COMP3310/6331 – #20

Measuring, monitoring and SNMP

Dr Markus Buchhorn: *markus.buchhorn@anu.edu.au*

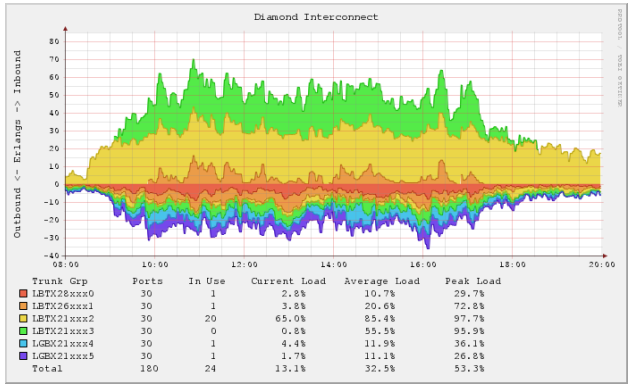
Network monitoring

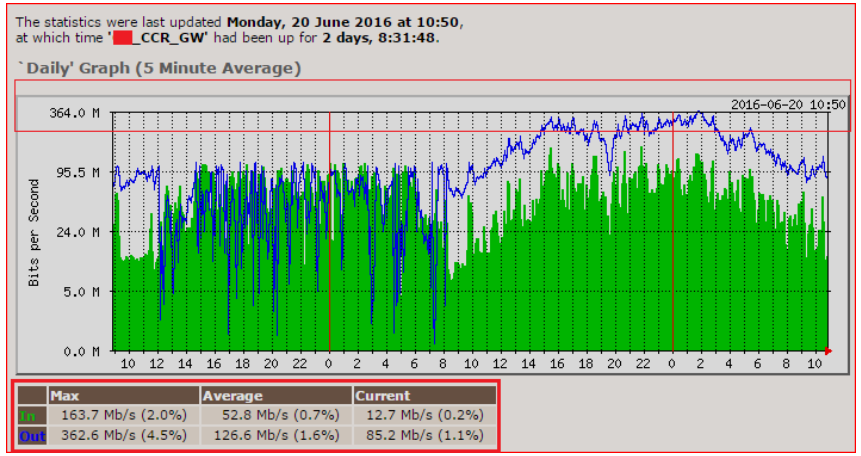
- Measuring networks – and monitoring
  - What do you measure
  - How do you measure it

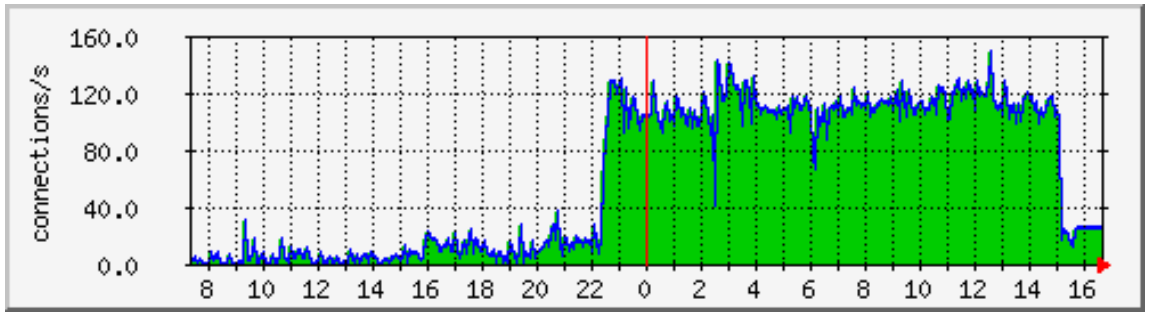
- Want to know:
  - How busy is some/all of the network?
  - Is there congestion (somewhere)?
  - Are there errors?
  - Is the hardware/software ok?
  - Is there a bug in the network? (literally?!?)
  - Has something changed, for the worse, or the better?
  - Are applications being fed the right packets?
  - Is routing behaving as expected?
  - …

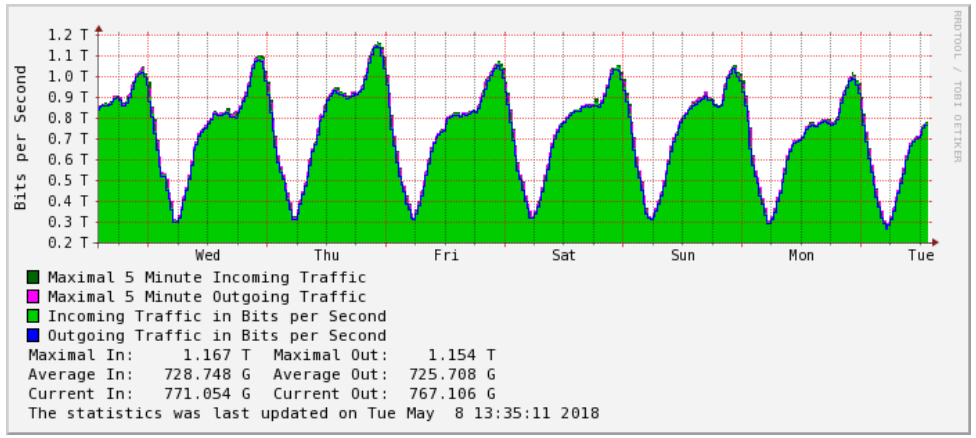
Performance over time

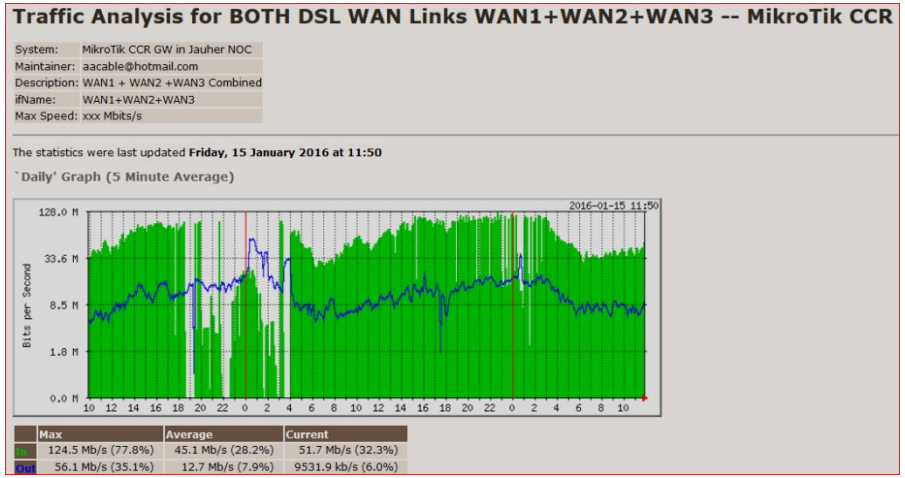
- *Capacity planning*
- *Outages*
- *Patterns*



