Sistemi di Elaborazione dell'Informazione: Elementi di Gestione di Rete

Prima Parte:
Paradigmi e Protocolli per la Gestione di Rete

Anno Accademico 2004/2005

Luca Deri <deri@ntop.org>

References

- Books:
  - *D. Mauro, K. Schmidt, Essential SNMP, O'Reilly & Associates, 2001.*
  - W. Stallings: SNMP, SNMPv2 and CMIP - The Practical Guide to Network Management Standards, Addison Wesley, 1993
  - D. Zeltserman, A Practical Guide to SNMPv3 and Network Management, Prentice Hall, 1999.
  - W. Stallings: SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Addison Wesley, 1998.
  - M.T. Rose: The Simple Book - An Introduction to Management of TCP/IP based Internets, Prentice Hall, 1996
  - M. Sloman: Network and Distributed Systems Management, Addison Wesley, 1994
  - D. Perkins, E. McGinnis: Understanding SNMP MIBs, Prentice Hall, 1997

1. Introduction

1. Introduction
   - 1.1 Motivation
   - 1.2 Terminology and basic concepts
   - 1.3 Networking Basics
   - 1.4 Abstract Syntax Notation One
2. Internet Management
3. OSI Management
4. Current Developments

1.1 Motivation: Why Do We Need Management ?

- Current situation:
  - increasing meaning of strategic resources "information”.
  - a computer network is no longer only a supporting item in an enterprise, but takes even more frequently a key position.
  - the number of interconnected computers rose dramatically in the past few years. This process will probably continue to persist.
  - Complexity and functionality of the components grows in correspondence with the performance of the available hardware.

- Demand:
  - Permanent availability of network services with optimal quality.
  - Cost reduction for the network infrastructure of the company.

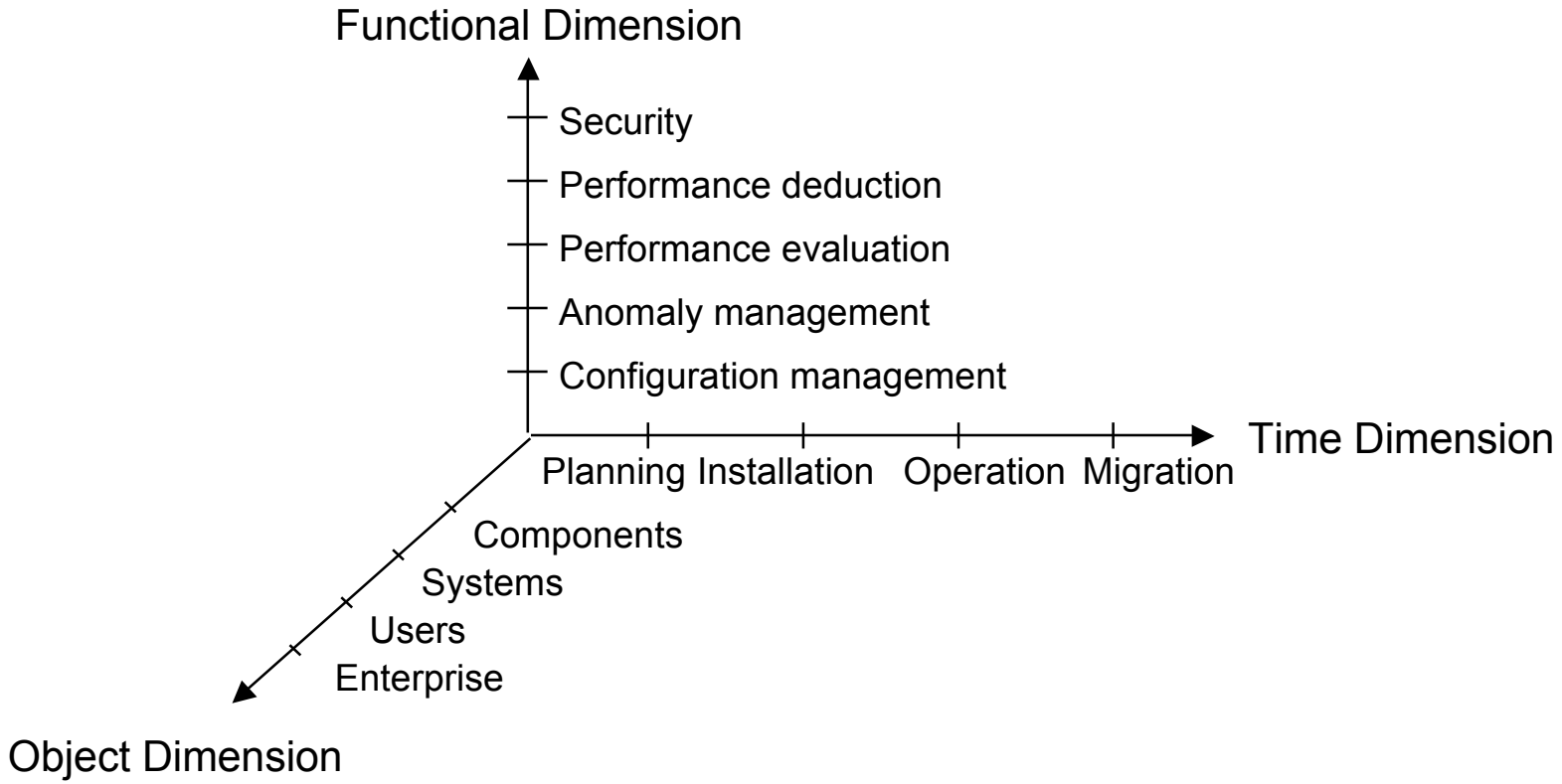
- Necessity:
  - computer-aided management of heterogeneous networks.

Subject of this Lecture

- Management implementation:
  - By humans (network administrators, operating surgeons), by special tools (hardware and software).
  - Hence network management is first of all an organisational problem.
  - Cost effective and flexible network guidelines and procedures need be compiled.
  - Tools and their technological bases are just aids to successful network management.

- Subject of this lecture:
  - Network management basics.
  - Architecture and functionality of network management
  - Open standards issued by independent organisations.

Network Management Dimensions



Time Horizons in the Operating Phase

The operating phase is divided into three time horizons:

- Short term horizon:
  - Management functions, that must be provided within second or minutes.
  - Complete automatization is required.
- Middle term horizon:
  - Management functions to be provided within an hour.
  - They are often handled semi-automatically with help of human experts.
- Long term horizon:
  - Strategic functions to be provided within the range of weeks, months or years.
  - The planning aspect is the centre of attention (manufacturer selection [scouting], procurements, cost planning).

System Management Requirements [1/2]

- Guarantee the availability of the function on the net:

- Service maintenance (availability, response time) need to face with technological changes and big quota increase.
  - Security of the services through the control of security components.
  - (Human) Mistake prevention and bottleneck identification/recovery.

- Automatic or semiautomatic reaction on operation anomalies:
  - Real-time configuration modification in case of error.
  - Activation of redundant components in case of error.

- Dynamic reactions to changes on the network and environment:
  - Changes regarding applications, users, components, services or fees.
  - Dynamic adaptation of the available transmission bandwidth according to requests originated by the management system.

System Management Requirements [2/2]

- Network control:
  - Collection and (compressed) representation of relevant network information.
  - Definition and maintenance of a database of network configurations.
  - When applicable, centralisation of the control over peripherals and implemented functions (central management console).
  - Integration of management procedures on heterogeneous environments

- Improvement of system/network administrators work conditions :
  - Improvement and standardisation of the available tools
  - Identify and implement gradual automation of management functions.
  - Good integration of tools into the existing operational sequences.

- Progress through standardisation :
  - Transition of existing, often proprietary, solutions in a standardised environment.

Current Situation

- Network Documentation:
  - Today large amount of data are handled manually with a big waste of time.
  - Inconsistent data produce problems with configuration modifications, error detection and localisation.
- Error diagnosis:
  - Errors are often partially recognised.
  - Only the symptoms of an error are often observable.
  - Error situations can cause a lot of error messages difficult to correlate.
- Detection of weak points:
  - Weak points and sporadically occurring errors can often be recognised with difficulty.
- Correlation statistics:
  - Analysers enable power measurements for individual nodes, transmission circuits or logs.
  - Interpretation of raw data is often possible just by experts.
- User friendliness:
  - Outputs of the tools must be completed by the know-how of an expert.
  - Personnel training is a substantial factor within the operating cost of a network.

Targets of the Current Developments

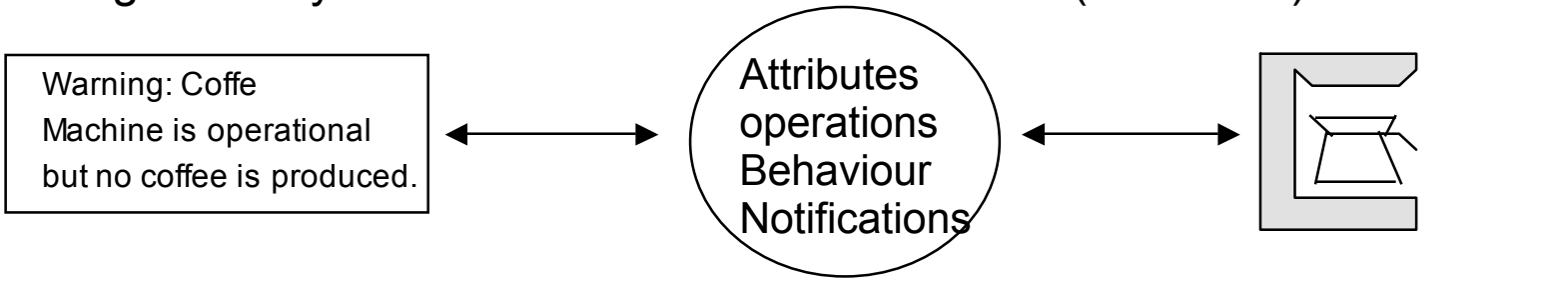
- Implementation of integrated management systems which cover all the requirements for the management of heterogeneous networks and systems.
- Good expandability and adaptability to the local network environment.
- Good support during the automation of management flows and conversion of management guidelines.
- Scalability of both the size of the network and increasing demanding requests of the management systems.
- Open interfaces to the existing infrastructure and their integration into operational sequences.
- Protection of the management against attacks of unauthorised persons.

1.2 Terminology and Fundamental Concepts

- Control, co-ordination and monitoring of resources takes place via the manipulation from so-called managed objects:

  "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)

- Managed objects are an abstract representation of a real resource.
- The boundary of a managed object specifies which details are accessible to a management system and which ones are shielded (black box).



- Managed objects do not necessarily correspond to objects, as one knows from object-oriented programming. Simple variables correspond to the MOs in the Internet management.

Managed Objects (MO)

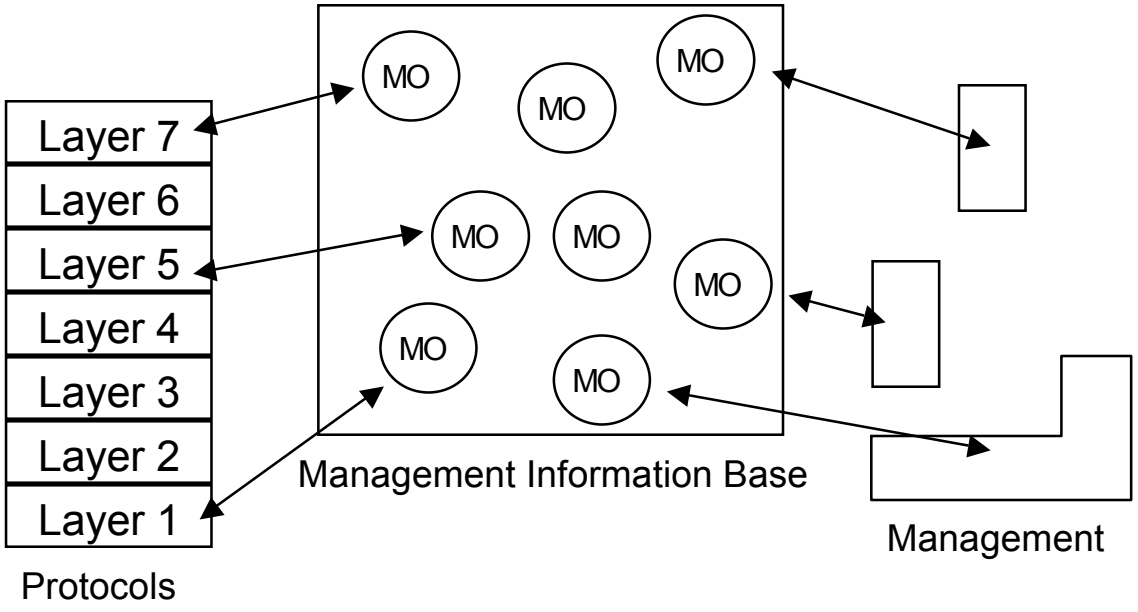
- Attributes:
  - Attributes describe the state/condition of managed objects.
  - Attributes can change when the condition of the real object changes.
  - Attributes can be manipulated by means of management operations.
- Operations:
  - Make it possible to access a managed object. Typical operations are get, set, create and delete.
  - The number and type of operations influence the object performance and complexity.
- Behaviour:
  - Determines the semantics and interaction with the real resource.
  - The behaviour of managed objects is normally defined in plain English.
- Notifications:
  - The quantity and type of the messages, which can be generated by pre-defined situations by a managed object when specific situations occur.

Management Information Base (MIB)

- The union of all managed objects contained in a system forms the *Management Information Base* (MIB) of the system:

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

- This is the first interpretation of the term "Management Information Base“ (more definitions will follow).



- A MIB should be known both to the implementer and the manager.

MIB Modularity

- Managed objects of a system are usually defined in multiple MIB definitions.

- Modules have been introduced in MIBs for enabling design modularity:
  - Different modules can be defined by different teams.
  - Management functionality can be gradually extended and modified.
  - Different systems can support different MIB modules/releases.
  - Vendors can extend the management functionality by means of proprietary MIBs.
  - MIBs are defined using a specification language

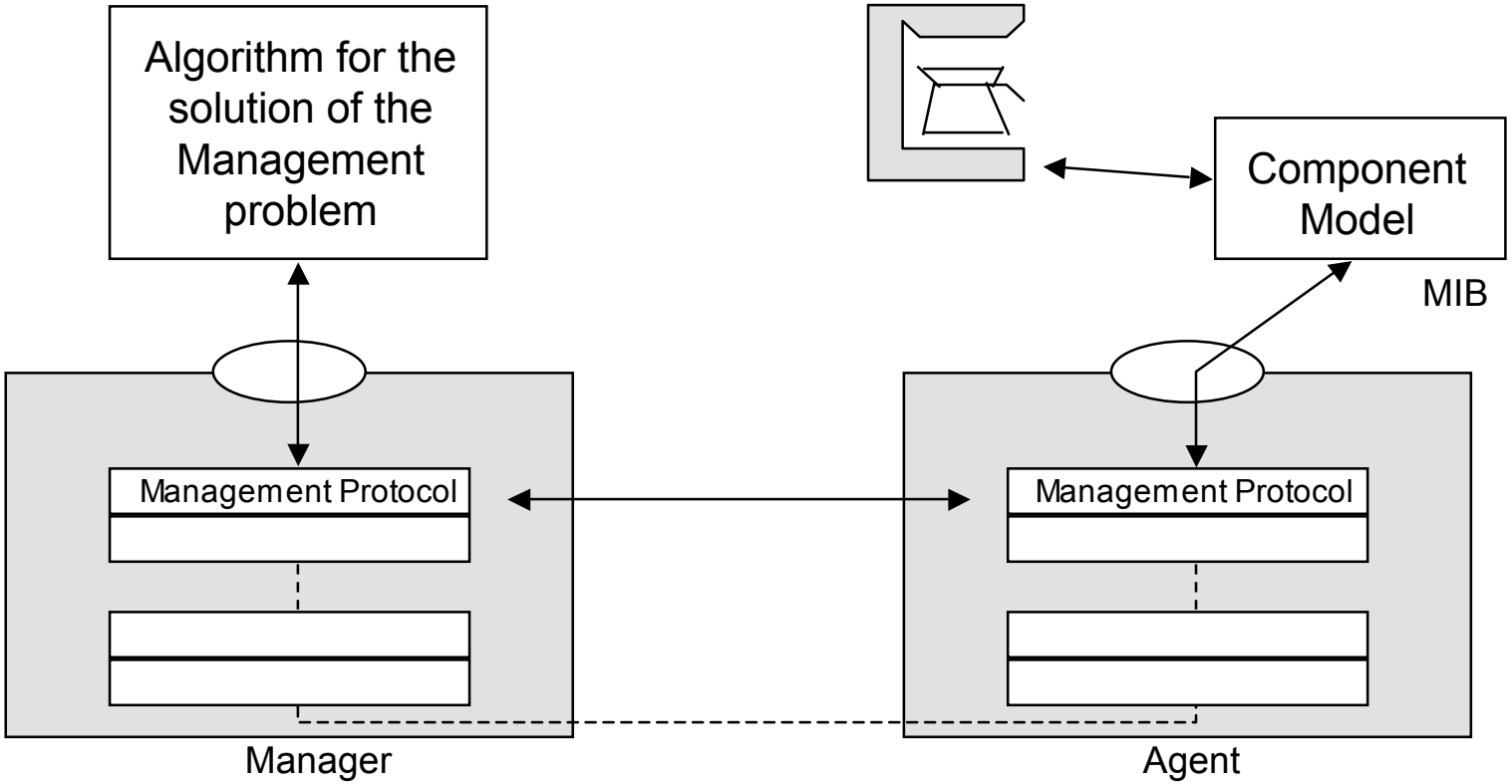
Manager/Agent Paradigm

- Agent:
  - Implements the MOs MIB by accessing the real resources.
  - Receives requests from a manager, processes them and transmits appropriate responses.
  - Dispatches notifications about important changes in status in the MIB.
  - Protects MOs against unauthorised accesses using access control rules and communication authentication with the partner.

- Manager:
  - Exercises control: it controls functions hence it is the crucial instance.
  - Starts up management operations by appropriate protocol operations for the manipulation of MOs.
  - Receives messages from agents and passes them on (for handling) to appropriate applications.

Management Protocol

- Management applications and MOs are not often on same node.
- A management protocol implements access to distant managed objects by encoding management data that is then secured during the transfer.



Functional Areas (FCAPS) [1/2]

- Management applications can be divided into 5 function areas:

- Fault management:
  - Error detection, isolation, and repair.

- Configuration management:
  - Production and administration of configuration information.
  - Name administration.
  - Start, check and termination of services.

- Account management:
  - Entry of consumption (usage) data.
  - Distribution and monitoring of contingents.
  - Customer billing for resource consumption.

Functional Areas (FCAPS) [2/2]

- Performance management:
  - Statistic data collection.
  - Determination of the system performance.
  - Systems modifications for increase in efficiency.

- Security management:
  - Production and verification of security policies.
  - Generation and distribution of passwords and accounts.
  - Report and analysis of security-relevant events.

- These 5 functional areas according to the initial letters of the English terms normally under the contraction FCAPS.
- These functional areas are not mutually independent (data measurement has often impact on system configuration).
- Basic functions (e.g. monitoring of a counter for threshold values) often reside in different functional areas.

Management Architectures Overview

- Structure of the management information:
  - ↪ defines the rules of the description of Managed Objects.
    - Identification and designation of Mos.
    - Composition of MOs.
    - Behaviour of MOs.
    - Relations to other MOs.
    - Possible operations and internal messages of the MOs.
  - ↪ Definition of the datatypes, structure and syntax for the description of the MOs.
  - ↪ The quantity of the descriptions of MOs in accordance with these rules defines the *Management Information Base* (MIB)

- Management Protocols and Services:
  - ↪ Defines the services and enable the access to remote MOs.
  - ↪ Several protocols can be used for the implementation of the defined services.
  - ↪ The service primitive and the appropriate protocol operations influence considerably the efficiency and the complexity of the management system.

Management Architectures Overview

- Organisational Model:
  - ↪ Definition of the distribution of roles of a management architecture.
    - co-operative management of similar systems.
    - systems belonging to different management authorities (hierarchical concept)
  - ↪ Partitioning in *Management Domains* according to different criteria

- Functional Model:
  - ↪ Analysis of the total function and partitioning into functional areas by means of generic auxiliary functions.
  - ↪ Definition of the auxiliary functions and their parameters.
  - ↪ Implementation using several MOs (management support objects)

1.3 Overview: Network Basics

- Copper Cable (Twisted Pair):
  - Inexpensive technology for low frequencies.
  - susceptible to disturbances (electromagnetic irradiation).

- Coax(ial) Cable:
  - technology for high to very high frequencies.
  - It consists of a central conductor embodied by a peripheral conductor.
  - Not much susceptibility to electromagnetic irradiation's.
  - Small absorption with partial loss of energy at high frequencies.

- Optical Fibres:
  - Support for very high frequencies.
  - Monomodal fibre carries light over almost 100 km without reinforcement (repeater).
  - Efficient and cheaper than coaxial cables, although the link technology is somewhat more complicated (transceiver).

Network Basics

- **Radio Link Relay (line of sight):**
  - ↪ Modulates the signal on an electromagnetic wave (carrier).
  - ↪ Application in areas, in those a wiring is not practicable (e.g. city, distant sites).
  - ↪ It must exist a line of sight connection between senders and recipients.
  - ↪ Error rate depends on the view conditions (weather, fog).

- **Radio Waves:**
  - ↪ Radiant emittance of an area (cell), determined by electromagnetic waves.
  - ↪ Possible mobility of the recipients and senders (GSM) .
  - ↪ Narrow bandwidth and high fault liability.

- **Satellites:**
  - ↪ Long transmission time [latency] (approx. 200 ms) between senders and recipients.
  - ↪ Very high frequencies and bandwidths between microwaves and multiplexers.
  - ↪ Suitable for transfer over geostationary satellites, which turn with the earth.

Line vs. Packet Switched Systems

- **Circuit Switching (Circuit Switched Network):**
  - From a sender to a recipient over a constantly established physical line.
  - Communication takes place in the following phases:
    1. Connection establishment.
    2. Data exchange.
    3. Connection clearing.
  - After the connection has been established the bandwidth is completely available to the sender (reserved bandwidth).
  - Examples: Plain Telephone, GSM, Leased Lines.

- **Packet Switched Networks:**
  - Messages divided into small units, so-called packets.
  - There is only a constant line from the sender to the next relay station.
  - Relay stations receive packets and send them towards the target.
  - Relay stations must know the path to individual targets (choice of route = routing).
  - The bandwidth between relays can be better used (over-planning, traffic multiplexing).
  - Examples: GPRS/UMTS, Internet.

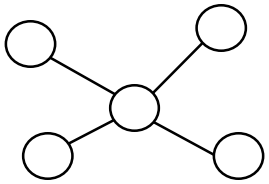
Connection Oriented vs. Connectionless

- **Connection Oriented (CO) Communications:**
  - Each communication requires first the establishment of a connection to the communication partner (signalling).
  - CO communications can be implemented on both circuit switching and packet switching system.
  - The address of the recipient is specified only at connection establishment.
  - Failures of network components lead to connection termination.

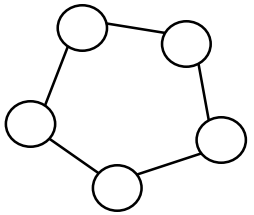
- **Connectionless (CL) Communications:**
  - Data exchange can begin at any time without connection establishment
  - CL communications can be implemented on both circuit switching and packet switching system.
  - Each dispatched message must contain address information of the recipient.
  - Failures and disturbances are less noticed but still cause loss of messages (packet loss).

Network Topologies [1/2]

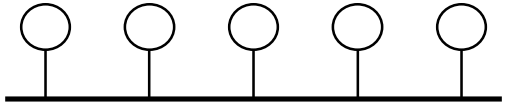
- Star
  - Simple way selection.
  - Bad reliability (single point of failure).



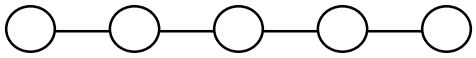
- Ring
  - Bad reliability (single point of failure).
  - Better support for network control.



- Bus
  - Stations share a common medium.
  - Good reliability.

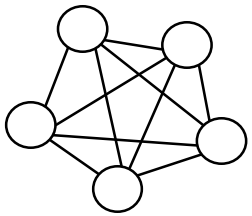


- Linear Network
  - Conceptually simple.
  - Average reliability.
  - The position in the network influences transmission times.

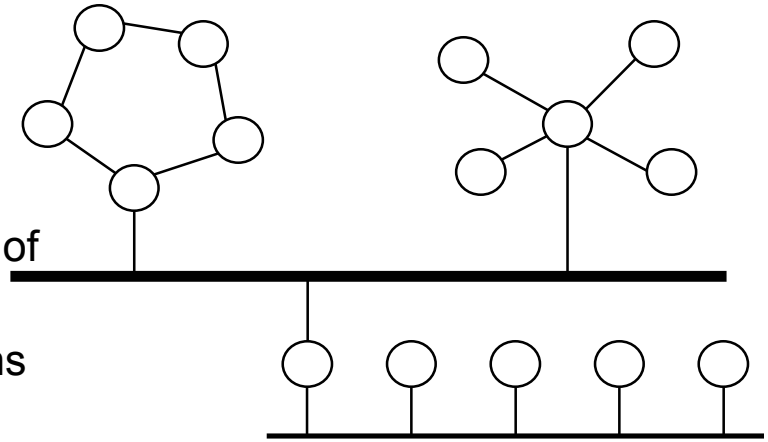


Network Topologies [2/2]

- Meshed Network
  - No switching necessarily
  - Good reliability
  - n (n-1) / 2 connections with n nodes (fully meshed)



- Backbone Network
  - Coupling of networks to larger units
  - Usually hierarchical structure
  - Reliability directly dependent on the reliability of the liaison vehicles
  - Fits well into existing hierarchical organisations



Services and Protocols: Some Definitions

- Service

  It is defined as an abstract function supplied by a network

- Service Primitive

  The individual elementary functions are called service-primitives. Typical ISO/OSI services are:

→ request	Service Request
→ indication	Indication that a service was requested
→ response	Reaction of the service to a service request
→ confirm	Acknowledgement that a requested service was provided

- Service Access Point (SAP)

  The interfaces over which the service primitive can be access as service access points.

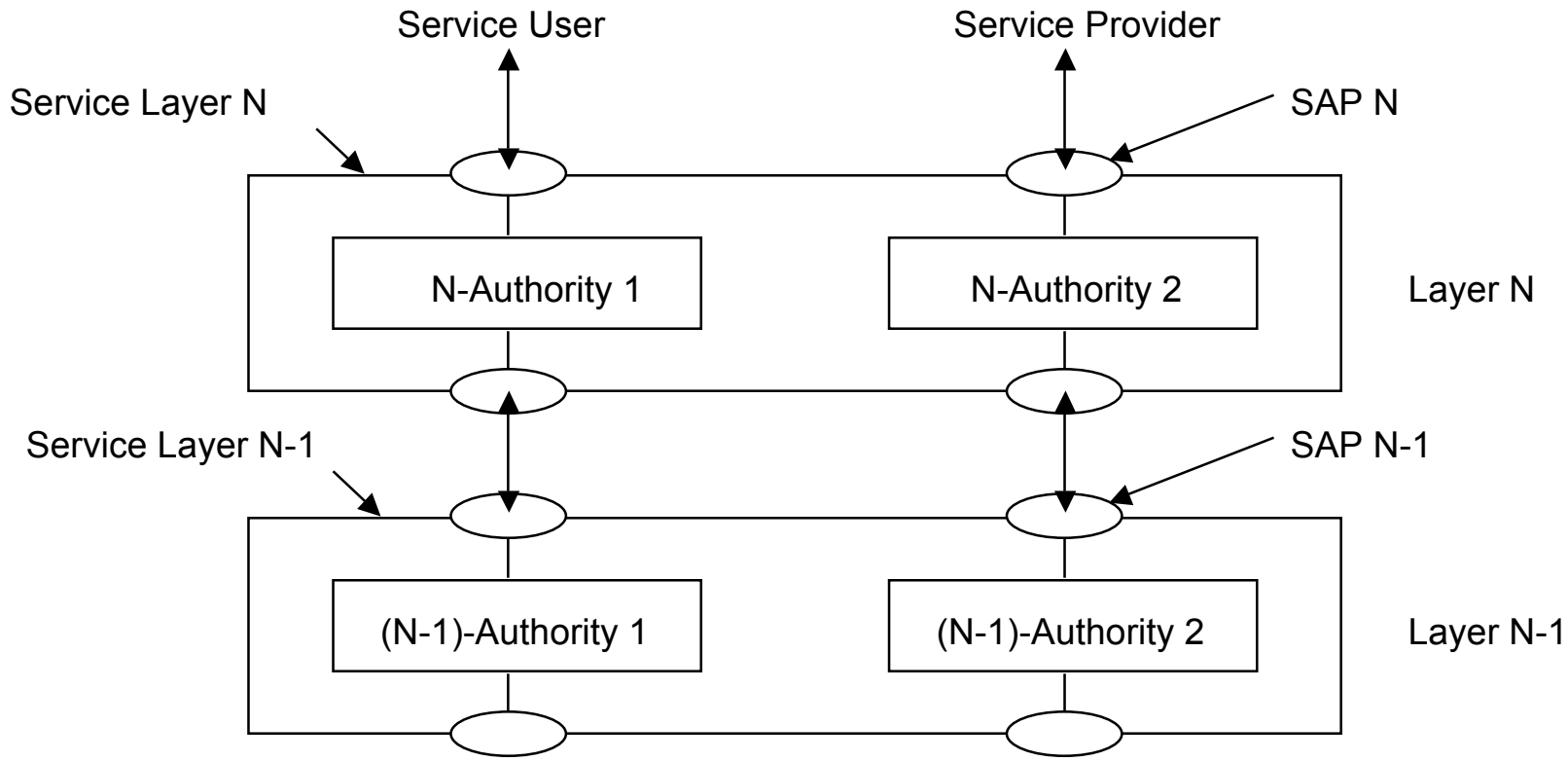
- Entities

  The services furnished by so-called instances.

- Protocol

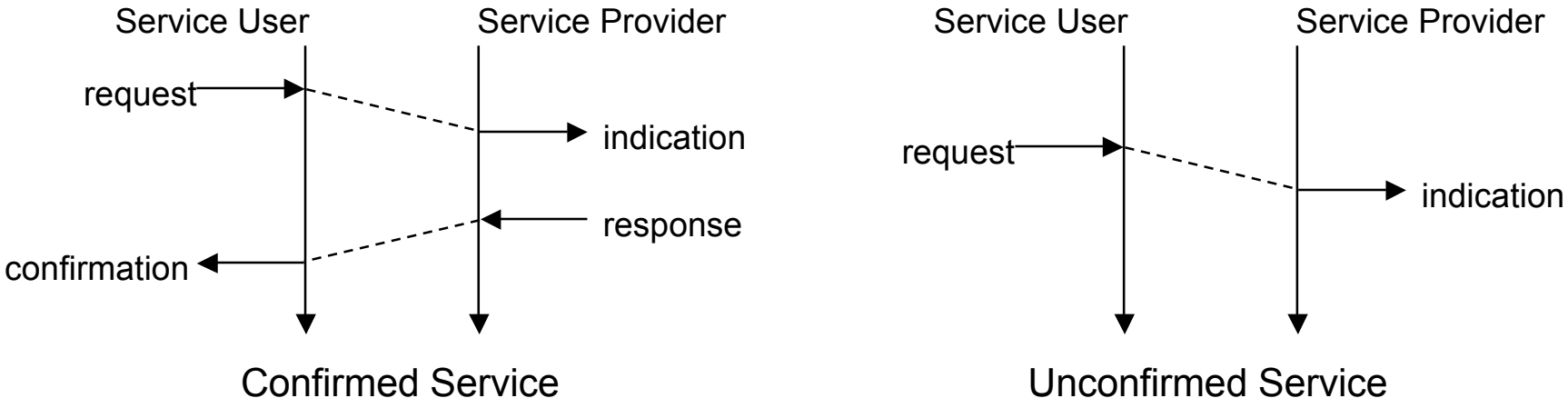
  The rules and the restrictions according to which instances interact with other instances.

Representation and Layering of Services



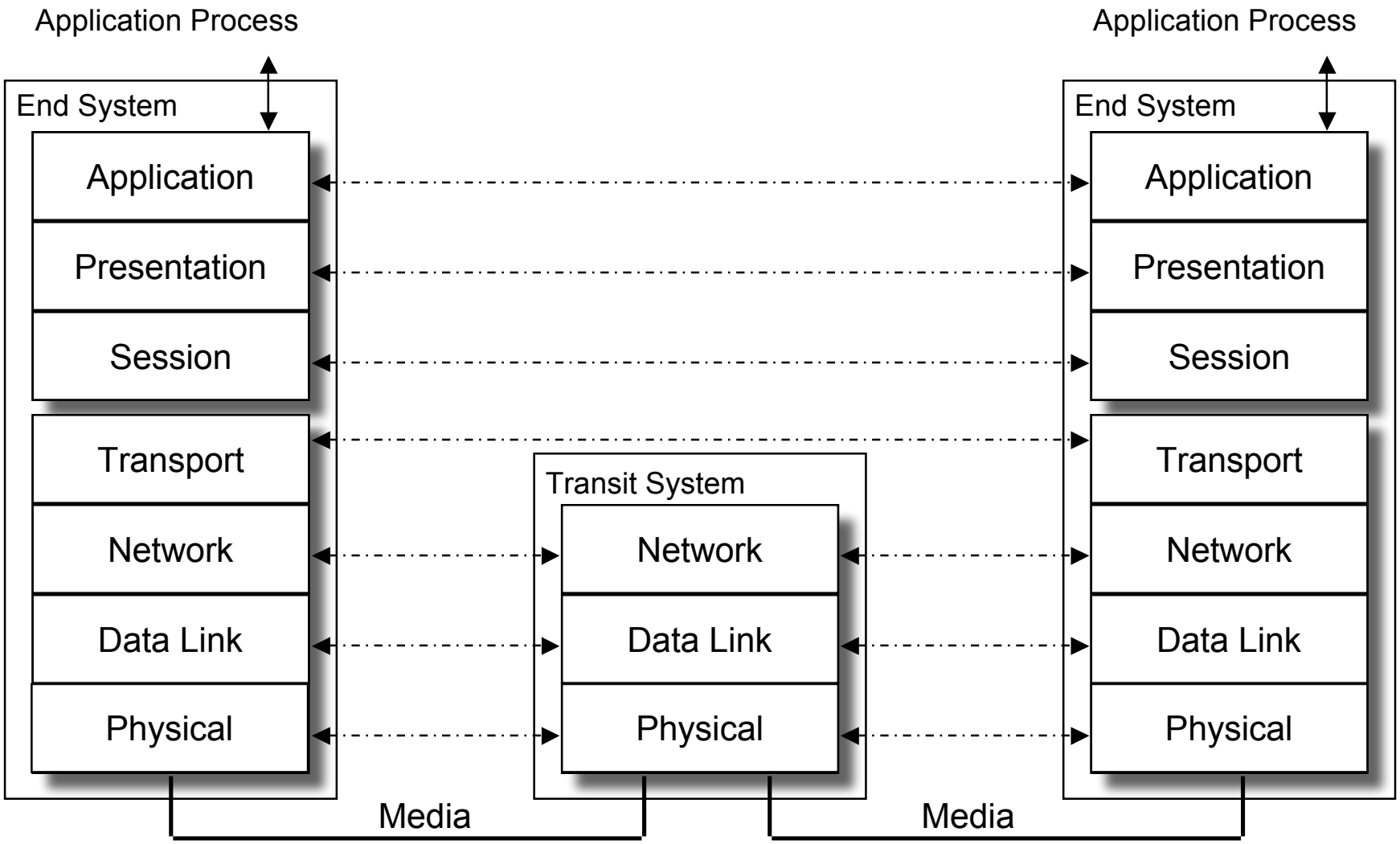
- The definition of layers is a fundamental principle for the structuring of communication systems.
- Services of a layer may only accept service primitives of services in adjacent layers.

