Network Management Architectures

Aiko Pras

CTIT Ph. D-thesis series No. 95-02

P.O. Box 217 - 7500 AE Enschede - The Netherlands
telephone +31-53-893779 / fax +31-53-333815

Centre for
Telematics and
Information
Technology

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Pras, Aiko

Network Management Architectures / Aiko Pras
[S.1. : s.n.]. - Ill - (CTIT Ph. D-thesis series, ISSN 1381-3617; no. 95-02)
Thesis University of Twente, Enschede. - With ref.
ISBN 90-365-0728-6
Subject headings: distributed systems; management

NETWORK MANAGEMENT ARCHITECTURES

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. Th.J.A. Popma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 17 februari 1995 te 16:45 uur

door
Aiko Pras

geboren op 30 april 1956
te Zwolle

Dit proefschrift is goedgekeurd door de promotoren
prof. dr. ir. C.A. Vissers
prof. dr. ir. C. Bakker

Abstract

Network management is needed to control and optimize the operation of the network and to respond to changing user requirements. Management includes the initialization, monitoring and modification of the network functions. In order to perform management, special functions are needed. To distinguish these functions from the normal network functions, this thesis introduces the terms ‘management functions’ and ‘primary functions’.

Management functions may be performed explicitly by human operators, but also automatically by dedicated hard- and software modules. In case human operators are responsible for network management, most management functions will be performed from a limited number of remote locations. In case management functions are performed automatically, it is possible to distribute the hard- and software modules that implement these functions over the various systems in the network.

Architectures for network management enable the designers to discuss management functions at a high level of abstraction and guide the design of management protocols and services. In this thesis it is assumed that architectures consist of:

- a set of architectural *concepts,*
- *rules* that tell how to use these concepts,
- *models* that show the application of these rules and concepts to design a specific class of systems.

All current management architectures, notably the ISO, ITU-T (the former CCITT) and the IETF architectures, have been developed after the design of the network functions have been completed. Such approach indicates a specific conceptual view on the role of management functions and invites to apply different architectural concepts for the design of management functions. This thesis proposes an alternative approach, in which no principle distinction is made between the management requirements and the requirements of primary functions. Both sets of requirements can be integrated into one set of requirements and elaborated in a single design process, which uses one architectural model.

The thesis consists of two parts; Part I (Chapter 2 - Chapter 4) analyses the state of the art in network management architectures and Part II (Chapter 5 - Chapter 9) develops an alternative network management architecture.

Chapter 2 analyses the ISO management architecture, which is defined in the ‘Management Framework’ and the ‘Systems Management Overview’ standards. As compared to other network management architectures, the ISO architecture received most attention within the research community.

The management architecture of the ITU-T is known as the 'Telecommunications Management Network' (TMN), and is discussed in Chapter 3. The name of this architecture already indicates that this architecture is primarily intended for management of telecommunication (e.g. telephony) networks. TMN in fact consists of multiple smaller architectures:
- a functional architecture
- a physical architecture
- an information architecture, which includes many ideas of ISO management
- a logical layered architecture, which includes a responsibility model.

In 1988 the 'Simple Network Management Protocol' (SNMP) was defined by the IETF to meet the immediate management needs of the Internet. Internet management is analysed in Chapter 4; as opposed to the ISO and ITU-T the IETF did not define a separate architectural standard to describe the concepts behind SNMP. The reason for this is that these concepts resembled the ones that were already described in drafts of the OSI Management Framework and were considered to be obvious.
In 1992 the IETF started the development of a second version of SNMP (SNMPv2). Although the concepts behind SNMPv2 are more difficult to understand and so should be defined in a separate standard, such a definition has not been produced.

The identification of management functions is discussed in Chapter 5. To bring some order in the large number of management functions, special attention is given to the classification of these functions.

Chapter 6 explains how management functions can be designed together with primary functions. It also discusses that it may not always be possible to design all management functions before the start of the operational phase. This is not necessarily a problem, since the management functions that remain can be established during the operational phase by human operators. After the start of the operational phase the designer may decide to add the remaining management functions by developing new generations of network systems.

The alternative management architecture, which integrates primary as well as management functions, is developed in Chapter 7. To demonstrate that both kind of functions can be expressed in the architectural concepts and rules as used by the OSI Reference Model, examples will be given. Several models are developed to explain how management can be performed from one or more remote locations. These models show a number of management protocols as well as special service providers for the exchange of management information. Chapter 9 discusses the management protocols and makes a distinction between two basic types (Variable Oriented and Command Oriented). The service providers to support the exchange of management information are discussed in Chapter 8.

Acknowledgements

This thesis could not be written without the support of others. I am grateful for this support and I would like to thank the following people in particular.

First of all my gratitude goes to my supervisors Kees Bakker and Chris Vissers who gave me the opportunity to do this research; they provided me with many new ideas and detailed comments on the various drafts of this thesis.

Furthermore I would like to thank the members of the TIOS group, especially Marten van Sinderen, for the pleasant and stimulating working environment. I hope this good atmosphere will be retained, despite the gloomy prospects of reorganizations and economy measures.

An enjoyable complement to the more fundamental problems I had to investigate as part of this Ph.D. study, was the applied work I carried out as member of the UT-SNMP group. This work gave me to opportunity to discuss many issues with researchers within the Internet community and provided me with a better understanding of the real problems designers are faced with. I would like to thank Eric van Hengstum and Vincent Berkhout, as well as the many students who participated in the UT-SNMP project.

People often forget the importance of the technical support that is provided by the B&O group. This group keeps our workstations alive and provides us with links to the outside world. I would like to thank the members of this group, in particular Tonnie Tibben.

Thanks also go to DirkJan Speelman, who made the cover design of this thesis.

Finally I want to thank my family and especially my parents for giving me the opportunity to perform my study. Special gratitude goes to my wife Wilma, whose continuous support and understanding I needed to perform this work.

Contents

Part II: an Alternative Approach to Network Management

1: Introduction

1 Introduction

In the next decade an impressive growth is to be expected in the use of communication networks. To initialize and optimize the operations of these networks, good network management facilities must be developed. The importance of research in this area is confirmed by a number of studies that show the state of current networks. A study in the UK for example showed that LANs go down an average of twenty times a year and subsequently stay out of service for more than four hours [27]. A study in the US showed that every hour of LAN inoperability, 'Fortune 1000' companies loose more than $30,000 [103]. The nine hours breakdown of AT&T's long-distance telephone network in January 1990 even resulted in a $60 million to $75 million loss in AT&T's revenues [30]!

The purpose of this thesis is to improve the understanding of network management and to develop an alternative architecture that avoids the deficiencies of existing management architectures. It is assumed in this thesis that architectures consist of the following elements (Figure 1.1):

- A set of architectural *concepts* or conceptual building blocks. Examples of such concepts are: service provider, service user and Service Access Point (SAP).
- *Rules* that tell how to use these concepts. An example of such rule is that service users must interact with their underlying provider via SAPs.
- *Models* that show how these concepts and rules can be applied to guide the design of a specific class of systems. An example is the OSI Reference Model [43], which was developed to guide the design of computer networks.

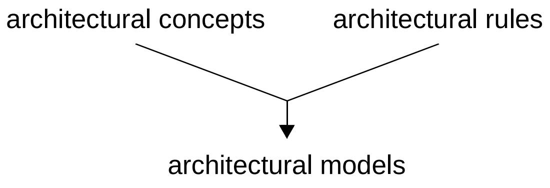


Figure 1.1: Elements of an architecture

1.1 What is management

In literature several definitions of network management exist [18][41][44][80]. Most of these definitions are produced by standardization organizations, which use specific terminology and aim their definitions at specific fields of application. For this reason, these definitions are not suitable for the scope of this thesis. Therefore this section starts with a definition of network management, as considered in this thesis.

> *Definition:* network management is the act of initializing, monitoring and modifying the operation of the primary network functions.

Primary network functions are those functions that directly support the user requirements. They allow for example users to access the network and they take care of the exchange of user data. During the design phase, the primary functions will be implemented and realized by the designer. How this is done, depends among others upon the skills and experiences of the designer.

Management is needed to bring and keep into operation the network systems that perform the primary functions.

This implies that management should first initialize the various network systems (configuration management). If no errors are made, the network comes into service and the operational phase starts. During this phase, management monitors the various network systems to check if no errors occur. In case of failures, malfunctioning systems will be identified, isolated and repaired (fault management). If systems can not be repaired, they will be replaced by new systems, which must be initialized too. New systems may also be added to allow the connection of new users, to increase performance or to add new functionality. Addition of new systems usually implies reconfiguration. Monitoring the network is also useful to detect changes in the traffic flow. Once such changes are detected, network parameters may be modified to optimize the network's performance (performance management).

To allow management actions to be performed during the operational phase, the designer should define a number of management functions. These functions should be added to the design and implemented and realized together with the primary functions.

An important idea of this thesis is that no *principal* difference exists between the design of primary and the design of management functions. In part II this idea will be elaborated and it will be demonstrated that both kinds of functions can in fact be included in a single architecture.

1.2 Why is management needed

It is interesting to see that the literature usually focuses on '*what* management functions can be identified' and that little has been published with respect to the question '*why* management functions must be performed'. This section discusses this last question and identifies reasons why network management is needed. It reinforces the view that management should not be considered as a set of functions that can immediately be derived from the user requirements.

Cost reduction

Users obviously want the best possible network at the lowest possible price. A way to satisfy this requirement, is to spread the costs of the design over a large number of users. This implies that the design should not be tailored to the specific requirements of a single user group, but be general enough to accommodate the requirements of many potential users (Figure 1.2). The design should thus be a multi purpose design, which means that it should be possible to use mass production techniques.

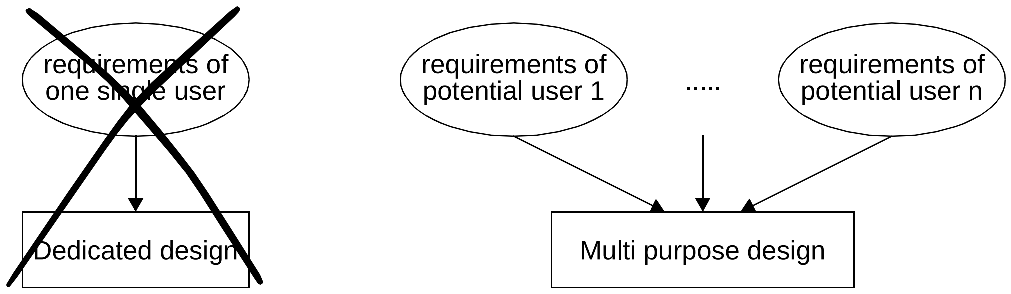

requirements of one single user
Dedicated design
requirements of potential user 1
.....
requirements of potential user n
Multi purpose design

Figure 1.2: Design should not be customized, but general purpose

A way for the designer to deal with the requirements of multiple groups of users, is to abstract from the differences in these requirements and parameterize the design. To allow the network to become operational, the parameters must be initialized to some user specific value. This initialization is the responsibility of management.

> *Example:* Some users want to have a network that spans the entire world, while others want a network that covers a local area. Assume that all users want their networks to be of the packet switched type, and require that every packet will be delivered. To meet this requirement, the designer may decide that after the reception of a packet the receiver should issue an acknowledgement to inform the sender. If the sender does not receive the acknowledgement in time, it will assume that the packet (or the acknowledgement) got lost and the packet will be retransmitted. The time the sender is prepared to wait for the acknowledgement, should be more than the round-trip delay. This delay is much higher in a world-wide network than in a local area network. To produce a multi purpose design, the designer should abstract from this difference and include a management function. This function should arrange that a special *time-out parameter* is set to a high value in case of the world-wide network and a low value in case of the local area network.

Lack of experience

There are rapid developments in the area of networks. In a short period of time both the capabilities of networks as well as their use have increased considerably. As a result the designer will be faced with a number of problems. Because the designer's experience is limited, it is unrealistic to expect that it will always be possible to find good solutions for each and every problem during the design phase. For some problems it may therefore be a good idea to postpone the search for solutions until the operational phase has started; solving such problems will than be the responsibility of management. The advantage of this approach is that it may be expected that during the operational phase additional experience will be obtained, which helps to solve these problems.

Example: Congestion control is a problem that has not yet been solved in a general way. This is due to the fact that there are many different causes for congestion; each cause requiring its own measures. In this example three possible causes will be discussed, including the measures that must be taken to solve each of them (Figure 1.3).

① In a TV-show the viewers are invited to call the studio. This may result in an overload of the telephone network, in which case measures must be taken by management. A strategy to follow could be to restrict the number of call attempts to the studio. This could be implemented by telling *all* switches to accept only a small number of call attempts which have the studio as destination. This measure reduces the amount of prospectless signalling information and allows call attempts to other destinations to proceed.

② Assume a traffic jam develops after an accident has occurred on the highway. In such case it is likely that many car drivers decide to use their mobile telephones to call their homes. The processing of all simultaneous call attempts may overload the network's signalling system; without special measures the switch where the mobile calls enter the network may try to process all call attempts and, as a result, none of them may succeed. This is undesirable: it would be better to tell the switch to accept only a limited number of call attempts. As opposed to the previous case, call attempts will be refused irrespective of their destinations and a single or only a small number of switches will be affected.

③ There is carnival in Rio. Many people stay in the city and the telephone network is heavily loaded. To prevent the Rio area from getting overloaded, calls between other cities which are usually routed through Rio, will now be rerouted around Rio.

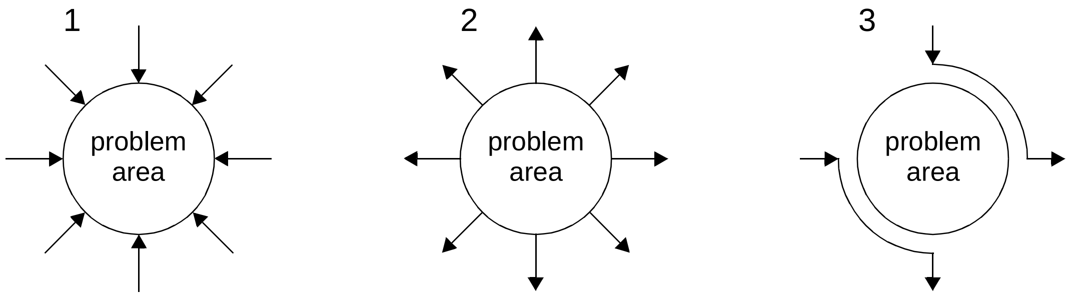


Figure 1.3: Three examples of congestion

These examples showed that it may not always be possible to anticipate all problems during the design phase. Some of the problems should therefore be solved during the operational phase by management. For this purpose the designer should include some general support functions that allow a manager to monitor what is going on in the network, to set alarms, to modify information in remote systems etc.

Fault handling

During a network's operational phase failures can occur suddenly. Failures are situations in which network components (or systems) do not behave in the way that has been specified. As a result of failures, networks may no longer provide the required service and it may even come to a complete breakdown.

The occurrence of failures can be due to ageing and decay of network components (hardware), as well as to human errors (e.g. a dragline that accidentally breaks a cable). The probability that failures occur, depends on the:

- quality of the network components: For a given price, components from certain manufacturers will have lower failure probabilities than components from other manufacturers. Still no manufacturer will be able to built network components that will never fail. No manufacturer will therefore be able to completely satisfy all user requirements.
- way of working: In many cases human errors are the result of unfamiliarity with local circumstances or not following the rules. Although many network failures may be caused by human errors, investigating the origins of these errors is outside the scope of this thesis.

Since it is not possible to prevent all failures and since failures can have severe consequences, the operation of a network should be controlled during the operational phase by management. Such controlling involves the *prediction* of potential failures, the *detection* of existing failures, the *reduction* of the effects of failures and off course their *repair*.

To predict and detect failures, managers should be able to:

- monitor the current behaviour of the network components.
- compare the current behaviour with previous and / or expected behaviour.
- signal exceptional behaviour.

To reduce the effects of failures and to allow reparation, management must have the means to change the state of the network. This may be accomplished by changing network parameters, such as the entries of a forwarding table.

Flexibility

Network designs are commonly described as top-down processes. Characteristic for such processes is the important role of user requirements; the design usually starts with the definition of the user requirements and many design decisions follow from these requirements. The outcome of the design process (the network) is thus primarily determined by the user requirements (Figure 1.4).

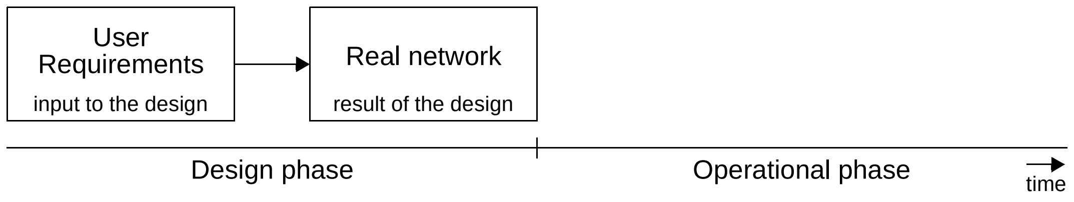


Figure 1.4: Simplified top-down design process

The danger of looking at the design process in this way, is that one may neglect the dynamic nature of the user requirements and consider these requirements as static entities. In reality user requirements change in time and should therefore not be considered as static entities (Figure 1.5).

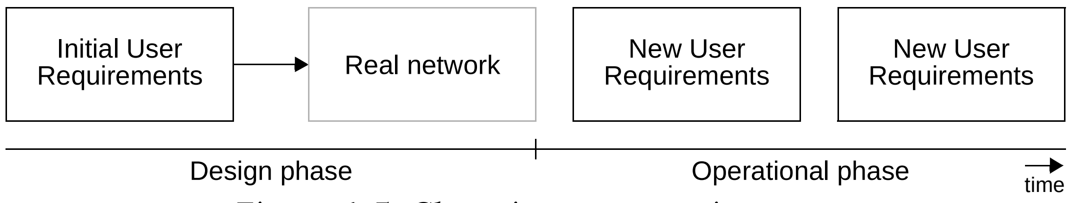


Figure 1.5: Changing user requirements

> *Example:* The user requirements may initially describe that there are only a limited number of users who want to be connected to the network. After the network becomes operational, it may be that others become interested in the network too. As a result, the initial requirements will be changed to accommodate the connection of more users. After some time, it may also be that the users require from the network new kind of services (to support for instance multi-media). Again the user requirements will be changed.

Instead of ordering a new network each time the user requirements change, it is better to built some flexibility into the network. Because of this flexibility the network manager will be able to react during the operational phase upon changes in the user requirements. The designer should anticipate this need and add a number of management functions to the design. Management issues should already be considered during the design phase!

1.3 How is management performed

While designing management functions, the designer will be confronted with a number of design questions. Two of these questions are particularly important because they affect the design process to a considerable extent. These questions are:

- Will management functions be performed by human beings, or will they completely be performed by hard- and software modules?
- Should management functionality be distributed over the entire network, or should it be concentrated as far as possible?

Both questions will be discussed in the following two subsections.

1.3.1 Explicit and implicit management

To denote the case in which human beings are responsible for the initiation of management operations, the term *'explicit management'* will be used. With this form of management, the decision to initiate management functions will explicitly be taken by (human) *operators* during the operational phase.

It should be noted that even this form of management requires the inclusion of a number of management functions during the design. The purpose of these functions is to support the operator while performing his task (see example below).

The opposite of explicit management will be called *'implicit management'*. With this form of management, all management functions will be performed by hard- and software modules; operator intervention is therefore not needed.

> *Example:* On page 3 of this thesis the use of a time-out parameter in a retransmission mechanism was explained. During the design phase, the designer should decide whether this parameter will be set by a human being (explicit management), or by some hard- or software module (implicit management).
> In case the parameter will be set by a human being, the designer should include for example user interface functions that allow the human operator to access this parameter. Such user interface functions may require the introduction of a keyboard and a display.
> In case of implicit management, the designer should include some kind of function that automatically determines the parameter's value. This function may for example measure the average transfer delay and use the result to calculate the best value for the time-out parameter.

An advantage of explicit management is that it is not necessary to elaborate all management functions during the design phase. This is particularly true for the functions that determine at which moment a particular management operation should be initiated and which values should be selected to achieve a specific goal (such functions may be considered as the management 'intelligence' or the management 'decision process').
As a result of this, the design process will be less complicated and requires less time as would be the case with implicit management. Explicit management is particularly useful for solving the unexpected problems that show up during the operational phase and require the invention of novel solutions; explicit management is thus well suited when it comes to fault management.

Since explicit management will be performed by human beings, response time may be poor if compared to implicit management. Other disadvantages of explicit management are its limited capacity and the potential high number of errors.

If we compare the costs associated with both forms of management, we can conclude that in case of explicit management the management functions that are elaborated *during the design phase* will be less complex and therefore *less expensive*. On the other hand, explicit management requires human intervention during the operational phase, thus explicit management will be *more expensive during the operational phase*.

It should be noted that the distinction between both types of management is primarily a matter of realization; in principle it is possible to perform the same kind of functions with both types of management. With the advent of Artificial Intelligence (AI) and expert systems, the distinction between both types diminishes. Real world examples usually show a combination of both forms: some management problems are solved via implicit, while others require the use of explicit management.

1.3.2 Centralized and distributed management

In this thesis the term ‘centralized management’ is used to denote the case in which management decisions will be taken from a limited number of central locations. The management functionality that takes these decisions is called the *manager*; it represents what can be considered as the management intelligence and is sometimes referred to as the management ‘application’.

To manage the operation of the primary functions, *agents* should be added to the systems that perform primary functions. Such agents represent the management support functionality through which manager(s) initialize, monitor and modify the behaviour of the primary functions. As compared to managers, agents are usually simple.

With centralized management, a large number of managed systems can be controlled by a single managing system (Figure 1.6). To allow managers to communicate with their agents, a management information protocol is necessary. Examples of such protocols are CMIP and SNMP (both will be discussed in subsequent chapters).

