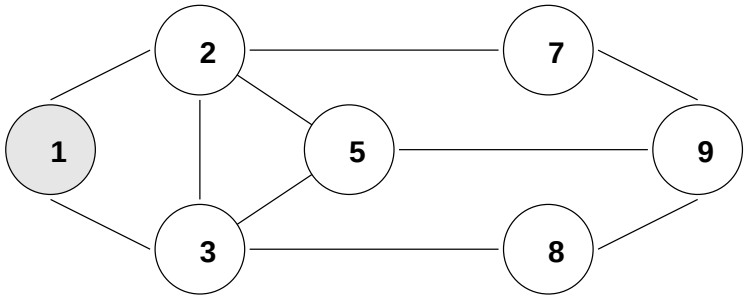
  (a) $x_j = y_j$ for $1 \leq j \leq k$ and $x_k < y_k$ with $k \leq n$ and $k \leq m$

  (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.

Simple Forwarding Table Example





fwdIndex (1)	fwdDest (2)	fwdNext (3)
1	2	2
2	3	3
3	5	2
4	7	2
5	8	3
6	9	3

Lexicographic Ordering Example

- Lexicographic ordered list of MIB instances:

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.

PDU Processing Errors

- An error response signals the complete failure of the corresponding request.
- An error response contains an *error status* (numeric error code) and an *error index* (position in the variable list where the error occured).
- Error responses contain no useful management information.
- There is only a single error status and error index even if there are multiple errors.
- An error in general implies that none of the actions has taken place during a write operation (as if simultaneous writes).

PDU 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)

PDU Error Codes (RFC 3416)

SNMPv3 Error Code	Get/GetNext/GetBulk	Set	Trap/Inform	SNMPv1 Error Code
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)

PDU Processing Exceptions

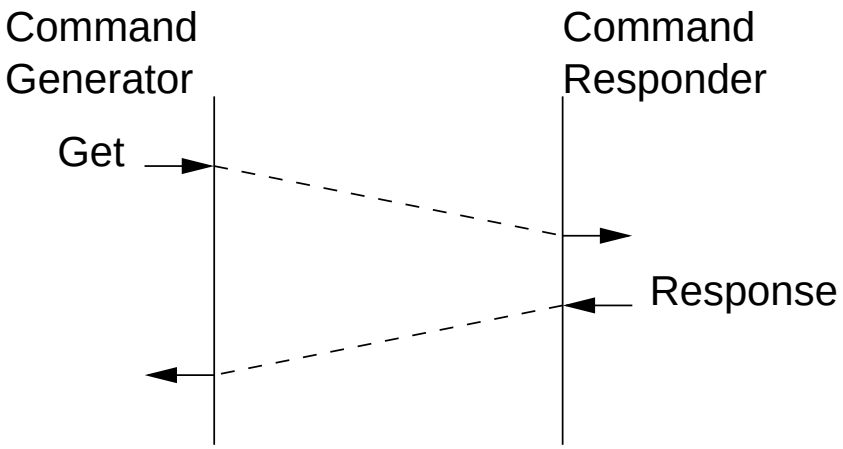
- A response can contain per variable binding exceptions.
- One or more exceptions in a response are not considered to be an error condition of the corresponding request.
- A response with exceptions still contains useful management information.
- Applications receiving response messages
  - must check the error code,
  - must detect exceptions, and
  - they must deal with them gracefully.
- Not all applications get this right...

PDU Exceptions (RFC 3416)

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

- 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.

Get Operation (RFC 3416)

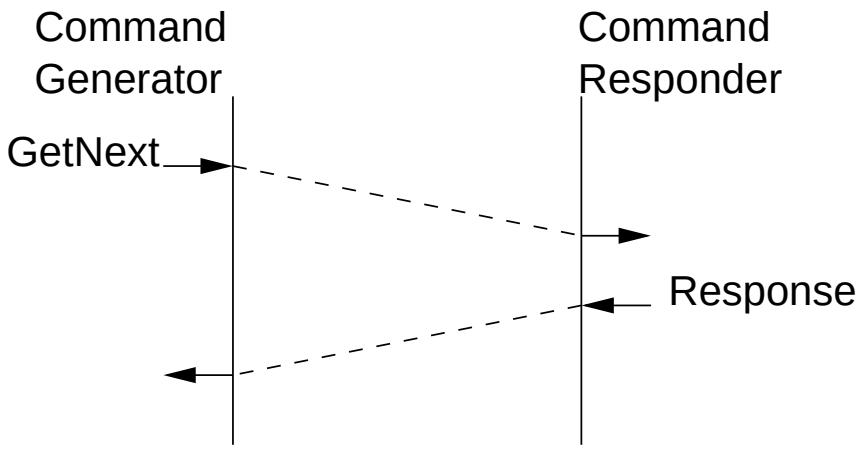


- The `Get` operation is used to read one or more MIB variables.
- Possible error codes: `tooBig`, `genErr`
- Possible exceptions: `noSuchObject`, `noSuchInstance`

Example 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)
```

GetNext Operation (RFC 3416)

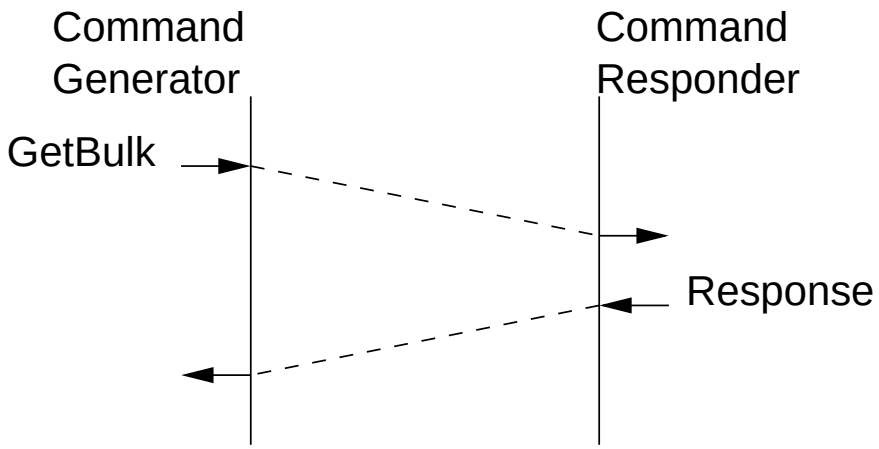


- The `GetNext` operation allows to retrieve the value of the next existing MIB instances in lexicographic order.
- Successive `GetNext` operations can be used to walk the MIB instances without prior knowledge about the MIB structure.
- Possible error codes: `tooBig`, `genErr`
- Possible exceptions: `endOfMibView`

Example 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)
```

GetBulk Operation (RFC 3416)



- The `GetBulk` operation is a generalization of the `GetNext` operation where the agent performs a series of `GetNext` operations internally.
- The `GetBulk` operation like all the other protocol operations operates only on the lexicographically ordered list of MIB instances and does therefore not respect conceptual table boundaries.

GetBulk Operation (RFC 3416)

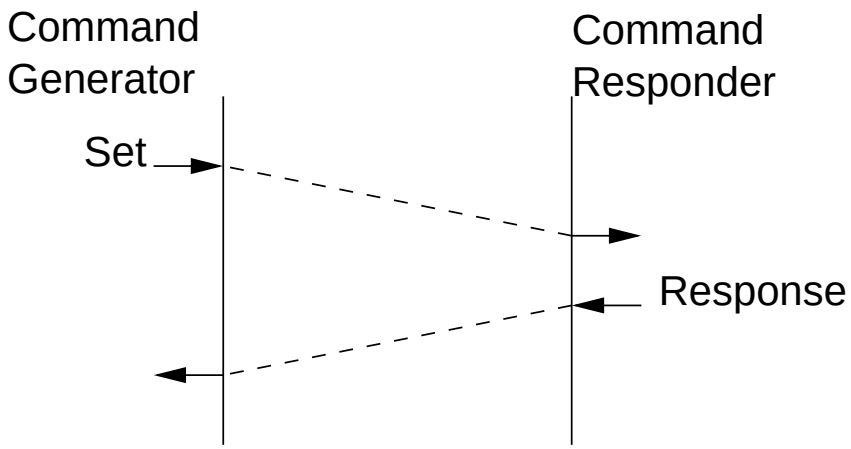
- GetBulk processing details:
  - The first N elements (`non-repeaters`) of the varbind list will be processed similar to the `GetNext` operation.
  - The remaining R elements of the varbind list are repeatedly processed similar to the `GetNext` operation.
  - The parameter M (`max-repetitions`) defines the upper bound of repetitions.
- The manager usually does not know how to choose a value for `max-repetitions`.
- If `max-repetitions` is too small, the potential gain will be small. If it is too large, there might be a costly overshoot.

Example 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 `non-repeaters` are typically used to retrieve a discontinuity indicating scalars, such as `sysUpTime.0`.
- Any ideas for a better GetBulk operation?

Set Operation (RFC 3416)



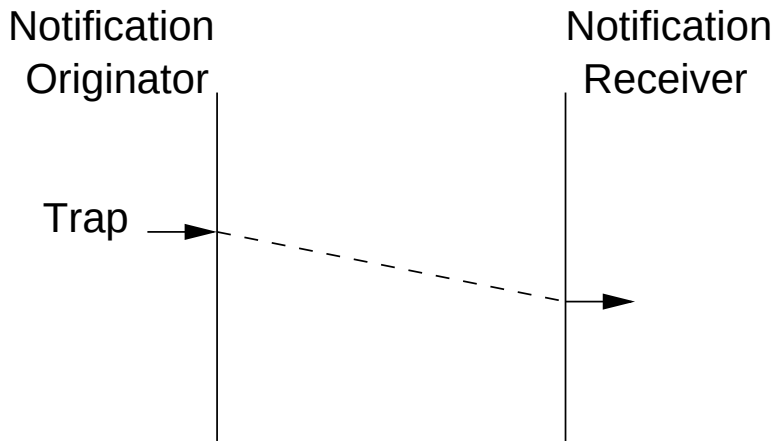
- The `Set` operation allows to modify a set of MIB instances. The operation is atomic (either all instances are modified or none).
- Possible error codes: `wrongValue`, `wrongEncoding`, `wrongType`, `wrongLength`, `inconsistentValue`, `noAccess`, `notWritable`, `noCreation`, `inconsistentName`, `resourceUnavailable`, `commitFailed`, `undoFailed`