Time Diagrams



- Time diagrams clarify the temporal and spatial connections between service primitives.
- Vertical axis are time axis, horizontal axis give the spatial distance between users and providers of services.
- Service requests of a confirmed service can result either in a positive or negative confirmation.
- Service requests of an unconfirmed service are not acknowledged.

ISO/OSI-Reference Model



ISO/OSI Transport System [1/2]

- Physical Layer
  - ↪ Transport of a stream of bits over a media.
  - ↪ Transport depending on the characteristics of the media being used.
  - ↪ Representation of values 0 and 1 (e.g. voltage levels).
  - ↪ Synchronisation between senders and recipients.
  - ↪ Definition of standard plugs for media interconnection.

- Data Link Layer
  - ↪ Transport of a frame of bits.
  - ↪ Data communication between systems that share a common media.
  - ↪ Detection and recovery of transfer errors.
  - ↪ Flow control for handling traffic peaks (traffic jam).
  - ↪ Implementation usually in hardware on adapter cards (e.g. Ethernet card).

ISO/OSI Transport System [2/2]

- Network Layer
  - Determination of a route through the network (routing).
  - Multiplex of network connections over a shared connection.
  - Error detection and recovery between end-systems.
  - Flow control between end-systems.
  - Division of a Packet in multiple frames.

- Transport Layer
  - End-to-end communication between applications.
  - Virtual connections over connectionless datagram services.
  - Error detection and recovery between applications.
  - Flow control between applications.
  - Concurrent usage of multiple services.

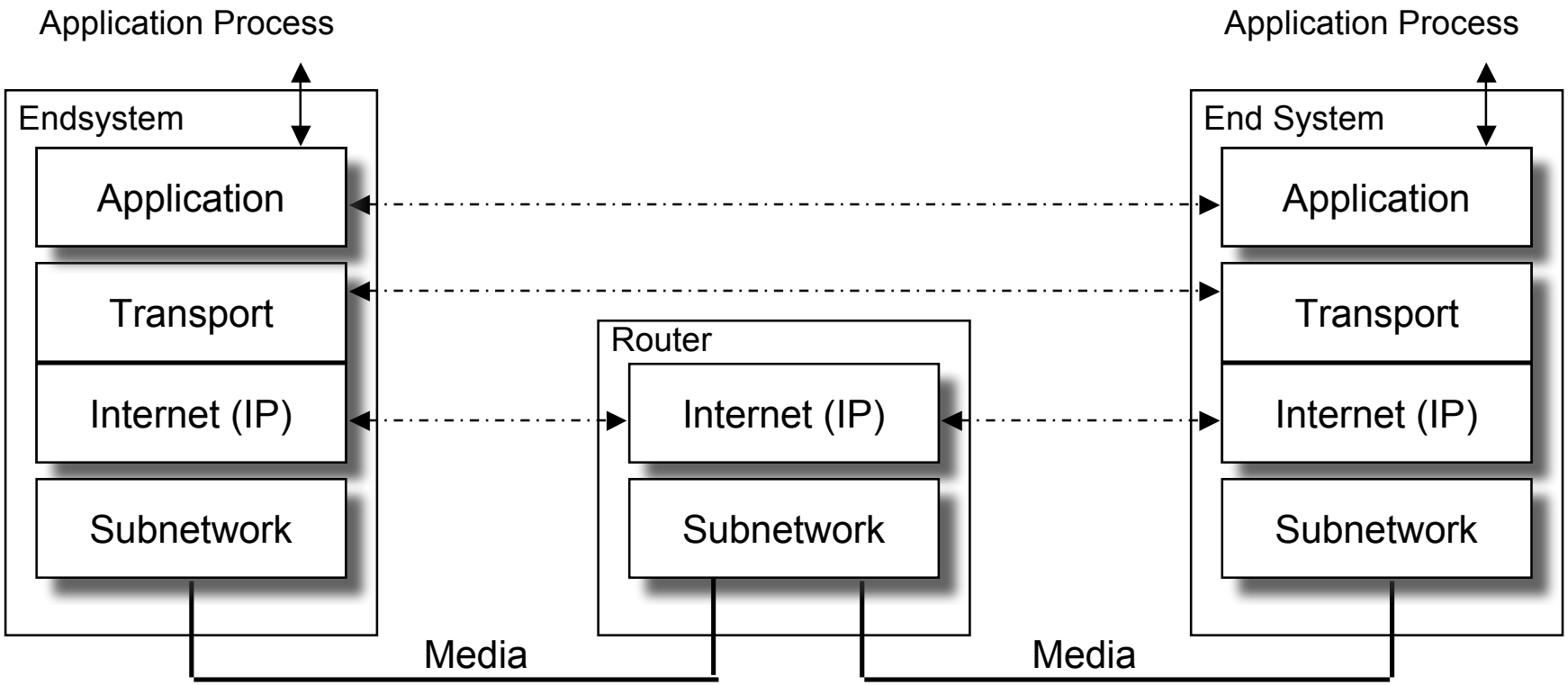
ISO/OSI Higher Layers

- Session Layer
  - Synchronisation and co-ordination of communicating processes.
  - Session control (checkpoints for recovery).

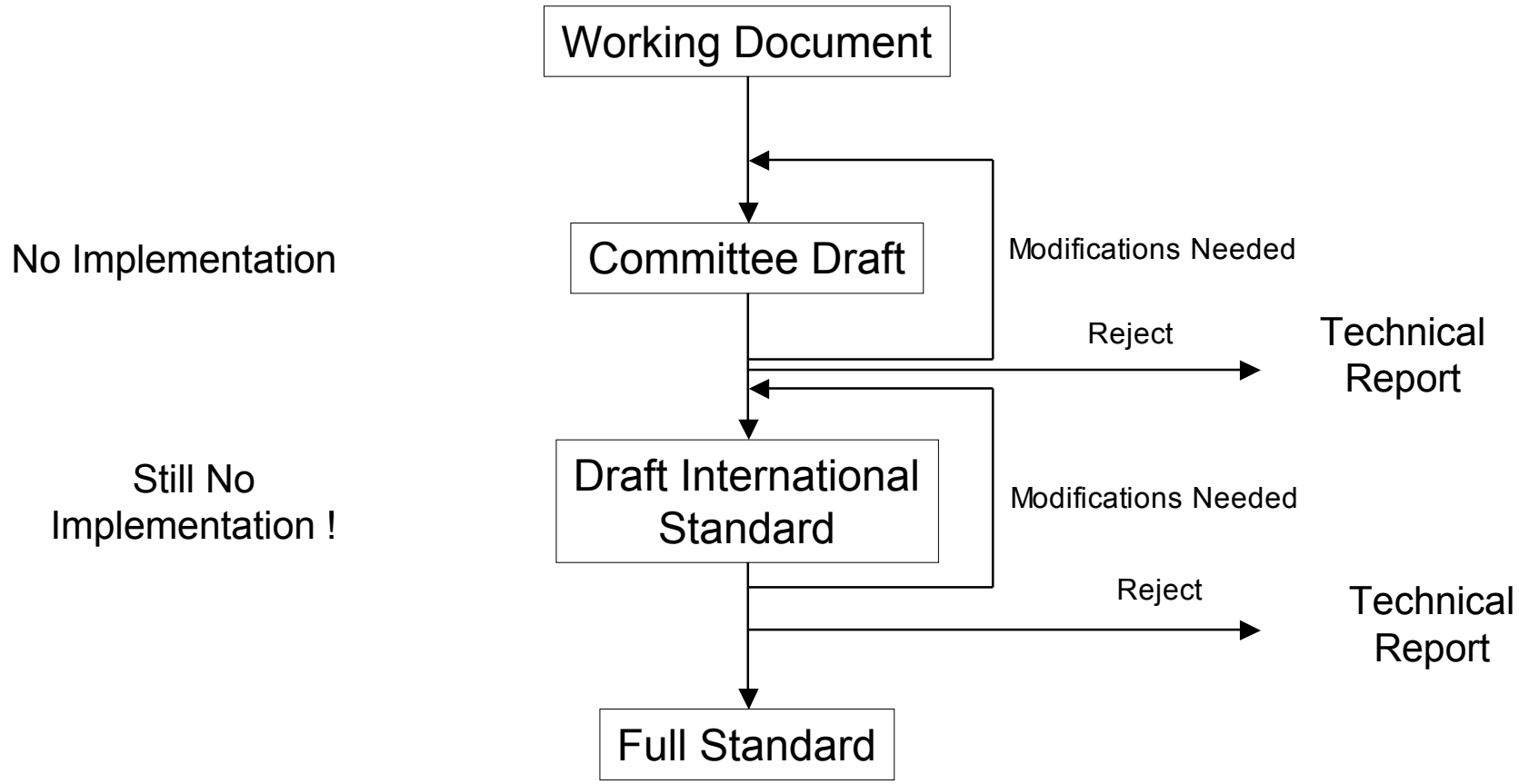
- Presentation Layer
  - Transformation and adaptation of data presentations (e.g ASCII EBCDIC).
  - Serialisation of data structures for the purpose of transfer.
  - Data compression.

- Application Layer
  - Supply of fundamental services, which can be used directly by any application including (but not limited to):
  - File transfer, virtual terminals, name space administration, database access, network management, electronic communication networks, process and print control...

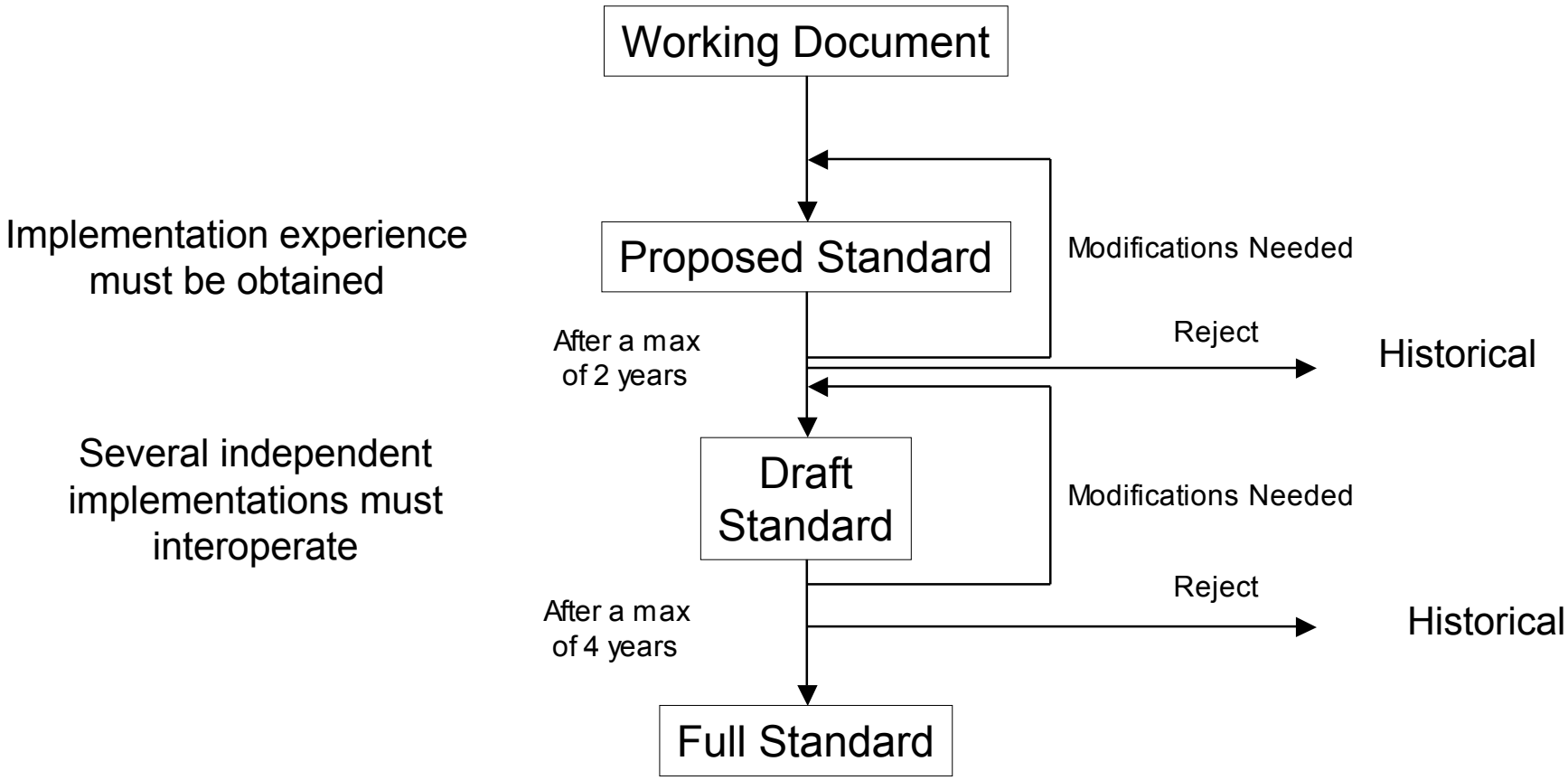
Internet Layer Model



ISO Standardisation



IETF Standardisation



1.4 Overview: Abstract Syntax Notation One

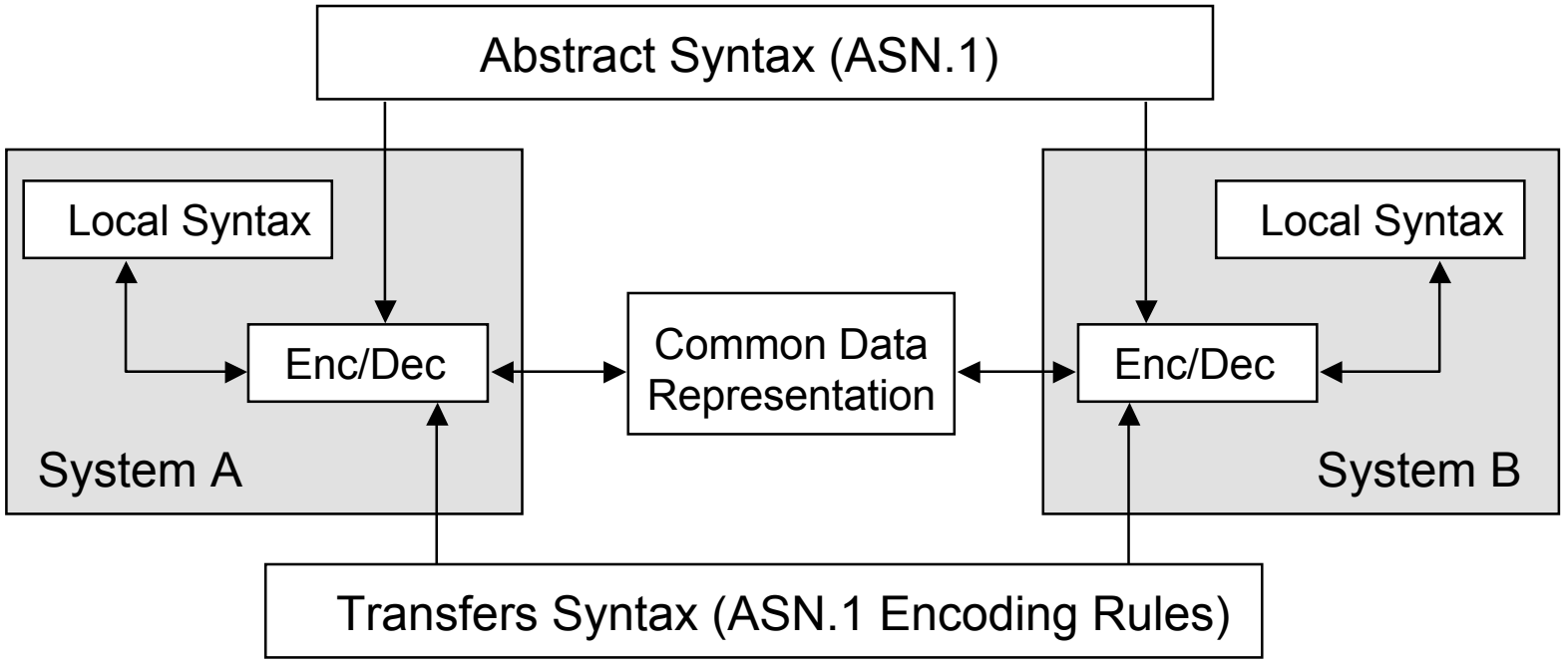
- Abstract Syntax Notation One (ASN.1) is a syntax user for the definition of data structures and message formats.

- ASN.1 goals:
  - Exchange of information between machines with different hardware architectures (8/16/32/64 bit, little/big-endian).
  - Independence from existing programming languages (language neutral).
  - Coding of the data during the transfer should be selectable between senders and recipients (negotiation).

- Separation of the data presentation from the application-specific data structure representation.

- The abstract syntax defines the data structures during the transfer and determines in which form these data structures will serially transfer over a network.

Abstract Syntax and Transfer Syntax



- ASN.1 defines a standardised abstract syntax.
- ASN.1 permits several encoding rules that transform the abstract syntax into a byte stream suitable for transfer. *BER* (Basic Encoding Rules) defines the mapping between abstract and transfer syntax.
- Applications normally use a local syntax depending on the programming language being used.

Primitive ASN.1 Datatypes

- Names of ASN.1 datatypes begin with a uppercase letter.
- Names of ASN.1 values (constants) begin with a lowercase letter.
- ASN.1 keywords and macro names consists only of uppercase letters.
- Comments are enclosed between `--` (e.g. `-- This is a comment --`).

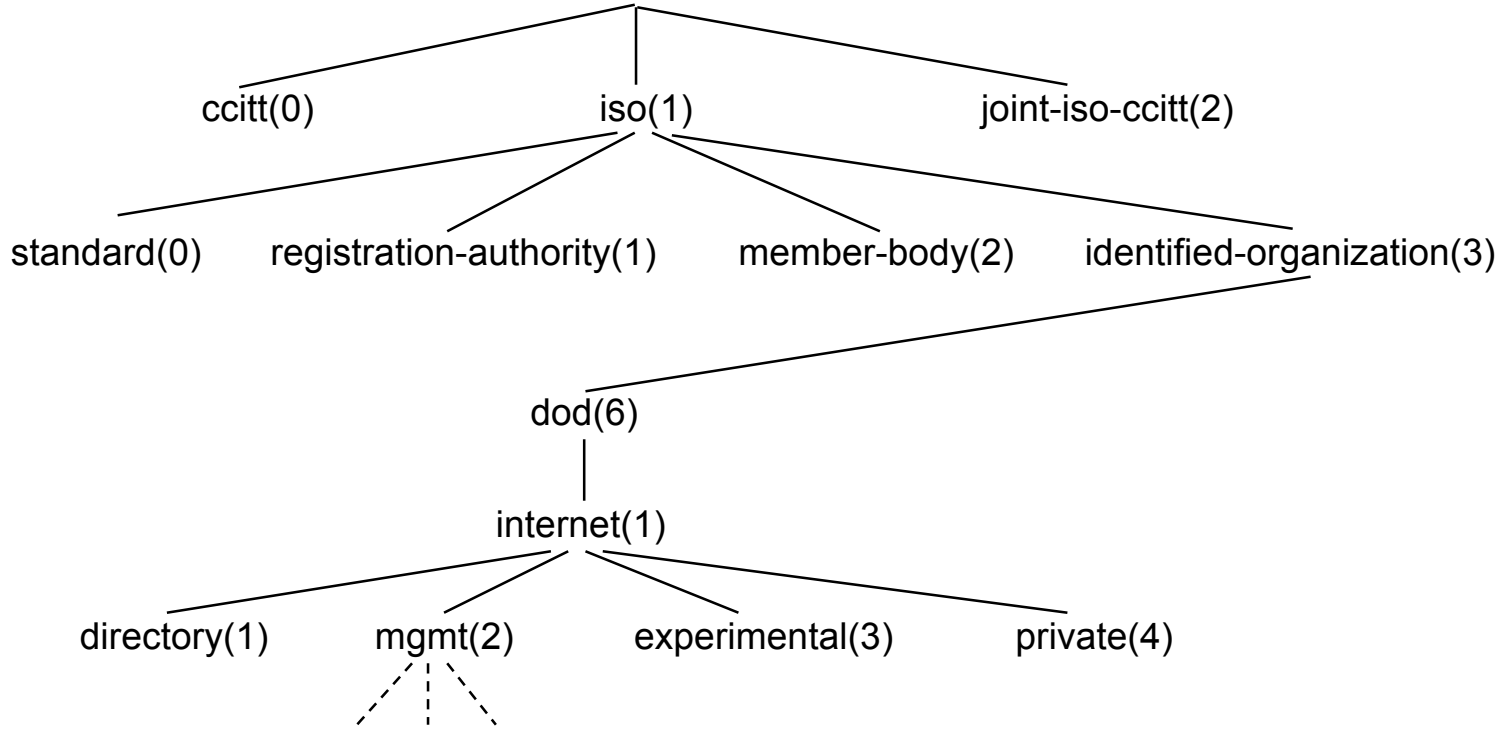
- `BOOLEAN`:
  - Can only take the predefined values TRUE and FALSE.
- `INTEGER`:
  - Covers all the possible integer numbers. No delimitation of the number range.
- `BIT STRING`:
  - A sequence of bits. The length does not have to be divisible by 8.
- `OCTET STRING`:
  - A sequence of octets (bytes). It is the base type for different character sets and other derived types (`GeneralizedTime`, `UTCTime`).

Primitive ASN.1-Datatypes

- `ENUMERATED`:
  - Type of enumerating. Possible values must be determined by the definition of derived datatypes.
- `OBJECT IDENTIFIER`:
  - Unique identification of a node in the ISO registration tree.
  - Path of the root of the tree to the target node.
- `ObjectDescriptor`:
  - A character string for the identification of a node in the Registration tree.
  - Not necessarily unique.
- `ANY`:
  - any ASN.1-datatype (Union of all ASN.1 datatypes as C ‘void’).
- `EXTERNAL`:
  - Data not described using an ASN.1 definition.
- `NULL`:
  - A substitute symbol, in order to indicate in an assembled datatype the absence of a value.

ISO Registration Tree

- Used for uniquely identifying definitions, documents, objects...
- Hierarchical structure, similar to hierarchical file systems.
- All nodes of a level identified by a unique number.
- The path from the root of the registration tree to a node results in a numerical sequence called Object Identifier (e.g. 1.3.6.1).



Assembled ASN.1 Datatypes

- `SEQUENCE`:
  - Corresponds to structures in C or records in Pascal.
  - The sequence of the items in a SEQUENCE is fixed.
- `SET`:
  - Similar to a `SEQUENCE`, with the difference that the sequence of the elements is not specified.
- `SEQUENCE OF`:
  - Ordered quantity (list) of homogeneous data.
- `SET OF`:
  - Unordered quantity of homogeneous data.
- `CHOICE`:
  - Type of selection, similar to the C union.
- `REAL`:
  - It consists of the `INTEGER` datatype extended with mantissa and exponent.

Reduced Datatypes

- Definition of further datatypes by restricting the scope of existing datatypes.
- Exact syntax dependent on the underlying primitive datatype.
- Examples:

```
LottoNumber ::= INTEGER (1..90)
MD5Key      ::= OCTET STRING (SIZE (16))
IPAddress   ::= OCTET STRING (SIZE (4|16))
Counter32   ::= INTEGER (0..4294967295)
Integer32   ::= INTEGER (-2147483648..2147483647)
Unsigned64  ::= INTEGER (0..18446744073709551615)
```

- Restrictions of the scope are applied to derived datatypes (e.g `SEQUENCE OF MD5Key`).
- The restriction of the `INTEGER` datatype makes sense as today's computers internally usually operate with 32-bit or 64-bit numbers.

Some Definitions of Types and Values

- Type definitions:

```
Number ::= INTEGER
DateAndTime ::= UTCTime
ID ::= OBJECT Identifier
```

- Value definitions :

```
ok BOOLEAN ::= TRUE
seven Number ::= 7
now DateAndTime ::= "971105012200-0100"
```

- Implicit Value Definitions :

```
Lotto ::= INTEGER { first(1), last(49) }
AccessRight ::= BIT STRING { read(1), write(2), execute(3) }
MaskAccessRight ::= { read, execute }
Sex ::= ENUMERATED { female(1), male(0) }
```

A Complex Example [1/2]

```
Message ::= SEQUENCE {
    version INTEGER,
    community OCTET STRING,
    data ANY                    -- e.g. PDUs if no authentication
}

PDUs ::= CHOICE {
     get-request            GetRequest-PDU,
     get-next-request       GetNextRequest-PDU,
     get-response           GetResponse-PDU,
     set-request            SetRequest-PDU
}

GetRequest-PDU      ::= [0] IMPLICIT PDU
GetNextRequest-PDU  ::= [1] IMPLICIT PDU
GetResponse-PDU     ::= [2] IMPLICIT PDU
SetRequest-PDU      ::= [3] IMPLICIT PDU
```

A Complex Example [2/2]

```
PDU ::= SEQUENCE {
     request-id      INTEGER,
     error-status    INTEGER {
                         noError(0), tooBig(1),
                         noSuchName(2), badValue(3),
                         readOnly(4), genErr(5)
                     },
     error-index     INTEGER,
     variable-bindings      VarBindLis
}

VarBindList ::= SEQUENCE OF VarBind

VarBind ::= SEQUENCE {
     name    ObjectName,
     value   ObjectSyntax
}
```

ASN.1 TAGS

- Identification of datatypes during the information transfer by means of tags.

- Tags consist of a tag number and a tag class. There are four classes defined:
  - UNIVERSAL:
    World-wide unique identifier (valid only within the ASN.1 standard).
  - APPLICATION:
    Unique identification within a module.
  - PRIVATE:
    For manufacturer-specific identification. Not generally used.
  - CONTEXT-SPECIFIC:
    Type identifier without class of specification (e.g. type of selection).

- Example:

```
INTEGER       identified by UNIVERSAL 2
OCTET STRING  identified by UNIVERSAL 4
```

Further ASN.1 Properties

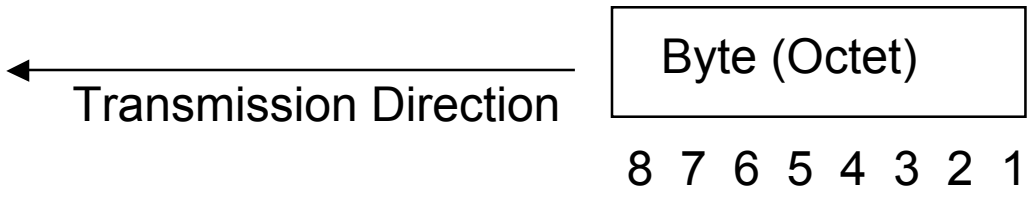
- Module Concept:
  - Explicit import of definitions.
  - Explicit export of definitions.
  - Registration of definitions in the ISO registration tree.

- Macros:
  - Efficient mechanism for the extension and description of new types.
  - Drawback: extension of the syntax makes it extremely difficult to build compilers that support the full ASN.1 syntax.
  - The macro system is very complex and often oddly used.
  - Newer developments for the simplification of ASN.1.

- Encoding Rules:
  - Basic Encoding Rules (BER) (part of the ASN.1 standard)
  - Packet Encoding Rules (PER)
  - Distinguished Encoding Rules (DER)

Basic Encoding Rules (BER)

- The Basic Encoding Rules determine how a ASN.1 datatype can be represented as a string of bytes.
- Based on tag/length/value coding (TLV) algorithm, where the each variable is identified by one tag, the length of the value in bytes and the value of those bytes.
- The TLV coding permits a recipient to reconstruct the type of a message from the received byte stream.
- BER coding is a little inefficient as there is often unnecessary information to be transferred.
- The use of OPTIONAL fields further complicated the BER definition.
- BER also defines the transmission direction of the bit stream other than the coding the ASN.1 datatypes:



Alternative Encoding Rules

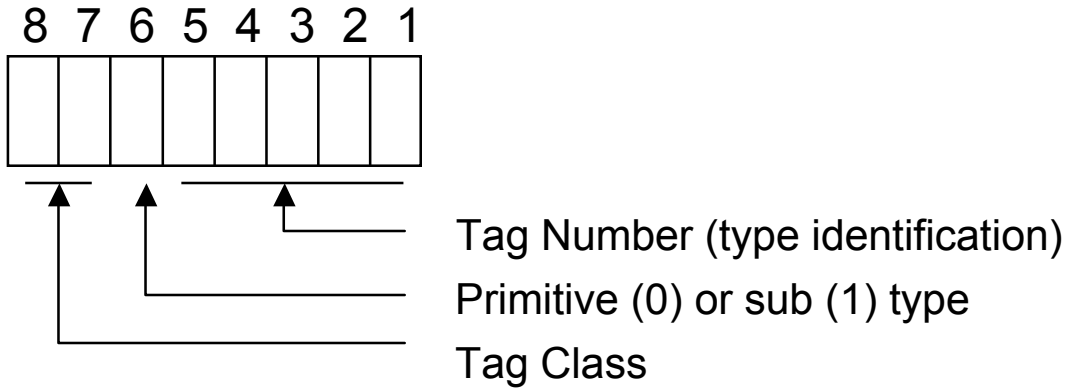
- Packed Encoding Rules (PER)
  - ↪ Very compressed encoding based on ASN.1 subtype information.

- Distinguished Encoding Rules (DER)
  - ↪ Subset of BER that gives exactly one way to represent any ASN.1 value (BER have long and short format). Used e.g. in X.509 certificates where computation of digital hash sums must be unique. Uses definite-length encoding (v.s. BER variable length).

- Canonical Encoding Rules (CER)
  - ↪ Subset of BER that gives exactly one way to represent any ASN.1 value. Unlike DER it is based on indefinite-length encoding.

Coding Tags Classes

- Each tags is coded in a byte:



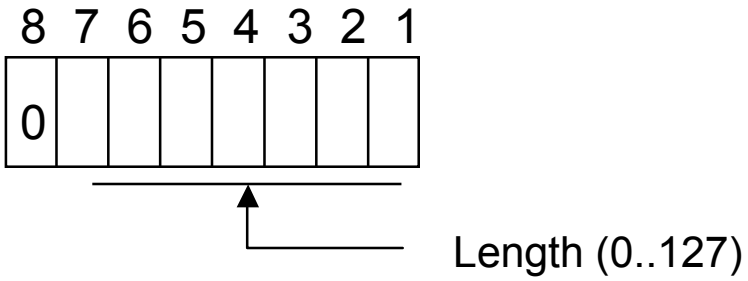
- Tag classes:

	Bit 8	Bit 7
UNIVERSAL	0	0
APPLICATION	0	1
CONTEXT-SPECIFIC	1	0
PRIVATE	1	1

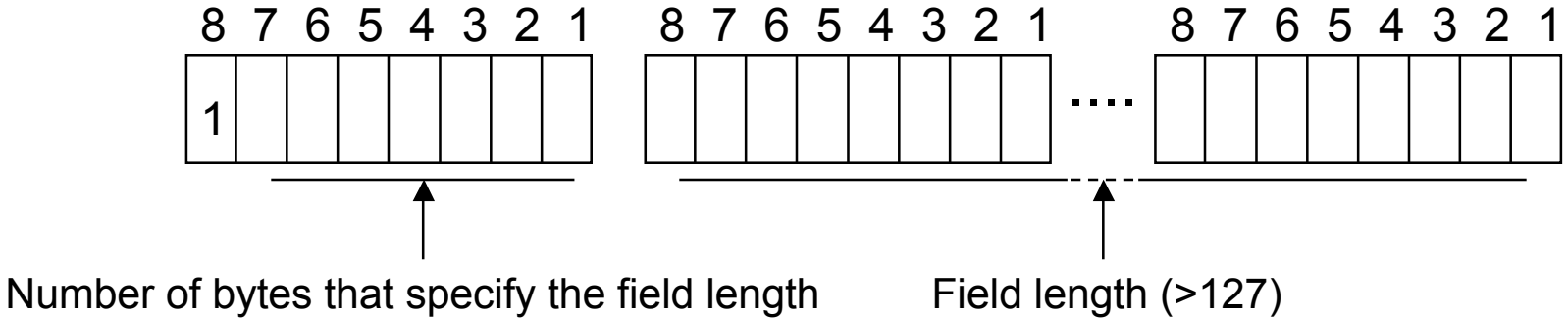
Coding Field Length

- The length field indicates the length of the directly following value.

  ↪ Length within 0..127:



  ↪ Length > 127 :



Value Coding

- For each primitive ASN.1 type there is a rule that allows values to be translated into a stream of bytes and vice-versa.
- The rules for `INTEGER` and `OCTET STRING` are simple.
- The rules for `OBJECT IDENTIFIER` are relatively complex.
- Assembled values (`SEQUENCE`, `SEQUENCE OF`) are easily represented by coding each individual item.
- With `CHOICE` constructs only the available value is transferred, therefore the associated tag must be unique.
- For further details:

  D. Steedman: *Abstract Syntax Notation One (ASN.1) - The Tutorial and Reference*, Technology Appraisals, 1990

Example of a BER Coded Message

```
30 1B                                  SEQUENCE, Length 27
   02 01 00                            INTEGER, Length 1, "0"
   04 06 70 75 62 6C 69 63             OCTET STRING, Length 6, "public"
   A1 0E                               GetNextRequest-PDU, Length 14
      02 04 36 A2 8F 07                INTEGER, Length 4, "916623111"
      02 01 00                         INTEGER, Length 1, "0"
      02 01 00                         INTEGER, Length 1, "0"
      30 00                            SEQUENCE OF, Length 0
```

- Length of the BER encoding must be well known (no dummy values) when a value is coded. With some restrictions it is also possible to specify the length after the value.
- The decoding is more difficult when the length is specified after the value.
- Coding the primitive values is not always as simple as in the example (some datatypes can be encoded in both short and long form).

