RCOMP - Redes de Computadores (Computer Networks)

2025/2026

Lecture 11

- Network management.
- SNMP application protocol.
- Monitoring systems.

Network management

Network management activities are mostly network maintenance related, they include infrastructure devices installation, configuration and monitoring. Nevertheless, project and design are also included, both for the initial deployment, and also for the unavoidable upgrades. All these tasks are assigned to the network administrator role.

One particularity of network management is targets are spread over a more or less extensive area. It's true most network management activities are enforced in active devices, but most of them are dispersed along the network infrastructure.

Thankfully, most management activities can be carried out remotely by using the network infrastructure itself (as far as it's working). A wise approach is creating a separate VLAN, dedicated to management traffic, users should not be allowed to reach this network, it's basically a DMZ (demilitarized zone).

Depending on the network device itself, several application protocols are usually available to access and manage it:

- Remote terminal emulation protocols, e.g., Secure Shell (SSH) and Telnet.
- HTTP (access from a standard web browser to a device's HTTP server).
- Specific network management protocols, like for instance the Simple Network Management Protocol (SNMP).

Remote management

The most basic, and yet probably the most powerful remote management tool is undoubtedly the remote terminal emulation. It usually allows the same wide range of operations also available at the console. The flipside is it requires some expertise on command line interface. Also, some cheaper devices may simply lack this service.

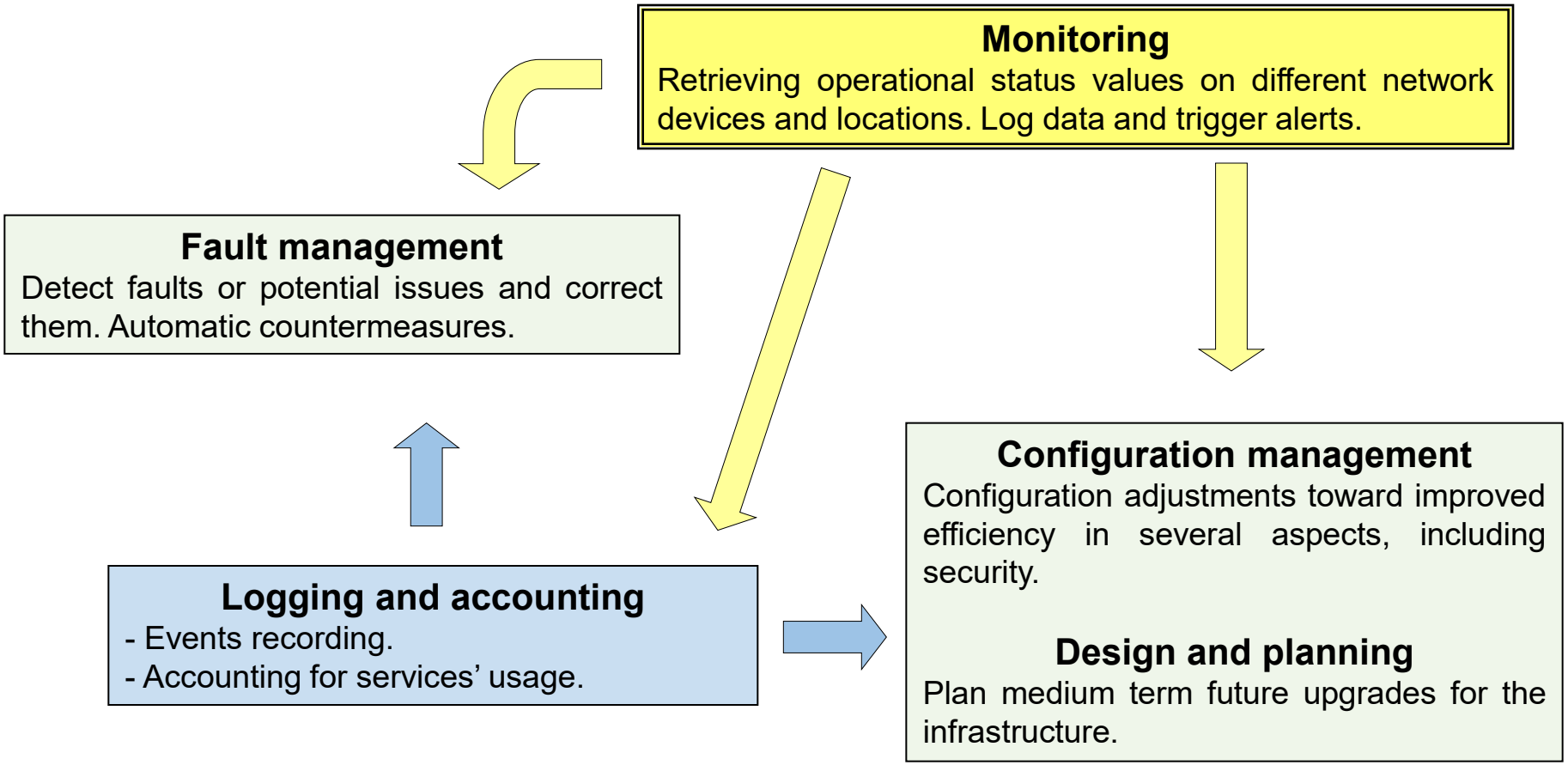
Most common remote terminal emulation application protocols are SSH and Telnet. Under usage point of view, they are fairly similar, though Telnet uses a simple TCP connection with no encryption, therefore, all data will be visible to sniffers, including used authentication credentials (passwords). Some cheaper devices may support only Telnet, otherwise, SSH should always be used instead.