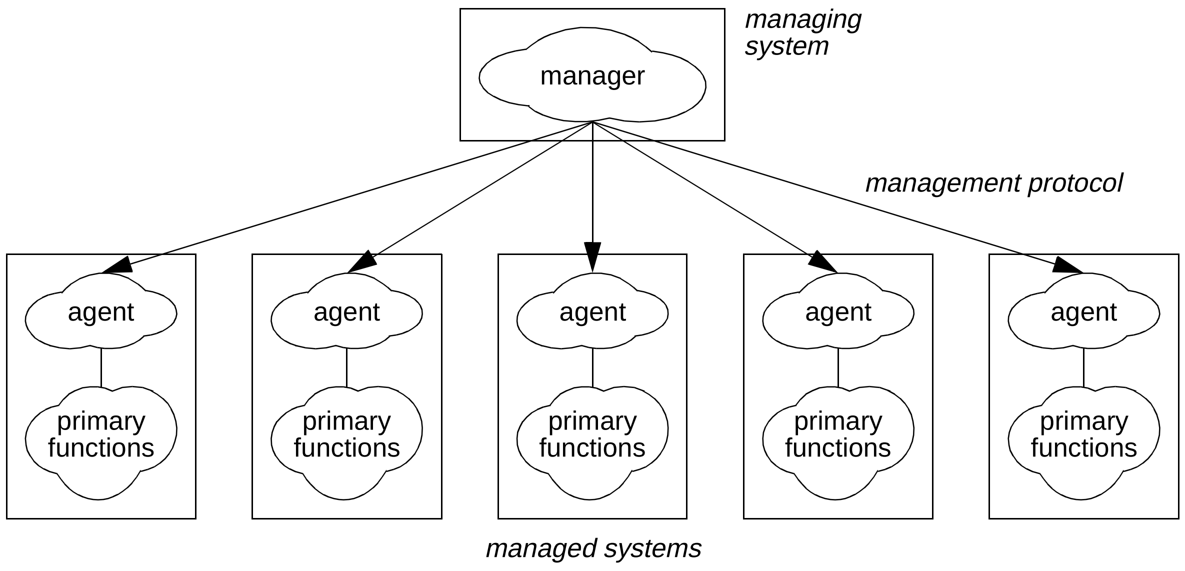


Figure 1.6: Centralized management

> *Example:* forwarding tables are used by the primary functions within the managed systems to determine the path that packets should take to reach their destination. The contents of these tables may be determined by a central manager, who calculates new values periodically or after a change in network topology. For large networks such calculations may be expensive in terms of CPU time and computer memory. After creation of new tables, the manager down loads these tables to the various managed systems.

The term 'distributed management' will be used in this thesis as the opposite of centralized management. With distributed management there are no central systems from which management decisions are taken. Instead, functions that take such decisions will be added to the systems that already perform the primary functions. Such addition will usually be performed on a proportional scale, which means that all systems that perform the same kind of primary functions get equivalent management functions.

> *Example:* with the arrival of powerful and cheap computer components, it has become possible for normal network systems to calculate their own forwarding tables. As a consequence there is no longer a need to bother central managers; management of forwarding tables can be performed in a distributed fashion.
> To perform this task, management information must be exchanged between the various network systems. A number of standardization organizations have already defined special routing protocols for this purpose; Figure 1.7 shows some of these protocols.

Number	Title
ISO 9542	ES-IS routing exchange protocol for use in conjunction with ISO 8473
ISO 10589	IS-IS intra-domain routing exchange protocol for use in conjunction with ISO 8473
ISO 10747	IS-IS inter-domain routing exchange protocol for use in conjunction with ISO 8473
ISO 10030	ES-IS routing exchange protocol for use in conjunction with ISO 8878
RFC 1058	Routing Information Protocol (RIP)
RFC 1267	Border Gateway Protocol (BGP)
RFC 1583	Open Shortest Path First (OSPF)

Figure 1.7: Some important routing protocols

Characteristic for distributed management is that each system takes its own management decisions. Because of the potential large number of systems, it will virtually be impossible to let human beings take these decisions. Distributed management must thus be realized in an implicit way.

A disadvantage of distributed management is that it will be difficult to change after the operational phase has started the functionality that makes the management decisions. This is because such changes require the modification of a large number of network systems, which will be expensive. In case the designer has little experience with a certain management solution, it may therefore be better to use the centralized management approach and concentrate the management functionality that makes the decisions within a single system. The motivation to use centralized management may thus be the same as the motivation to introduce Intelligent Networks (IN).

As opposed to distributed management, centralized management can be realized in an implicit as well as an explicit way. A disadvantage of centralized management is that the entire network may get out of control after failure of a single manager. Compared to distributed management, centralized management may also be less efficient: it is likely that more management information needs to be exchanged and the central managers may become performance bottlenecks.

With some management problems, for instance in case integrity or fairness come into play, it may be better to rely upon centralized management. The determination of system priorities and token holding times, for example, can be better performed by an independent system and not by the systems to which the decisions apply.

1.3.3 Concluding remarks

In this section the following two design questions were discussed:
- should management be performed in an implicit, or an explicit way?
- should management functionality be distributed on a proportional scale over all network systems, or should most management functionality be concentrated within one or more central systems?

Both questions are important, but not specific for the design of management: in principle it is also possible to realize primary functions in an explicit way or to concentrate major parts of the primary functionality within a small number of central systems. The fact that functions are performed in an explicit way or the fact that functions are concentrated within a few number of systems, does not necessarily imply that these functions should be considered as management functions.

> *Example:* An example of a primary function is switching. In the early days of telephony, switching was explicitly performed by human beings. Nowadays switching is implicitly realized by hard and software components.

> *Example:* Controlling the access to a shared medium (e.g. in case of a LAN) may be considered as a primary function. An old form of access control is polling, which is based upon a single master serving many slaves. The slaves are polled by the master to determine if they have data ready for transmission. The slaves are only allowed to start their

actual data transfer after access is granted by the master.
Current medium access control mechanisms have abandoned this centralized approach and use distributed approaches instead.

A second remark to be made is that during the design and the operational phase different views of network management may exist. In this thesis the emphasis will be on the design process, and the definition of management as given on page 1 will be applicable. After the operational phase has been entered, it may be difficult however for network users and operators to distinguish between the primary functions and those management functions that are performed in an implicit and distributed way. For this reason several people restrict their view of management to only those functions that are performed from a central location in an explicit way.

1.4 Open questions and contribution of this thesis

During the last decade several organizations recognized the need for network management and developed architectures to guide the design of network management services and protocols. Although these architectures proved their applicability in many cases, they still suffer from a number of problems. In this section some of these problems are identified; the emphasis will be on those problems that will be tackled in Part II of this thesis.

A first problem with current management architectures, is that they are not always properly defined. Some architectures are not even documented, which means that only an intuitive understanding of such architectures can be obtained. Other architectures have been documented, but the definition of their management concepts lacks precision. Finally there are architectures in which the concepts are well defined, but the application of these concepts in the associated management models is done in an inconsistent way.
As a consequence, progression of management standardization is sometimes slow and implementors may not always obtain a sufficient understanding of these standards.

In some management architectures the implicit assumption seems to be, that functions that are being managed can also be used for the transfer of management information. With such architectures, problems may occur in case the functions that are managed cease correct operation. In such cases it is conceivable that the exchange of management information might also be interrupted. As a result, management may no longer be able to reach the functions that should be controlled and it becomes impossible to restore proper operation.

A large number of managed objects have already been defined by the various groups that are active in the area of management standardization. Unfortunately, not all management architectures have paid attention to the classification of these objects. In case the intervention by management is required, the manager has to select the appropriate managed object from an unstructured list containing thousands of managed objects. The problem of this is that the manager may not have sufficient experience to determine which object should be selected.

Finally it is remarkable that all standardization organizations consider network management as something special that can be tackled as a separate design process after all primary functions have been developed. Although this approach has certain advantages (e.g. separation of concerns), it also has disadvantages. One major disadvantage is that it will be difficult to comprehend the relationship between primary and management functions; a clear view of *which* primary functions require *which* kind of management functions may not easily be obtained.

If we consider the management products that have been developed thus far, it is apparent that most of these products can be seen as general purpose building blocks that can not immediately be used to solve particular management problems. To obtain practical solutions for real management problems, these general purpose building blocks should be enhanced with ‘intelligent’ functions that tell how to apply these building blocks in solving actual management problems. To develop such functions, the designer must understand the relationship between primary and management functions. Until now, little progress has been made in the understanding of this relationship and the development of ‘intelligent’ functions.

A plausible explanation for this problem, which has also been described in literature [101] and discussed on a number of network management mailing lists on the Internet, is that management is investigated in isolation from the primary functions that are the subject of management.

Strategy to solve these problems

Simple measures that solve all of the above mentioned problems are difficult to find (and probably do not exist). This thesis therefore proposes an alternative approach, in which the designer considers the *complete* set of requirements from the outset in an *integrated* way. The principle idea that is put forward in this thesis is that *no difference exists between the design of primary and the design of management functions* (this idea is elaborated in Chapter 5 and Chapter 6 of this thesis). As an implication it should be possible to model primary as well as management functions as part of the same, integrated architecture (Figure 1.8).

How such an integrated architectural model can be developed, is explained in Chapter 7 of this thesis. The advantage of including primary as well as management functions in one single model, is that also the *relationship* between both kind of functions is modelled. The lack of such relationship is one of the reasons why current management products can not directly be used to meet actual management needs (the last problem of the previous subsection). The

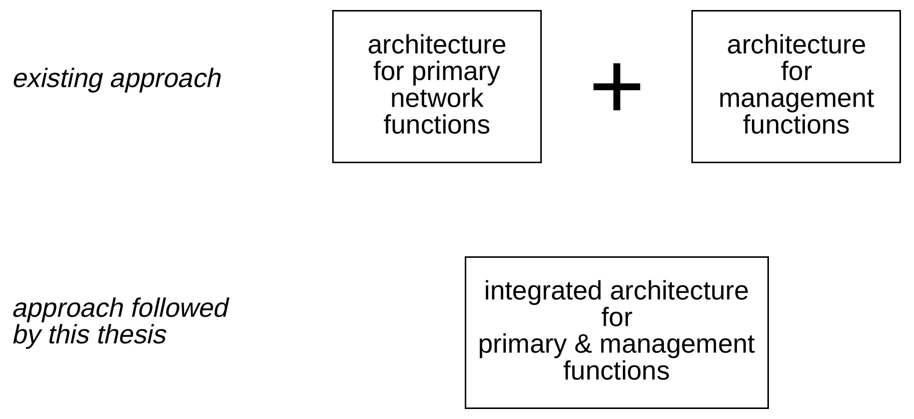

existing approach
architecture for primary network functions
+
architecture for management functions
approach followed by this thesis
integrated architecture for primary & management functions

Figure 1.8: Integrated network management architecture

alternative architectural model of Chapter 7 is meant to provide a solution for that problem.

The idea that no difference exists between the design of primary functions and the design of management functions, also implies that it should be possible to model both kind of functions in terms of a single set of architectural *concepts* and *rules*. Instead of developing a management architecture from scratch, one should use the architectural concepts and rules that are already applied for the design of primary functions. To meet the problem that the concepts that are used in current management architectures are not always properly defined (the first problem mentioned in the previous subsection), this thesis proposes to use the architectural concepts and rules of the OSI Reference Model [43]. As compared to others, these rules and concepts have been clearly identified and can be applied in a consistent way [113].

An attempt to use these concepts and rules has been previously made by the members of the OSI management group. As will be explained in Chapter 2 of this thesis, this group was unable to present a consistent architectural model. One of the reasons why this group did not succeed, was the fact that they confused different abstraction levels. This lack of understanding of the various abstraction levels was also one of the reasons why this group suggested to use the managed functions for the exchange of management information (the second problem of the previous subsection).

This thesis demonstrates that it is possible to use in a consistent way the concepts and rules as defined by the OSI Reference Model for modelling management functions. The model that is presented in Chapter 7 can in fact be seen as an extension of the OSI Reference Model [43] or a replacement of the OSI Management Framework [44].

To provide some structure in the large list of managed objects (the third problem of the previous subsection), this thesis proposes to distinguish between three classes of management aspects: service management, protocol management and element management. Instead of a monolithic list containing thousands of managed objects, this classification takes care that the manager will be confronted with a limited number of smaller lists.

Contribution of this thesis

The objective of this thesis is to *improve insight and understanding of network management,* and *to present an alternative network management model* (Figure 1.9). This model can be useful to guide the design of network management services and protocols. It should be noted that even though this thesis concludes with a number of general remarks with respect to such management services and protocols, this thesis does not propose any specific new service or protocol. Other issues that will not be addressed are:

- The implementation and realization of individual network systems.
- The definition of new management information models or MIBs.
- The provision of concrete solutions for specific management problems.

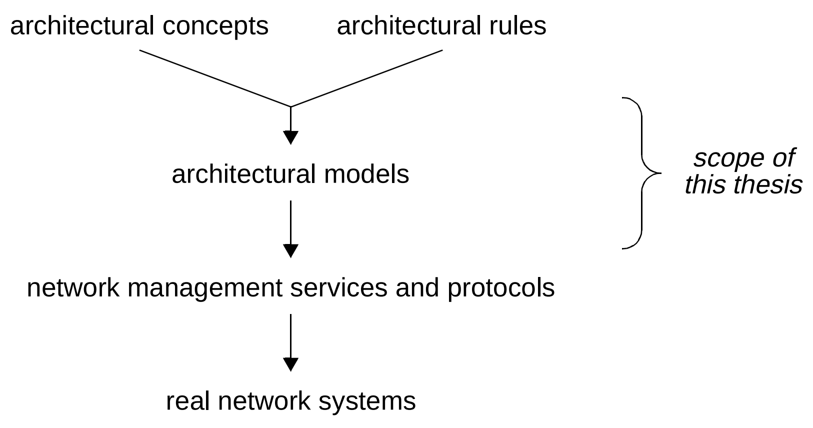


Figure 1.9: Scope of this thesis

1.5 Structure of this thesis

This thesis consists of two parts.

Part 1 discusses the state of the art. It starts to identify the various organizations that have defined network management architectures, and subsequently analyses the three most prominent architectures:

- ISO-OSI's management architecture (Chapter 2).
- The Telecommunications Management Network (TMN), defined by the ITU-T (Chapter 3).
- The Internet network management framework (Chapter 4).

The deficiencies of these architectures are identified and discussed; these deficiencies form the input to part 2 of this thesis (Figure 1.10).

Part 2 discusses an alternative approach to network management. The Chapters 5 and 6 explain how primary as well as management functions might be

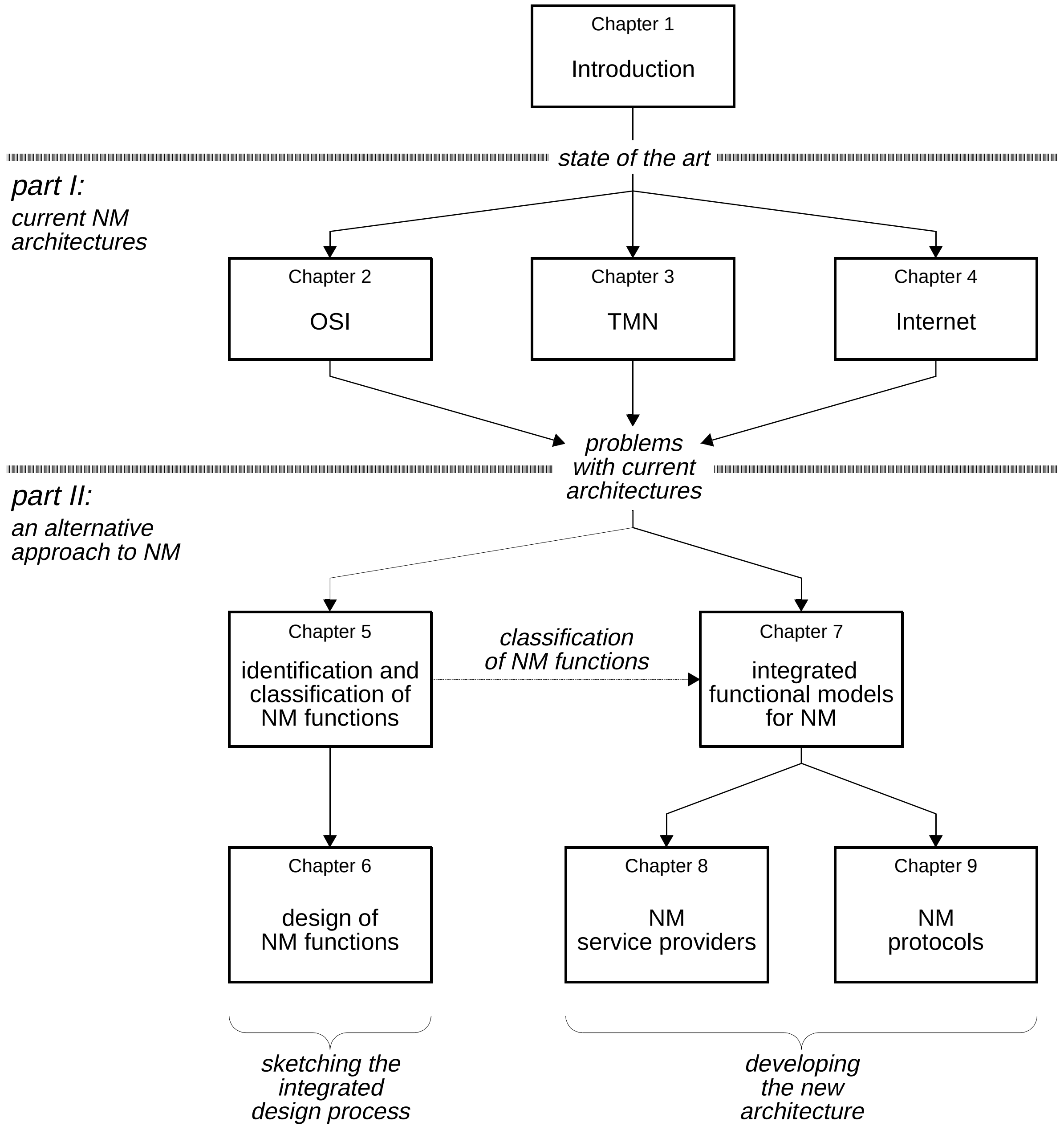

Chapter 1
Introduction
state of the art
part I:
current NM architectures
Chapter 2
OSI
Chapter 3
TMN
Chapter 4
Internet
problems with current architectures
part II:
an alternative approach to NM
Chapter 5
identification and classification of NM functions
classification of NM functions
Chapter 7
integrated functional models for NM
Chapter 6
design of NM functions
Chapter 8
NM service providers
Chapter 9
NM protocols
sketching the integrated design process
developing the new architecture

Figure 1.10: Structure of this thesis

tackled as part of a single design process; the Chapters 7 through 9 present the integrated architecture that models the relationship between both kinds of functions.

To identify and classify potential management functions, Chapter 5 discusses the design of primary functions. An important result of this chapter is the proposal to distinguish between three classes of management functions: service management, protocol management and element management.

The purpose of Chapter 6 is to explain when management functions should be developed during the design process. The explanation in this chapter will be based upon the cyclic design model; it will be shown that management functions should preferably be tackled during the later cycles of such design. Since it may not always be possible to complete the design of all management functions before the start of the operational phase, Chapter 6 proposes to extent the cyclic design model to include the design of future system *generations*.

In Chapter 7 an integrated architecture for primary as well as management functions will be developed. To provide some structure for the various management issues, this chapter uses the classification of management functions as proposed in Chapter 5. This results into three functional models: one for service management, one for protocol management and one for element management. All models show the relationship between primary and management functions and may be useful for the development of management simulators.
The models that are defined in Chapter 7 introduce special service providers for the exchange of management information; these service providers will be discussed in Chapter 8.
Chapter 9 further discusses some important characteristics of the management protocols that have been identified in Chapter 7. Two opposite approaches will be presented: a variable oriented approach and a command oriented approach. The object oriented approach, which is used for OSI management, can be considered as a mixture of both approaches and will be discussed too.

1.6 Intended audience

This thesis is intended for:
- Those who want an overview of current network management architectures.
- Those who want an understanding of some of the basic problems with current management architectures.
- Those who are interested in alternative design models for network management.
- Those who want a better understanding of the relationship between primary and network management functions.

In this thesis it is assumed that the reader is familiar with the architectural concepts and rules as defined by the OSI Reference Model.

Part I

Introduction and Analysis of Standardized Management Approaches

There are several organizations who have developed services, protocols and architectures for network management. The three most important organizations are:

- The International Organization for Standardardization (ISO).
- The Comité Consultative Internationale de Telegraphique et Telephonique (CCITT); this organization is nowadays called the Telecommunication Standardization Sector (T) of the International Telecommunication Union (ITU).
- The Internet Engineering Task Force (IETF).

Of these three ISO was the first who started, as part of its 'Open Systems Interconnection' (OSI) program, the development of an architecture for network management. The first proposals for such an architecture appeared during the early 1980; nowadays a large number of standards exist for the architecture as well as for network management services and protocols. Of these standards the 'OSI Management Framework', the 'OSI Systems Management Overview' and the 'Common Management Information Protocol' (CMIP) are probably the best known examples. In Chapter 2 of this thesis the OSI management approach will be discussed.

Initially the aim of ISO was to define management standards for datacom networks; development of management standards for telecom networks was left to CCITT. In 1985 CCITT started the development of such management standards; these standards have become known as the 'Telecommunications Management Network' (TMN) recommendations. Originally these recommendations were self standing, but during the 1988-1992 study period they have been rewritten to include the ideas of OSI management. Nowadays OSI management and TMN can be seen as each others complements; Chapter 3 of this thesis discusses TMN.

Looking back at the last decade it may be concluded that the growth of the Internet has played a decisive role in the development of network management protocols. Initially the Internet Architecture Board (IAB) intended to apply the OSI management approach, but at the time the size of the Internet reached a level at which management became indispensable, OSI management groups were still busy with discussing the OSI management framework. Since implementations of OSI management were not expected to appear soon, the IAB requested the IETF (the organization who is responsible for the development of Internet protocols) to define an ad hoc management protocol. This 'Simple Network Management Protocol' (SNMP) was completed within a year and soon many manufacturers started the production of SNMP compliant systems. Although SNMP has several deficiencies, it has become the de facto standard for management of datacom networks. In 1993 an attempt was made to tackle these deficiencies and an improved version of SNMP (SNMPv2) appeared. Chapter 4 of this thesis discusses this Internet management approach.

Next to ISO, CCITT and the IETF also other organizations are worth mentioning for their role in the development of network management. Because of their comparatively modest role, this thesis will not devote separate chapters to discuss the details of these developments. Instead, a short overview will be given on the next pages.

IEEE

The Institute of Electrical and Electronics Engineers (IEEE) is a professional organization which, amongst others, defines standards for Local and Metropolitan Area Networks (LANs and MANs). These standards are commonly known as the IEEE 802 standards. Some of these standards define how management should be performed in LAN and MAN environments (Figure 1).