2. Internet Management

1. Introduction
2. **Internet Management**
   - 2.1 Overview
   - 2.2 Structure the Management Information (SMIv2)
   - 2.3 Fundamental MIBs
   - 2.4 Simple Network Management Protocol Version 1 (SNMPv1)
   - 2.5 Simple Network Management Protocol Version 2c (SNMPv2c)
   - 2.6 Simple Network Management Protocol Version 3 (SNMPv3)
   - 2.7 MIB Implementation and Agent Extensibility Protocol (AgentX)
3. OSI-Management
6. Current Developments

2.1 Overview

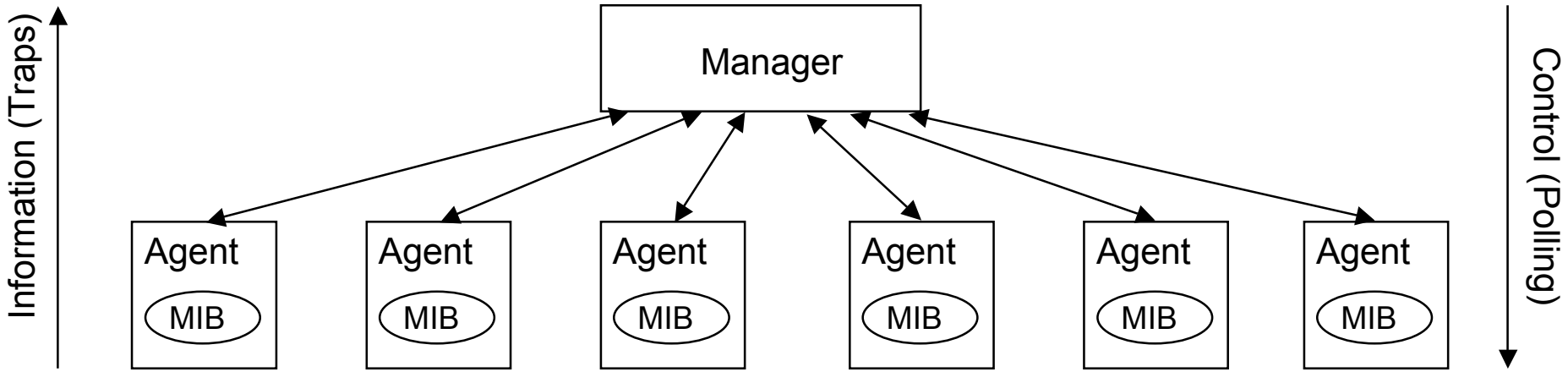
1987	Simple Gateway Monitoring Protocol (SGMP)	
1987	High-level Entity Management System (HEMS)	
1988	Simple Network Management Protocol (SNMPv1)	proposed
1990	Simple Network Management Protocol (SNMPv1)	standard 15, 16
1991	Management Information Base II	standard 17
1993	SNMP Version 2 (Party/Party/Context)	historical
1996	SNMP Version 2 (Communities)	draft/experimental
1998	SNMP Version 3 (User-based)	draft

- SNMPv1 has a large spreading particularly in data communication.
- The attempts for the standardisation of SNMPv2 failed.
- SNMPv3 with SNMPv1 has been accepted by a large community of network manufacturers.
- The user community has accepted SNMPv3 very well in terms of support and development.

SNMP Development Goals

- Minimisation of the number and complexity of the management functions, which are implemented by an agent:
  - ↪ Reduction of development costs for management agents (simple applications).
  - ↪ Ubiquity: use the same management technology for all devices (printers or Cray).
  - ↪ Application extensibility: development of new management functions without the need to modify the agents.
- Extensibility by defining new MIBs.
- Independence from existing computer or network architectures.
- Robustness by a simple, connectionless transport service (UDP).
- No dependency on other network services.
- Addition of management to new/existing devices/applications should be inexpensive, simple to develop and of limited functionality.
- Unfortunately some of these original goals have been lost: the term "simple" refers to the protocol and not to the specifications or the implementation of management applications.

Trap Directed Polling



- SNMP managers polls in regular intervals the SNMP agents.
- Agents can signal exceptional cases to a manager by sending a trap.
- The SNMP manager can adapt the polling strategy upon the receipt of traps (trap directed polling).

- SNMP is a strictly centralised model, where the manager implements the whole functionality and responsibility.

SNMP Application Areas

- SNMP can be used not only for network management:
  - control and monitoring of production processes.
  - control and monitoring of complex computer systems.
  - monitoring of complex application programs (relational databases, SAP R/3 components...).

- Many good SNMP toolkits are available on the market.

- Very few applications are available for solving complex management problems.

- The implementation of special applications or the conversion of local procedure guidelines is generally relatively complex and expensive.

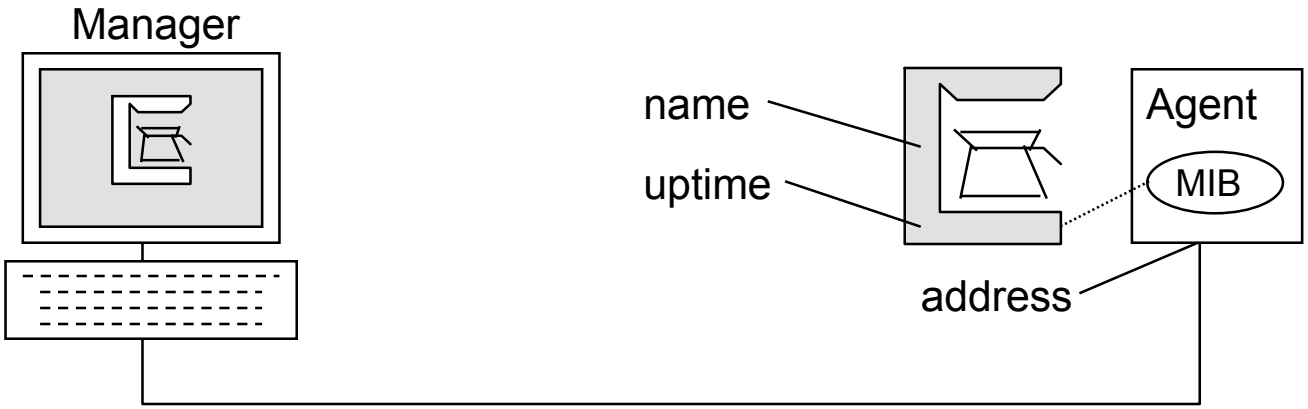
2.2 Structure the Management Information (SMIv2)

- The current information model known as "*Structure of Management Information* version 2" (SMIv2) is defined and based on simple typed variables.
- SMIv2 is based on extended subset of ASN.1 (1998).
- Each variable has a primitive, not assembled ASN.1 datatype:
  - `INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL`
  - `Integer32, Unsigned32, Gauge32, Counter32, Counter64, IpAddress, TimeTicks, Opaque`
- It does not implement complex data structures and operations on the variables.
- Variables are either scalars (exactly one instance) or columns in a “conceptual” two dimensional table (zero or several variables).
- On the variables only "read" and "write" operations can be applied. However the SNMP protocol permits the manipulation of lists of variables.
- SMIv2 management information Bases (MIBs) are defined using special ASN.1 macros.
- It leverages the complexity of new MIBs definitions: definition of basic functionality and primitive types to be used in new MIBs.

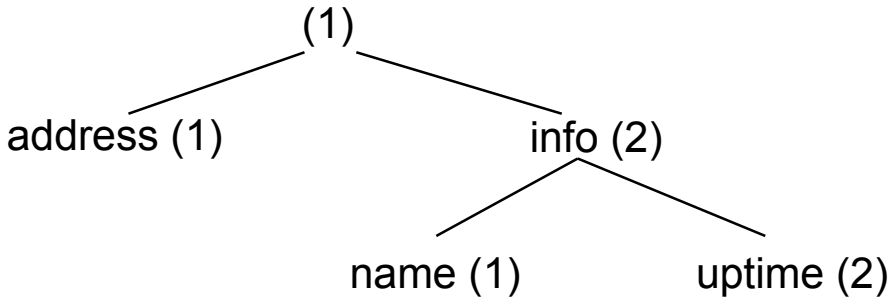
SMIv2 Basic Datatypes (RFC 2578)

SMIv2	SMIv1	Description
`INTEGER`	`INTEGER`	Integer Numbers (-2147483648..2147483647)
`OCTET STRING`	`OCTET STRING`	Sequence of bytes (octets).
`OBJECT IDENTIFIER`	`OBJECT IDENTIFIER`	Unique identifier.
`Integer32`	`INTEGER`	32 bit Integers (-2147483648..2147483647)
`Unsigned32`	-	32 bit Positive Integers (0..4294967295)
`Gauge32`	`Gauge`	"Thermometer" Integer (0..4294967295)
`Counter32`	`Counter`	32 bit non decreasing counter (0..4294967295)
`Counter64`	-	64 bit non decreasing counter (0..18446744073709551615)
`TimeTicks`	`TimeTicks`	Time in 1/100th of seconds
`IpAddress`	`IpAddress`	4 Byte IPv4 Address
`Opaque`	`Opaque`	Unspecified ASN.1 Type (not recommended)
`BITS`	-	Bits in a `OCTET STRING`
-	`NetworkAddress`	Network Address (not recommended)

A MIB Use Case



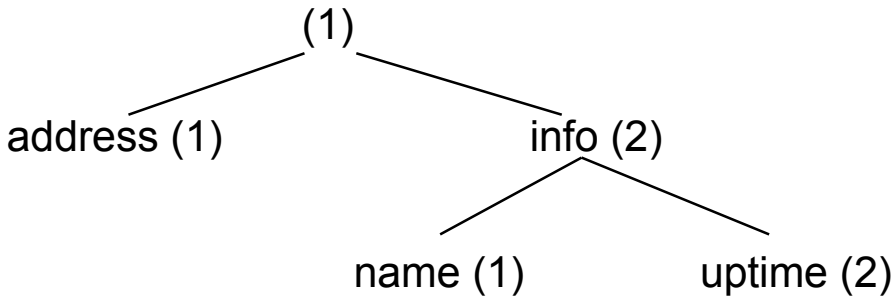
- Definition of the variables in the ISO Registration tree.
- Nodes are defined for naming purposes.
- The leave of the tree represent the managed objects (i.e. “the meat”).
- Sub nodes can be used in order to logically organise the object types.



Object Identifier and Instance Identifier

- In the registration tree each object can be identified by means of a unique object identifier.
- Concrete developments (instance) of a type of object are unique designated by a so-called *Instance Identifier*.
- A unique instance identifier is obtained by attaching an instance identifiers to the object identifier.
- Scalar object have basically only one instance, where the instance identifier has basically the value 0 (e.g. sysName.0).
- Instance identifiers for non-scalar variables are derived from the unique naming of a conceptual table.
- As object identifier can have up to 128 elements, hence instance names cannot be infinitely complex.

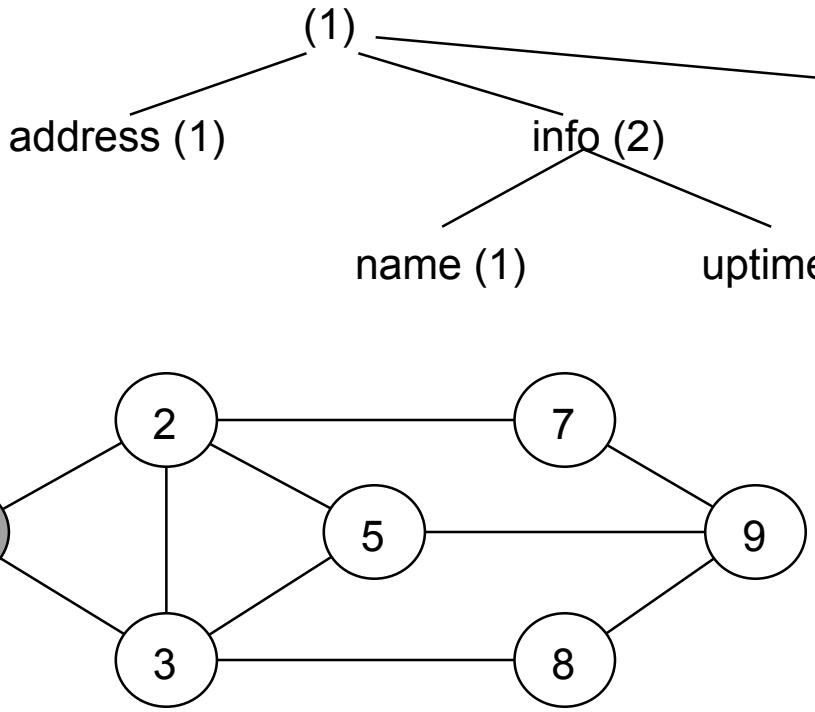
Example of Object and Instance Identifiers



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

- MIB nodes names are relevant for human users only.
- Descriptors must be unique within a MIB module, although can be used several times in different MIB modules (one gets unique descriptors by the combining module names and descriptors).

Extension of the Example MIB with a Routing table



index(1)	mask(2)	next(3)
1	2	2
2	3	3
3	5	2
4	7	2
5	8	3
6	9	3

- For matter of simplicity in the above example addresses are represented using natural numbers.

Identification of Table Entries

- Tables are defined basically with two "auxiliary nodes":
  - ↪ the first node defines the table and is of type `SEQUENCE OF`.
  - ↪ the second node defines an entry (a row) in the table and is of type `SEQUENCE`.
  - ↪ this is the only permitted use of `SEQUENCE` and `SEQUENCE OF` in SNMP SMIv2.

- The result of the column and instance identifier (code of the table) is a unique object identifier for each table entry.

- Table Example (convention OID => value):

1.3.1.1.1 => 1	1.3.1.3.1 => 2	1.3.1.2.4 => 7
1.3.1.2.1 => 2	1.3.1.1.4 => 4	1.3.1.2.7 => not existing

Tables Naming [1/3]

- Table naming is very important as it affects the way tables are accessed.
- Two kind of tables naming:
  - ↪ Use row numbers (not being used by SNMP).

1	2	2
2	3	3
3	*5*	*2*
4	7	2
5	8	3

← This is row number 3

  - ↪ Use an index column (the SNMP way).

This is the index column ↓

1	2	2
2	3	3
3	5	2
4	7	2
5	8	3

Tables Naming [2/3]

- A table index is not necessarily an INTEGER. For instance the routingTable uses an IP address as table index.
- A table index can be made of several components:
  - X . C . I1 . I2 …… In

OID of the table

Column number

Index value 1

Index value n

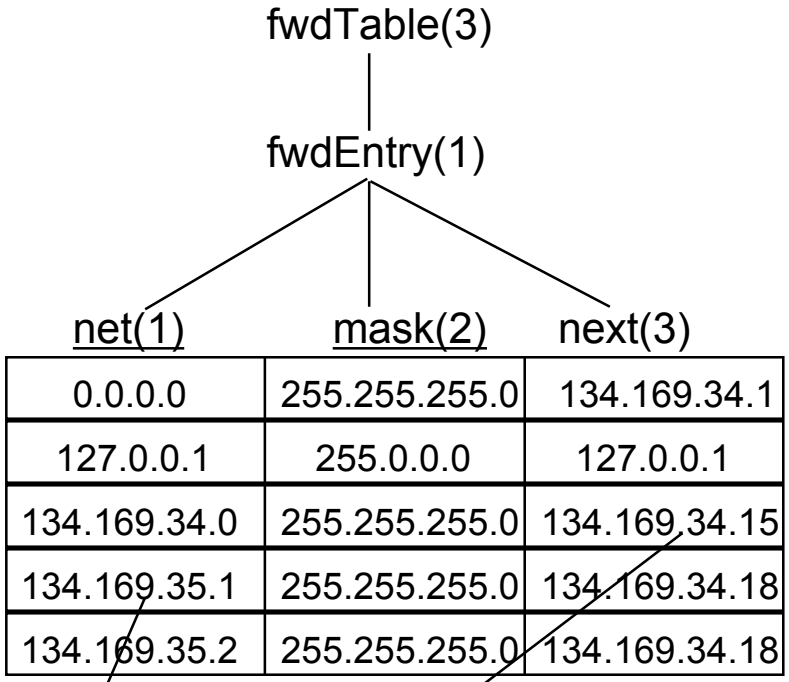
routingTable

policy (2) 1 = low cost, 2 = high reliability

destination (1)		next (3)
130.89.16.23	**1**	130.89.16.23
130.89.16.23	**2**	130.89.16.127
192.168.10.12	**1**	172.16.1.18
192.168.10.12	**2**	172.16.1.12

Tables Naming: Complex Table Indexes [3/3]

- An IP Routing table is the combination of IP address and the IP netmask necessary to satisfy the routing rules.
- The individual bytes of the IP address are specified as individual sub identifiers.
- Example:



Instance Identifier

1.3.1.1.134.169.35.1.255.255.255.0 => 134.169.35.1

net mask

1.3.1.3.134.169.34.0.255.255.255.0 => 134.169.34.15

Rules for the Specification of Instance Identifier values

- Values for fundamental types:
  - ↪ Values for `INTEGER:`

    A single integer value.
  - ↪ Values for fixed length `OCTET STRING:`

    Each individual byte is treated as an individual value.
  - ↪ Values for variable length `OCTET STRING:`

    The first value is the length, followed by each individual byte.
  - ↪ Values for `OBJECT IDENTIFIER`:

    The first value is the length, followed by each individual byte.

- The `IMPLIED` keyword can be used without the length byte if it does not lead to ambiguities.

- The max length of `OBJECT IDENTIFIER` values is limited to 128 items, so instance identifiers will not be arbitrary complex.

MIB Module

- Similar object types are combined into MIB modules.
- Each MIB module must have a unique name (uppercase letters).
- MIB modules are (almost) normal ASN.1 modules and obey to the lexical ASN.1 rules.
- Definitions can be imported by other MIB modules with the help of of the ASN.1 `IMPORT` statement.
- All used ASN.1 SMI Macros must be explicitly imported

```
COFFEE-MIB DEFINITIONS ::= BEGIN

IMPORT      MODULE-IDENTITY, OBJECT-TYPE, enterprises,
            IpAddress, TimeTicks   FROM SNMPv2-SMI;
...

END
```

Module-Identities (RFC 2578)

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

- Defines administrative information e.g. contact information and version number.
- the `REVISION` and `DESCRIPTION` clauses are not mandatory and can occur several times.
- ExtUTCTime contains a date in the format„YYMMDDHHMMZ“ (UTC) or „YYYYMMDDHHMMZ“, e.g.. „9502192015Z“ or „199502192015Z“.

Example of Module Identities (RFC 2233)

```
IF-MIB DEFINITIONS ::= BEGIN

IMPORTS ...

ifMIB MODULE-IDENTITY
    LAST-UPDATED "9611031355Z"
    ORGANIZATION "IETF Interface MIB Working Group"
    CONTACT-INFO "  Keith McCloghrie       408-526-5260
                    Cisco Systems, Inc.    kzm@cisco.com
                    170 West Tasman Drive
                    San Jose, CA 95134-1706, US"
    DESCRIPTION  "The MIB module to of describe generic objects for network interface
                  sub-layers. This MIB is an updated version of MIB II´s ifTable,
                  and incorporates the extensions defined in RFC 1229."
    REVISION     "9602282155Z"
    DESCRIPTION "Revisions made by the Interfaces MIB WG"
    REVISION     "9311082155Z"
    DESCRIPTION "Initial revision, published as part of RFC 1573."
    ::= { mib-2 31 }

...

END
```

Object Identities (RFC 2578)

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

- Defines and registers an object identifier value.
- Permits the allocation of any node within the registration tree.
- The `STATUS` clause defines whether the allocated node is "obsolete" "current", or "deprecated".
- The optional `REFERENCE` is used to refer to further information (similar to HTML hyperlinks).

Example of Object Identities (RFC 2578, RFC 1906)

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

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

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

Object Types (RFC 2578)

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

- Macro for the definition of object types and conceptual tables.
- The `INDEX` and `AUGMENTS` clauses are permitted only for the definition by tables.
- Exactly one of the above clauses must be specified during table definition.

Example for ObjectTypes (RFC 2012)

```
tcpRtoMin OBJECT-TYPE
    SYNTAX      Integer32
    UNITS       "milliseconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The minimum value permitted by a TCP implementation for the
            retransmission timeout, measured in milliseconds.  More
            refined semantics for objects of this type depend upon the
            algorithm used to determine the retransmission timeout.  In
            particular, when the timeout algorithm is rsre(3), an object
            of this type has the semantics of the LBOand quantity
            of thecribed in RFC 793."
    ::= { tcp 2 }
```

Example for ObjectTypes (RFC 1907)

```
sysORTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF SysOREntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The (conceptual) table lising the capabilities of the
            local SNMPv2 entity acting in an agent role with respect to
            various MIB modules.  SNMPv2 entities having dynamically-
            configurable support of MIB modules will have a
            dynamically-varying number of conceptual rows."
    ::= { system 9 }

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

Notification-Types (RFC 2578)

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

- Macro for the registration of an event.
- In case of event a manager or an agent can send an appropriate notification to another manager.
- The `OBJECTS` clauses defines which MIB objects must be contained in the event description.
- The `DESCRIPTION` clause must describe which instances are meant in each case.

Example for Notification Types (RFC 2233)

```
linkDown NOTIFICATION-TYPE
    OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS  current
    DESCRIPTION
         "A linkDown trap signifies that the SNMPv2 entity,
         acting in an agent role, has detected that the
         ifOperStatus object for one of its communication links
         is about to enter the down state from some other state
         (but not from the notPresent state).  This other state
         is indicated by the included value of ifOperStatus."
    ::= { snmpTraps 3 }

linkUp NOTIFICATION-TYPE
    OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS  current
    DESCRIPTION
         "A linkDown trap signifies that the SNMPv2 entity,
         acting in an agent role, has detected that the
         ifOperStatus object for one of its communication links
         left the down state and transitioned into some other
         state (but not into the notPresent state).  This other
         state is indicated by the included value of ifOperStatus."
    ::= { snmpTraps 4 }
```

New Types from Textual Conventions

- Textual conventions allow new types to be derived from SMIv2 base types.
- However, additional types may not be derived from a textual convention.
- A `DISPLAY-HINT` clause defines a simple figure of the ASN.1 representation of a value into a format readable for humans.
- The `DISPLAY-HINT` clause can be used only together with the `INTEGER` and `OCTET STRING` datatype and from which it derives.
- A Textual convention can determine restrictions on the scope.
- A Textual convention cannot define an assembled type.

Textual Conventions [1/2]

- Textual conventions are defined in RFC 2579.

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

- The `DISPLAY-HINT` clause defines a bi-directional figure of the internally used representation on a representation readable for humans. .

- In the `SYNTAX` clause only base datatypes may be used (one can thus limit not existing Textual Conventions even further).

- All further semantics must be defined in the `DESCRIPTION` clause.

Textual Conventions [2/2]

- The followings are the textual conventions defined in RFC 2579:
  - PhysAddress
  - MacAddress
  - TruthValue
  - AutonomousType
  - InstancePointer
  - VariablePointer
  - RowPointer
  - RowStatus
  - TimeStamp
  - TimeInterval
  - DateAndTime
  - StorageType
  - TDomain
  - TAddress

`INTEGER` DISPLAY-HINTS

Format	Description
d	Representation of an Integer
d-<number>	Representation of `d` with a decimal point
o	Octal Representation
x	Hex Representation

- Example:
  - ↳ ´´d´´ stands for ´´143´´
  - ↳ ´´d-2´´ stands for ´´1.43´´
  - ↳ ´´o´´ stands for ´´217´´
  - ↳ ´´x´´ stands for ´´8F´´

OCTET STRING DISPLAY-HINTS

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

Field	Description (similar to C/C++ `printf`)
<repeat>	Indicator for the specification repetition
<number>	# bytes in the following format field
<format>	Format (a ASCII, d Decimal, x Hexadecimal, o Octal, t UTF8)
<separator>	Separator among multiple values
<terminator>	Terminator specified at the end of the rule

- Example:
  - ´´255a´´ format for the ASCII characters ´´aBc´´ in the string ´´aBc´´
  - ´´1x:´´ format for the ASCII characters ´´aBc´´ in the string ´´61:42:63´´
  - ´´0aH0ae0a10a10ao0a 1a´´
    format for the ASCII characters ´´World´´ in the string ´´Hello World´´

Example for Textual-Conventions (RFC 2579)

```
RunState ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
            "This TC of describes the current execution state of
            a running application or process."
    SYNTAX       INTEGER {
                     running(1), runnable(2),
                     waiting(3), exiting(4), other(5)
                 }

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

Example for Textual-Conventions (RFC 2579)

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

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

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

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

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

Further SMIv2 Macros

- `OBJECT-GROUPS`
  - ↪ It enables the definition of groups of related object types.
  - ↪ This macro can be used in the `MODULE-COMPLIANCE` macro.

- `NOTIFICATION-GROUPS`
  - ↪ It enables the definition of groups of related notification types.
  - ↪ This macro can be used in the `MODULE-COMPLIANCE` macro.

- `MODULE-COMPLIANCE`
  - ↪ It defines one or more constraints that a MIB implementations must fulfil.

- `AGENT-CAPABILITIES`
  - ↪ It describes the capabilities of a real MIB implementation.

2.3 Fundamental MIBs

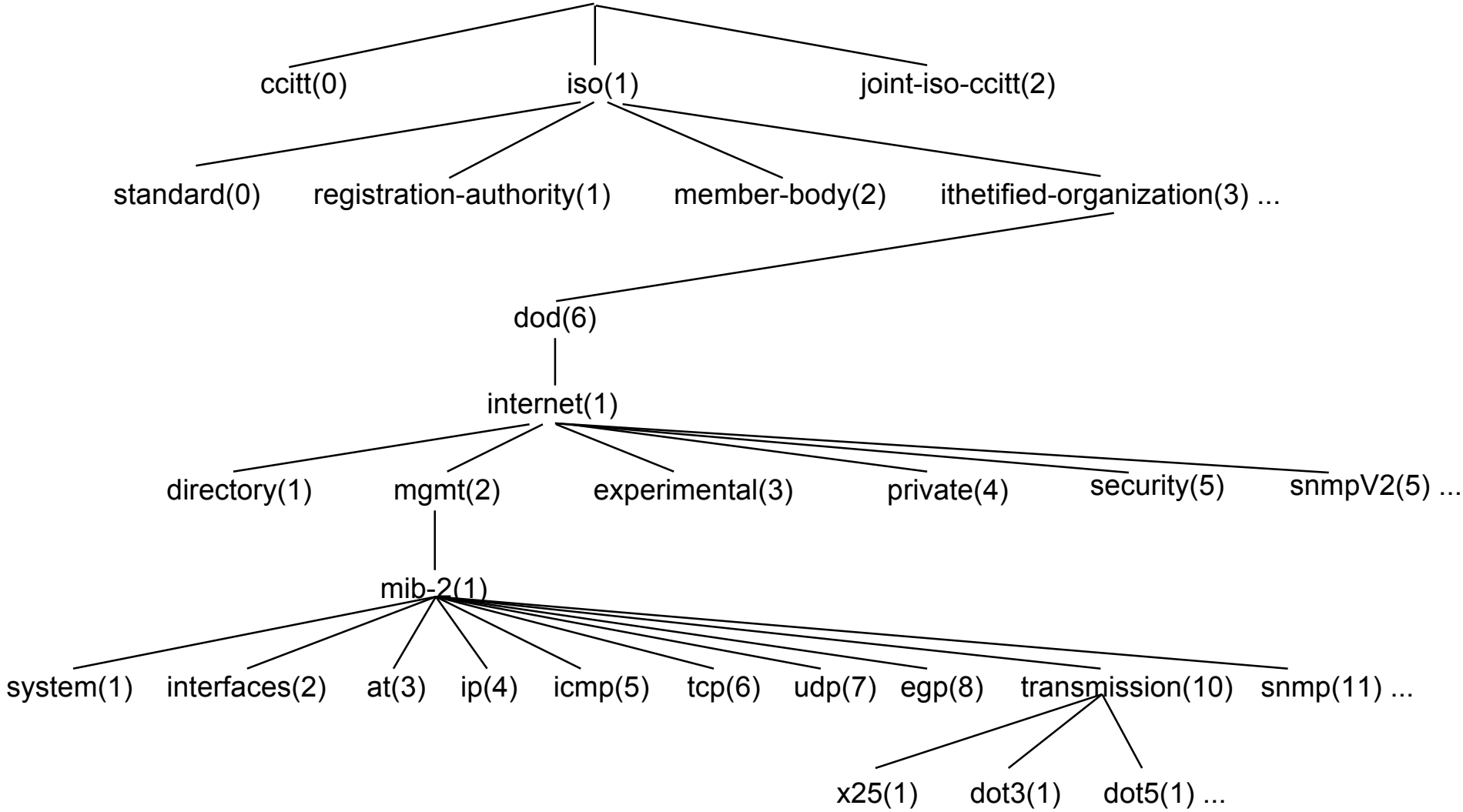
- MIB-II (RFC 1213) defines object types for the Internet Protocols IP, ICMP, UDP, TCP, SNMP (and other definitions not relevant here). Basically it models the management of the TCP/IP protocol stack.

- Goals of the MIB-II definition:
  - Define basic error and configuration management for Internet protocols.
  - Very few and weak control objects.
  - Avoidance of redundant information in the MIB.
  - MIB implementation should not interfere with the normal network activities.
  - No implementation-dependent object types.

- Altogether 170 object types.

- Some MIB definitions turned out to be too simple and minimal (Routing table, Interface table).
- Some MIB definitions presuppose a 4-Byte address format, hence these tables must be redefined for IP version 6 (IPv6).

Registration and Structure of MIB-II



Remarks on MIB-II

- The “transmission” branch accommodates all the MIB definitions that deals with information transmission (X.25, PPP, RS232, SONET, ISDN, IEEE 802,3, IEEE 802,5, FDDI...)
- The "at" (ARP Table) branch was replaced by an extension of the group of IP.
- The EGP (External Gateway protocol) branch is no longer used as the EGP protocol nowadays does not have any importance.
- Many further MIB modules have been registered under the " mib-2” node. The assignment of the registration numbers is delegated to the Internet Assigned Numbers Authority (IANA).
- These days it would be good to introduce a certain modularity in the MIB so that different branches could be updated independently.

Further MIBs: Transmission [1/5]

MIB Description	RFC
IEEE 802.3 Repeater Devices	2108
Data Link Switching	2024
IEEE 802.5	1748
ATM	1695
SMDS	1694
Ethernet	1650
Frame Relay	1604
SONET / SDH	1595
FDDI	1512
Link Control Protocol of PPP	1471
Multiprotocol Interconnect over X.25	1461
DS3 / E3	1407
DS1 / E1	1406
Frame Relay DTEs	1315

Further MIBs: Network Layer [2/5]

MIB Description	RFC
IP Forwarding Table	2096
RMON	1757
RMON Version 2	2021
IP Mobility Support	2006
OSPF Version 2	1850
RIP	1724
BGP Version 4	1657
Token Ring extensions to RMON	1513
Identification MIB	1414
BGP Version 3	1269

Further MIBs: Application Layer [3/5]

MIB Description	RFC
WWW servers	2039
RDBMS	1697
DNS Resolver	1612
DNS Server	1611
X.500 Directory	1657
Mail	1566
Network Services	1565
Host Resources	1514

Further MIBs: Hardware Specific [4/5]

MIB Description	RFC
Entity	2037
Printer	1759
Modem	1696
Parallel printer-like Hardware	1660
RS-232-like Hardware	1659
Character Stream Devices	1658
UPS	1628

Further MIBs: Vendor Specific [5/5]

MIB Description	RFC
APPC	2051
TCP/IPX Connection	1792
SNA Data Link Control (SDLC)	1747
AppleTalk	1742
SNA NAUs	1666
DECNET Phase IV	1559
SNMP over IPX	1420
SNMP over AppleTalk	1419

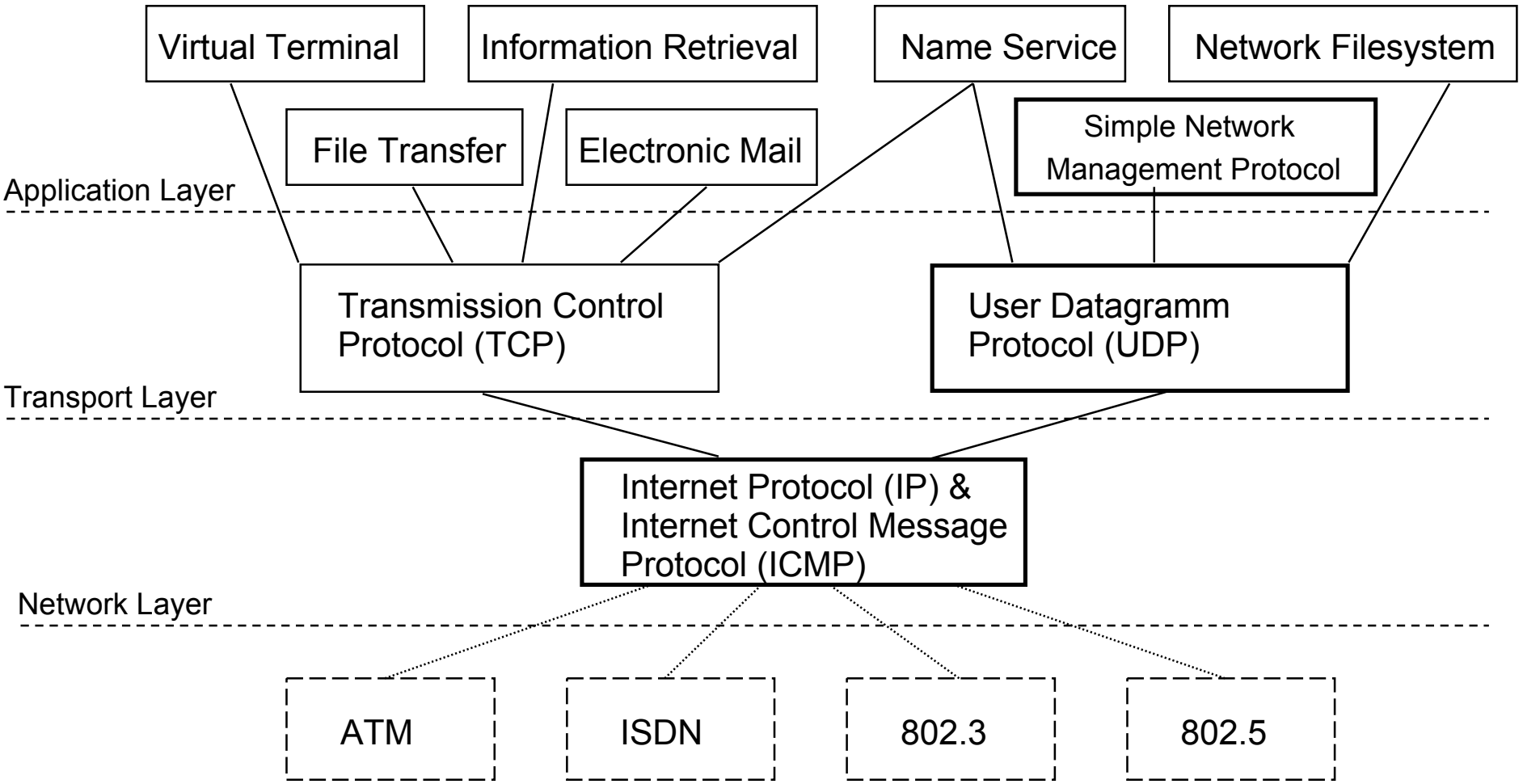
Relations Between MIBs [1/2]

	MIB-II	Host	Repeater	Bridge	RMON
Interface Statistics	X				
IP, TCP & UDP Statistics	X				
SNMP Statistics	X				
Host Job Counts		X			
Host File System Information		X			
Link Testing			X	X	
Network Traffic Statistics			X	X	X
Address Tables			X		X
Host Statistics			X		X

Relations Between MIBs [2/2]

	MIB-II	Host	Repeater	Bridge	RMON
Historical Statistics					X
Spanning Tree Performance				X	
Wide Area Link Performance				X	
Thresholds for any variable					X
Configurable Statistics				X	
Traffic Matrix with all Nodes					X
‘Host Top N’ Information					X
Packet/Protocol Analysis					X
Distributed Logging					X

2.4 Simple Network Management Protocol Version 1



Lexicographical Ordering

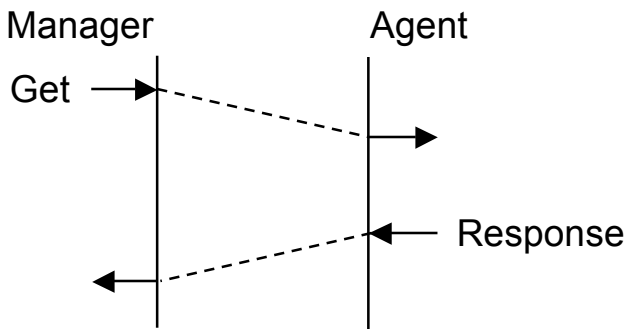
- MIB instances are arranged in the MIB according to their lexicographical ordering.
- The ordering is determined by the value of the object identifier that identify the instance.
- The SNMP log uses the lexicographical order, in order to read (walk) conceptual tables or unknown MIBs.