Nowadays, many vendors also provide web interfaces to their devices, this means the device has a small HTTP server, and thus, a standard web browser can be used to manage it. This can be a complement to remote terminal access or the only available management access to the device. Bear in mind that, if the HTTP access is a complement to remote terminal access or console, then most often it will only have a small subset of the available options at the console.

Several network management protocols are also available; they have been developed with the specific purpose of remotely managing network devices.

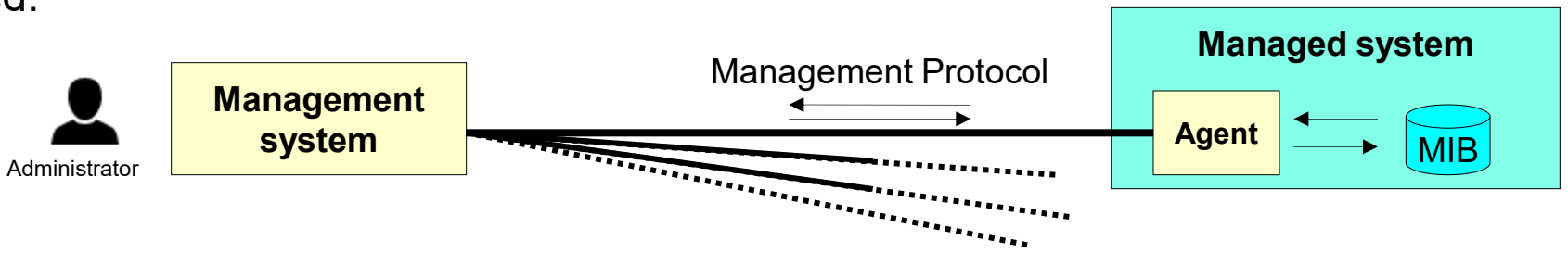
Network monitoring

Monitoring is one essential activity within network management. It may be easily automated by creating a central service that periodically check devices and services status.



Agent – Manager model

Most network management systems use a client-server model with two entity types involved:



The **management system** is the central control point from where requests are sent to **managed systems**. A **management protocol** is used for these client-server dialogs.

The **managed system** may be any kind of network device, like for instance, a hub, a switch, a router, a server, or a printer. Each holds information about its own status and configuration, this information is known as the MIB (Management Information Base).

