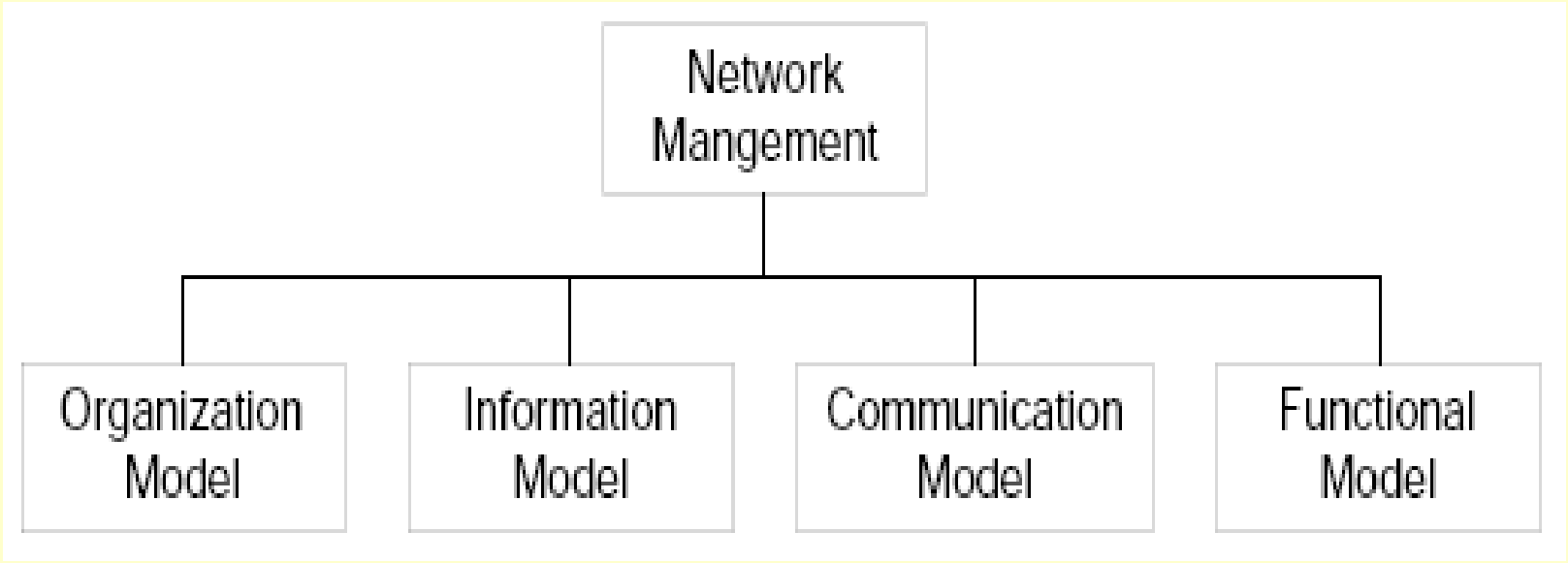
TELE9752 Network Operations and Control

Lecture 3: Management Information

Course outline



- Organization model (OIDs) _ this week
- Information model (SMI) /
- Communication model (SNMP) next week
- Functional model (FCAPS) – forthcoming weeks

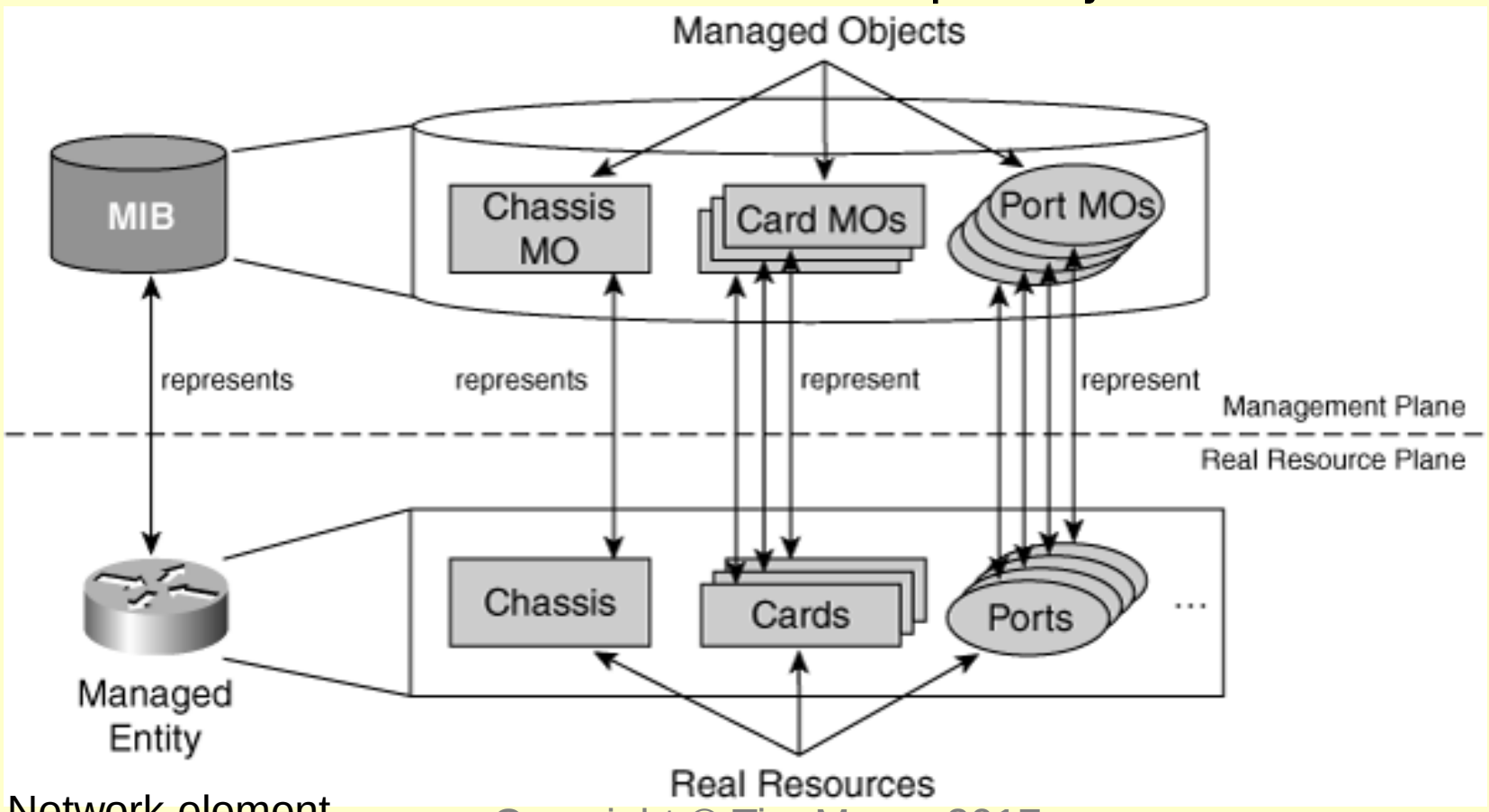
RFC 3444 suggests a distinction between "Information Models" and more concrete "Data Models" (presumably between Information Model & Communication Model) and that SMI is a "Data Model".

2C1 Fig. 3.1 from M. Subramanian: *Network Management: Principles and Practice*, Addison Wesley,

Objects vs resources

Objects represent aspects of network elements

Managed Objects (MOs) give an abstracted view of real resources in network elements. Can have multiple objects/resource.



Network element

Fig. 6.2 from Clemm

e.g. IPv6 router ad objects

How does a host discover which router to use? (IPv4: DHCP)

IPv6: Receive a router advertisement (sent over ICMP).

Ads are sent periodically or in response to router solicitation.

A table of objects holds config. info (1 row per interface)

ipv6RouterAdvertTable OBJECT-TYPE [5C>[K0>[U4>[QL>
SYNTAX SEQUENCE OF Ipv6RouterAdvertEntry
::= { ip 39 }

ipv6RouterAdvertEntry OBJECT-TYPE **[MW>**[K0>[U4>[QL>
SYNTAX Ipv6RouterAdvertEntry
::= { ipv6RouterAdvertTable 1 }

ipv6RouterAdvertSpinLock OBJECT-TYPE [5C>[YN>
SYNTAX TestAndIncr
::= { ip 38 }

Future slides that use this example

Definitions from RFC 4293, heavily edited to reduce clutter

Content of router ad table row/entry

```
Ipv6RouterAdvertEntry ::= SEQUENCE {                    [K0>[QL>
  ipv6RouterAdvertIfIndex          InterfaceIndex,  [9W>
  ipv6RouterAdvertSendAdverts       TruthValue,
  ipv6RouterAdvertMaxInterval       Unsigned32, [9z>[0A>[K0>[90>[L1>
  ipv6RouterAdvertMinInterval       Unsigned32,
  ipv6RouterAdvertManagedFlag       TruthValue,
  ipv6RouterAdvertOtherConfigFlag   TruthValue,
  ipv6RouterAdvertLinkMTU           Unsigned32,
  ipv6RouterAdvertReachableTime     Unsigned32,
  ipv6RouterAdvertRetransmitTime    Unsigned32,
  ipv6RouterAdvertCurHopLimit       Unsigned32,
  ipv6RouterAdvertDefaultLifetime   Unsigned32,
  ipv6RouterAdvertRowStatus         RowStatus    [5C>
}
```

Black: carried in ads. Red: Manage ads.

Sample SMI definition of an object

```
ipv6RouterAdvertMaxInterval OBJECT-TYPE
    SYNTAX      Unsigned32 (4..1800)
    UNITS       "seconds"
    MAX-ACCESS read-create
    STATUS      current
    DESCRIPTION "The maximum time allowed
      between sending unsolicited router
      advertisements from this interface."
    REFERENCE "RFC 2461 Section 6.2.1"
    DEFVAL { 600 }
    ::= { ipv6RouterAdvertEntry 3 }
```

(2) type (ASN.1)

(3)

(3)

documentation

default value

(1) position in OID hierarchy

Topics to be covered: (1) OIDs, (2) ASN.1 basics, (3) SMI conventions [4] MIBs [PK>

Outline

- Objects
  - Types
  - Identifiers
- Structure of Management Information
  - ASN.1 basics
  - RFC conventions
- MIBs

Programming analogy:

SMI: type definitions

MIB: define variables in terms of types

BER (next week) – like compiler (though separate analogy to SMI/MIB)

Types of objects

Objects may represent data (this slide) or behaviour (next slide)

Data objects: (See MIB discussion [PK> for concrete examples)

- Configuration
  - Static info, e.g. make/model, serial #, link speed
  - Programmed info, e.g. IP address

  ↕ these are differentiated by NETCONF/YANG [WG>
- Protocol state
  - e.g. link state learned through routing protocol, TCP connection table
- Statistics
  - e.g. current buffer/CPU utilisation, count of packets sent/received, errors observed

Objects that represent actions

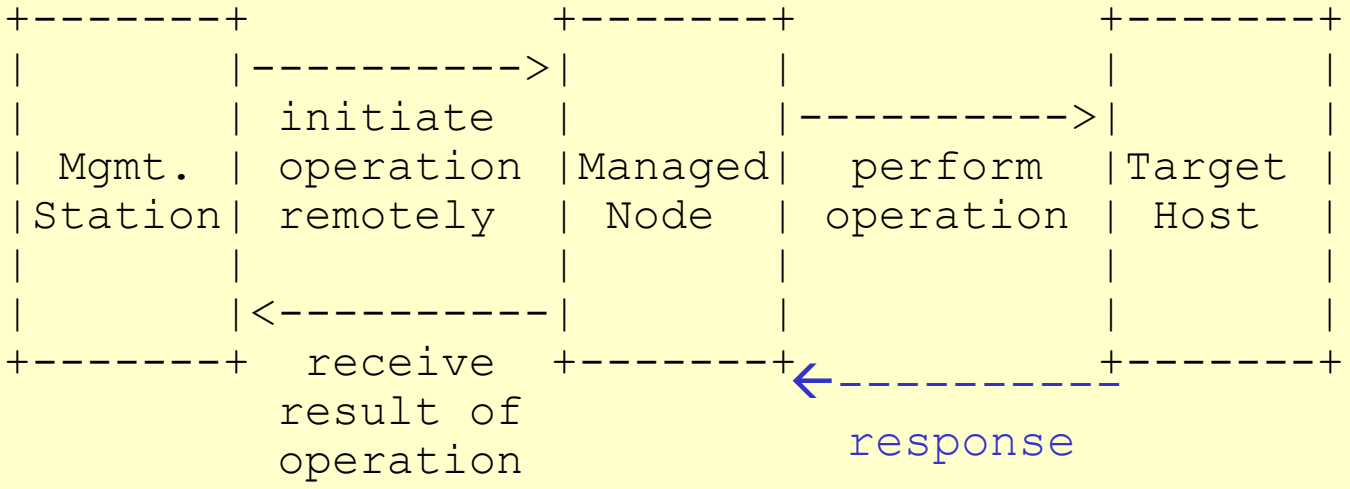
NM involves **commanding elements** to act, e.g.