Example 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)
```

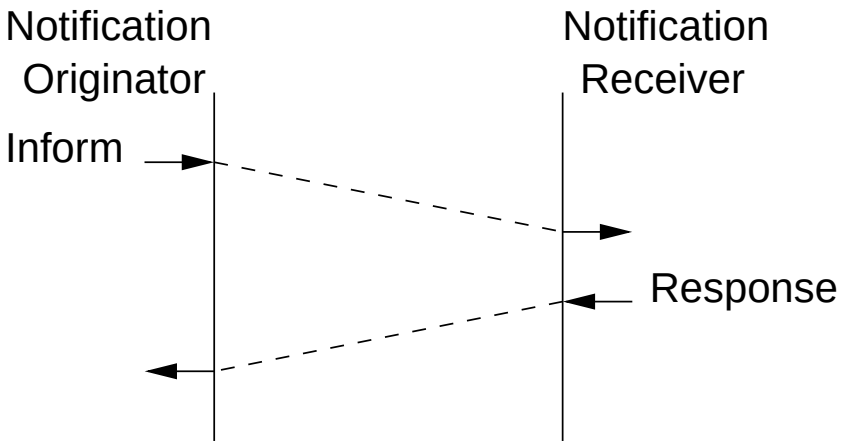
- The error codes `authorizationError` and `readOnly` are not used.
- No support for object type specific error codes.

Trap Operation (RFC 3416)



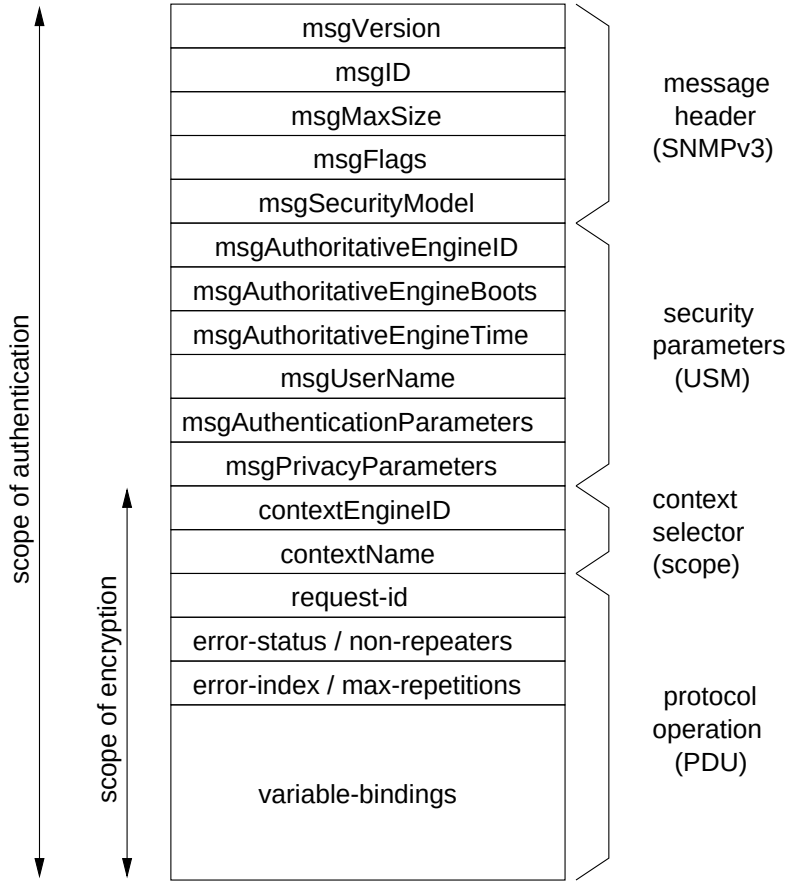
- The `Trap` operation is used to notify a manager of the occurance of an event.
- The `Trap` operation is unconfirmed: The sending agent does not know whether the trap was received and processed by a manager.
- All trap specific information in encoded in the varbind list (`sysUpTime`, `snmpTrapOID`, `snmpTrapEnterprise`).

Inform Operation (RFC 3416)



- The `Inform` operation is a confirmed trap.
- 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 message from the notification receiver.
- Confirmation indicates that the notification was delivered, not that the notification was understood.

Message Format (RFC 3412, RFC 3414)



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

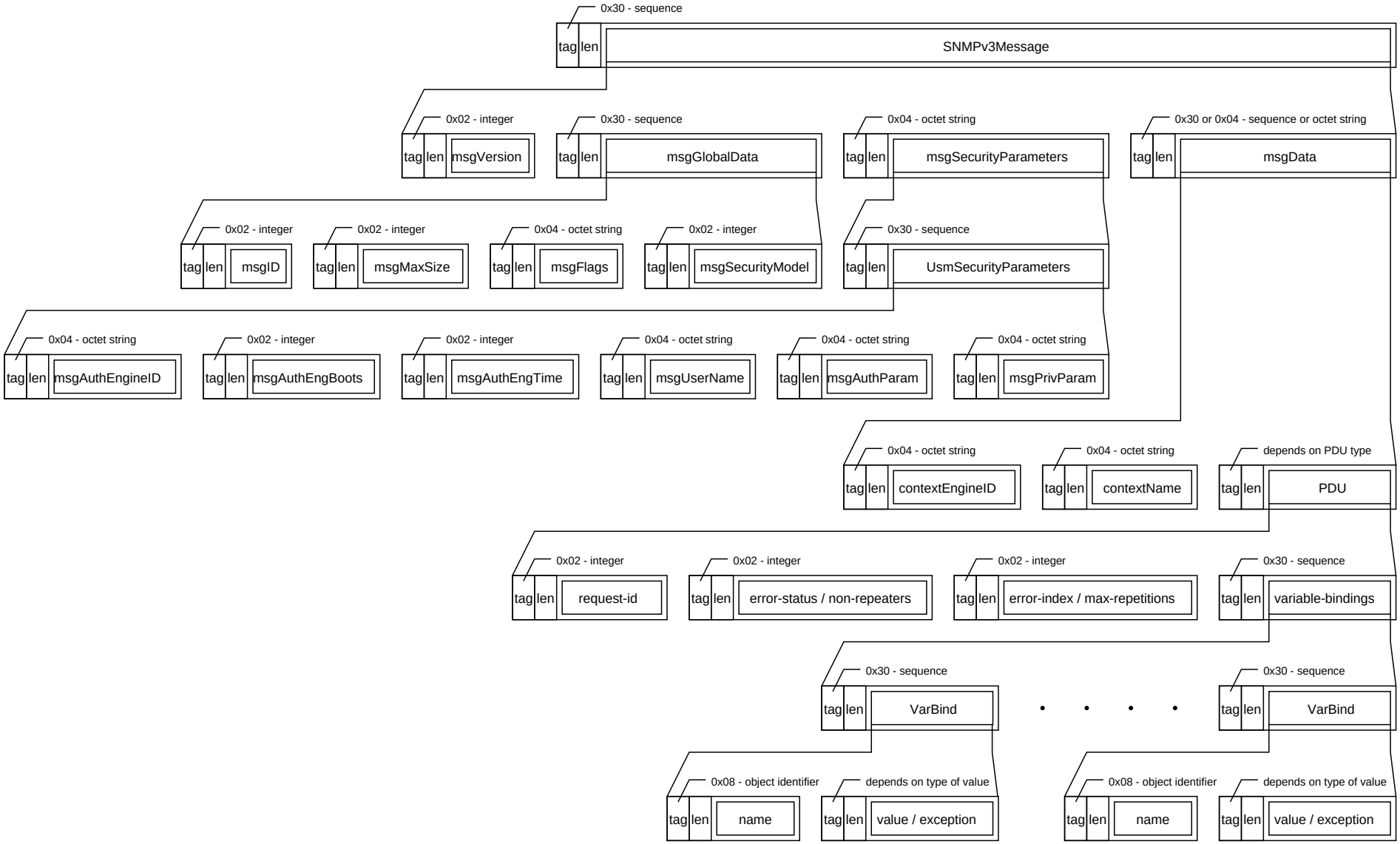
Classes of Protocol Operations (RFC 3411

- 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.

- PDU classes support the introduction of new protocol operations without changes the core specifications.
- However, no indication of the set of protocol operations supported by an SNMP engine implementation.

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).

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.)

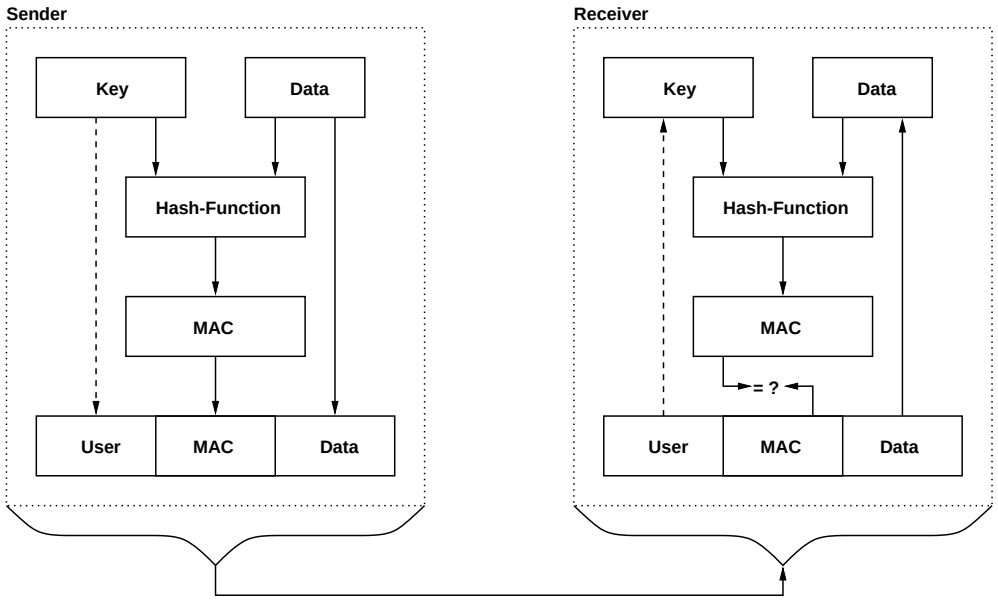
USM Security Services (RFC 3414)

- 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.

USM Security Services (RFC 3414)

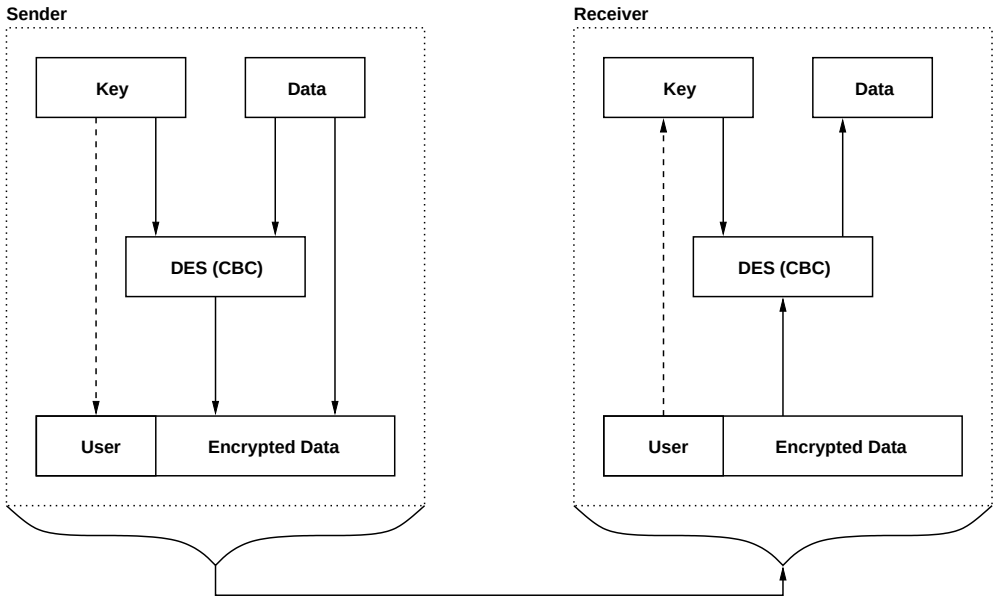
- Message Timeliness and Limited Replay Protection
  - A message whose generation time is outside of a time window is not accepted.
  - Message reordering is not dealt with and can occur in normal conditions too.
- No protection against Denial of Service attacks
  - Too hard of a problem to solve.
- No protection against Traffic Analysis attacks
  - Many management traffic patterns are predictable.
  - Hiding periodic management traffic would be extremly costly.

Data Integrity and Data Origin Authentication



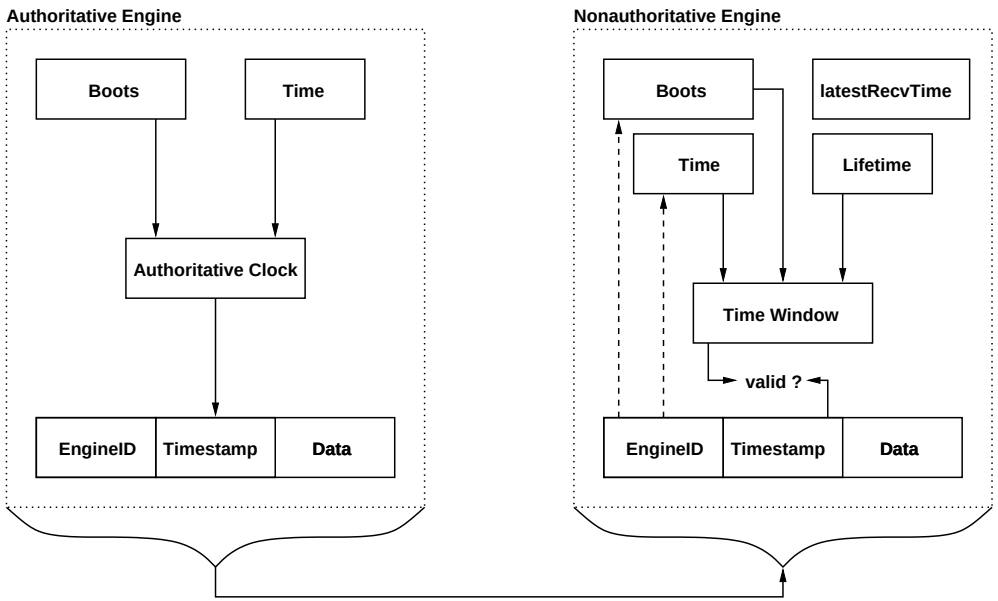
- Cryptographic strong oneway hash functions generate message authentication codes (MACs).
- The MAC ensures integrity, the symmetric key provides for authentication.
- USM currently uses HMAC-MD5-96 or HMAC-SHA-96.
- Other hash functions may be added in the future.

Data Confidentiality



- Optional encryption of the `ScopedPDU` using symmetric but localized keys.
- USM currently uses CBC-DES.
- Other encryption functions may be added in the future.
- Encryption is CPU expensive — use only when needed.

Message Timeliness and Replay Protection



- A non-authoritative engine maintains a notion of the time at the authoritative engine.
- A non-authoritative engine 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.

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 $K_U = SHA(S)$.
- Slows down naive brute force password attacks.
- No serious barrier for an attacker with a transformed dictionary.

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.
- Very important in environments where devices can easily be stolen or accessed physically by attackers.

Key Changes

- 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 $d \oplus \delta = d \oplus (d \oplus K_{new}) = K_{new}$.

Key Change Properties

- Key changes must be possible without encryption since encryption is optional.
- An attacker who is able to catch all key updates can calculate the current keys once an old key has been broken.
- Attackers thus get an unlimited amount of time to break keys if they can catch all key change requests.

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

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.
- The determination whether a message is recent is made relative to the authoritative engine.

Timeliness Checks (Authoritative Receiver)

- A message is outside the time window if any of the following holds true:
  1. `snmpEngineBoots` $= 2^{31} - 1$
  2. `msgAuthoritativeEngineBoots` $\neq$ `snmpEngineBoots`
  3. $abs($`msgAuthoritativeEngineTime` $-$ `snmpEngineTime`$) > 150$ seconds

Timeliness Checks (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

Clock Synchronization

- 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`

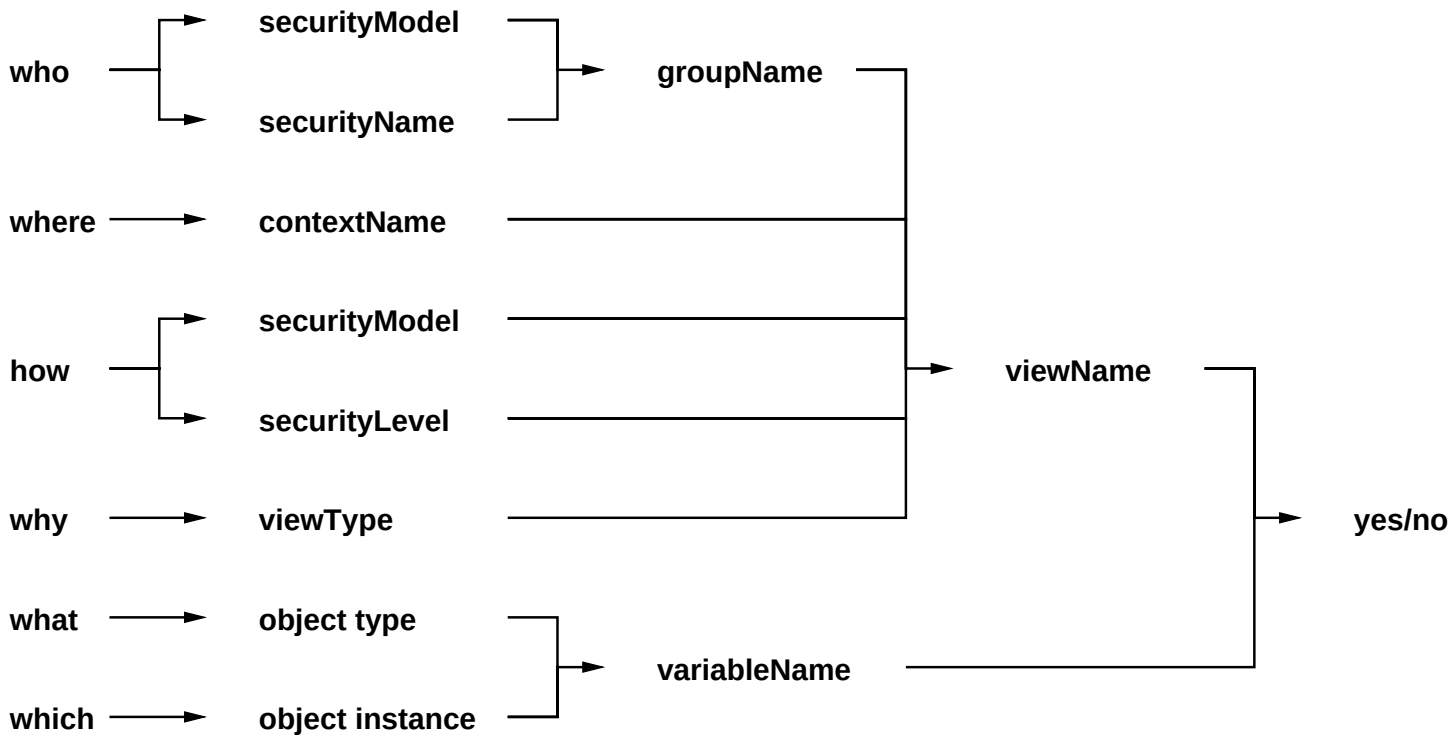
Discovery and Initial Synchronization

- 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)

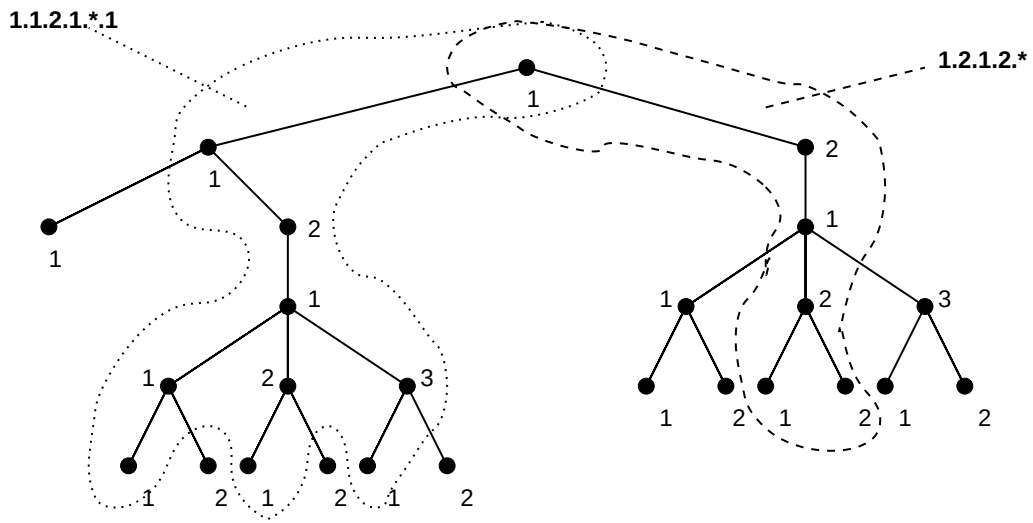
- The `usmUserTable` maps USM user names to `securityNames`.
- New entries may be created by cloning existing entries (together with their keys).
- The `usmUserAuthKeyChange` and `usmUserPrivKeyChange` objects may be used by the security administrator to change the user's keys.
- The `usmUserOwnAuthKeyChange`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

Authorization and Access Control (RFC 3415)



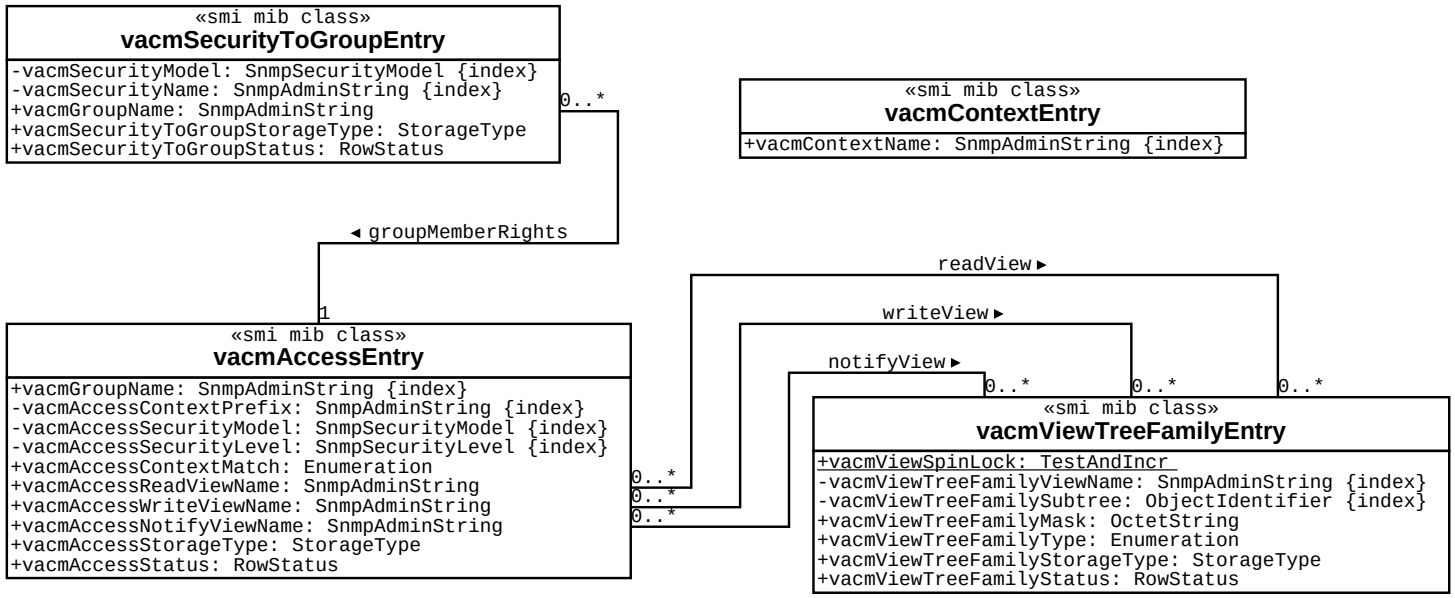
- Three different `securityLevels`: `noAuthNoPriv`, `authNoPriv`, `authPriv`
- A `securityName` is a security model independent name for a principal.

View-based Access Control (RFC 3415)



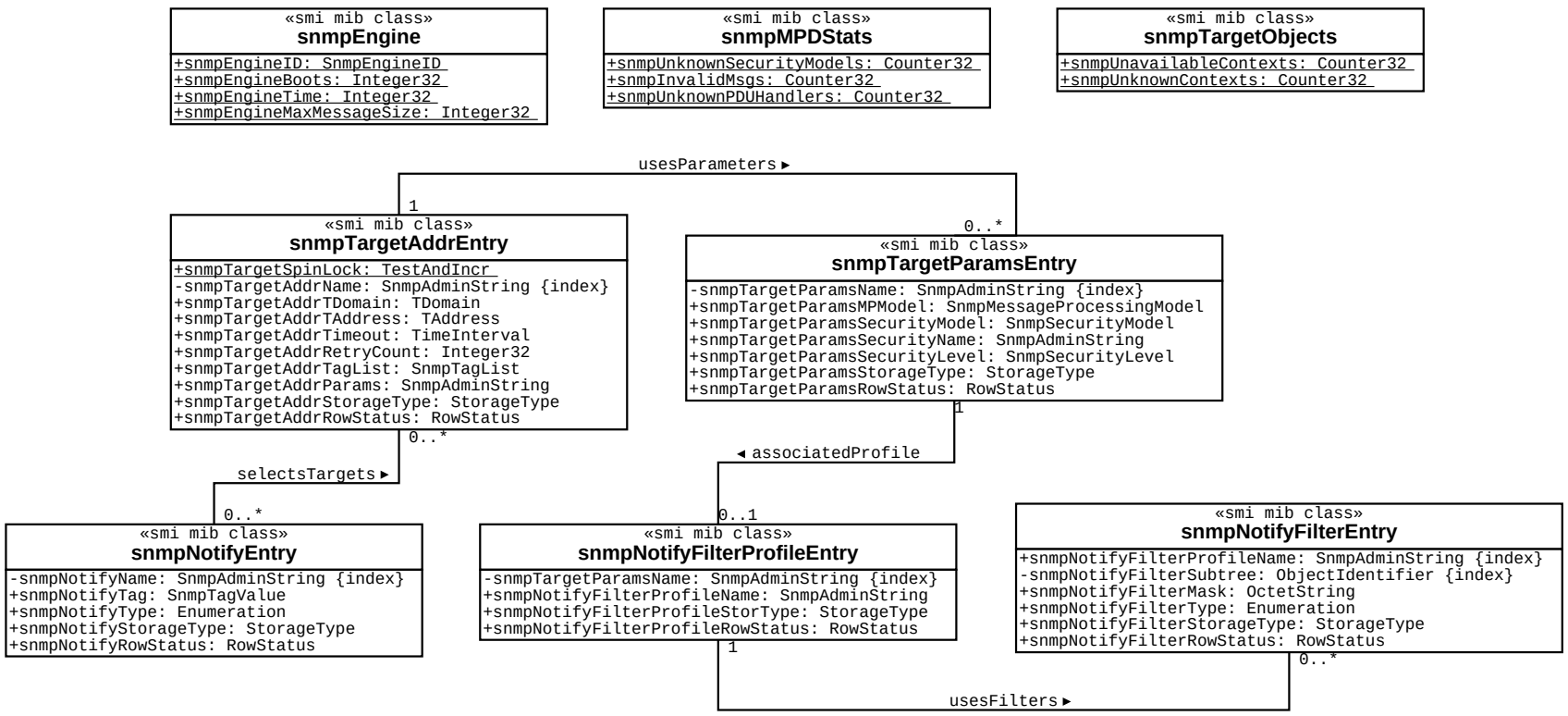
- A view subtree is a set of managed object instances with a common OID prefix.
- A view tree family is the combination of an OID prefix with a bit mask (wildcarding of OID prefix components).
- A view is an ordered set of view tree families.
- Access control rights are defined by a read view, write 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.

Remote Configuration (RFC 3413)



- SNMPv3 defines several MIB modules for remote configuration of SNMP entities.

SNMPv3 Status and Limitations

- Many implementations and products are available.
- Visit the SNMPv3 Web page for up-to-date information.

  `<http://www.ibr.cs.tu-bs.de/ietf/snmpv3/>`
- Some technology domains (e.g., cable modem industry in the US) require SNMPv3 support.
- However, general deployment happens much slower than originally expected.
- Manual configuration is an error prone and time consuming.
- Lack of integration in deployed AAA systems.
- Remote configuration and key management requires nontrivial applications.

SNMPv3 Status and Limitations

- Missing extensibility for new base data types (e.g., Unsigned64).
- Missing extensibility for new protocol operations (e.g., `GetRange`).
- 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 (no standardized Diffie-Helman exchange, no integration with other key management systems).

References

[1] J. Case, R. Mundy, D. Partain, and B. Steward. Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410, SNMP Research, Network Associates Laboratories, Ericsson, December 2002.

[2] W. Stallings. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. Addison-Wesley, 3 edition, 1999.

[3] D. Zeltserman. A Practical Guide to SNMPv3 and Network Management. Prentice Hall, 1999.

[4] U. Blumenthal and B. Wijnen. Security Features of SNMPv3. Simple Times, 5(1), December 1997.

Integrated Security Models

What is wrong with USM?

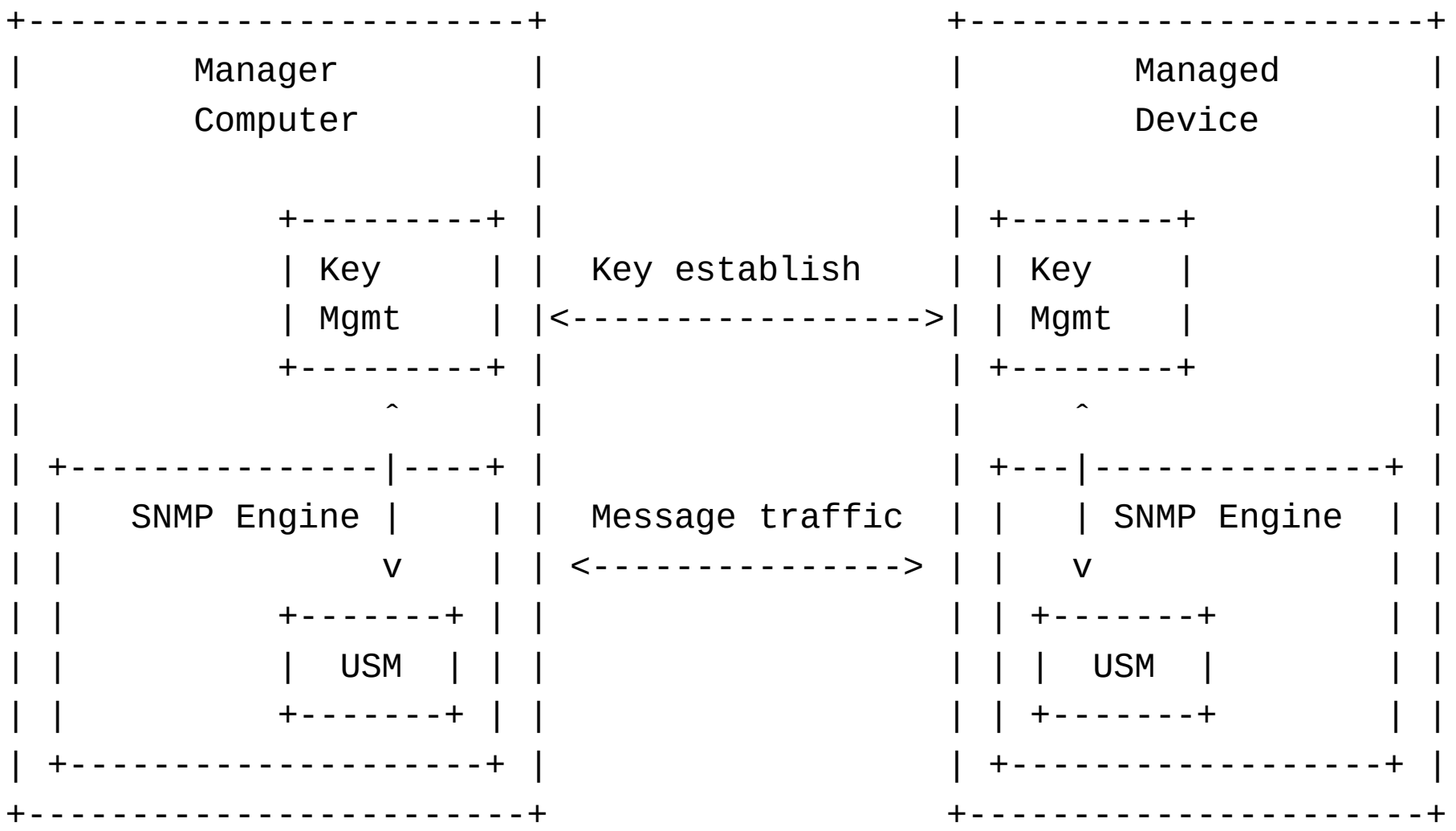
- The SNMP USM security model and VACM access control model are self-contained (following the original SNMP design goals).
- They do not integrate well into deployed authentication and authorization infrastructures.
- Operators prefer to keep the number of authentication and authorization systems that must be managed to a minimum.
- SNMPv3 deployment and especially key and access control management therefore introduces high costs for operators.

$\Longrightarrow$ Slow deployment of SNMPv3.

ISMS Requirements

- Must be at least as secure as USM.
- Must not preclude the use of USM, particularly if network instability could cause problems for the proposed solution.
- Must be able to work with VACM.
- The protocol itself should support multiple security infrastructures, but an implementation may support some subset of these.
- Must not break basic device discovery. (Retaining USM support would satisfy this goal.)