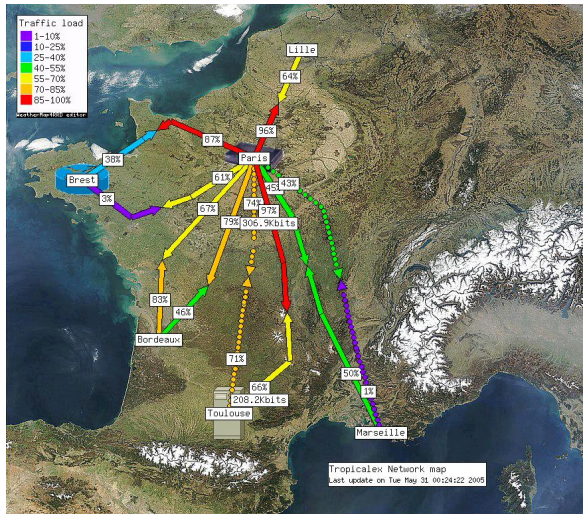


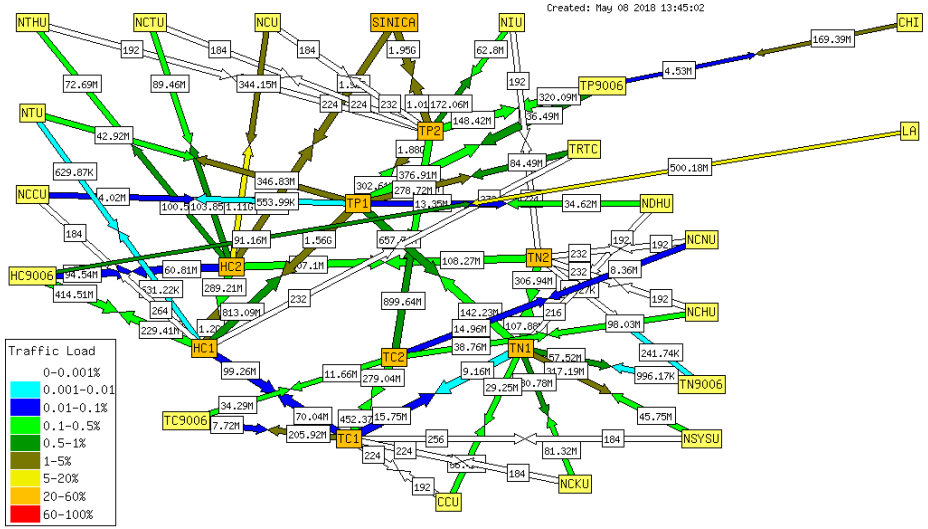


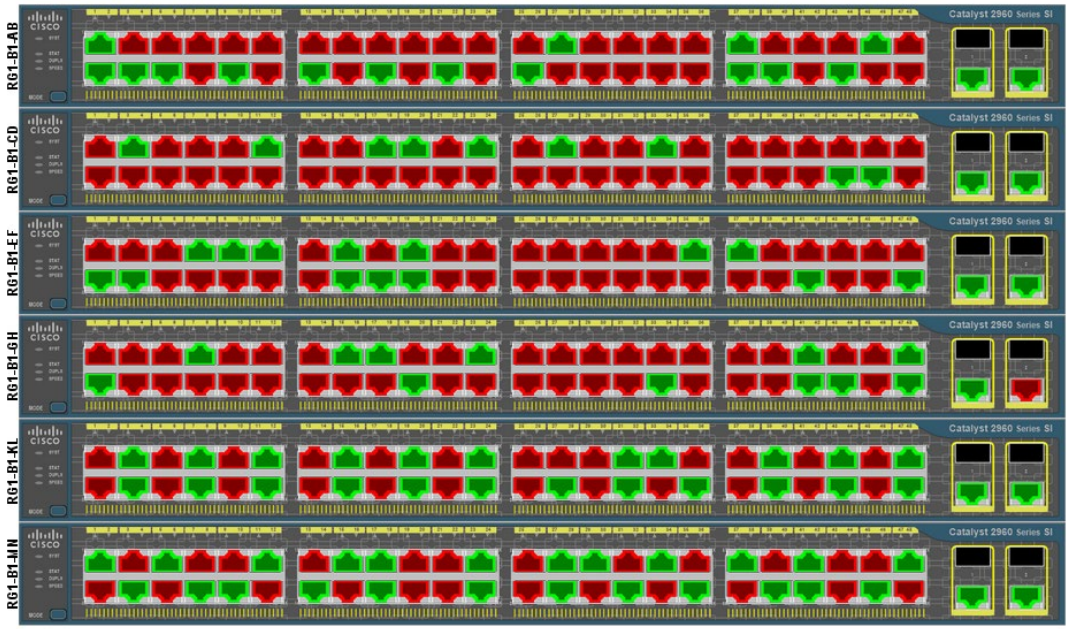




Performance at a moment: Network status







And:
<u>What's out there?</u>
Network Discovery

Network feedback

- **ECN** – Explicit Congestion Notification
- **ICMP** – Internet Control Management Protocols
  - Used passively and actively (ping, traceroute, …)
- **TCP ACK**nowledgements
- **Application** measures
- …

- *No unified view*
- *No aggregated view, in space or time*

Two domains

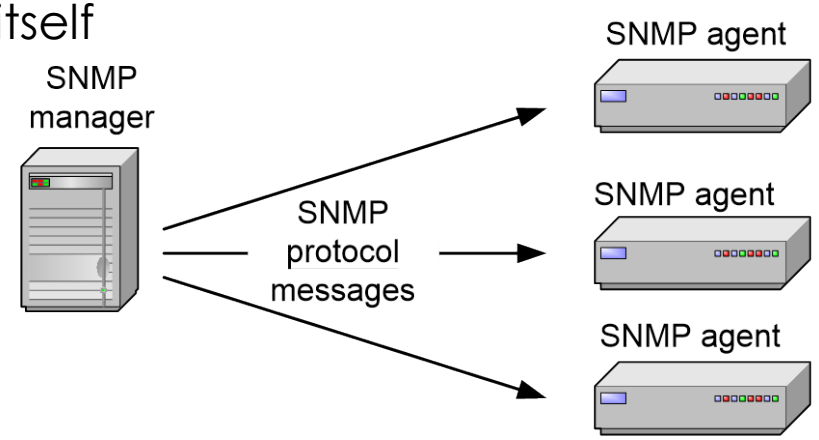
- Within your administrative domain (interior)
  - You have authority
  - Get information from everywhere on your network
  - Put some software on each device
  - Probe, measure, scan, …
- Beyond your administrative domain (exterior)
  - No authority
    - Except maybe a contract?
  - Ask somebody else to put some software on each device, and share

Simple Network Management Protocol (SNMP)

- **Design requirements: We want…**
- Reach everywhere
  - All sizes, types of devices
    - Switches, routers, access points, printers, servers, …
    - Support devices that are too small, too simple, too hard, too old, …
- Lightweight
  - no interference on device
- Operate when things are under stress
  - Identify what is struggling/failing, and when
  - Help to fix/improve things
- Scale to large number of devices and parameters
  - Global naming, delegated, vendor-independent, extensible
- Provide both queries/response and command/control
- *And add some trivial security and upgrade it much, much later*

SNMP

- An application framework
- For managing/monitoring network resources
- Components of SNMP:
  - SNMP agents
  - SNMP managers
  - Management information bases (MIBs)
  - SNMP protocol itself



SNMP components

- **Agent:** software on the equipment
  - maintains configuration and current state in a database.
  - **Proxies:** an agent that talks with non-SNMP devices

- **Management Information Bases (MIBs)** describes the database.
  - MIB, MIB-II (RFC 1213) – and millions more
  - *Structure of Mgmt Info (SMI) defines sets of related objects in a MIB*