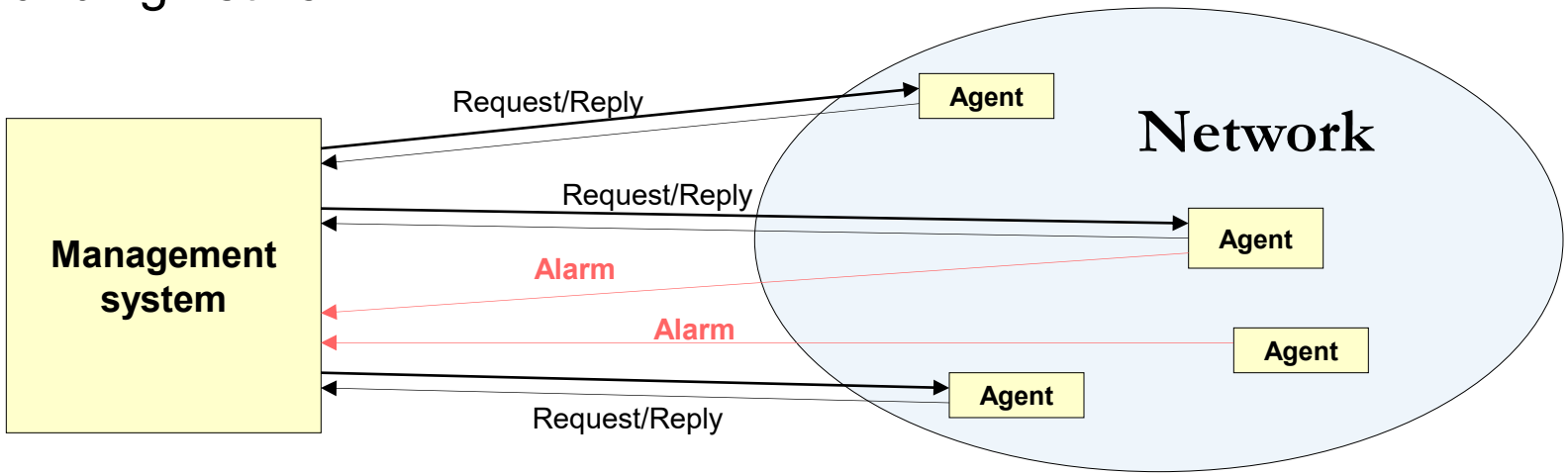
Both the **management system** and **managed systems** must use appropriate software to talk to each other using the same **management protocol**.

On the **managed system** side this software is called an **Agent**, it's essentially a network server, accordingly, the software running on the **management system** is essentially a client.
We must say essentially, because occasionally they may reverse roles and the Agent may then also act as a client.

The Management protocol

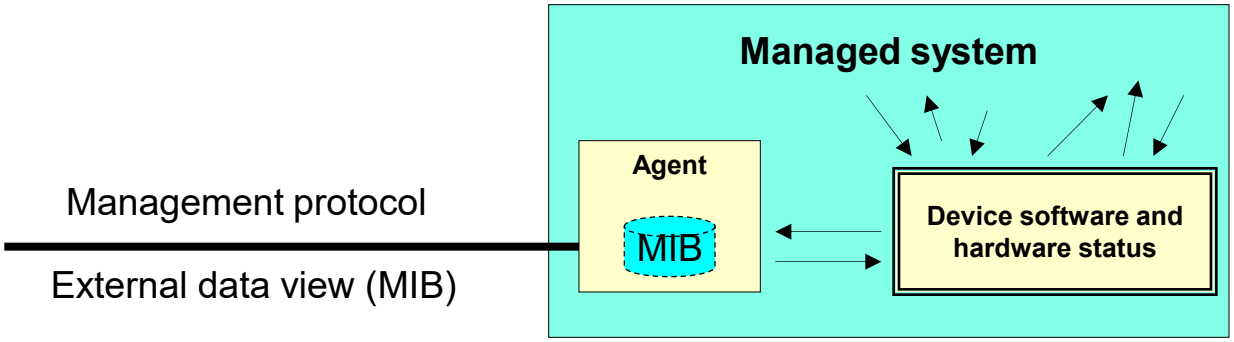
Most management protocols enforce two transaction types:

- Requests from the management system to the agent. In this case, the agent assumes a server role and replies to requests. Requests refer to management operations, either queries to the MIB, or change requests to the MIB (configuration changes).
- Alarms may be sent to the management system by an agent's initiative. Usually, they are alerts about events on the device itself or at the surrounding network.



MIB (Management Information Base)

The MIB is a data collection representing the device internal status and configuration.



Actually, the MIB is the management protocol's external view of the device's internal status. Agents create this view and interact with management systems accordingly, thus, for management systems, all agents have a standard MIB with a similar structure.