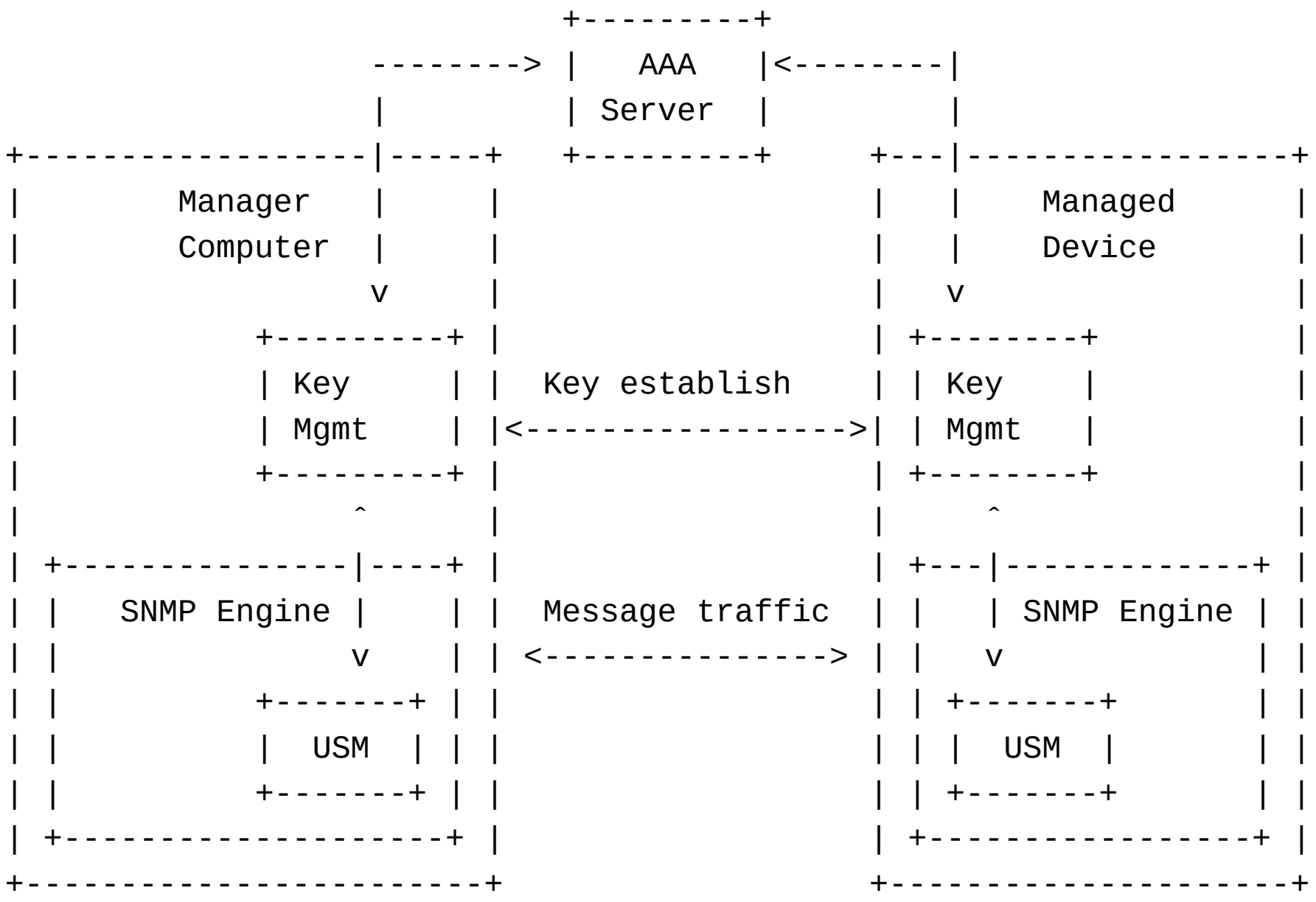
External User Security Model (EUSM)



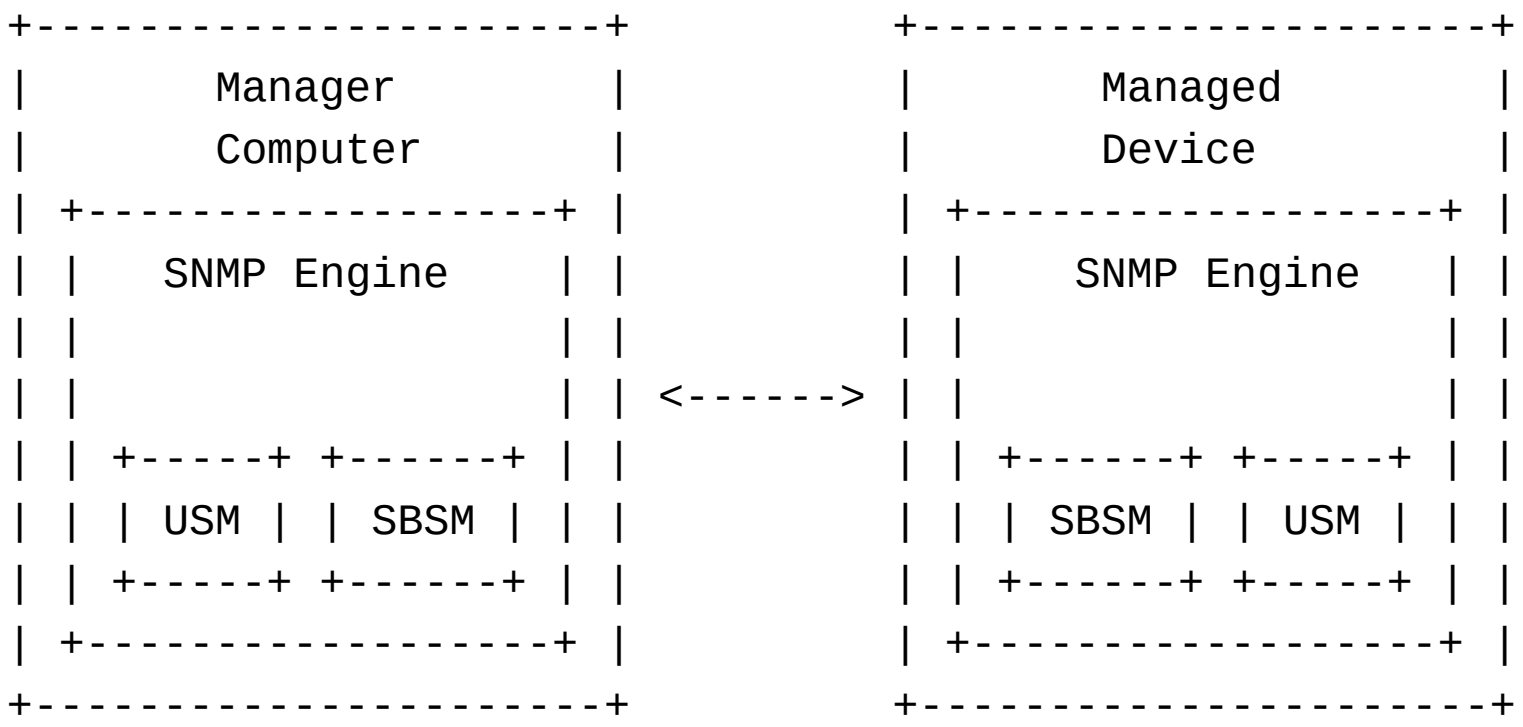
EUSM Properties

- Replaces USM's key management but leaves USM transport alone.
- Assumes that an external key management process will be co-resident with SNMP engines, and will install the keys, as with IKE/IPsec.
- Originally proposed to use the Extensible Authentication Protocol (EAP) to install keys.
- Can be integrated with AAA systems (Radius, Diameter)

EUSM AAA Integration



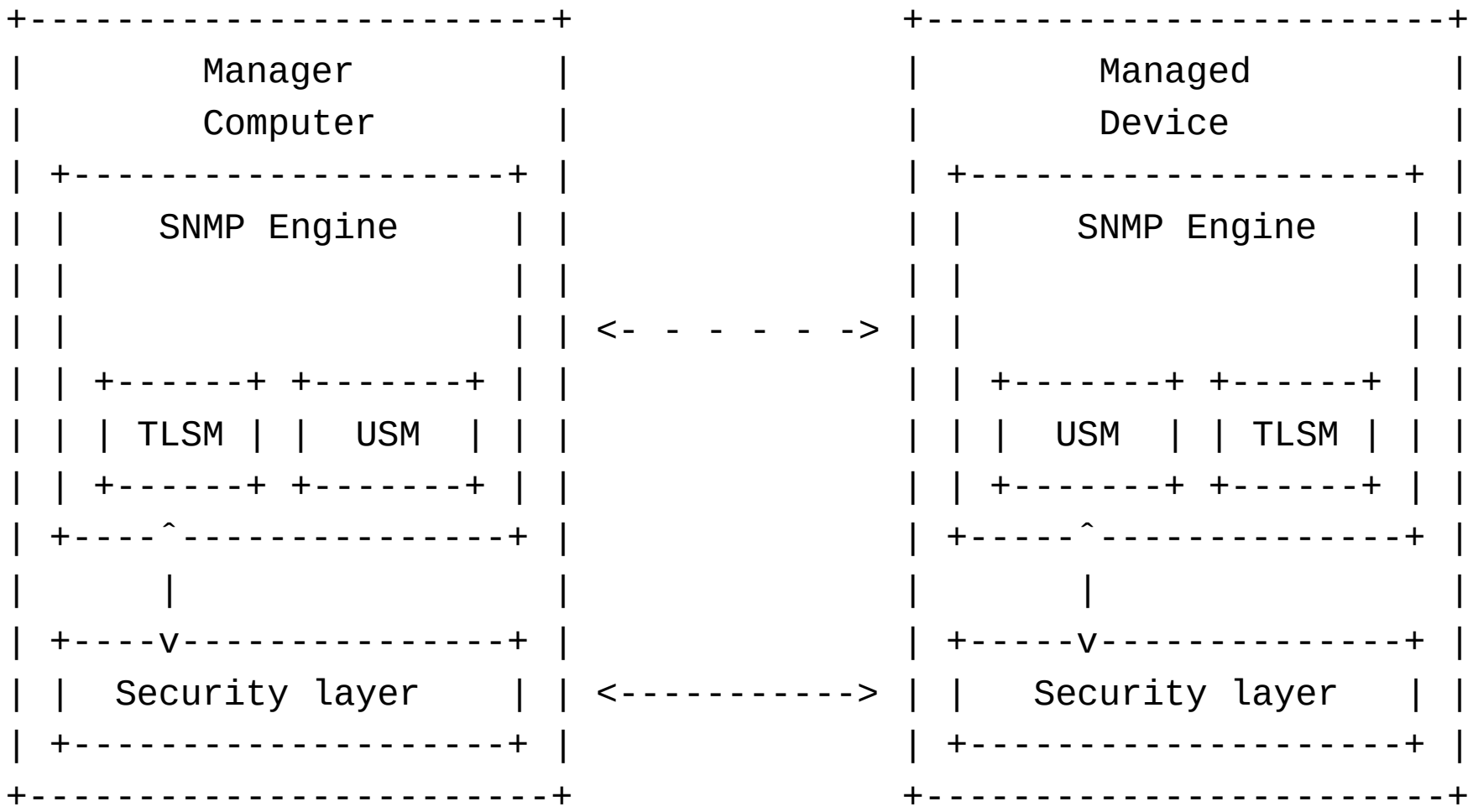
Session-Based Security Model (SBSM)



SBSM Properties

- SBSM is a new security model replacing USM entirely.
- Integrated session establishment and messaging protocol.
- Tight coupling between security system and the rest of the SNMP implementation
- Completely new security protocol requires careful evaluation

Transport-Layer Security Model (TLSM)



TLSM Properties

- Reuses standard security protocols (e.g., TLS or SSH)
- TLSM security model is a shim to provide required information (e.g., snmp security name and security level)
- Implies the usage of TCP (unless DTLS becomes a success).
- Weak coupling between user authentication and security layer.
- Scalability concerns wrt. TCP-based transports.

Evaluation and Status

- Evaluation team recommended to adopt EUSM
- Security ADs announced that neither EAP nor IKE are suitable key management protocols
- WG discussions lead to a very rough concensus towards TLSM
- Attend the 63rd IETF meeting in Paris to see how the story continues

References

[1] U. Blumenthal, L. Dondeti, R. Presuhn, and E. Rescorla. Comparison of Proposals for Integrated Security Models for SNMP (Simple Network Management Protocol). Internet Draft draft-ietf-isms-proposal-comparison-00.txt, Intel, Nortel, Consultant, RTFM, February 2005.

[2] D. Harrington and J. Schönwälder. Transport Mapping Security Model (TMSM) for the Simple Network Management Protocol version 3 (SNMPv3). Internet Draft draft-schoenw-snmp-tlsm-01.txt, Independent, IU Bremen, October 2004.

[3] K. Narayan, K. McCloghrie, and J. Salowey. External User Security Model (EUSM) for version 3 of the Simple Network Management Protocol (SNMPv3). Internet Draft draft-kaushik-snmp-external-usm-02.txt, Cisco Systems, February 2005.

[4] W. Hardacker and D. Perkins. A Session-Based Security Model (SBSM) for version 3 of the Simple Network Management Protocol (SNMPv3). Internet Draft draft-hardaker-snmp-session-sm-03.txt, Sparta, SNMPInfo, October 2004.

Evolutionary Research

Evolutionary Research

- → Next Generation Structure of Management Information (SMIng)
- → SNMP over TCP
- → SNMP Payload Compression
- → Extended SNMP Protocol Operations
- → AES Cipher Algorithm for USM
- → SNMP Uniform Resource Locators
- → Session-Based SNMP Security Model

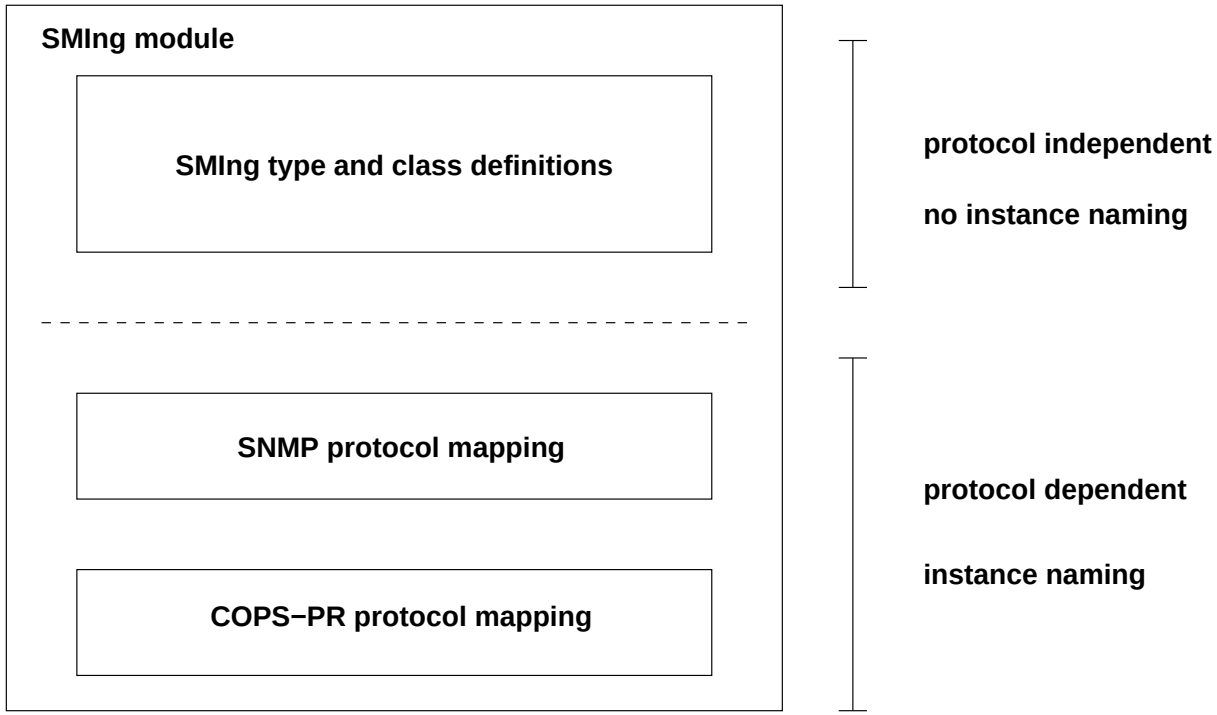
SMIv2 Limitations and Problems

- SMIv2 misses some important base types such as 64 bit numbers.
- SMIv2 lacks reusable compound 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.

SMIng Approach

- Next generation data modeling language called SMI (SMIng)
- History of SMIng:
  - Research project at TU Braunschweig (1999-2000)
  - Network Management Research Group (2000)
  - SMIng Working Group (2000-2003)
  - Network Management Research Group (2003-2004)
- Detailed objectives are documented in RFC 3216.
- Published as Experimental RFCs (RFC 3780, 3781).

SMIng Module Structure

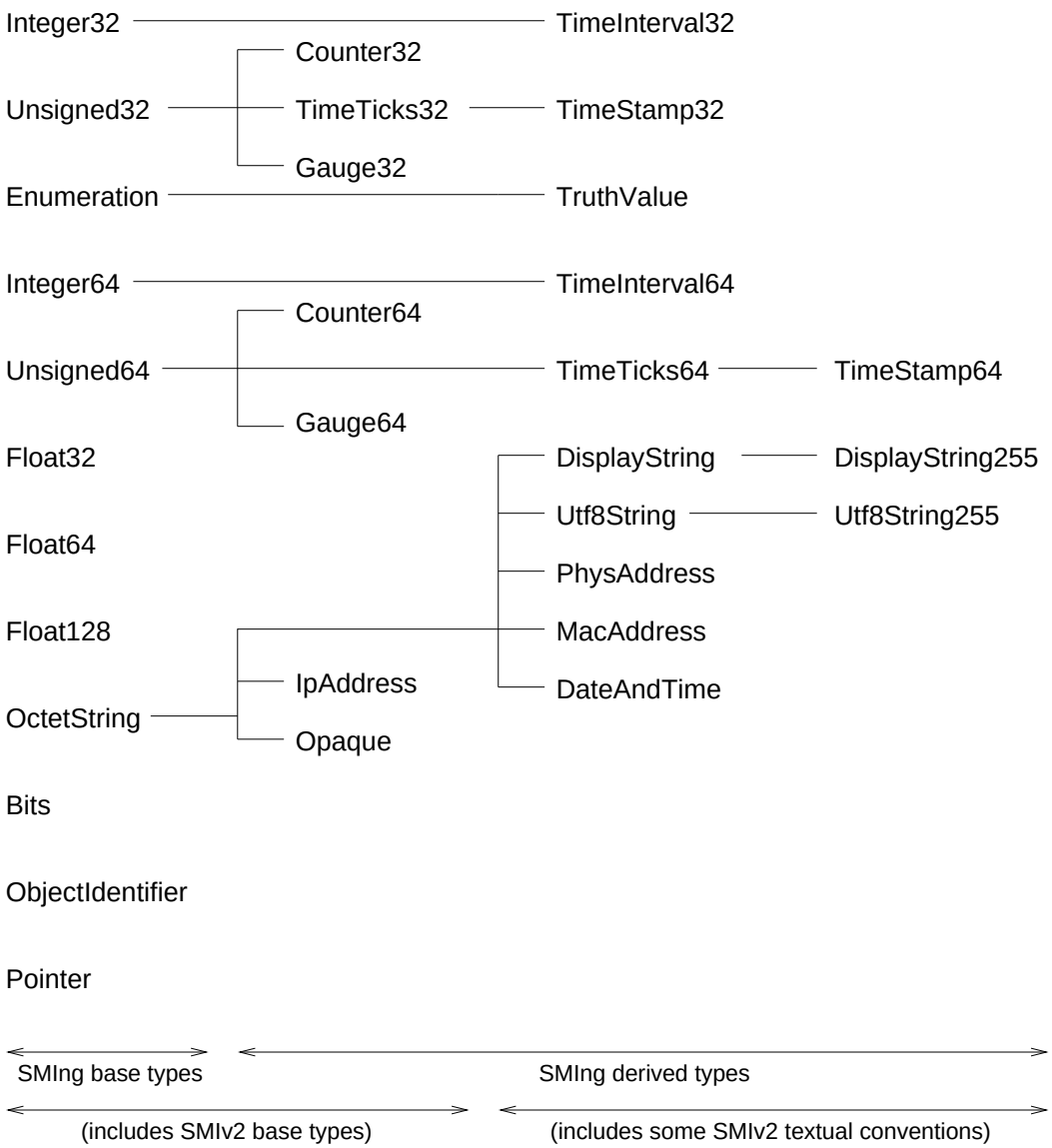


- Reusable type and class definitions are separated from protocol specific mappings.
- Abstraction of instance naming is the most difficult problem to solve.

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 separators help to recover from errors
  - complete grammar specified in ABNF (RFC 2234)
- Language extensibility:
  - declaration of new statements, parsers skip unknown statements

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 (compound 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 a future version of SMIng.

SMIng Example