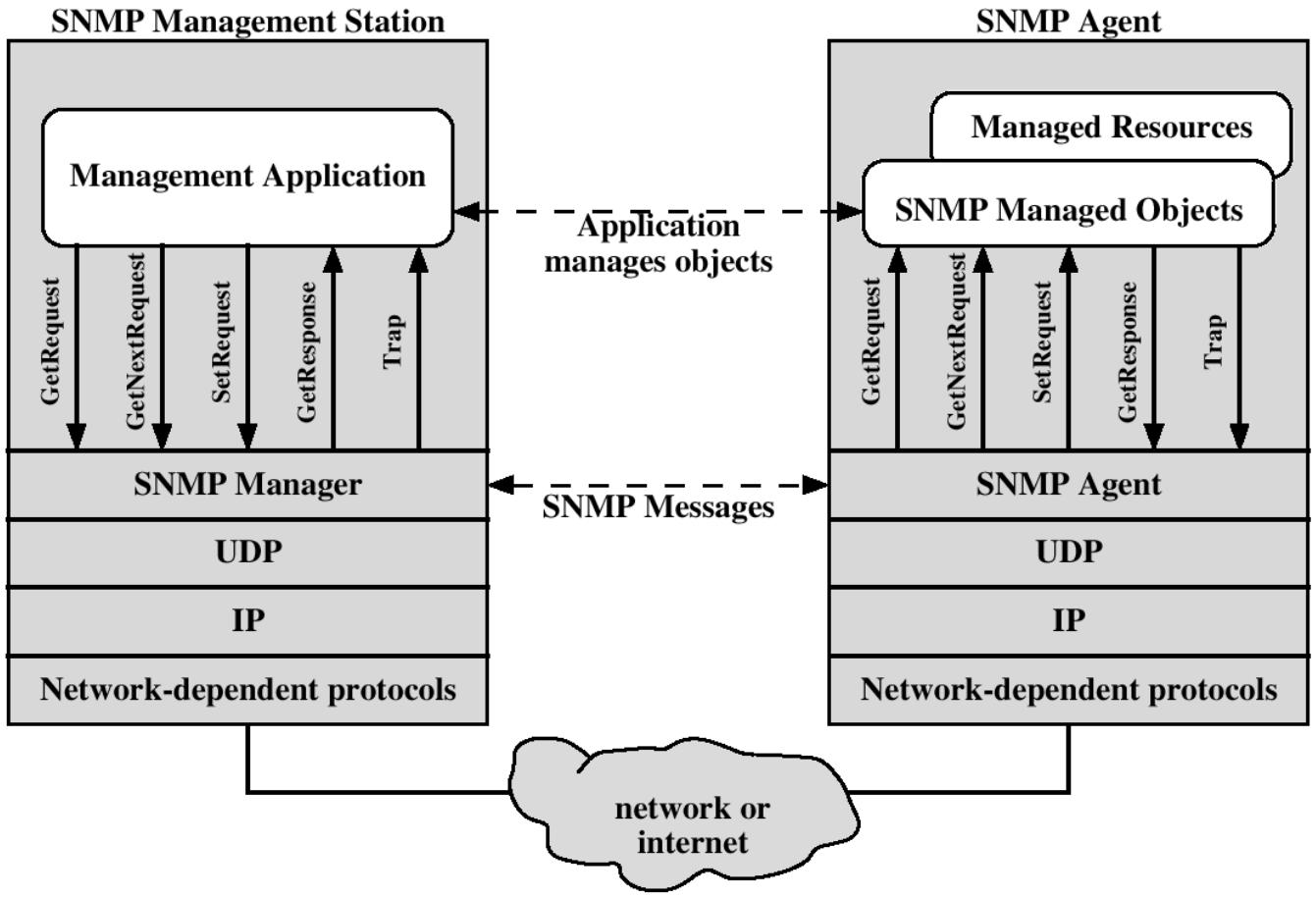
- **Manager:** application that contacts an agent
  - to query or modify the database at the agent.
  - Part of Network Management Systems (NMS)

- **SNMP protocol:**
  - SNMPv1, v2(*), v3

Information design for lightweight SNMP agents

- No rates, no calculations
- No absolute clocks
- No history
- Just
  - Counters and gauges,
  - Time since start-up
  - Strings, Identifiers
- “Timeticks”, in 1/100ths sec.
- Command/control through variable setting

SNMP protocol



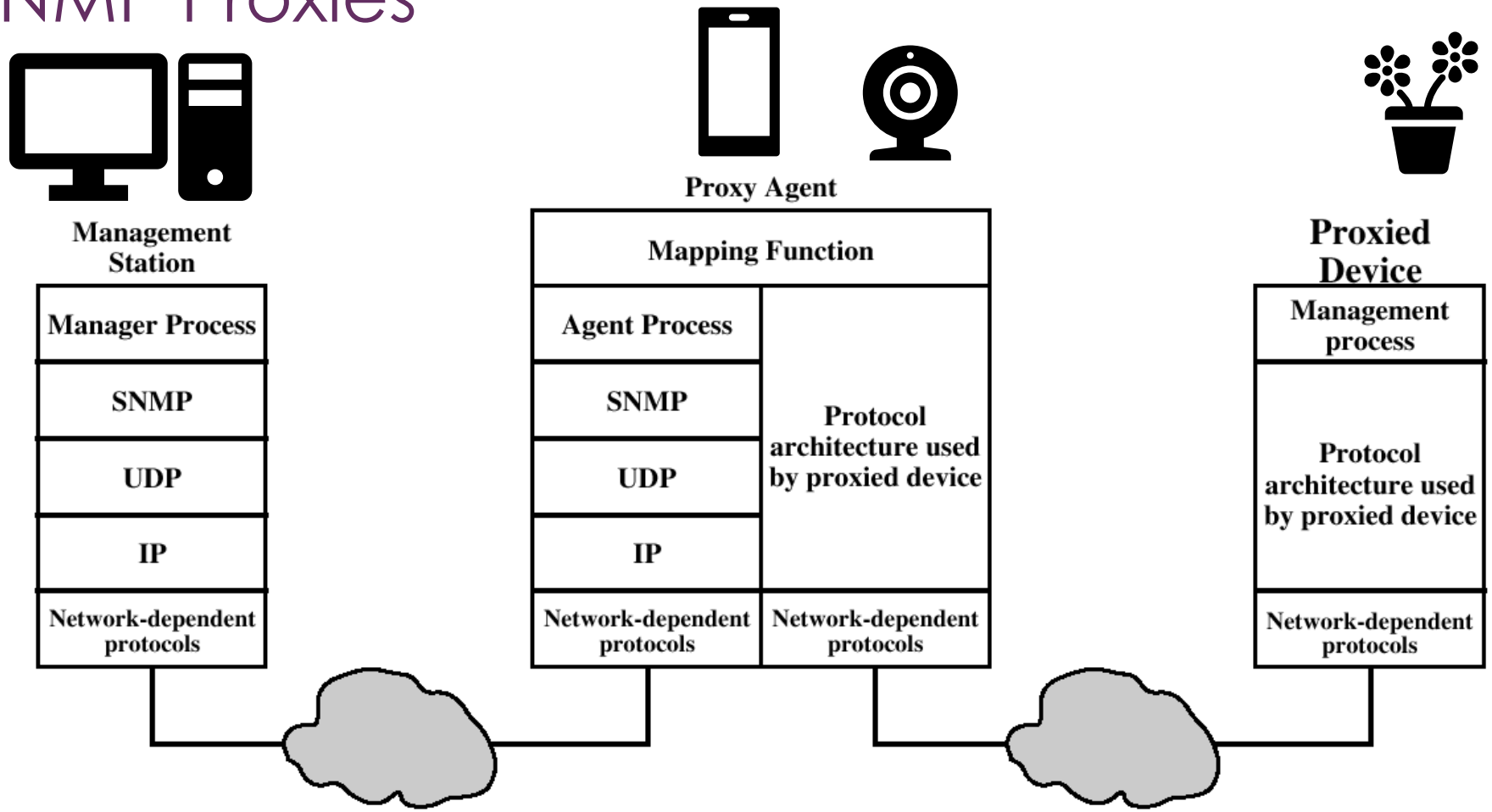
SNMP protocol

a. Passive open by both client and server

b. Exchange of request and response messages

c. Server sends trap message

SNMP Proxies



SNMP messages

- SNMP/UDP is connectionless
  - Use a request ID to maintain a session
- SNMP messages are 'protocol data units' (PDUs)
  - Different versions of SNMP use the same PDU for different messages
    - We're still living through that pain…
- Messages have particular capabilities (SNMPv1):
  - **Get** – the value of a object from an agent
  - **Set** – the value of a object from an agent
  - **Notify** – a manager that the agent has had an event