Number	Title
IEEE 802.1B	LAN/WAN Management
IEEE 802.1E	System Load Protocol
IEEE 802.1F	Common Definitions and procedures for IEEE 802 Management Information

Figure 1: IEEE Management standards

The IEEE management standards are based upon the ISO CMIP standard. As opposed to ISO, IEEE does not use this protocol at application level (layer 7), but at data link level (layer 2). The name that is used for the IEEE approach, is Common Management Over LLC (CMOL). A problem with this approach is that it is impossible to manage stations located at other sides of routers (routers, by definition, relay via layer 3). IEEE management is thus restricted to single (bridged) LANs or MANs; to manage LANs interconnected by routers, IEEE proposes to use the combination of IEEE and ISO management.

> *Example:* an important advocate of CMOL is IBM. It seems that the restriction that CMOL can not operate over layer 3 routers is acceptable for IBM. This may be because IBM's interconnection strategy is based upon 'source-routing bridges'; usage of layer 3 routers is avoided whenever possible. Since CMOL operates well over source-routing bridges, it is always possible to manage from a central location multiple (IBM) LANs.

Network Management Forum

In 1988 the 'OSI/Network Management Forum' was formed to promote the rapid development, acceptance and implementation of OSI and CCITT management standards [31][32]. The Forum is a non-profit organization whose members are manufacturers, operating companies and research laboratories. After a few years the prefix 'OSI' was removed to indicate that the Forum had widened its scope to reference management standards from other sources.

Examples of such standards are:

- SNMP from the IETF.
- The 'Distributed Management Environment' (DME) [2] from the Open Software Foundation (OSF).
- The 'Management Protocol API' (XMP) and the 'OSI-Abstract Data Manipulation API' (XOM) from X/Open.
- The 'Common Object Request Broker Architecture' (CORBA) from the Open Management Group (OMG).

To organize its work, the NM Forum has defined the OMNI*Point*[1] program. This program comprises "a set of standards, implementation specifications, testing methods plus tools, object libraries that make possible the development of interoperable management systems and applications" [72][74]. The success of the program is somewhat disappointing, presumably because some parts turned out to be more complex than expected (e.g. XOM [25]) and because the delivery schedule could not always be met (e.g. in case of CORBA).

RACE

RACE is a program of the European Community to promote Research and development in Advanced Communications technologies in Europe. The objective of RACE is to introduce community wide Integrated Broadband Communication (IBC) by 1995. To accomplish this goal, the RACE programme includes more than hundred different projects. Many of these projects address management aspects of the IBC. Some projects even invest all of their resources on IBC management (Figure 2).

Number	Name	Description
R1003	GUIDELINE	Advanced Information Processing (AIP) standards for TMN
R1005	NEMESYS	Traffic and Quality of Service (QoS) management for IBCN
R1006	AIM	AIP application to IBCN maintenance
R1009	ADVANCE	Network and customer administration systems for IBCN
R1024	NETMAN	Functional specifications for IBC telecommunications management
R1053	TERRACE	TMN evolution of reference configurations for RACE
R1082	QOSMIC	QoS verification methodology and tools for IBC
R2002	GEMA	General Maintenance Application
R2004	PREPARE	Pre-pilot in advanced resource management
R2021	DESSERT	Decision support system for service management
R2041	PRISM	Pan-European reference configuration for IBC services
R2051	ICM	Integrated communication management

Figure 2: RACE projects on IBC management

1. OMNI*Point* stands for Open Management Interoperability Point

Within RACE, Special Interest Groups (SIGs) and Task Groups (TGs) have been formed to coordinate the results (deliverables) of the different projects. In case multiple projects agree within a TG on some common result, this result can be published as a Common Functional Specification (CFS). Such a specification is often submitted to one of the standardization bodies (usually ETSI).

The RACE programme is dominated by the telecommunications industry and operating companies. It is therefore not surprising to see that research within RACE is based on the work of ETSI and CCITT (TMN in particular). RACE also uses the results of OSI management, because TMN includes pointers to this work. Other standards (e.g. Internet and IEEE) have virtually no impact on RACE.
It is difficult to judge the effect of management CFSs outside RACE. CFSs should not be seen as specifications that can immediately be used by implementers to solve particular management problems. Instead, CFSs can better be considered as collections of ideas that may be useful for standardization organizations such as ETSI and CCITT.

2: OSI Management

2 OSI Management

The purpose of this chapter is to *explain* and *analyse* OSI management. The *explanation* of OSI management may be useful for readers to obtain sufficient background information to understand the remainder of this thesis. The *analysis* will be needed to identify which problems must be solved in Part II of this thesis.

Although the origin of OSI management can be found in ISO, most of the work is performed in collaboration with the ITU-T (the former CCITT). The standards that result from this cooperation are published by both organizations without technical differences. Within the ITU-T, the OSI management recommendations are published as part of the X.700 series.

The first standard that describes OSI management, is the *OSI Reference Model* [43]. This standard identifies OSI management as an important working area and provides initial definitions. Around 1980, a special Working Group (ISO/TC 97/SC 21/WG 4[1]) was formed within ISO to further develop OSI management. The first outcome of this WG was the *OSI Management Framework* [44]. Although the production of this framework took considerable time, it was not generally accepted as an adequate starting point. It was therefore decided to produce an additional standard, which was called the *Systems Management Overview* [53]. Together these standards provide the basis for OSI management (Figure 2.1).

Title	ISO/IEC	ITU-T	Year of publication
OSI Management Framework	7498/4	X.700	1989
OSI Systems management Overview	10040	X.701	1992

Figure 2.1: Basis of OSI Management

This chapter is structured as follows. Section 2.1 and Section 2.2 *explain* OSI management. Reading is recommended for those who do not yet understand this type of management; people who are familiar with it may skip these sections. In Section 2.1 the OSI Management Framework will be discussed; the problem areas that may be solved with OSI management will be identified and the ways in which management information can be exchanged will be discussed. Section 2.2 addresses the OSI Systems Management Overview; it discusses functional, information, communication and organizational aspects of Systems Management.

The *analysis* of OSI management is given in Section 2.3. The purpose of this section is to identify which problems must be resolved in part II of this thesis.

1. Nowadays the group is called ISO-IEC/JTC 1/SC 21/WG 4

2.1 OSI Management Framework

This section discusses the first standard in which OSI management is defined: the OSI Management Framework.

Subsection 2.1.1 describes the problem areas for which OSI management was developed. These areas are the so-called *functional areas* of OSI management. Subsection 2.1.2 discusses the possible ways to exchange management information; these ways are: systems management, layer management and layer operation. Finally Subsection 2.1.3 discusses managed objects, management information and the Management Information Base (MIB).

2.1.1 Functional Areas

The first Working Drafts of the Management Framework already contained sections on management functions. These management functions gradually evolved into what is presently known as the five functional areas of OSI. To denote these areas, the term 'FCAPS' is commonly used (this term is a contraction of the five initial letters of the functional areas).

2.1.1.1 Fault management

Fault management is the set of facilities which enables the detection, isolation and correction of abnormal operation. Possible causes for abnormal operation are: design and implementation errors, overload errors, external disturbances, and lifetime expiration. Fault management includes functions to:

- Maintain and examine error logs.
- Accept and act upon error notifications.
- Trace and identify faults.
- Carry out diagnostic tests.
- Correct faults.

2.1.1.2 Configuration management

Configuration management is the set of facilities which:

- Records the current configuration.
- Records changes in the configuration.
- Identifies network components (give addresses to Service Access Points and titles to network entities).
- Initializes and closes down network systems.
- Changes network parameters (e.g. routing tables).

An important aspect of configuration management, is the assignment of names. To stress this importance, the term *configuration and name management* is sometimes used [110].

2.1.1.3 Accounting management

Accounting management is the set of facilities which enables charges to be established, and costs to be identified for the use of network resources. These resources can be:
- The network service provider, which is responsible for the transfer of user data (e.g. the public network).
- Network applications (e.g. directory services).

Accounting management may:
- Inform users of the costs thus far.
- Inform users of the expected costs in the future.
- Set cost limits (e.g. disable 06 telephone connections).
- Combine costs (to prevent the user from receiving separate bills for each individual connection or, in case of international connections, from each country traversed).

2.1.1.4 Performance management

Performance management is needed to optimize the Quality of Service (QoS). To detect changes in the network's performance, statistical data (e.g. timer and counter values) should be collected and logged on an incidental or periodical basis. The use of such logs is not restricted to performance management; also other management areas take advantage of these logs:
- Performance logs can be used by fault management to detect faults.
- Performance logs can be used by configuration management to decide when changes are needed in the configuration.
- Performance logs can be used by accounting management to adjust bills.

To allow a meaningful comparison of performance logs, also the configuration must be known that existed at the time the logs were made. Configuration information must therefore be logged too.

2.1.1.5 Security management

Security management is the set of facilities which enables the manager to initialize and modify those functions that secure the network from user misbehaviour and unauthorized access. Important parts of security management are key management (for authorization, encryption and authentication), maintenance of firewalls [12][24] and creation of security logs.

2.1.2 Exchange of management information

Three different ways to exchange management information were already identified in the OSI Reference Model: systems management, application management and layer management. Although one would expect that SC 21/WG 4 would use these three approaches as starting point in the development of the

Management Framework, this did not happen. Instead, SC 21/WG 4 decided to remove *application management* and include *layer operation*.

2.1.2.1 Systems management

The initial definition of systems management, as found in the OSI Reference Model, distinguishes between two different properties:

- Systems management is related to the management of OSI resources and their status across all layers of the OSI architecture.
- Protocols for systems management reside in the application layer.

The first property explains *what* is being managed, the second explains *how* management information should be exchanged.

It is interesting to see that the OSI Management Framework focuses on the *information exchange* aspect of systems management (and ignores the aspect of *what* is being managed). Systems management can thus be characterized by the fact that application protocols should be used for the exchange of management information. Application protocols are built upon reliable, connection-oriented underlying services (the term *'royal route'* has sometimes been used to characterize this way of management information exchange [61]).

The decision to use application layer protocols is based upon the assumption that management information should be exchanged in the same way as all other forms of information. According to this view, management should be regarded as just another application on top of the network[1].
To model the exchange of management information, the concept of Systems Management Application Entities (SMAEs) was introduced. SMAEs reside in the application layer and realize the communication aspects of the systems management functions (Figure 2.2).

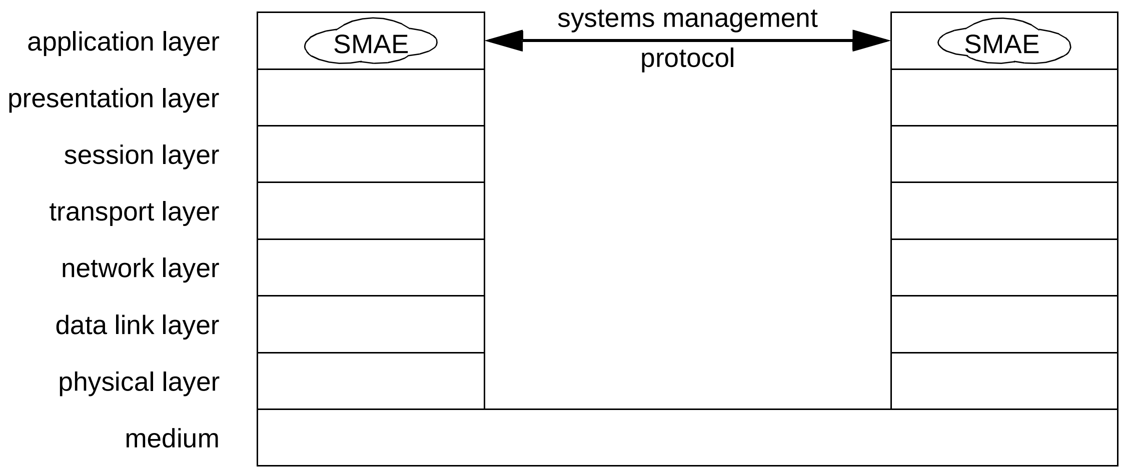


Figure 2.2: Systems management should be seen as an application protocol

1. The OSI Management Framework includes the following text: "it is perceived that the majority of management information exchanges will require context negotiation, the establishment of a management session, a reliable end-to-end transport service etc., in exactly the same way as other application layer exchanges".

The defenders of management exchanges at application level use the following arguments:

- Application layer protocols are the most 'powerful' kind of protocols. One *single* application layer protocol will be capable to transfer many types of management information. Defining *one* powerful management protocol will be much better than defining *many* futile management protocols.
- Services that are provided by lower layers are usually not good enough to satisfy all management needs[1]. To exchange for example large routing tables, the full capabilities of all OSI layers may be required (e.g. error detection, error correction, segmentation, reassembly, context negotiation etc.).
- Management is seen as an application on top of a network. If ISO would model this application not within the application layer, it would undermine its own approach.

The opponents of management exchanges at application level use the following arguments:

- Implementing all seven layers of the Reference Model is expensive. There are many systems that, for their normal operation, do not need to implement all seven layers (e.g. bridges and routers). In these systems it may be a waste of money to implement the remaining layers, just to allow management.
- After a network collapse, an important management responsibility is to restore network services. As a result of the collapse, application layer protocols may no longer function well. In case the exchange of management information relies upon the correct operation of these protocols, management functions may no longer be reachable.
- Application layer protocols involve a lot of processing and are relatively slow.
- Application layer protocols do not have multicast or broadcast facilities.

2.1.2.2 Layer management

While systems management has been defined as the *preferred* way to exchange management information, it is not the only way. The OSI Management Framework allows as alternative for example *layer management*, which has the following properties:

- (N)-layer management supports the monitoring, control and coordination of (N)-layer managed objects.
- (N)-layer management protocols are supported by protocols of the layers (N-1) and below.

The first item relates layer management to *what* is being managed, the second tells us *how* (N)-layer management information should be *exchanged*. Figure 2.3 shows the example of OSI network layer management information, which is exchanged by means of a special purpose layer management protocol located on top of normal communication protocols (a similar figure can be found in the annex to the OSI Management Framework).

1. It is interesting to remember that IEEE's CMOL defines that CMIP (which is a systems management protocol) should be run on top of LLC (page 19).

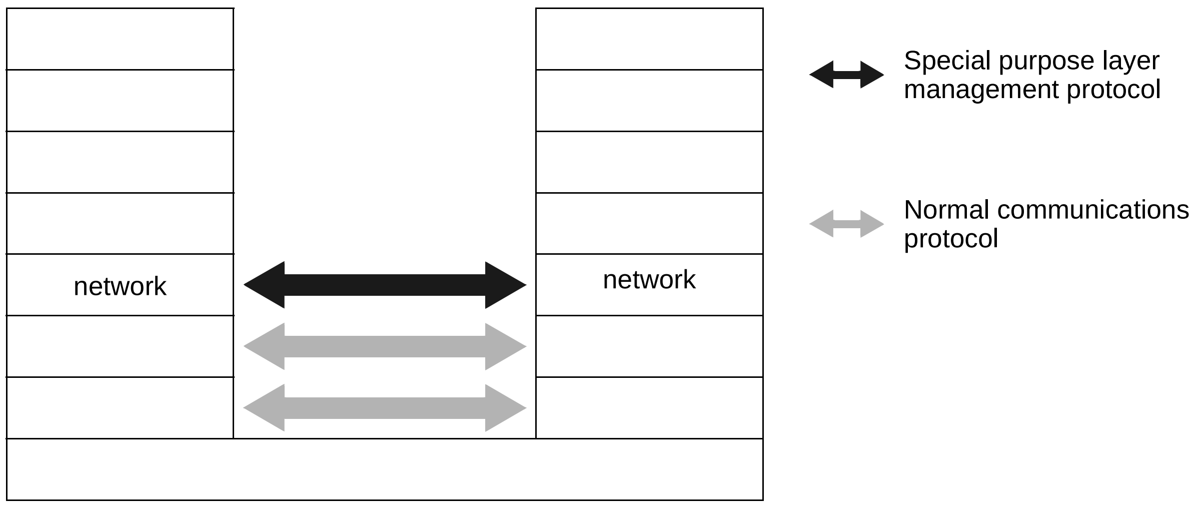

network
network
Special purpose layer management protocol
Normal communications protocol

Figure 2.3: Layer management versus normal communication protocols

An important distinction between systems management and layer management, is that systems management uses the *presentation service* for the exchange of management information, whereas (N)-layer management uses the *(N-1)-service*. According to the Management Framework, "usage of layer management is restricted to those cases where usage of systems management is inappropriate".

> *Example:* Layer management is commonly used for the exchange of routing information. In a number of cases, routing information must be broadcasted over an entire routing domain. Since the presentation service has no broadcast capabilities, it may be inefficient to use systems management. Several existing routing strategies therefore rely upon layer management protocols (Figure 1.7).

Other examples of layer management are given in Figure 2.4. The standards which are mentioned in this figure are implemented in many networks that support the OSI ConnectionLess Network Protocol (CLNP) [46]. The figure is included to demonstrate that, contrary to what is sometimes suggested, in real networks layer management exchanges occur frequently.

PDU type	Defined by	When generated
Bridge PDUs	ISO 10038	Generated by all bridges after expiration of Hello timer (default value: 2 seconds)
Configuration PDUs	ISO 9542	Generated by all network entities after expiration of Configuration timer (min. value: several seconds; max. value: several minutes)
Hello PDUs	ISO 10589	Generated by all routers after expiration of Hello timer (default value: 10 seconds)

Figure 2.4: Examples of layer management exchanges

2.1.2.3 Layer operation

The last type of management information exchange is *layer operation*. This form was first defined by the Management Framework and has not been mentioned in the OSI Reference Model. Layer operation is defined as "monitoring and controlling a *single instance of communication*[1]". In case of layer operation, management information is carried as part of a normal layer protocol. Just as with (N)-layer management, (N)-layer operation uses the underlying (N-1)-protocols for the exchange of management information (Figure 2.5).

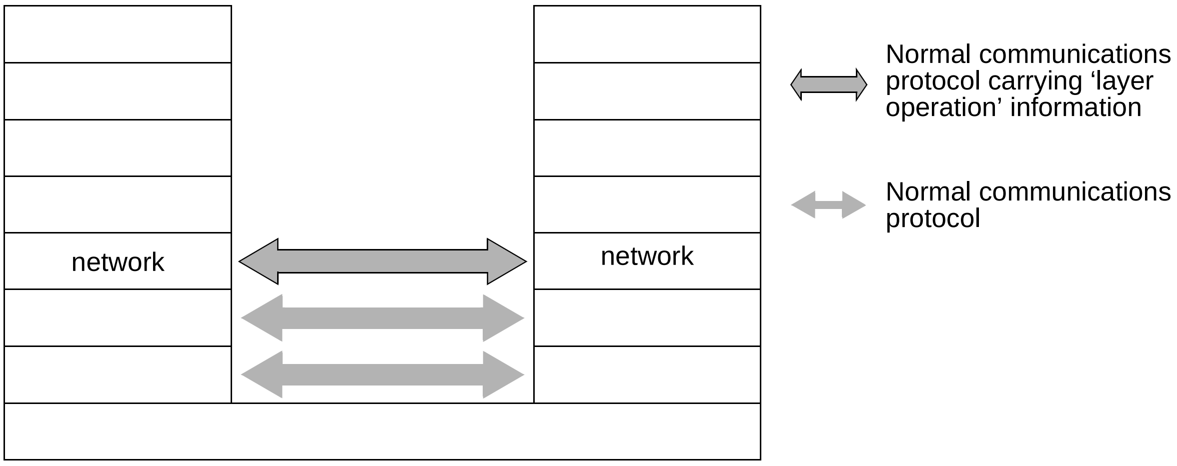

network
network
Normal communications protocol carrying 'layer operation' information
Normal communications protocol

Figure 2.5: Layer operation versus normal communication protocols

2.1.3 Managed objects, management information and the MIB.

To understand the relationship between managed objects, management information and the 'Management Information Base' (MIB), it may be helpful to take a look at the development of the Management Framework standard. Several draft versions of this standard contained the following definitions:

- *Managed object:* "those data processing and data communications resources (whether OSI resources or not) that may be managed through the use of an OSI management protocol".
- *Management information:* "Information *associated* with a managed object that is operated on by the OSI Management protocol to control and monitor that object".

These definitions suggest that a difference exists between managed objects and management information. Although the various drafts of the Management Framework document are sometimes difficult to understand, also the following view emerges:

- managed objects reside in the various layers of the OSI RM,
- management information resides in the *Management Information Base* (MIB).

1. A single instance of communication is a single connection (in case of a connection oriented service) or a single request-response pair (in case of a connectionless service).

The MIB can be seen as a kind of database. The contents of this database is not the set of managed objects themselves, but the information that is *associated* with the managed objects. Layer Managers (LMs) are responsible to maintain the association between MIB information and managed objects (Figure 2.6). In case of problems with Layer Managers, it might occur that the information in the MIB does not accurately reflect the state of the managed objects any more.

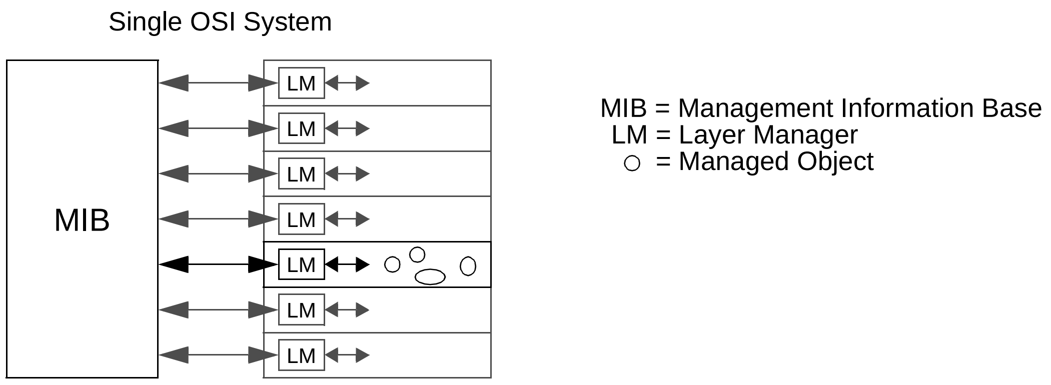


Figure 2.6: Early view of MIB, managed objects and Layer Managers

This view of managed objects, management information and the MIB was still expressed in the DIS version of the management framework (1988). In fact this view is still supported by many people[1]. For unclear reasons (none of the national bodies made an explicit request) the editing meeting that was responsible for resolution of the comments on the DIS ballot:

- Removed the definition of management information.
- Changed the definition of managed objects.
- Changed the description of the MIB.
- Removed an explanatory picture from the Annex.