- reboot
- block concurrent access by another manager
- routing table entries: create new or delete (not just set data to another value)

- Problem: Actions vary with elements, and are likely to proliferate with new elements => potentially massive # of commands to complicate a management protocol.
- "Solution": Create objects that represent actions. NM writes to object to command element to perform action.
  - e.g. `ipv6RouterAdvertSpinLock` [YN> controls access to `ipv6RouterAdvertTable` for which rows include `ipv6RouterAdvertRowStatus` [66> that allows deletion
  - e.g. 2: ping MIB [JD>

ping MIB

RFC 4560 defines a ping MIB that allows NM to request a managed device to ping a 3rd party



(blue text not in RFC 4560)

```
PingCtlEntry ::=
   SEQUENCE {
      pingCtlOwnerIndex              SnmpAdminString,
      pingCtlTestName                SnmpAdminString,
      pingCtlTargetAddressType       InetAddressType,
      pingCtlTargetAddress           InetAddress,
...
      pingCtlAdminStatus             INTEGER,  set to 1 to start
...
```


2JD

Object Identifiers

- Need to precisely identify which object we seek to access
- Use a hierarchical tree of identifiers, e.g. `iso.org.dod.1.mgmt.1.ip.39.1.ipv6RouterAdvertMaxInterval`
- Each joint/leaf identified by context (position in hierarchy) + either
  - Number – intended for machine processing
  - String – chars &#s - Intended for humans

  Strings identifying Internet objects tend to be flat & unique (e.g. `ipv6RouterAdvertMaxInterval`, not `ipv6.RouterAdvert.MaxInterval`) unlike hierarchical *numeric identifiers* (`ip.39(ipv6RouterAdvertTable).1(Entry).3(MaxInterval)`)

1.2.840 (USA).113556 (Microsoft).4 (file formats).2 - Microsoft Word
1.2.36 (Australia). AustralianCompanyNumber (e.g. UNSW = 124 669 736)

OID tree

- Root
  - 0 itu
  - 1 iso
    - 2 members
    - 3 org
      - 6 dod
        - 1 internet
          - 1 directory
          - 2 mgmt
            - 1 mib-2
              - 1 system
              - 2 interfaces
              - 3 at
              - 4 ip
                - 39 ipv6RouterAdvertTable
                  - 1 ipv6RouterAdvertEntry
                    - 3 ipv6RouterAdvertMaxInterval
          - 3 experimental
          - 4 private
            - 1 enterprises
              - 9 cisco
              - 11 hp
  - 2 iso-itu

iso.org.dod.internet.mgmt.mib-2.ip.ipv6RouterAdvertTable.ipv6RouterAdvertEntry.ipv6RouterAdvertMaxInterval
or
1.3.6.1.2.1.ip.ipv6RouterAdvertTable.ipv6RouterAdvertEntry.3

See http://www.iana.org/assignments/enterprise-numbers

Object Identifiers (ctd)

- Hierarchy can extend in depth (# levels) and width (# branches per node).
  - In theory, unlimited
  - In IETF practice: Limited: ?255 levels & 2^{32}-1 branches per node?

- Object *types* are defined universally, but object *values* are *defined in context of element* being managed.
  - e.g. can get/set `ipv6RouterAdvertMaxInterval` for a particular router (indeed, particular entry of table within router), but OID doesn't identify which router

Scalar and aggregate objects

- Only leaf nodes have values; interior nodes exist only to create order/hierarchy
- Leaf nodes may be
  - Scalar: single value† per Managed Element
    - value of scalar referred to by OID.0, e.g. `udpInDatagrams.0`
  - Aggregate: Potentially multiple values per Element, e.g. list of interfaces, rows of routing table
    - aka “columnar object” since often represent columns of tables
    - value of item in aggregate referred to by OID.index
    - more when we discuss tables [QL>

† In computing, strings are considered to be scalars, http://en.wikipedia.org/wiki/Scalar_(computing)

3rd dimension to OIDs

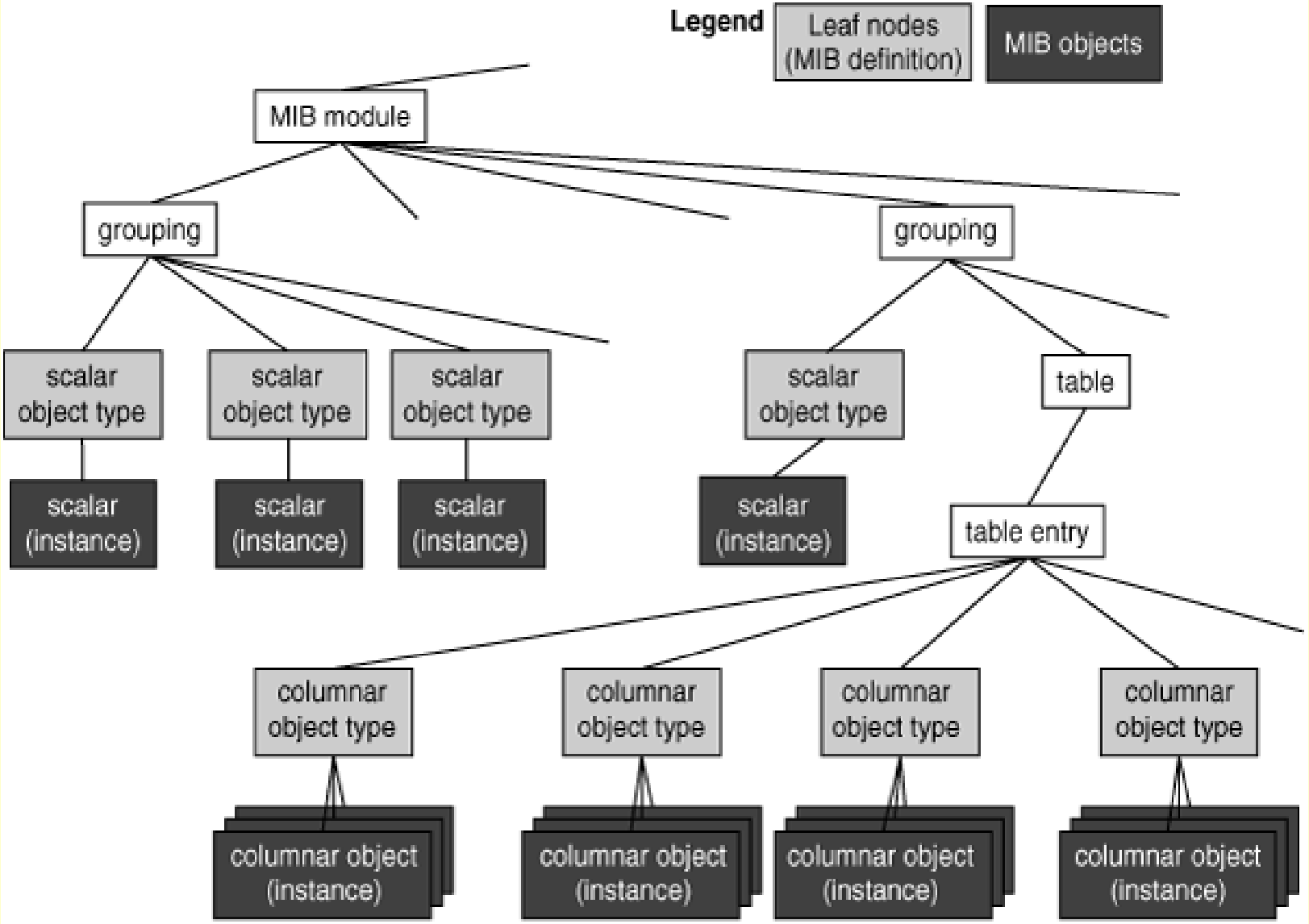


Fig. 6.14 from Clemm

Outline

Object Identifiers

Structure of Management Information

- **ASN.1 basics**
- RFC conventions

MIBs

Abstract Syntax Notation 1

ASN.1 is a language for defining data types.

- types of managed objects
- types of protocol format!, e.g. SNMP and OSI layer 4+

“Abstract” in the sense that Encoding Rules are defined separately.
Analogous to programming language types + compiler.

Not the most enjoyable topic!
“Like vegetables, knowledge about ASN.1 ... is something that is “good for you”” - Kurose & Ross, 4th ed p. 781
Thankfully there is no ASN.2!

Defined in ITU Recommendation X.680-683 (originally X.208)
Tutorial: B. Kaliski: “A Layman's Guide to a Subset of ASN.1, BER, and DER”, RSA Laboratories Technical Note

ASN can also refer, in context of BGP, to Autonomous System Number

Why ASN.1? Outsider's view

“"The official reason for using ASN.1 is to ease the eventual transition to OSI-based network management protocols. The actual reason is that the Internet research community got caught napping on this one, having never spent much time dealing with application-layer structuring.”
- M. Rose: *The Simple Book, 1994*

Why ASN.1?: Insider's view

"In retrospect, the use of ASN. 1 in Internet management was a good choice based on political criteria,

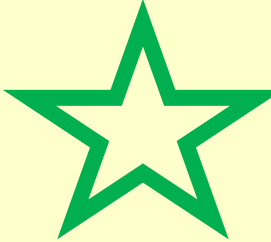
> "without this choice, it is unlikely that … SNMP would have had sufficient political acceptability to obtain high levels of acceptance"

even if it was a poor choice based on technical criteria."

> "First, ASN.1 is unnecessarily complex, leading to unnecessarily large code sizes and slower execution. ASN.1 packs data into the smallest possible number of bytes on the network. As a result, data is neither word- nor longword-aligned, resulting in poorer performance and the trading of CPU cycles for media bandwidth."

- J. Case: "Network Management" in D. Lynch and M. Rose: *Internet System Handbook*, Addison-Wesley, 1993

Actually, it isn't ASN.1 that "packs data" but rather the encoding rules that are used with it. See SNMP lecture.