SNMP(v1) Protocol

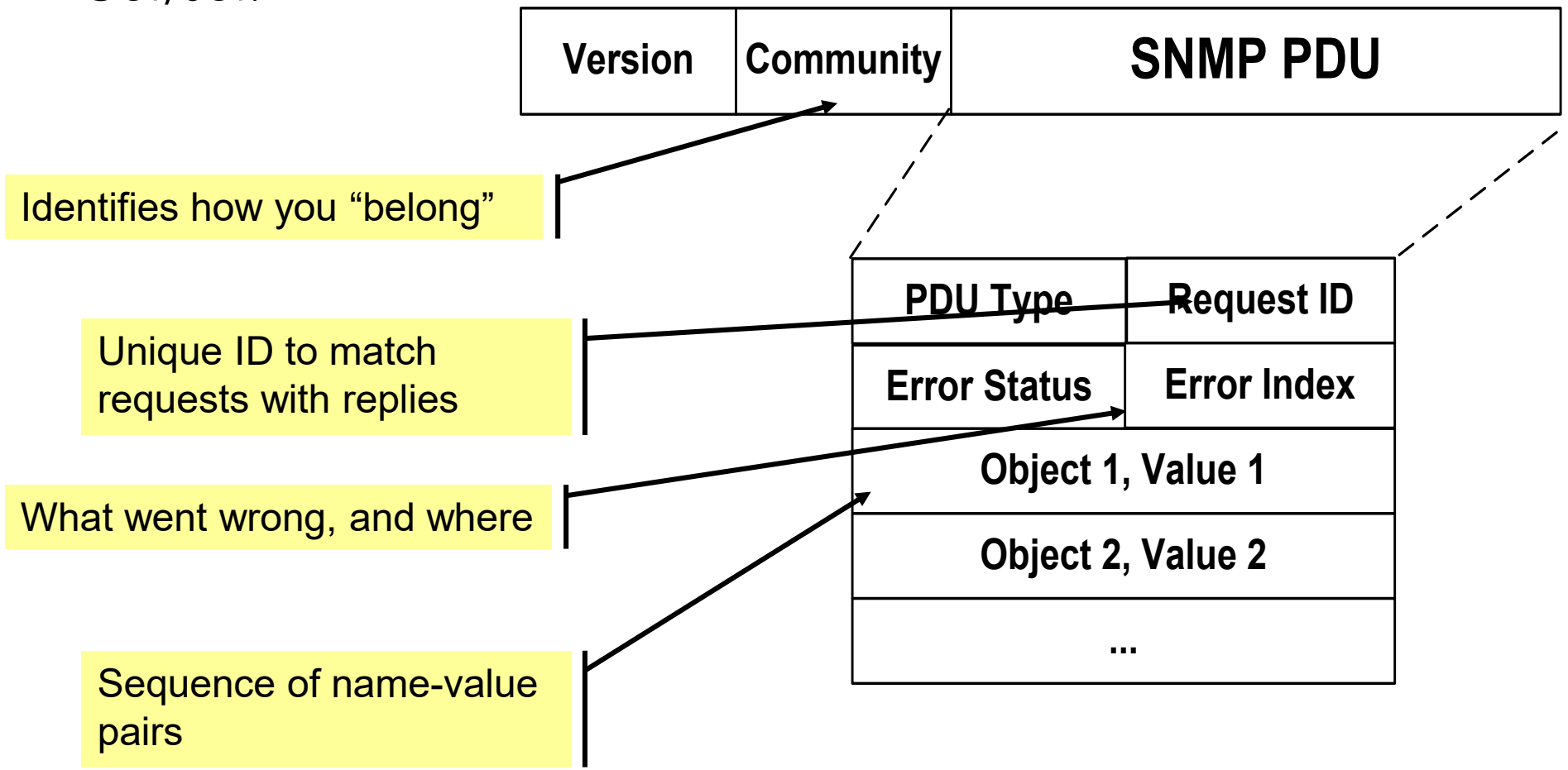
- *On-demand:*
  - **Get-request:** Request the values of one/several objects
  - **Get-next-request.** Requests the value of the "next" object.
  - **Set-request.** Modify the value of one or more objects
  - **Get-response.** Agent response to a request.
- *Triggered:*
  - **Trap:** A notification from an agent to a manager, some event at the agent.

Traps

- Traps are sent asynchronously by an agent to a manager
- 6 core traps:
  - **linkDown**: An interface went down
  - **linkUp**: An interface came up
  - **coldStart**: Unexpected restart (system crash)
  - **warmStart**: Expected restart (manual reboot)
  - **AuthenticationFailure**: Somebody tried to query, but …
  - **egpNeighbourLoss**: Link is up but my neighbour has gone
- And ~2^{32} others (vendor specific)

Format of SNMP (v1/v2) Packets

- Get/Set:



SNMP community

- SNMPv1 defines "communities"
  - specify access to specific variable sets
  - read-write, read only, none
- Each SNMP message includes community name
  - Like a password
  - Unencrypted!!
- Typical values:
  - Read-only: "Public"
  - Read-write: "Private"
- Slight enhancement: agent/manager relationship
  - IP address of permitted managers, stored on agent
- First thing fixed in v2…

SNMP Versions

- Three versions in use today:
  - SNMPv1 (1990)
  - SNMPv2c – *[and three more]* (1996)
    - *Adds "GetBulk" function*
    - *Adds federated monitoring capabilities (manager to manager)*
    - *Adds TCP transport option*
    - *Adds 64bit counters*
  - SNMPv3 (2002)
    - *SNMPv3 started from SNMPv1 (and not SNMPv2c)*
    - *Addresses security*

- All versions are still used today.
- Many SNMP agents and managers support all three versions.

SNMP Security

- SNMPv1 uses "community" strings for authentication
  - In plain text without encryption