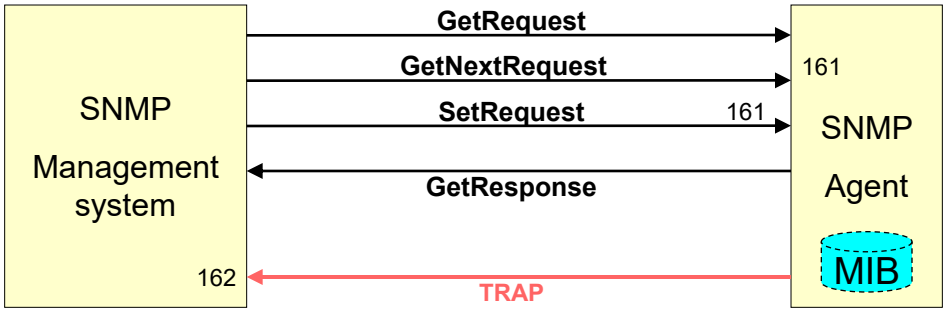
The MIB is usually objects oriented, how far the objects paradigm is taken depends on implementations. It can go from objects as simple data holders (variables) with no associated methods, up to true objects as class instantiations with a class structure and subclasses with inheritance.

SNMP (Simple Network Management Protocol)

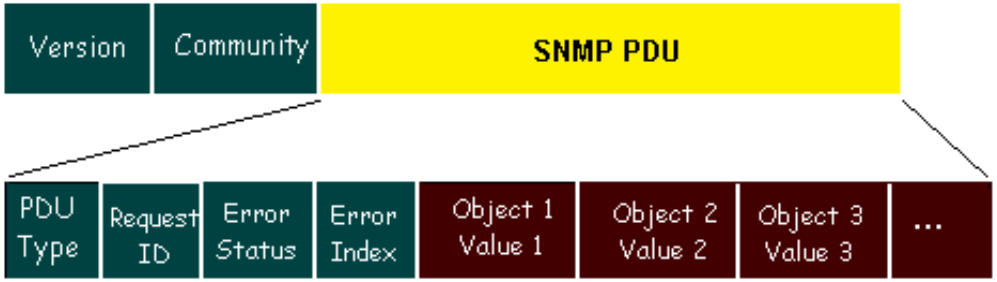
Whilst there are several other management protocols, like for instance CMIP (Common Management Information Protocol), currently the most widely supported management protocol is SNMP. In fact, many current networks devices support SNMP and no other protocols, so for such cases there's really no alternative.

Version 1 (SNMPv1) is still widely used due to the number of devices that don't support later versions. In SNMPv1 there are only five message types, they are encapsulated in UDP datagrams. Agents receive requests on port number 161, following the client-server model, and reply to them. Additionally, the management system may receive alarms (named **traps**) on port number 162.
In SNMPv1, MIB objects are simple variables. By using the **GetRequest** message the value of a variable/object is obtained through a **GetResponse** reply message.

The **SetRequest** message is used to change a variable/object value, it will be confirmed by a **GetResponse** reply message as well.
MIB objects are stored in a sequence, the **GetNextRequest** message allows a MIB sequential query. A full MIB dump can be accomplished with GetNextRequests starting in OID zero.



SNMPv1 - Messages



All SNMPv1 messages have the same general format, the **Version** field identifies the protocol version (zero stands for SNMPv1).
The **Community** field holds a string that can be used for simple access control.

The **PDU Type** identifies the message type:

0 - GetRequest
1 - GetNextRequest
2 - GetResponse
3 - SetRequest
4 - Trap

The **Request ID** field identifies a specific request, the value is copied to the reply, relating one with the other. Because SNMPv1 uses an unreliable transport (UDP) establishing this relation is rather important.

The **Error Status** field identifies the operation result:

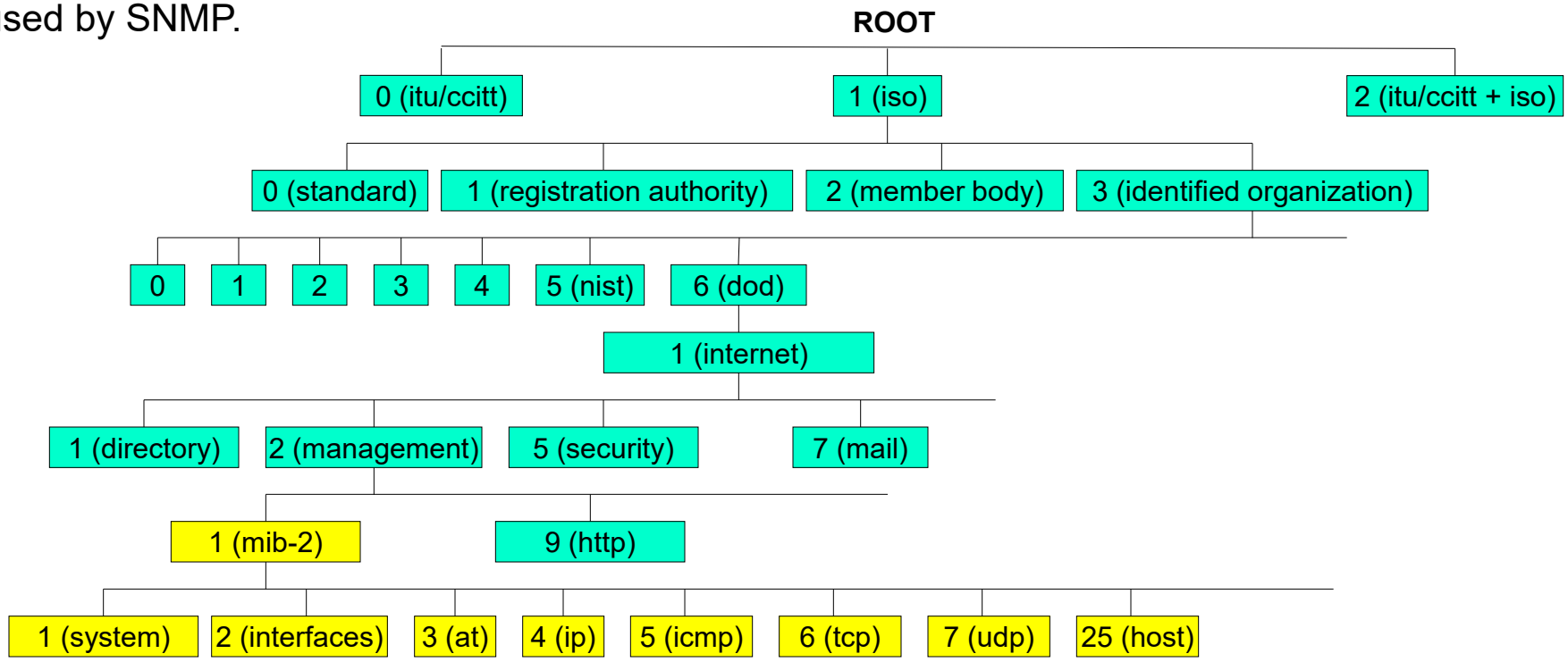
0 - noError
1 - tooBig
2 - noSuchName
3 - badValue
4 - readOnly
5 - genErr

In case of error, **Error Index** will point which object is associated with the occurred error. Messages types 0, 1, and 3 will always have zero on error fields.

Trap messages use a different distinct **SNMP PDU** format.

OID – Object Identifier

Objects (variables) stored in the SNMP MIB are not identified by names. The ASN.1 (Abstract Syntax Notation One) standard settles a universal naming system based on a numeric tree and it's used by SNMP.



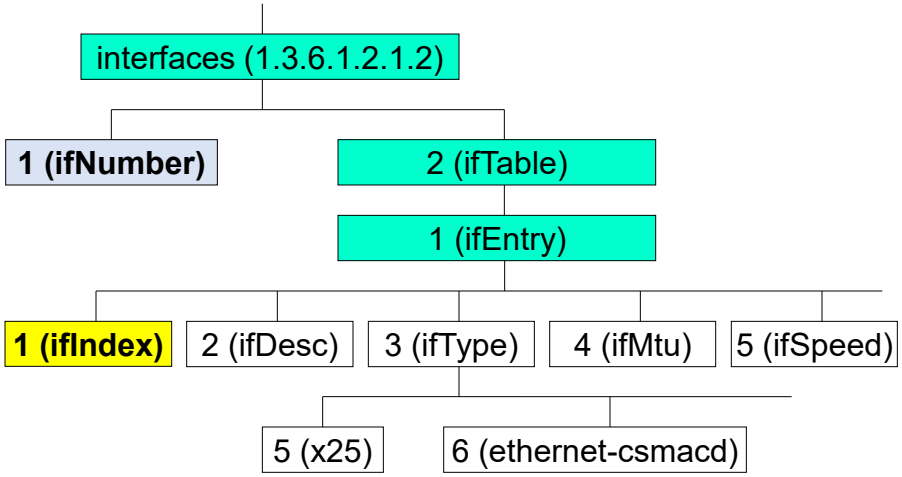
Each object class is identified by following the numbers sequence starting from the root until the branch is reached, for instance the **internet** object is identified by the **1.3.6.1** OID, thus all objects in subbranches will be started by that OID.