ASN.1 foundations

Comments follow “--”on line, e.g.

```
-- Internet Address Table
```

assignment operator: ::=

```
ipAddressTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF IpAddressEntry
...
    ::= { ip 34 }
```

blocks defined by BEGIN/END, IMPORT types from other modules

```
IP-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Integer32, ... FROM SNMPv2-SMI
...
END
```

ASN.1 naming conventions

Identifiers: can contain chars, digits and hyphens

all-UPPERCASE for

- primitive types, e.g. INTEGER
- MIB modules, e.g. IP-MIB
- macros, e.g. MODULE-COMPLIANCE

initial uppercase for application-defined types and modules, e.g. `InetAddress`

initial *lower*case for object names, e.g. ipv6RouterAdvertRowStatus

Types of type

Simple: atomic types, e.g. `INTEGER`
RFC1155 calls these “primitive” types

Structured: Contains other types as components, e.g. `SEQUENCE` of `INTEGER`s (or even SEQUENCE OF SEQUENCES)

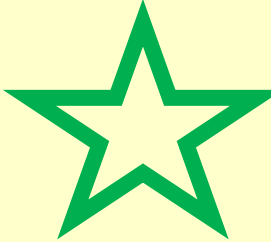
Miscelaneous: `CHOICE`† (union), `ANY` (arbitrary - defined elsewhere in system)

Another type of type is “Tagged”: derived from another type but assigned a different tag (type code – see slide [UV>)

- Note that “Every ASN.1 type other than CHOICE and ANY has a tag”

2UZ

† Used in the `NetworkAddress type [QC>`



Simple types for IETF NM

`INTEGER`

arbitrarily large (Encoding Rules handle magnitude)

`OCTET STRING`

string of bytes, e.g. for text or binary data
(`DisplayString` is a newer type for text)

`OBJECT IDENTIFIER`

as previously discussed <0A]

`NULL`

e.g. used for error value field when no error occurs

`SEQUENCE` / `SEQUENCE OF`

Often used to define tables

Subtypes

Subtyping further constrains existing types, e.g.

- Enumerated integers:

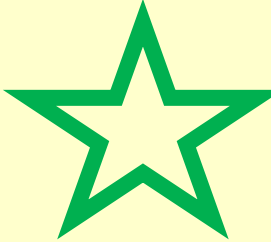
```
IpAddressOriginTC ...
SYNTAX      INTEGER {
              other(1), manual(2), dhcp(4),
              linklayer(5), random(6)
            }
```

- Integers with constrained values:

```
ipv6RouterAdvertMaxInterval …
SYNTAX      Unsigned32 (4..1800)
```

- Strings of restricted length

```
OCTET STRING (SIZE 0..255)
OCTET STRING (SIZE 4 | 8)
```



Defined types

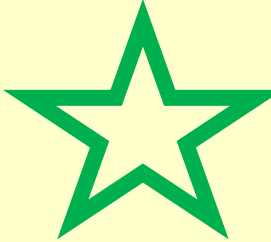
- Counters, e.g. `Counter32`
  - non-negative
  - *monotonically* increasing
  - wrap around
  - e.g. for packet counts. For fast links, a Counter64 is also needed to record a High Count (HC) <D7] which won't wrap around within 1 hour (between reads)
- Gauges, e.g. `Gauge32`
  - non-negative
  - increase *or decrease*
  - capped at extremes, e.g. $++2^{32}-1 = 2^{32}-1$

    not stuck/latched there [RFC2578], e.g. $--2^{32}-1 = 2^{32}-2$
    - c.f. Stevens TCP/IP Illustrated vol. 1 p. 363

more Defined types

- `TimeTicks`
  - non-negative integer representing time, in 1/100 sec, since an “epoch” (generally system reboot)
  - Measuring time relative to system reboot leaves task of synchronising times to NM
- IP addresses
  - `IpAddress`
    - 4B IPv4 address in network byte order.
  - `InetAddress, InetAddressIPv4, InetAddressIPv6` etc [RFC4001]
    - binary encodings of IP addresses

PhysAddress etc



Sequences

- SEQUENCE defines a structure; often used to define format of a row (i.e. define columns) of a table

```
Ipv6RouterAdvertEntry ::= SEQUENCE {
  ipv6RouterAdvertIfIndex   InterfaceIndex,
  ipv6RouterAdvertSendAdverts TruthValue,
...
}
```

- A "SEQUENCE OF" is a list of instances of a type. Often used to define a table as a SEQUENCE OF rows.

```
ipv6RouterAdvertTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ipv6RouterAdvertEntry
    INDEX { ipv6RouterAdvertIfIndex }
    ::= { ip 39 }
```


If you're familiar with the language Python: SEQUENCE=Python tuple, SEQUENCE OF = Python list

Outline

Object Identifiers

Structure of Management Information

- ASN.1 basics
- **RFC conventions**

MIBs

Structure of Management Information

- SMI defines how RFCs use ASN.1
- Several versions:
  - SMI [STD 16, RFC 1155 & 1212]
  - **SMIv2 [STD 58, RFCs 2578-80]** <- our focus
  - SMIng discussion, e.g. RFC3780
- Alternatives to SMI include:
  - Managed Object Format (MOF) from DMTF
    - IT systems
  - Guidelines for the Definition of Managed Objects (GDMO) for CMIP
    - ISO standard
  - XML Schema Definitions
    - increasingly important, e.g. used by DSL Forum and Netconf

We consider individual objects first, then table rows of multiple objects [QL> and groups of objects [JR>

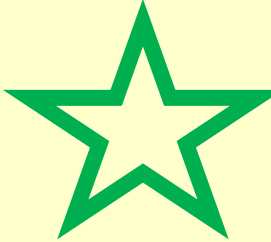
Macro for defining OBJECT-TYPE

```
OBJECT-TYPE MACRO ::=
BEGIN
   TYPE NOTATION ::=
            "SYNTAX" Syntax
            UnitsPart
            "MAX-ACCESS" Access
            "STATUS" Status
            "DESCRIPTION" Text
            ReferPart
            IndexPart
            DefValPart
```

```
ipv6RouterAdvertMaxInter
    SYNTAX     Unsigned3
    UNITS      "seconds"
    MAX-ACCESS read-crea
    STATUS     current
    DESCRIPTION "The max
      between sending un
      advertisements fro
    REFERENCE "RFC 2461
    DEFVAL { 600 }
    ::= { ipv6RouterAdve
```

...

from RFC2578



SMI clauses

SYNTAX: type of object
UnitsPart: Elaborates on type
- e.g. seconds, milliseconds

DESCRIPTION
- documents the object, e.g. why is status deprecated?

ReferPart
- text reference to sections of documents relevant to this object

DefValPart
- default value

On following slides:
- MAX-ACCESS [Y5>
- STATUS [FT>
- IndexPart [VD>

Access allowed for an object

- “defines whether it makes "protocol sense" to read, write and/or create an instance of the object, or to include its value in a notification. This is the maximal level of access for the object.” [RFC 2578]
- `MAX-ACCESS` clause uses the `Access` type:

```
Access ::=
                "not-accessible"    can't get/set value, but
                                    can use name, e.g. for auxiliary
                                    objects that identify table columns
              | "accessible-for-notify"  only via trap
                   e.g., snmpTrapOID identifies notifications
              | "read-only"
              | "read-write"          but not create
              | "read-create"         => can also write
```

Examples of access

ipForwarding† OBJECT-TYPE
... MAX-ACCESS read-write

ipv4InterfaceTableLastChange OBJECT-TYPE
... MAX-ACCESS read-only

ipv4InterfaceTable OBJECT-TYPE
... MAX-ACCESS not-accessible

ipAddressIfIndex OBJECT-TYPE
... MAX-ACCESS read-create

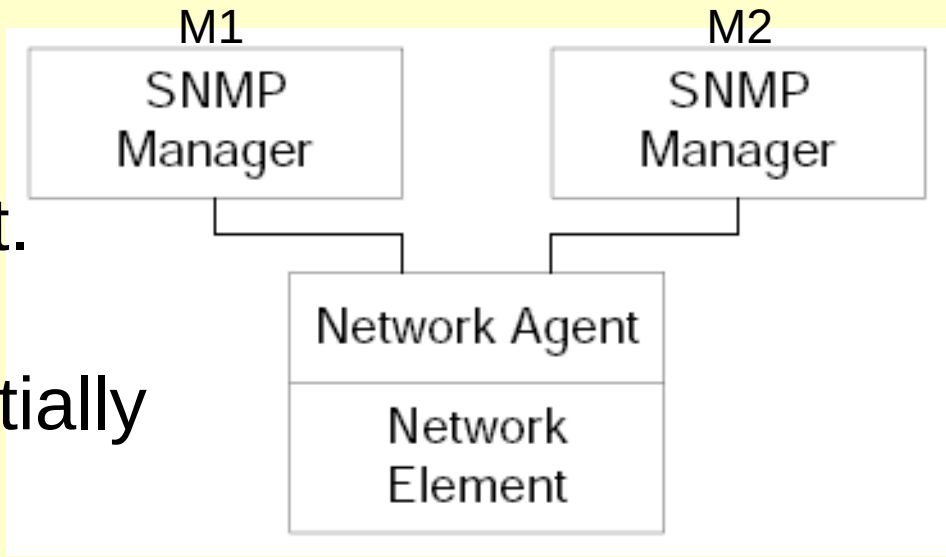
Bonus mark: Since the optimal ipReasmTimeout depends on local conditions, e.g. delays, why should it be read-only?

† You'll use this object in the VLAN configuration exercise

Multiple access scenarios

Multiple managers <TX]
may need to access an object.
Problems if one Mgr accesses object(s) that another has partially changed

Manager may also try to access object while element is updating the object



 Text in figures: SNMP manager, network agent, network element

Fig 4.5 from Subramanian

Multiple access problems

Problems can arise when one “party” (network element or manager) tries to write some values that take multiple steps (e.g. multiple bytes or table row with multiple columns) while another tries to access (read or write) those values.

e.g.:

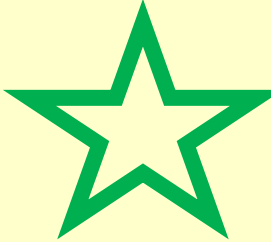
- 2B counter has value 0x00FF. Element increments value, but temporarily only updates LSB first; Manager reads 0x0000; element then sets MSB to 0x01.
- 2 managers (M1, M2) writing to a 2B object:
  - M1 wants to write 0xBEEF, M2 wants to write 0xCAFE
    M1 writes 0xBE, M2 writes 0xCAFE, M1 writes 0xEF
    Result: 0xCAEF

Multiple access solutions

Must prevent interruption of multi-step writing.
=> “atomic operations”† / mutual exclusion / arbitration

- between Manager and Element: No problem:
  - Element can block Manager requests while modifying.
  - Managers don't usually modify objects modified by Element. e.g. performance statistics are read-only Counters: Manager will not get then set & lose events between those requests.
- between Managers: Use SpinLocks ->

† The concept of atomicity here is the same as for “atomic types” previously <UZ] but here it relates to processing whereas before it related to data types.