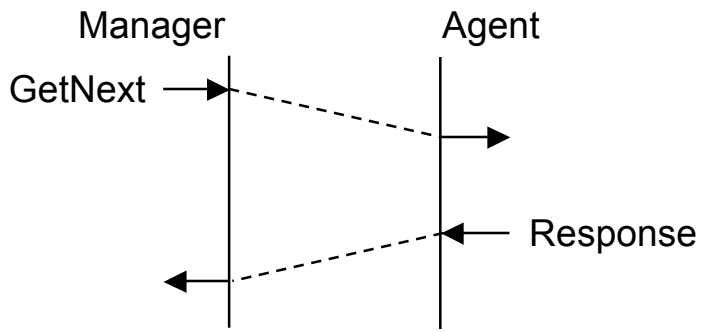
Example of Lexicographical Ordering

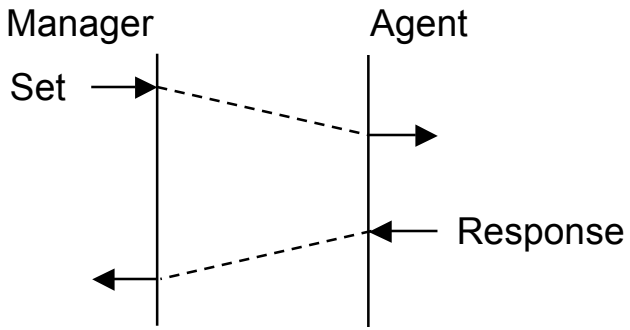
Object Identifier:	Value:	Object Identifier	Value :
1.1.0	10.1.2.3	1.3.1.2.4	7
1.2.1.0	"FilterFresh"	1.3.1.2.5	8
1.2.2.0	54321	1.3.1.2.6	9
1.3.1.1.1	1	1.3.1.3.1	2
1.3.1.1.2	2	1.3.1.3.2	3
1.3.1.1.3	3	1.3.1.3.3	2
1.3.1.1.4	4	1.3.1.3.4	2
1.3.1.1.5	5	1.3.1.3.5	3
1.3.1.1.6	6	1.3.1.3.6	3
1.3.1.2.1	2		
1.3.1.2.2	3		
1.3.1.2.3	5		

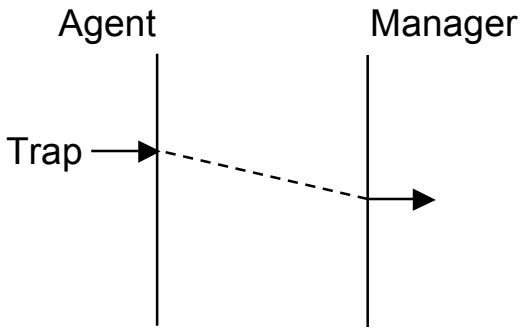
- With this ordering the conceptual table structure is lost as the walk output is a list and no longer a table.
- the SNMP protocol operates only on this arranged list.

SNMPv1 protocol operations (RFC 1157)









Note: the SNMP protocol can only exchange (a list of) scalars.

SNMPv1 Message Format

SNMP message:

version	community	SNMP PDU

GetRequest, GetNextRequest, SetRequest:

PDU type	request-id	0	0	variable-bindings

GetResponse:

PDU type	request-id	error-status	error-index	variable-bindings

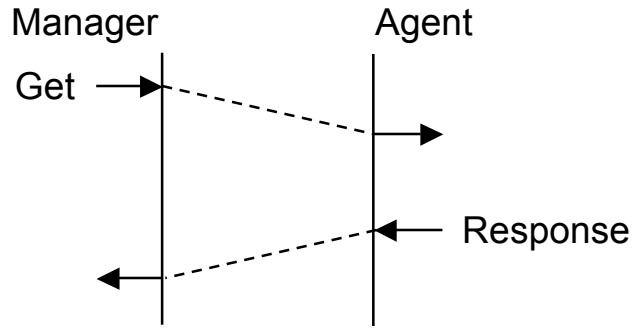
Trap:

PDU type	enterprise	address	generic	specific	timestamp	vbs

variable-bindings:

$name_1$	$value_1$	$name_2$	$value_2$	...	$name_n$	$value_n$

SNMPv1 Get Operation



- The Get operation can be used for reading one or more variables.

- Possible errors when processing a GET operation:
  - noSuchName — the requested instance does not exist or is not a leaf.
  - tooBig — the result of the request does not fit not into the response (UDP).
  - genErr — any other error occurred.

- In the case of several errors occurred, only one error is signalled as error-index and error-status are unique in the PDU.

Example of Get Operation

- Get(1.1.0)
  Response(noError@0, 1.1.0=10.1.2.3)

- Get(1.2.0)
  Response(noSuchName@1, 1.2.0)

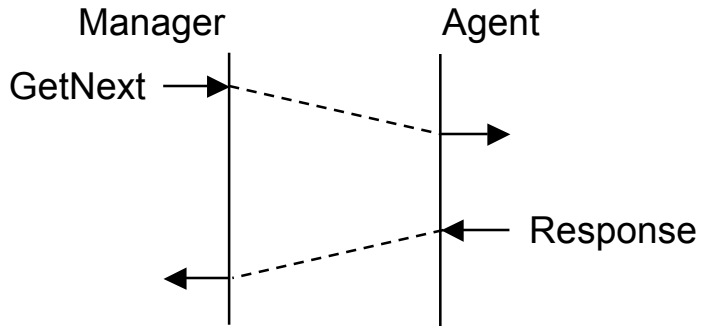
- Get(1.1)
  Response(noSuchName@1, 1.1)

- Get(1.1.0, 1.2.2.0)
  Response(noError@0, 1.1.0=10.1.2.3, 1.2.2.0=54321)

- 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)

- Get(1.1.0, 1.2.2.0, 1.1)
  Response(noSuchName@3, 1.1.0, 1.2.2.0, 1.1)

SNMPv1 GetNext Operation



- It retrieves the object name and the value of the next instance. This operation is used to discover MIB structures and read tables.
- The GetNext operation allows MIB instances to be read in accordance to the lexicographical order.
- Using multiple/successive GetNext operations it is possible to read the complete MIB without knowing its structure.
- Possible errors when processing a GetNext Operation:
  - noSuchName — the requested instance does not exist (= end of MIB).
  - tooBig — the result of the request does not fit not into the response (UDP).
  - genErr — any other error occurred.

Example of GetNext Operation

- GetNext(1.1.0)
  Response(noError@0, 1.2.1.0=FilterFresh)

- GetNext(1.2.1.0)
  Response(noError@0, 1.2.2.0=54321)

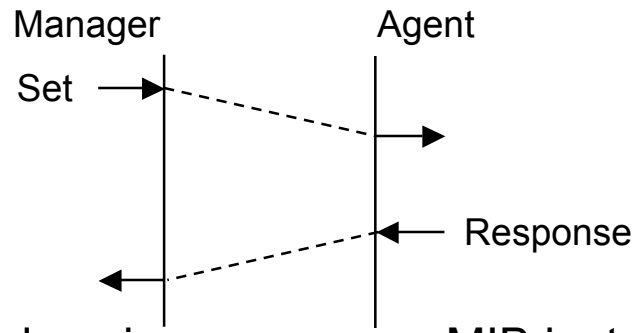
- GetNext(1.1)
  Response(noError@0, 1.1.0=10.1.2.3)

- GetNext(1.3.1.1.1)
  Response(noError@0, 1.3.1.1.2=2)

- GetNext(1.3.1.1.6)
  Response(noError@0, 1.3.1.2.1=2)

- 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)

SNMPv1 Set Operation



- The Set Operation writes values in one or more MIB instances.
- The Set Operation is atomic.
- With the help of the set operation new MIB instances can also be created, if the MIB definition permits (there is no standard procedure defined in SNMPv1 for instance creation).
- Possible errors when processing a Set operation:
  - noSuchName — the requested instance does not exist and cannot be created.
  - badValue — the specified value is of wrong type.
  - tooBig — the result of the request does not fit not into the response (UDP).
  - genErr — any other error occurred.
- The error code readOnly is also defined, but not usually used!

Example of Set Operation

- ```
  Set(1.2.1.0=HotJava)
  Response(noError@0, 1.2.1.0=HotJava)
  ```

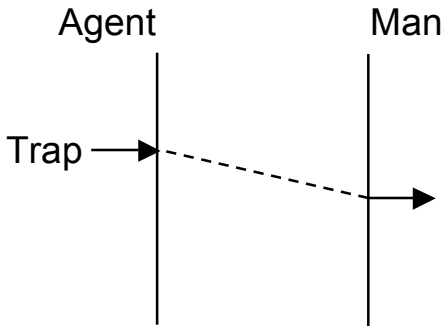
- ```
  Set(1.1.0=foo.bar.com)
  Response(badValue@1, 1.1.0=foo.bar.com)
  ```

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

- ```
  Set(1.2.1.0=HotJava, 1.1.0=foo.bar.com)
  Response(badValue@2, 1.2.1.0=HotJava, 1.1.0=foo.bar.com)
  ```

- ```
  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)
  ```
  ```

SNMPv1 Trap Operation



NOTE:
- The only operation Agt ->Mgr
- Unsolicited operation

- With the trap operation and agent can emit an event and inform a manager. Note: a manager can be configured to discard traps!
- The receipt of a trap operation is not acknowledged thus is unreliable as it can be lost during the transfer.
- The production of traps can lead to so-called trap storms, if e.g. after a power failure all devices want to display the restart at the same time.
- Agents can be normally configured with the IP addresses of hosts where traps can be dispatched. However there is no standard technique in SNMPv1 for such agent configuration. Usually a configuration file (not the MIB) is used.
- Although if traps are used, polling is still necessary (for instance the agent might be down)

Example of SNMPv1 Trap Operation

- ColdStart
  Trap(generic=0, specific=0)

- WarmStart
  Trap(generic=1, specific=0)

- LinkDown
  Trap(generic=2, specific=0, 1.3.6.1.2.1.2.2.1.1.2=2)

- LinkUp
  Trap(generic=3, specific=0, 1.3.6.1.2.1.2.2.1.1.2=2)

- AuthenticationFailure
  Trap(generic=4, specific=0)

- EnterpriseSpecific (QMS, qmsPtrErrorMsg)
  Trap(generic=6, specific=1, enterprise=1.3.6.1.4.1.480,
  1.3.6.1.4.1.480.2.1.1.1=out of paper)

2.5 Simple Network Management Protocol Version 2c

- There are some variants of of SNMP Version 2:
  - SNMPv2p

    SNMPv2 version with party-based security model, historical
  - SNMPv2c

    SNMPv2 with trivial community-based security model, experimental
  - SNMPv2u

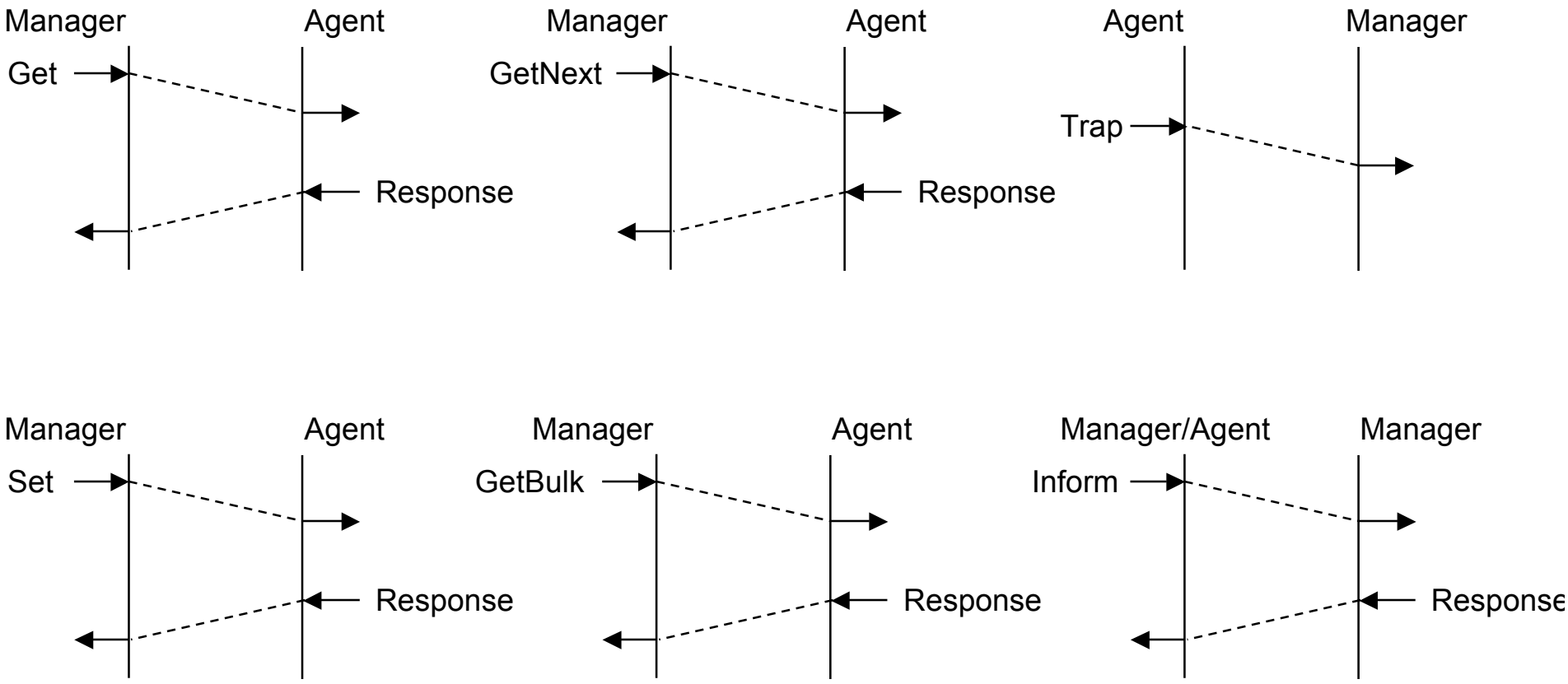
    SNMPv2 with a user-based security model, historical
  - SNMPv2*

    SNMPv2 with security and administration model, historical

- The term SNMPv2 is ambiguous. SNMPv2c found some spreading, although IETF has never standardised it.

- Work on a solution of the security problems has blocked improvements of other protocol characteristics (too) for a long time.

SNMPv2c protocol operations (RFC 1905)



SNMPv2c Message Format

SNMP message:

version	community	SNMP PDU

GetRequest, GetNextRequest, SetRequest, Trap, InformRequest:

PDU type	request-id	0	0	variable-bindings

GetResponse:

PDU type	request-id	error-status	error-index	variable-bindings

GetBulkRequest:

PDU type	request-id	non-reps	max-reps	variable-bindings

variable-bindings:

$name_1$	$value_1$	$name_2$	$value_2$	...	$name_n$	$value_n$

SNMPv2 Exceptions (RFC 1905)

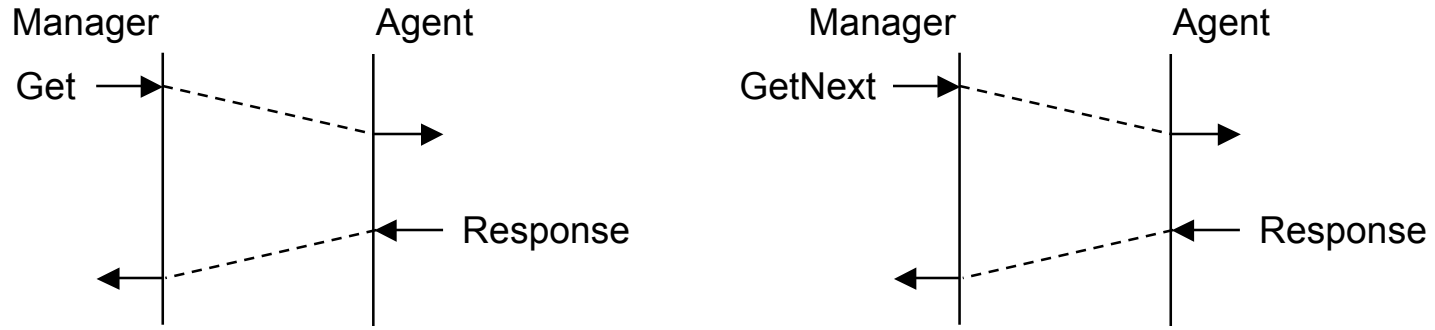
SNMPv2 Exception	SNMPv1 Status	Used by
noSuchObject	noSuchName	Get
noSuchInstance	noSuchName	Get
endOfMibView	noSuchName	GetNext, GetBulk

- Exceptions allow instance access errors to be signaled to MIB authorities, without causing the whole operation to fail (as happened in SNMPv1).

- Example:

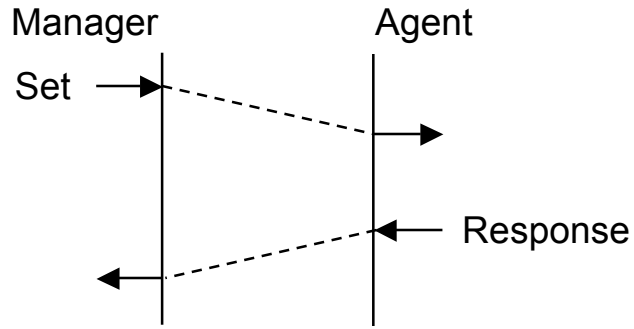
```
Get(1.1.0, 1.1.1, 1.2.0)
Response(noError@0, 1.1.0=10.1.2.3, 1.1.1=noSuchInstance,
        1.2.0=noSuchObject)
GetNext(1.1, 1.5.42)
Response(noError@0, 1.1.0=10.1.2.3, 1.5.42=endOfMibView)
```

SNMPv2c Get and GetNext Operations



- Not existing MIB instances produce an exception and not an error.
- Similar to the equivalent SNMPv1 operations.

SNMPv2c Set Operation



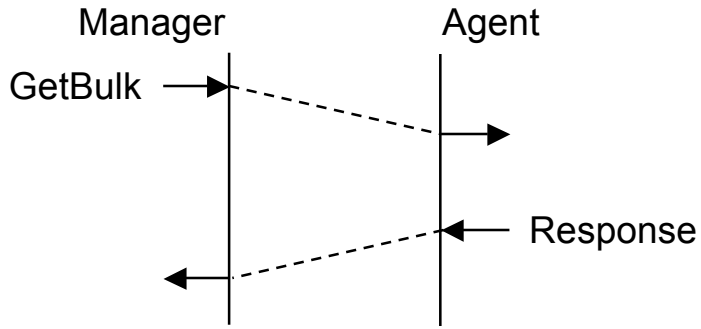
- There are 14 possible error codes during processing of set operations:

`wrongValue`	`wrongEncoding`	`wrongType`
`wrongLength`	`inconsisentValue`	`noAccess`
`notWritable`	`noCreation`	`inconsisentName`
`resourceUnavailable`	`commitFailed`	`undoFailed`

- There are two more error codes that have been defined but not really used: `readOnly`, `authorizationError`

- No support of error codes that depend on the object type.

SNMPv2c GetBulk Operation



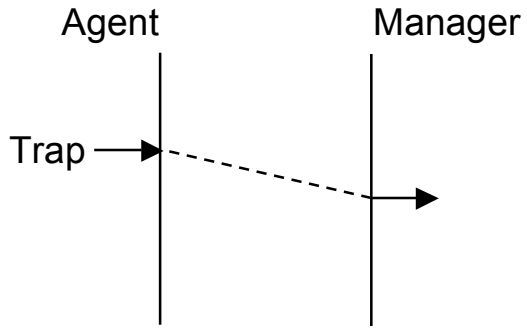
- An extension of the GetNext operation:
  - It returns the first N elements (non repetition) of the varbind list treated as normal GetNext operations.
  - The following items of the varbind list treated as repeated Get Next operation, whereby M (max repetition) indicates the max number of repetitions.

- The GetBulk operation is similar to the GetNext operation on the lexicographical arranged list of the MIB instances and has therefore no knowledge of table boundaries.

Example of the GetBulk Operation

- ```
  GetBulk(non-repeaters=0, max-repetitions=4, 1.1)
  Response(noError@0, 1.1.0=10.1.2.3, 1.2.1.0=FilterFresh,
           1.2.2.0=54321, 1.3.1.1.1=1)
  ```
- ```
  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)
  ```
- Without knowledge about the length of a table it is difficult for the manager to select a suitable number for max repetitions:
  - if max-repetitions is too small, then there is no efficiency increase of GetBulk with respect to the GetNext operation .
  - if max-repetitions is too large, then a large number of unnecessary instances are read .
- The agent can possibly produce a response, which can either get lost in large/busy networks or not be processed at all by the manager (this causes the manager to retransmit the request).
- If max repetitions is large and reading the MIB instances is time-consuming, agents can receive multiple times the manager's request (e.g. due to retransmission) thus blocking the agent for some time.

SNMPv2c Trap Operation



- It corresponds logically to the SNMPv1 Trap operation.
- Trap specific information (sysUpTime, trapType) is accommodated in the varbind list.
- Trap types are indicated by Object Identifier and not by a pair of numbers (generic, specific) as in SNMPv1.

SNMPv1 vs. SNMPv2c Traps

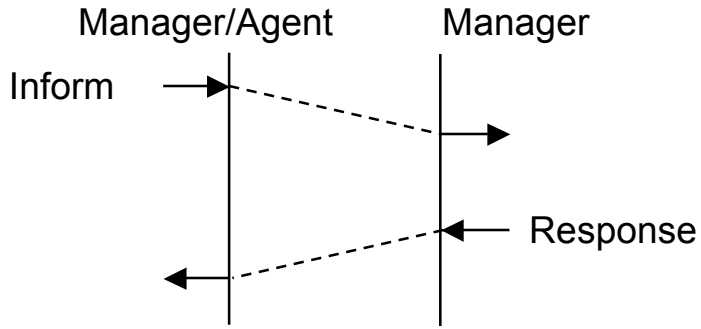
- In SNMPv2 MIBs may now include `NOTIFICATION-TYPE` macros.
- SNMPv1 Trap

```
myLinkDown TRAP-TYPE
           ENTERPRISE  myEnterprise
           VARIABLES   { ifIndex }
           DESCRIPTION
           "A myLinkDown trap signifies that the sending SNMP application
            entity recognises a failure in one of the communications links
            represented in the agent's configuration."
           ::= 2
```

- SNMPv2 Trap

```
linkUp NOTIFICATION-TYPE
OBJECTS { ifIndex }
STATUS current
DESCRIPTION
"A linkUp trap means that the entity has detected that the ifOperStatus
   object has changed to Up"
::= { snmpTraps 4 }
```

SNMPv2c Inform Operation



- The structure of the PDU corresponds to a SNMPv2 Trap PDU.
- It allows (new) managers to talk each other (SNMPv1 limited interaction to agent-manager or vice-versa).
- The receipt of a Inform message is acknowledged with a Response message.
- Example:
- ```
  Inform(1.2.2.0=54321, 1.4.1.0=1.4.2.43,
         1.3.1.2.2=16, 1.3.1.3.2=3)
  Response(noError@0, 1.2.2.0=54321, 1.4.1.0=1.4.2.43,
           1.3.1.2.2=16, 1.3.1.3.2=3)
  ```
  ```

SNMPv2c and SNMPv1 Error Codes

SNMPv2	SNMPv1	Comment
noError	noError	all operations
tooBig	tooBig	Get, GetNext, Set, Inform
noSuchName	noSuchName	Get, GetNext, Set (only with SNMPv1)
badValue	badValue	Set (only with SNMPv1)
readOnly	readOnly	not used
genErr	genErr	Get, GetNext, GetBulk, Set
wrongValue	badValue	Set (only with SNMPv2c)
wrongEncoding	badValue	Set (only with SNMPv2c)
wrongType	badValue	Set (only with SNMPv2c)
wrongLength	badValue	Set (only with SNMPv2c)
inconsisentValue	badValue	Set (only with SNMPv2c)
noAccess	noSuchName	Set (only with SNMPv2c)
notWritable	noSuchName	Set (only with SNMPv2c)
noCreation	noSuchName	Set (only with SNMPv2c)
inconsisentName	noSuchName	Set (only with SNMPv2c)
resourceUnavailable	genErr	Set (only with SNMPv2c)
commitFailed	genErr	Set (only with SNMPv2c)
undoFailed	genErr	Set (only with SNMPv2c)
authorizationError	noSuchName	not used

SNMP v2 vs SNMP v1

- Improved Performance via the Get-Bulk PDU.
- Definition of additional datatypes and formalisms based on implementation experience of SNMPv1 agents/managers.
- Transport Service Independence: mappings for SNMPv2 have been defined for several transports and not for just UDP (TCP is on the way!).
- Redefined the Trap PDU:
  - ↪ It has the same format of the other PDUs
  - ↪ It may be sent to zero, one or many managers

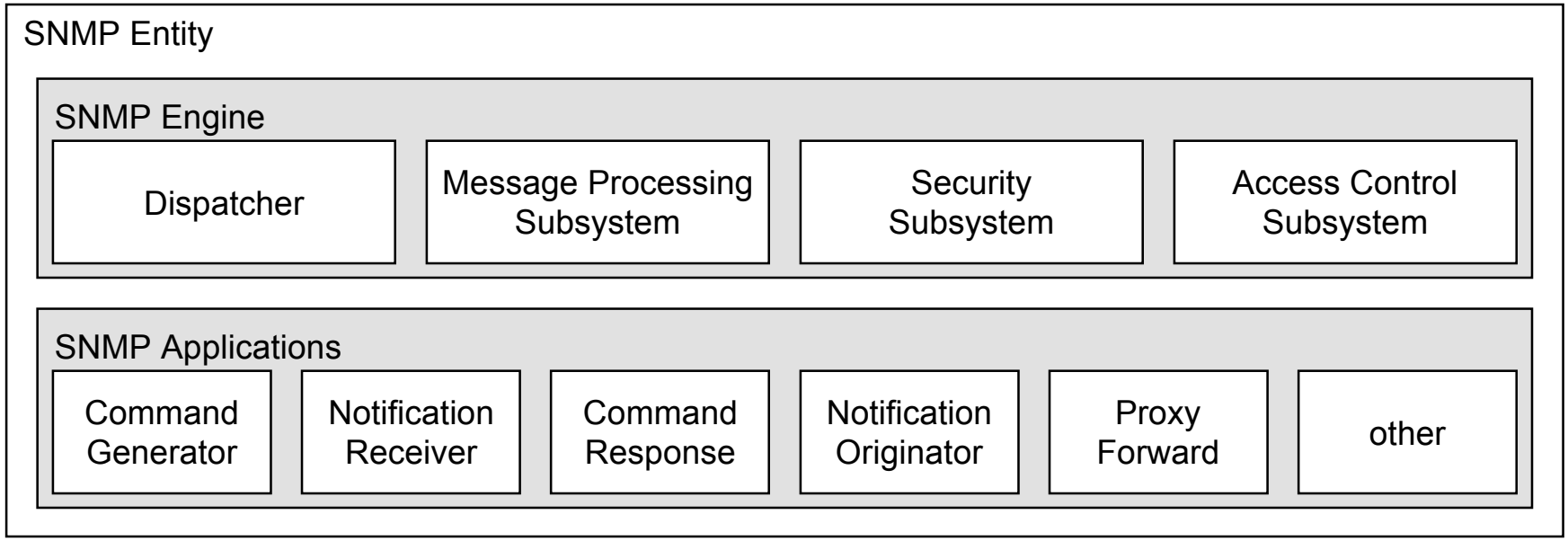
2.6 SNMP V3

- Design goals of SNMPv3:
  - Issue of secure SET protocol operations.
  - Definition (hopefully) of a long-living architecture model
  - Support of cheap simple and more expensive complex implementations (scalability).
  - Independence of the standards
  - Use of existing material (mostly MIBs) when possible (design reuse)
  - SNMP is to remain as simply as possible

- Several (commercial and open source) implementation available.

- Spreading in real networks still relatively small (most network devices still use SNMPv1).

Architectural Model of SNMPv3 (RFC 2571)



- The SNMP engine of a SNMP entity consists of several subsystems and a dispatcher.
- The manager/agent model is replaced by a number of smaller “applications ”.
- The modularity permits incremental advancement of SNMP by means of SNMP Context (RFC 2571)

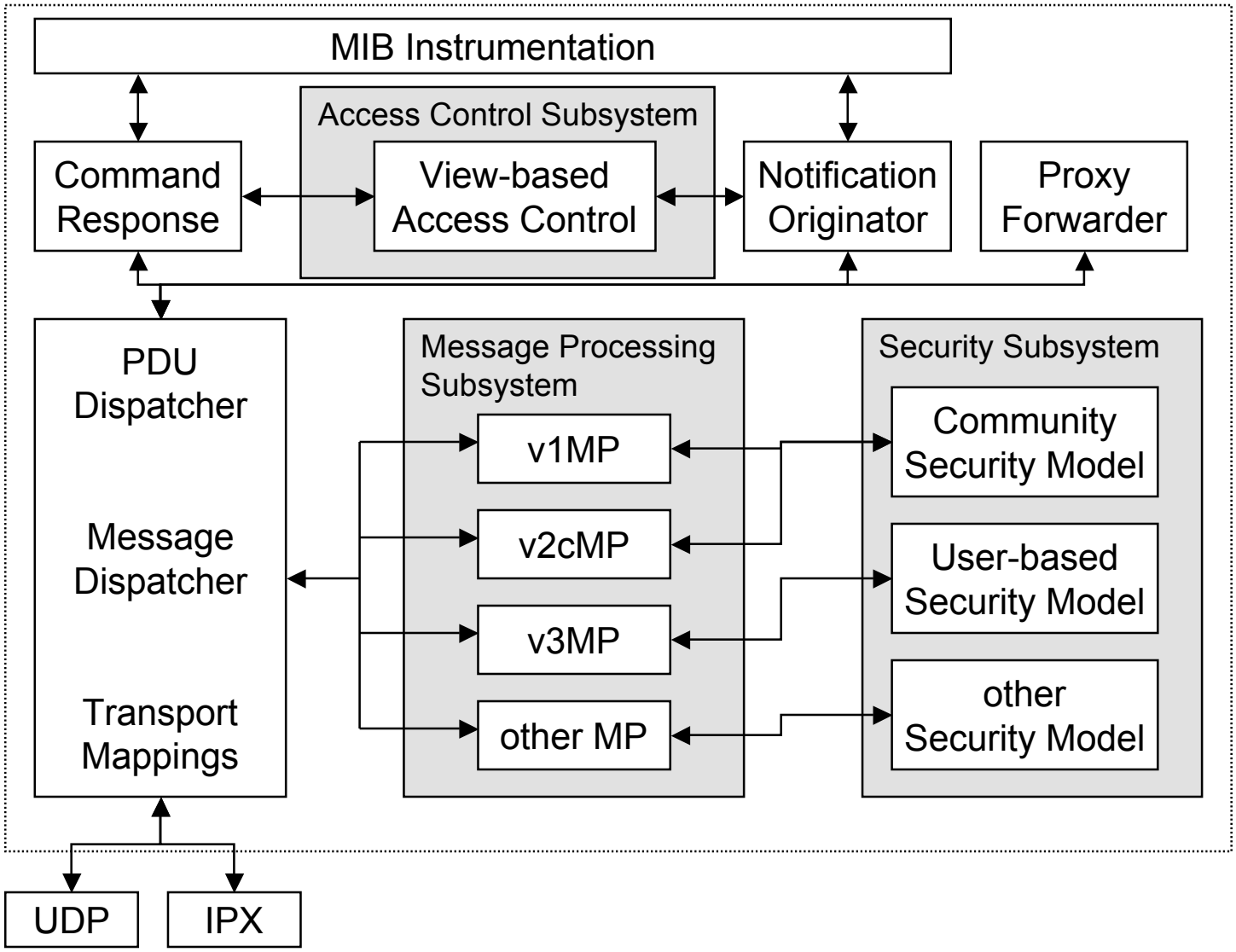
SNMP Context (RFC 2571)

- A context is a quantity of management information that a SNMP Entity can have accessed to. For each subsystem:
  - ↪ a SNMP-Entity has potentially access to several contexts.
  - ↪ The same information can be present in several contexts.

- In a management domain an instance of a Managed Objects is uniquely identified by the following items:
  - the identification of the SNMP engines in a SNMP Entity (e.g. „xzy“).
  - the name of the context in a SNMP Entity (e.g. „board1“).
  - the identification of the type of the Managed Objects (e.g. „IF-MIB::ifDescr).
  - the identification of the Instance (e.g. „1“).

- ➢ Note: the identification of an SNMP engine does not have to do anything with their addressing.

SNMPv3 Agent in SNMPv3: Architectural Model



SNMPv3 Message Format (RFC 2572)

SNMPv3Message:

msgVersion	msgGlobalData	msgSecurityParameter	msgData (scopedPDU)

MsgGlobalData:

msgID	msgMaxSize	msgFlags	msgSecurityModel

UsmSecurityParameter:

msgEngineID	msgEngineBoots	msgEngineTime	msgUserName	msgAuthParams	msgPrivParams

ScopedPDU:

contextEngineID	contextName	SNMPv2 PDU (as defined in RFC 1905)

- Security information are in the centre of the message.
- msgData contain either a ScopedPDU or an encoded ScopedPDU.
- msgID is used for the association of responses to pending inquiries.
- msgSecurityParameter depends on msgSecurityModel.