```
class BasicInOutErrStats {
    attribute inOctets {
        type   Counter32;
        access readonly;
        status current;
        description
            "A counter for the number of received octets.";
    };
    attribute inErrors {  // ...
    };
    attribute outOctets { // ...
    };
    attribute outErrors { // ...
    };
    status current;
    description
        "A class for basic input/output statistics.";
};
```

SMIng Example

```
class Interface {
    attribute index {
        type   InterfaceIndex;
        access readonly;
        status current;
        description
            "Unique identification of an interface.";
    };
    attribute stats {
        type BasicInOutErrStats;
        access readonly;
        status current;
        description
            "Basic input/output statistics for an interface.";
    };
    // ...
};
```

SNMP Protocol Mapping (RFC 3781)

- Defines how SMIng base data types are mapped to SNMP data types.
- Uses Opaque wrapping to support new base types.
- Complex compound 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.

SNMP Protocol Mapping Example

```
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; ... };
        ...
    };
    ...
};
```

- The mapping is explicit, but might be generated by automated processes.
- Explicit mappings allow to handle non-standard assignments.

SMIng Status

- SMIng was a nice research / engineering effort.
- Java implementation available from INRIA (France)
- Failed to succeed in the IETF, so largely irrelevant now
- Lessons learned:
  - Naming is crucial and mapping between naming systems is hard
  - Some seemingly simple ideas sometimes take years
- Other lessons learned:
  - Good intentions and hard work are not enough to succeed in an IETF standardization effort
  - Big players can easily kill your efforts if they want
  - IETF standardization is often a subtle power game

SNMP over TCP (RFC 3430)

- Support larger message sizes to improve bulk transfers.
- Support session-based security mechanisms.
- No vehicle to turn unconfirmed operations into confirmed operations.
- Optional transport mapping (UDP still required).
- Originator of a request-response transaction chooses the transport 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!

SNMP Payload Compression

- Improve encoding efficiency to pack more useful data in SNMP messages.
- Lossless compression of SNMP payloads with minimal processing overhead.
- Compression must happen before encryption.
- Each SNMP message is compressed and decompressed in isolation ("stateless compression").
- The size of a compressed SNMP message must never exceed the size of the uncompressed SNMP message ("non-expansion policy").
- Compressed messages must have a valid ASN.1/BER encoding.

OID Delta Compression (ODC)

- Reduce the OID overhead inherent in SNMP messages
- Idea: Encode the OID of a variable names as a delta to the previous OID variable name
- 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
- Minimize the storage and processing overhead.

ODC 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 BER encoded OID, pass the encoded OID to the output buffer, else pass the encoded CompOID to the output buffer.
5. Update the last OID and goto step two if there are more varbinds.

Extended SNMP Protocol Operations

- Additional protocol operations can substantially improve SNMP's capabilities:
  - `GetRange` to improve the `GetBulk` operation
  - `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.

AES Cipher Algorithm for USM

- Problem:
  - The SNMP USM security model uses the DES cipher algorithm which is not considered very secure these days.
  - The Advanced Encryption Standard (AES) is widely accepted as a stronger replacement for DES
- AES Cipher Algorithm for the USM:
  - AES in Cipher Feedback Mode (CFB) with a key size of 128 bits.
  - Defines AES key localization and creation of the 128 bit initialization vector (IV) from the localized key.

$\Longrightarrow$ Proposed Standards (RFC 3826)

$\Longrightarrow$ Implementations available.

SNMP Uniform Resource Locators

- Problem:
  - No common mechanism to indicate how to contact the device for management.
  - Especially important when out-of-band IP management is used via a separate management interface
- SNMP Uniform Resource Locators
  - Use URL notation to identify SNMPv3 management communication endpoints.
  - Transport protocol selection (UDP vs. TCP) is implicit.

SNMP URL Examples

```
snmp://snmp.example.com
snmp://tester5@snmp.example.com:8161
snmp://snmp.example.com/bridge1
snmp://snmp.example.com/bridge1;engine=0x800002b804616263

snmp://snmp.example.com//1.3.6.1.2.1.1.3.0
snmp://snmp.example.com//1.3.6.1.2.1.1.3+
snmp://snmp.example.com//1.3.6.1.2.1.1.3.*

snmp://snmp.example.com/bridge1/1.3.6.1.2.1.2.2.1.8.*

snmp://example.com//(1.3.6.1.2.1.2.2.1.7,1.3.6.1.2.1.2.2.1.8).*
```

$\Longrightarrow$ Approved as Proposed Standard

References

[1] C. Elliot, D. Harrington, J. Jason, J. Schönwälder, F. Strauß, and W. Weiss. SMIng Objectives. RFC 3216, Cisco Systems, Enterasys Networks, Intel Corp., TU Braunschweig, Ellacoya, December 2001.

[2] F. Strauß and J. Schönwälder. SMIng - Next Generation Structure of Management Information. RFC 3780, TU Braunschweig, IU Bremen, May 2004.

[3] F. Strauß and J. Schönwälder. Next Generation Structure of Management Information (SMIng) Mappings to the Simple Network Management Protocol (SNMP). RFC 3781, TU Braunschweig, IU Bremen, May 2004.

[4] J. Schönwälder. Simple Network Management Protocol (SNMP) over Transmission Control Protocol (TCP) Transport Mapping. RFC 3430, TU Braunschweig, December 2002.

[5] J. Schönwälder. GetRange Operation for the Simple Network Management Protocol (SNMP). Internet Draft <draft-irtf-nmrg-snmp-getrange-00.txt>, International University Bremen, November 2003.

[6] U. Blumenthal, F. Maino, and K. McCloghrie. The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model. RFC 3826, Lucent Technologies, Andiamo Systems, Cisco Systems, June 2004.

[7] D. Black, K. McCloghrie, and J. Schönwälder. Uniform Resource Identifier (URI) Scheme for the Simple Network Management Protocol (SNMP). Internet Draft <draft-black-snmp-uri-09.txt>, EMC Corporation, Cisco Systems, International University Bremen, December 2004.

XML Technologies

XML Technologies

- → XML Acronyms
- → XML, DTD, XML Schema
- → XML DOM
- → XPATH
- → XSLT
- → Web Services, WSDL, SOAP

XML Acronyms

XML	The eXtensible Markup Language is a standard markup language that allows applications 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.

XML Acronyms

XQUERY	The XML Query Language is a query langauge to extract data from XML documents.
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.
WSDL	Web Services Description Language is a language to describe the behavior of collections of XML encoded messages.
SOAP	The ~~Simple Object Access Protocol~~ is for exchanging XML encoded messages.

eXtensible Markup Language (XML)

- The eXtensible Markup Language, (XML) is a standard markup language that allows applications to exchange structured documents.
- XML is a lightweight version of the Standard Generalized Markup Language (SGML) (ISO 8879).
- XML has been developed and is standardized by the World Wide Web Consortium (W3C).
- XML is the foundation of newer versions of the Hypertext Markup Language (HTML).
- XML documents can be easily parsed and processed in almost all computer languages.

Example XML Document

```
<?xml version="1.0"?>
<!DOCTYPE staff SYSTEM "staff.dtd">

<staff>
  <person>
    <name>
      <first>Peter</first>
      <last>Mustermann</last>
    </name>
    <email>peter@example.com</email>
    <email category="private">peter@yahoo.com</email>
    <phone category="work">+49 541 969 4242</phone>
    <phone category="private">+49 541 123 4242</phone>
  </person>
</staff>
```

XML Information Set

- Document Information Item
- Element Information Items
- Attribute Information Items
- Processing Instruction Information Items
- Unexpanded Entity Reference Information Items
- Character Information Items
- Comment Information Items
- Document Type Declaration Information Item
- Unparsed Entity Information Items
- Notation Information Items
- Namespace Information Items

XML Example Tree Structure

```
<?xml version="1.0"?>
DOCUMENT
version=1.0
URL=/home/schoenw/xml/staff.xml
standalone=true
  DTD(staff), SYSTEM staff.dtd
  ELEMENT staff
    TEXT content=
    ELEMENT person
      TEXT content=
      ELEMENT name
        TEXT content=
        ELEMENT first
          TEXT content=Peter
        TEXT content=
        ELEMENT last
          TEXT content=Mustermann
        TEXT content=
      TEXT content=
```

XML Example Tree Structure

```
ELEMENT email
  TEXT content=peter@example.com
TEXT content=
ELEMENT email
  ATTRIBUTE category
    TEXT content=private
  TEXT content=peter@yahoo.com
TEXT content=
ELEMENT phone
  ATTRIBUTE category
    TEXT content=work
  TEXT content=+49 541 969 4242
TEXT content=
ELEMENT phone
  ATTRIBUTE category
    TEXT content=private
  TEXT content=+49 541 123 4242
TEXT content=
TEXT content=
```

Document Type Definitions

- A Document Type Definition (DTD) is a formal description in *XML Declaration Syntax* of a particular type of document.
- A DTD defines what names are to be used for the different types of elements, where they may occur, and how they all fit together.
- A DTD provides applications with information of what names and structures can be used in a particular document type.
- Applications which are aware of a document's DTD will be able to detect illegal constructions (validation).
- The *XML Declaration Syntax* is rooted in the SGML standards.

Example XML DTD

```
<!-- DTD for the staff.xml file -->

<!ENTITY % CTEXT          "#PCDATA">

<!ELEMENT staff           (person*)>
<!ELEMENT person          (name, email+, phone+)>
<!ELEMENT name            (title?, first, middle?, last)>
<!ELEMENT title           (%CTEXT;)>
<!ELEMENT first           (%CTEXT;)>
<!ELEMENT middle          (%CTEXT;)>
<!ELEMENT last            (%CTEXT;)>
<!ELEMENT email           (%CTEXT;)>
<!ELEMENT phone           (%CTEXT;)>

<!ATTLIST email
          category        (work|private|other)    "work">
<!ATTLIST phone
          category        (work|private|other)    "work">
```

XML Schema Definitions

- XML Schema is an alternative to a DTD.
- XML Schema definitions are written in *XML Instance Syntax* and provide much more extensive validation facilities.
- The W3C XML Schema recommendation provides a means of specifying *formal data typing* and *validation of element content* in terms of those data types.
- XML Schemas are written as XML files, avoiding the need for processing software to be able to read *XML Declaration Syntax* as well as *XML Instance Syntax*.

Example XML Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:per="http://www.inf.uos.de/schoenw/person"
            xml:lang="en">

  <xsd:annotation>
    <xsd:documentation>
      This schema defines the formal syntax of the staff
      structured XML schema type.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="staff">
    <xsd:sequence>
      <xsd:element name="person" type="per:person"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Example XML Schema

```
<?xml version="1.0"?>

<xsd:schema targetNamespace="http://www.inf.uos.de/schoenw/person"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xml:lang="en">

  <xsd:annotation>
    <xsd:documentation>
      This schema defines the formal syntax of the person
      structured XML schema type.
    </xsd:documentation>
  </xsd:annotation>

  <!--
        The following two complex types define the person and
        name sequences of elements. This is still simple...
   -->
```

Example XML Schema

```
<xsd:complexType name="person">
  <xsd:sequence>
    <xsd:element name="name" type="name"/>
    <xsd:element name="email" type="email"
      minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element name="phone" type="phone"
      minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="name">
  <xsd:sequence>
    <xsd:element name="title"  type="xsd:string"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="first"  type="xsd:string"/>
    <xsd:element name="middle" type="xsd:string"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="last"   type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Example XML Schema

```
<xsd:complexType name="email">
  <xsd:simpleContent>
    <xsd:extension base="emailString">
      <xsd:attributeGroup ref="categoryAttributeGroup"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="phone">
  <xsd:simpleContent>
    <xsd:extension base="phoneString">
      <xsd:attributeGroup ref="categoryAttributeGroup"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

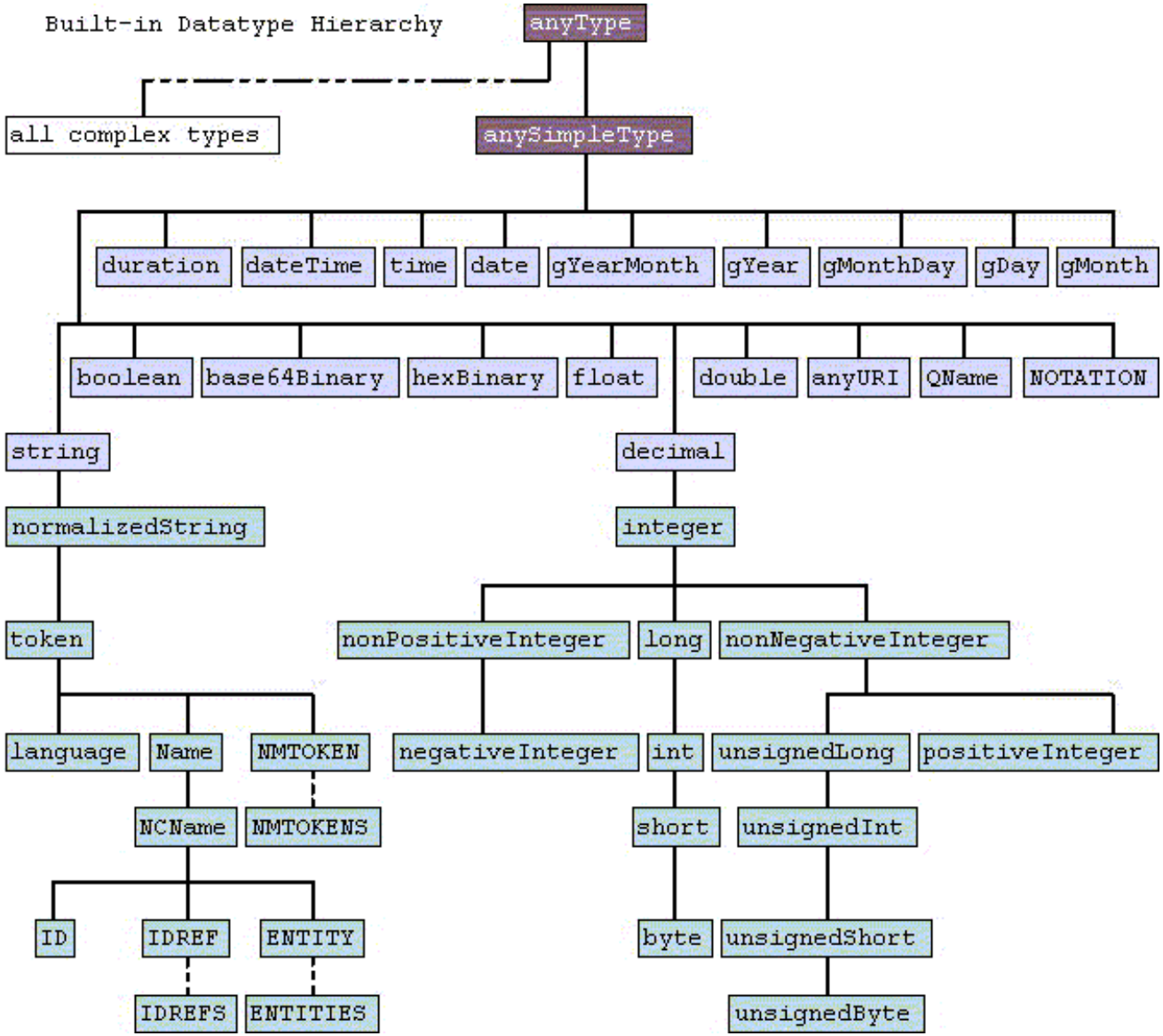
Example XML Schema

```
<!--  These are our simple types for email and phone strings.
      Regular expressions are used to restrict the set of legal
      values. -->

<xsd:simpleType name="emailString">
  <xsd:restriction base="xsd:string">
    <!-- <xsd:pattern value=""/> -->
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="phoneString">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\+?[0-9 ]+"/>
  </xsd:restriction>
</xsd:simpleType>
```

Example XML Schema

```
  <!--  The attribute group allows to define the category
        attribute in one place. See the above reference to
        categoryAttributeGroup. -->

  <xsd:attributeGroup name="categoryAttributeGroup">
    <xsd:attribute name="category">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="work"/>
          <xsd:enumeration value="private"/>
          <xsd:enumeration value="other"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:attributeGroup>