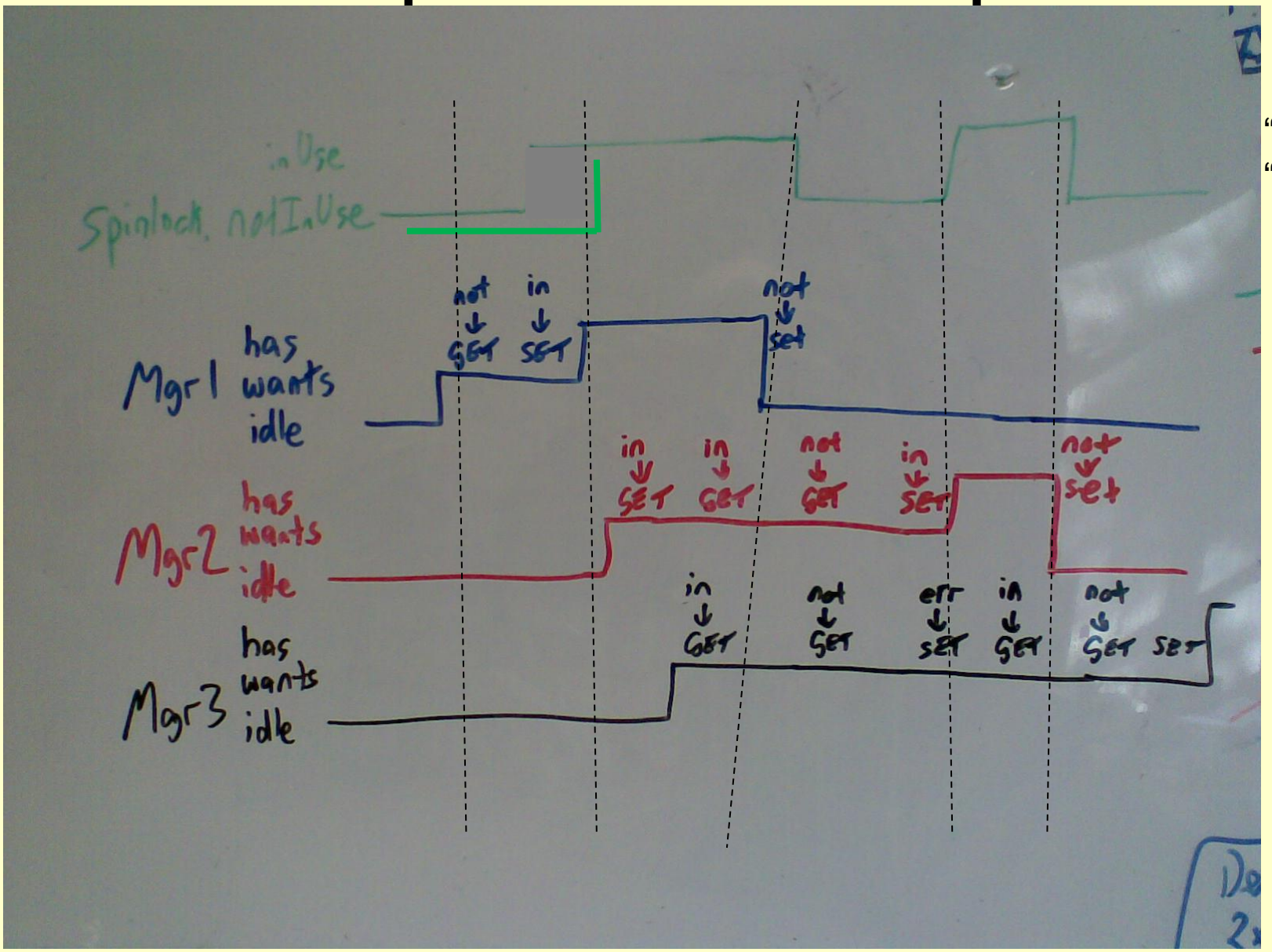


SpinLock mechanism

SMI introduces a `TestAndIncr` type to provide a "SpinLock" mechanism:.

- For SET requests: If new value matches current value, increment value. Else return "`inconsistentValue`" error [1Q>.
- e.g. values of `notInUse(1)` and `inUse(2)`
- Manager reads object until `notInUse`, then tries to set the value; if no other manager got in 1st, value is incremented to `inUse`, else error.

spinLock example



"in" = inUse
"not"="notInUse"

GD

Object status

Once defined in RFCs, objects never disappear s.t.
identifiers won't be reused for another purpose
=> `STATUS` clause indicates degree of support expected.
Possible values:

- `current`: State-of-the art. Implementation is expected (required or optional depending on MODULE-COMPLIANCE <D7] [Y7> grouping)
- `deprecated`: Old-fashioned. Implementation is permitted for interworking with old implementations.
- `obsolete`: Old-fashioned. Should not be implemented.

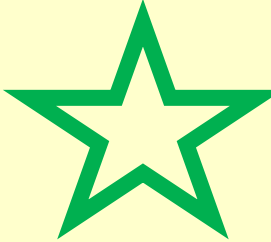


Table issues

Table = SEQUENCE OF rows (“entries” in SNMP-speak)

```
ipv6RouterAdvertTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ipv6RouterAdvertEntry
    INDEX { ipv6RouterAdvertIfIndex }
   ::= { ip 39 }
```

Row/entry = SEQUENCE that defines columns

```
Ipv6RouterAdvertEntry ::= SEQUENCE {
        ipv6RouterAdvertIfIndex          InterfaceIndex,
        ipv6RouterAdvertSendAdverts      TruthValue,
        ipv6RouterAdvertMaxInterval      Unsigned32,
            …
        ipv6RouterAdvertDefaultLifetime  Unsigned32,
        ipv6RouterAdvertRowStatus        RowStatus
    }
```

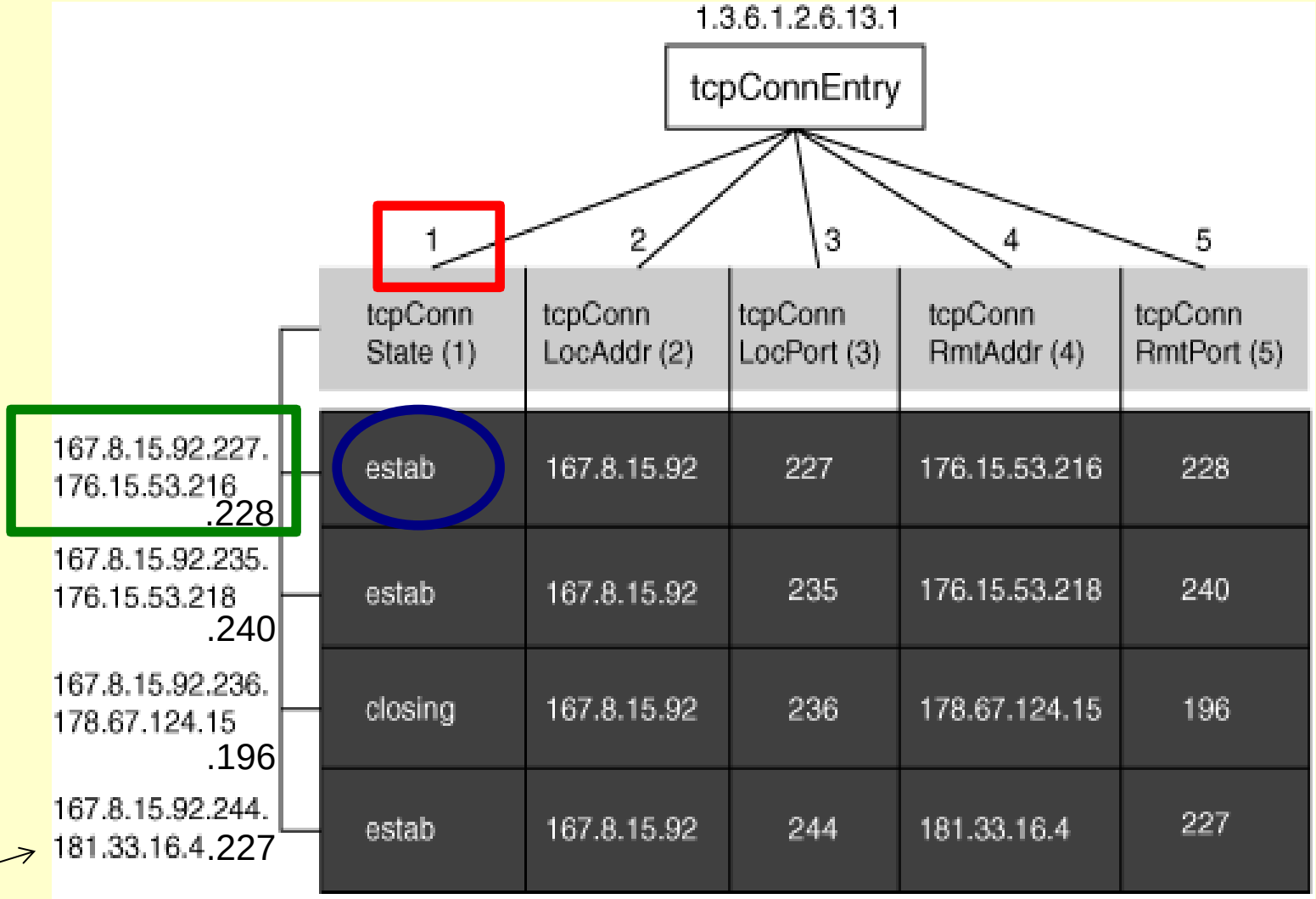

2QL The `ipv6RouterAdvertIfIndex` here is not used to index this table, but rather points to an entry in the interface table

INDEX for a table

- Tables need to specify which columns are used to index/identify rows of the table.
- INDEX clause specifies which (combination of) columns identify a row, e.g. from RFC4022:

```
tcpConnectionEntry OBJECT-TYPE
    SYNTAX     TcpConnectionEntry
    INDEX   { tcpConnectionLocalAddressType,
  tcpConnectionLocalAddress, tcpConnectionLocalPort,
              tcpConnectionRemAddressType,
  tcpConnectionRemAddress, tcpConnectionRemPort }
    ::= { tcpConnectionTable 1 }
```

OID of cell in the oval: 1.3.6.1.2.6.13.1.1.167.8.15.92.227.176.15.53.216.228



These entry identifiers should specify remote port at end, e.g. Top row = 167.8.15.92.227.176.15.53.216.**228**

xz Fig. 6.15 from Clemm

For another example of indexing table objects see slide [QC>

Auxiliary index objects for tables

- OIDs for cells depend on value of cell in column used to index row. If the value of that cell changes, references to other cells in row will be stale.
- Resolve by introducing auxiliary index objects for each row which
  - unambiguously define each row (set by element)
  - max-access = not-accessible - *value* cannot be accessed; OID merely used to identify other cells

```
ipv6RouterAdvertTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ipv6RouterAdvertEntry
    INDEX { ipv6RouterAdvertIfIndex }
    ::= { ip 39 }
```

Creating/deleting rows

- How can a manager create or delete rows?
  - What if creation requires multiple steps <AN], e.g. element creates row with unique index, then manager sets values of cells of row.
- SMIv2 adds `RowStatus` type => RS column
  - Values that may be read: `active, notInService, notReady`
  - Values that may be written:
    - `createAndGo` (row becomes active)
    - `createAndWait` (row becomes?notInService)
    - `active` (change from notInService or notReady)
    - `destroy`

266 See RFC 1903

Object groups

Related objects can be grouped

e.g.

```
ipSystemStatsGroup OBJECT-GROUP
    OBJECTS   { ipSystemStatsInReceives,
                ipSystemStatsInOctets,
                ipSystemStatsInHdrErrors,
                ipSystemStatsInNoRoutes,
                ipSystemStatsInAddrErrors,
                ipSystemStatsInUnknownProtos,
                ipSystemStatsInTruncatedPkts,
                ipSystemStatsInForwDatagrams,
...
} ::= { ipMIBGroups 8 }
```


ipInReceives, ipInHdrErrors, ipInAddrErrors are part of the IP MIB [JV>

Compliance requirements

MODULE-COMPLIANCE macro indicates which groups are mandatory, or otherwise, for compliance to standard <01].