As a result of these changes, there exists no longer a difference between the management information that can be stored within a MIB, and the managed objects themselves. According to the final version of the Management Framework "the set of managed objects within a system, constitutes that system's MIB". Since this text implies that a MIB is conceptually nothing more than the collection of all managed objects within that system, the MIB concept does not seem to be very useful any more [115].

2.1.4 Impact of the OSI Management Framework

The OSI Management Framework is the first in a series of OSI management standards. It would therefore be reasonable to expect that this standard contains important information and is generally accepted. In this subsection both of these assumptions will be analysed.

1. This view is for instance still being used by the Internet management (SNMP) group.

Information

Although its difficult to determine the *quality* of the information that is in the OSI Management Framework, it is well possible to examine its *quantity*. It is, for instance, interesting to look at the development of the text on *systems management*[1], which is the most important form of OSI management.

Systems management was first defined by the OSI Reference Model, which included about 150 words to explain the idea.

The first working draft of the OSI Management Framework appeared in June 1981. This draft contained about 200 words and several pictures to explain systems management. In subsequent working drafts, new text was being added and existing text was being modified. In the final working draft (number 7, November 1985), about 1600 words were used to explain systems management. Besides, several pictures were included.

Unfortunately, the working drafts were not very consistent and contained several ambiguities. During the various ballot stages[2], SC 21/WG 4 was unable to resolve these ambiguities. As a result, there was no alternative than the removal of controversial parts, including all pictures. In the final text of the OSI Management Framework, the explanation of systems management has been reduced to something less than a single page (200 words).

Although production of the OSI Management Framework took eight years, the final text contains the same number of words on systems management as the first working draft...

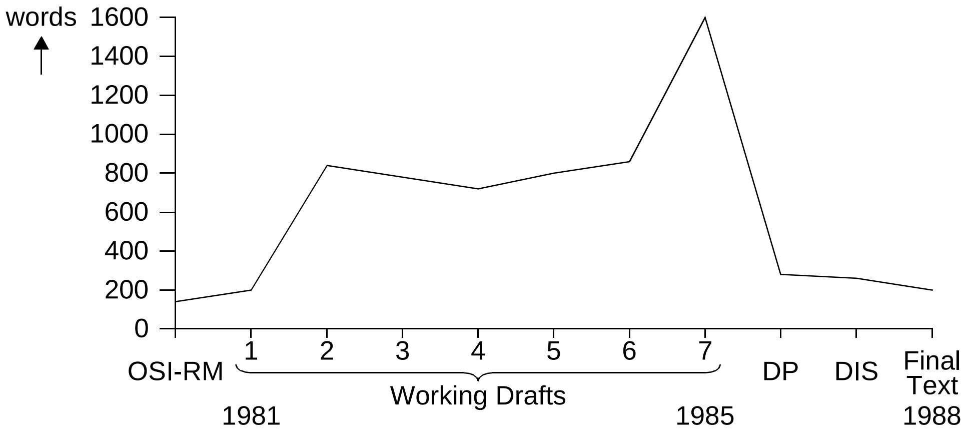


Figure 2.7: Length of Systems Management text (in words)

Level of acceptance

The fact that major pieces of text had to be removed during the various ballot stages should not be a problem, provided that the remaining text was generally accepted. This is barely the case however, as becomes clear from the following examples:

1. In this subsection the standards's informative (non-integral) annexes will not be taken into account.
2. DP: September 1986; DIS: January 1988; acceptance of final text: December 1988; date of publication: November 1989

- The last opportunity for ISO national bodies to judge the Management Framework, was during the ballot on the DIS. National bodies had to vote on an incomplete document however. The section on managed objects contained for example just a single sentence, plus an editors note saying that further detail may be required. The (non-integral) Annex contained multiple ‘to be provided’ statements.
- The Summary of Voting [62] on the DIS version of the Management Framework showed 13 approvals, 2 abstentions and 2 negative votes. At first sight, this result suggests a reasonable level of acceptance. However, a number of member bodies had severe reservations. Some of these are shown below:
  - "AFNOR is aware that technical architectural material is still missing ..."
  - "This DIS is in no way wrong or misleading. It is, however, according to our opinion, completely insufficient ..." (DIN)
  - "NNI has the strong feeling that the current DIS does not contain those concepts that should be part of the management framework"
  - "The UK disapproves DIS 7498-4, because major revisions are required to remove general inconsistencies ..."
- The editing meeting held to resolve the comments on the DIS version of the Management Framework, was attended by only six people from four countries (previous meetings showed a much better participation). Still it was decided to make major technical changes to the document (see for instance page 30). Despite these changes, the editing meeting did not find it necessary to hold a second DIS ballot.

The sequel

Even the final versions of the Management Framework document could not get substantial technical support. The fact that the document was eventually accepted, should therefore be understood from the following perspectives:

- Most people did not want to spend more time on the document.
- For political reasons it would be unwise not to go to IS (International Standard) status. The alternative would be the much lower TR (Technical Report) status.
- Work had recently started on the OSI Systems Management Overview document. Outstanding issues could be discussed during the progression of this document.

2.2 OSI Systems Management Overview

The definition of the OSI Systems Management Overview (SMO) started around 1987. In June 1991 the final SMO text was ready and the document was submitted for registration as IS. Compared to the OSI Management Framework, the SMO contains much more information and is far better accepted.

The SMO includes a further description of systems management. This description distinguishes between the following aspects:

- information
- organizational
- functional
- communication

The following four subsections discuss each of these aspects. The scope of these discussions is not restricted to the SMO document; each subsection also includes references to derived ISO/ITU-T standards and parts of these derived standards will also be explained.

2.2.1 Information aspects

The information aspects of the systems management model deal with the resources that are being managed. These resources are viewed as 'managed objects'.

The concept of managed objects was introduced as part of the OSI's Management Framework. Initially this introduction was considered to be sufficient; the concept of managed objects was not further elaborated because it was thought obvious and in violation with the OSI principle that stated that only external behaviour of systems may be standardized [112]. As time went on, it appeared that different people interpreted the managed object concept in different ways: the initial assumption that the concept was obvious, turned out to be wrong! After SC 21/WG 4 realized this problem, it decided to refine the description of managed objects as follows:

> "A managed object is the OSI Management view of a resource that is subject to management, such as a layer entity, a connection or an item of physical communications equipment. Thus, a managed object is the abstraction of such a resource that represents its properties as seen by (and for the purpose of) management. An essential part of the definition of a managed object is the relationship between these properties and the operational behaviour of the resource. This relationship is not modelled in a general way."

An interesting part of this description is the last sentence, which states that the relationship between operational behaviour and management properties is not modelled in a general way. Without such a relationship, it is not possible however to express the effect of management operations upon the managed resources. This is clearly undesirable. An important difference between the OSI management approach and the management approach that is presented in part II of this thesis, is that the latter does in fact model such a relationship.

According to OSI's Management Information Model [54], the management view of a managed object is visible at the *managed object boundary*. At this boundary, the management view is described in terms of (Figure 2.8):

- Attributes, which are the properties or characteristics of the object.
- Operations, which are performed upon the object.
- Behaviour, which is exhibited in response to operations.
- Notifications, which are emitted by the object.

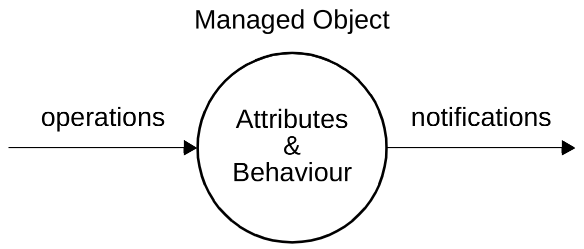


Figure 2.8: A managed object

Next to the managed objects that represent resources, there are also *'management support objects'*. Such objects may be introduced by the designer of management functions during the implementation phase. An example of a management support object is a 'log record', which may be used to store management information.

The managed object concept is refined in a number of additional standards, which are called the Structure of Management Information (SMI) standards (the first six entries of Figure 2.9). The SMI standards do not specify the actual managed objects; managed objects are defined by the working groups responsible for the various layers of the OSI Reference Model (examples of such standards are given in the last four entries of Figure 2.9).

Title	ISO/IEC	ITU-T
Management Information Model	10165-1	X.720
Definition of Management Information	10165-2	X.721
Guidelines for the definition of Managed Objects	10165-4	X.722
Generic Management Information	10165-5	X.723
Guidelines for Conformance Proformas	10165-6	X.724
General Relationship Model	10165-7	X.725
Management Information related to the transport layer	10737	X.284
Management Information related to the network layer	10733	X.283
Management Information related to the data link layer	10742	X.282
Management Information related to the physical layer	13642	X.281

Figure 2.9: Standards for managed objects

2.2.2 Organisational aspects

OSI systems management is organized in a centralized fashion (Subsection 1.3.2). According to this scheme, a single manager may control several agents. The manager performs operations upon (the managed objects within) the agents, agents forward notifications to their managers. Figure 2.10 illustrates this *manager-agent* concept.

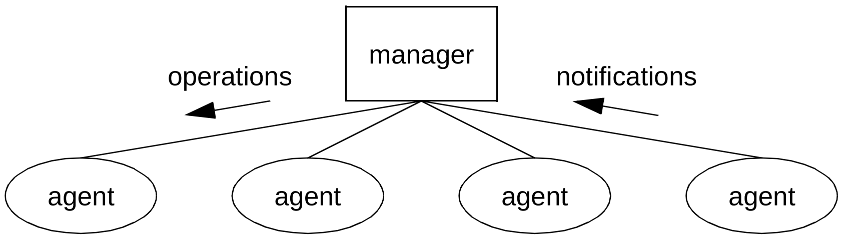


Figure 2.10: Manager-agent concept

The OSI management environment may be partitioned into a number of *management domains*. The partitioning can be based on functional requirements (e.g. security, accounting and fault management), but also on other requirements (e.g. geographical and technological). The idea of management domains is still under development by ISO.

2.2.3 Functional aspects

Soon after the first working drafts of the Management Framework appeared, ISO started to define protocol standards for each of the five functional areas. After some time an interesting observation was made: *most of the functional area protocols used a similar set of elementary management functions*. At the Sydney meeting of SC 21/WG 4 (December 1988), it was therefore decided to stop further progression of the five functional area protocols [66] and concentrate on the definition of *elementary management functions* (Figure 2.11).

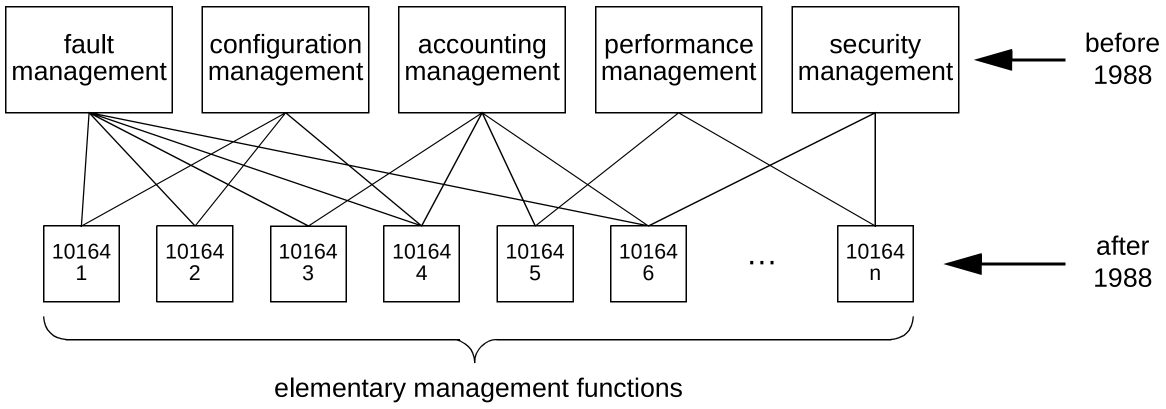


Figure 2.11: Functional areas and elementary management functions

The elementary functions, which are defined at a much lower abstraction level than the original functional areas, are called ‘Systems Management Functions’ (SMF). Figure 2.12 gives a list of these functions; the functions that have no associated ISO/IEC sequence number are still Working Drafts.

Title	ISO/IEC	ITU-T
Object Management Function	10164-1	X.730
State Management Function	10164-2	X.731
Attributes for representing Relationships	10164-3	X.732
Alarm Reporting Function	10164-4	X.733
Event Report Management Function	10164-5	X.734
Log Control Function	10164-6	X.735
Security Alarm Reporting Function	10164-7	X.736
Security Audit Trail Function	10164-8	X.740
Objects and Attributes for Access Control	10164-9	X.741
Accounting Meter Function	10164-10	X.742
Workload Monitoring Function	10164-11	X.739
Test Management Function	10164-12	X.745
Measurement Summarization Function	10164-13	X.738
Confidence and Diagnostic Test Classes	10164-14	X.737
Scheduling Function	10164-15	X.746
Management Knowledge Management Function	10164-16	X.750
Time Management Function		X.743
Software Management Function		X.744
General Relationship Model		X.747
Response Time Monitoring Function		X.748
Management Domain Management Function		X.749
Changeover Function		X.751
Enhanced Event Control Function		X.752

Figure 2.12: Systems Management Functions

It is outside the scope of this thesis to give an in-depth discussion of all Systems Management Functions. The interested reader is referred to [4][110] and [115].

2.2.4 Communications aspects

OSI has defined the 'Common Management Information Service' (CMIS - [50]) as the preferred service for the exchange of management information (although the use of other exchange services is still allowed, such as services provided by TP and FTAM). CMIS' role is restricted to the transfer of management information; actual control of systems is left to the MIS-users which are located on top of CMIS (Figure 2.13).

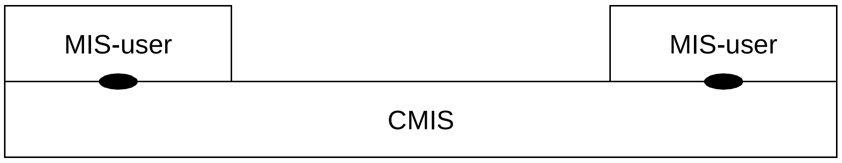


Figure 2.13: MIS-users on top of CMIS

The CMIS service provider may be decomposed, in which case two or more *Systems Management Application Entities* (SMAEs) appear. These entities contain a number of Application Service Elements (ASEs[1]) and use the presentation service provider to transfer their data (Figure 2.14). The interaction between SMAEs is defined by the 'Common Management Information Protocol' (CMIP - [51]).

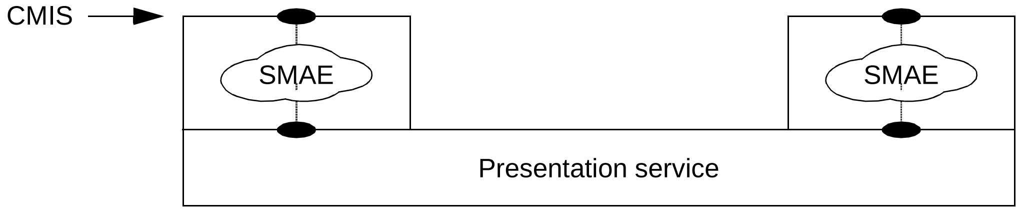


Figure 2.14: Decomposition of CMIS

The CMIS standard defines the following service primitives:

- *M-GET:* to retrieve management information. It can for example be used by a manager to retrieve an agent's network address.
- *M-CANCEL-GET:* to cancel a previously invoked *M-GET*. It is helpful in those cases where the *M-GET* delivers too much information or consumes too many resources. This can happen if, for example, a manager requests an agent to present its entire routing table.
- *M-SET:* to modify the attributes of a managed object. It can for example be used by a manager to change an agent's network address.
- *M-ACTION:* to perform some action on a managed object. It can for example be used by a manager to reboot another network system.
- *M-CREATE:* to create a new instance of a managed object. It can for example be used to add an entry to a routing table.
- *M-DELETE:* to delete an existing managed object instantiation. It is the reverse function of *M-CREATE* and can for example by used to remove an entry from a routing table.

1. See [105] for an explanation of ASEs and application layer structuring.

- *M-EVENT-REPORT:* to report the occurrence of some kind of event. It can for example be invoked by an agent to inform the manager that one of the agent's outgoing links can not be used any more.

The first six primitives define *operations*, the *M-EVENT-REPORT* primitive defines a *notification* (see Figure 2.8). While all primitives can be used in a confirmed way, some (*M-SET*, *M-ACTION* and *M-EVENT-REPORT*) may also be used in an unconfirmed way.

Figure 2.15 lists the ISO / ITU-T standards that define how systems management information should be exchanged; the list does not include the amendments and additions to these standards.

Title	ISO/IEC	ITU-T
Common Management Information Service	9595	X.710
Common Management Information Protocol (CMIP)	9596	X.711

Figure 2.15: Standards for communication aspects

2.3 Analysis

This section discusses some of the main problems of OSI management. It is not the intention to be exhaustive; the focus will be on problems that will somehow be tackled in the alternative management architecture that is presented in Part II of this thesis.

2.3.1 Architectural integrity

An important problem of OSI's management architecture, is that it does not apply the modelling principles of the OSI Reference Model in a proper way. OSI management violates for example the layering principle, which says that users in a particular layer need not know the internal structure of their underlying service provider. According to the layering principle, entities can only interact with entities in adjacent layers via service primitives; it is not possible that entities randomly access components in arbitrary layers by some other means. Still this is exactly what OSI systems management does, as will be explained below.

Consider two systems: one in a manager and one in an agent role (Figure 2.16). The system that operates in the agent role is the one that is being managed; it contains several managed objects to represent the resources that can be managed. The managed objects can be accessed by a SMAE. This SMAE communicates via a systems management protocol (CMIP) with a SMAE that is located in the manager system.

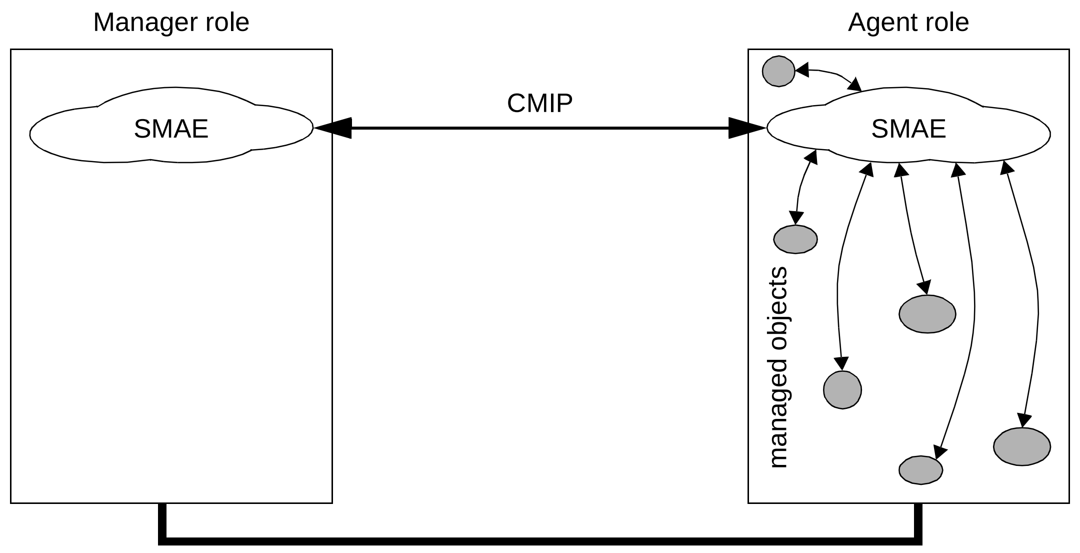


Figure 2.16: OSI systems management

Each layer of the OSI Reference Model may need management. Managed objects can thus be found in all layers of the OSI Reference Model. The SMAE is by definition located in the application layer (Figure 2.17). According to OSI management, the SMAE will be able however to manipulate managed objects, irrespective of the layer in which these objects are located. The implication of this is that the SMAE should have knowledge about the internal structure of the underlying service provider and be able to access components within this provider via some ‘magic’ interaction mechanism. This is in violation with the modelling principles as defined by the OSI Reference Model.

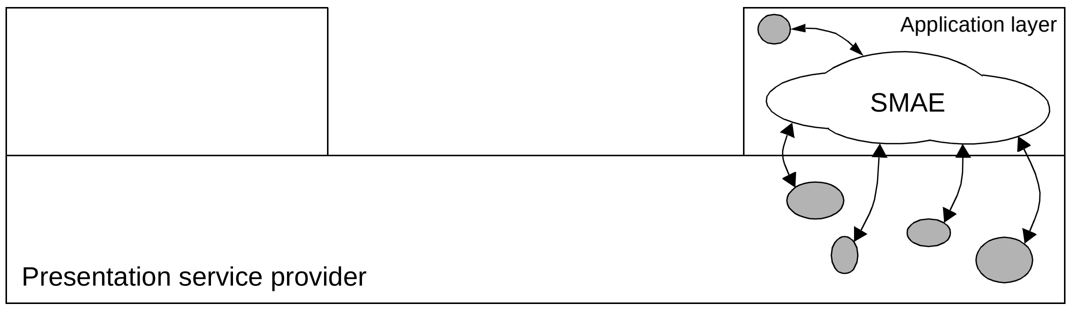


Figure 2.17: The SMAE knows the structure of the underlying provider

Although several people in the OSI management community are aware of this problem, they have until now not been able to find a solution (in fact the removal of pictures from the management framework drafts can be seen as an attempt to disguise the problem). Part II of this thesis proposes an alternative architecture that resolves this problem.

2.3.2 Problems with fault management

Another weak point of OSI’s management approach is that (implicitly) the layer protocols that are being managed, are used for the exchange of management information too. Figure 2.18 will be used to illustrate this relation[1].