</xsd:schema>
```

XML Schema Datatype Hierarchy



Canonical XML

- Encode the document in UTF-8
- Line breaks normalized to #xA on input, before parsing
- Attribute values are normalized, as if by a validating processor
- Character and parsed entity references are replaced
- CDATA sections are replaced with their character content
- The XML declaration and document type declaration (DTD) are removed
- Empty elements are converted to start-end tag pairs
- Whitespace outside of the document element and within start and end tags is normalized

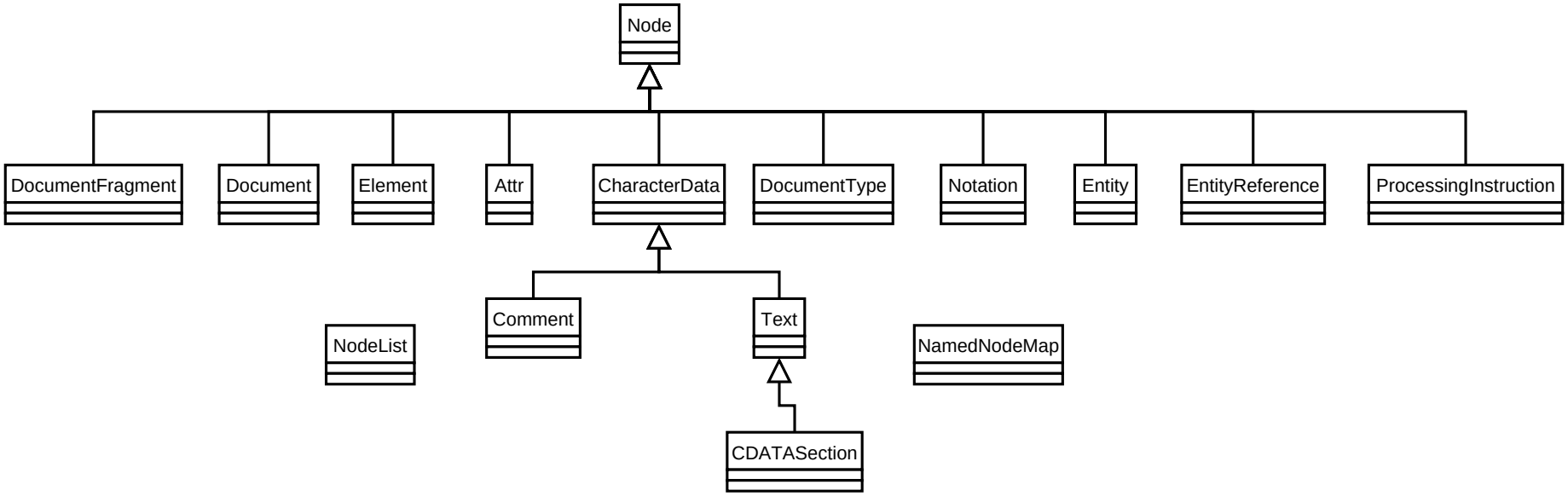
Canonical XML

- All whitespace in character content is retained (excluding characters removed during line feed normalization)
- Attribute value delimiters are set to quotation marks (double quotes)
- Special characters in attribute values and character content are replaced by character references
- Superfluous namespace declarations are removed from each element
- Default attributes are added to each element
- Lexicographic order is imposed on the namespace declarations and attributes of each element

Document Object Model (DOM)

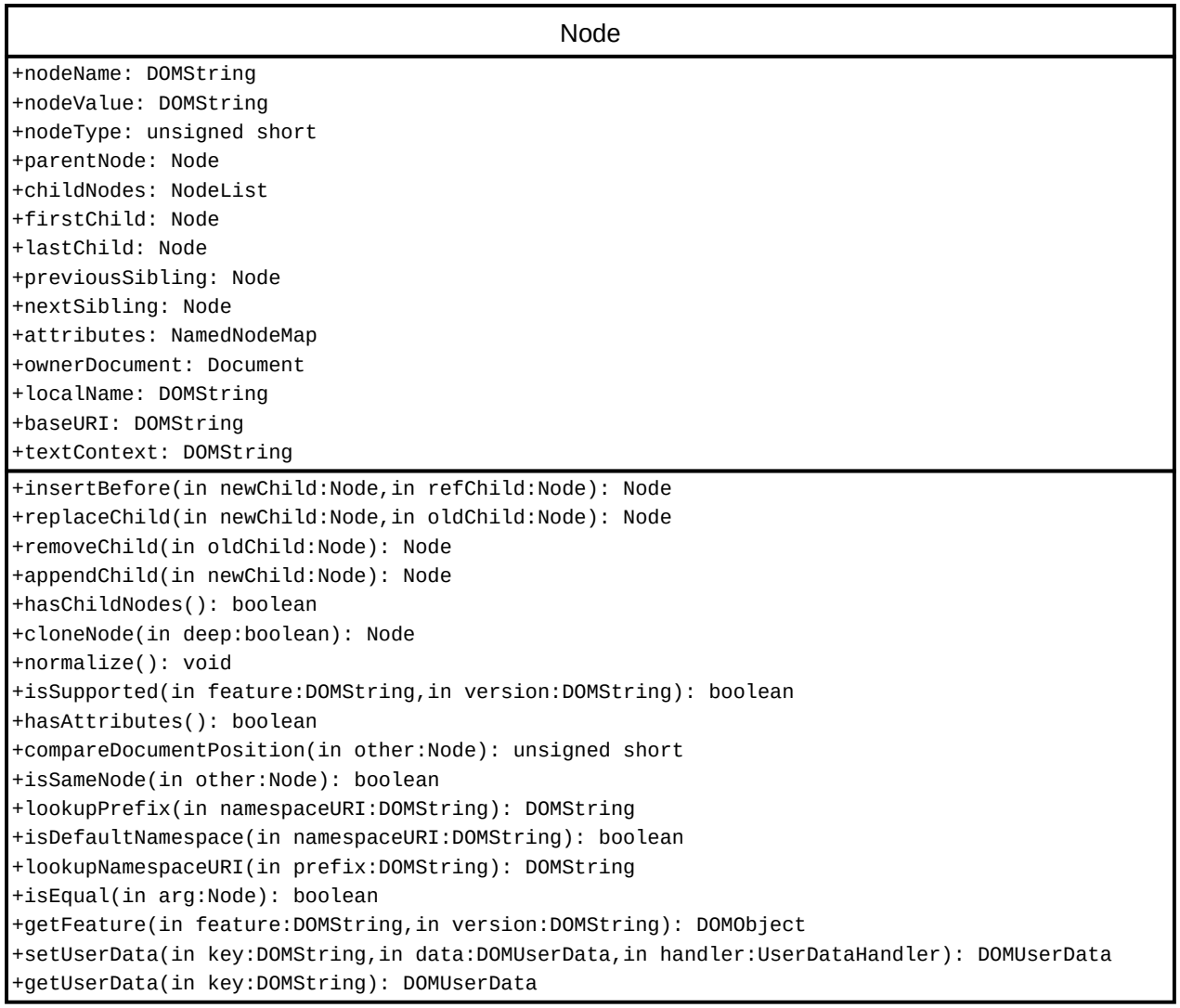
- The Document Object Model (Core) represents documents as a hierarchy of Node objects
- The DOM basically maps the XML Information Set to programmatic interfaces
- The DOM specification uses CORBA IDL as a mechanism to describe DOM
- Holding full DOMs in memory is quite expensive
- In many cases, the DOM representation is actually bigger than the original XML document...

Core DOM Definitions (UML)



- Classe for all XML node types are derived from the DOM `Node` class
- Several DOM helper classes/interfaces are not shown here
- The `Node` class itself is rather heavy-weight

DOM Node Class in UML



XML Path Language (XPath)

- The primary purpose of XPath is to address parts of an XML document.
- In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans.
- XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values.
- XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax.
- XPath gets its name from its use of a path notation as in URLs for navigating through the hierarchical structure of an XML document.

XPath Expressions

- The result of an XPATH expression has one of the following four basic types:
  1. node-set (a set of nodes without duplicates)
  2. boolean (true or false)
  3. number (a floating-point number)
  4. string (a sequence of UCS characters)
- Expression evaluation occurs with respect to a context:
  - a node (the context node)
  - a pair of non-zero positive integers (the context position and the context size)
  - a set of variable bindings
  - a function library
  - the set of namespace declarations in scope

XPATH Examples

- Select all `phone` elements in the document:

  ```
  //phone
  ```

- Select all `phone` elements in a `person` element:

  ```
  //person/phone
  ```

- Select all `person` elements that are children of the `staff` root element:

  ```
  /staff/person
  ```

  Note that this is now an absolute path!

- Select all child elements of all `person` elements that are children of the `staff` root element

  ```
  /staff/person/*
  ```

XPATH Examples

- Select all elements with a child element named `phone`:

```
//*[phone]
```

- Select all elements with an attribute named `category`:

```
//*[@category]
```

- Select all element with a child elements `name` and `first` where the contents of `first` equals `Peter`.

```
//*[name/first="Peter"]
```

- Select all elements with an attribute named `category` holding the value `private`.

```
//*[@category="private"]
```

XPATH Examples

- Select all element with a child element named `phone` and a child element named `email`:

```
//*[phone and email]
```

- Select all elements with a child element named `phone` whose `category` attribute has the value `private`:

```
//*[phone/@category="private"]
```

- Select all private email address and all work phone numbers:

```
//email[@category="private"] | //phone[@category="work"]
```

- Even more elaborate matches are possible by using XPATH functions (see XPATH specification).

XSLT

- XSLT is a language for transforming XML documents into other XML documents.
- A transformation in the XSLT language is expressed as a well-formed XML document.
- XSLT is template-driven.
- Transformation templates are applied to nodesets, which can be selected using XPATH expressions.
- XSLT also features imperative programming constructs, such as conditional statements and loops.

XSLT Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/1999/xhtml" version="1.0">

  <xsl:template match="staff">
    <table>
    <xsl:apply-templates select="person"/>
    </table>
  </xsl:template>

  <xsl:template match="person">
    <tr>
      <td><xsl:apply-templates select="name"/></td>
      <td><xsl:apply-templates select="email"/></td>
      <td><xsl:apply-templates select="phone"/></td>
    </tr>
  </xsl:template>
```

XSLT Example

```
<xsl:template match="name">
  <xsl:if test="title">
    <xsl:text> </xsl:text>
    <xsl:value-of select="title"/>
  </xsl:if>
  <xsl:value-of select="first"/>
  <xsl:if test="middle">
    <xsl:text> </xsl:text>
    <xsl:value-of select="middle"/>
  </xsl:if>
  <xsl:text> </xsl:text>
  <xsl:value-of select="last"/>
</xsl:template>

<xsl:template match="email">
  <xsl:apply-templates/>
  <xsl:text> </xsl:text>
</xsl:template>
```

XSLT Example

```
  <xsl:template match="phone">
    <xsl:apply-templates/>
    <xsl:text> </xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

- XSLT is kind of unusual to write at the beginning due to the implicit matching loops.
- The `xsltproc` implementation is pretty fast (compared to some other Java implementations).
- XSLT can be used extensively to generate HTML Web pages from XML files describing the content.

Web Services

- A web service is a collection of functions packaged as a single entity and published to the network for use by other applications
  - Stock quote lookup services
  - Web search services (google)
  - Ticket purchase services
- Web services can aggregate other web services to provide a higher-level set of features
- Ultimate goal: In the future software will be assembled from a web of services

$\Longrightarrow$ Note that there is no object orientation!

Web Services Vision

- Build your applications just-in-time
- Dynamically discover and coordinate (orchestrate) the execution of services on the network
- Will be able to choose between alternative implementations of the same service
- Access the application from everywhere at any time

Foundation Standards

- Web Service Description Language (WSDL)
  - Describe a web service in WSDL (often automated by tools, and generated from a Java interface or a C/C++ header file)
- ~~Simple Object Access Protocol~~ (SOAP)
  - Invoke the web service using SOAP as the message format (usually transparent)
  - Typically runs over HTTP (with all the pros and cons)
- Universal Description, Discovery and Integration (UDDI)
  - Publish the service description in UDDI registry
  - Organized by business type, business, and service

Web Service Description Language

- WSDL is a specification defining how to describe web services in a common XML grammar
- WSDL describes four critical pieces of data:
  - Interface information describing all publicly available functions
  - Data type information for all message requests and message responses
  - Binding information about the transport protocol to be used
  - Address information for locating the specified service

WSDL Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions ...>

  <wsdl:types ...>              <!-- data type definitions (xsd) -->
  </wsd:types>

  <wsdl:message ...>            <!-- message format definitions -->
  </wsdl:message>

  <wsdl:portType ...>           <!-- operation definitions -->
  </wsdl:portType>

  <wsdl:binding ...>            <!-- binding to the transport(s) -->
  </wsdl:binding>

  <wsdl:service ...>            <!-- service location definition -->
  </wsdl:service>

</wsdl:definitions>
```

SOAP

- *SOAP Envelope*:
  The outermost element information item of a SOAP message.
- *SOAP Header*:
  A collection of zero or more SOAP header blocks each of which might be targeted at any SOAP receiver within the SOAP message path.
- *SOAP Body*:
  A collection of zero or more element information items targeted at an ultimate SOAP receiver in the SOAP message path.

SOAP Example

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelo
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding
      <key xsi:type="xsd:string">00000000000000000000000000000000</key>
      <q xsi:type="xsd:string">shrdlu winograd maclisp teletype</q>
      <start xsi:type="xsd:int">0</start>
      <maxResults xsi:type="xsd:int">10</maxResults>
      <filter xsi:type="xsd:boolean">true</filter>
      <restrict xsi:type="xsd:string"></restrict>
      <safeSearch xsi:type="xsd:boolean">false</safeSearch>
      <lr xsi:type="xsd:string"></lr>
      <ie xsi:type="xsd:string">latin1</ie>
      <oe xsi:type="xsd:string">latin1</oe>
    </ns1:doGoogleSearch>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Examples

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelo
  <SOAP-ENV:Body>
    <ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-ENV:enc
      <return xsi:type="ns1:GoogleSearchResult">
        <documentFiltering xsi:type="xsd:boolean">false</documentFilterin
        <estimatedTotalResultsCount xsi:type="xsd:int">3</estimatedTotalR
        <directoryCategories xmlns:ns2="http://schemas.xmlsoap.org/soap/e
        <searchTime xsi:type="xsd:double">0.194871</searchTime>
        <resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encodi
          <!-- result items removed, long lines not wrapped -->
        </resultElements>
        <endIndex xsi:type="xsd:int">3</endIndex>
        <searchTips xsi:type="xsd:string"></searchTips>
        <searchComments xsi:type="xsd:string"></searchComments>
        <startIndex xsi:type="xsd:int">1</startIndex>
        <estimateIsExact xsi:type="xsd:boolean">true</estimateIsExact>
        <searchQuery xsi:type="xsd:string">shrdlu winograd maclisp telety
      </return>
    </ns1:doGoogleSearchResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

References

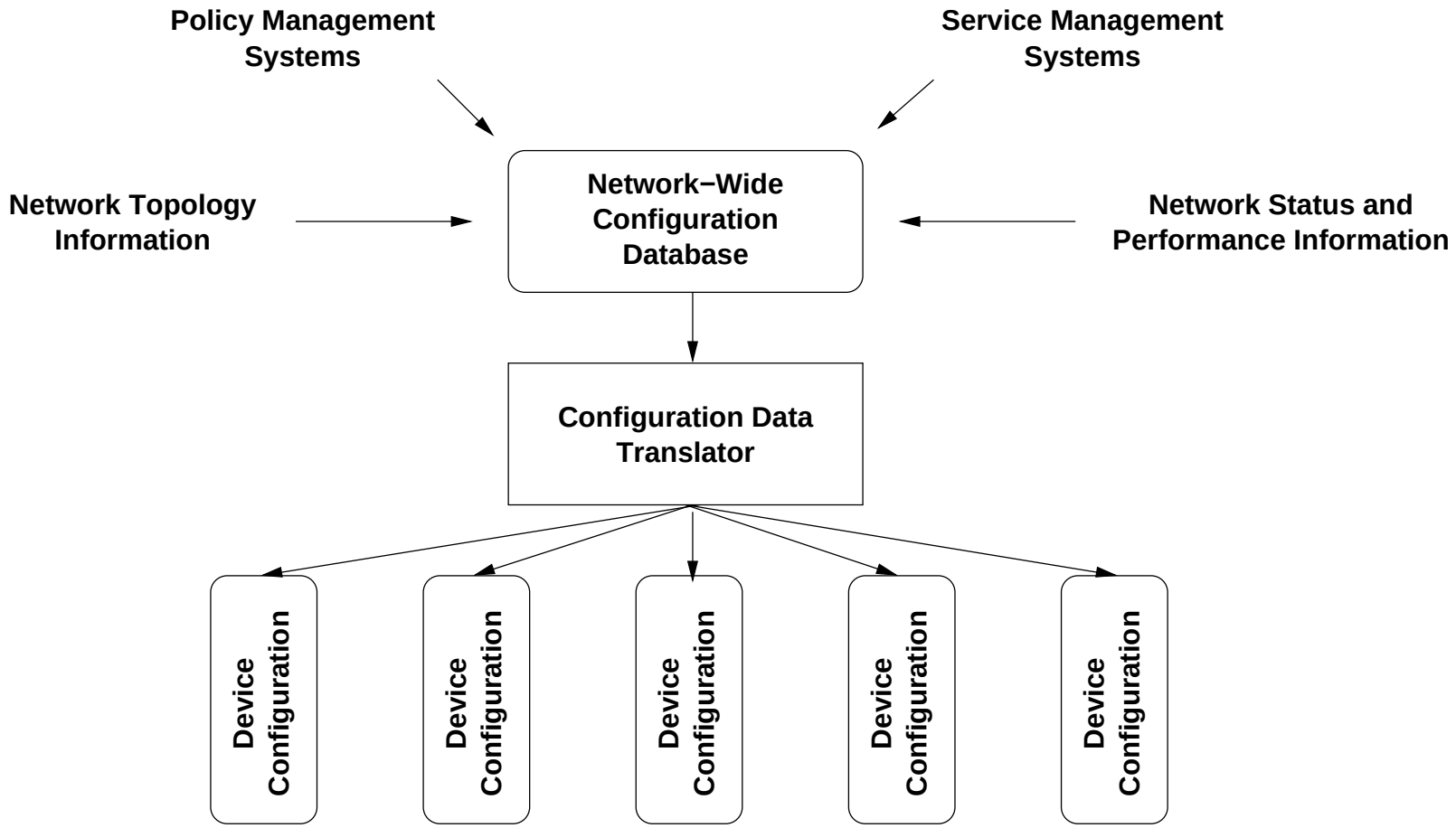
[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. W3C Recommendation, Textuality and Netscape, Microsoft, University of Illinois, February 1998.

Revolutionary Research

Revolutionary Research

- → Network-Wide Configuration Management
- → JunoScript by Juniper Networks
- → IETF NetConf Protocol
- → Web Services for Management

Network-Wide Configuration Management

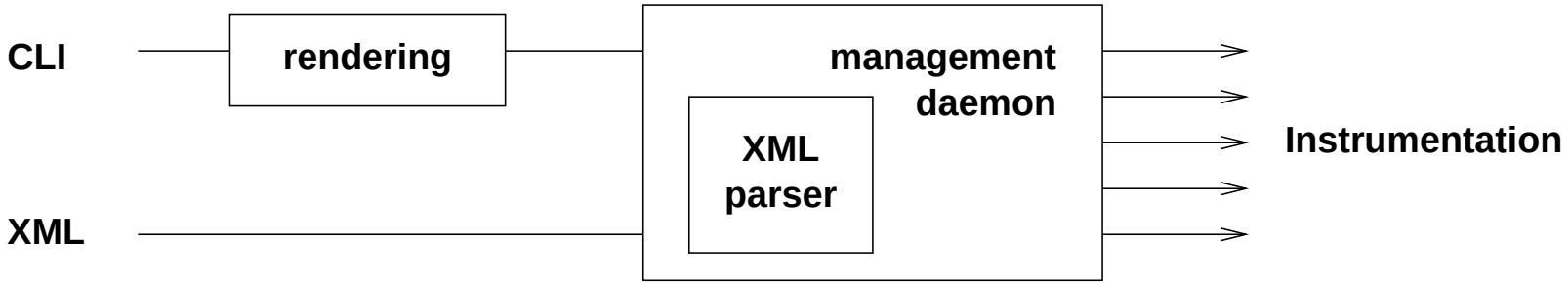


$\Longrightarrow$ Treating configurations as documents leads naturally to the application of XML.