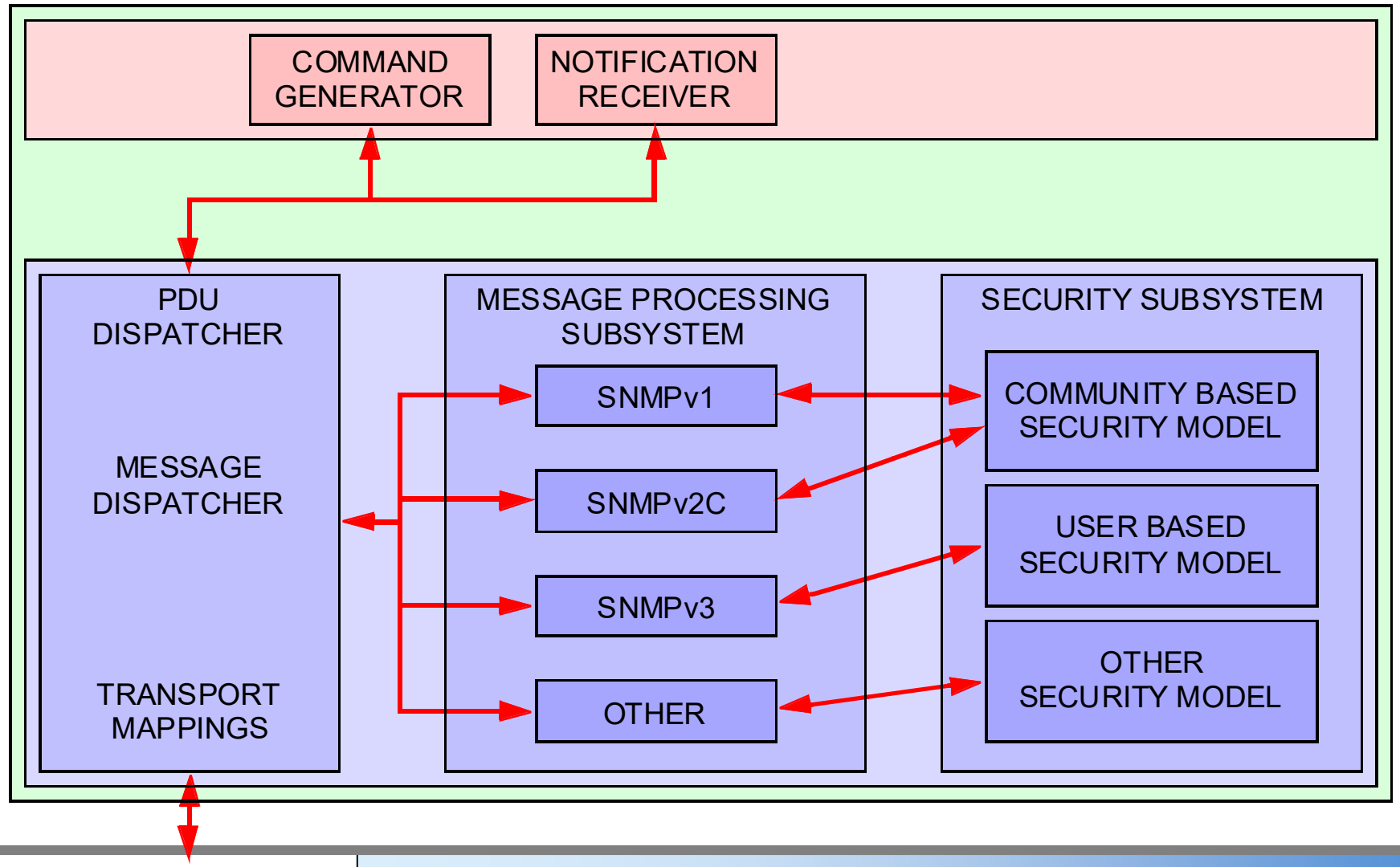
- SNMPv2 was supposed to fix security problems, but effort derailed
  - The "c" in SNMPv2c stands for "community"??

- SNMPv3 has key security features:
  - Ensure that a packet has not been tampered with (**integrity**),
  - Ensures that a message is from a valid source (**authentication**)
  - Ensures that a message cannot be read (**privacy**).

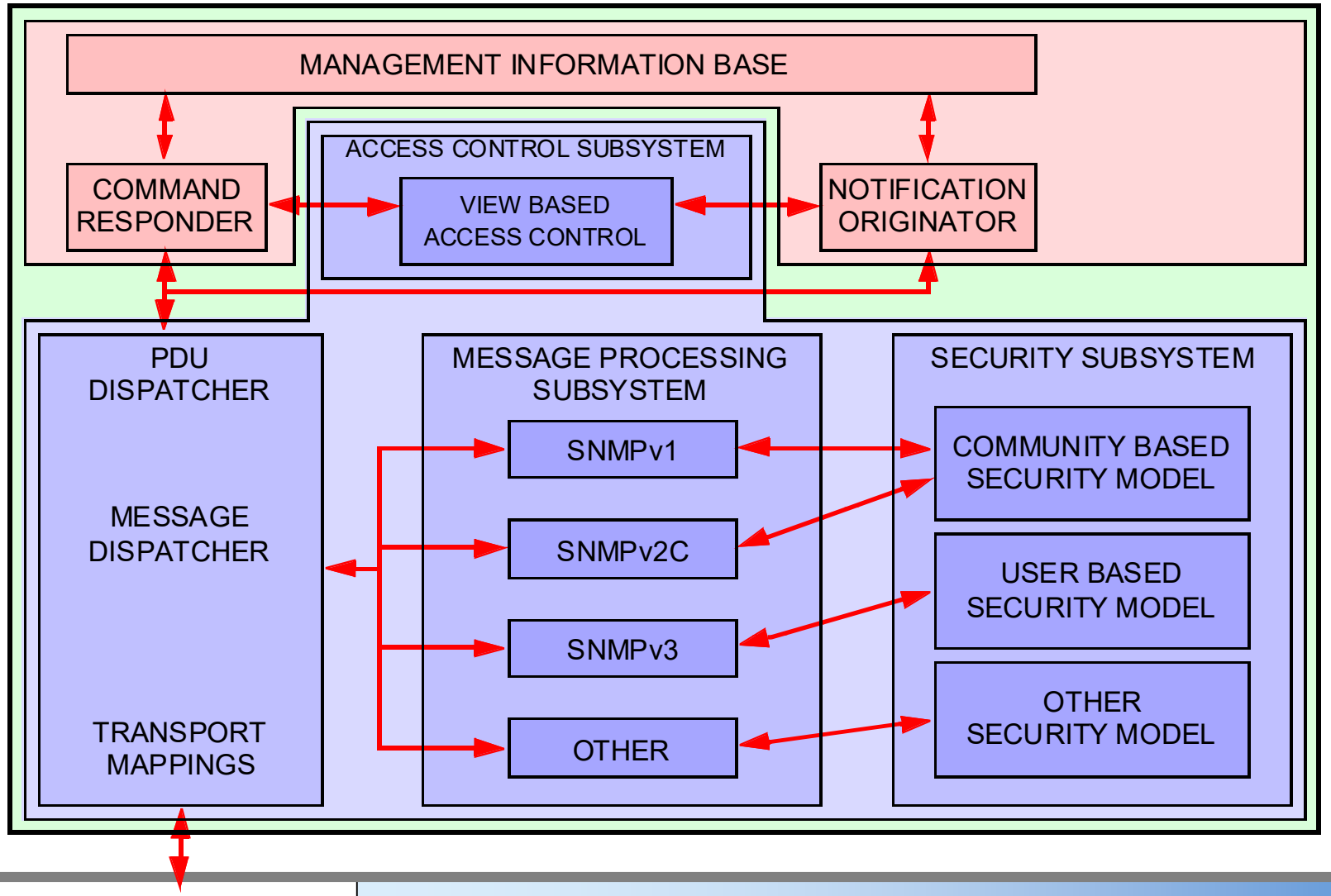
SNMPv3

- Has three security levels:
  - *Depending on how you connect – you get more access rights*
- *noAuthNoPriv*: Authentication by matching a user name.
- *authNoPriv*: Authentication with message digests.
- *authPriv:* Authentication with message digests, and encryption

SNMPv3 Manager



SNMPv3 Agent



What are we GET/SETting in those packets?