e.g.

```
ipMIBCompliance2 MODULE-COMPLIANCE
    MANDATORY-GROUPS { ipSystemStatsGroup,
  ipAddressGroup,... }

    GROUP ipSystemStatsHCOctetGroup
    DESCRIPTION
          "This group is mandatory for systems that
  have an aggregate bandwidth of greater than 20MB.
  Including this group does not allow an entity to
  neglect the 32 bit versions of these objects."
...
::= { ipMIBCompliances 2 }
```

Outline

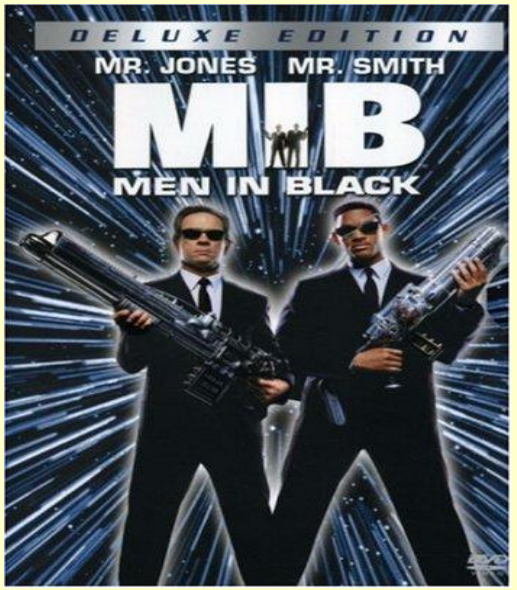
Object Identifiers

Structure of Management Information

- ASN.1 basics
- RFC conventions

MIBs

Management Information Bases (MIB)



- MIB = set of objects for a protocol/resource
  - MIBs defined for most protocols/equipment
- MIB repository (for standard and proprietary elements) at http://www.oidview.com/mibs/detail.html

- MIB for Internet Protocols
  - originally defined in RFC1156
  - MIB-2 (aka MIB-II) defined in RFC1213 (STD 17) and updated with RFCs including RFC 4293

2PK Image from http://images.amazon.com/images/P/B0000640SB.01.LZZZZZZZ.jpg

See http://tele9752.wikia.com/wiki/MIBs_for_common_protocols

Figures on the following slides are from Subramanian

mib-2

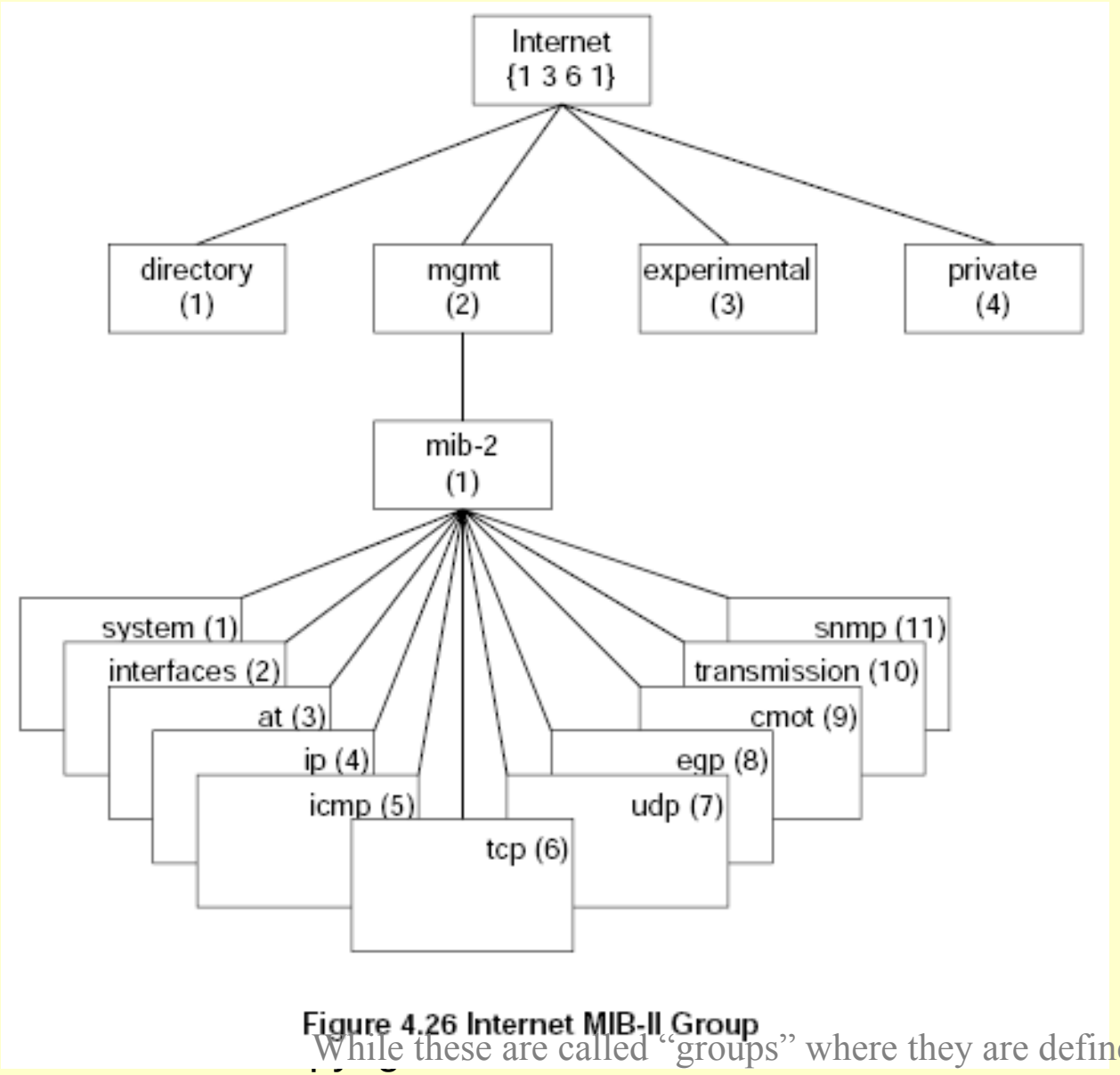


Figure 4.26 Internet MIB-II Group

While these are called "groups" where they are defined in RFC1213, that RFC does not group these objects in the sense of <JR]

2H5 Text in figures: mib-2, system, interfaces, at, ip

System MIB

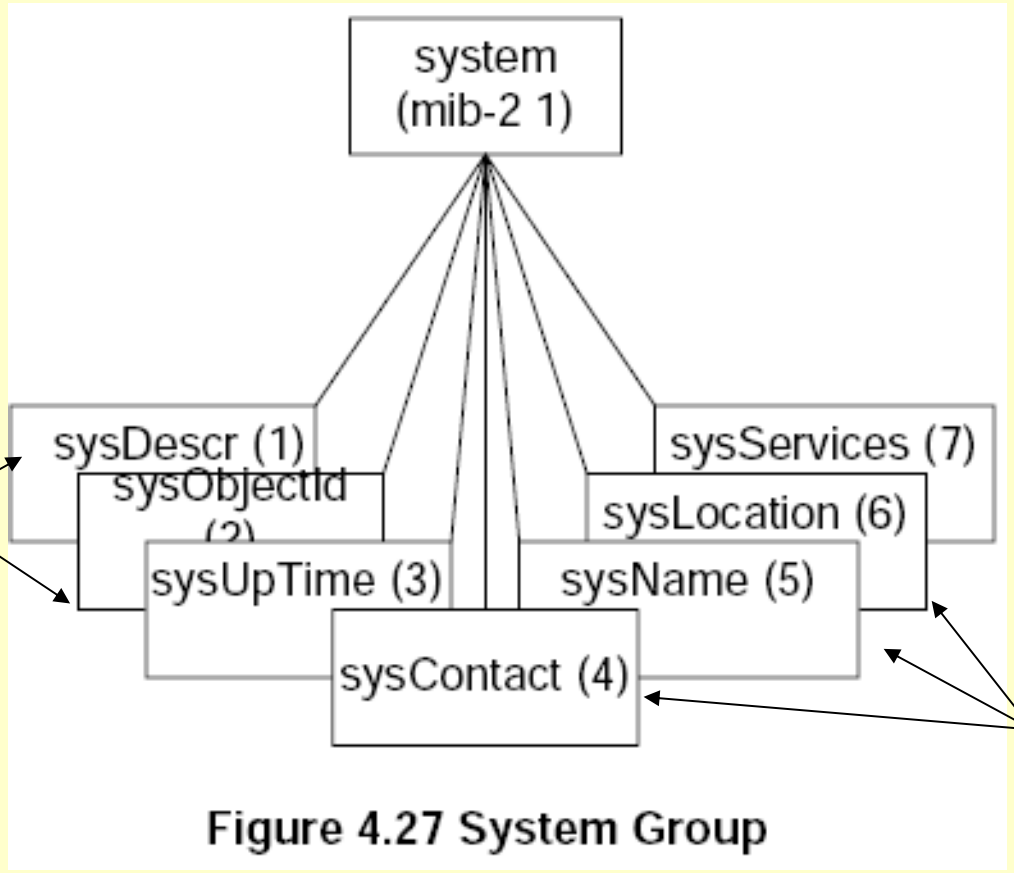
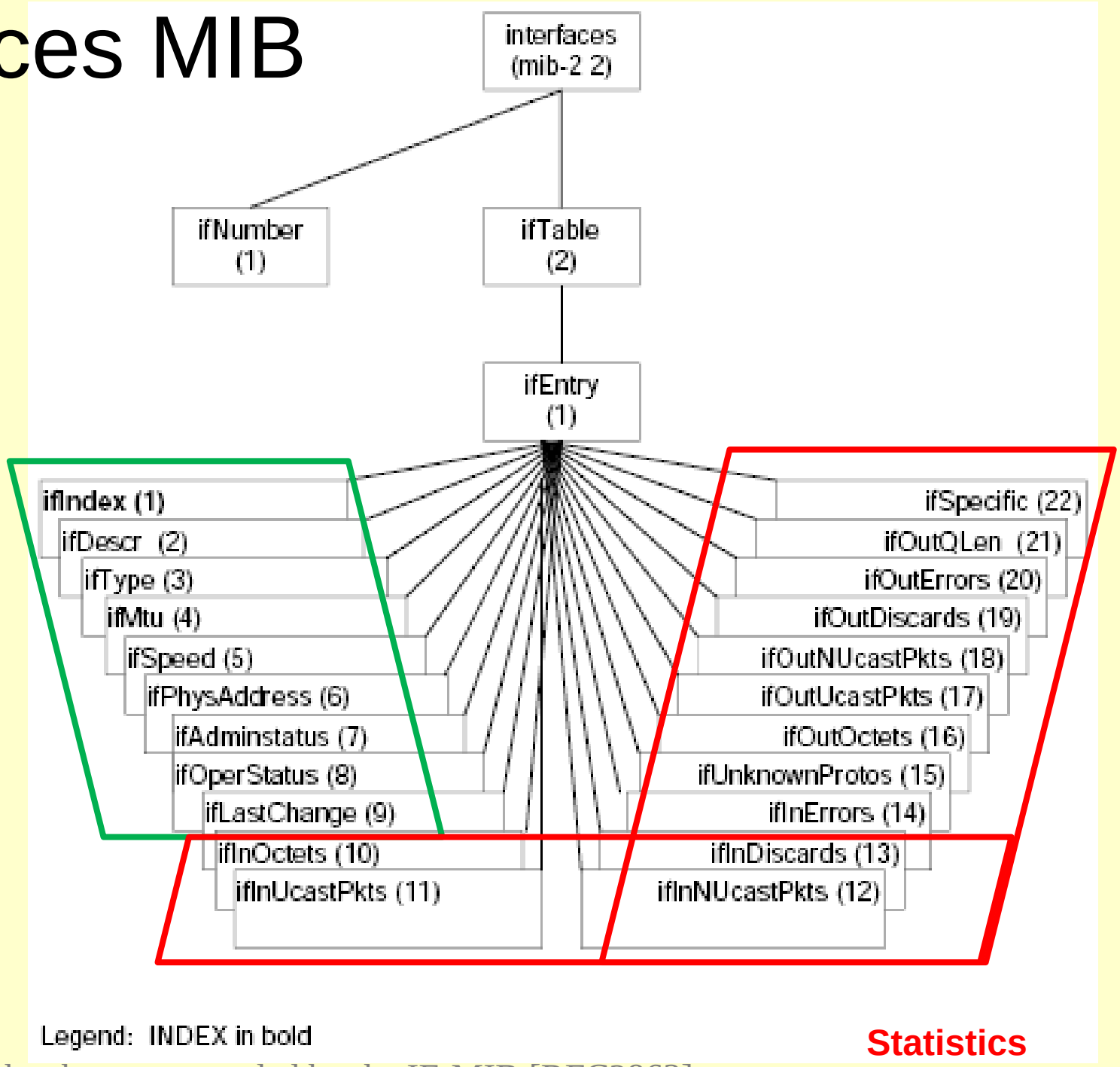