Objects related to network management are started by **1.3.6.1.2.1**, highlighted in yellow. An object class describes some features, namely the value type it can hold, if it's readable (get allowed), if it's writable (set allowed), and if it's mandatory or optional.

The case of **interfaces**, OID = 1.3.6.1.2.1.2

The interfaces branch (1.3.6.1.2.1.2), is mandatory, it holds information about the device's network interfaces. The **ifNumber** object has the number of existing interfaces.

The **ifTable** is a table of **ifEntry** objects (1.3.6.1.2.1.2.2.1), each one has an **ifIndex** for the interface number, interface numbers are unique and will go from 1 up to **ifNumber**.



SNMP uses OIDs, however an OID refers to an object class, not an instance. And of course, otften there can be several instances of the same class. SNMP adds one more number to identify instances, starting from zero. To get the number of interfaces in a network device the request must be for the first instance of **ifNumber** class: **GET 1.3.6.1.2.1.2.1.0**

For tables, SNMP adds the index object value to identify each tables element.
In the case of **ifTable** its **ifIndex**. The index value is then added to identify table's members, so the **1.3.6.1.2.1.2.2.1.2.8** OID means the eighth device interface description, it can also be represented by **ifDesc.8**.

Other table class objects are: **atTable**, **ipAddrTable**, **ipRoutingTable**, **tcpConnTable** and **egpNeighTable**. For any of them, there's an index object used to identify each table's element.

SNMP security

Security was rather neglected in SNMP v1 and SNMP v2c. Communities can be defined for read-only and read-write access, but there's no password, so the community name itself is often used as a secret password. Also, there's no encryption, so, the secret community name may be captured by any network sniffer.

SNMP v2p and SNMP v2u introduced some sophisticated security mechanisms, however, they were regarded as too complex and were never broadly deployed.

In SNMP v3 security was revised and symmetrical encryption with a shared secret key is used. The secret key can be generated from a user password, it may be used for authentication only, or both for authentication and encryption. The following security levels can be used:

- **noAuthNoPriv** – no username/password authentication, neither data encryption. Though, the username is checked.
- **authNoPriv** - username/password authentication, but no data encryption after authentication.
- **authPriv** - username/password authentication, and data encryption.

Both authentication and encryption are optional, but encryption requires authentication because the encryption key is established during the authentication.

Another approach to SNMP security issues

It must be said, security is not that solid in SNMPv3, authentication uses MD5 (Message Digest 5) or SHA (Secure Hash Algorithm) and encryption uses the old DES (Data Encryption Standard). Especially regarding encryption, DES with 56-bits keys is not really safe nowadays.

Because SNMP is deployed in a closely controlled network environment, traffic **access control** can be used to improve security. The best option is having an independent network dedicated to network management. It may be either a physically separated LAN, or a VLAN. As far as **no physical access to this network is possible, and all SNMP traffic is kept within it**, the protocol security issues become less relevant.

We must remember most agents run on limited capabilities dedicated hardware, this often means, we will have to use the SNMP version available for the hardware and not the one we would desire. There's a wide range of devices that support only SNMPv1.

Finally, security issues should be analysed under a risk and impact perspective. For instance, the security impact of read-only access to the MIB is highly questionable.

If SNMP is used merely for getting devices status, a read-only community is enough. By omitting read-write communities, significant exposure to security risk is avoided.

New SNMP messages

Even though SNMPv2 security features were not very popular, some new messages were introduced and later inherited by SNMPv3.