- Values stored in a Management Information Base (MIB)
  - Collected under a Structure for Management Information (SMI)
- Written in a formal language (ASN.1)
  - A formalism, rather than a language
- Field day for informaticians, logicians and other purists…

Table 24.1 *Data Types*

Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31}-1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from zero to 2^{32}; when it reaches its maximum value it wraps back to zero
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in 1/100ths of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted string

On Counters and Gauges…

- Reading Counters/Gauges tell you about "now"
  - **Counter** e.g. packets on an interface (can wrap)
  - **Gauge** e.g. memory/disk space (ranges between zero and <maximum>)

- Agents don't have history, and don't calculate rates/changes
  - Agents only have a temporary clock - Time since boot

- Managers have to ask more than once, and make assumptions
  - Counter doesn't change = World hasn't changed
  - Gauge doesn't change = World may have changed, or not, between requests

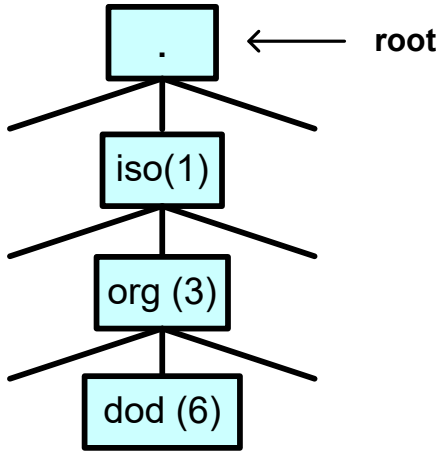
  - MIB designers might need multiple fields/types for related information

ASN.1

Know it exists and where to look it up…

- *Abstract Syntax Notation One (1980's) – predates XML, etc.*
- Formal description of data structures, message formats
  - Type, length, value (TLV)
- Predefined basic types
  - BOOLEAN, INTEGER, OCTET STRING, BIT STRING, REAL,
  - ENUMERATED, CHARACTER STRING, **OBJECT IDENTIFIER**
- Constructed types
  - SEQUENCE, SEQUENCE OF, CHOICE
  - Arbitrary nesting of types and sub-types
- Encoding types (10+) = *TLV to bytes* – we'll stick with 'BASIC'

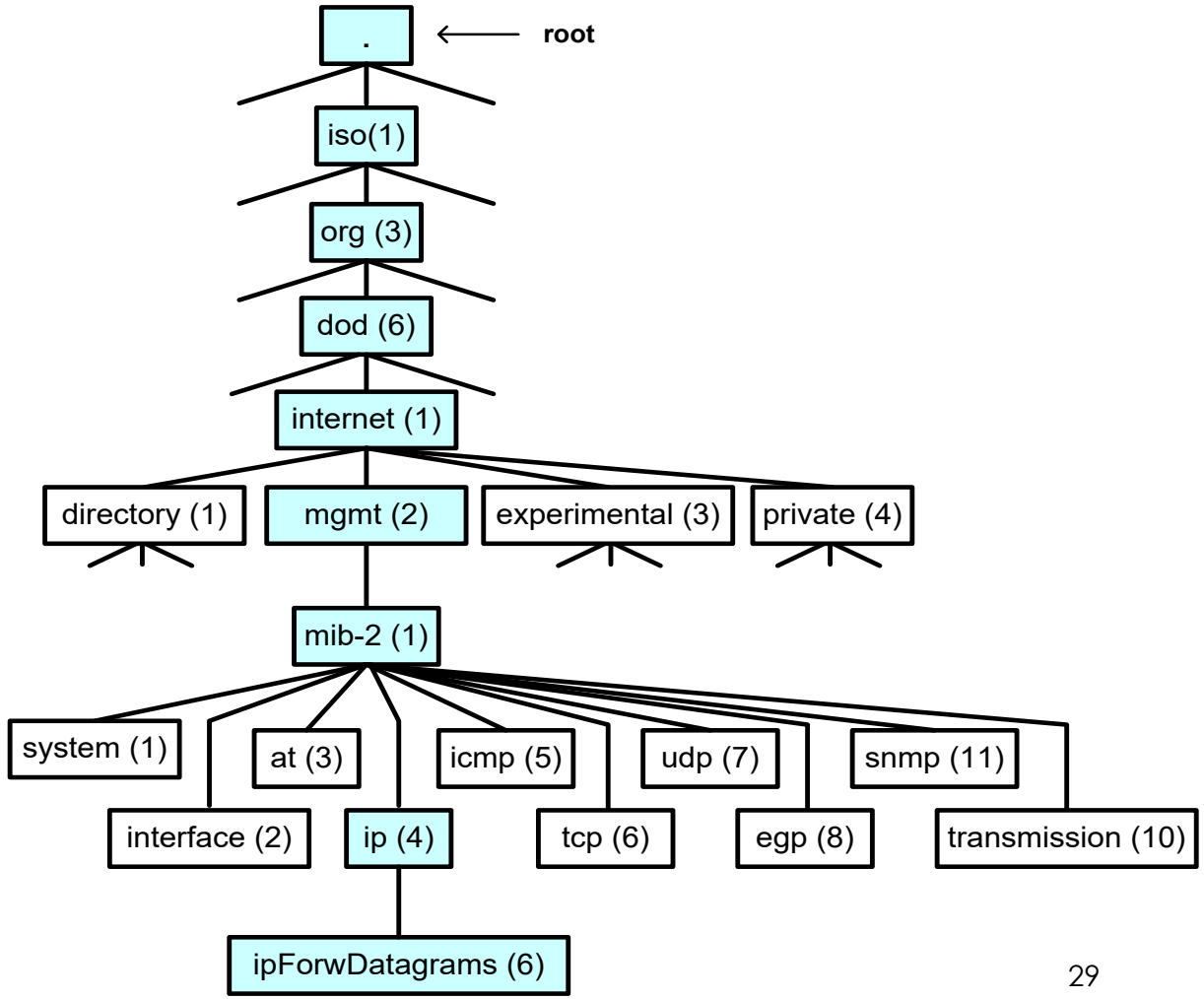
ASN.1 OBJECT IDENTIFIER (MIB)



- Define an information object and reference
- Managed at the international level
- `internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }`
- Globally unique

OID Organisation

- Tree hierarchy – like DNS
- Each OID is a node in the tree.
- Most internet stuff is 1.3.6.1.2.1.xyz
- Manufacturers can add product specific objects to the <private> hierarchy. 1.3.6.1.4.abc
- SNMP uses OID for reference
- MIBs map OID to readable form
  - And specify their type, etc.



ASN.1 examples

- **Type** definitions
  - `NumberofStudents ::= INTEGER`
  - `PassOrFail ::= BOOLEAN`
  - `GradeType ::= ENUMERATED {A, B, C, D, E, F}`
  - `PointsScored ::= REAL`
  - `Image ::= BIT STRING`
  - `Data ::= OCTET STRING`
- **Value** definitions and assignments
  - `studentsFridaySession NumberofStudents ::= 9`
  - `passCourse PassOrFail ::= TRUE`
- **Combine** type/value definitions

```
StudentType ::= INTEGER {
    ugrad  (0)
    ms     (1)
    phd    (2)
}
```

ASN.1 string examples

- Access ::= "read-only"
  | "read-write"
  | "write-only"
  | "not-accessible“
- Status ::= "mandatory“
  | “current"
  | "optional"
  | "obsolete"

A MIB "object"