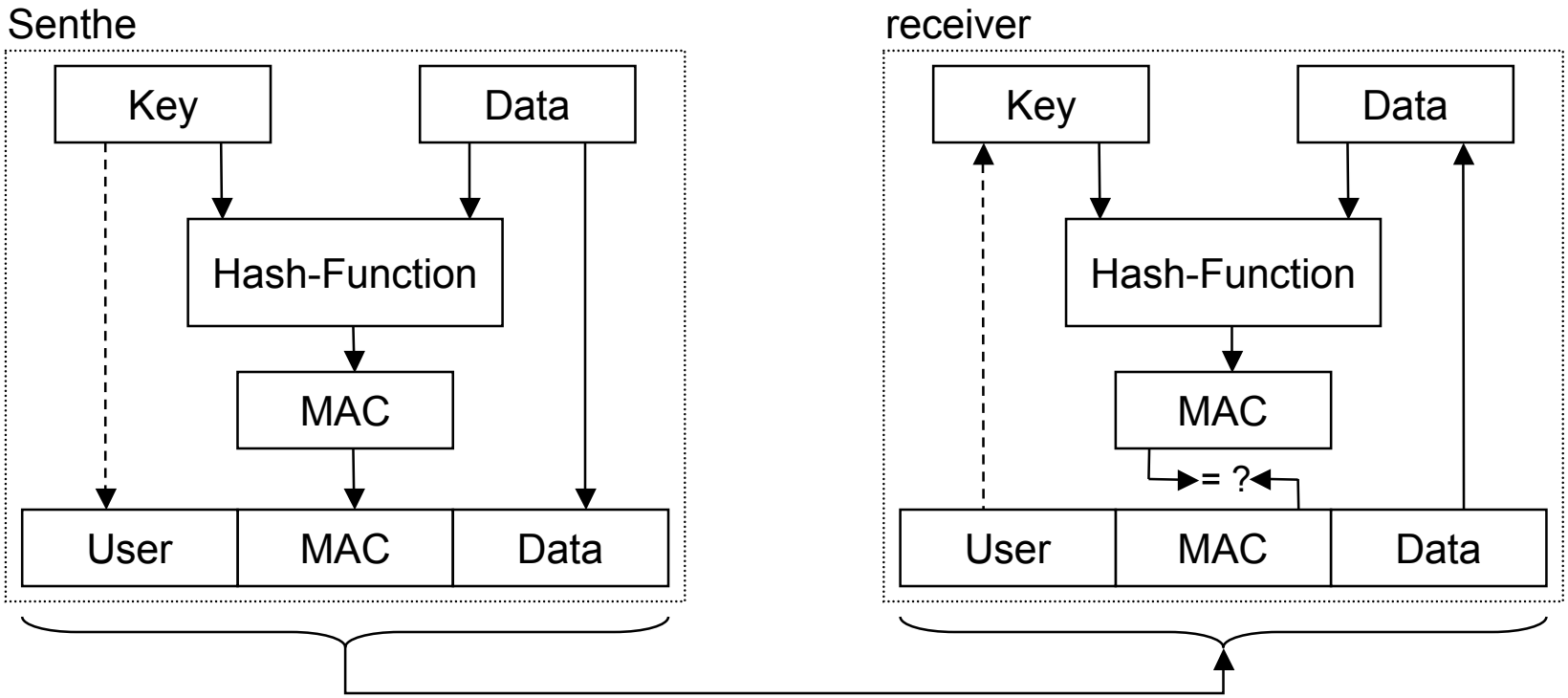
Security Issues

- Blow you can find the questions which must be answered when a decision whether an operation has to be performed:
  - Is the received message authentic?
  - Who (requestor name) would like to get the operation executed?
  - Which objects are affected by the operation?
  - Which access rights has the requestor regarding the objects concerned?
- Questions 1 and 2 are answered by the measures to the protection of the messages (authentication, encoding).
- Questions 3 and 4 are answered by a model to the access supervision (Unix-like).

Authentication in SNMPv3 (RFC 2574)

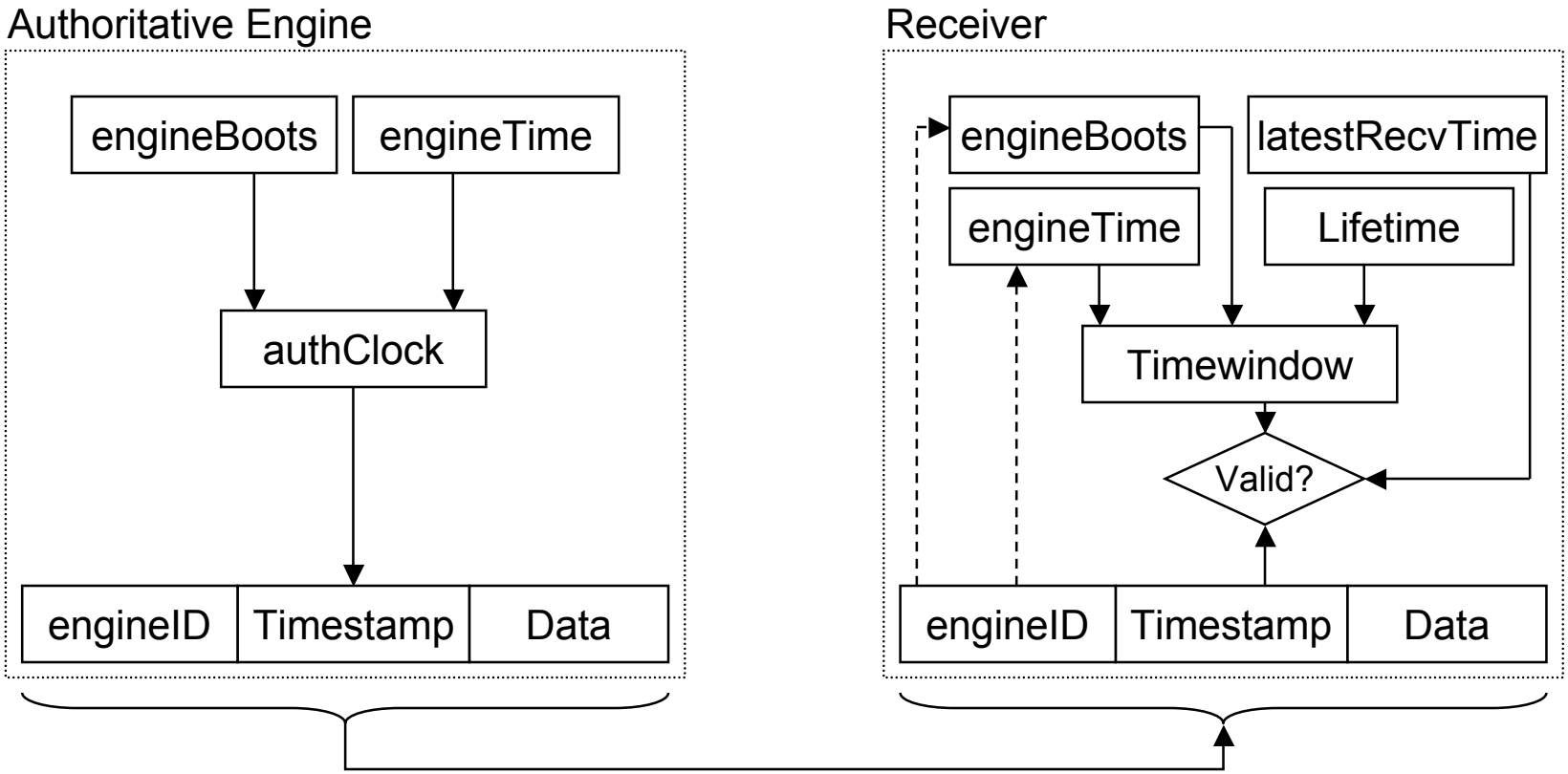
- Cryptographic strong hash functions (MD5, SHA-1) for the calculation of cryptographically strong checksums (Message Authentication Code or MAC).
- Algorithmic production of key K_U from a password is done by applying a hash function to the 1 MB long password concatenation.
- Derivative "more localised" key K_{UL} is derived from key K_U for the SNMP Engine E using MD5: $K_{UL} = MD5(K_U, E, K_U)$
- HMAC (keyed hashing) of a message M using K_{UL} and MD5:
  - Extension of K_{UL} to 64 bytes by adding 0x00 bytes.
  - Set IPAD to a 64 bytes vector of value 0x36.
  - Set OPAD to a 64 bytes vector of value 0x56.
  - Calculate $K_I := K_{ULX}$ XOR IPAD and $K_O := K_{ULX}$ XOR OPAD.
  - Calculate HMAC := $MD5(K_O, MD5(K_I, M))$.
  - The first 96 bits of HMAC result in the MAC of the message M.
- SNMPv3 includes protection against repetitions of old messages from loosely synchronised clocks and time stamp data integrity.

Data Integrity and Authentication



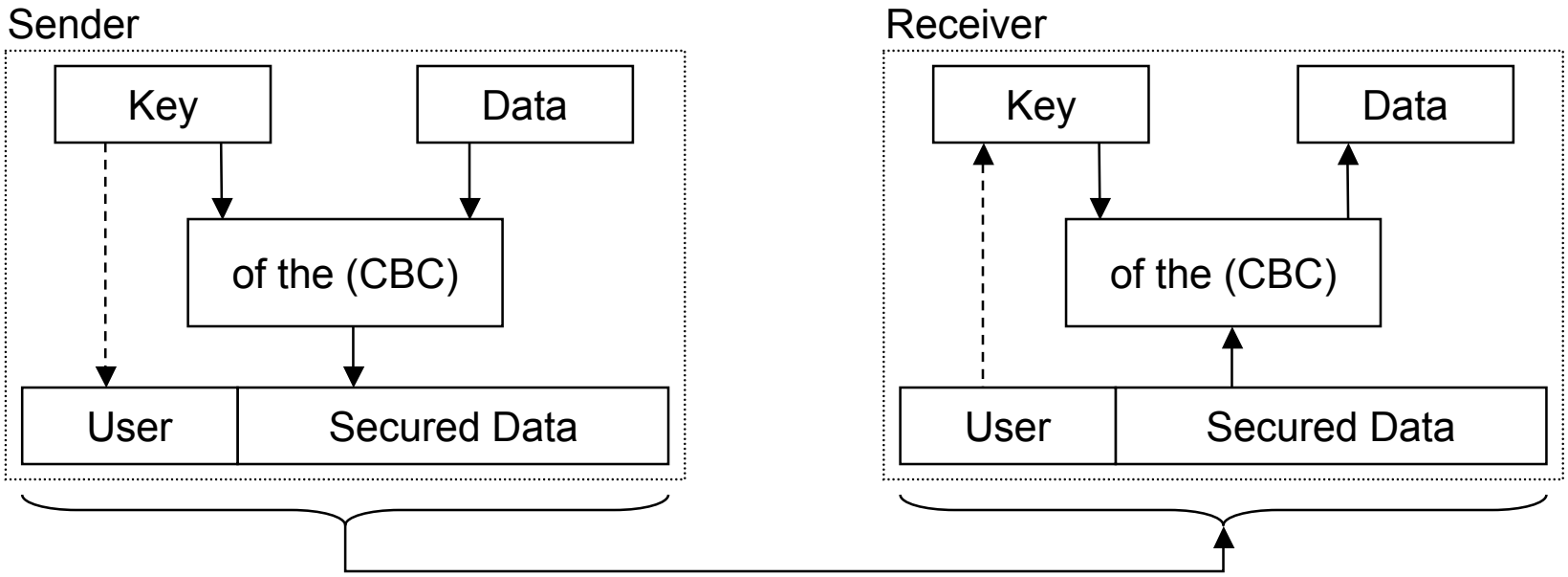
- Authentication with Message Authentication Code (MAC) is efficient to implement.
- The Hash function must be cryptographically strong and a "good" MAC producer.
- The MD5 algorithm (RFC 1321) can be implemented in software with acceptable performance (128 bit digest).
- The Secure Hash algorithm 1 (SHA-1) is considered at present stronger of MD5 .

Protection Against Repetitions of Old Messages



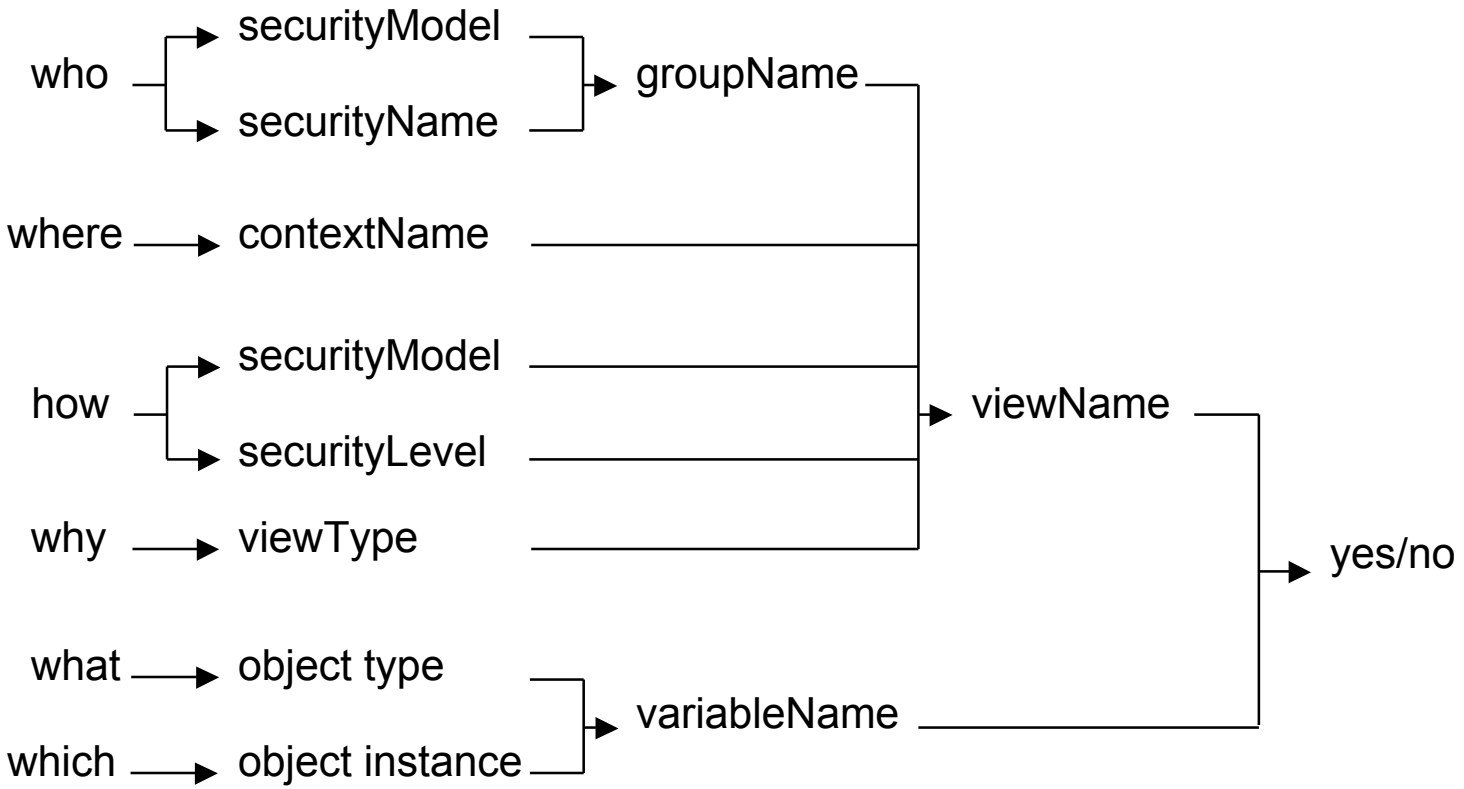
- A recipient must know the "time-of-day" of the authoritative SNMP engine for the message.
- If the received message is situated in the validity interval and is ”younger" than the last valid message, then the message will become processed and the clocks adapted.
- Before the beginning of authenticated communication the clocks must be synchronised.

Protection Against Sniffing



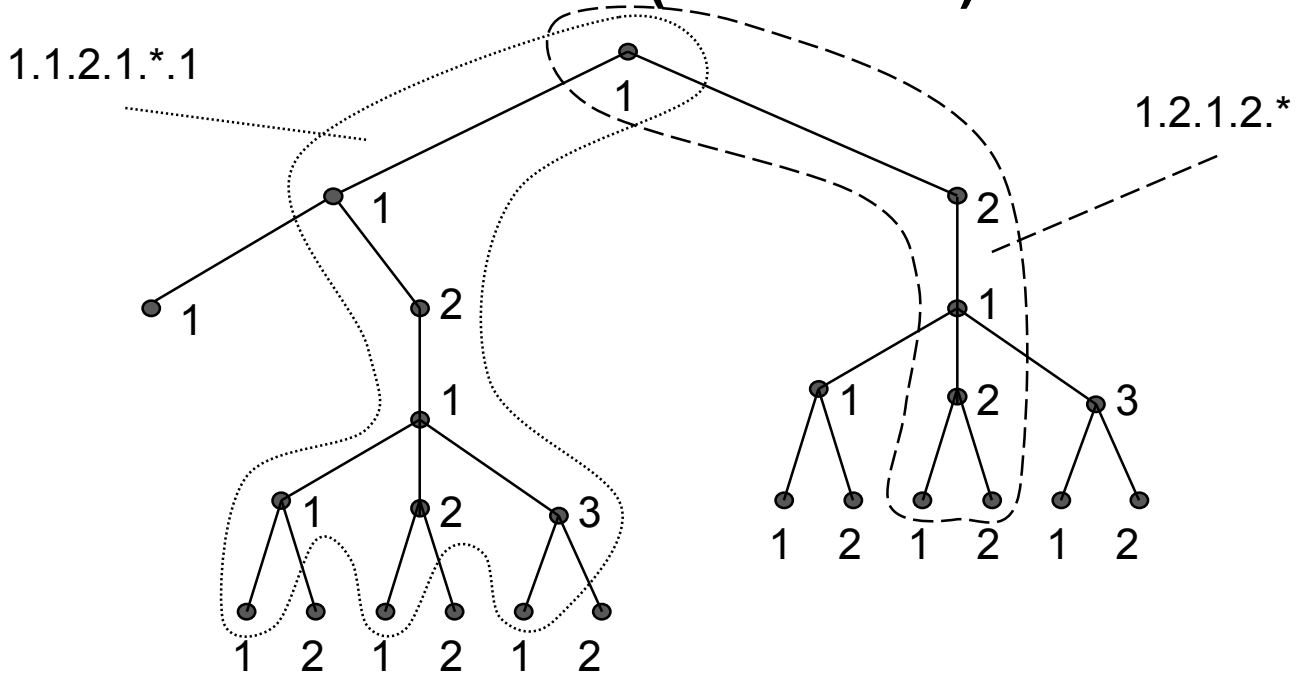
- For protecting against sniffing the ScopedPDU can be optionally encoded.
- Data Encryption Standard (of the) in Cipher Block Chaining Modus (CBC) is used for encryption.
- Encryption is relatively complex and should only be used in area/situations where an encoding is really necessarily.
- SNMPv3 permits "relatively protected" code modification without encryption (by using message digest).

Access Control in SNMPv3 (RFC 2575)



- The securityLevel determines whether a message without authentication (noAuth/noPriv), with authentication (auth/noPriv) or with authentication and encoding (auth/priv) is dispatched,
- The securityName is a name independent of the securityModel.

MIB Views (RFC 2575)

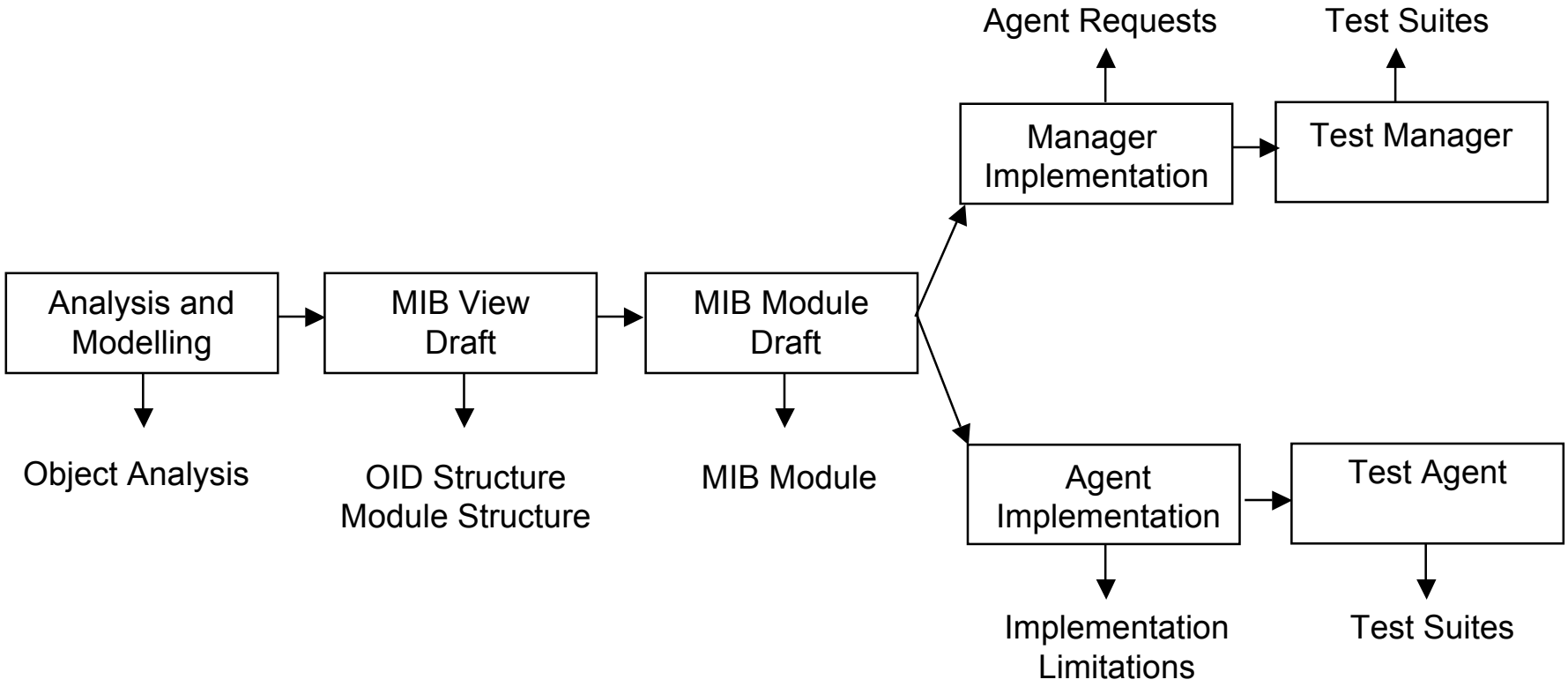


- A view subtree is the quantity of all MIB objects, which possess common OID prefix.
- A view tree family is the combination of one view subtree OID prefix with a filter (bitmask), which determines whether an item of the prefix is significant or not.
- A view is an ordered set of view tree families.
- It defines the access rights for read view, write view and notify view.

Characteristics and Issues of Access Control

- The access decision is determined by both the instance name and type.
- By means of a suitable MIB structure and bitmask it is possible to reduce the number of necessary access control rules.
- The examination of access rules in an agent can be very expensive, since it is often not possible to know how many instances with a GetNext or GetBulk operation have to be skipped.
- The access control (RFC 2575) forbids the affiliation of a securityName in several groups. As consequence the configuration of pretty similar but slightly different access rules for different securityNames are complex to configure.

2.7 MIB Implementation of the Agent Extensibility Protocol



- It is possible for have several iterative phases for the MIB definitions until it is in draft status.
- MIB definitions cannot however be further changed, if they were released.

MIB Name Conventions

- Similar definitions should be registered together in the registration tree.
- Names of object types should begin the logical grouping with a common prefix, that suggest (e.g. sysDescr, sysUpTime).
- Names for counter are to be selected in the Plural form.
- Names of conceptual tables should possess the ending Table (e.g. ifTable).
- Names of lines of a conceptual table should possess the ending entry (e.g. ifEntry).
- All items of a conceptual table should use common prefix in the name (e.g. ifType, ifDescr).

Tables Indexing

- The selection of the indexing of a table (as for databases) has different consequences, which must be analysed and be weighed out against each other with several alternatives:

  - Access Control
    The selection and the sequence of index items can strongly influence the number of necessary rules for the definition of MIB Views hence the administration expenditure.

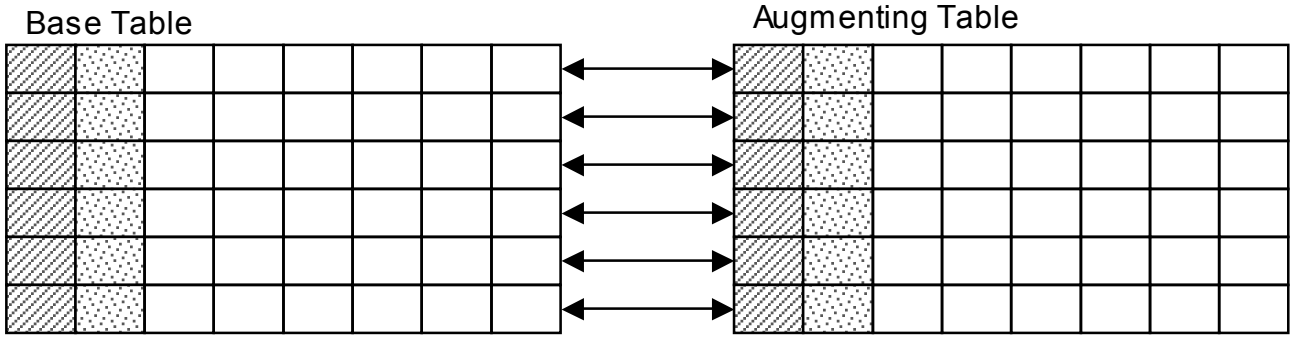
  - Efficient accesses to table sections
    Frequently needed access to subsets of a table can be supported by suitable selection of the index items.

  - Object Identifier Length
    Object Identifier can have a max. length of 128 components in SNMP. Long object identifiers produces high overhead.

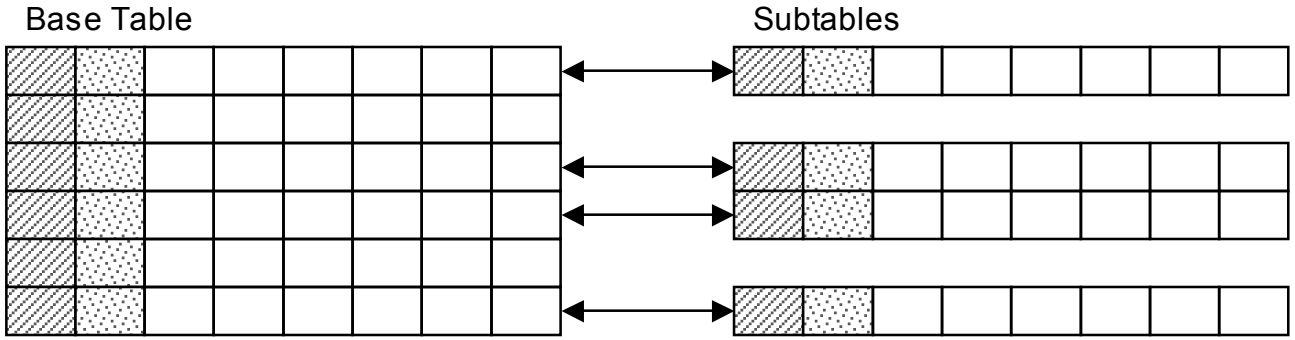
  - Lexicographical Ordering
  - The order influences the usage and efficiency of an implementation.

Relations between MIB Tables

- Tables can be extended exactly if there is a 1:1 relationship between all possible lines of the tables.

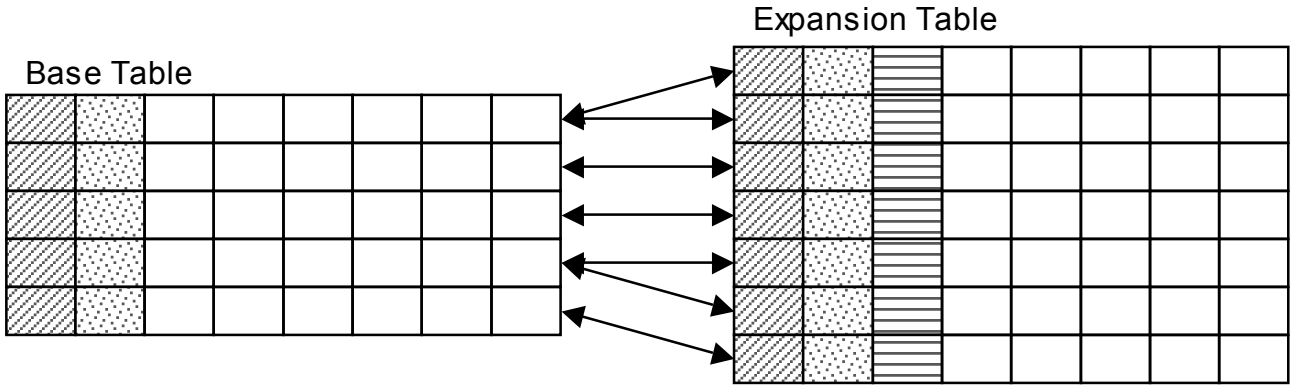


- Tables which possess only 1:1 relationship for a part of the possible lines, can use an extended version of the prior indexing.

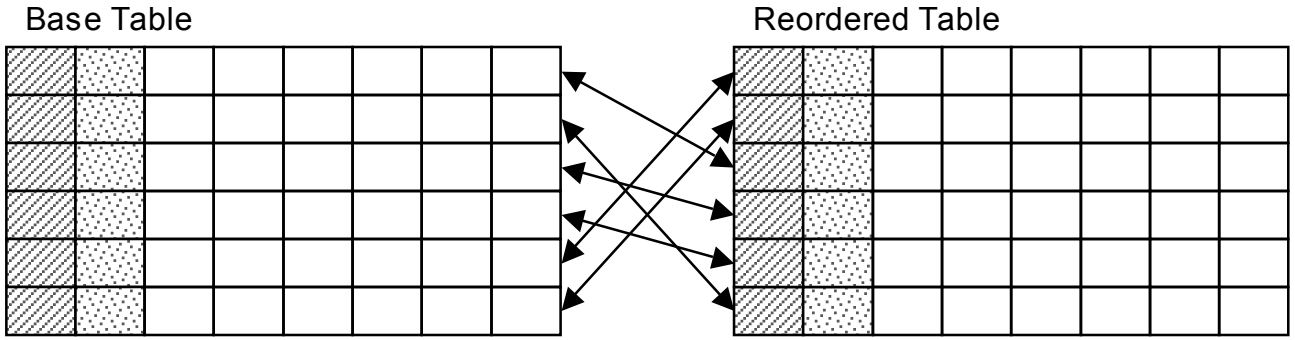


Relations between MIB Tables

- N:M relations with M > N can be expressed by extending index of the first table with further index variables.



- Tables with the same contents but with different ordering sequences can be represented by appropriately rearranging of the index items.



Datatypes

- Definition of Datatypes:
  - A Textual Conventions should be used, if a datatype is used several times or when other modules should be able to reuse the datatype.
  - Datatypes can be defined „inline“ only if they are used once.
  - In large MIB modules the definition of frequently used datatypes is recommended in their own module (reduce module cross references).
- Floating Point Numbers:
  - Selection of a suitable scaling to avoid rounding floating point numbers
  - Several objects can be used to represent mantissa, base and exponent.
  - IEEE floating point numbers are packed up in a OCTET STRINGER and represented as a character sequence in the ASCII alphabet.
- Handling Large Numbers:
  - To use several objects with different scaling.
  - To allocate e.g. 64-Bit Integer in two 32-Bit Integer.
  - Large numbers are packed up in a OCTET STRINGER and represented as a character sequence in the ASCII alphabet.

Data Structures

- Complex Data Structures (Tables in Tables):
  - It is not possible to use complex data structures! (good)
  - It is always possible to emulate complex data structures by extension tables (workaround).

- Linked Lists:
  - Definition of a table with an INTEGER value index
  - Definition of a ”next" column, similar to a next elements pointer.
  - Definition of a scalar that defines the beginning of the list.

- Binary Trees:
  - Definition of a table with an INTEGER value index.
  - Definition of one "left" and a "right" column similar to pointers.
  - Definition of a scalar, that stands for the root of the tree.

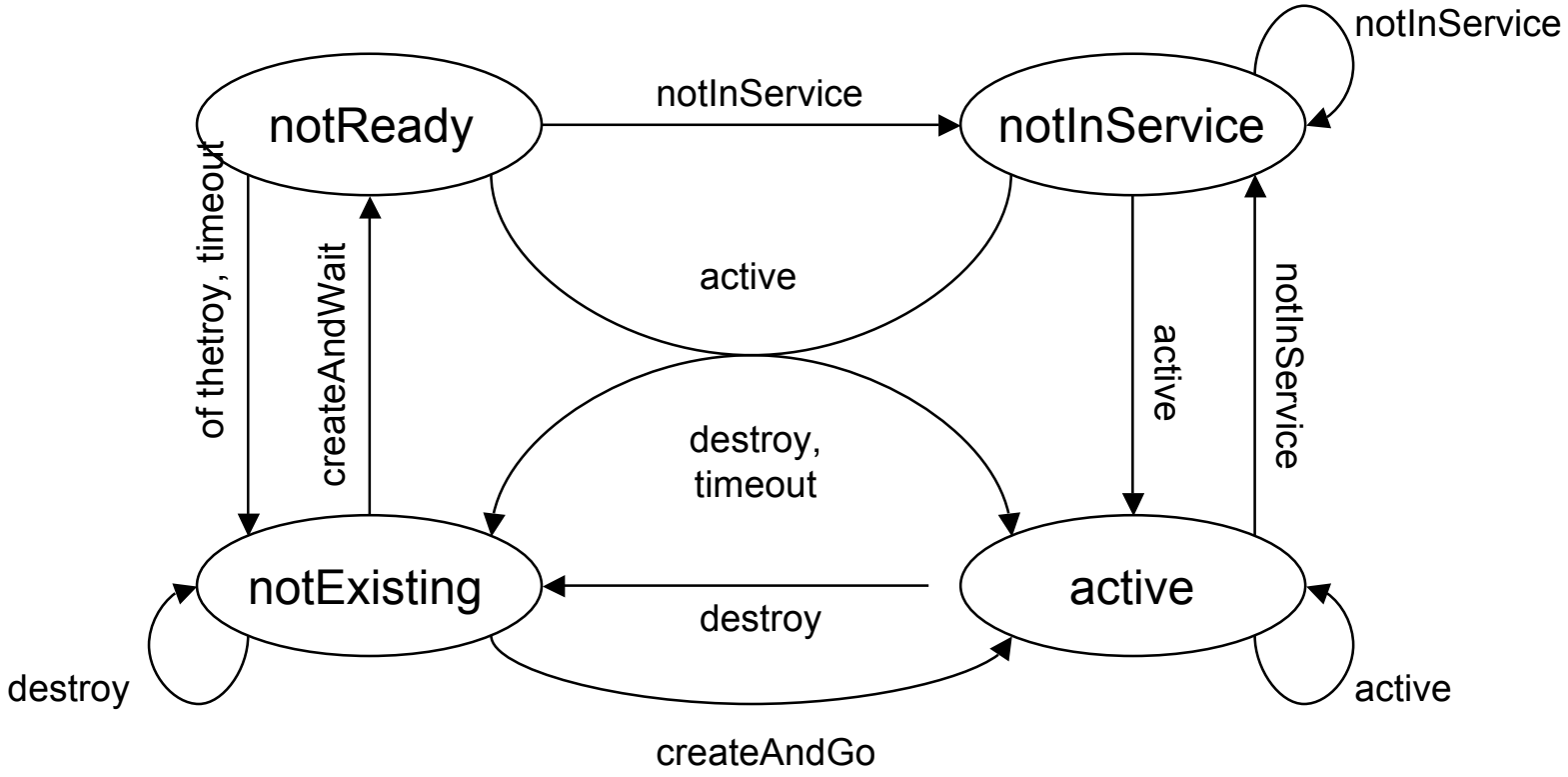
- In a similar way still more complex data structures can be implemented. It should be checked however very exactly whether the data structures cannot be transferred into simpler tabular forms.