- **GetBulkRequest** – similar to **GetNextRequest**, however instead of requesting the next variable in the MIB, request for several next variables in a single request. It will be replied with a **GetResponse** message contenting several variables. Notice however that on SNMP v2, and above, the **GetResponse** message was renamed to simply **Response** message.
- **InformRequest** - similar to the **Trap** message, it's an asynchronous (unsolicited) notification sent by the agent to the management system. But unlike with Trap, there is a **Response** message from the management system. This response is obviously relevant if we remember SNMP operates over UDP, an unreliable protocol, thus this response is, above all, an acknowledgement.

One other innovation introduced in SNMP v2 is the concept that an SNMP v2 entity may undertake both the agent and the manager role. By playing both roles at the same time, a SNMP v2 and above agent, can now act as proxy agent.

Proxy agents are used to group together into a single view a number of local agents MIBs (including SNMP v1 agents).

MIBs and the device's MIB

The MIB is an objects' collection (OIDs), appropriate to represent devices' status and configuration, it encompasses all kind of devices. Thus, one device will only have some parts of the MIB that make sense for that device's features.

The objects that are made available by each device's agent depend on the device's features; this is called the **device's MIB**. Nevertheless, parts the MIB will make sense for most devices, for instance all devices have a number of network interfaces.

To analyse a device through SNMP, the device's MIB should be known in the first place, otherwise, one would be asking for objects that don't exist on the device. Of course, one can always start by OID zero and walk through the whole device's MIB.

There are thousands of different MIBs for many purposes, many of them are standard. Some are present in every device; others are vendor specific. The private branch (OID=1.3.6.1.4) is managed by IANA, for instance the enterprise branch (1.3.6.1.4.1) is used to assign object trees to hardware vendors for registering their own devices' special MIBs.

MIB-2 - RFC1213

MIB-2 (1.3.6.1.2.1) is standard for every SNMP agent. Among others, it defines the System branch (1.3.6.1.2.1.1):

1.3.6.1.2.1.1.1 – sysDescr	(string; read-only; mandatory)
1.3.6.1.2.1.1.2 – sysObjectID	(vendor OID; read-only; mandatory)
1.3.6.1.2.1.1.3 – sysUpTime	(integer; read-only; mandatory)
1.3.6.1.2.1.1.4 – sysContact	(string; read-write; mandatory)
1.3.6.1.2.1.1.5 – sysName	(string; read-write; mandatory)
1.3.6.1.2.1.1.6 – sysLocation	(string; read-write; mandatory)
1.3.6.1.2.1.1.7 – sysServices	(integer; read only; mandatory)

RMON – Remote Monitoring

Usually, each device's MIB represents the device's own status and configuration. However, that may not be always the case.

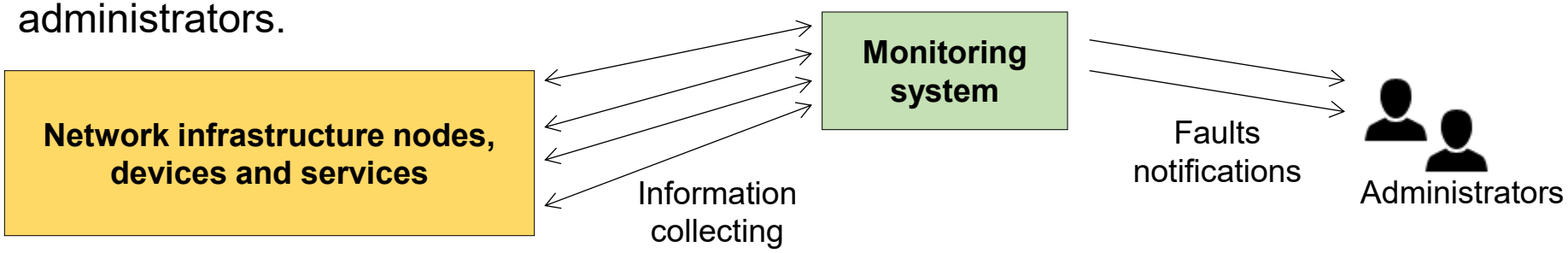
Monitoring devices may be placed at some network locations with the mission of collecting information about what is happening around them, this is called **remote monitoring**, and these devices are usually referred to as **probes**.

SNMP can be used to manage probes. Typically, probes collect statistical information about local network traffic and store that information in a specific database called **RMON MIB**.