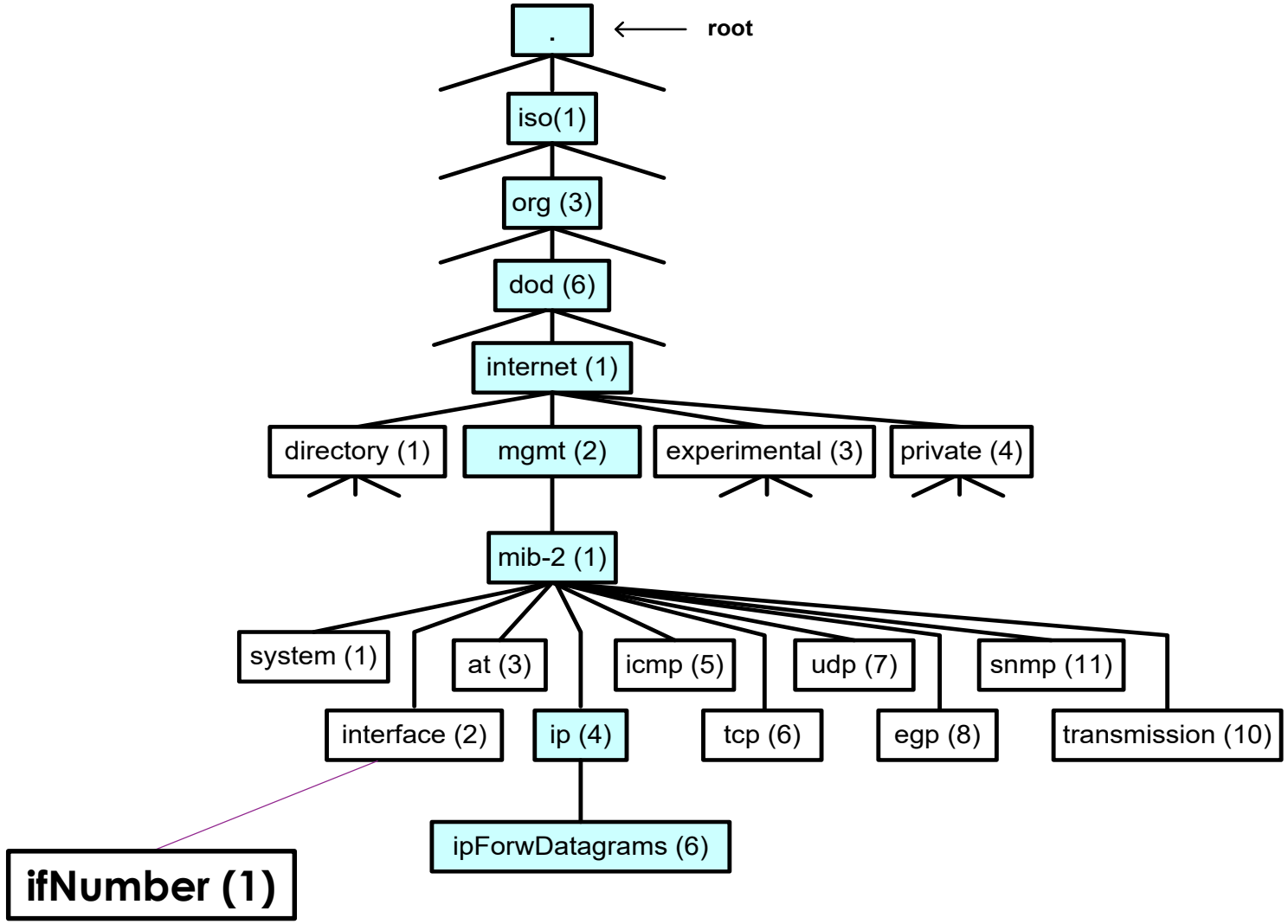
```
-- The Interfaces group

-- Implementation of the Interfaces group is mandatory for all systems.

 ifNumber OBJECT-TYPE
                SYNTAX  INTEGER
                ACCESS  read-only
                STATUS  mandatory
                DESCRIPTION
                        "The number of network interfaces (regardless of
                        their current state) present on this system."
                ::= { interfaces 1 }
```

Variable names are aliases for digit strings (defined by MIB)

***interfaces** defined in MIB as 1.3.6.1.2.1.2, so **ifNumber** = 1.3.6.1.2.1.2.1*


.
root
iso(1)
org (3)
dod (6)
internet (1)
directory (1)
mgmt (2)
experimental (3)
private (4)
mib-2 (1)
system (1)
at (3)
icmp (5)
udp (7)
snmp (11)
interface (2)
ip (4)
tcp (6)
egp (8)
transmission (10)
ipForwDatagrams (6)
ifNumber (1)

MIB-2 object counting packets

```
ipForwDatagrams OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  current
    DESCRIPTION
            "The number of input datagrams for which this
            entity was not their final IP destination, as a
            result of which an attempt was made to find a
            route to forward them to that final destination.
            In entities which do not act as IP Gateways, this
            counter will include only those packets which were
            Source-Routed via this entity, and the Source-
            Route option processing was successful."
    ::= { ip 6 }
```

Aka 1.3.6.1.2.1.4.6

More on the interfaces (MIB-II)

Name	Description
ifMTU	Maximum packet size
ifSpeed	Bits/sec
ifPhysAddress	e.g. MAC address
ifOperStatus	Up(1), Down(2), Testing(3)
ifInErrors	# incoming packets discarded due to errors
ifInDiscards	# incoming packets discarded due to buffer overflow
ifOutQLen	# packets in outbound queue
ifInUcastpkts	# incoming packets received

Why?

- OIDs provide global uniqueness – and extensibility
- OIDs provide human-readable-names for tree-position-identifiers
- Also: ASN.1 does not offer tables
  - But humans need them

Interface #	IP address	State	Packets	Errors	Rate
1	150.203.1.1	Up	1172	5	100Mb/s
2	130.56.3.1	Up	1234	3	100Mb/s
3	197.197.4.1	Down	5678	4	100Mb/s
4	197.197.5.1	Up	8451	197	1000Mb/s
5	8.8.8.1	Up	9191	2	10Mb/s

Tables and GetNext

- Each table cell has a *1.3.6.1.2.x.y.z.abc.label* identifier
- Rows in a table get sequential entries based on the index
  - E.g. Interface number

- Manager doesn't know how many rows (interfaces) there are
  - There is no 'row/column-count'. Don't need it. May change anyway!

- *Get ("Interface.1.ipAddress")* ➔ *interface.1.ipAddess = 150.203.1.1*
- *Get-next ("Interface.1.ipAddress")* ➔ *interface.**2**.ipAddress = 130.56.3.1*
- …
- *Get-next ("Interface.5.ipAddress")* ➔ *something else in the MIB*

- This works even if you don't know the names/columns/rows – Lexicographical Order for OIDs