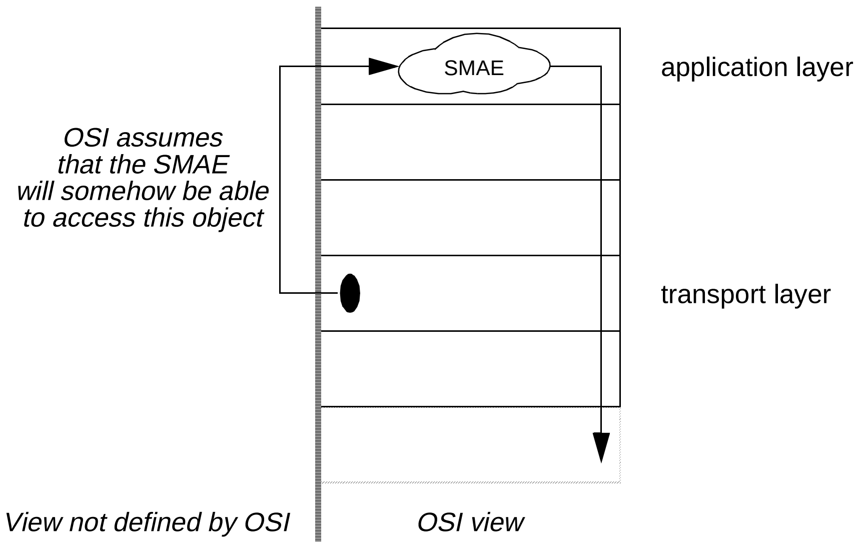

SMAE
application layer
OSI assumes that the SMAE will somehow be able to access this object
transport layer
View not defined by OSI
OSI view

Figure 2.18: Example showing the double role of layer protocols

The figure shows a transport layer managed object, such as a counter that reflects the number of CRC errors. This CRC counter can be read by the SMAE, which resides within the application layer. As explained in the previous subsection, OSI does not describe how the SMAE accesses the transport layer managed object; OSI management assumes however that some form of interaction will be possible. After the SMAE has read the counter, it may decide to send CRC information to other systems. For this purpose, the SMAE presents the information as user data to the underlying presentation service provider. The transport layer protocol is part of this provider, however; the transport layer protocol is thus also used for the exchange of management information.

The protocols that are being managed, will thus be used to exchange management information too. The problem with this dependence is, that fault management may become impossible. Consider for example a system in which the transport entity suddenly breaks. In case all other entities within that system remain operational, the failure may be detected by the SMAE, which may decide to generate an alarm report. This alarm report can not be transmitted however, because of failures within the local transport entity.

2.3.3 Other problems

Besides the two problems that were mentioned in the previous subsections, OSI management is faced with several other problems:

- OSI management explains how individual management operations, such as *GETs* and *SETs*, should be performed. The current management standards do not specify however the sequence in which these operations should be performed to solve specific management problems. Until now, *solutions* for real management problems hardly exist.

1. The merits of Figure 2.18 are, from an architectural point of view, questionable. Variations of it are given however in many publications ([17][33][42][58][59][60][73][109]).

- OSI management is rather complicated. SC 21/WG 4 has introduced several new concepts, which are sometimes difficult to comprehend. Other barriers are the large number of management standards and the size of these standards.
- During the standardization process considerable changes were made in some of the main concepts of OSI management. Examples of such changes are the redefinition of 'managed objects' (see page 30), the removal of 'application management' and the introduction of 'layer operation' (see page 25).
- The standardization of OSI management took too much time. Other approaches, such as SNMP, could therefore emerge.
- Although most manufacturers declared their support for OSI management, only a few offer implementations.
- Management systems that are based on the OSI architecture are presently more expensive than management systems that are based on the Internet management architecture (SNMP).

Due to these problems, it is questionable whether OSI management will reach the dominant market position that has originally been anticipated.

3: TMN Management

3 TMN Management

The term TMN is introduced by the ITU-T (the former CCITT) as an abbreviation for ‘Telecommunications Management Network’. The concept of a TMN is defined by Recommendation M.3010 [20].

According to M.3010, "a TMN is conceptually a separate network that interfaces a telecommunications network at several different points". The relationship between a TMN and the telecommunication network that is managed, is shown in Figure 3.1. According to this figure, the interface points between the TMN and the telecommunication network are formed by *Exchanges* and *Transmission systems*. For the purpose of management, these Exchanges and Transmission systems are connected via a *Data Communication Network* to one or more *Operations Systems*. The Operations Systems perform most of the management functions; these functions may be carried out by human operators but also automatically. It is possible that a single management function will be performed by multiple Operations Systems. In this case, the Data Communication Network is used to exchange management information between the Operation Systems. The Data Communication Network is also used to connect *Work Stations*, which allow operators to interpret management information. Work Stations have man-machine interfaces, the definition of such interfaces fall outside the scope of TMN (Work Stations are therefore drawn at the border of the TMN).

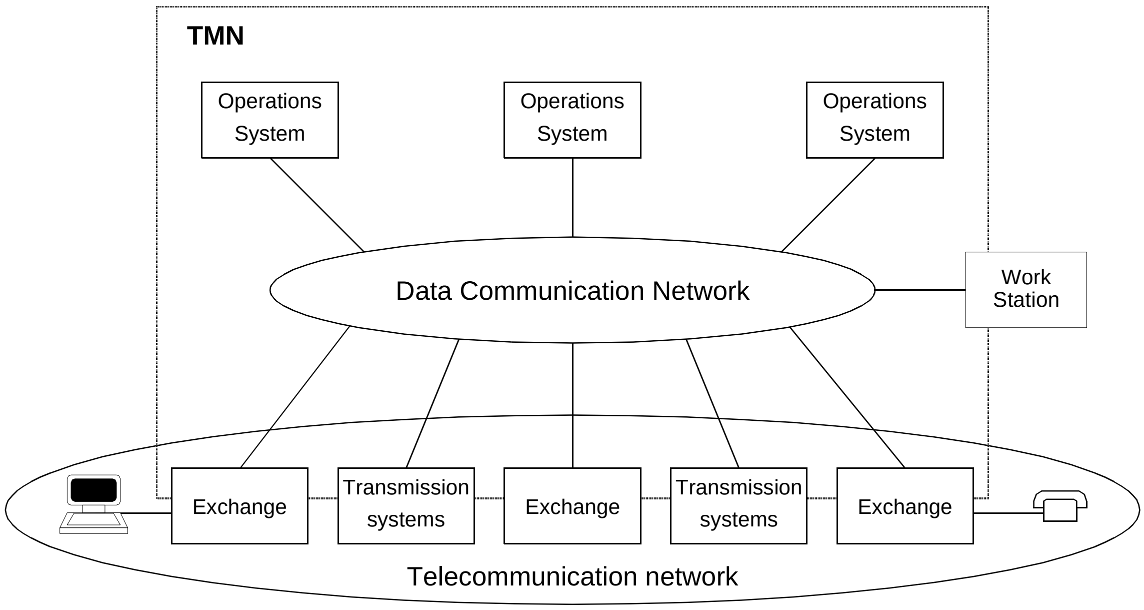


Figure 3.1: General relationship of a TMN to a telecommunication network

In this chapter TMN is introduced and analysed. Section 3.1 identifies which TMN standards exist and how these standards relate to the OSI management standards. The subsequent sections concentrate upon the most important of these standards: M.3010. This standard defines the general TMN management concepts and introduces several management architectures at different levels of abstraction. Section 3.2 discusses TMN’s functional architecture, which describes various management functions. Section 3.3 discusses TMN's physical architecture, which defines how these management functions may be implemented into physical equipment. Section 3.4 discusses one of the best known ideas of TMN: the responsibility model. This model shows how the various management responsibilities can be structured into a convenient arrangement. The analysis is provided in Section 3.5.

3.1 TMN standardization

The TMN standardization was started in 1985 by CCITT Study Group IV [69]. The first TMN recommendation was called M.30 [19] and was published in 1988 as part of the *blue books*. In 1992 a completely revised version appeared and the number of the recommendation was changed into M.3010 [20].

In the 1988-1992 study period, work started on a number of related recommendations (see Figure 3.2). These recommendations define specific aspects of TMN and use M.3010 as the architectural basis.

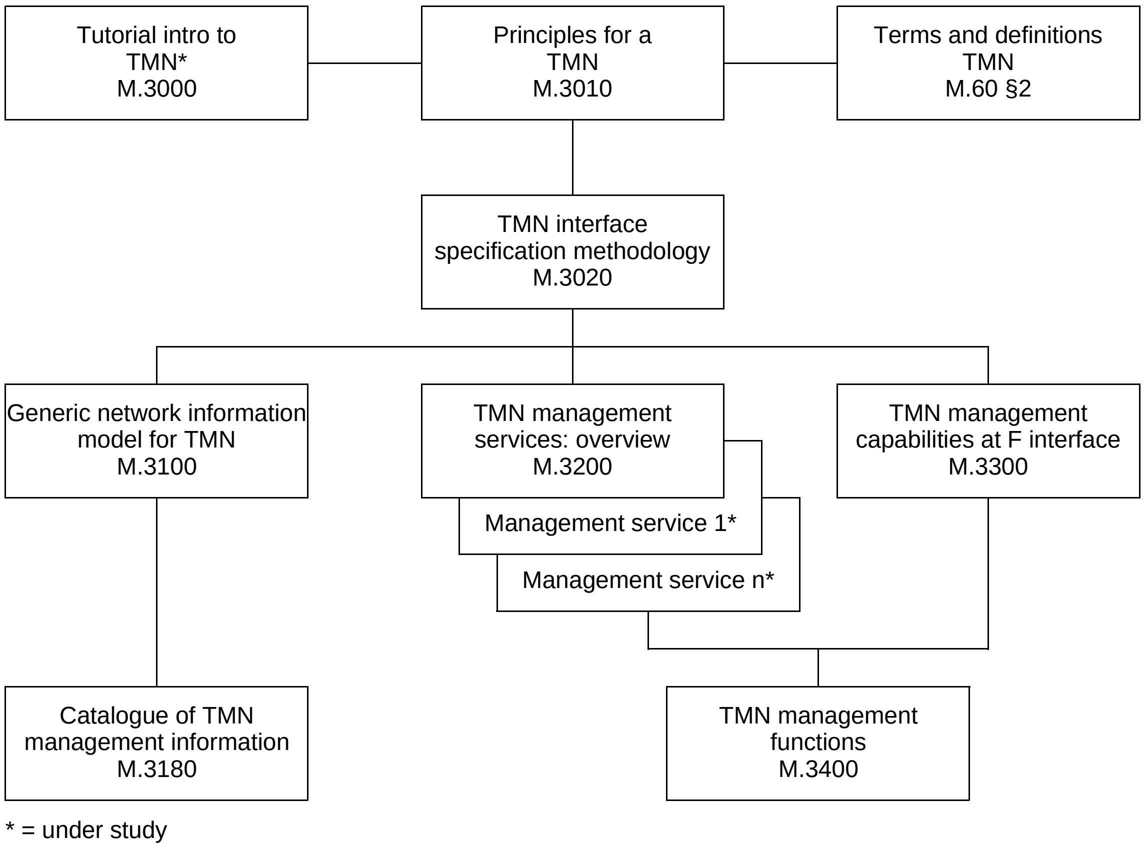

Tutorial intro to TMN* M.3000
Principles for a TMN M.3010
Terms and definitions TMN M.60 §2
TMN interface specification methodology M.3020
Generic network information model for TMN M.3100
TMN management services: overview M.3200
TMN management capabilities at F interface M.3300
Management service 1*
Management service n*
Catalogue of TMN management information M.3180
TMN management functions M.3400
* = under study

Figure 3.2: TMN related recommendations

As compared to the 1988 (blue book) version of M.30, M.3010 has been completely restructured. In M.3010 several sections have been removed from the main text and new sections have been included. Among the sections that have been removed, are those on 'Planning and Design' (which has become an appendix) and 'Functions associated with TMN'. Among the sections that have been included, are those on the 'TMN Information Architecture'.

3.1.1 Relation with ISO/IEC

Initially there was little collaboration between the management groups of CCITT and ISO/IEC. As a result, the 1988 version of Recommendation M.30 had no ISO/IEC counterpart and ISO/IEC standards had little impact on TMN. After publication of M.30 the collaboration between CCITT and ISO/IEC was improved, which resulted in the incorporation of many OSI management ideas into TMN.

The most important changes to TMN were:

- The 'manager-agent' concept, as originally developed by ISO/IEC, was adopted. The current TMN text contains for instance a statement saying that "The description of the manager/agent concept ... is intended to reflect the definitions given in X.701" (the OSI Systems Management Overview).
- ISO/IEC's 'Object Oriented' approach was copied. The current TMN text says: "... the TMN methodology makes use of the OSI systems management principles and is based on an object oriented paradigm".
- The idea of 'Management Domains' was included. A number of TMN drafts that were developed during the 1988-1992 study period contained notes saying: "CCITT SG VII and ISO have a work item on the definition of Management Domains. Resulting material should be used or referenced when available".

Despite this cooperation between the ITU-T and OSI management groups, fundamental differences in philosophy still exist. Members of the ITU management group, for example, prefer to introduce a *separate* network for the transfer of management information. This preference is clearly illustrated in Figure 3.1, which shows that information to manage the *Telecommunication network* should be transferred over a separate *Data Communication Network.*

As explained in Subsection 2.3.2, members of the OSI management group took a different approach. They preferred to use the *same* components for the network that is managed and the network over which management information is transferred[1].

The idea to introduce a separate network to transfer management information is comparable to the idea to introduce a separate network to exchange signalling information. In this sense TMN resembles to SS No. 7 networks.

1. Unfortunately, the standards do not explicitly discuss this difference. The difference becomes clear, however, after discussion with various experts or the study of existing implementations and literature (e.g. [8]).

3.1.2 Recommendation M.3010

Recommendation M.3010 defines three different architectures:
• A functional architecture.
• A physical architecture.
• An information architecture.
TMN's functional and physical architectures will be discussed in Section 3.2 and Section 3.3. TMN's information architecture describes primarily the concepts that have been adopted from OSI management; since the relevant concepts have already been discussed in Chapter 2, the information architecture will not be discussed in this chapter.

Before presenting TMN's functional and physical architecture, the most important TMN concepts will shortly be explained in terms of OSI concepts. Also the relationship between TMN's functional and physical architecture will be explained.

TMN's functional architecture is defined in terms of *function blocks* and *reference points*. Function blocks contain *functional components* (such as 'Presentation Functions' or MIBs) and may be compared to OSI protocol entities. Reference points are used to interconnect function blocks and may in OSI terminology be compared to underlying (management information) service providers (Figure 3.3).

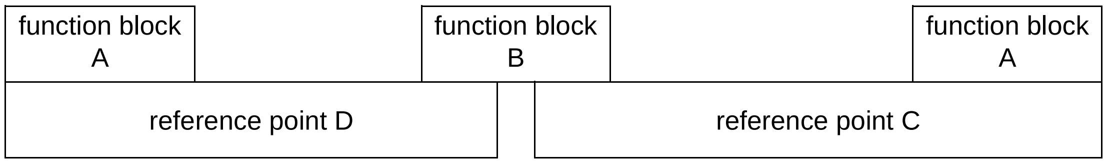


Figure 3.3: Relation between TMN concepts and OSI concepts

TMN's physical architecture is defined at a lower abstraction level. It shows how function blocks can be implemented into *physical equipment* (or *building blocks*[1]) and reference points into *interfaces* (Figure 3.4).

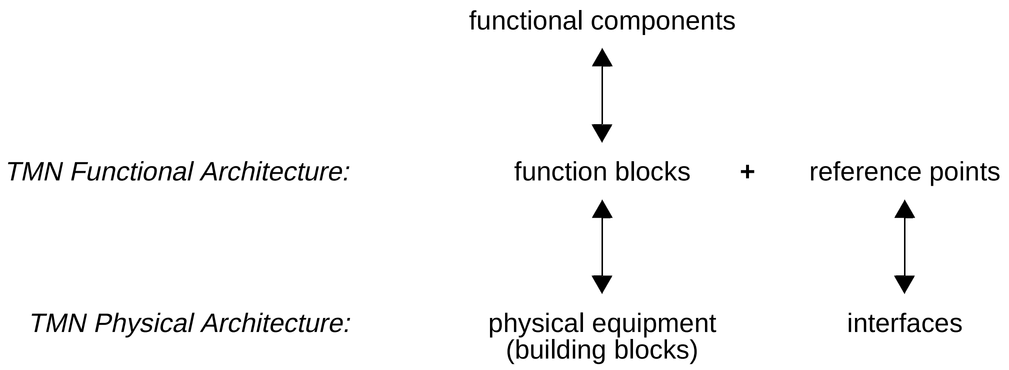

functional components
TMN Functional Architecture:
function blocks
+
reference points
TMN Physical Architecture:
physical equipment (building blocks)
interfaces

Figure 3.4: Relation between TMN Architectures

1. Recommendation M.3010 use the terms *building block* and *physical equipment* as equivalents. In the remainder of this text the term building block will be used.

3.2 Functional Architecture

Five different types of *function blocks* are defined by TMN's functional architecture. It is not necessary that all of these types are present in each possible TMN configuration. On the other hand, most TMN configurations will support multiple function blocks of the same type.

Figure 3.5 has been copied from the TMN recommendations and shows all five types of function blocks[1]. In this figure, two types (OSF and MF) are completely drawn within the box labelled 'TMN'. This way of drawing indicates that these function blocks are completely specified by the TMN recommendations. The other three types (WSF, NEF and QAF) are drawn at the edge of the box to indicate that only parts of these function blocks are specified by TMN. Subsection 3.2.1 until Subsection 3.2.5 give short descriptions these five function blocks.

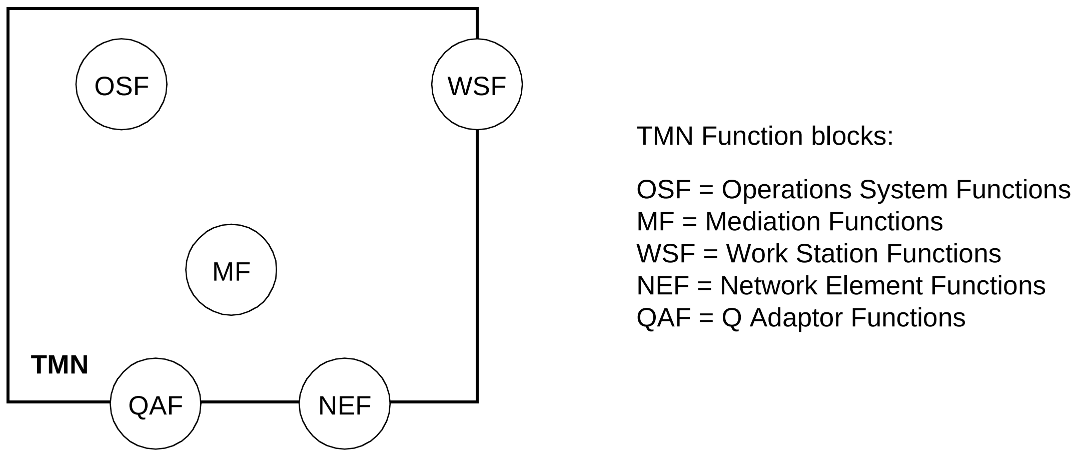

OSF
WSF
TMN Function blocks:
OSF = Operations System Functions
MF = Mediation Functions
WSF = Work Station Functions
NEF = Network Element Functions
QAF = Q Adaptor Functions
MF
TMN
QAF
NEF

Figure 3.5: TMN Function blocks

The TMN functional architecture introduces the concept of reference point to delineate function blocks. Five different classes of *reference points* are identified. Three of them (q, f and x) are completely described by the TMN recommendations; the other classes (g and m) are located outside the TMN and only partially described.

Figure 3.6 provides an example of reference points and function blocks. The picture shows for instance that the Mediation Function (MF) can be reached via *q* reference points and that the *m* reference point can be used to reach the Q Adaptor Function (QAF) from outside TMN.

3.2.1 Network Element Functions

As explained on page 43, a typical telecommunication network consists of *exchanges* and *transmission systems*. In TMN terminology, exchanges and transmission systems are examples of *network elements* (NEs).

1. To avoid adventitious interpretations, it was decided to copy as far as possible drawings from Recommendation M.3010.

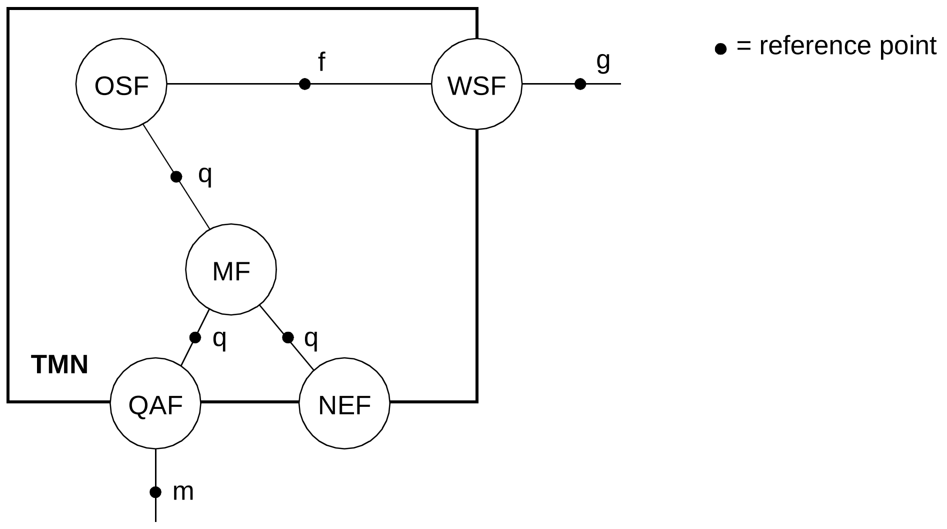


Figure 3.6: Example of reference points between function blocks

The functions that are performed by NEs are 'Network Element Functions' (NEFs). According to TMN, these functions include:

- Primary (or telecommunications) functions. These functions are the subject of management and support the exchange of data between the users of the telecommunication network.
- Management functions, which allow the NEF block to operate in an agent specific role.

As opposed to the second kind, the first kind of functions are not further defined by TMN. This explains why Figure 3.5 locates the NEF at the edge of the TMN.

3.2.2 Operations System Functions

The Operations System Functions (OSF) block initiates management operations and receive notifications. In terms of the manager-agent model, the OSF may be seen as the manager specific functions. An OS communicates with the NEs over a q_3[1] reference point.The service that is provided at such reference point is the *Common Management Information Service* (CMIS [50]).

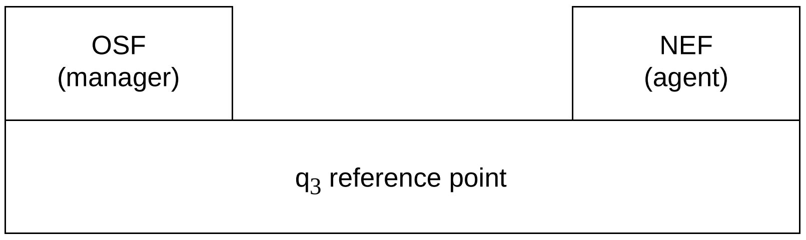


Figure 3.7: Relation between OSF, NEF and q_3

1. The 1988 version of M.30 defined three different q reference points: q_1, q_2 and q_3. After some time it appeared that an acceptable distinction between q_1 and q_2 could not be made. These two reference points were therefore replaced by the generic q_x reference point.