JunoScript by Juniper Networks

- Juniper Networks developed JunoScript as a programmatic interface for their router products.
- JunoScript uses XML for data representation and the protocol messages.
- JunoScript uses a simple RPC protocol running over Telnet or SSH.
- Operators like the JunoScript because it makes it easier to automate processes.
- JunoScript provides special the primitives to build robust network-wide configuration management systems (e.g., timed confirmed commits).

JunoScript by Juniper Networks



- The Juniper command line interface internally uses JunoScript.
- A rendering engine converts the XML data representation into a more compact human readable format.
- Requests from the CLI are processed internally in exactly the same way as requests coming from the programmatic interface.

JunoScript RPC Example

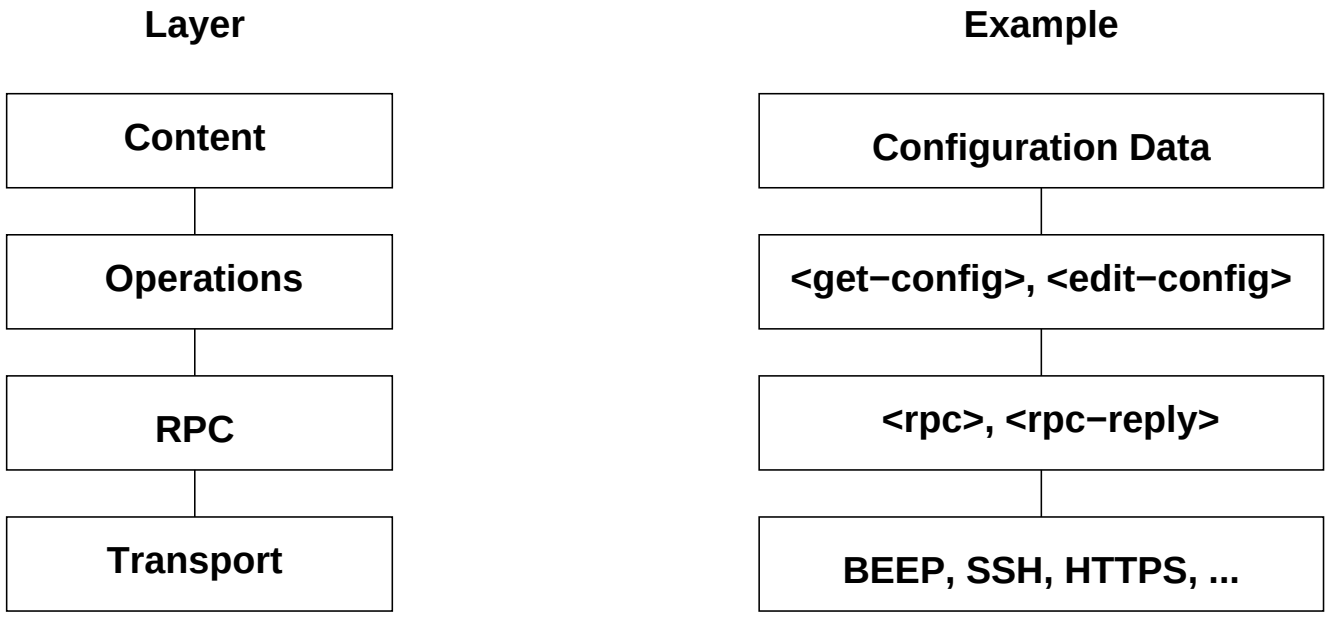
```
<rpc>
  <get-interface-information>
    <statistics/>
  </get-interface-information>
</rpc>

<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 IETF Working Group

- Chartered to define an XML-based configuration management protocol on the basis of JunoScript.
- Core contributors from Juniper Networks and Cisco.
- Actively seeking input from network operators.
- No work on NetConf data models before the protocol work has been finished.
- Some design decisions are difficult to take.
- Running behind schedule (like many IETF WGs)
- Prototyping efforts at least at INRIA (France), IUB (Germany), Postech (Korea)

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 any standardization efforts.

Configuration Datastores

- A configuration datastore is defined as the complete set of configuration data that is required to get a device from its initial default state into a desired operational state.
- The <*running*> configuration datastore represents the currently active configuration of a device and is always present.
- The <*startup*> configuration datastore represents the configuration that will be used during the next startup.
- The <*candidate*> configuration datastore represents a configuration which may become a <*running*> or <*startup*> configuration.
- Only the <*running*> configuration datastore is required.

NetConf Operations (mostly finalized)

- `get-config(source, filter)`
  Retrieve all or part of a specified configuration from a given source.
- `edit-config(target, options, config)`
  Edit target configuration, merge / replace / delete embedded in config data.
- `copy-config(source, target)`
  Create or replace an entire configuration with the contents of the source.
- `delete-config(target)`
  Delete a configuration datastore.
- `get(filter)`
  Retrieve device state information.

NetConf Operations (mostly finalized)

- `validate(source)`
  Validate the contents of the specified configuration (capability).
- `lock(source)`
  Lock a configuration source.
- `unlock(config)`
  Unlock a configuration source.
- `commit(confirmed, confirmed-timeout)`
  Commit candidate config as the new current configuration (capability).

SSH Protocol

- SSH is a protocol for secure remote login and other secure network services over an insecure network.
- The SSH protocol consists of three major components:
  1. The *Transport Layer Protocol* provides server authentication, confidentiality, and integrity with perfect forward secrecy.
  2. The *User Authentication Protocol* authenticates the client-side user to the server.
  3. The *Connection Protocol* multiplexes the encrypted tunnel into several logical channels. It runs over the user authentication protocol.
- SSH is widely deployed on network devices as a secure protocol to access the command line interface.

NetConf over SSH

- Motivation: Use an already deployed security protocol, thereby reducing the operational costs associated with key management.
- SSH supports multiple logical channels over one transport layer association.
- For framing purposes, the special end of message marker "]]>]]>" (without the quotes) has been introduced.
- NetConf over SSH has been selected as the default transport for NetConf.

NetConf over SSH Example

```
S: <?xml version="1.0" encoding="UTF-8"?>
S: <hello>
S:   <capabilities>
S:     <capability>
S:       urn:ietf:params:xml:ns:netconf:base:1.0
S:     </capability>
S:     <capability>
S:       urn:ietf:params:xml:ns:netconf:base:1.0#startup
S:     </capability>
S:   </capabilities>
S:   <session-id>4<session-id>
S: </hello>
S: ]]>]]>
```

NetConf over SSH Example

```
C: <?xml version="1.0" encoding="UTF-8"?>
C: <hello>
C:   <capabilities>
C:     <capability>
C:       urn:ietf:params:xml:ns:netconf:base:1.0
C:     </capability>
C:   </capabilities>
C: </hello>
C: ]]>]]>
```

NetConf over SSH Example

```
C: <?xml version="1.0" encoding="UTF-8"?>
C: <rpc message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1
C:   <get-config>
C:     <source><running/></source>
C:     <config xmlns="http://example.com/schema/1.2/config">
C:      <users/>
C:     </config>
C:   </get-config>
C: </rpc>
C: ]]>]]>
```

NetConf over SSH Example

```
S: <?xml version="1.0" encoding="UTF-8"?>
S: <rpc-reply message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:
S:   <config xmlns="http://example.com/schema/1.2/config">
S:     <users>
S:       <user><name>root</name><type>superuser</type></user>
S:       <user><name>fred</name><type>admin</type></user>
S:       <user><name>barney</name><type>admin</type></user>
S:     </users>
S:   </config>
S: </rpc-reply>
S: ]]>]]>
```

BEEP Protocol (RFC 3080)

- BEEP is a generic application protocol kernel for connection-oriented, asynchronous interactions.
- BEEP supports multiple channels, application layer framing and fragmentation.
- BEEP exchange styles:
  - `MSG/RPY`
  - `MSG/ERR`
  - `MSG/ANS`
- Integrates into SASL (RFC 2222) and TLS (RFC 2246) for security.
- Connections can be initiated by both participating peers (no strict client/server roles).

NetConf over BEEP

- BEEP supports multiple logical channels.
- Every peer can be the initiator of a connection.
- SASL allows to map to existing security infrastructures.
- Framing and fragmentation services provided by BEEP.
- BEEP is currently not widely deployed and there is a lack of operational experience with BEEP in the operator community.
- BEEP is considered to be an optional NetConf transport.

NetConf over BEEP Example

```
M: MSG 0 1 . 10 48 101
M: Content-type: application/beep+xml
M: <start number=’’1’’>
M:   <profile uri=’’http://iana.org/beep/netconf’’ />
M: </start>
M: END
A: RPY 0 1 . 38 87
A: Content-Type: application/beep+xml
A:
A: <profile uri=’’http://iana.org/beep/netconf’’ />
A: END
```

NetConf over BEEP Example

```
A: MSG 1 0 . 0 436
A: Content-type: application/beep+xml
A:
A: <hello xmlns=’’urn:ietf:params:xml:ns:netconf:base:1.0’’>
A:   <capabilities>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0
A:     </capability>
A:     <capability>
A:       urn:ietf:params:xml:ns:netconf:base:1.0#startup
A:     </capability>
A:   </capabilities>
A:   <session-id>4</session-id>
A: </hello>
A: END
M: RPY 1 0 . 0 0
M: END
```

NetConf over BEEP Example

```
M: MSG 1 42 . 24 344
M: Content-Type: text/xml; charset=utf-8
M:
M: <rpc message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1
M:   <get-config>
M:     <source><running/></source>
M:     <config xmlns="http://example.com/schema/1.2/config">
M:      <users/>
M:     </config>
M:   </get-config>
M: </rpc>
M: END
```

NetConf over BEEP Example

```
A: RPY 1 42 . 24 542
A: Content-Type: text/xml; charset=utf-8
A:
A: <rpc-reply message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:
A:   <config xmlns="http://example.com/schema/1.2/config">
A:     <users>
A:       <user><name>root</name><type>superuser</type></user>
A:       <user><name>fred</name><type>admin</type></user>
A:       <user><name>barney</name><type>admin</type></user>
A:     </users>
A:   </config>
A: </rpc-reply>
A: END
```

NetConf over SOAP/HTTP[S]

- Instead of inventing a special purpose RPC protocol, use existing Web Services standards.
- Pros:
  - more developers / tools available
  - better integration with IT infrastructure
- Cons:
  - base technology not under control of the IETF
  - unneeded complexity
  - interoperability problems (immature technology)
  - HTTP is a bad generic application protocol kernel
- Note: Proposal does not map NetConf operations to SOAP operations!

NetConf WSDL Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:SOAP="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="urn:ietf:params:xml:ns:netconf:soap:1.0"
  xmlns:xb="urn:ietf:params:xml:ns:netconf:base:1.0"
  targetNamespace="urn:ietf:params:xml:ns:netconf:soap:1.0"
  name="soap_1.0.wsdl">

  <import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
          location="http://iana.org/ietf/netconf/base_1.0.xsd"/>

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

NetConf WSDL Definition

```
  <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="urn:ietf:params:xml:ns:netconf:base:1.0"/>
      </input>
      <output>
        <SOAP:body use="literal"
             namespace="urn:ietf:params:xml:ns:netconf:base:1.0"/>
      </output>
    </operation>
  </binding>

</definitions>
```

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="urn:ietf:params:xml:ns:netconf:soap:1.0"
  targetNamespace="urn:myNetconfService"
  name="myNetconfService.wsdl">

  <import namespace="urn:ietf:params:xml:ns:netconf:soap:1.0"
          location="http://iana.org/ietf/netconf/soap_1.0.wsdl"/>

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

NetConf over SOAP/HTTP Example

```
C: POST /netconf HTTP/1.1
C: Host: netconfdevice
C: Content-Type: text/xml; charset=utf-8
C: Accept: application/soap+xml, text/*
C: Cache-Control: no-cache
C: Pragma: no-cache
C: Content-Length: 465
C:
C: <?xml version=''1.0'' encoding=''UTF-8''?>
C: <soapenv:Envelope
C:   xmlns:soapenv=''http://www.w3.org/2003/05/soap-envelope''>
C:   <soapenv:Body>
C:     <rpc message-id=''101''
C:          xmlns=''xmlns=''urn:ietf:params:xml:ns:netconf:base:1.0''>
C:       <get-config>
C:         <filter type=''subtree''>
C:           <top xmlns=''http://example.com/schema/1.2/config''>
C:             <users/>
C:           </top>
C:         </filter>
C:       </get-config>
C:     </rpc>
```

NetConf over SOAP/HTTP Example

```
S: HTTP/1.1 200 OK
S: Content-Type: application/soap+xml; charset=utf-8
S: Content-Length: 917
S:
S: <?xml version=''1.0'' encoding=''UTF-8''?>
S: <soapenv:Envelope
S:   xmlns:soapenv=''http://www.w3.org/2003/05/soap-envelope''>
S:   <soapenv:Body>
S:     <rpc-reply message-id=''101''
S:                xmlns=''urn:ietf:params:xml:ns:netconf:base:1.0''>
S:       <data>
S:         <top xmlns=''http://example.com/schema/1.2/config''>
S:           <users>
S:             <user>
S:               <name>root</name>
S:               <type>superuser</type>
S:               <full-name>Charlie Root</full-name>
S:                 <dept>1</dept>
S:                 <id>1</id>
S:               </company-info>
S:             </user>
S:             <user>
```

NetConf Status

- Still a number of open questions:
  - Operations: mandatory primitives?
  - Filtering: ad-hoc subtree?, XPATH?, XPATH light?, XQUERY?, ...
  - Transport: SSH (must), BEEP (may), SOAP/HTTP[S] (may), TLS?, SCTP? ...
  - Modeling: XML Schema?, RELAXng?, SMIng?, ...
  - Integration: SNMP?, CLI?, ...
- Must come to conclusions fast, otherwise NetConf might be too late to be successful.

Use Web Services for Management

- What is the right granularity?
  1. Variable granularity:

     `getIfAlias(ifIndex, ...)`, `setIfAlias(ifIndex, ...)`
  2. Object granularity:

     `getInterface(ifIndex, ...)`, `setInterface(ifIndex, ...)`
  3. Collection granularity:

     `getAllInterfaces(...)`, `setAllInterfaces(...)`
  4. Operation granularity:

     `get(...)`, `set(...)`
- Fine granularity simplifies integration but might be inefficient.
- Coarse granularity requires to parse structured data, but might use powerful filtering mechanisms.

Web Services Measurements

- Question: What is the performance relative to SNMP?
- Prototyped
  - `GetIfCell()`
  - `GetIfColumn()`
  - `GetIfRow()`
  - `GetIfTable()`

  using Web Services for the IF-MIB.
- Prototype uses NET-SNMP (5.0.X) instrumentation and the gSOAP Web Services toolkit.
- Measurements done on a 800 MHz Pentium machine running Debian Linux (kernel 2.4.22).
- Work done at the University of Twente

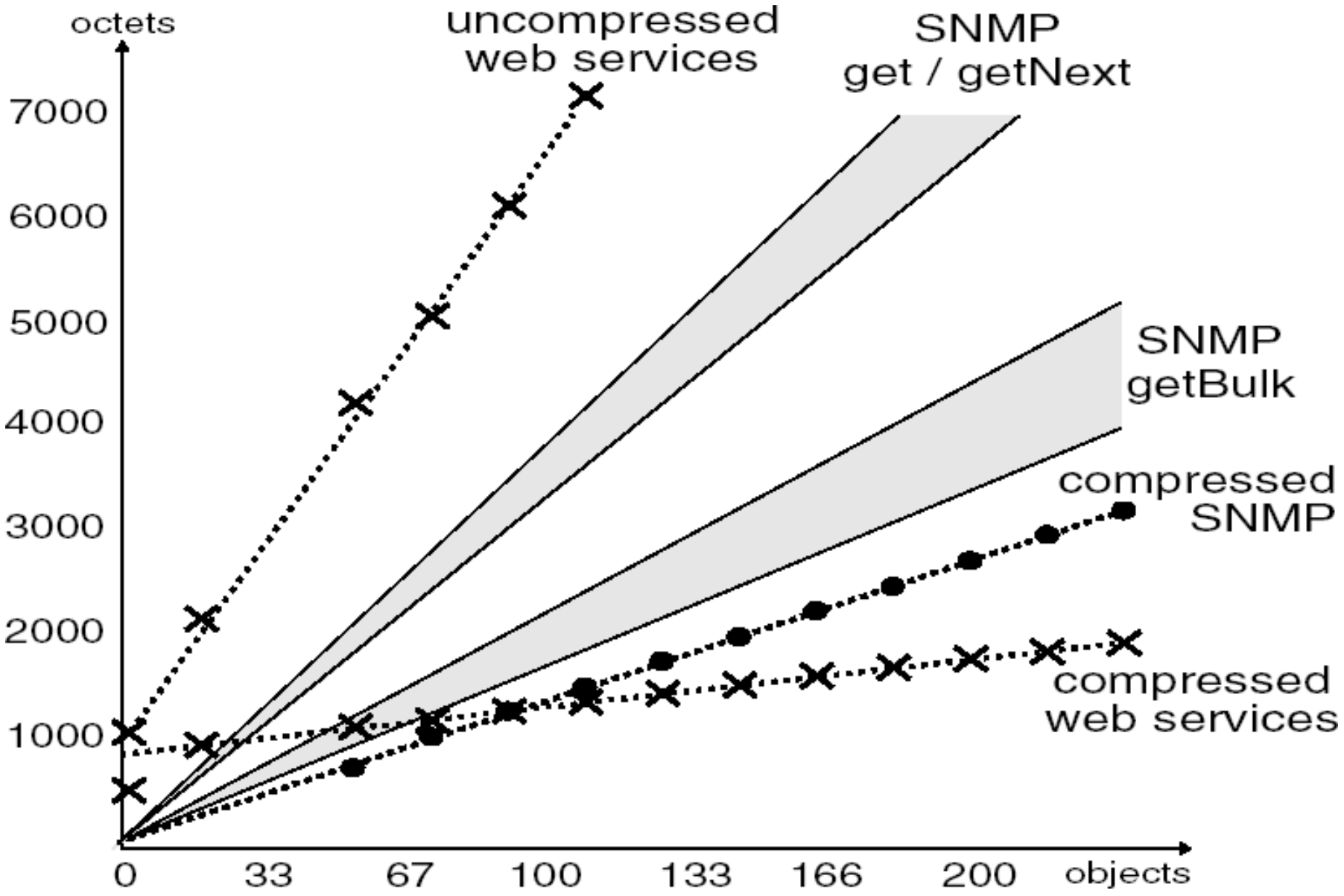
WSDL Fragment