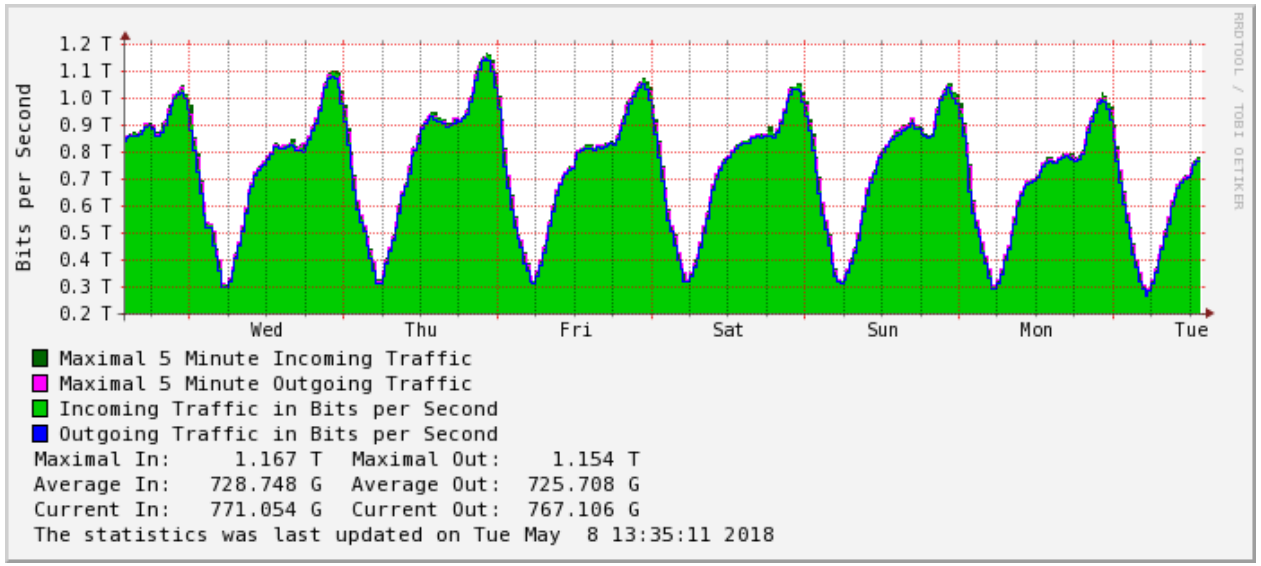
Ok, but…

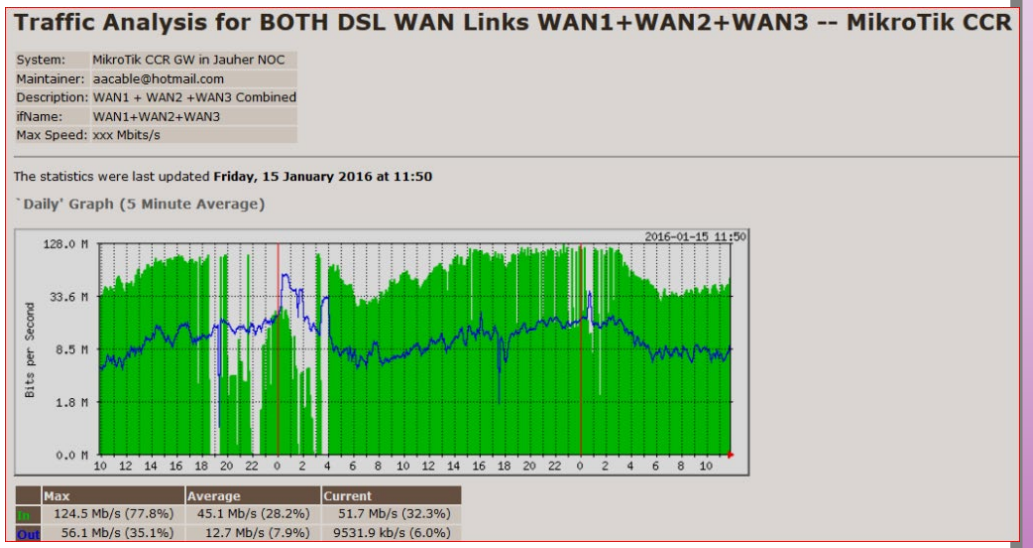
- Repeated Get-next:
  - Lots of extra there-and-back traffic
  - More state to maintain/evolve in Manager (row#/column#)

- SNMPv2 introduced Get-Bulk request
  - Get-Bulk("interface") ➔ every row, every column

  - But only one UDP packet comes back
  - Error response "tooBig" (64kB UDP limit)

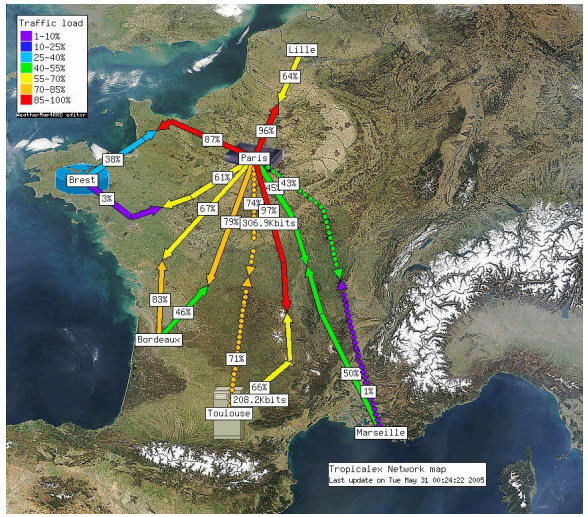
Review: Performance over time



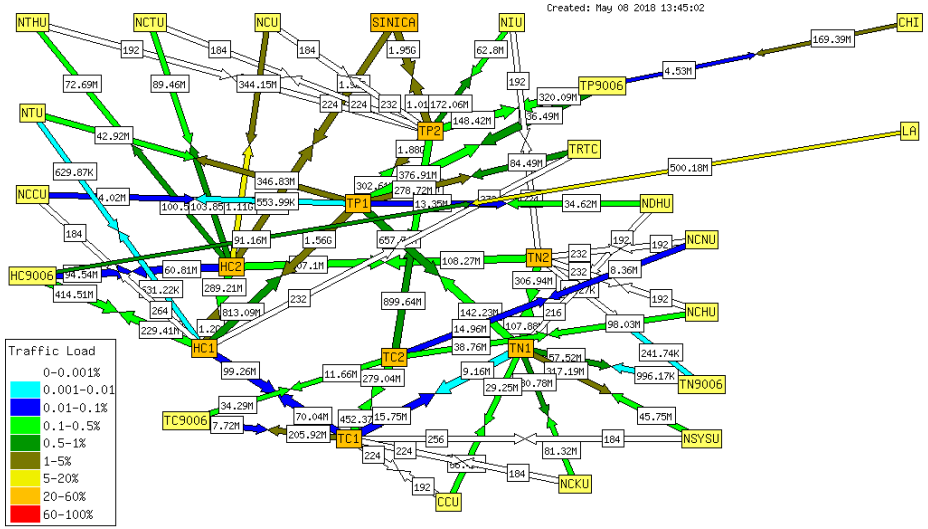


MRTG, Cacti, Nagios - as monitoring/graphing tool

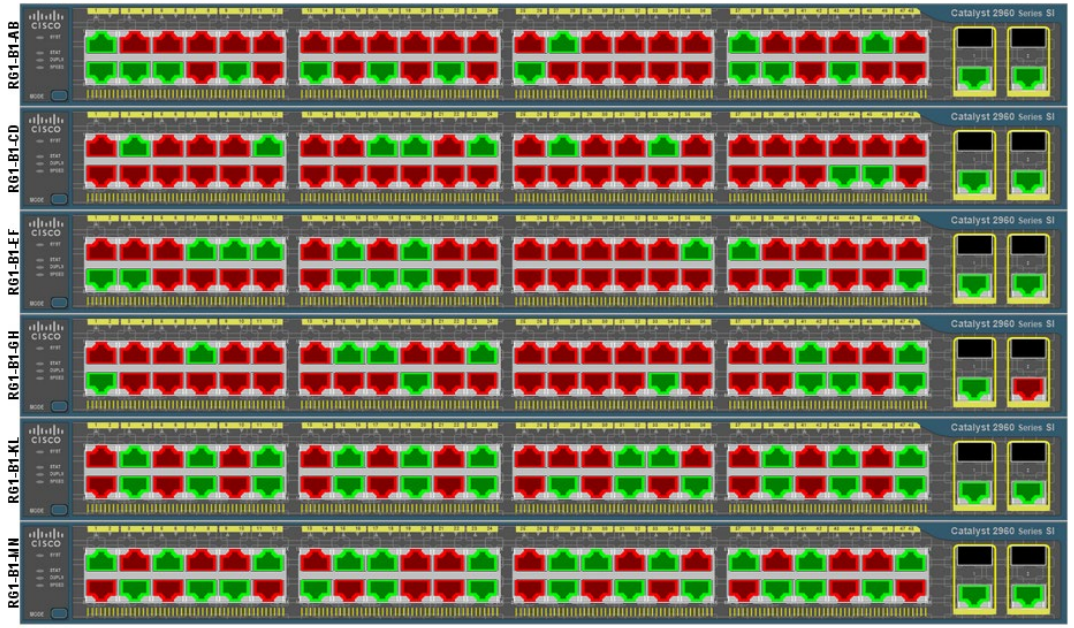
Network status



Network Weathermap
(PHP, reads MRTG data)



Vendor tool
(can draw layouts right)



SNMP beyond my domain?

- SNMP in the wide area is … unwise
  - SNMPv1/v2 agents should not be visible. Ever.
  - Lots of traffic
  - Easy to scan/map network

- Becomes a human problem
  - Need to ask for favours

- **Beacons** (e.g. multicast)
- **perfSONAR**
- **Looking Glass**
  - Remote login
  - Limited (read-only) queries
  - Various ISPs