Within a single TMN (operated by a single administration) multiple OSFs may be defined. If necessary, these OSFs can communicate with each other over q_3 reference points. It is also possible that OSFs in different TMNs (operated by different administrations) communicate with each other; in this case communication takes place over a *x* reference points.

3.2.3 Work Station Functions

"The Work Station Function (WSF) block provides the means to interpret TMN information for the management information user. The WSF includes support for interfacing to a human user (at the g reference point). Such aspects of support are not considered to be part of the TMN". Figure 3.5 therefore locates the WSF at the edge, and the g reference point outside the TMN.

3.2.4 Q Adaptor Functions

The *Q Adaptor Function* (QAF) block is used to connect to the TMN those entities which do not support standard TMN reference points. An example is shown in Figure 3.8; in this figure a non-TMN OSF and a non-TMN NEF are connected to the TMN. The responsibility of both QAFs is to translate between *q* reference points (which are TMN reference points) and *m* reference points. Since the m reference point is a non-TMN (e.g. proprietary) reference point, Figure 3.5 showed the QAF at the edge of the TMN.

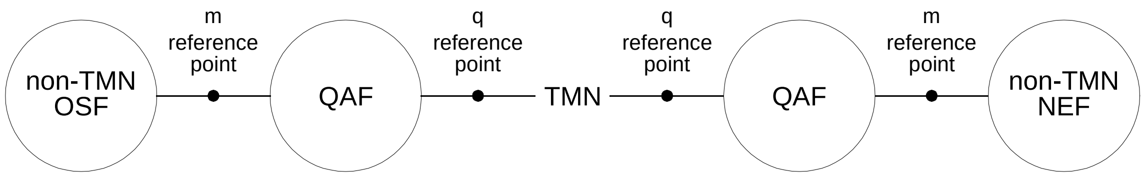


Figure 3.8: Q Adaptor Functions

3.2.5 Mediation Functions

The *Mediation Function* (MF) block is located within the TMN and acts on information passing between NEFs or QAFs, and OSFs. A MF block can be used to connect a single (Figure 3.9), as well as multiple NEFs and QAFs to an OSF. MF blocks can also be cascaded.

Among the types of MFs that can be recognized, are those that:

- Augment OSFs; examples are storage and filtering of management information.
- Augment NEFs; an example is the transformation from the local representation of management information into a standardized form.

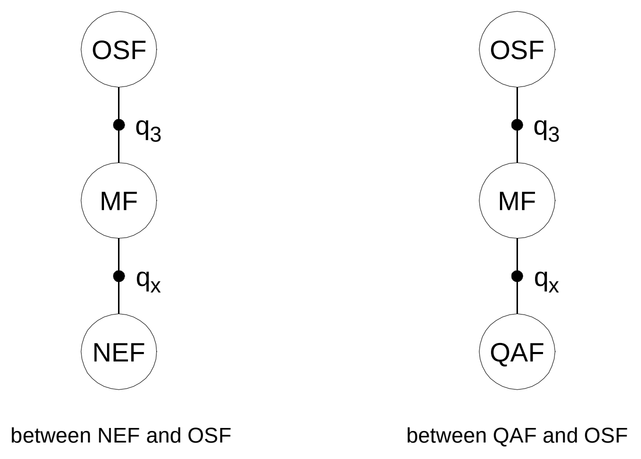


Figure 3.9: MF related to other function blocks

3.2.6 Relationship between function blocks

Now that an initial understanding of all function blocks and reference points exists, it is possible to discern all relationships between these function blocks and reference points. This relationship is given in Figure 3.10.

	NEF	OSF	MF	QAF$_{q3}$	QAF$_{qx}$	WSF	Non-TMN
NEF		q_3	q_x				
OSF	q_3	x*, q_3	q_3	q_3		f	
MF	q_x	q_3	q_x		q_x	f	
QAF$_{q3}$		q_3					m
QAF$_{qx}$			q_x				m
WSF		f	f				g**
Non-TMN				m	m	g**	

m, g = non TMN reference points

* = x reference point only applies when each OSF is in a different TMN
** = The g reference point lies between the WSF and the human user

Figure 3.10: Relation between function blocks

A function block at the top of a column may exchange management information with a function block at the left of a row over the reference point that is mentioned at the intersection of the column and row. In case an intersection is empty, the associated function blocks can not directly exchange management information between each other.

3.2.7 Further remarks

Besides the function blocks and reference points, the TMN functional architecture introduces some additional concepts. These concepts are:
- TMN's Data Communication Function
- TMN's functional components

According to recommendation M.3010, "TMN's Data Communication Function (DCF) will be used by the function blocks for exchanging information. The DCF provides layers 1 to 3 of the OSI RM".
The definition of the DCF concept has historical reasons: in initial drafts of TMN the DCF was modelled as a function block; it was therefore part of TMN's functional architecture. At present the DCF is no longer modelled as a function block; the text that describes the DCF remained, however.

Each of TMN's function blocks is itself composed of a number of *functional components*. The following functional components are defined:
- Management Application Function.
- Management Information Base.
- Information Conversion Function.
- Human Machine Adaptation.
- Presentation Function.
- Message Communication Function (MCF).

These functional components can be divided into two categories:
- The first five components belong to the first category. These components perform the actual management actions; they do not address problems related to the exchange of management information.
- The last component (MCF) belongs to the second category. This component is associated with all function blocks that require an underlying service for the exchange of their management information. "The MCF is composed of a protocol stack that allows connection of function blocks to DCFs". In many cases the MCF provides the end-to-end functions such as those found in OSI layers 4 to 7.

Recommendation M.3010 contains a picture (Figure 3.11) to illustrate the relation between function blocks, functional components, the MCF and the DCF.

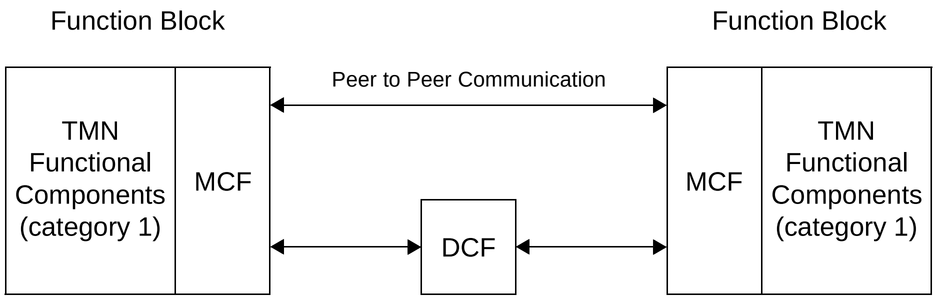

Function Block
Function Block
Peer to Peer Communication
TMN Functional Components (category 1)
MCF
DCF
MCF
TMN Functional Components (category 1)

Figure 3.11: Function blocks, components, MCF and DCF

3.3 Physical Architecture

Next to a functional architecture, TMN also defines a physical architecture. The purpose of the latter is to show how function blocks should be mapped upon *building blocks* (physical equipment) and reference points upon *interfaces*. In fact, the physical architecture defines how function blocks and reference points can be implemented.

To avoid confusion between the functional and physical architecture, it is helpful to understand the following conventions. Names of reference points are written in lower case, names of interfaces in upper case (subscripts may be added). Reference points are drawn as small filled circles (bullets), interfaces as open circles. Function blocks are shown as big circles or ellipses, building blocks are drawn as boxes.

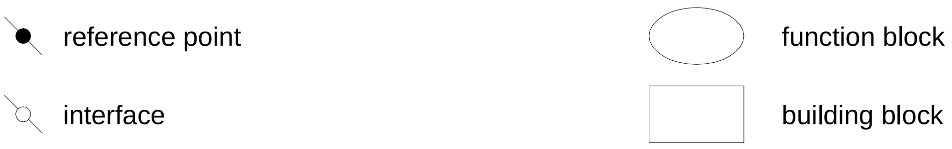


Figure 3.12: Drawing conventions

3.3.1 Building blocks

TMN's Physical Architecture defines the following building blocks:
- Network Element (NE).
- Mediation Device (MD).
- Q Adaptor (QA).
- Operations System (OS).
- Work Station (WS).
- Data Communication Network (DCN).

Building blocks always implement the function blocks of the same name (e.g. Network Elements perform Network Element Functions, Mediation Devices perform Mediation Functions etc.).

It is possible to implement multiple function blocks (of the same or of a different type) into a single building block. The Operations System, for example, may be used to implement multiple OSFs, but may also be used to implement an OSF, MF and a WSF. In the case a building block implements multiple function blocks of different types, "the choice on the building block's name is determined by the predominate usage of the block".
Figure 3.13 shows which function blocks may be implemented into which building blocks.

A special kind of building block is the Data Communication Network (DCN). As opposed to the others, this building block does not implement any function block. In fact, the DCN is *used* by other building blocks for the exchange of management information; the DCN's task is to act as a transport network.

	NEF	MF	QAF	OSF	WSF
NE	M	O	O	O	O*
MD		M	O	O	O
QA			M		
OS		O	O	M	O
WS					M

M = Mandatory
O = Optional
O* = may only be present if OSF or MF is also present

Figure 3.13: Relation between function blocks and building blocks

At first sight it seems strange that TMN defines a building block that does not implement any function block. The existence of the DCN can be understood however when we remember that previous TMN drafts (e.g. [21]) modelled the DCF as a function block (see also Subsection 3.2.7). According to these drafts, the DCF had to be implemented by a DCN and, in that case, each building block implemented at least one function block. In 1990 it was decided however to model the DCF no longer as a function block [22]. After this decision was made, the standard was not rewritten in a consistent way and the DCN is therefore still modelled as a building block.

3.3.2 Interfaces

Interfaces may be regarded as the implementations of TMN reference points. Whereas reference points may generally be compared with underlying *services*, interfaces may be compared with the *protocol stacks* that implement these services.

In most cases reference points and interfaces have a one to one mapping. However, no interfaces exist for those reference points that:

- interconnect function blocks that are implemented within a single building block,
- lay outside TMN (g and m, see Figure 3.6). Implementation of these reference points is outside the scope of TMN.

The naming of interfaces is also straightforward: an interface gets the same name (this time written in upper case) as the related reference point. Figure 3.14 shows all possible mappings.

$$\begin{tikzcd}
\text{reference point} & q_x \arrow[d] & q_3 \arrow[d] & x \arrow[d] & f \arrow[d] & (g & m) \\
\text{interface} & Q_x & Q_3 & X & F & &
\end{tikzcd}$$

Figure 3.14: Mapping reference points upon interfaces

3.4 Responsibility Model

TMN recognizes that, corresponding to human society, a hierarchy of management responsibilities exist. Such hierarchies can be described in terms of management *layers*. This concept of management layers is discussed in the main text of recommendation M.3010 (Logical Layered Architecture). A specific application of this concept, sometimes called the *responsibility model*[1], is given in appendix II of M.3010. This application is considered to be an important aspect of TMN, this section will therefore discuss this model.

The following layers are defined by the model:
- business management layer.
- service management layer.
- network management layer.
- network element management layer.
- network element layer.

These layers, including their function blocks and reference points, are shown in Figure 3.15.

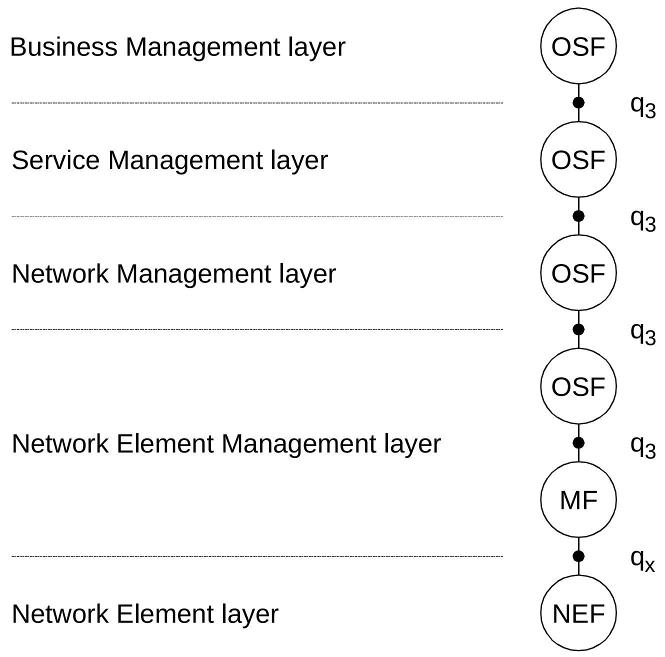


Figure 3.15: TMN Functional hierarchy

The bottom of the management hierarchy is formed by the *network element layer*. This layer contains the Network Element Functions (NEFs). In those cases where the NEFs can only be managed via a q_x reference point, a Mediation Function (MF) is needed at the next higher layer. In terms of a manager-agent relationship, the MF is the manager and the NEF is the agent. The MF will in turn be managed by an Operations Systems Function (OSF) in the same

1. The responsibility model has originally been developed by BT [9] as part of its Open Network Architecture (ONA). BT uses the name *structural architecture* for this model [71].

Network Element Management layer. Examples of functions performed by this management layer are error detection and logging of statistical data.

The Network Element Management layer is responsible for managing NEFs implemented within *single* pieces of equipment. In case the relation between NEFs implemented within *multiple* pieces of equipment becomes important, intervention of an OSF located at the *network management layer* is necessary. Routing can be seen as an example of a management activity located at this layer.

The network management layer, again, is managed by the *service management layer.* Service management is concerned with management of those aspects that may directly be observed by the users of the telecommunication network. An important part of service management is for instance Quality of Service management.

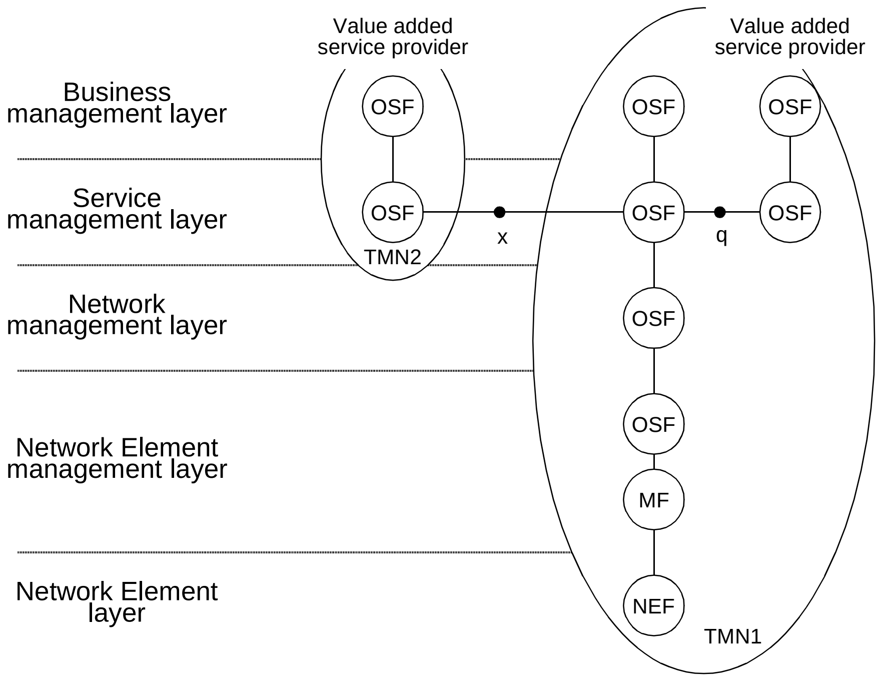

Value added service provider
Value added service provider
Business management layer
Service management layer
Network management layer
Network Element management layer
Network Element layer
OSF
OSF
OSF
OSF
OSF
OSF
OSF
OSF
MF
NEF
x
q
TMN2
TMN1

Figure 3.16: Example of Value Added Services

The idea of service Management is particularly useful in the case of Value Added Services (VAS). In such case one OSF may be responsible for management of the VAS and another OSF may be responsible for management of the telecommunications network. Both OSFs must be able to communicate with each other. If these OSFs belong to the same TMN (administration), communication is realized over a *q* reference point. If both OSFs belong to different TMNs, the *x* reference point will be used (Figure 3.16).

The business management layer is responsible for the management of the whole enterprise. This layer has a broad scope; communications management is just a part of it. Business management can be seen as *goal setting,* rather than *goal achieving.*

3.5 Analysis

The current TMN architecture, and in particular the part on TMN's Information Architecture, includes many ideas of OSI systems management (page 45). As a consequence, the analysis that was given in the previous chapter on OSI management is to a large extent also applicable to this chapter. Despite the large number of similarities between TMN and OSI management, there are also some differences; the most interesting will be examined in Subsection 3.5.1. As opposed to OSI, the concepts that have been developed specifically for TMN are not always properly defined. This is a deficiency, since without good definitions multiple interpretation may arise. Some of these interpretations will be discussed in Subsection 3.5.2.

3.5.1 Differences between TMN and OSI

An interesting difference between OSI and TMN management is that OSI has defined a *single* management architecture whereas TMN defined *multiple* architectures at different levels of abstraction.
Subsection 3.1.2 explained that TMN's functional architecture shows the various TMN management functions and TMN's physical architecture shows how these functions can be implemented into physical equipment (Figure 3.17). TMN's physical architecture is thus defined at a lower abstraction level than the functional architecture (at functional level we abstract from equipment issues, at physical level not).

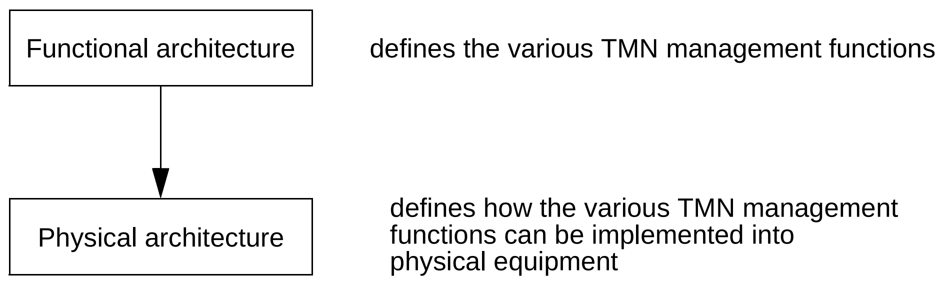

Functional architecture
defines the various TMN management functions
Physical architecture
defines how the various TMN management functions can be implemented into physical equipment

Figure 3.17: TMN has defined multiple, related architectures

In general it may be a good idea to define multiple architectures. This is particularly true in case each architecture elaborates an additional, orthogonal issue. Care should be taken, however, that the relationship between the various architectures remains easy to understand. In the specific example of TMN's functional and physical architecture, this has been the case.

A second difference between TMN and OSI management is, that TMN provides a structure for the multiple levels of management responsibility that exist in real networks; OSI management does not provide such structure. The TMN structure is known as the 'responsibility model' and was discussed in Section 3.4. The advantage of having such structure, is that it becomes easier to understand and distinguish the various management responsibilities.

A final difference between TMN and OSI management is that, as opposed to OSI, TMN suggests a conceptual separation between the network that is managed (the telecommunication network) and the network that transfers the management information (the DCN). This difference was already identified at the end of Subsection 3.1.1.
Such separation prevents the problems with fault management as discussed in the analysis section of OSI management (Subsection 2.3.2). Despite of failures in the managed network, management will always be able to access failing components. TMN has thus better fault management capabilities than OSI.

Unfortunately, a DCN requires the introduction of additional equipment and transmission systems. Besides, failures in the DCN can not be excluded, which implies that it will be necessary to manage the DCN too. The costs of introducing a DCN should therefore not be neglected!

There are also other reasons to introduce a DCN. An important reason may, for instance, be that the managed network does not provide adequate facilities to transfer management information. This is, for example, the case with telephony networks, which provide an isochronous type of service. Such type of service does not correspond to the asynchronous (packet oriented) type of service that is required by most management protocols; a DCN may thus be inevitable to manage such kinds of networks. The better fault management capabilities of the DCN are in such case only a secondary consideration.
As opposed to TMN, OSI is particularly aimed at management of datacommunication networks. The type of service provided by such networks is usually the same as the type of service required for the exchange of management information. With datacommunication networks, and thus in case of OSI, a serious consideration is needed whether the advantages of a DCN outweigh its costs.

3.5.2 Imprecise and ambiguous concepts

As opposed to the OSI management standards, recommendation M.3010 does not include a separate section that clearly *defines* its main architectural concepts (such as function block, reference point, building block and interface). To get an understanding of these concepts, readers have to derive the ideas behind these concepts from the various pieces of text in which these concepts are mentioned. As will be demonstrated in this subsection, different readers may draw different conclusions, depending on the text that has been read.

To proof that different interpretations of TMN's architectural concepts are possible, this subsection explains the *function block* and *reference point* concepts in terms of the relatively well understood concepts of the OSI Reference Model [43]. Readers who are interested in an analysis of the *building block* and *interface* concept are referred to the literature [78].

Function blocks

TMN's functional architecture contains only two pieces of text that explain what function blocks are:

- "Function blocks provide the TMN general function which enable a TMN to perform the TMN management functions".
- "Each function block is itself composed of functional components".

By reading the text on functional components, it becomes clear that most of these components can be compared to OSI application layer functions. It seems reasonable to conclude that function blocks have some relationship to OSI application layer entities.

There is one particular functional component that does not perform application layer functions, but functions of OSI layer 4 until 6. This component is the Message Communication Function, which is "associated with all function blocks". Depending on the meaning of the word *associated*, function blocks possibly also perform layer 4-6 functions.

At the time the Data Communication Function was still modelled as a function block (page 51), function blocks even performed functions that belonged to layers 1-3 of the OSI Reference Model. We may thus conclude that multiple interpretations of the term *function block* are possible.

Reference points

M.3010 gives the following descriptions of TMN reference points:

- "Reference points define service boundaries between two management function blocks. The purpose of reference points is to identify the information passing between function blocks".
- "Reference points are conceptual points of information exchange between non-overlapping management function blocks".

Also in case of reference points, multiple interpretations are possible. Figure 3.18, which is copied from M.3010, shows two of such interpretations.
In case the DCF is explicitly modelled (left part of the figure), each reference point is used to connect the Message Communication Function (MCF) to the Data Communication Function (DCF). Since the MCF provides functions similar to those found in the layers 4-6 of the OSI Reference Model and the DCF provides functions similar to those of layer 1-3, a reference point can be seen as an OSI (network layer) Service Access Point.

In case of an implicit DCF (right part of the figure), a single reference point is sufficient to connect two remote function blocks. In this case, reference points seem to bridge distance and can be seen as OSI (network) service providers.