Production of new Conceptual Rows

- The RowStatus Textual-Convention defines a status machine, that allows new rows to be created inside conceptual tables.
- This mechanism is needed when large rows (I.e. the row scalars wouldn't fit in a single SNMP SET)
- The following status exist:

active	The active conceptual row is active and ready to use.
notInService	The conceptual row exists in the agent. It is however not available for use (this status is associated with a timer so that unused lines do not exist indefinitely).
notReady	The conceptual row exists in the agent, however it is still incomplete and requires further information.
createAndGo	Set by the manager, in order to produce a new line which immediately becomes active.
createAndWait	Set by the manager, in order to produce a new line with notReady.
destroy	Set by a manager in order to delete a row (e.g. set an instance to an invalid value).

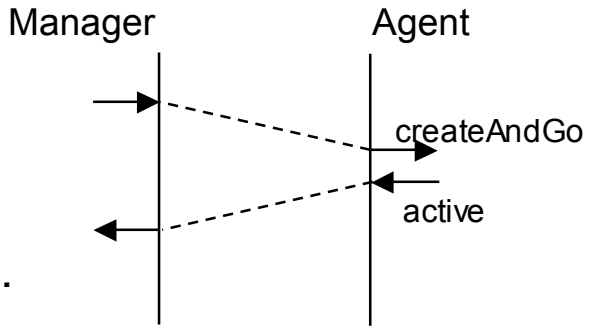
RowStatus State Transition Diagram



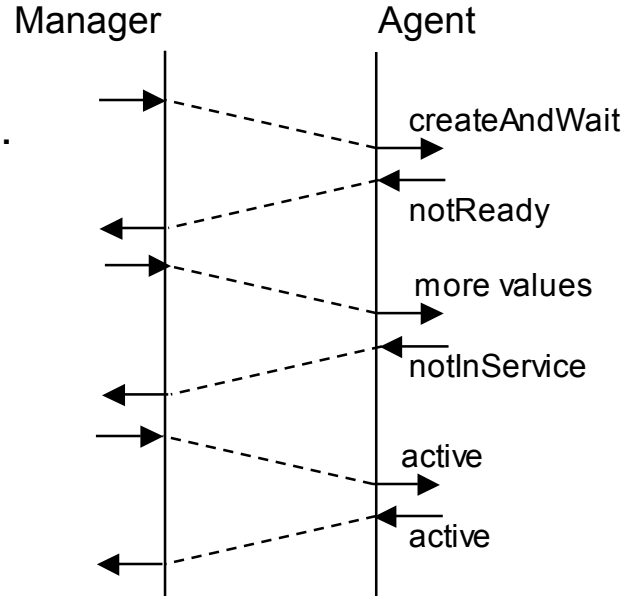
- The state transition diagram is simplified as for some transitions (e.g. notReady to active) further conditions must be met.
- Unspecified values can produce an SNMP error message, but no status change.

One-shot Mode vs. Dribble Mode

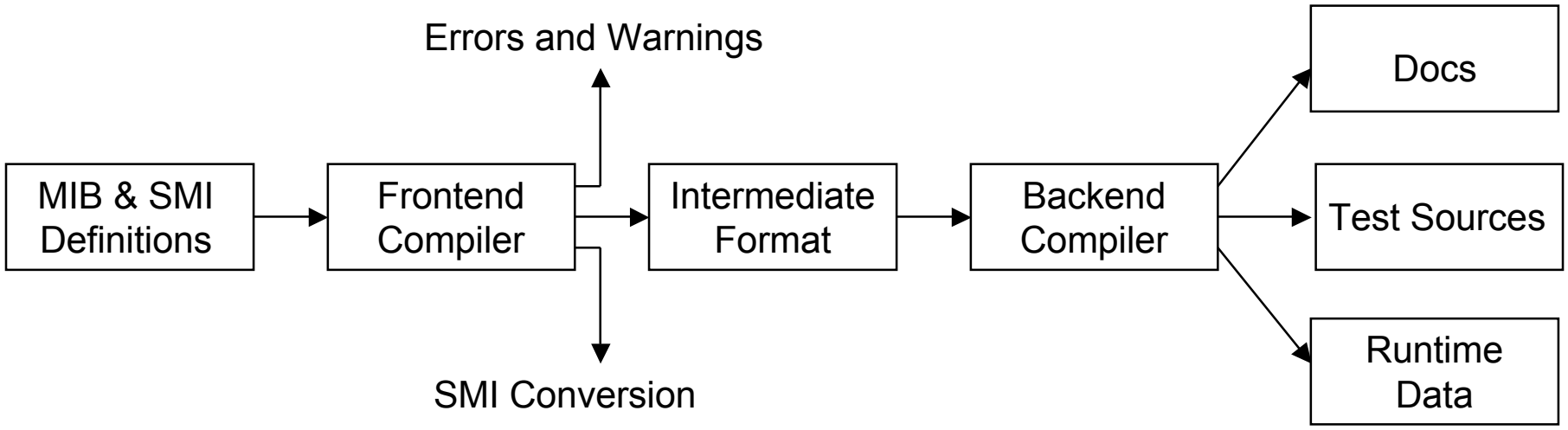
- One-shot mode:
  - Identify an unused instance identifier.
  - Issue a SET for such instance identifier with RowStatus=createAndGo and all variables which are necessary in order to initialise a row completely.



- Dribble mode:
  - Identify an unused instance identifier
  - Create a row by setting RowStatus=createAndWait.
  - Missing values are specified with further SET operations.
  - When all the values have been specified, RowStatus is set to active.



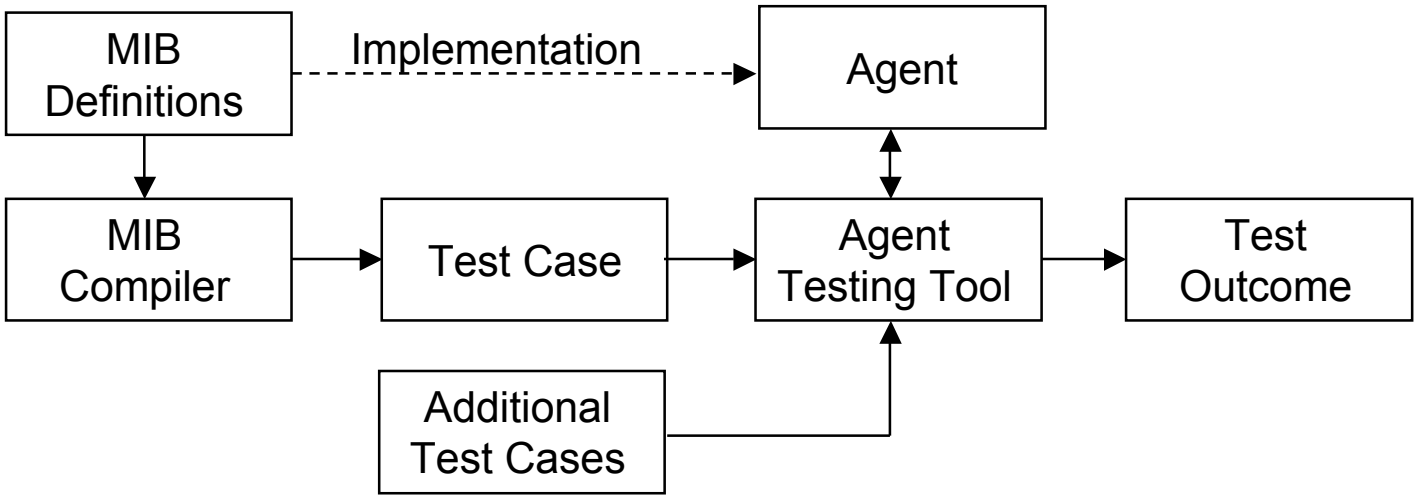
MIB-Compiler



- Backend-Compiler can produce the following outputs:
  - Documentation (hypertext versions of MIB modules, diagrams)
  - Source code for the semiautomatic implementation of agents
  - Test-cases for testing manager and agent implementations
  - Inputs for management applications, the MIB definitions needed at run-time.

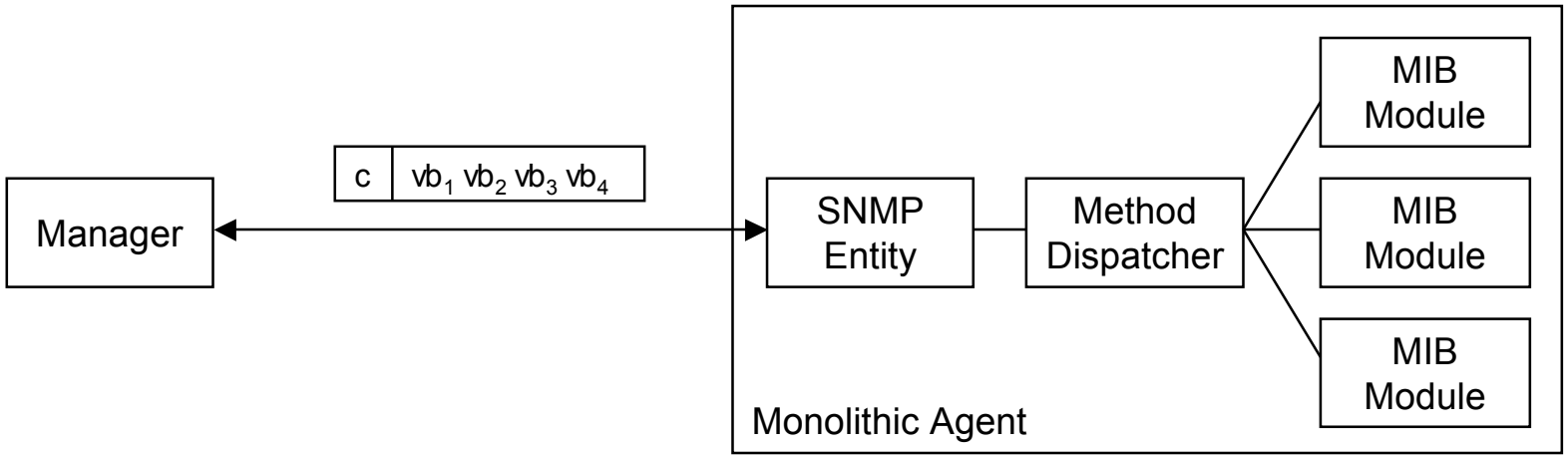
- There is no standardised or generally accepted intermediate format.

Agent Test Cases



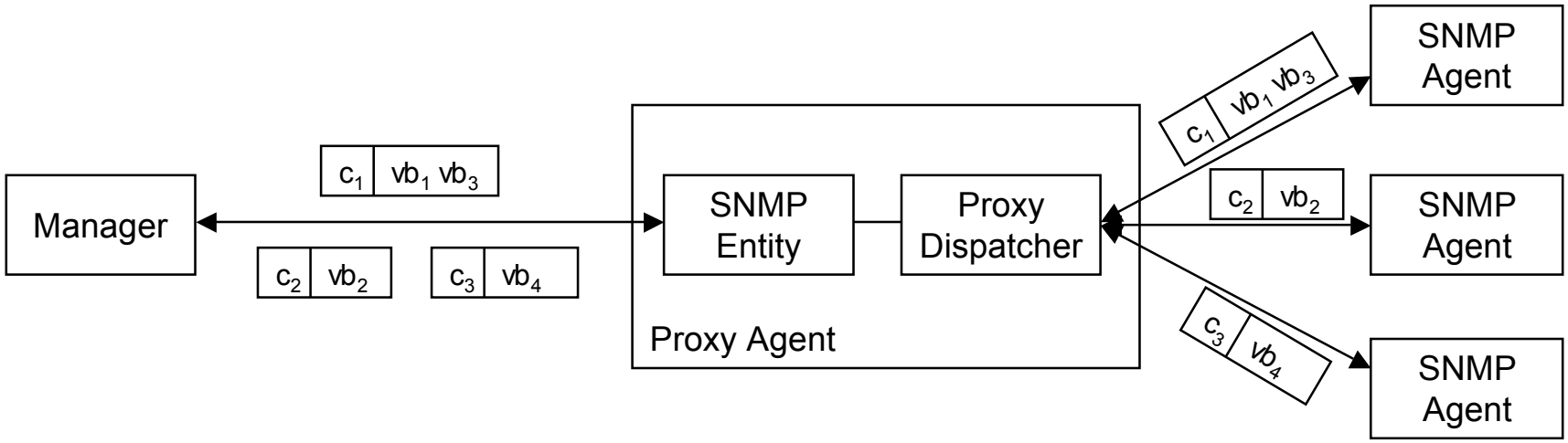
- Automatic regression tests for quality assurance.
- Many test cases can be generated automatically from MIB definitions:
  - Tests for the validation of the object types (type mismatch check).
  - Tests for checking correct protocol behaviour in particular with incorrect protocol messages.
  - Tests for checking correct lexicographical order of MIB instances.
- Additional in-depth test cases should be written by an independent (i.e. not the one who developed the agent) software developer on base of the MIB definition.

Monolithic Agents



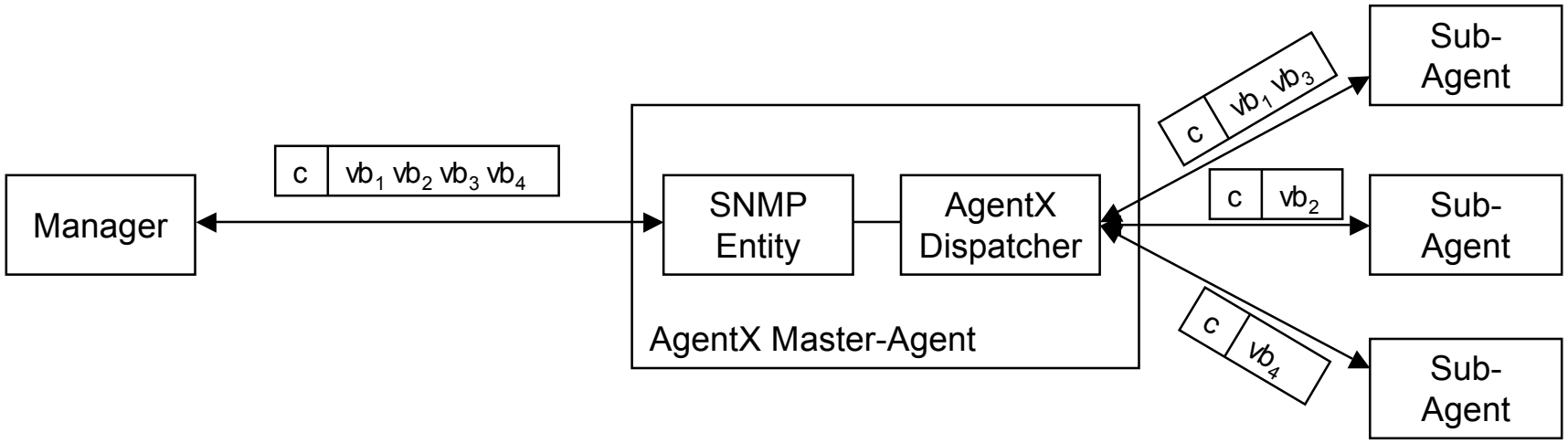
- A monolithic agent is normally implemented by an individual process which contains the SNMP protocol machine and the MIB instrumentation.
- The supported MIB modules is determined at compilation time.
- The method dispatchers is called during processing of SNMP messages, which can either read or modify values from relevant resources.

Proxy-Agents



- SNMP Proxy agents permit managers to access other SNMP agents that are not reachable directly (e.g. behind a firewall) or that are reachable using non IP protocols (e.g. IPX).
- Management applications must (usually) select the appropriate community string or context in order to enable the proxy to reach the agents (no transparency).
- Proxy are important for the implementation of firewalls or for conversion between different SNMP protocol versions.

Extensible Agents



- Extensible SNMP agents separate the SNMP protocol machine (master agent) from the MIB instrumentation (subagent).
- MIB modules can be added by starting further subagents dynamically at runtime.
- Expandable agents are transparent for management applications.
- A special protocol regulate communications between the master agent and the subagents

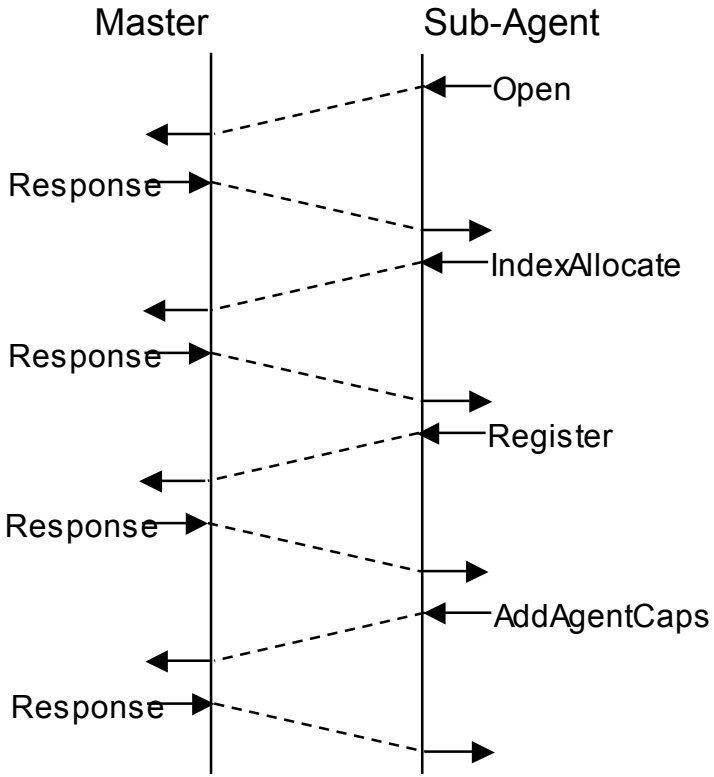
AgentX-Protocol Version 1 (RFC 2257)

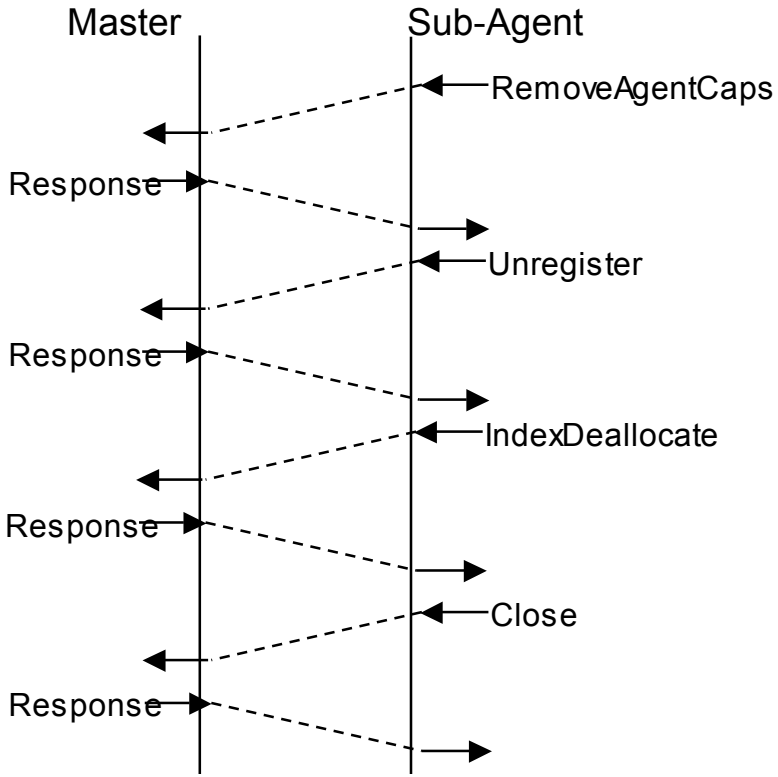
- The AgentX protocol is a new standard protocol for the implementation of expandable SNMP agents.

- AgentX Message Coding:
  - No ASN.1 coding.
  - Compact representation of object identifier values by coding repetitive OID prefixes.
  - Byte order is selected by the subagent (no transformations necessarily, if master agent and subagent on the same system).

- AgentX Message Transport:
  - TCP connections to the port 705.
    (It is possible to have several AgentX sessions over the same TCP connection)
  - UNIX Domain Sockets (/var/agentx/master).
  - Can be likewise used other local (not standardised) IPC mechanisms.

Administrative AgentX Protocol Operations





- AgentX Session Establishment
- Index Allocation
- MIB Registration
- Registration of the Agent Capabilities

- Deregistration of the Agent Capabilities
- MIB Deregistration
- Free of allocated indexes
- AgentX Session Termination

Index-Allocation, OID Registration, Scoping

- Index allocation for common tables between subagents:
  - Allocation of specific (private) indexes.
  - Allocation of indexes not used at present.
  - Allocation of indexes no longer in use.

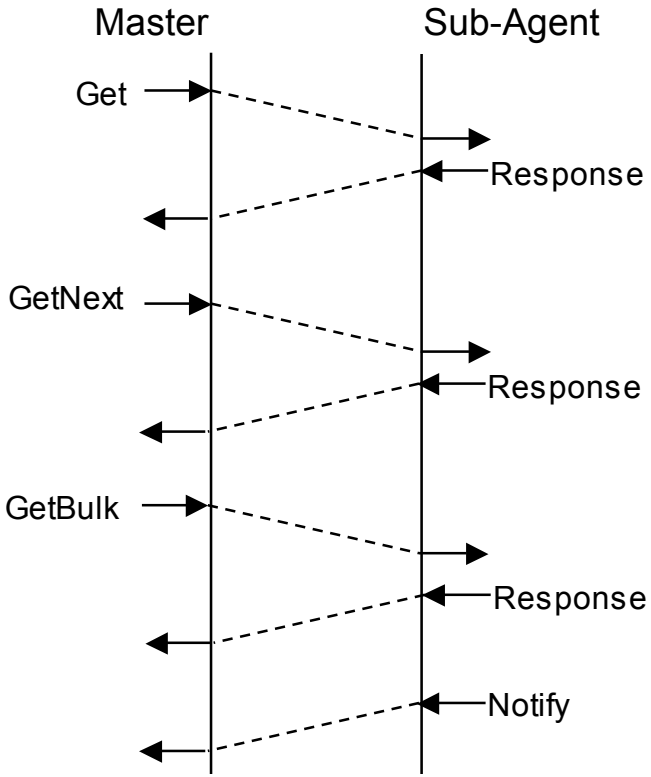
- OID Registration:
  - Registration of individual instances (instance level registration)

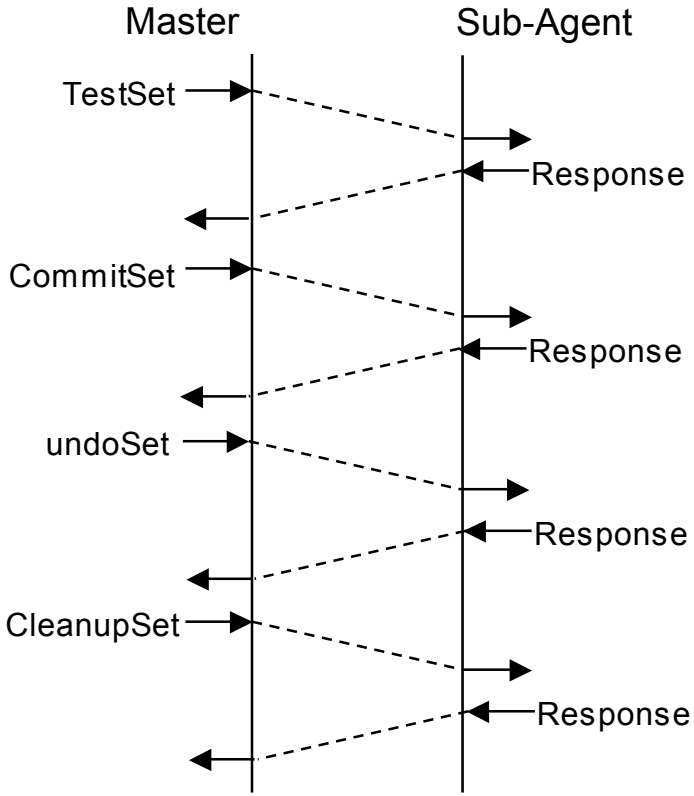
    1.3.6.1.2.1.2.2.1.1.42 (ifIndex.42)
    1.3.6.1.2.1.2.2.1.2.42 (ifDescr.42)
    1.3.6.1.2.1.2.2.1.3.42 (ifType.42)
  - Registration of MIB Ranges:

    1.3.6.1.2.1.2.2.1.[1-22].42 (ifIndex.42 - ifSpecific.42)

- Scoping:
  - AgentX can specify scoping with GetBulk operations (similar to CMIP Scope).

AgentX Protocol Operations for SNMP Operations





- SNMP-operations correspond to AgentX operations.
- A SNMP operation can concern several subagents.

- Atomicity of SNMP SET operations is guaranteed by the AgentX protocol.

Why did SNMP Succeed?

- Standards can be obtained for free by everybody whereas OSI standards cost money and can be distributed only to the members of the association.

- Standards are freely available from FTP/WWW servers in an electronic form whereas OSI standards are usually on printed paper and need to be ordered by plain mail to ITU that is based on Geneva.

- SNMP standards need a relatively short period of time for their standardisation. This has enabled many organisations to define new standards. OSI standards require a long time to evolve and several committee votations.

- Prototypes must demonstrate the need for and the feasibility of Internet standards. OSI standards are usually first defined on papers and finally implemented. In order to produce an RFC it is required to release two or more independent implementation of the standard in order to prove standard's feasibility.

Example of Free SNMP Implementations

- NET-SNMP: http://net-snmp.sourceforge.net/
- Scotty: http://wwwhome.cs.utwente.nl/~schoenw/scotty/
- JMAPI: http://java.sun.com/products/JavaManagement/
- Advent: http://www.adventnet.com/

3. OSI Network Management

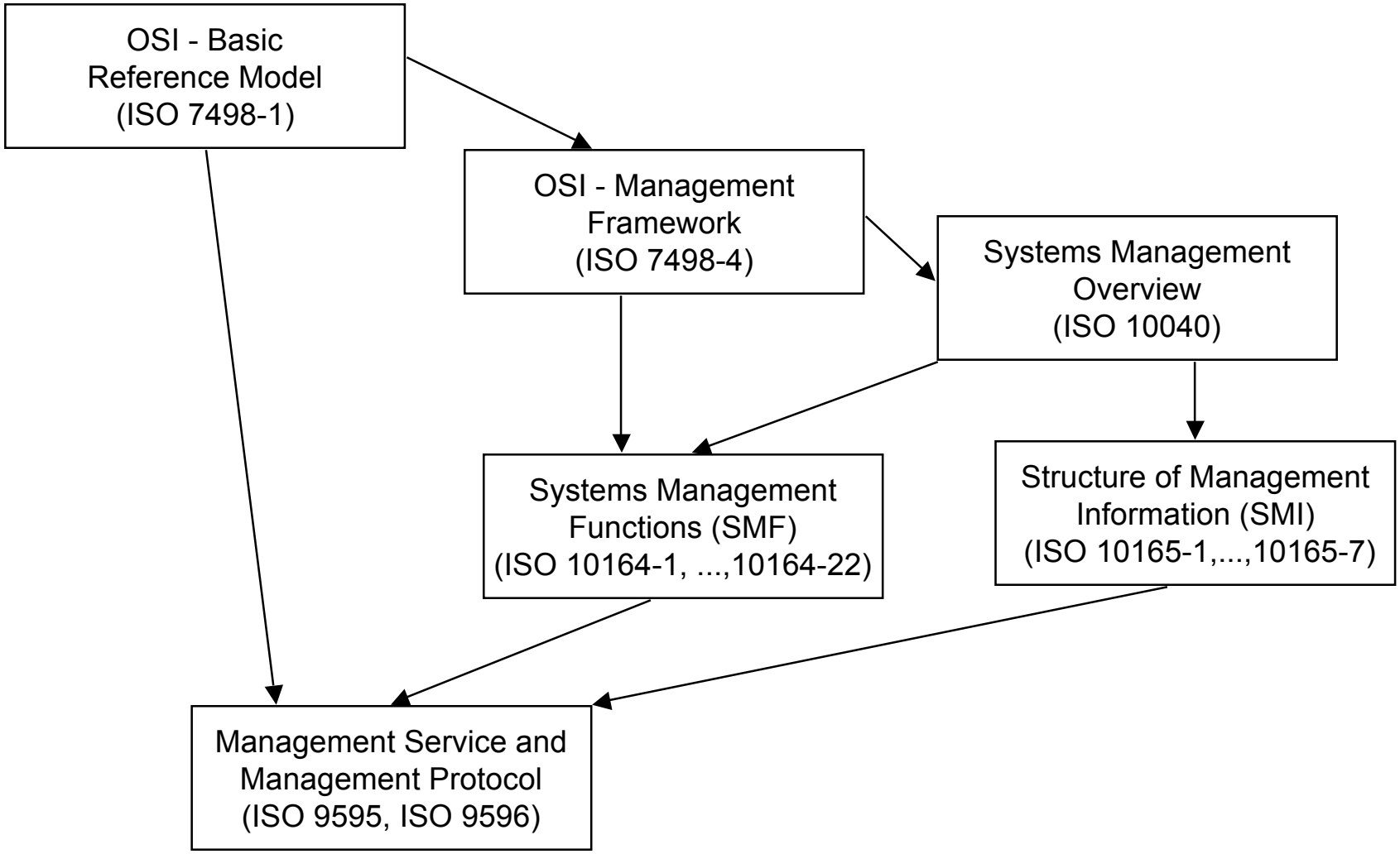
1. Introduction
2. Internet Management
3. **OSI Network Management**
   - 3.1 Overview
   - 3.2 Structure the Management Information (GDMO)
   - 3.3 Common Management Information Protocol (CMIP)
   - 3.4 Management Functions
4. Current Developments

3.1 Overview

1980	ISO/OSI starts activities in the area network management as part of the development of the ISO/OSI reference model.
1985	CCITT (today ITU) begins the definition of the Telecommunication Management Network (TMN)
1987	Beginning of the work on OSI System Management.
1988	Establishment of the OSI Network Management Forum (NMF), a manufacturer union for the conversion of the standards.
1988-1992	Adjustment of the ISO/OSI standards and the TMN standards TMN is based on the ISO/OSI standards.
1991	OSI Systems Management Overview becomes International Standard.
1990-1997	Deployment of Standards, the basic management functions are defined (e.g. forwarding and filtering of messages).

- OSI has small meaning for data communication, in particular since the success of the Internet.
- Still the most important standard within the area of the telecommunication systems, in particular in Europe.

Standards Overview



Types of Management in the OSI Model

- Protocol Management:
  - A protocol specifies the mechanisms and functions that the management protocol must support. Example: Token management in Token-Ring.

- Layer Management:
  - Management functions to be implemented by a layer in the OSI reference model and by the System Management.
  - Example : Mechanisms for the initialisation of an OSI environment.

- System Management:
  - All management functions necessary for the operation of an OSI communication system.
  - Prerequisites an operational OSI protocol stack including the management protocols in the layer 7 of the reference model.
  - The OSI term "system management" refer to a OSI communication system and is different from the "system management" term used for computers.

3.2 Structure the Management-Information (GDMO)

- The OSI information model is based on an object-oriented paradigm that specifies the type of data, classes, definition and data transmission.

- Each *Managed Object* is an instance of a *Managed Object Class* (MOC), and specifies the attributes, behaviour, executable operations, messages generated by the managed object and optional elements.

- The behaviour is defined in plain English and defines the relations between attribute values:
  - ↪ the semantic of the attributes, operations and notifications.
  - ↪ the meaning of the return values of operations.
  - ↪ the circumstances where a managed object emits a message (operation).
  - ↪ the relations among the instance attributes.
  - ↪ the relations with other managed objects.

OSI Information Model

- The following operations can be executed on an attribute:
  - Attribute read (get attribute value)
  - Attribute set (replace attribute value)
  - Attribute set to the standard value (if any) (set attribute value to default)
  - Add a value to a quantity attribute (add member)
  - Remove a value from a quantity attribute (remove member)

- The following operations can be executed on a MO:
  - Instantiation of a new managed object (create)
  - Deletion of an existing managed objects (delete)
  - Execution of an operation on a managed object (action)

- Notifications are messages emitted asynchronous by a MO at each state change.

- Packages permit the modular structure of MOs. It is differentiated between mandatory constituents and optional package members.

Registration, Inheritance and Containment

- Registration:
  - All definitions of a managed object class are registered in the ISO registration tree in order to uniquely identify it.

- Inheritance:
  - Managed object classes can be derived from other managed object classes (inheritance of attributes, operations and notifications).
  - Promotes the creation of reusable, generic definitions.
  - The tree hierarchy imposes the specialisation of managed object classes on the inheritance tree.
  - A managed object class can be derived from several managed object classes (multiple inheritance).

- Containment:
  - Models the "be contained in" relationship between managed objects.
  - A managed object can be contained in exactly one managed object (tree containment).
  - Defines the "be contained in" relationship by means of naming.

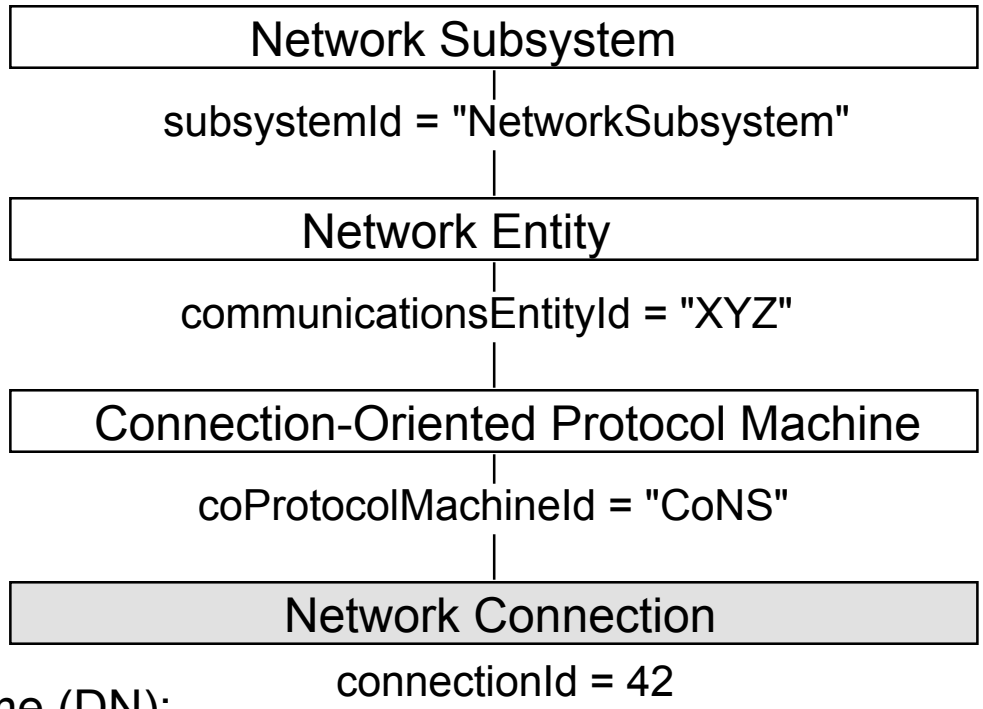
Naming

- Local Form:
  - A local name is used for the unique identification of a managed objects within the enclosing managed objects.
  - The local name is specified by means of a *relative distinguished name* (RDN)
  - A RDN is defined by the value of an identified attribute.
  - Is defined with attribute value tuples, as *attributes values association* (AVA).