Figure 4.27 System Group

Another generic element MIB is the Entity MIB [RFC4133]

28F Text in figures: system, sysObjectID, sysUpTime,

Interfaces MIB



Configuration

Statistics

The interfaces MIB has been superseded by the IF-MIB [RFC2863].

2XW Text in figures: interfaces

Updated by the INTERFACES group MIB [RFC 2863]

“at” MIB

- ARP results stored in “Address Translation” / “ipNetToMedia*” & “ipNetToPhysicalTable” [RFC4293]

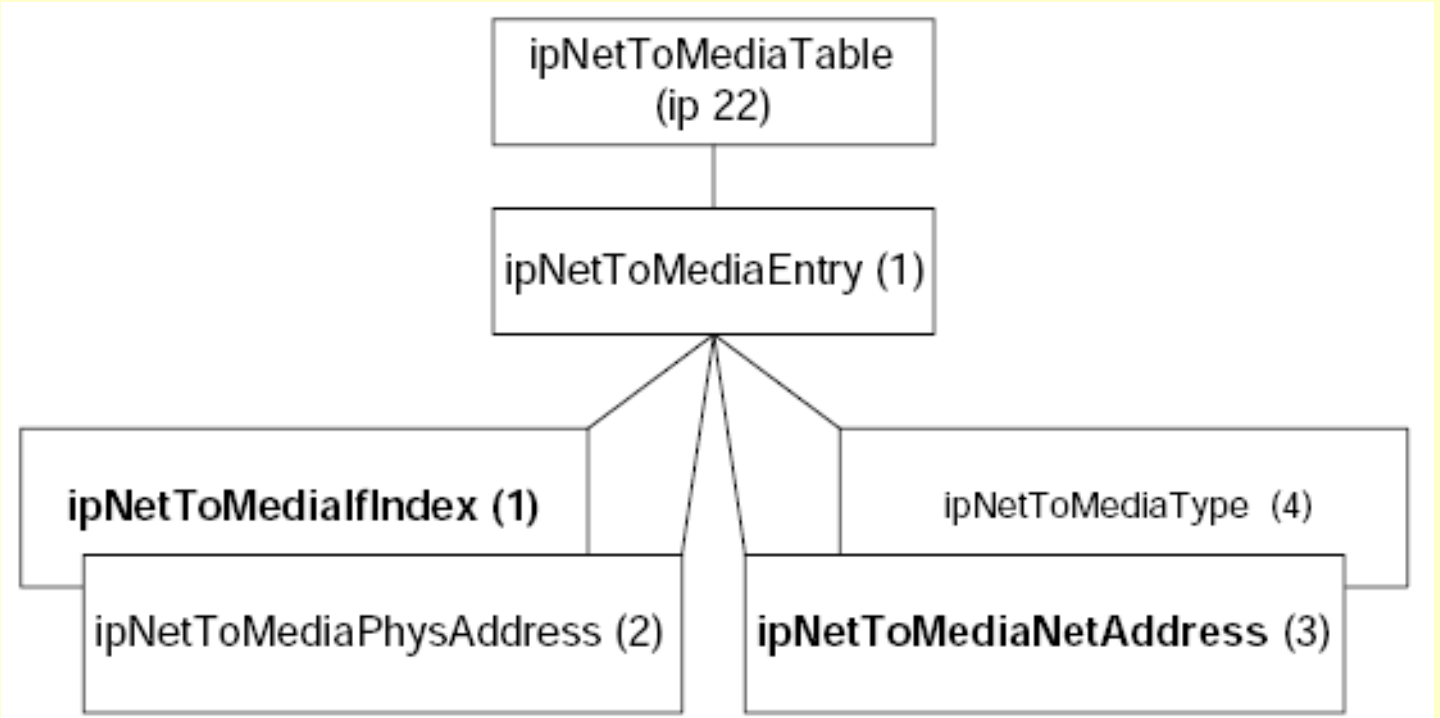
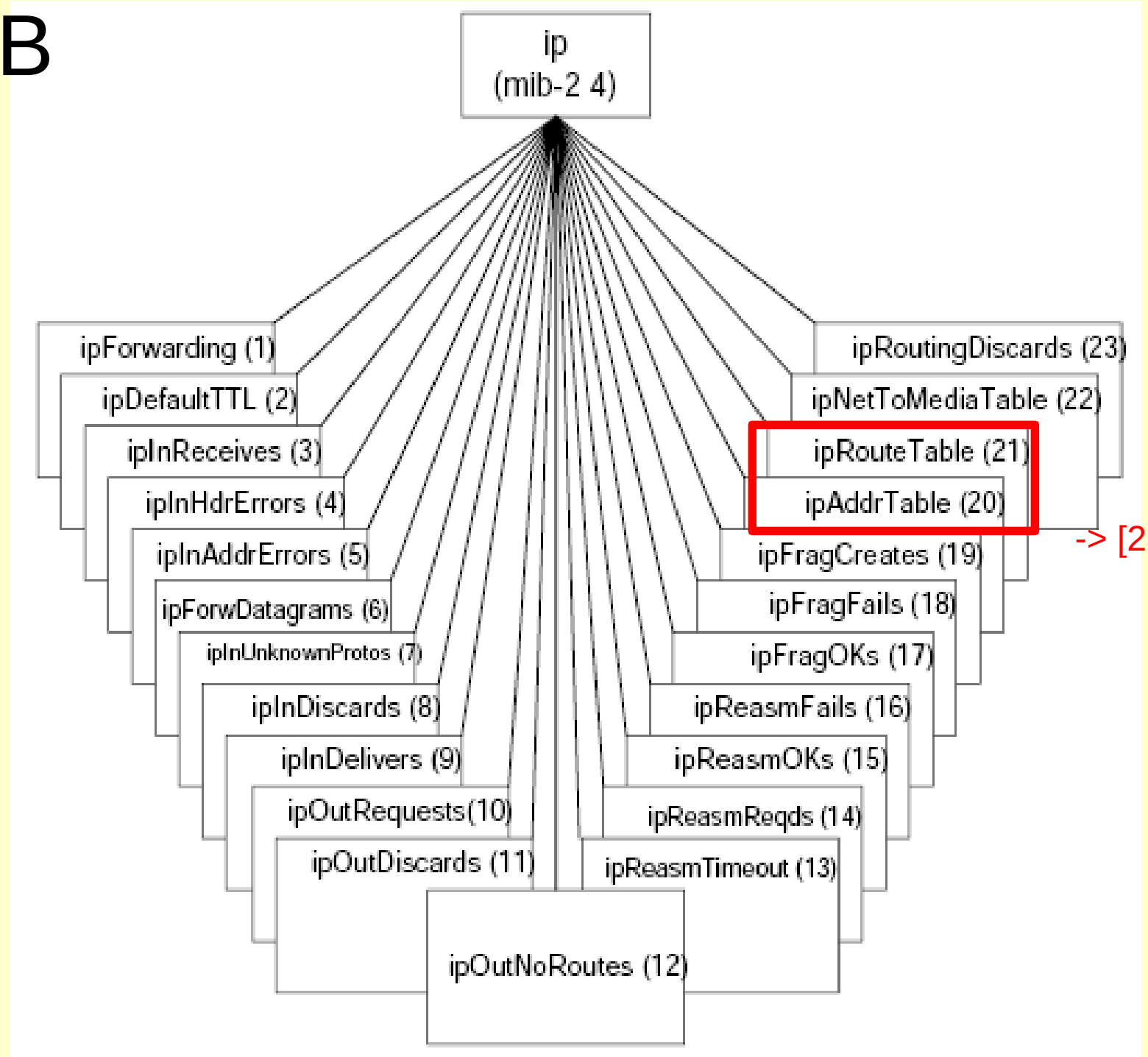


Figure 4.32 IP Address Translation Table

IP MIB



-> [23] [59]