There is even a third possibility to consider, since reference points may also be used to connect function blocks that are located at the *same* place (these func-

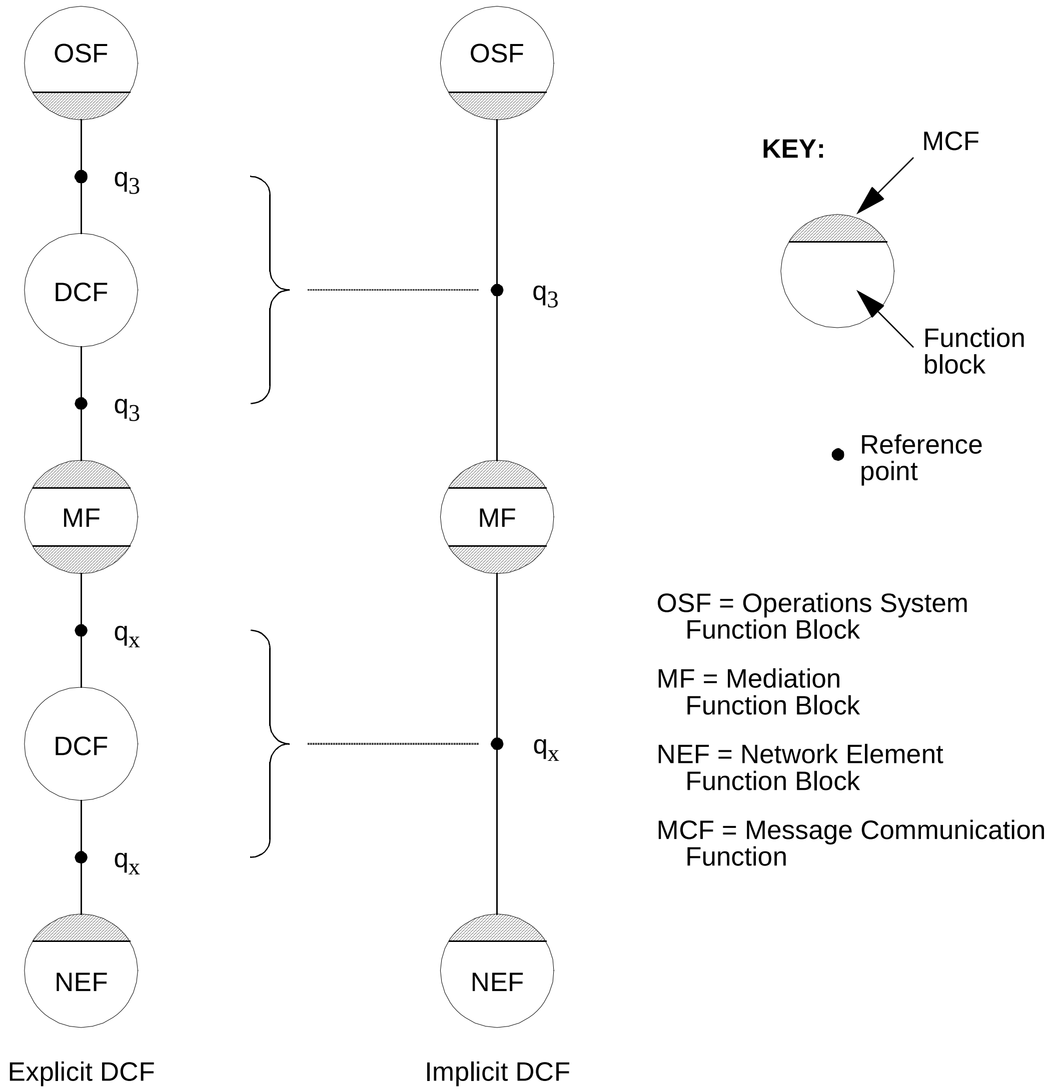

OSF
OSF
q3
DCF
q3
q3
MF
MF
qx
DCF
qx
qx
NEF
NEF
Explicit DCF
Implicit DCF
KEY:
MCF
Function block
Reference point
OSF = Operations System Function Block
MF = Mediation Function Block
NEF = Network Element Function Block
MCF = Message Communication Function

Figure 3.18: Implicit and explicit DCF

tion blocks will later be implemented into a single building block). For this possibility no corresponding OSI concept exists (OSI concepts are not meant to model the internal structure of a system).

4: Internet Management

4 Internet Management

This chapter discusses and analyses the management approach that is standardized by the Internet Engineering Task Force (IETF). This approach is also known as the Simple Network Management Protocol (SNMP) or the TCP/IP management approach.

In the second half of the past decade the Internet grew to a size that management of the Internet could no longer be provided on an ad hoc basis: a structured and standardized approach to Internet management was required.
In 1987 three management proposals therefore appeared. One of these, the *High-level Entity Management System / Protocol* (HEMS / HEMP) was withdrawn soon [86][87], so only two remained: the *Simple Network Management Protocol* (SNMP) and *Common Management Over TCP/IP* (CMOT). At the March 1988 meeting of the Internet board, the decision was made to use SNMP in the short-term and CMOT in the long-term [89].

CMOT [93] was an attempt to use OSI systems management standards (such as CMIP) in the Internet environment [23][70]. CMOT faced the same problems as OSI management: the specifications did not appear in time, there were virtually no implementations and operational experience could not be obtained. As a result, the support for CMOT slowly diminished. In 1992 all work on CMOT was stopped.

SNMP [92] is actually a further development of SGMP (Simple Gateway Monitoring Protocol)[88]. SGMP was aimed at management of Intermediate Systems (gateways)[13]. Because SGMP appeared to be a success, it was decided to extend its scope and include management of End Systems. To reflect this change, the protocol was renamed into SNMP.

An interesting difference between the IETF and ISO is that the IETF takes a more pragmatic and result driven approach than ISO. In the IETF it is for instance unusual to spent much time on architectural discussions; people prefer to use their time on the development of protocols and implementations. This different attitude explains why *no special standards have been defined for the Internet management architecture*; only protocols and MIBs have been standardized. Fortunately many articles and books have been written (even by the editors of the standards) that describe the principles behind Internet management (e.g. [3][14][15][16][101][102]). From these publications the following ideas appear:

- *All* systems connected to the network should be manageable with SNMP.
- The cost of adding network management to existing systems should be minimal.
- It should be relatively easy to extend the management capabilities of existing systems (by extending the Management Information Base).
- Network management must be robust. Even in case of failures, a small set of management capabilities must still be available.

Apparently SNMP was the right solution at the right time. Already a few years after publication of the standard most datacommunication equipment could be managed via SNMP; SNMP had become the de facto standard for management of datacommunication networks. Still SNMP has some deficiencies. In 1992 work was therefore started to develop an improved version of SNMP; this new version was called SNMPv2.

In this chapter both protocol versions will be presented. Section 4.1 discusses the original SNMP protocol, while SNMPv2 will be discussed in Section 4.2. It should be noted that both protocols only define *how* management information should be exchanged; they do not define *which* management information exists. Such information is defined by the various MIB standards; Section 4.3 discusses some of the most important ones. Section 4.4 provides an analysis of Internet management.

4.1 The original SNMP protocol

The ideas behind SNMP are relatively straightforward and easy to understand. In fact, there is little difference between the ideas behind SNMP and the ideas that existed in ISO around 1987, thus before ISO adopted the Object Oriented approach for management. Examples of such common ideas are the manager-agent concept, the idea to use the managed functions also for the exchange of management information, the idea to use *GET* and *SET* PDUs for operations on management information, the idea to use ASN.1 for the definition of management information and the idea of a MIB.

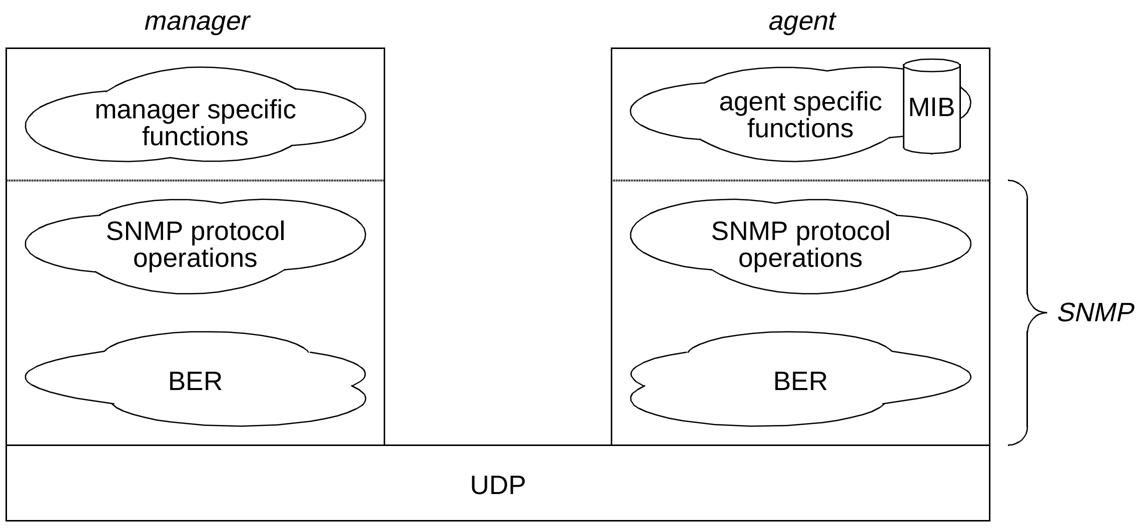


Figure 4.1: Internet management structure

With SNMP, a single manager may control many agents. As shown in Figure 4.1. the SNMP protocol is built upon the User Datagram Protocol (UDP), which is a connectionless transport protocol [83]. Since the Internet management information as well as the formats of SNMP PDUs are defined according to (a subset of) the ASN.1 syntax, encoding functions are needed immediately on top of UDP. These functions operate according to the Basic Encoding Rules

(BER). Five types of SNMP PDUs are defined: *GetRequest*, *GetNextRequest*, *SetRequest*, *Response* and *Trap*. The SNMP protocol standard does not address the functions that are specific for managers or agents; the SNMP standard is thus restricted to the functions below the dotted line (Figure 4.1). This implies that the scope of SNMP is equivalent to the one of CMIP; as opposed to CMIP no standard exists, however, which defines the service that is provided on top of SNMP.

The IETF has not (yet) defined *manager specific* functions; further work in this area is therefore urgently needed [101]. The lack of manager specific functions sharply contrasts to the overwhelming number of agent specific functions (primarily MIBs) that have been defined. Section 4.3 gives an overview of these MIBs.

4.1.1 Transport mappings

The choice to operate SNMP over the connectionless UDP has several implications.

In the first place UDP is unreliable, which means that user data may get lost. The decision to use an unreliable transport service provider, has been taken deliberately. The reason is that even in case of repeated provider failures, it should still be possible to exchange *some* part of the management information. With a reliable (connection-oriented) provider this may not be possible. Connection-oriented providers are designed according to an ‘all or nothing’ approach: either *all* data will be delivered or *nothing* will be delivered. If data can not be delivered, the connection will be released. Connectionless providers are designed according to an ‘best-effort’ approach: even in case of failures some of the data may arrive at the destination. Management may therefore still be possible, although in a limited way.

It is interesting to see that the SNMP protocol does not itself perform retransmissions. The responsibility to detect data loss and initiate retransmission is left to the manager, because it is assumed that managers are usually better equipped to determine *whether* and *when* retransmissions are required.

A second implication of using a connectionless transport protocol, is that managers should perform some kind of *polling* to detect whether agents are still operational. With connection-oriented providers (e.g. OSI’s presentation service) this would not be necessary, because such providers already include lifetime control functions[1]. Such functions periodically check whether the remote systems (in our case the agents) are still operational. In case they went down, the provider takes the initiative to release the connection and informs the user (in our case the manager).

1. In OSI, this function is part of the connection oriented transport protocol.

A characteristic of UDP is that packets can not exceed a certain size. To ensure that only limited size packets will be generated, the SNMP protocol has defined a number of rules. One of these rules is that, if the response to a certain SNMP request would exceed the maximum packet size, no information will be returned at all. Managers should be aware of this rule and, instead of issuing a single all-embracing request, issue multiple smaller request to get the information piece by piece. Unfortunately, managers will in many cases not be able to predict the amount of information that can be obtained via a single request.

Although SNMP is intended to operate over UDP, there are also RFCs that define how to operate SNMP on top of other protocols (e.g. Ethernet, IPX or even OSI).

4.1.2 Protocol operations

In SNMP, communication from the manager to the agent system is performed in a confirmed way. The SNMP entity at the manager's side takes the initiative by sending one of the following PDUs: *GetRequest*, *GetNextRequest* or *SetRequest*. The *GetRequest* and *GetNextRequest* are used to retrieve management information from the agent, the *SetRequest* is used to store (or change) management information. After reception of one of these PDUs, the SNMP entity at the agent's side responds with a *Response* PDU (Figure 4.2). This PDU carries the requested information or indicates failure of the previous request.

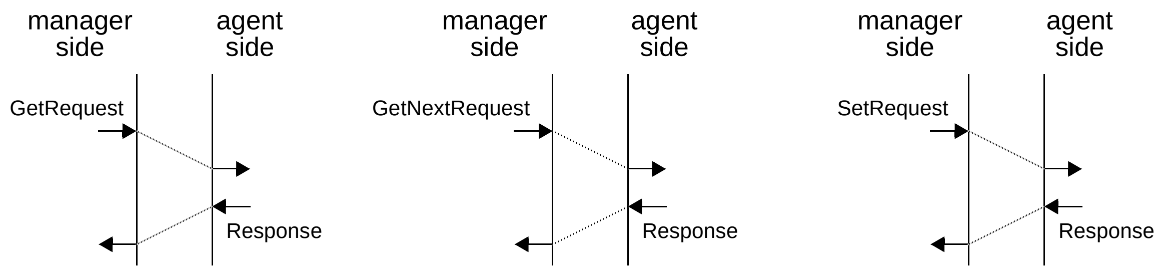


Figure 4.2: Managing system takes the initiative

It is also possible that the SNMP entity at the agent's side takes the initiative. This happens in case the agent detects some extraordinary event, such as a re-initialization or a status change at one of its links. As a reaction, the agent's SNMP entity sends a *Trap* PDU to the managing system (*Traps* may be compared to OSI *Event-Reports*). Reception of the *Trap* is not confirmed (Figure 4.3).

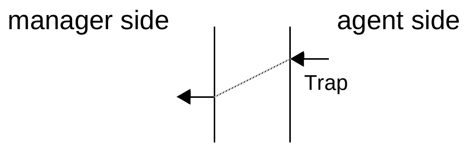


Figure 4.3: Agent system takes the initiative

SNMP does not describe how to relate the various *Get*, *Set* and *Trap* interactions. What to do after reception of a *Trap* is, for example, not defined by SNMP. Instead, determination of this relationship is considered to be a responsibility of the manager specific functions.

4.2 SNMPv2

Since publication of the original SNMP protocol, several proposals have been presented to improve SNMP. In 1992 it was decided to collect these proposals and produce a new standard: SNMPv2. Unfortunately SNMPv2 became far more complex than the original SNMP [79]; whereas the description of the original SNMP protocol required, for example, only 35 pages, the description of SNMPv2 required about 250 pages[1].

The main achievements of SNMPv2 are the improved performance (Subsection 4.2.1), the better security (Subsection 4.2.2) and the possibility to build a hierarchy of managers (Subsection 4.2.3).

4.2.1 Performance

As explained on page 64, the original SNMP protocol includes a rule which states that if the response to a *Get* or *GetNext* request would exceed the maximum size of a packet, no information will be returned at all. Since managers can not determine the precise size of response packets in advance, they usually take a conservative guess and request per PDU just a small amount. To obtain all information, managers may be required to issue a large number of consecutive requests.

To improve performance, SNMPv2 introduced the *GetBulk* PDU. As opposed to the *Get* and *GetNext*, the response to the *GetBulk* always returns as much information as possible. If the requested information exceeds the maximum size of an UDP packet, the information will be truncated and only the part that fits within the packet will be returned.

4.2.2 Security

The original SNMP protocol had, except for a simple mechanism which involved the exchange of passwords (the term ‘community string’ was used to denote this password), no security features. To solve this deficiency, SNMPv2 introduced a full-fledged security mechanism. This mechanism is based upon the use of ‘parties’ and ‘contexts’; two concepts that can not be found in other management approaches. Although the SNMPv2 standards include definitions of both concepts, these definitions are difficult to understand. This subsection presents a somewhat simplified view of these concepts.

Parties have some resemblance to protocol entities. Usually multiple parties are active in a single SNMPv2 subsystem and these various parties will be configured in different ways. One party may, for instance, be configured such that it is prepared to communicate with every other party in every other system.

1. These numbers do not include the pages describing the Structure of Management Information.

Another party may be configured such that it is only prepared to interact with one particular remote party. In such case, the MD5 authentication mechanism is used to ensure the authentication of the other party. Finally parties may be configured in a way that they are only prepared to interact with particular remote parties and in addition require that all management information is encrypted according to the DES algorithm.

A graphical representation of parties is provided in Figure 4.4. In this figure three parties have been configured in the manager system (Pa1, Pa2 and Pa3) and three parties in the agent system (Pb1, Pb2 and Pb3).

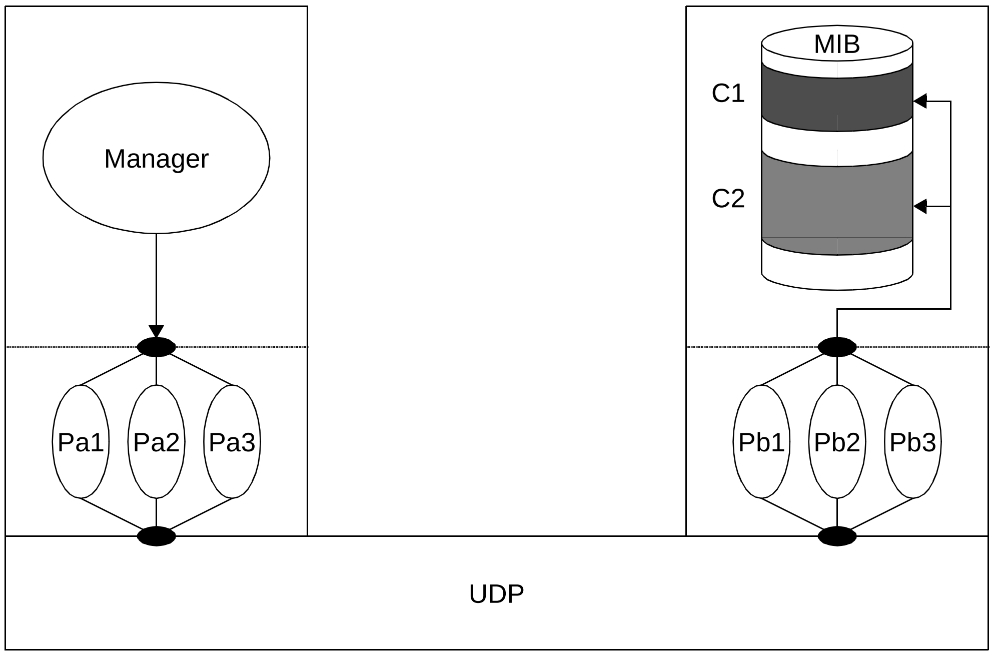


Figure 4.4: Parties and contexts

To control access to the various parts of a MIB, SNMPv2 has introduced the *context* concept. Each context refers a specific part of a MIB. In the example of Figure 4.4, context C1 and context C2 refer to the two dotted areas in the MIB. Contexts may be overlapping and are dynamically configurable, which means that contexts may be created, deleted or modified during the network's operational phase. Different contexts may be configured for different systems.

remote party	local party	context	operation
Pa1	Pb1	C1	get
Pa2	Pb2	C1	get
Pa3	Pb3	C1	get + set
Pa3	Pb3	C2	get

Figure 4.5: Example of an Access Control List (ACL)

To determine which parties are allowed to perform which operations upon which part of the MIB, SNMPv2 has associated with each agent an Access Control List (ACL). Figure 4.5 shows an example of such a list. The first row indicates that party `Pa1`(in the manager system) may perform *Get* operations via party `Pb1` (in the agent system) on that part of the MIB that is identified by context `C1`. The third row shows that `Pa3` may via `Pb3` also perform *Set* operations on this MIB part.

4.2.3 Management hierarchy

Practical experience with the original SNMP protocol showed that in many cases managers are unable to manage more than a few hundred agent systems [3]. The cause for this restriction is in SNMP's polling nature: the manager must periodically poll every system under his control, which takes time. To solve this problem, SNMPv2 introduced the idea of *intermediate level managers*. Polling is now performed by a number of such intermediate level managers under control of the top level manager.

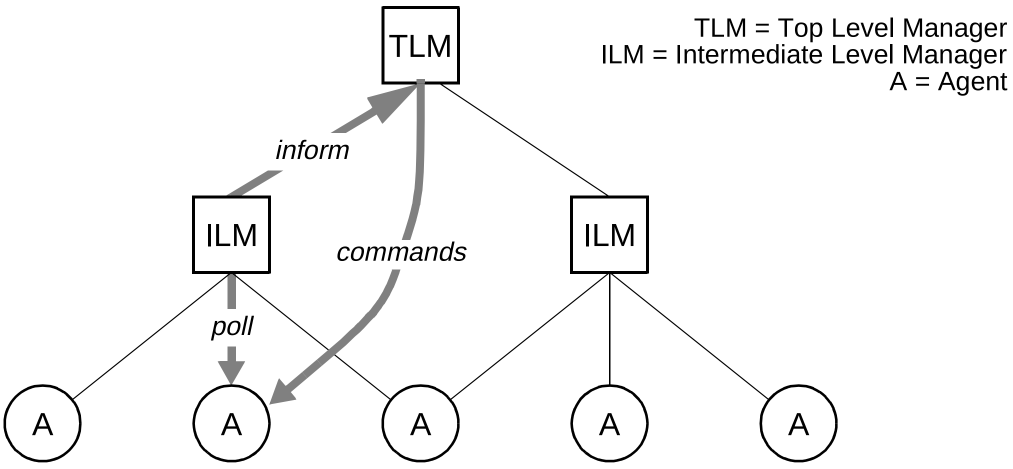


Figure 4.6: Intermediate Level Managers

Figure 4.6 shows an example. Before the intermediate level managers start polling, the top level manager tells the intermediate level managers which variables must be polled in which agents. Besides, the top level manager tells the intermediate level managers of the events he wants to be informed about. After the intermediate level managers are configured, they start polling. In case an intermediate level manager detects in a particular agent an event about which the top level managers wanted to be informed, a special *Inform* PDU is generated. After reception of this PDU, the top level manager directly operates upon the agent that caused the event.