- Global Form:
  - A global name specified for contained management ranging from global names of the containing managed objects to local object names.
  - The global name is named *distinguished name* (DN).
  - A DN is a sequence of AVAs.

- The objects at the root of the name tree have firmly given attributes for the definition of AVAs.

Containment Example



- Distinguished Name (DN):

  systemID="University of Pisa", subsystemId="Network Subsystem", communicationsEntityId="XYZ", coProtocolMachineId="CoNS", connectionId=42

- Relative Distinguished Name (RDN):

  connectionId=42

Allomorphism

- A managed object of class C_1 behaves exactly as a managed object of the class C_2.
- Classes C_1 and C_2 do not have to be in a leaving (father-son) relationship.
- The implementation must guarantee that only the characteristics of the class are visible from outside of C_2:
  - ↪ Operations, which concern all attributes of the class C_2 are executed only on the attributes defined in the class C_2.
  - ↪ Notifications are passed on, only if the messages are defined in the class C_2.

  Additional operations are determined by the real class affiliation. For example with "set attributes VALUE to default" operation the specification for the class C_1 avoids to create inconsistencies.
- Allomorphism is rather important for enabling the transition of a MIB to a new version.

GDMO Templates

- The *Guidelines for the definition of Managed Objects* (GDMO) determine the notation, user for the definition of the Managed Objects.

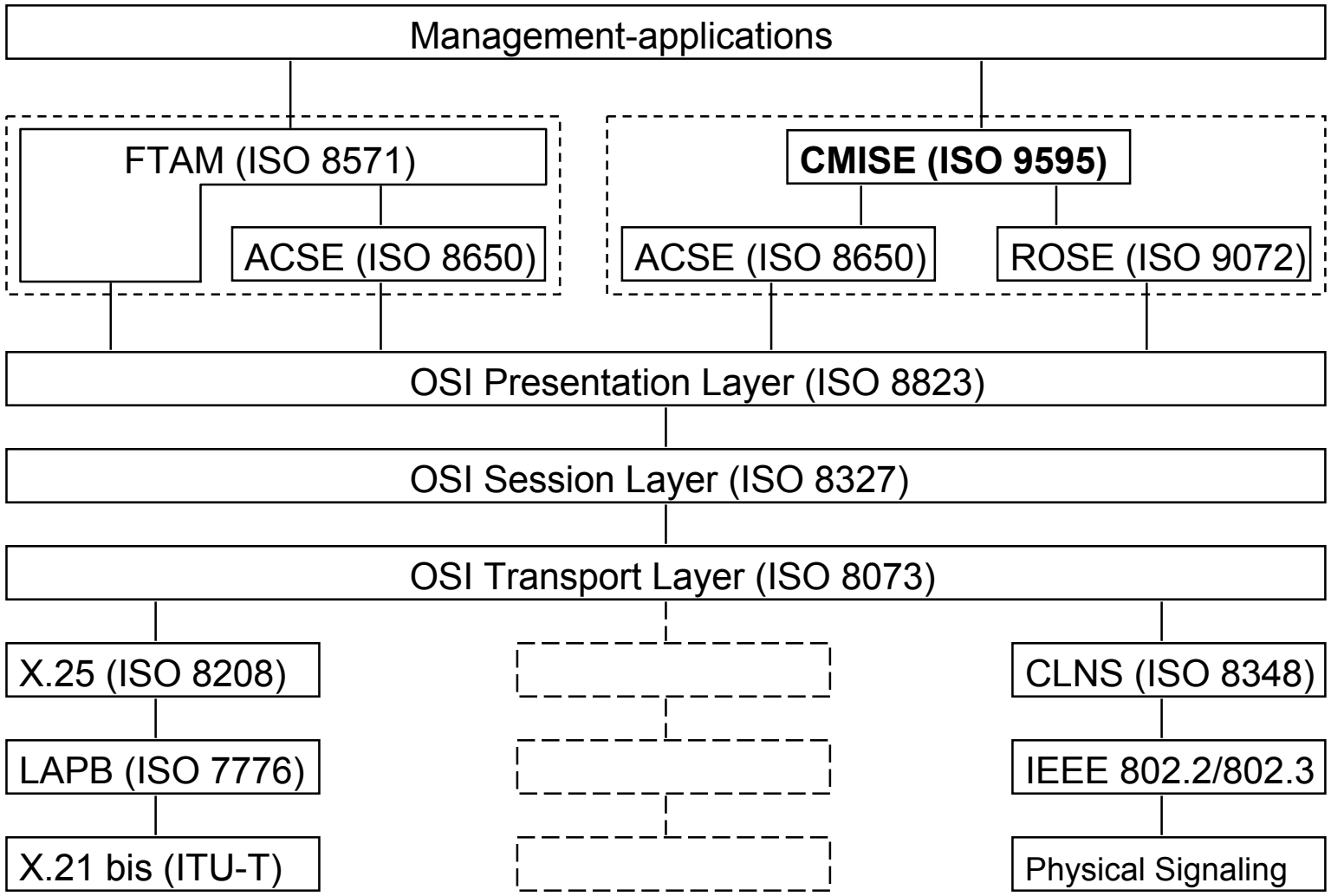
- For MO definition are used templates based on the language ASN.1:
  - managed object class template — Definition of a managed object class.
  - package template — Logical group of related definitions.
  - parameter template — Extension mechanism through parameter.
  - attribute template — Definition of individual attributes.
  - attribute group template — Grouping of attributes.
  - behaviour template — Describes the behaviour of managed objects.
  - action template — Defines the operations on a managed object.
  - notification template — Definition the notifications that can be emitted.
  - name binding template — Defines the legal positioning in the naming tree.

- References between individual templates are implemented based on the registration by means of references.

GDMO Example

```
pduCounterObject MANAGED OBJECT CLASS
   DERIVED FROM "CCITT REC.X.721(1992)|ISO/IEC 10165-2: 1992":top;
   CHARACTERIZED BY
        basePackage PACKAGE    -- in-line PACKAGE definition
                ATTRIBUTES
                        pduCounterName
                                GET;
                        pduCounter
                                INITIAL VALUE syntax.initialZero
                                GET;
        ; -- end of in-line PACKAGE definition
   ; -- end of CHARACTERIZED BY construct
   CONDITIONAL PACKAGES additionalPackages
        PRESENT IF *enable/disable control is required*;
REGISTERED AS { object-Identifier 1 }
```

3.3 Common Management Information Protocol (CMIP)



Protocols and Services Overview

- Common Management Information Service Element (CMISE):
  - Services for accessing the management information.
  - Based on ACSE and ROSE
- Common Management Information Protocol (CMIP):
  - Protocol used for the implementation of the CMISE-Services
- Association Control Service Element (ACSE):
  - Services for the setup and deletion of connections between applications.
- Remote Operations Service Element (ROSE):
  - OSI equivalent of the Remote Procedure Call (RPC).
- File Transfer, Access and Management (FTAM):
  - Protocol for data transfer (similar to Internet FTP).

CMIS Service Primitives

- Operations on Attributes of Managed Objects:
  - Attribute value read — `M-GET`
  - Attribute value modification — `M-SET`
  - Abort of an in progress `M-GET` — `M-CANCEL-GET`
- Operations on Managed Objects:
  - Creation of Managed Objects — `M-CREATE`
  - Deletion of Managed Objects — `M-DELETE`
  - Execution of an operation on a Managed Object — `M-ACTION`
- Event Transmission:
  - Transmission of event messages — `M-EVENT-REPORT`

- M-SET and M-ACTION can acknowledged (confirmed) and unconfirmed. Remaining service are always acknowledged (confirmed).
- CMIS services can be applied to several managed objects and attributes of a managed objects (scoping).
- The services that concern several managed objects or attributes can be called with "best effort" or "atomic" synchronisation.

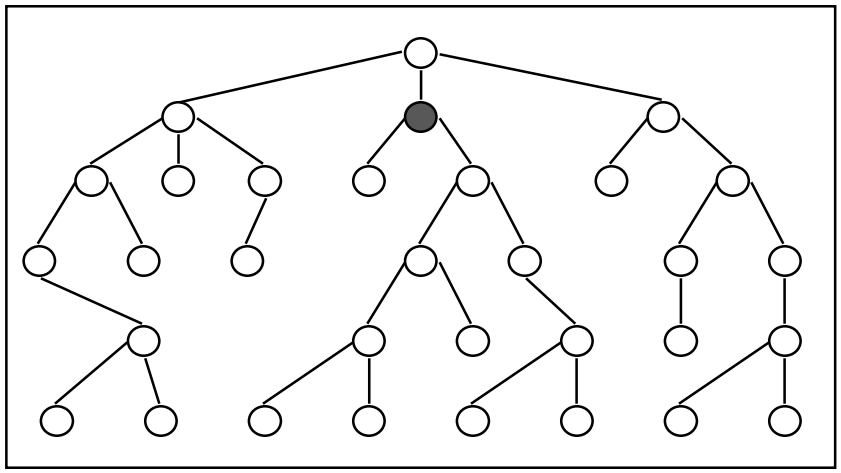
Scoping

- For *scoping* one understands the selection of several managed objects for the execution of the same operation due to its position in the containment tree.

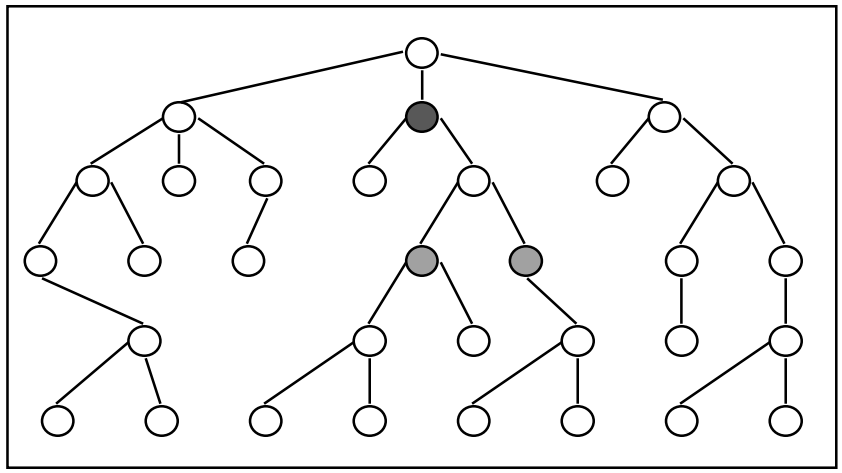
- Selections can be based:
  - the base instance only
  - all object instances of the n^{th} layer underneath the base instance
  - From the base instance to the n^{th} layer underneath the base instance
  - The entire tree including the base instance

- Scoping can be used for `M-GET`, `M-SET`, `M-ACTION` and `M-DELETE`.

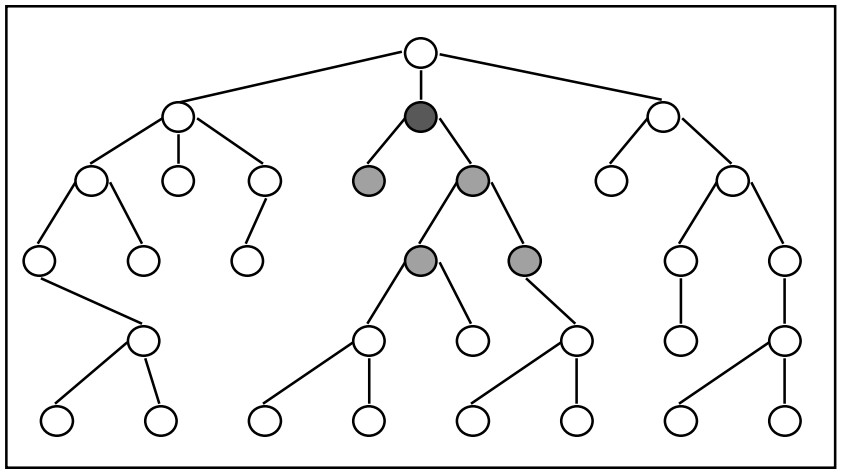
Scoping Example



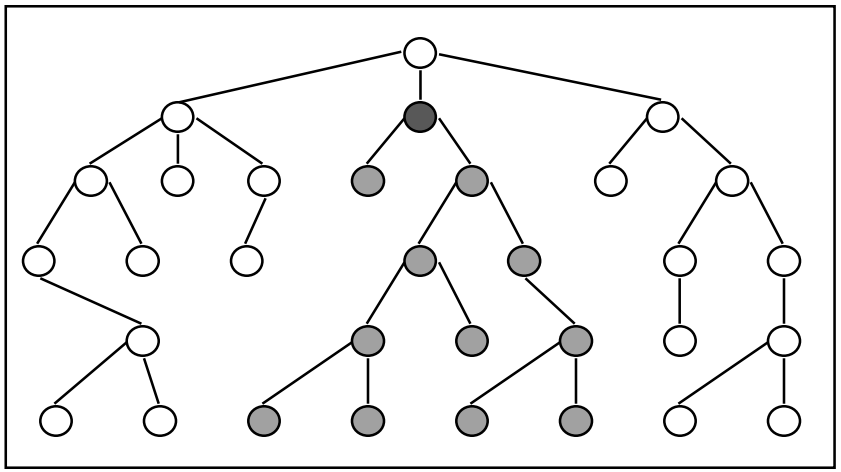
Basis instance



Instances of the n^{th} layer



All instances up to the n^{th} layer



All instances under the base instance

Filtering

- For Filtering one understands the selection of several Managed Objects for the execution of the same operation based on the instance attribute values.

- A test is defined by a filter printout, which may contain boolean operators `and`, `or`, `not` and the predicates and relations `equality`, `substrings`, `greaterOrEqual`, `lessOrEqual`, `present` (for optional attributes), and `subsetOf`, `supersetOf` and `nonNullSetIntersection` (for multivalue attributes only).

- If a test is applied to an not existing attribute, then the test fails, and thus the operation on the Managed Object is not executed..

- The filter is applied only on those instances selected by scoping.

Filtering Example

Filter all managed objects of the class protocolEntity whose names begins with "123 " and the attribute "severity" is different than "minor" and "badPduCout" has a value of less or equal to "20":

```
test-filter CMISFilter ::=
and {
   item equality {objectClass, Object-Class protocolEntity},
   item substrings {initialString {entityID, PrintableString ''123''}},
   or {
      not item equality {severity, Severity minor},
      item lessOrEqual {badPduCount INTEGER 20}
   }
}
```

- Special tools normally translate filters from a user friendly syntax to the notation required by the protocol.

Synchronization

- As seen before:
  - Scoping selects the set of MOs under the specified base instance on which the specified operation (e.g. M-GET) will be applied.
  - Filtering further restricts the selection: only on the scoped MOs that satisfy the filter the operation will be applied.

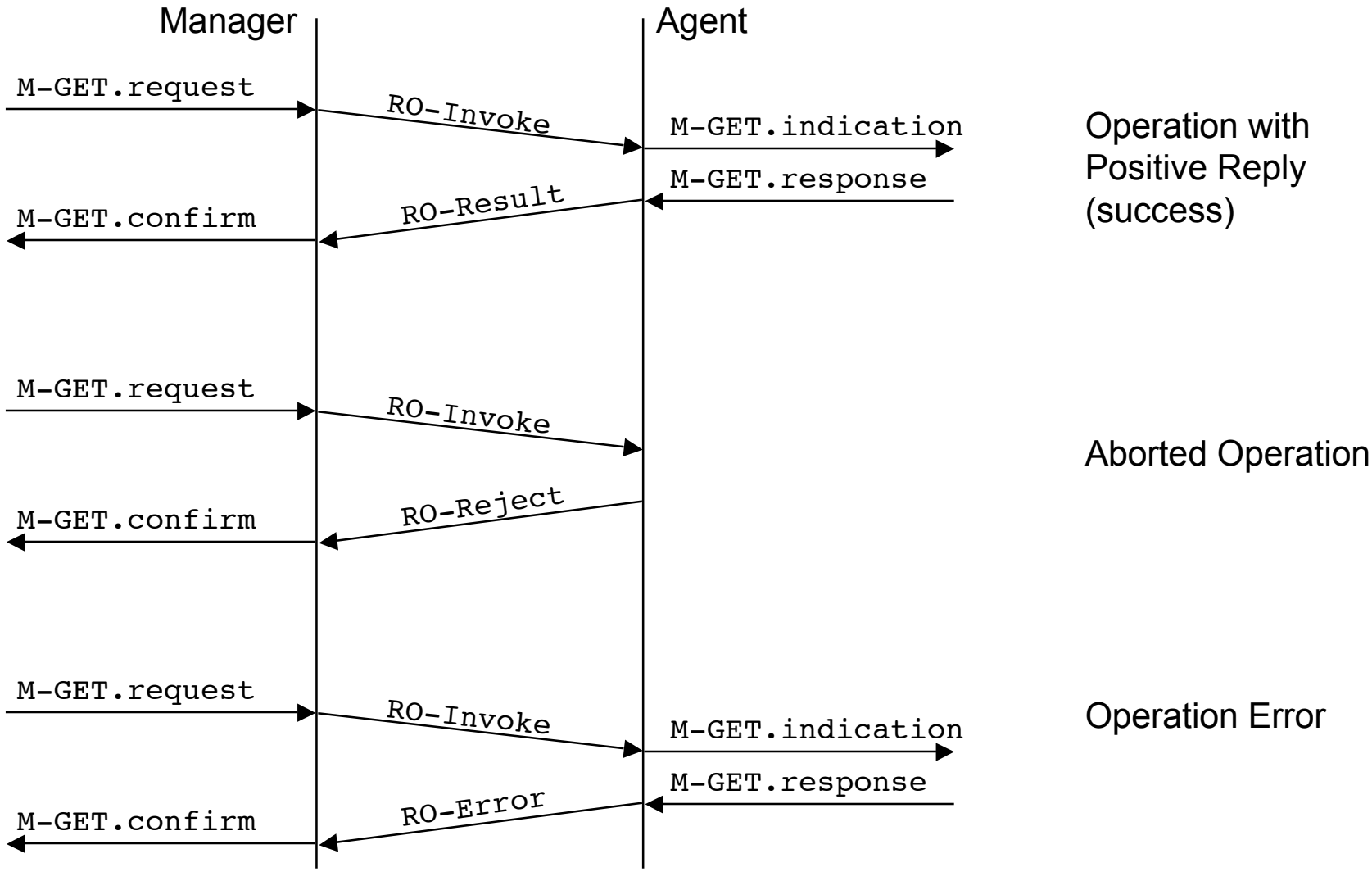
- Synchronization specifies the operation behavior in case of failure.

- CMIP supports two types of synchronizations:
  - Best effort: in case of failure on one or more MOs , the agent returns negative responses for those MOs where the operation failed, and positive replies for the remaining MOs .
  - Atomic: in case of failure on one or more Mos, the whole operation fails and only negative replies are defined.

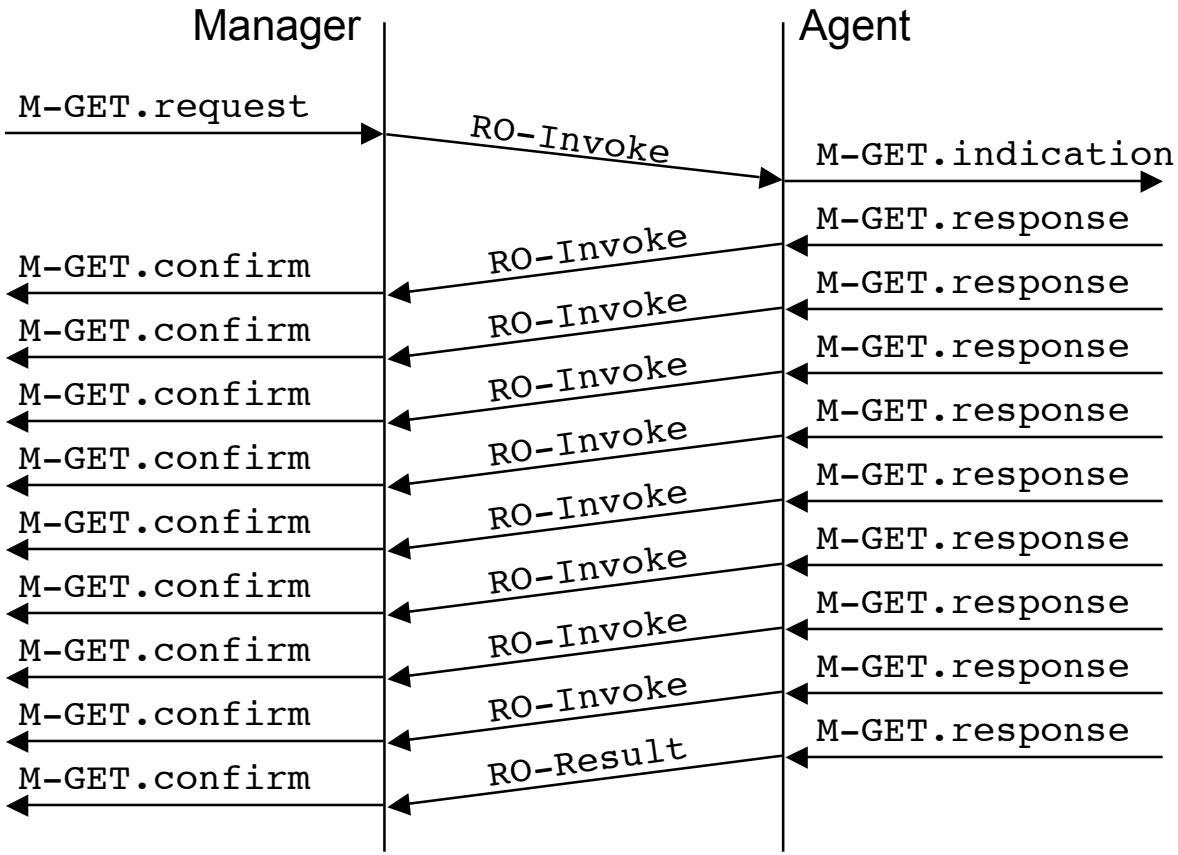
Scoping, Filtering and Synchronization

	Scoping	Filtering	Synchronization
M-CREATE	No	No	No
M-DELETE	Yes	Yes	Yes
M-GET	Yes	Yes	Yes
M-SET	Yes	Yes	Yes
M-EVENT REPORT	No	No	No
M-CANCEL GET	No	No	No
M-ACTION	Yes	Yes	Yes

Diagram for the M-GET service



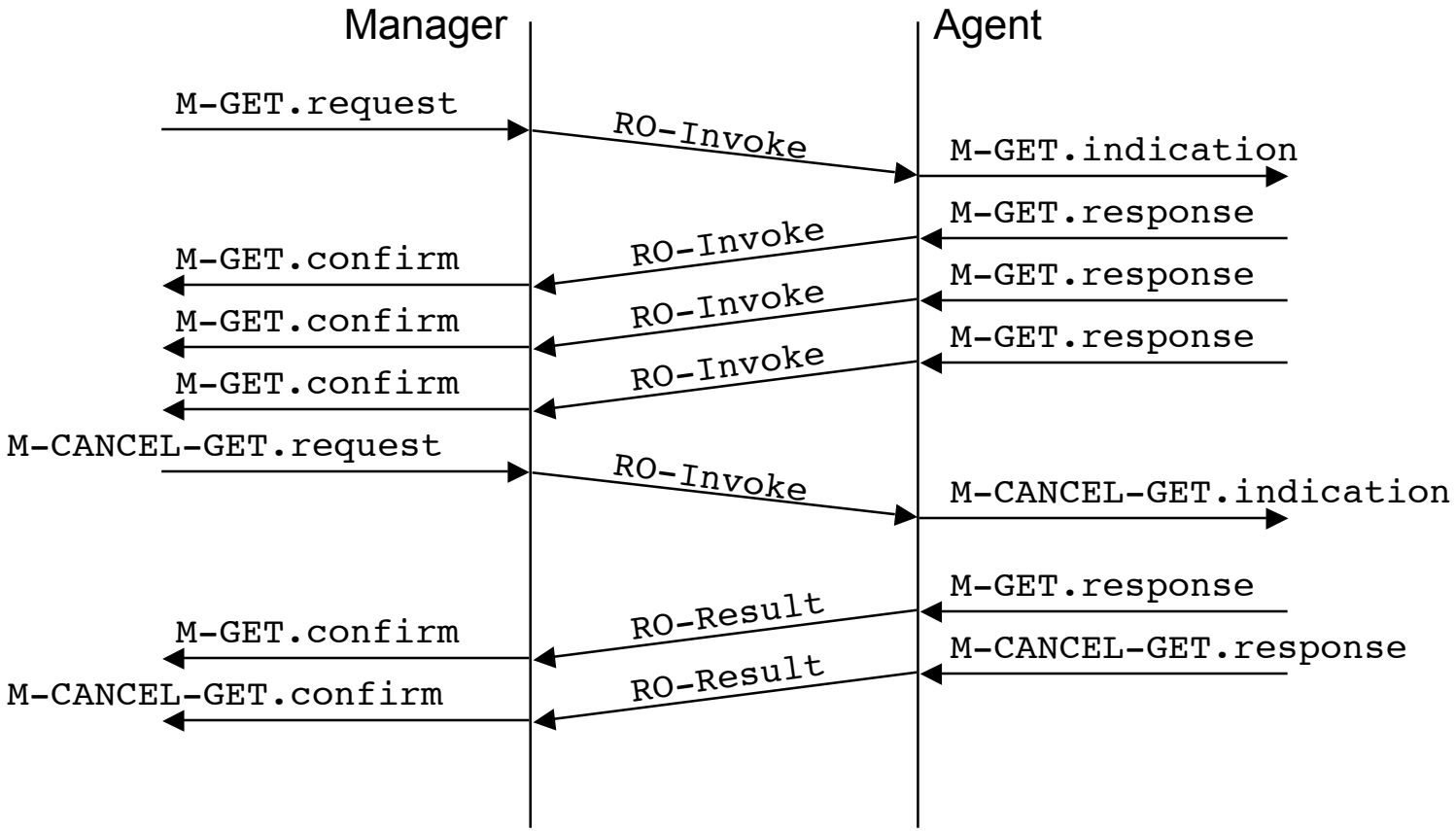
M-GET Service with Multiple Responses



Multiple results are returned when the operation is applied to multiple MOs

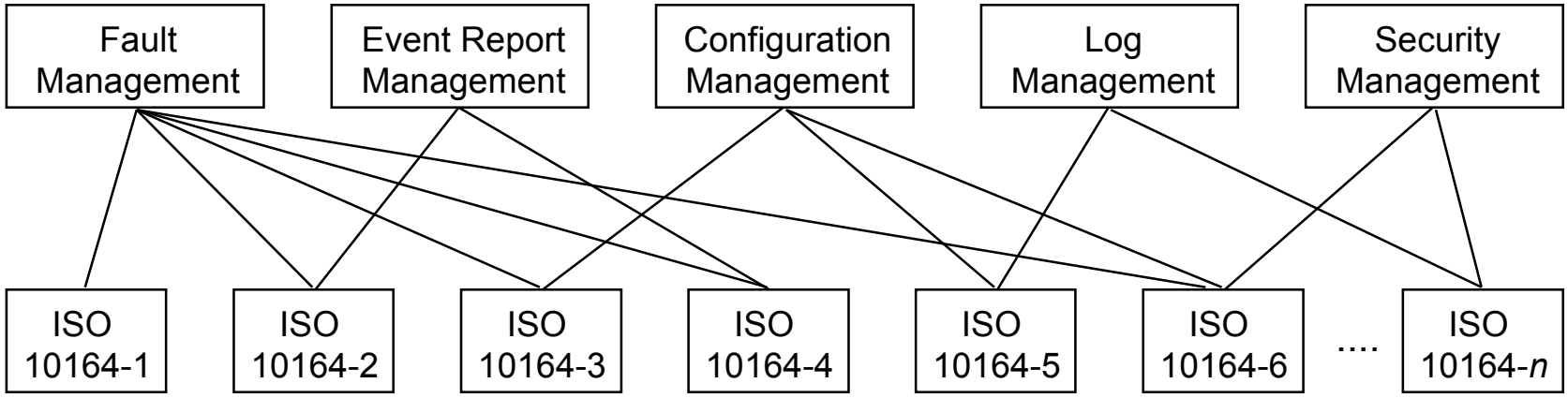
- Several results are implemented by a "role exchange": the calling manager becomes the recipient of calls, which are started by the agent.

M-GET Abort by M-CANCEL-GET



- Issued during the processing of a call with n>1 of results: the results 1... n are returned by means of RO Invoke. Subsequent results are sent by means of RO Result for the termination of the original service call.

3.4 Management Functions



- For the implementation of the 5 functional areas (FCAPS) several management functions have been developed and standardised (system management function, SMF).

- The management functions represent generic basic modules, with which the 5 functional areas can be implemented.

OSI-Management Functions (ISO Standards)

❑ Object Management Functions	ISO 10164-1:1993 X.730
❑ State Management Functions	ISO 10164-2:1993 X.731
❑ Attributes for Representing Relationships	ISO 10164-3:1993 X.732
❑ Alarm Reporting Functions	ISO 10164-4:1992 X.733
❑ Event Report Management Function	ISO 10164-5:1993 X.734
❑ Log Control Function	ISO 10164-6:1993 X.735
❑ Security Alarm Reporting Function	ISO 10164-7:1992 X.736
❑ Security Audit Trail Function	ISO 10164-8:1993 X.740
❑ Objects and Attributes for Access Control	ISO 10164-9:1995 X.741
❑ Usage Metering Function	ISO 10164-10:1995 X.742
❑ Metric Objects and Attributes	ISO 10164-11:1994 X.739
❑ Test Management Function	ISO 10164-12:1994 X.745
❑ Summarization Function	ISO 10164-13:1995 X.738
❑ Confidence and Diagnostic Test Categories	ISO 10164-14:1996
❑ Scheduling Function	ISO 10164-15:1995 X.746
❑ Management Knowledge Management Function	ISO 10164-16:1997
❑ Change over Function	ISO 10164-17:1996
❑ Software Management Function	ISO 10164-18:1997

OSI-Management Functions (Draft Standards)

- Domain and Policy Management Function — ISO 10164-19
- Time Management Function — ISO 10164-20
- Command Sequencer — ISO 10164-21
- Response Time Monitoring Function — ISO 10164-22

- Most OSI Management functions are taken over after short time of the ITU and published with same contents as X.7 ** recommendation..

- Further management functions are under development.

- Relatively current outlines can be found on the WWW pages of the ISO or ITU.

- ISO or International Telecommunication Union documents are (finally) freely available on their respective web sites.

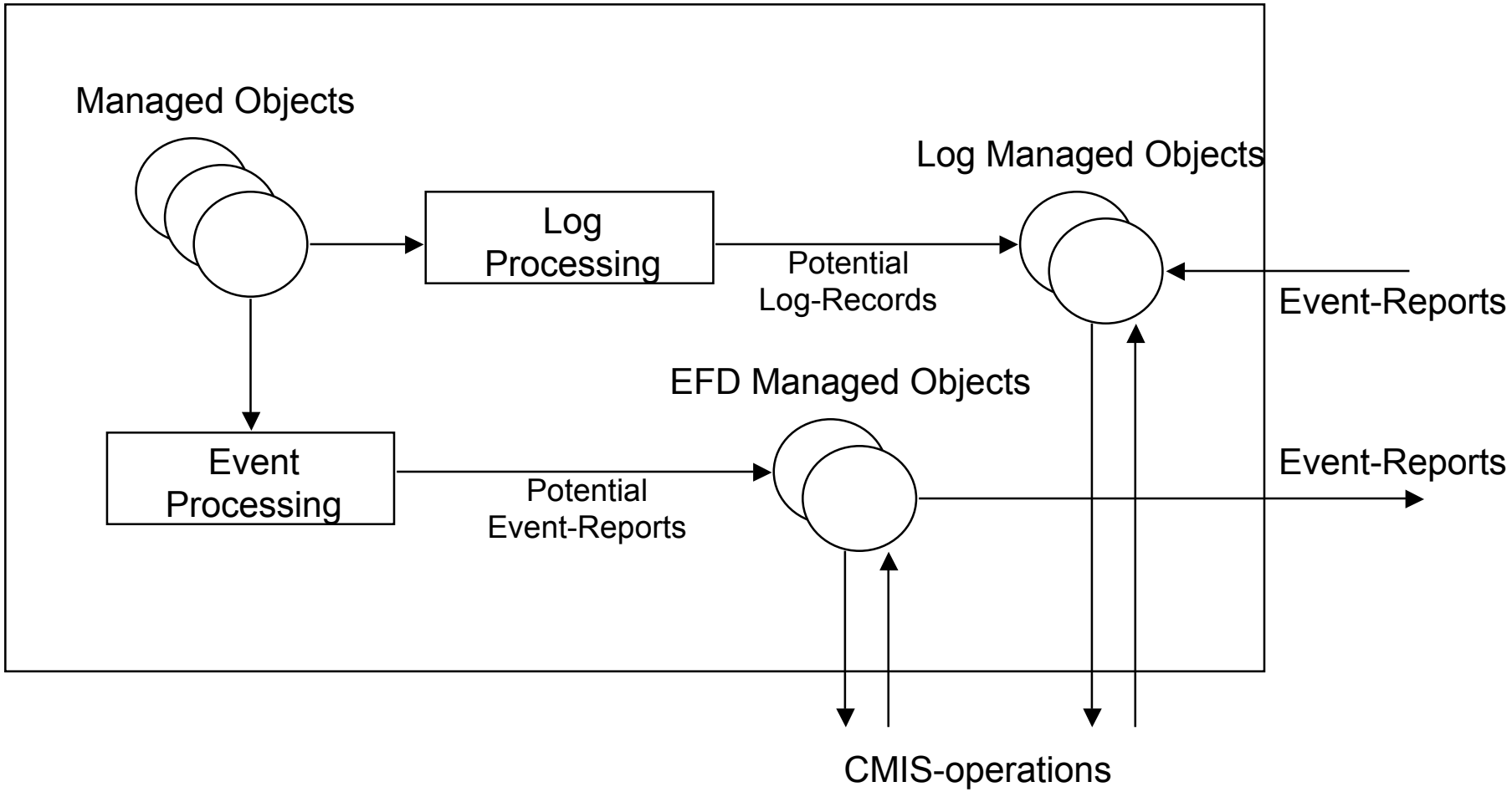
Event Forwarding and Storage

- Event Report Management Function ISO 10164-5 X.734
  - A substantial feature is the filtering of event messages by so-called *Event Forwarding Discriminator* (EFD) Managed Objects. Similar to SNMP Trap.
  - A EFD Managed Object possesses an Discriminator attribute, which contains a CMIS filter, that is used for deciding whether to forwarding the message.