2JV Text in figures: ipInReceives, ipInHdrErrors, ipInAddrErrors

<JR] described a different grouping of the IP statistics

IP addr table

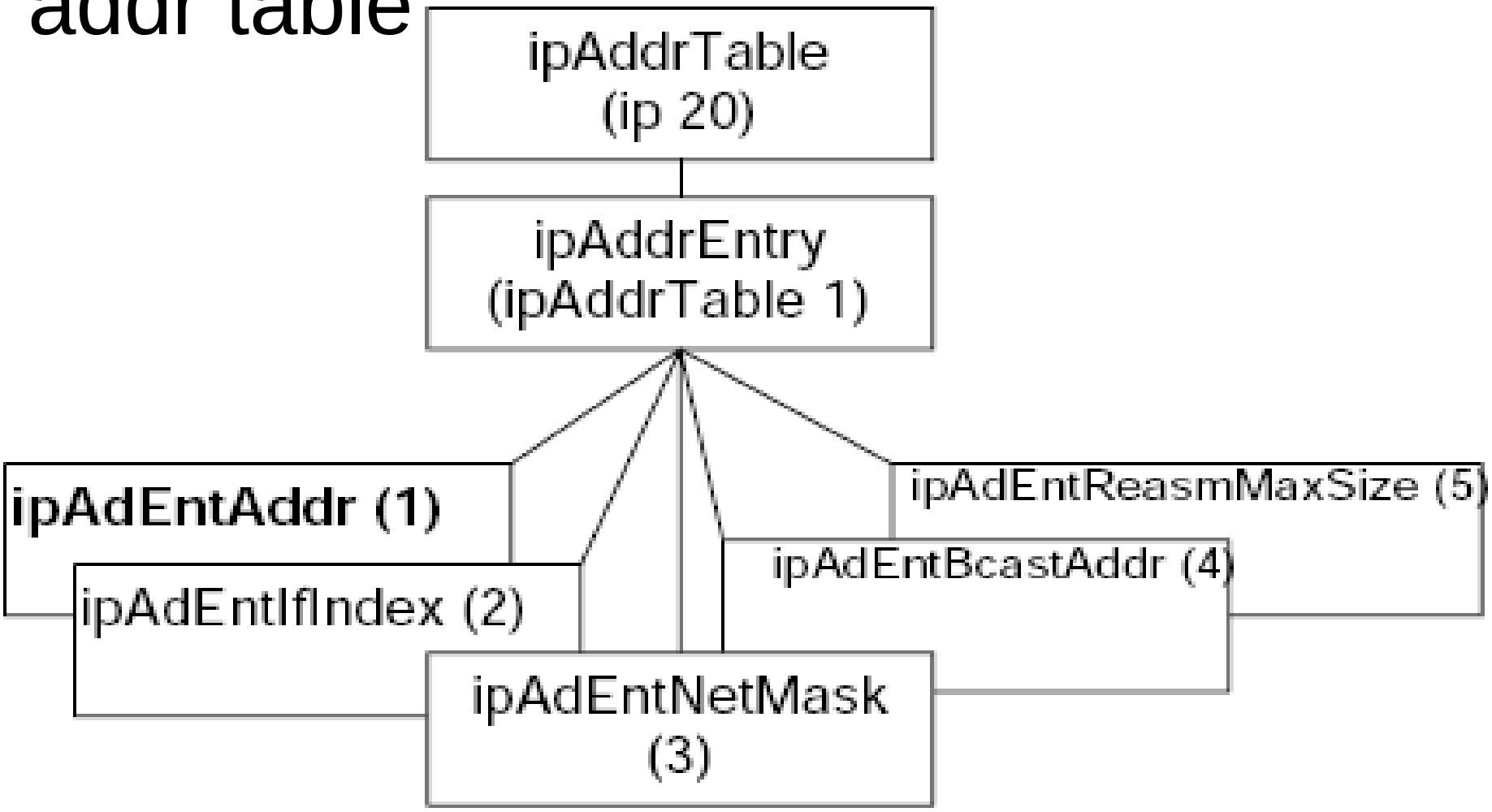


Figure 4.30 IP Address Table

RFC4293 renames `ipAddrTable` to `ipAddressTable` and moves `ipAdEntReasmMaxSize` to more appropriate `ipv4InterfaceTable`

IP route table

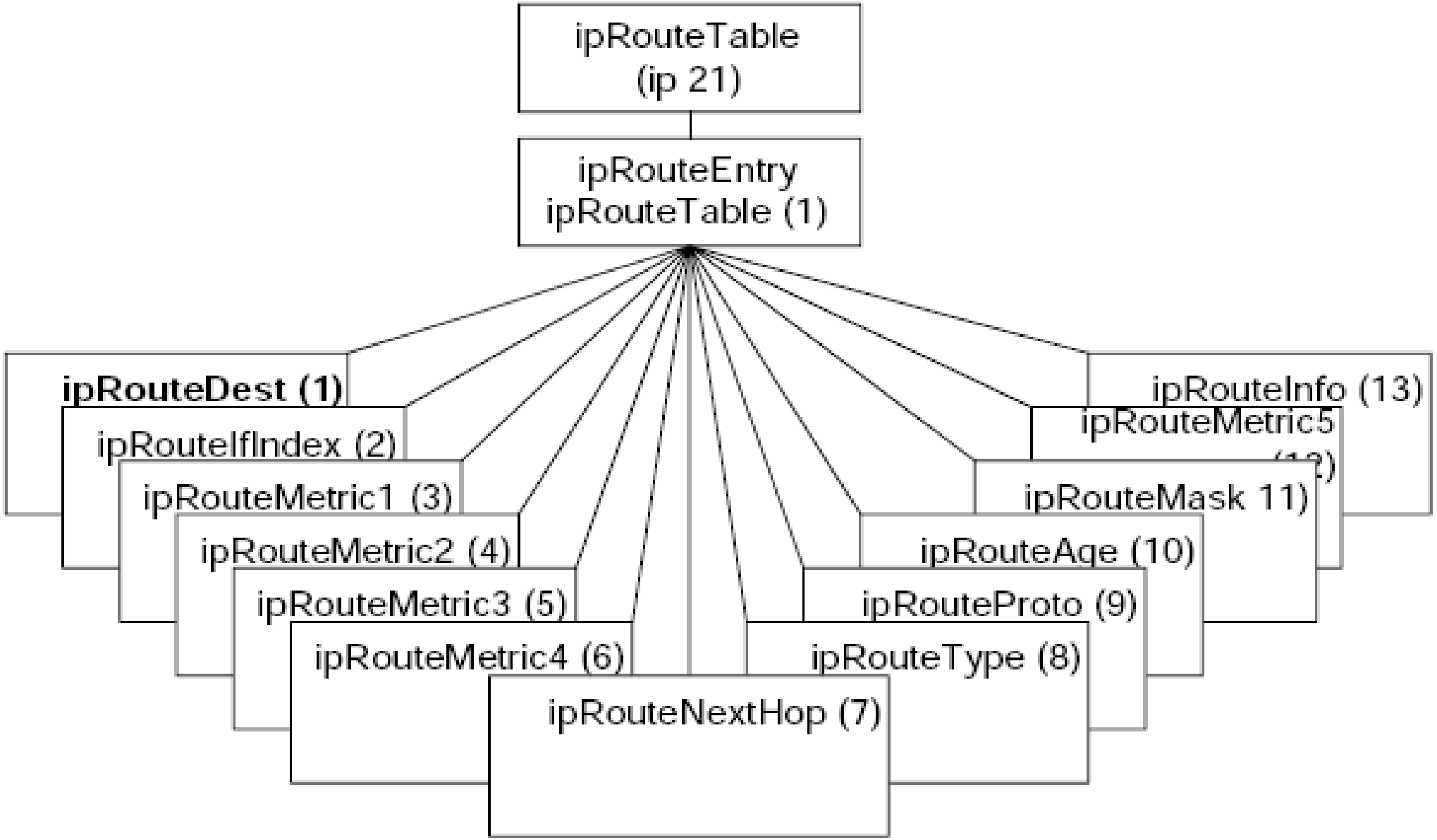
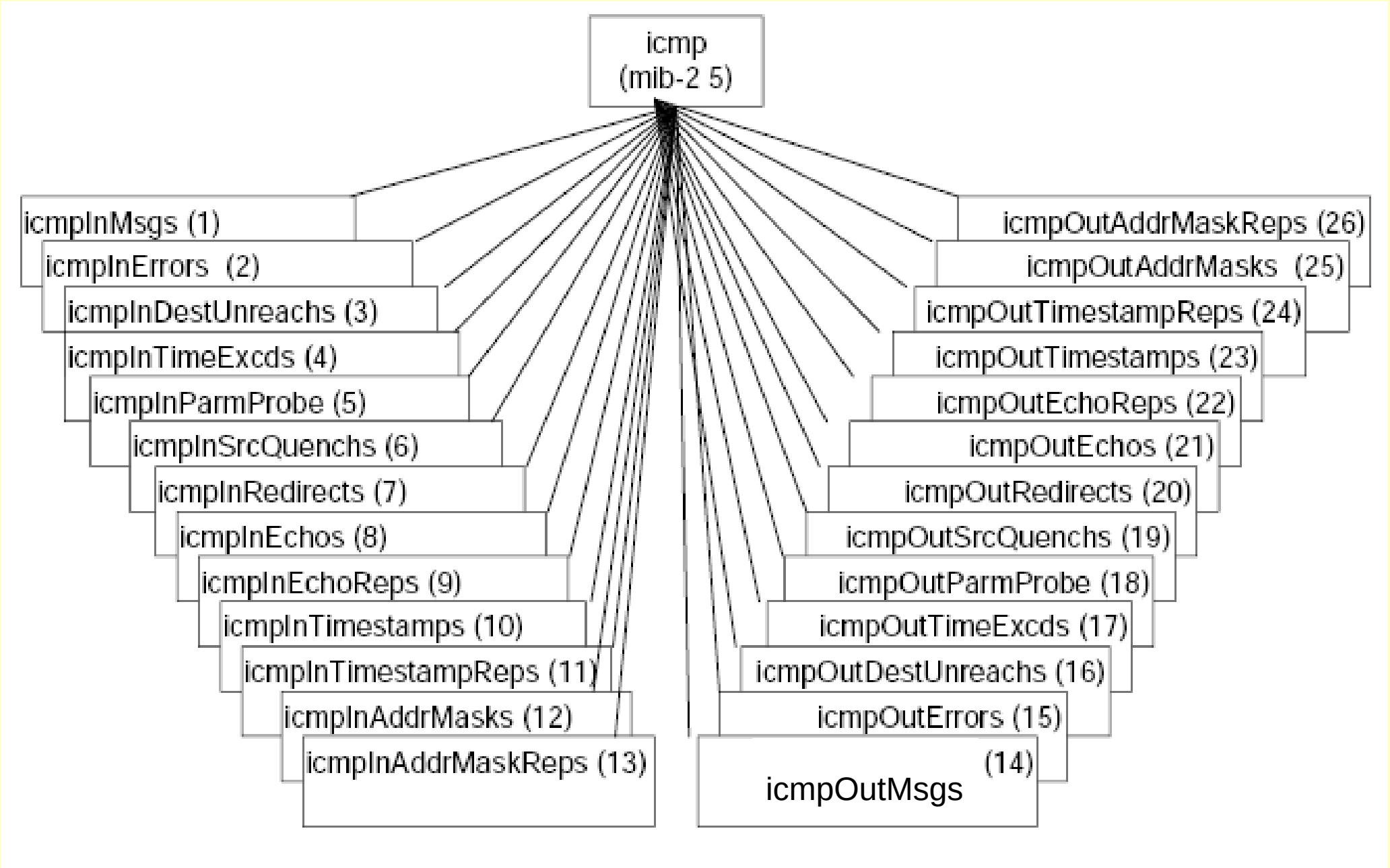


Figure 4.31 IP Routing Table

ICMP MIB



SNMP MIB

Traffic stats

snmpInPkts
snmpOutPkts
snmpInTotalReqVars
snmpInTotalSetVars
snmpInGetRequests
snmpInSetRequests
snmpInGetResponses
snmpInTraps
snmpOutGetNexts
snmpOutGetRequests
snmpOutSetRequests
snmpOutGetResponses
snmpOutTraps

Control

snmpEnableAuthenTraps
(whether to generate authentication-failure traps)

Error counts

snmpInBadVersions
snmpInBadCommunityNames
snmpInBadCommunityUses
snmpInASNParseErrs
snmpInTooBigs
snmpInNoSuchNames
snmpInBadValues
snmpInReadOnlys
snmpInGenErrs
snmpOutTooBigs
snmpOutNoSuchNames
snmpOutBadValues
snmpOutGenErrs

Case Diagrams

Given multitude of objects, how can designers/users:

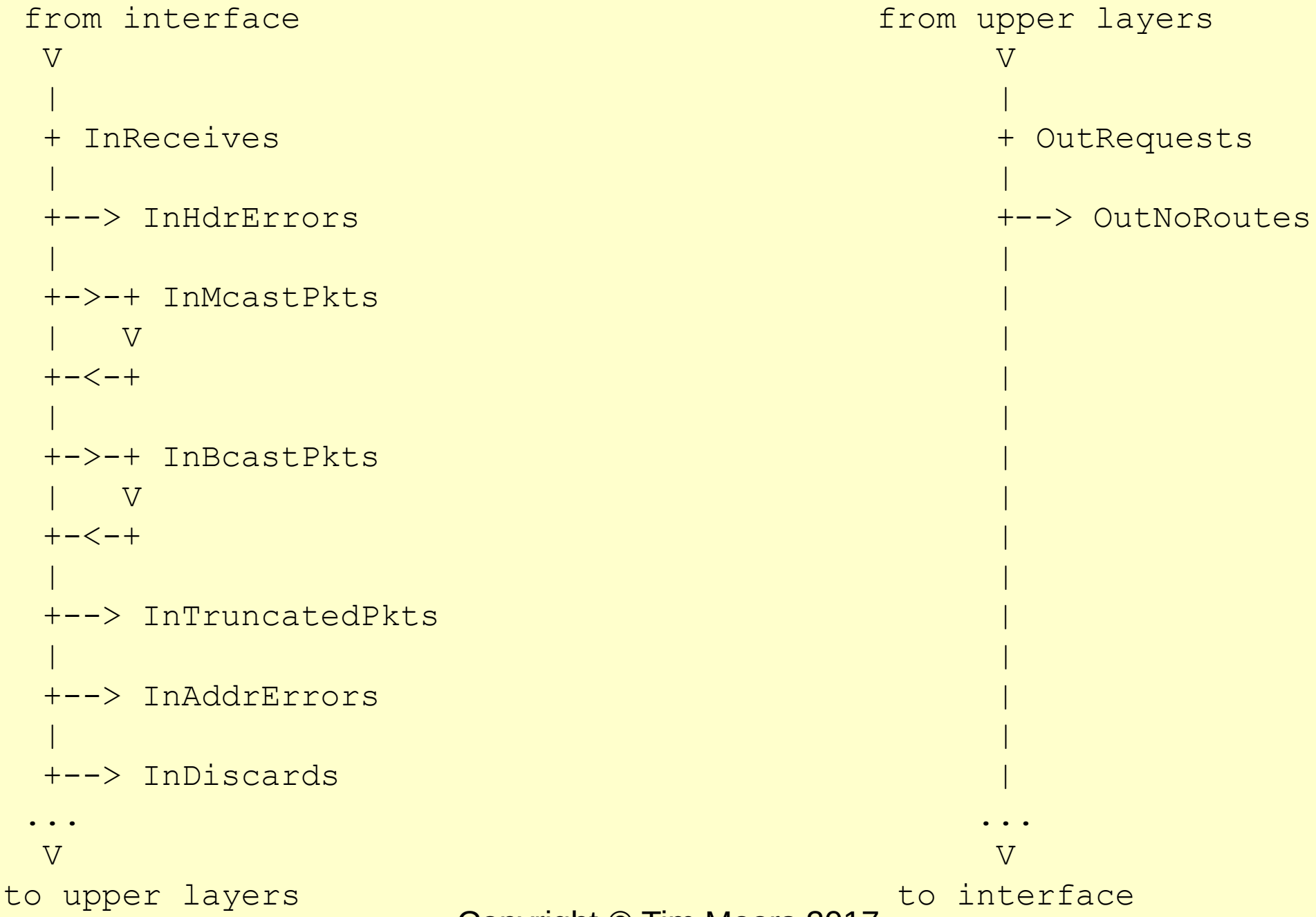
- ensure objects uniquely but comprehensively cover all events?
- visualise relationship between objects?

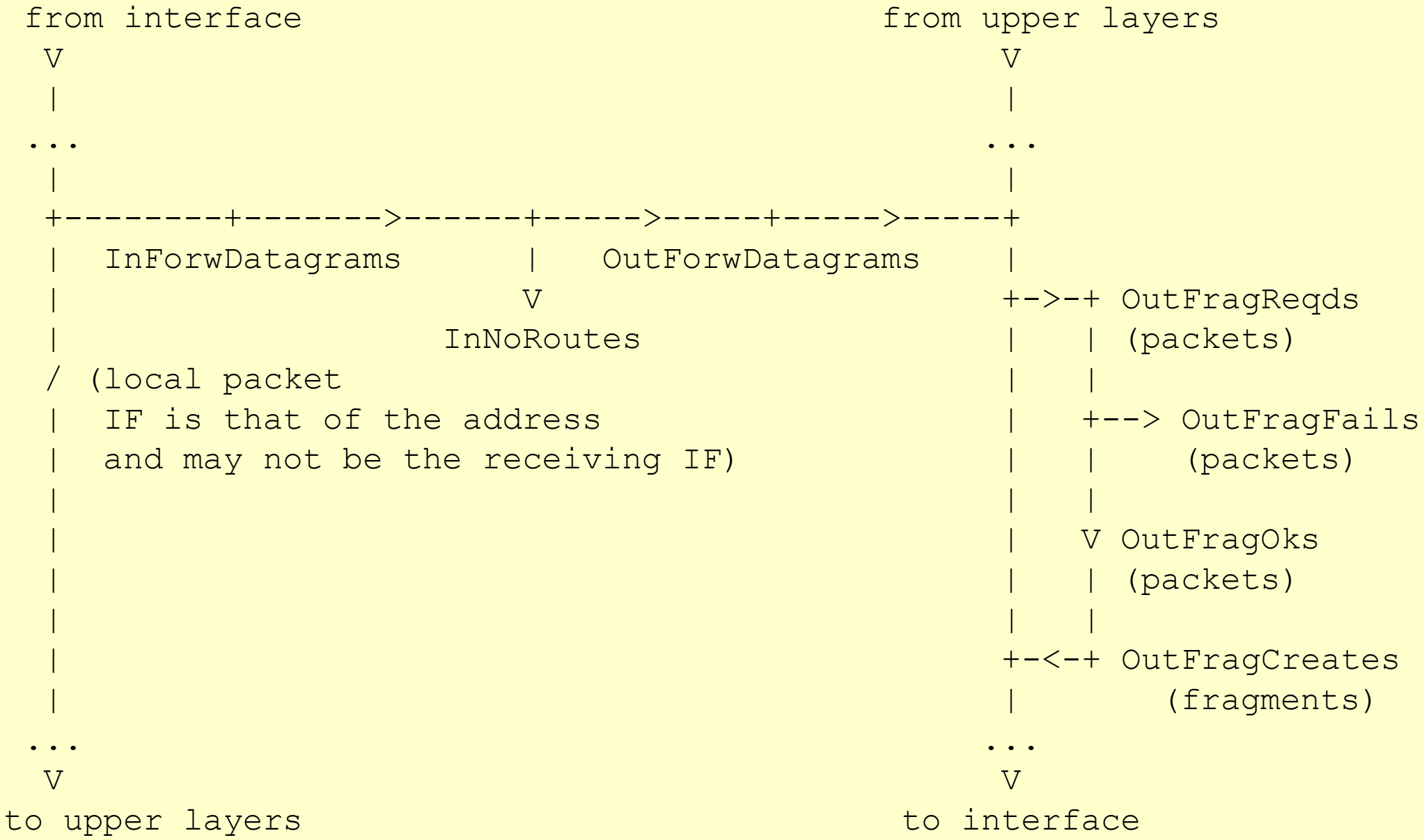
A: Case Diagrams (Named after inventor)

Illustrate propagation of a packet through a layer, with counters covering each possible path.

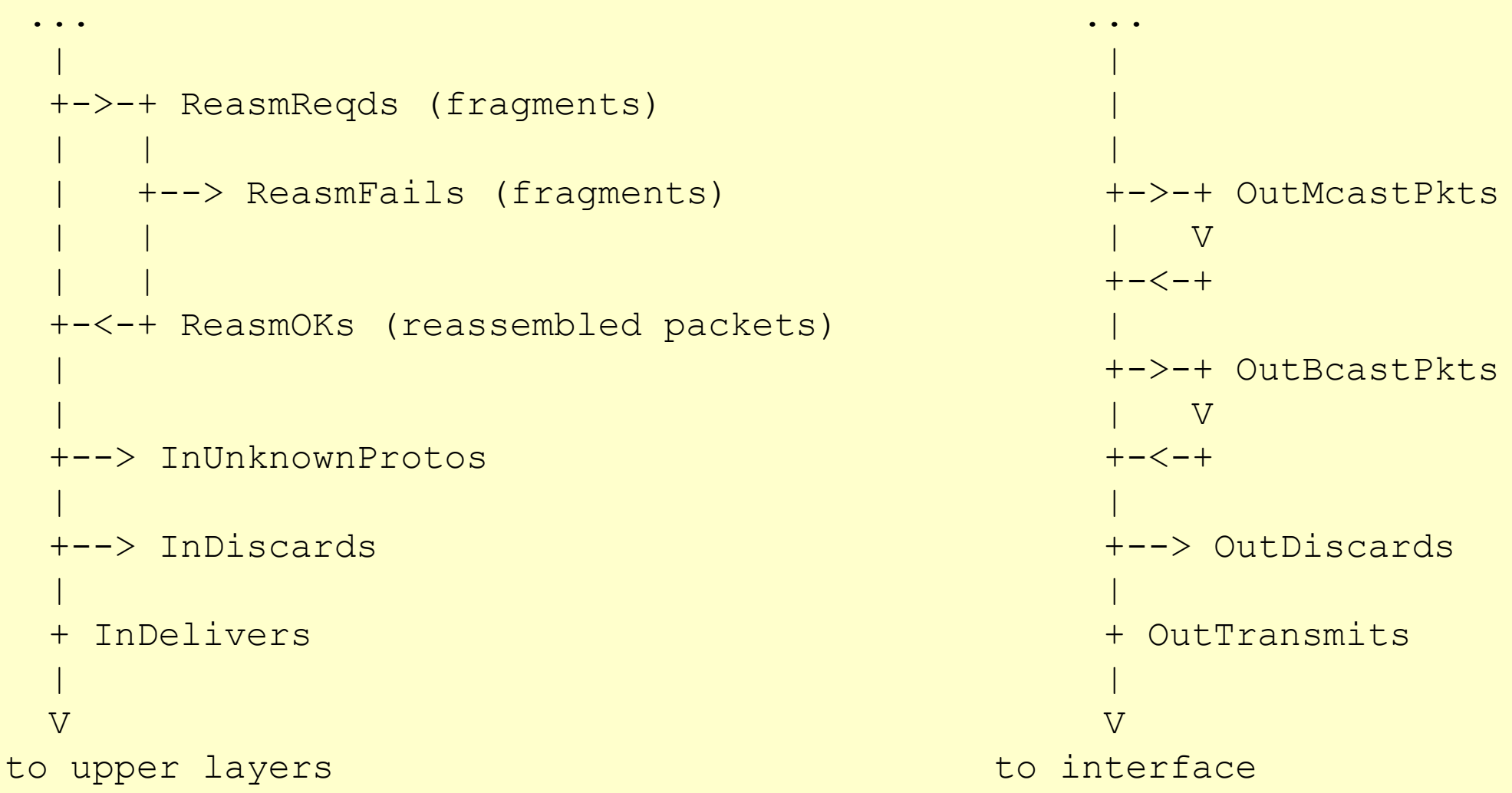
Case diagram for IP MIB from RFC 4293 appears on the following 3 slides (slightly edited to remove references to footnotes & extra vertical space)

J. Case and C. Partridge: “Case diagrams: a first step to diagrammed management information bases”, *ACM Computer Communication Review*, 19(1):13-6, DOI:10.1145/66093.66094





(“IF” = interface)



Things to think about

- Critical thinking:
  - Given that modern programming languages (e.g. Python) have weaker typing than earlier ones (e.g. C), should the strong typing of MIB objects be reconsidered?
  - Read RFC3197 about why MIBs weren't useful for DNS
- Engineering methods:
  - How are data sources for other sophisticated systems (e.g. instrumentation on chemical and power plants) managed? (in terms of identifying sources and their relationships)
- Links to other areas:
  - The MIB tree provides a naming hierarchy like DNS
- Independent learning:
  - Explore available objects through these resources: Registry, Repository, MIB Repository