Network and services monitoring systems

Even a rather small network infrastructure should be thoroughly monitored, this can be achieved by installing an automated centralized monitoring system.

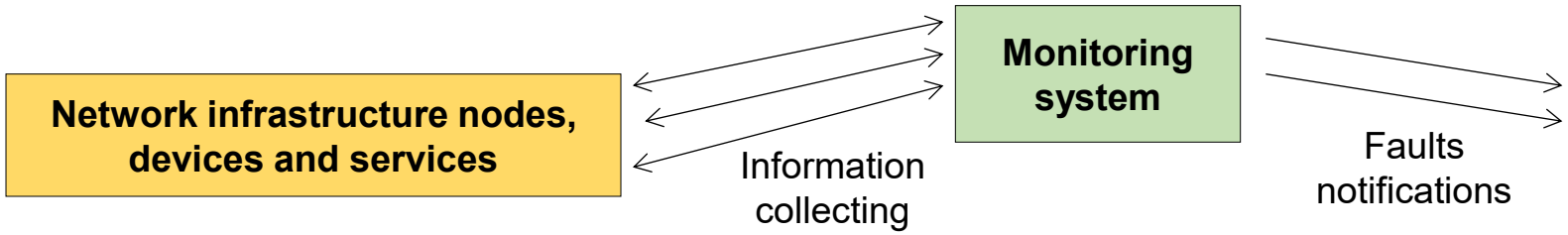
The monitoring system is a software service running in a central host and collecting information about the infrastructure's nodes and services status. The number one mission of the monitoring system is detecting faulty components and notify administrators.



Typically, the monitoring system has a dashboard where administrators can view the current status and manage the system, but ultimately notifications are crucial. Notifications are sent in real-time as soon as a faulty component is detected, they may be implemented through e-mail or preferably through **instant messaging**.

Even though the main goal is notifying in real-time administrators about faults, collected information along the time is most important and must be preserved. Several kinds of data analysis over collected information may be undertaken to extract very valuable conclusions, e.g. regarding usage trends and security issues.

Monitoring systems – status information collecting



The main activity for a monitoring system is information collecting; this is most often accomplished through **active checks**. With active checks, for each item to be checked, the monitoring system will periodically take an action to retrieve its status.

For network services, active checks are rather strait forward. The monitoring system **simulates a network service client**, performs a request to the network service to be tested, and checks if the result is as expected. In some extent, it's a unity test.

Take for instance monitoring an HTTP server. The monitoring system simulates an HTTP client, performs an HTTP request to the server, and checks if the response is as expected.

A network service response being as expected can take in account several aspects, including the result type (success, failure, no reply), the reply's content and the time it took for the service to reply. It's up to the monitoring system evaluating the response and classify in as faulty or not, or possibly something in between like a warning state.

Monitoring systems – status information collecting

Actively monitoring a network service is very strait forward. At a lower level, the same approach can be used to check if a network node is reachable. Because currently every network node uses IPv4 or IPv6, and ICMP is implemented together, the monitoring system can send an ICMP echo request to it, and check if a reply is received within some time (ping test).

In principle if a node fails to reply to an ICMP echo request, we can assume it's not operating (it is down). Therefore, further checking to network services on that node are pointless.

Often, we will need to check other items beyond network available services. This is the case of **nodes internal status items** like for instance the CPU load, free memory, disk usage, internal and environment temperatures, fan status, network interfaces traffic. To retrieve this kind of information specific protocols are required, and one possible option is SNMP.

SNMP may be the only option on simple network devices like switches and printers. For higher level devices, like servers, other options may be available, including specific monitoring software (agents) to be installed on the monitored device itself.

Monitoring systems – passive checking

Even though the active checking model is the most widely used, one alternative is passive checking.

Unlike with active checking, in passive checking it's not up to the monitoring system performing checks. Instead, it's up to the monitored device to periodically contacting the monitoring system and report its status.

Nevertheless, the monitoring system must detect if one monitored device has failed to report within the expected time period. It's rather similar to a heat beat.

A maximum report interval for each monitored item can be configured on the monitoring system, if no report is received within this time period the item is regarded as faulty.

One other option often used is the monitored item report including a TTL (Time To Live) field, if the report expires before being replaced by a new one, then the monitoring system deliberates the item is faulty.