- Log Control Function ISO 10164-6 X.735
  - Log Managed Objects regulate storage event messages from other systems, received from event messages in so-called Log Records.
  - A Log Record can also store event log records belonging to remote systems.

- The OSI event transmission and storage permits an early and distributed filtering or logging of event messages, requiring however relatively complex and efficient agents.

Model of the Event Forwarding and Storage



CMIP vs. SNMP

	CMIP	SNMP
Model	Event Based	Polling Based
Information Approach	Object Oriented	Variable Oriented
Agent Complexity	Complex	Simple
State Information	Kept by the Agent	Kept by the Manager
Underlying Service	CO - reliable	CL - unreliable
Implementation	Difficult	Simple (v2 and v3 not too easy)
Retrieval	Objects	Scalars (tables)
Actions	Supported	Tricky
Objects Selection	Scoping & Filtering	-
Security	Via Underlying Services	Authentication/Encryption/ACLs
Management Functions	Many	Few
Price	High	Low
Market Acceptance	Limited	Large

SNMP vs. ISO vs. TMN

- SNMP (IETF)
  - ↪ Management should be simple
  - ↪ Variable-oriented approach.
  - ↪ Management information exchanges may be unreliable.

- ISO
  - ↪ Management should be powerful.
  - ↪ Object Oriented approach.
  - ↪ Management information must be exchanged in a reliable fashion.

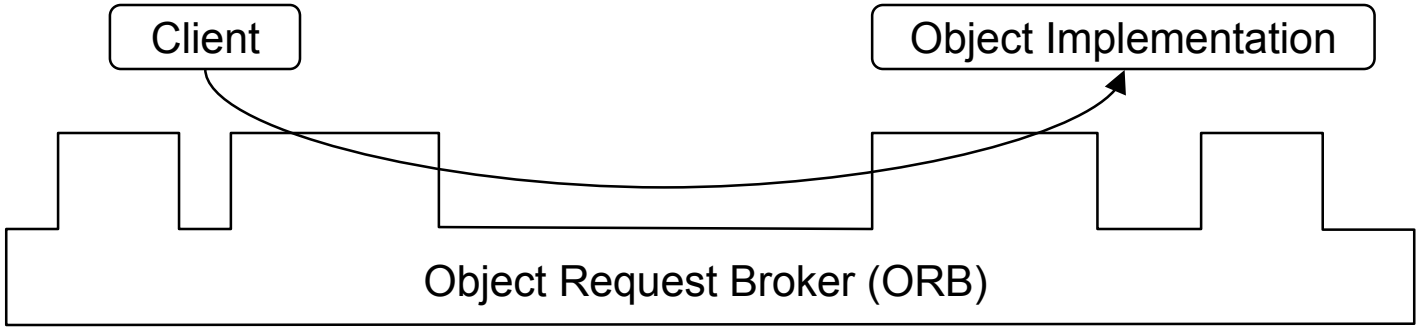
- TMN
  - ↪ It defines only a management architecture.
  - ↪ The actual management protocols have been inherited by OSI.
  - ↪ The management should be out of band.

4. Current Developments

1. Introduction
2. Internet Management
3. OSI Management
4. Current Developments
   - 4.1 Network Management with CORBA
   - 4.2 Java-based Network Management
   - 4.3 Application and Service Management
   - 4.4 Latest Developments

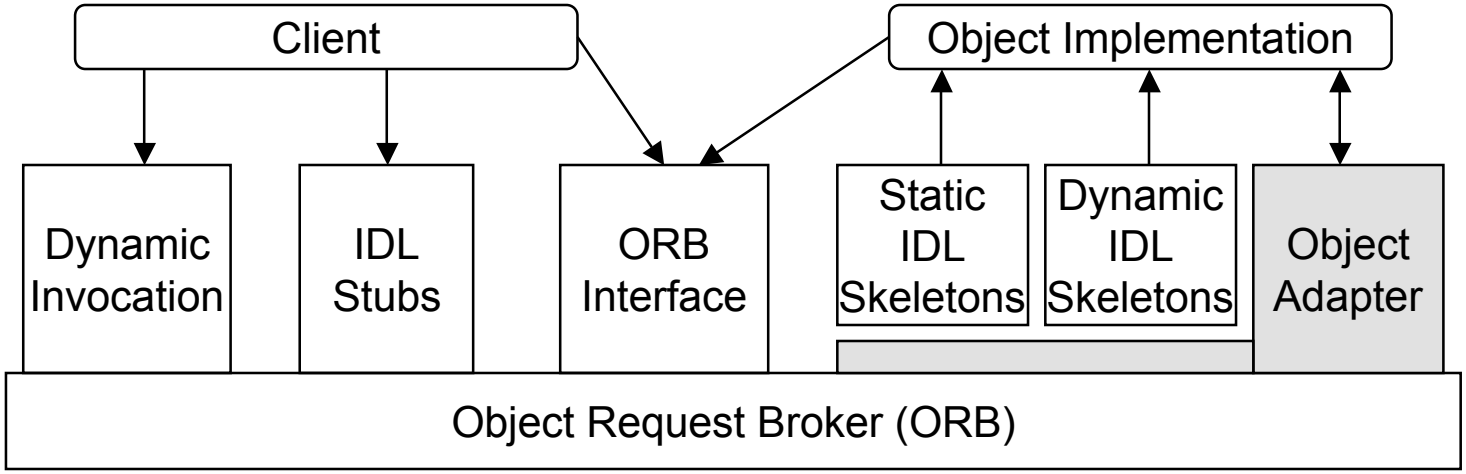
4.1 Common Object Request Broker Architecture (CORBA)

- Goal: a quantity of interacting objects, which can to be implemented in (almost) any programming languages and used without knowledge of their location.



- Corba has been defined by the OMG (Object Management Group) consortium.
- Relevant Corba components:
  - Object Request Broker (ORB): the ORB locates the implementation of the target object and directs client communication to the target (no via-ORB communications).
  - Client: beside the object reference it has no information about the target object.
  - Object Implementation: implements the target object (code and data).
  - Object Reference: uniquely identifies an object in a ORB.

Structure of CORBA 2.0 ORB



- Dynamic Invocation Interface (DII):
  - ↪ Similar to Java RMI, it allows runtime dynamic method/procedure calls.
- Client Stubs:
  - ↪ It implements the procedure/method calls.
- Object Adapter:
  - ↪ Adapts a non-native Corba object to Corba (object model adaptation).
  - ↪ Calls methods.
  - ↪ Activates and deactivates an object implementation.

CORBA Advantages

- Vendor independent Client/Server Computing in heterogeneous environments.
- Transparent method calls of any local and remote CORBA object.
- Static method calls with strong type checking at translation time.
- Ability to dynamically issue method calls of unknown CORBA objects at runtime.
- Objects are defined independently of the programming language using (IDL).
- There exist IDL compilers for C, C++, SmallTalk, Java, Ada,...
- Applications can call object methods regardless of the programming language used for their implementation.
- An interface repository contains Meta-information about well-known objects.
- Corba interoperability guaranteed (in theory) thanks to the Inter-ORB-Protocol (IIOP).
- The ORB guarantees location transparency and operating system independence.
- Many basic CORBA services are implemented as CORBA objects (Naming Service, Trader, Location Service, Event Service, ...).

Interface Definition Language

- Basic Datatypes:

  `short, long, long long, unsigned short, unsigned long, unsigned long long, float, double, octet, char, boolean, enum, any`

- Additional Datatypes:

  `structure, sequence, union, string, array`

- Interface Definitions:
  - Declaration of object relations (inheritance).
  - Declaration of datatypes.
  - Declaration of constants.
  - Declaration of attributes.
  - Declaration of operations.
  - Declaration of object exceptions.
- Module Definitions:
  - Declaration of Datatypes, constanten and exceptions.
  - Declaration of interfaces.
  - Declaration of additional modules.

CORBA Services and Common Facilities

- CORBA Services define functions which are often used:
  - Naming Service
  - Event Service
  - Life Cycle Service
  - Persistence Service
  - Object Relationship Service
  - Object Externalization Service
  - Object Transaction Service
- Additional functional modules, in particular for special areas of application, are developed within OMG and standardised as Common Facilities.

➢ Some the CORBA services are very important event outside the CORBA world for the functionality they provide. In particular the Event service is important for management.

Internet Inter-ORB Protocol (IIOP)

- Common Data Representation (CDR)
  - Defines the bytecode representation of CORBA basic types (similar to ASN.1 BER).
  - The sender selects the byte ordering (little endian vs. big endian).
  - Performs the alignment of data to „natural“ (e.g. 32, 64 bits) boundaries (data alignment).

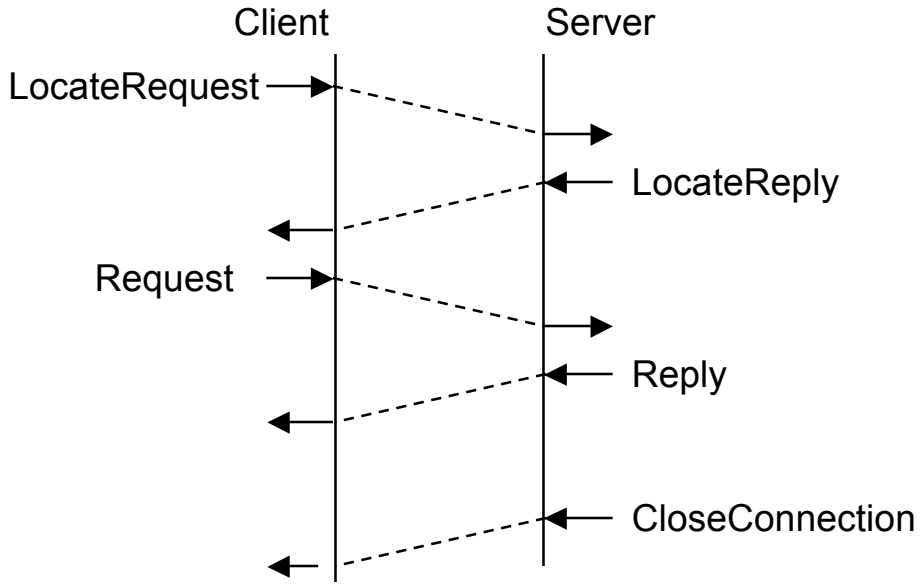
- General Inter-ORB Protocol (GIOP)
  - Definition of the message types and their formats:

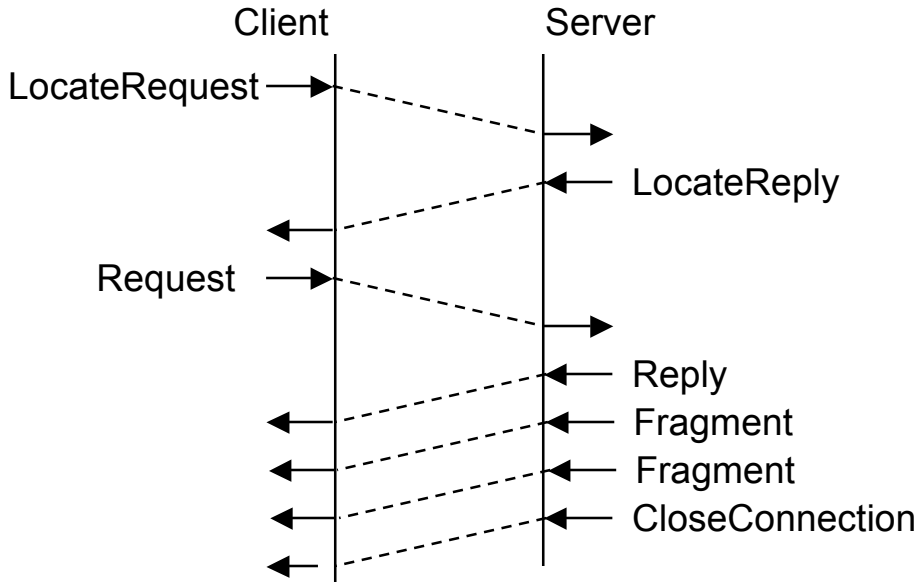
Request	Reply
CancelRequest	CloseConnection
LocateRequest	LocateReply
MessageError	Fragment

  - Definition of the message representation in CDR.
  - Independent of the underlying transport protocol.
  - Large messages can be fragmented.

- Internet Inter-ORB Protocol (IIOP)
  - It defines the transfer of GIOP over TCP/IP.

Example of GIOP Message Calls



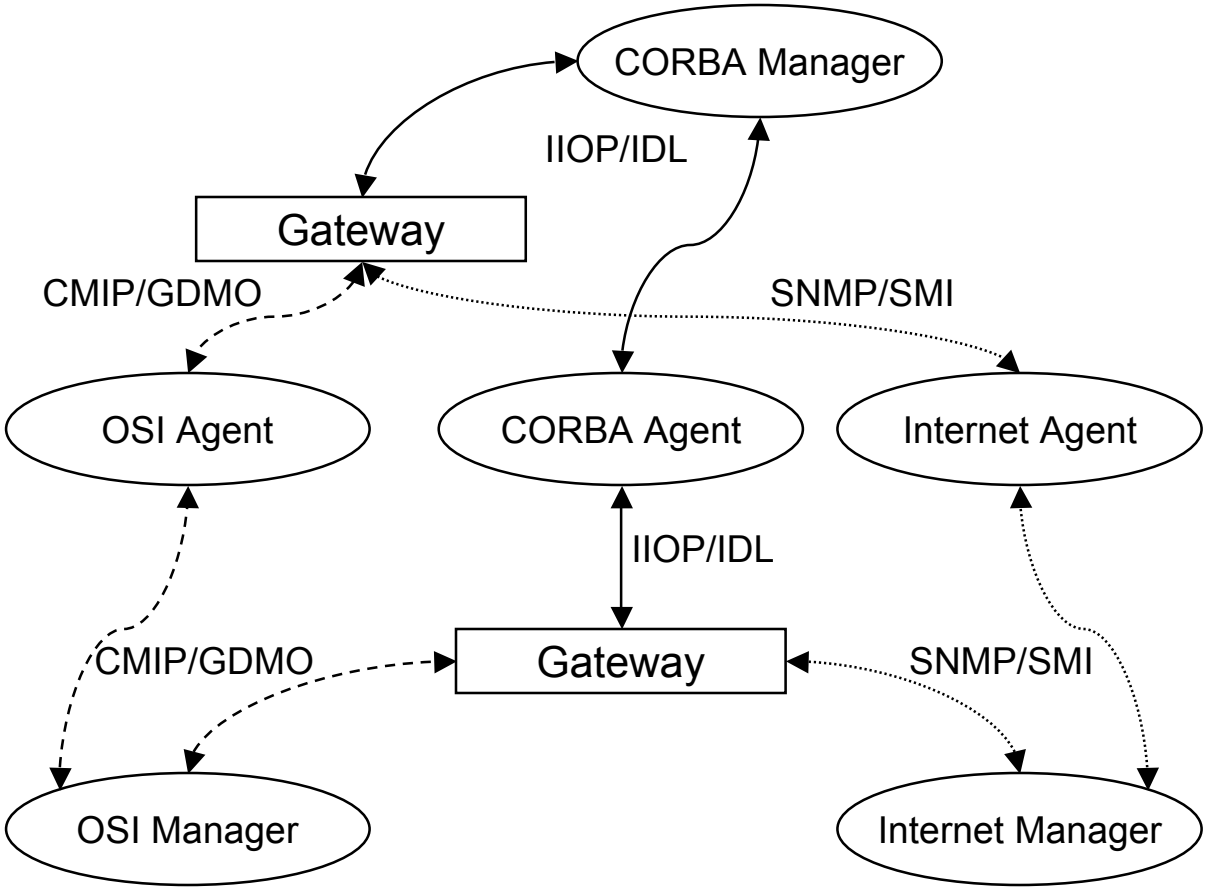


- A LocateRequest verifies the existence of an object and it is sent to the ORB where the CORBA Object is known to exist.
- A MessageError is used to report errors.
- A CancelRequest is sent by a client to abort the transfer of reply fragments.
- From the communication point of view, CORBA is rather similar to CMIP.

Joint Inter-Domain Management (XoJIDM)

- Sponsored by X/Open and the Network Management Forum (NMF).
- Introduction of an object-oriented development environment for the network management.
- Simple, specifications which can be easily read.
- Creation of reusable class libraries based on GDMO object classes:
  - ↪ Reuse of the MOs already standardised for the telecommunications world.
  - ↪ Use of CORBA for language independence .
- Definition of a class library for SNMP SMI definitions:
  - ↪ Integration of extensive and widespread Internet SMI MIB definitions
  - ↪ Use of CORBA for language independence.
- Reduce development complexity for the implementation of management applications based on GDMO or SMI MOs for the implementation of management applications.
- Portability and interoperability of management applications beyond platform boundaries.

Interoperability between Management Architectures



- By means of gateways it is possible to obtain interoperability between existing management systems and agents.

IDL Datatypes for SMIv2-Datatypes

- Mapping of relevant ASN.1 Datatypes on IDL Datatypes:
  - NULL — typedef char ASN1_Null;
  - INTEGER — typedef long ASN1_Integer;
    typedef long long ASN1_Integer64;
    typedef unsigned long ASN1_Unsigned;
    typedef unsigned long long ASN1_Unsigned64;
  - OCTET STRING — typedef sequence<octet> ASN1_OctetString;
  - OBJECT IDENTIFIER — typedef string ASN1_ObjectIdentifier;

- Mapping of SMI-Datatypes on IDL-Datatypes:
  - Integer32 — typedef ASN1_Integer SNMPv2_SMI::Integer32Type;
  - Unsigned32 — typedef ASN1_Unsigned SNMPv2_SMI::Unsigned32Type;
  - Gauge32 — typedef ASN1_Unsigned SNMPv2_SMI::Gauge32Type;
  - Counter32 — typedef ASN1_Unsigned SNMPv2_SMI::Counter32Type;
  - TimeTicks — typedef ASN1_Unsigned SNMPv2_SMI::TimeTicksType;
  - IpAddress — typedef sequence<octet,4> SNMPv2_SMI::IpAddressType;
  - Opaque — typedef ASN1_OctetString SNMPv2_SMI::OpaqueType;
  - Counter64 — typedef ASN1_Unsigned64 SNMPv2_SMI::Counter64Type;
  - BITS — typedef ASN1_OctetString SNMPv2_SMI::BitsType;

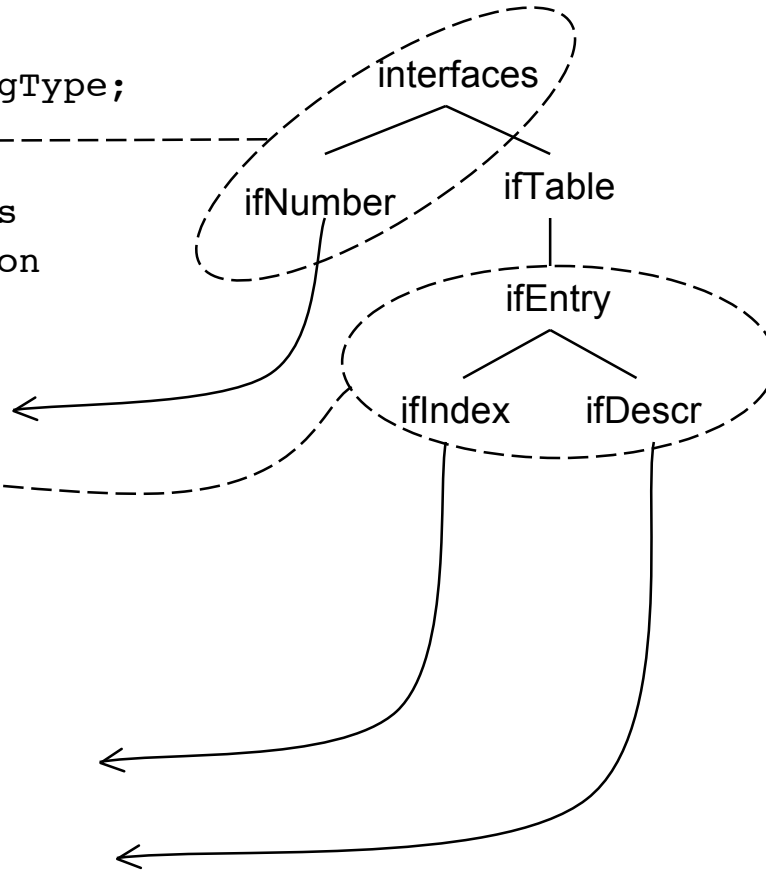
Example of SMIv2 -> IDL Translation [1/2]

```
module IF-MIB {

  typedef SNMPv2_SMII::Integer32Type Integer32Type;
  typedef SNMPv2_TC::DisplayStringType DisplayStringType;

  interface interfaces : SNMPMgmt::SmiEntry {
    /* DESCRIPTION: The number of network interfaces
       (regardless of their current state) present on
       this system. */
    readonly attribute Integer32Type ifNumber;
  };

  interface ifEntry : SNMPMgmt::SmiEntry {
    /* INDEX : (ifIndex) */
    const string EntryIndexVarLis = „ifIndex“;
    /* DESCRIPTION: ... */
    readonly attribute InterfacadexType ifIndex;
    /* DESCRIPTION: ... */
    readonly attribute DisplayStringType ifDescr;
    /* ... */
  }
}
```



Example of SMIv2 -> IDL Translation [2/2]

```
struct IfIndexType {
  ASN1_ObjectIdentifier var_name;
  ASN1_ObjectIdentifier var_index;
  InterfaceIndexType var_value;
}

struct LinkDownType {
  IfIndexType ifIndex;
  IfAdminStatusType ifAdminStatus;
  IfOperStatusType ifOperStatus;
}

interface Notifications : SNMPv2::Notifications {
  void linkDown(in string agent_ip_address, in string community_name,
         in SNMPv2TC::TimeStampType event_time, in LinkDownType notification_info);
}

interface PullNotifications : SNMPv2::PullNotifications {
  void pull_linkDown(out CosNaming::Name src_entry_name,
         out SNMPv2TC::TimeStampType event_time, out LinkDownType notification_info);
  boolean try_linkDown(out CosNaming::Name src_entry_name,
               out SNMPv2TC::TimeStampType event_time, out LinkDownType notification_info);
}
```

```
struct IfAdminStatusType {
  ASN1_ObjectIdentifier var_name;
  ASN1_ObjectIdentifier var_index;
  IF-MIB::ifEntry::IfAdminStatusType var_value;
}

struct IfOperStatusType {
  ASN1_ObjectIdentifier var_name;
  ASN1_ObjectIdentifier var_index;
  IF-MIB::ifEntry::IfOperStatusType var_value;
}
```

XML-based Management vs. Other Approaches

- SNMP
  - Internet standard management protocol widely used for network monitoring and fault management but not so widely used for configuration
  - Many vendors don't support their full configuration (or even monitoring) capability via SNMP
  - Security problems addressed with SNMPv3

- Expect [http://expect.nist.gov/]
  - Allows programmatic interaction with device command line interfaces
  - Widely used as a practical network management tool
    - Used to acquire data from CLI not exposed via SNMP MIBs
    - ASCII format is appealing for simplicity and debugging
  - Error prone, especially when vendors upgrade their CLI

- Corba
  - Network object access protocol
  - Versioning and backwards compatibility issues
  - Vendor incompatibility between ORBs
  - Large footprint a tight fit on embedded systems

XML-based Management: Introduction to XML [1/2]

- eXtensible Markup Language: http://www.w3c.org/xml
- XML is a generally self-describing data format
  - Application reads data, parses it, and knows exactly what each constituent part of the data means
- An XML document is a “text file with structure” easy to understand, parse and debug.
- Six main constructs
  - Open tags: `<tag>`
  - Close tags: `</tag>`
  - Data: `<tag>data</tag>`
  - Empty tags: `<tag/>`
  - Attributes: `<tag foo=“bar” go=“gar”/>`
  - Namespaces:

```
<ns1:home>
    <address>123 Main Street</address>
    <network xmlns:ns2=“my.identifying.string”>
       <ns2:address>10.0.0.1</ns2.address>
    </network>
</ns1.home>
```

XML-based Management: Introduction to XML [2/2]

- XML data definition tools
  - ↪ Document Type Definitions (DTDs)

    Lists the elements that may appear in an XML document and their relationships to one another
  - ↪ XML Schemas

    Defines content and semantics in addition to element relationships
- XSLT
  - ↪ Language for transforming XML documents
  - ↪ XML -> XML
  - ↪ XML -> (XHTML, XSL-FO, DocBook) -> formatted output

XML-based Management: JUNOScript [1/4]

- JUNOScript is an XML-based API to JUNOS [http://www.juniper.net/techpubs/software.html]
- Uses an RPC paradigm
  - Client sends requests enclosed in <rpc> tags

```
<rpc>
    <get-interface-information>
        <name>so-1/3/0</name>
    </get-interface-information>
</rpc>
```

  - Server response is enclosed in <rpc-reply> tags

```
<rpc-reply>
    <interface-information>
        <name>so-1/3/0</name>
        <admin-status>up</admin-status>
        …
    </interface-information>
</rpc-reply>
```

XML-based Management: JUNOScript [2/4]

- If reply is affirmative, an empty `<rpc-reply></rpc-reply>` tag pair is returned
  - ↪ Errors are indicated by the `<junos:error>` tag
- Any attributes included on the `<rpc>` tag are echoed on the `<rpc-reply>`
  - ↪ Useful for client bookkeeping
- The top-level tag in the reply includes an attribute defining the XML namespace for the returned data
- Only 1 request can be outstanding at a time
- Server returns errors as <junos:error> tags

```
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/junos.dtd">
    <junos:error>
        <message>configuration database modified</message>
    </junos:error>
</rpc-reply>
```

- Client can send `<junos:abort/>` to abort server's request processing
  - ↪ Server responds with `<junos:abort-acknowledgement/>`

XML-based Management: JUNOScript [3/4]

```
<rpc>
  <get-chassis-inventory/>
  <!-- Same as CLI's 'show chassis hardware'→
</rpc>
```

```
<rpc-reply>
  <chassis-inventory xmlns="…/junos-
    chassis.dtd">
   <chassis junos:style="inventory">
    <name>Chassis</name>
    <serial-number>20388</serial-number>
    <description>M20</description>
    <chassis-module>
    <name>Backplane</name>
    <version>REV 07</version>
    <part-number>710-001517</part-number>
    <serial-number>AB6192</serial-number>
    </chassis-module>
    <chassis-module>
    <name>Power supply A</name>
    <version>Rev 02</version>
    <part-number>740-001465</part-number>
    <serial-number>000476</serial-number>
    <description>AC</description>
    </chassis-module>
    <chassis-module>
    <name>Power supply B</name>
    <version>Rev 02</version>
    <part-number>740-001465</part-number>
    <serial-number>000506</serial-number>
    <description>AC</description>
    </chassis-module>
```

```
    <chassis-module>
      <name>Display</name>
    <version>REV-04</version>
    <part-number>710-001519</part-number>
    <serial-number>AC2803</serial-number>
  </chassis-module>
  <chassis-module>
    <name>Host 0</name>
    <serial-
     number>0e000006175d8a01</serial-
     number>
<description>Present</description>
</chassis-module>
<chassis-module>
<name>Host 1</name>
<serial-number>af0000062ae60201</serial-
     number>
<description>Present</description>
</chassis-module>
<chassis-module>
<name>SSB slot 0</name>
<version>REV 05</version>
<part-number>710-001411</part-number>
<serial-number>AE2680</serial-number>
<description>Internet Processor
     I</description>
</chassis-module>
…
</rpc-reply>
```

XML-based Management: JUNOScript [4/4]

- Simplifies application development
  - Standard, commonly used language
  - Widely available tools and information
  - Easy to understand text format
  - Larger talent pool of engineers
- Offers a reliable alternative to Expect scripts
  - XML's self-describing data prevents problems with variations in CLI output
- Enhances Interoperability
  - XML is a standard method of exchanging information between programs
  - Adopted by many industries – eCommerce, databases, networking, etc.

XML-based Management: WebServices

- A *web service* is a software system identified by a URL, whose public interfaces and bindings are defined and described using XML. [Official W3C Definition]

- Definitions:
  - **XML** (eXtensible Markup Language)

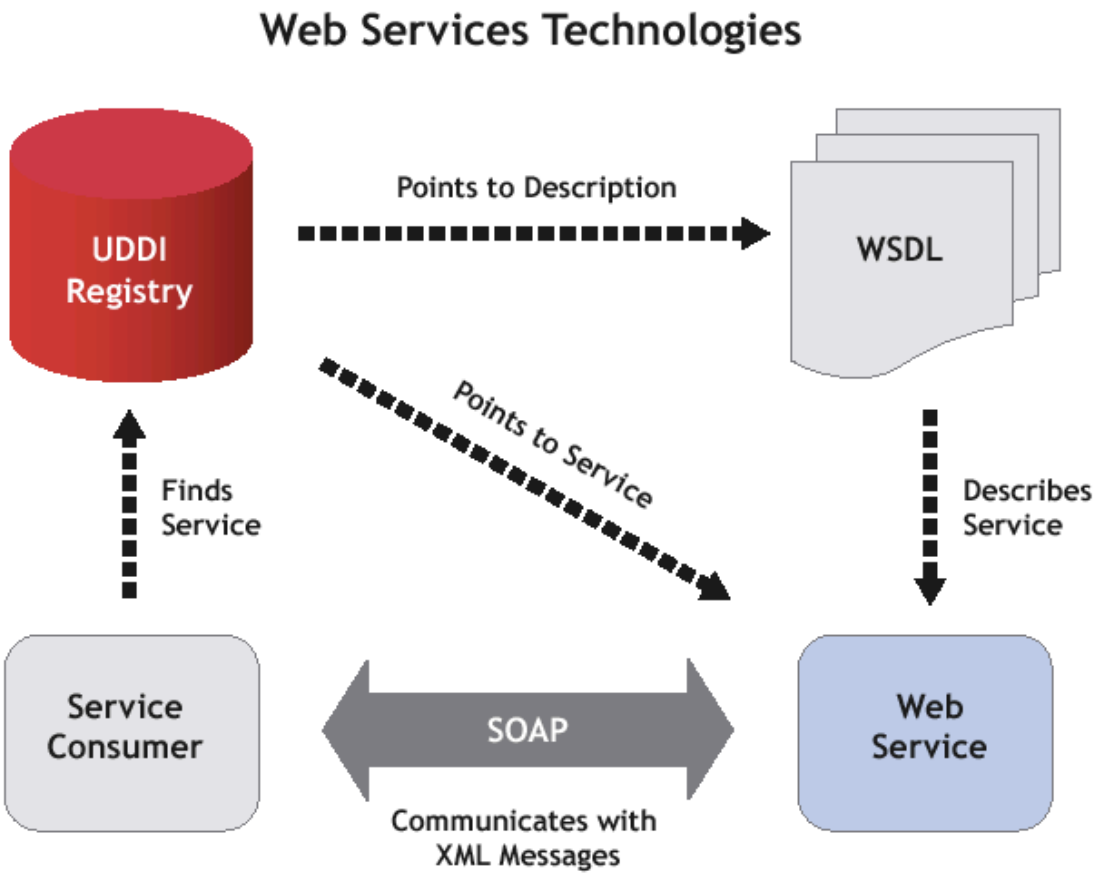
    Markup language that underlies most of the specifications used for Web services.
  - **SOAP** (Simple Object Access Protocol)

    A network, transport, and programming language-neutral protocol that allows a client to call a remote service. The message formatis XML.
  - **WSDL** (Web services description language)

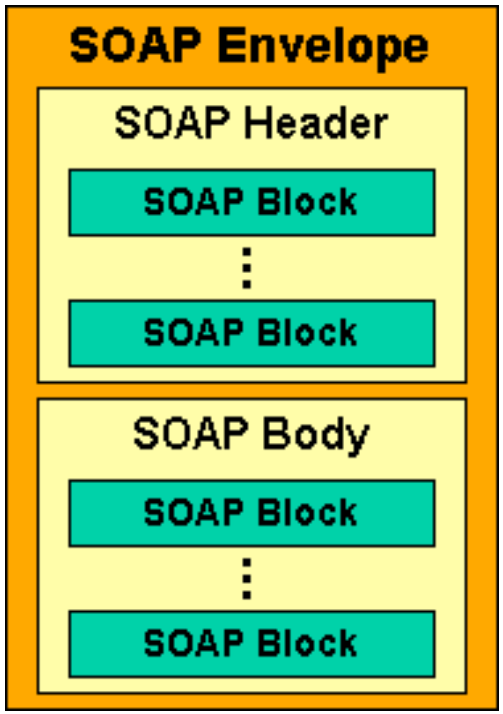
    An XML-based interface and implementation description language. The service provider uses a WSDL document in order to specify the operations a Web service provides.
  - **UDDI** (universal description, discovery, and integration)

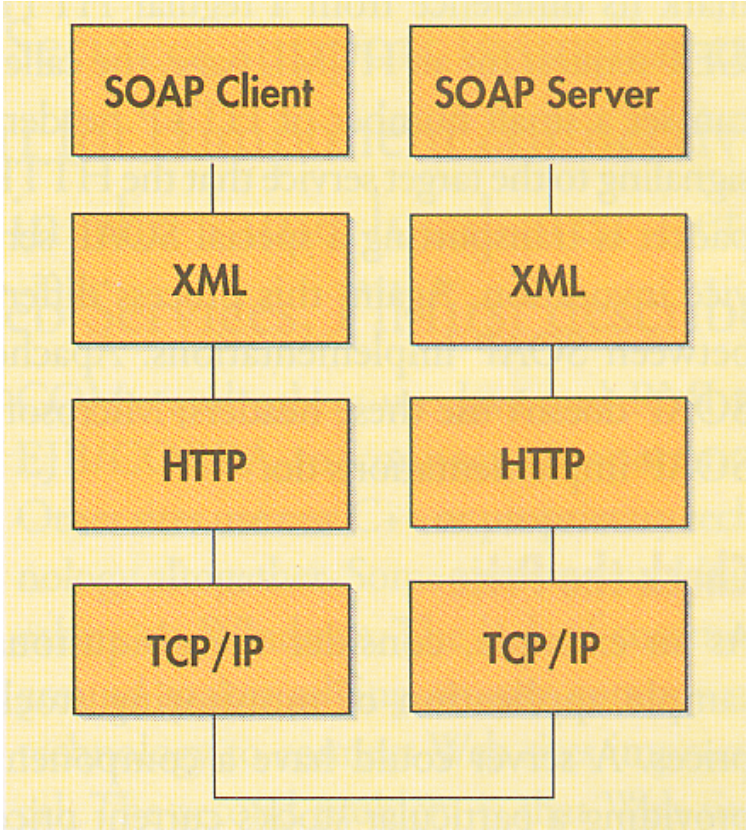
    Both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information on service providers and Web services.

WebServices: Architecture



WebServices: SOAP





WebServices and Network Management

Why the management world is becoming interested in web services?

- SNMP is somehow complicated for common users that instead know/use web technologies such as HTTP and XML.
- The web services technologies are already used in other fields so they are basically for free (no software to install nor encoders/decoders).
- Many available applications are web services aware. For instance it is possible to use Microsoft Excel to grab monitoring data in a few Visual Basic lines of code (.NET is required).

How is the management world using web services?

- Create gateways (e.g. from SNMP) to web services.
- Reuse existing MIB experience for modeling new systems using past experience.
- Develop simple SOAP-based services to be embedded in appliances so that they can be managed using web services instead of SNMP.