```
<complexType name="GetIfTableResponse">
    <sequence>
        <element name="ifEntry" type="utMon:ifEntry"
                 minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<message name="GetIfTableRequest">
    <part name="commuity" type="xsd:string"/>
</message>
<message name="GetIfTableResponse">
    <part name="-sizeTable" type="xsd:int"/>
    <part name="ifEntry" type="utMon:ifEntry"/>
</message>

<portType name="GetIfTableServicePortType">
    <operation name="GetIfTable">
        <input message="tns:GetIfTableRequest"/>
        <output message="tns:GetIfTableResponse"/>
    </operation>
</portType>
```

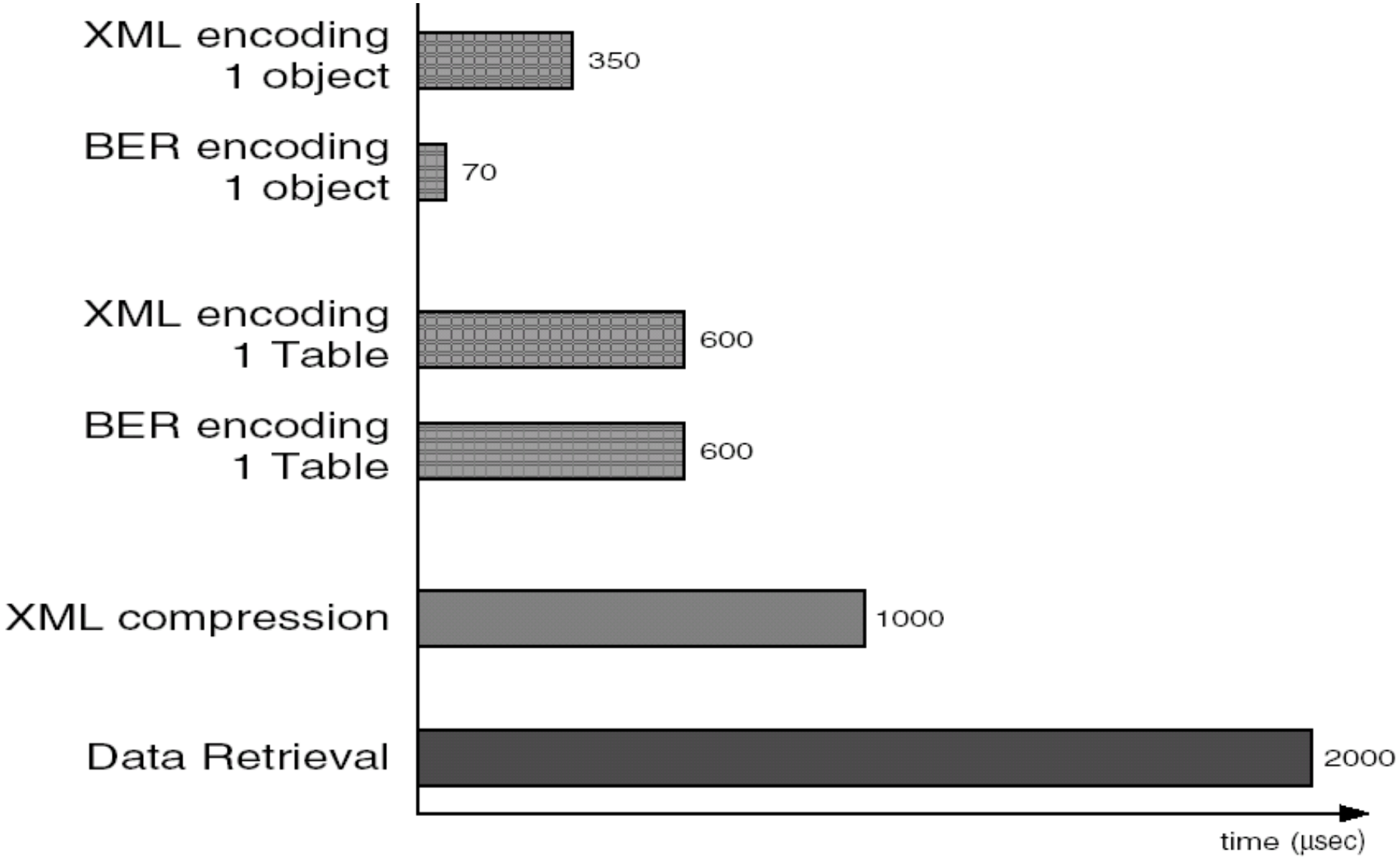
XSD Fragment (simplified)

```
<complexType name="ifEntry">
  <sequence>
    <element name="ifIndex" type="xsd:unsignedInt"/>
    <element name="ifDescr" type="xsd:string"/>
    <element name="ifType" type="xsd:unsignedInt"/>
    <element name="ifMtu" type="xsd:unsignedInt"/>
    <element name="ifSpeed" type="xsd:unsignedInt"/>
    <element name="ifPhysAddress" type="xsd:string"/>
    <element name="ifAdminStatus" type="xsd:unsignedInt"/>
    <element name="ifOperStatus" type="xsd:unsignedInt"/>
    <element name="ifLastChange" type="xsd:unsignedInt"/>
    <element name="ifInOctets" type="xsd:unsignedInt"/>
    <element name="ifInUcastPkts" type="xsd:unsignedInt"/>
    <element name="ifInDiscards" type="xsd:unsignedInt"/>
    <element name="ifInErrors" type="xsd:unsignedInt"/>
    <element name="ifInUnknownProtos" type="xsd:unsignedInt"/>
    <element name="ifOutOctets" type="xsd:unsignedInt"/>
    <element name="ifOutUcastPkts" type="xsd:unsignedInt"/>
    <element name="ifOutErrors" type="xsd:unsignedInt"/>
  </sequence>
</complexType>
```

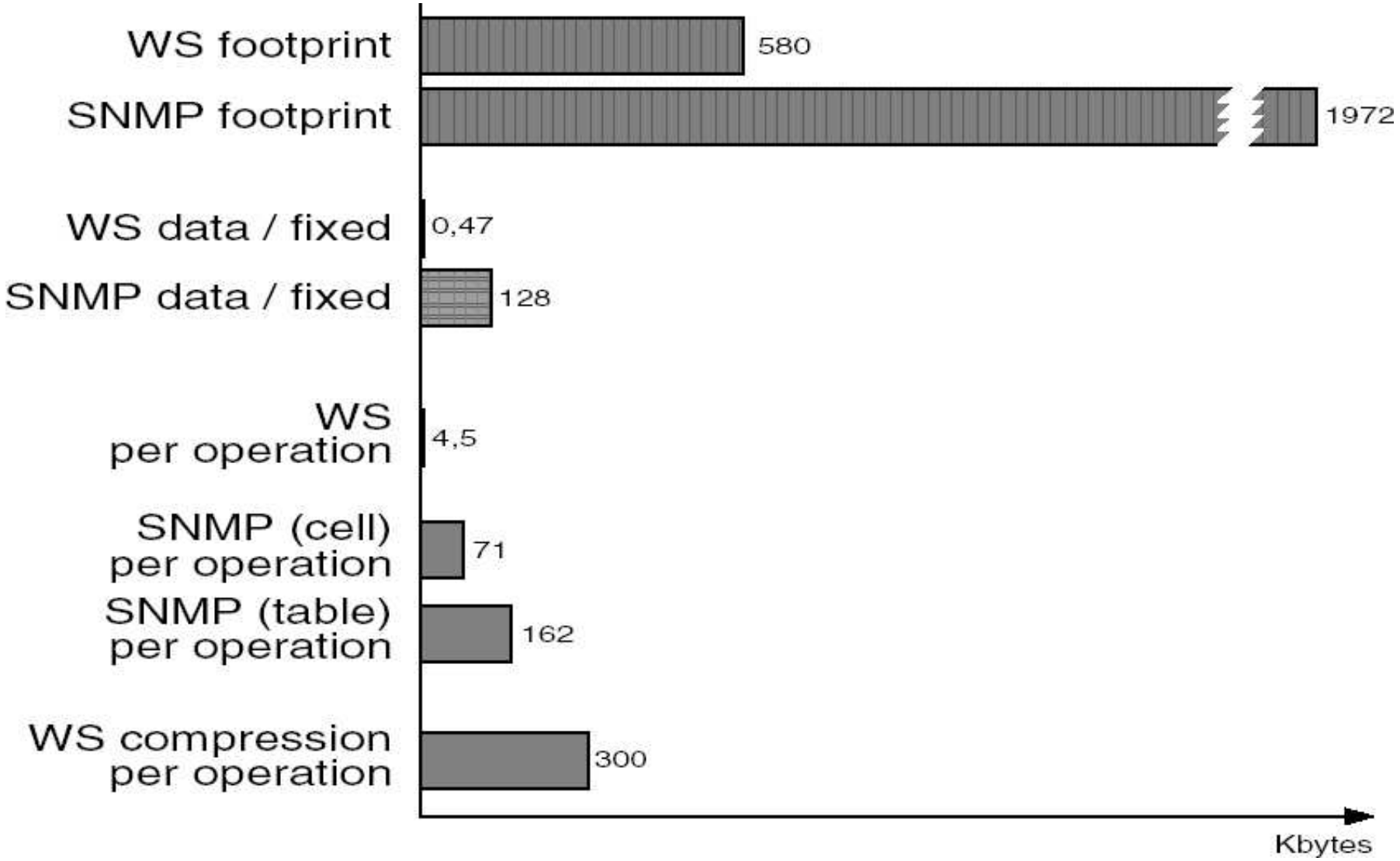
Bandwidth Usage



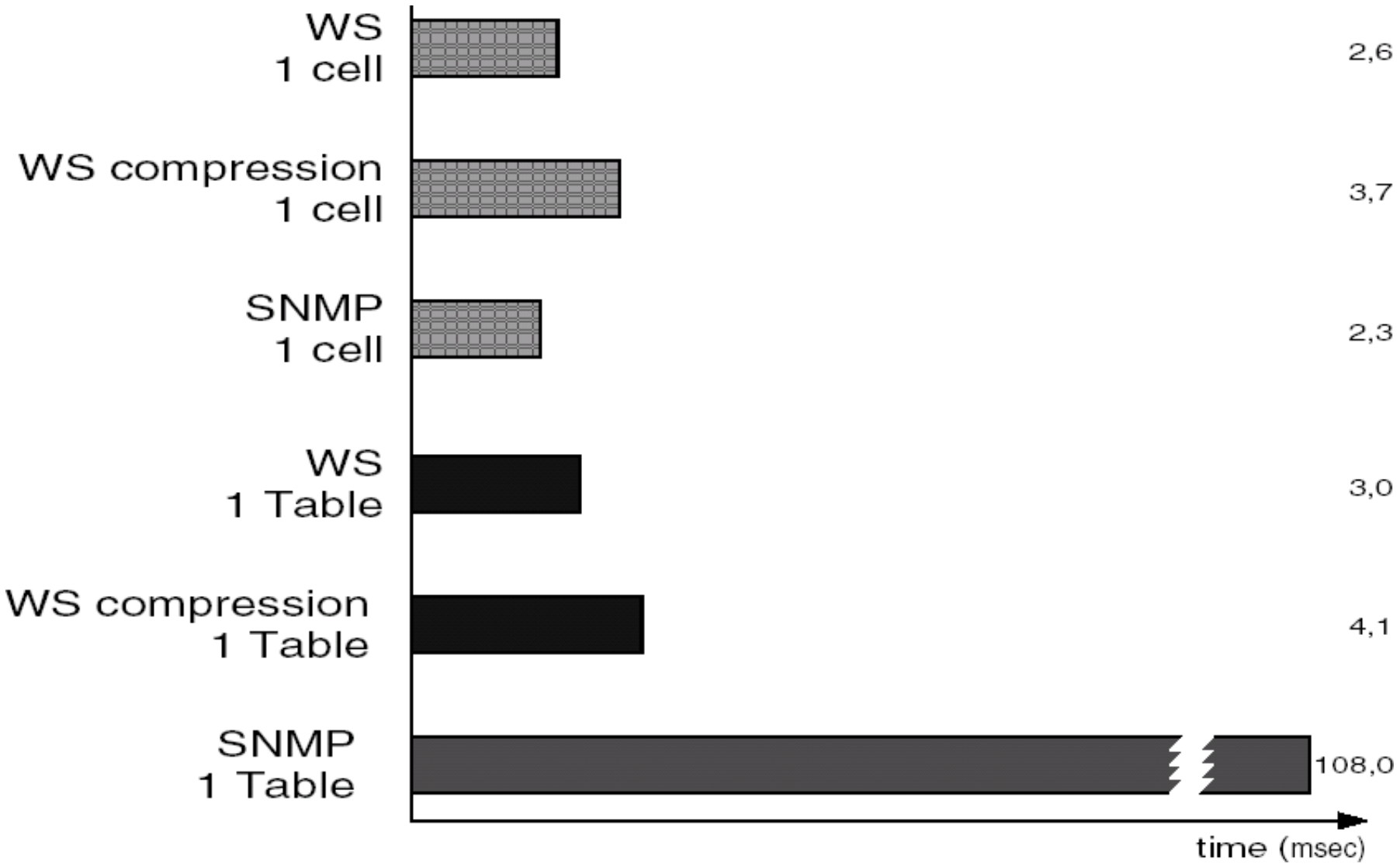
CPU Usage


XML encoding
1 object
350
BER encoding
1 object
70
XML encoding
1 Table
600
BER encoding
1 Table
600
XML compression
1000
Data Retrieval
2000
time (µsec)

Memory Usage



End to End Latency



References

[1] J. Schönwälder. Overview of the 2002 IAB Network Management Workshop. RFC 3535, International University Bremen, May 2003.

[2] L. Sanchez, K. McCloghrie, and J. Saperia. Requirements for Configuration Management of IP-based Networks. RFC 3139, Megisto, Cisco, JDS Consultant, June 2001.

[3] R. Enns. NETCONF Configuration Protocol. Internet Draft <draft-ietf-netconf-prot-06.txt>, Juniper Networks, April 2005.

[4] M. Wasserman and T. Goddard. Using the NETCONF Configuration Protocol over Secure Shell (SSH). Internet Draft <draft-ietf-netconf-ssh-04.txt>, ThingMagic, IceSoft Technologies, April 2005.

[5] E. Lear and K. Crozier. Using the NETCONF Protocol over Blocks Extensible Exchange Protocol (BEEP). Internet Draft <draft-ietf-netconf-beep-05.txt>, Cisco Systems, April 2005.

[6] T. Goddard. Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP). Internet Draft <draft-ietf-netconf-soap-05.txt>, ICEsoft Technologies Inc., April 2005.

[7] A. Pras, T. Drevers, R. van de Meent, and D. Quartel. Comparing the Performance of SNMP and Web Services based Management. IEEE electronic Transactions on Network and Service Management, 1(2), November 2004.

Discussion