4.3 MIBs

To identify all variables that can be managed, a large number of Management Information Base (MIB) standards have been developed. Next to these standards, one special standard exists defining *how to describe* MIB variables. This standard is called the Structure of Management Information (SMI). It defines for instance the subset of ASN.1 constructs that can be used to describe management variables [90].

To ensure the unique identification of each management variable, the SMI introduces the concept of a naming tree. The leaves of this tree represent the actual management information. An (imaginary) example of this is shown in Figure 4.7; in this figure the object identifier of the network address is *root.1.1*, the object identifier of the collision counter is *root.2.2* and the identifier of the token holding timer is *root.3.4*.

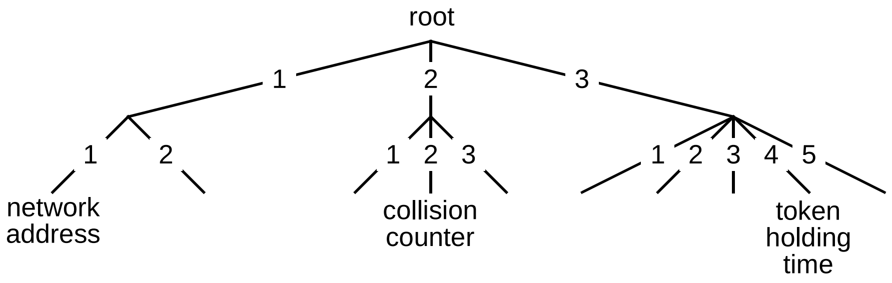


Figure 4.7: Concept of naming tree

The MIB-II[1] [94] is the most important and probably best known MIB; it contains all the variables to control the major Internet protocols (e.g. IP, ICMP, UDP, TCP, EGP and SNMP). The structure of this MIB is simple: all management variables that belong to the same protocol are grouped together (Figure 4.8). Within a protocol group there is hardly any additional structure that helps understanding the various variables within that group.

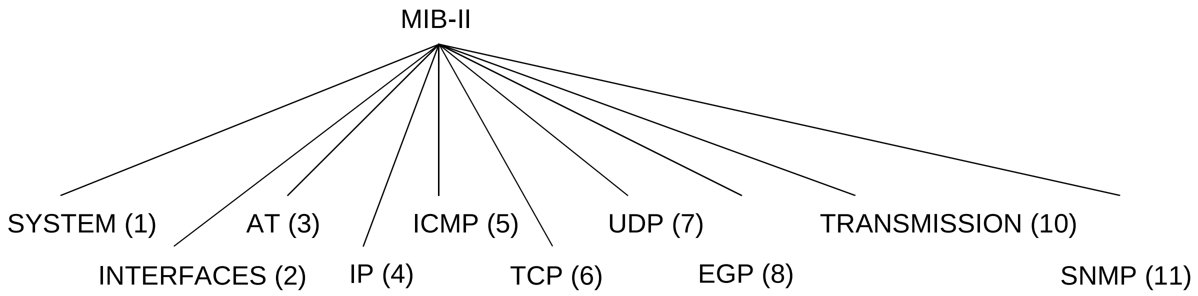


Figure 4.8: The various protocol groups of the MIB-II

Soon after definition of the MIB-II other MIBs appeared; Figure 4.9 shows some of the standardized ones.

Next to the standardized MIBs there are also a large number of enterprise specific MIBs. Together these MIBs define more than twenty-thousand management variables [63]. Unfortunately no clear structure has been developed to explain the relationship between these MIBs; the only indication of a MIBs purpose is its name.

1. The suffix II was added to indicate that this MIB replaces the earlier defined 'Management Information Base for management of TCP/IP based internets' [91].

Title	RFC	Date
MIB-II	1213	March 1991
IEEE 802.5 Token Ring	1231	May 1991
Appletalk	1243	July 1991
OSPF version 2	1253	August 1991
Remote Network Monitoring	1271	November 1991
IP Forwarding Table MIB	1354	July 1992
RIP Version 2	1389	January 1993
DS1 and E1 Interface Types	1406	January 1993
DS3 and E3 Interface Types	1407	January 1993
X.25	1461	May 1993
Point-to-Point Protocol	1471-1474	June 1993
Bridges	1493	July 1993
FDDI	1512	September 1993
Remote Network Monitoring - Token Ring	1513	September 1993
Host Resources	1514	September 1993
IEEE 802.3 Medium Attachment Units	1515	September 1993
IEEE 802.3 Repeater Devices	1516	September 1993
Source Routing Bridges	1525	September 1993
DECnet Phase IV Extensions	1559	December 1993
Network Services Monitoring	1565	January 1994
Mail Monitoring	1566	January 1994
X.500 Directory Monitoring	1567	January 1994
SNA APPN Node	1593	March 1994
SONET/SDH Interface	1595	March 1994
Frame Relay Service	1604	March 1994
Domain Name System	1611-1612	May 1994
Uninterrupted Power Supply	1628	May 1994
Ethernet-like Interface Types	1643	July 1994
Border Gateway Protocol	1657	July 1994
Character Stream Devices	1658	July 1994
RS-232-like Hardware Devices	1659	July 1994
Parallel-printer-like Hardware Devices	1660	July 1994
SNA NAU	1666	August 1994
SMDS - SIP Interface Type	1694	August 1994
ATM	1695	August 1994
Modem	1696	August 1994
Relational Database Management System	1697	August 1994

Figure 4.9: Some existing MIB definitions

4.4 Analysis

Internet management can be compared to OSI management. In fact, Internet management uses many of the concepts that existed in OSI at the time SNMP started (around 1988). As a result, the remarks that were made in the analysis section of OSI management (Section 2.3) are to some extent also applicable to Internet management. As opposed to OSI management, however, Internet management uses only a small part of the managed functions for the exchange of management information. Problems with fault management (see Subsection 2.3.2) are therefore less likely to occur.

4.4.1 The management architecture has not been described

No standards have been produced defining the Internet management architecture. To get an understanding of the architectural concepts behind Internet management, readers have to derive the meaning of the various concepts from the protocol standards. Although this may not be a problem in case of the original version of SNMP, it certainly is a problem with SNMPv2. Experience has shown that without a good understanding of these concepts, it is difficult to implement SNMPv2 [79]. A recommendation for the IETF is therefore to develop such a standard.

The concepts that cause most of the problems, are parties and contexts. Although the interpretation of these concepts given in this thesis (Subsection 4.2.2) may be sufficient to understand most parts of the SNMPv2 standards, certain parts of the standards are based upon some other interpretation. A good example of such alternative interpretation can be found in the standards that define how to use intermediate level managers. According to these standards a context does not only refer to the specific part of a MIB, but also identifies one of the agents that is controlled by the intermediate level manager.

4.4.2 Too many management variables

Now that thousands of management variables have been defined, the lack of a good functional structure to classify these variables has become a problem. Without such structure, managers will be confronted with large lists of management variables. To determine which variables must be watched and which modifications must be made, managers must understand the precise meaning of many variables.

In case management is performed by human beings, it is unlikely that there will be many people with sufficient ready knowledge. As a consequence, it can be expected that managers need a lot of time before they decide what to do. Network management may therefore become a time-consuming and thus expensive activity.

4.4.3 Manager specific functions have not been defined

The Internet management standards explain how individual management operations, such as *GET* and *SET*, should be performed. Currently they do not specify, however, the sequence in which these operations should be performed to solve particular management problems. Such sequences are part of the 'manager specific functions' (see Figure 4.1); until now the IETF has not defined such functions.

> *Example:* Suppose a router breaks down. Actions must be initiated by management to prevent data from getting lost. These actions include the change of routing tables. Since networks consist of thousands of systems, management must decide which tables to change and which not. Of course, management must also specify the exact contents of the modified routing tables.
> Internet management standards do not describe any of these actions. Instead, Internet management provides only a general approach to read and modify individual management variables.

The approach that is taken by Internet to manage networks is comparable to an approach in which debuggers are used to 'manage' computer programs.
Ordinary debuggers allow programmers to watch and modify program variables. A debug program does not help, however, to determine which variables must be watched and which modifications must be made. Such decisions must be made by the programmer; the debugger only helps to access the variables.
Internet management standards define *distributed* 'debuggers'. These 'debuggers' allow managers to watch and modify management variables; they do not tell which variables must be watched and which modifications must be made. Such decisions must be taken by the 'manager specific functions' (e.g. the operator); Internet management standards only tell how to access management variables.

Part II

An Alternative Approach to Network Management

5: Identification & classification of management

5 Identification & classification of management

The purpose of this chapter is to *identify* and *classify* potential management functions. To find management functions, the network's design process will be considered. Within such process the designer elaborates the network's primary functions. As will be demonstrated in this chapter, this elaboration introduces a number of management problems. To solve such problems, management functions must be introduced.

There are several models that describe the design process. In this chapter the step-wise design model will be considered. This model is easy to understand and allows the identification and classification of management functions in a structured way[1]. Three *classes* of management functions will be introduced, each class belonging to a particular phase of the design process. In should be noted that this chapter does not discuss how to *design* management functions, such discussion can be found in Chapter 6.

The structure of this chapter is as follows.

- Section 5.1 provides an introduction to step-wise design. This section recognizes three important phases: the architectural phase, the implementation phase and the realization phase.
- Section 5.2 discusses management functions that can already be identified during the architectural phase. Since the outcome of this phase will be a service specification, the term 'service management' will be introduced to denote these functions.
- Section 5.3 discusses management functions that can be identified during the implementation phase. To denote these functions, the term 'protocol management' will be introduced.
- Section 5.4 discusses management functions that can be identified during the realization phase. The outcome of this phase are individual network systems, such as switches and terminals. In the telecommunication world these systems are commonly called 'network elements'. The term 'element management' will therefore be introduced to denote these management functions.
- Section 5.5 shows that the distinction between service, protocol and element management may not only be useful to classify management functions during the design phase, but may also be used to structure the various management operations during the operational phase. As will be shown, such structured approach may be beneficial for fault detection purposes.
- Section 5.6 provides the conclusions.

1. Note that literature describes several other approaches to structure management. An interesting approach is for instance to consider the boundaries between different management authorities and introduce management domains. Each domain may be considered as a group of managed objects to which the same management policy applies [107][108]. The idea of domains and policies is currently investigated by multiple groups and will not be discussed in this thesis.

5.1 Introduction to step-wise design

The starting point of a step-wise design process is the definition of user requirements (See also page 5). In most cases these requirements are written in a natural language without any particular structure. According to Bogaards “user requirements may be incomplete and inarticulate and insofar they are recorded they may be vague, imprecise, ambiguous, redundant and sometimes even contradictory” [7].

The user requirements are transformed via a number of steps into a realization (Figure 5.1). There is no general rule stating how many steps are needed: that decision is made by the designer and depends amongst others upon the complexity of the design.

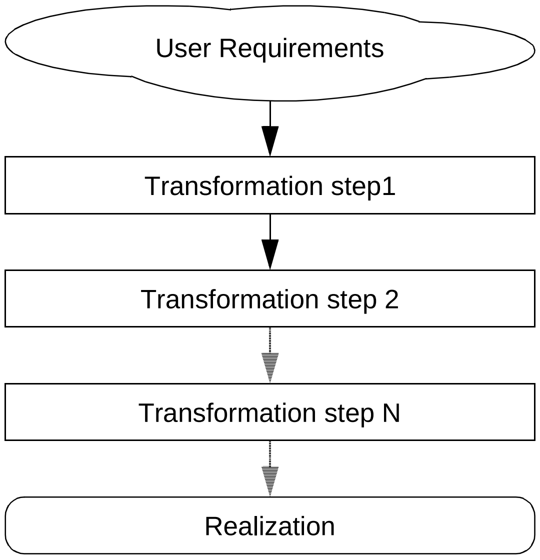


Figure 5.1: Step-wise design process

5.1.1 Phases in a step-wise design

Although the exact number of design steps varies from case to case, several theories [5][10] propose to distinguish between the following phases (Figure 5.2):

- Architectural phase: this phase starts from the user requirements and results into the intermediate specification called the ‘architecture’.
- Implementation phase: this phase starts from the architecture and results into one or more implementations. At this level, the internal structure of the system is defined. The fact that a single architecture can be implemented in different ways, is shown in the figure by means of multiple arrows. A good example that multiple implementations can be derived from a single architecture, is given by Digital Equipment. This company has developed many implementations (e.g. 11/03, 11/04, 11/34, 11/60, 11/70) from the single PDP-11 minicomputer architecture.

- Realization phase: this phase starts from the implementation and results into one or more realizations. At this level, functions are mapped upon hard- and software components. There are many possibilities to perform these mappings: functions may for instance be implemented in TTL logic, VLSI chips, micro controllers etc.

The realization phase is the last phase of the design process; after the realization is ready the operational phase may be started.

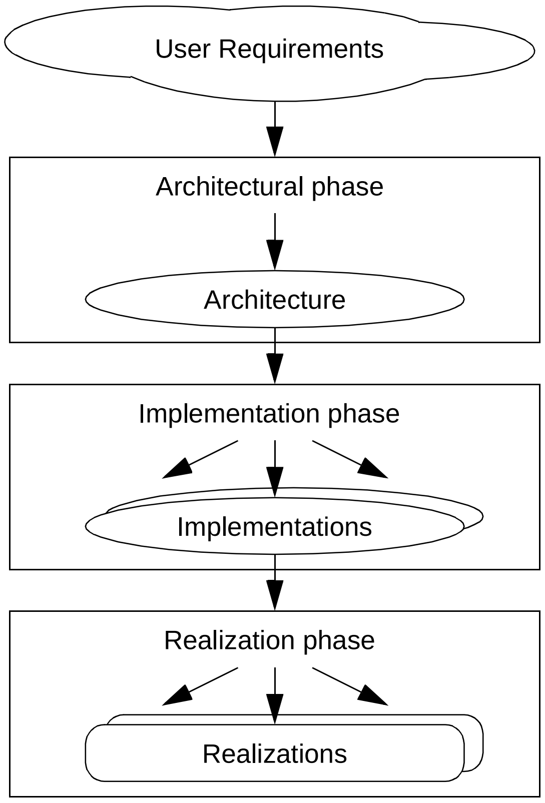


Figure 5.2: Phases in the design process

5.1.2 Step-wise design and distributed systems

The theory that was described in the previous subsection can be applied to the specific case of distributed systems design (Figure 5.3). In such case the phases are usually described as follows:

- Architectural phase: this phase starts from the user requirements and results into the intermediate specification called the 'service'. The service defines the common behaviour of users and provider; it hides the internal structure of the provider from the user. This implies that, at architectural level, the provider is considered as a black-box with a number of interaction points.
- Implementation phase: this phase starts from the service specification and results into protocol specifications. At this level, the internal structure of the distributed system is defined. This can be done in a number of 'horizontal and vertical structuring steps'.

- Realization phase: this phase starts from the protocol specifications and results into one or more realizations. At this level, protocol functions are implemented in hard- and software. We may expect that different manufacturers follow different paths towards their realizations. From a single protocol definition different realizations are therefore possible.

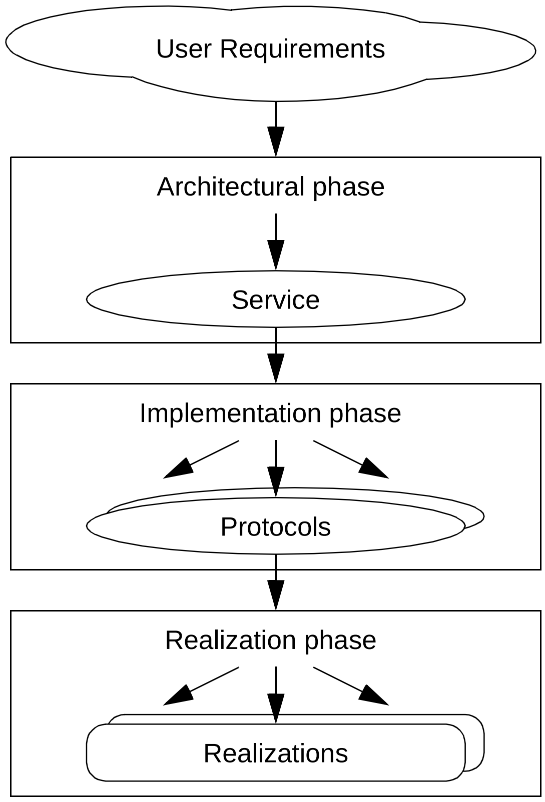


Figure 5.3: Phases in the design of distributed systems

5.2 Management issues in the architectural phase

To support the primary functions that are defined during the architectural phase, it may be necessary to introduce some management functions.

To give an idea of such management functions, Subsection 5.2.1 provides some examples. After an understanding of these management functions has been obtained, Subsection 5.2.2 discusses some general characteristics. To distinguish these management functions from other management functions, Subsection 5.2.3 defines the concept of service management. The section concludes with some general remarks concerning complexity and standardization of service management (Subsection 5.2.4).

5.2.1 Examples

Management functions may be introduced in the architectural phase to *initialize* and *modify* service characteristics, but also to *obtain information* from the service provider.

Initialization

Services, as defined by standardization organizations, generally describe the possibility to exchange user data between SAPs. The way in which individual SAP addresses should be assigned, is usually not described by these service definitions. Such initialization functions can be considered as management functions.

Modification

Service definitions usually concentrate on primary functions and ignore the question of how to connect and remove users from the provider. That question is usually considered to be a management problem.

There are several reasons why users should be connected or removed from the network. Connection to the network may be demanded in case new users appear. Removal from the network may be necessary in case users refuse payment of their bills or show some other form of misbehaviour (e.g. hackers).
In the case of networks for mobile communications, it is also conceivable that users request temporary removal of themselves. Such requests may be a matter of money: even in case users do not participate in connections, they may be charged. This is because networks for mobile communications exchange user specific signalling information, even in case no connections are made.

Obtaining information

Management functions may, for example, also be introduced to monitor the QoS. QoS figures can be useful to determine whether sufficient capacity is available to connect new users or to determine that QoS has dropped below the negotiated level, in which case billing may be adapted.

5.2.2 General characteristics

The examples in the previous subsection showed that service definitions usually include descriptions of the primary functions, but lack descriptions of management functions. Since primary functions specify only parts of the provider's behaviour, management functions should be added to complete the description of this behaviour (Figure 5.4).

The management functions that can be identified during the architectural phase, have a direct relationship to the user requirements. This can be under-

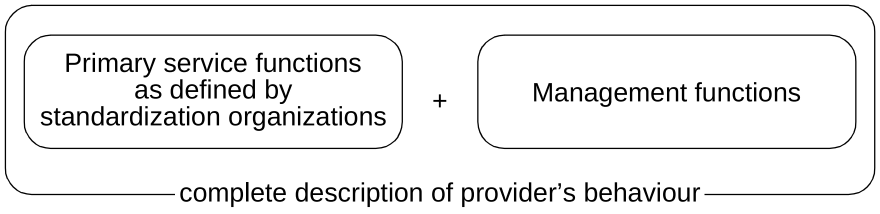

Primary service functions as defined by standardization organizations
+
Management functions
complete description of provider's behaviour

Figure 5.4: Complete description of the provider's behaviour

stood if one considers the problems that lead to the introduction of these management functions:

- Networks are usually not designed as dedicated systems that satisfy the requirements of one specific set of users, but as general purpose systems that have the potential to support different sets of users. Before a network can be used by a certain set of users, it must be initialized. The purpose of such initialization is to adjust the network's behaviour to the requirements of a particular set of users. The term 'provisioning' is sometimes used for such initialization [82]. Tasks that may be performed during the initialization phase are: the connection of users to the network, the assignment of SAP addresses, the definition of group addresses, the establishment of closed user groups etc.
- User requirements usually change in time. Tasks that may be performed to cope with these changing requirements are: the connection of new users to the network, the removal of existing users from the network, the modification of group addresses and closed user groups, the modification of cost parameters etc.
- Users and provider may not always behave in the way that has been negotiated. Functions must therefore be added to ease the detection of potential misbehaviour. Examples of such functions are: monitoring of activities that take place at a certain SAP, request QoS figures, perform reachability tests etc.

It may be noted that these three reasons correspond to what has been discussed in the subsection on 'why management' (Subsection 1.2) under the headings: cost reduction, flexibility and faults.

In the architectural phase no knowledge is needed of the provider's internal structure. This implies that also for the identification of management functions the provider may be considered as a black-box. The identification of management functions will for instance not be effected by the choice whether the provider should internally use the OSI ConnectionLess Network Protocol (CLNP) or the Internet Protocol (IP).

As a consequence, the management functions that may be identified during the architectural phase can not be used to manipulate individual components within the provider, such as for example routing tables.

5.2.3 Definition of service management

To distinguish the management functions that are identified in the architectural phase from other management functions, the term ‘service management’ is introduced.

> *Definition:* Service management is that part of management that is responsible for the proper interaction between users[1] and provider. The object of service management is the service provider. Service management functions can be initiated by service users and allow the initialization, modification and observation of the provider’s behaviour.

The OSI and Internet management architectures have not defined the service management concept, even though some of their MIBs include objects that can be used to manage the interactions between user and service provider.

As opposed to OSI and Internet, the TMN management architecture did define service management (see Section 3.4). Still there are certain differences between service management as defined in this theses, and service management as defined by TMN. These differences come from the fact that this thesis uses, in accordance to the OSI Reference Model, the term ‘service’ in a specific sense. As opposed to TMN who uses this term in a general sense, this thesis does for example not always regard applications and information servers as services.

5.2.4 Concluding remarks

The fact that service management considers the service provider as a blackbox implies that service management should be easier to comprehend than the management functions that can be identified during the implementation and realization phase.

Because service management can be defined independent of a particular implementation or realization, the possibility exists to develop service management standards.

This section concludes with a remark concerning a potential misinterpretation of the term ‘service management’. Consider the example of an implementor who decides to add functions *within the provider* that monitor the provided QoS for the purpose of optimizing the routing tables. If these monitoring functions are performed invisible for the service users (e.g. there is no possibility for users to manipulate these functions) these functions should (according to the definition) not be considered as service management functions.

1. It should be noted that this thesis uses the term ‘user’ in the sense of the OSI Reference Model. According to this interpretation, users may be considered as functional entities on top of underlying service providers. In telecommunication standards, the term ‘user’ is frequently used as equivalent to ‘end-user’ or ‘customer’.

5.3 Management issues in the implementation phase

The aim of the implementation phase is to develop one or more protocol definitions. These definitions usually describe the primary functions that must be performed by protocol entities. To support these functions, it will generally be necessary to add management functions.

To illustrate which management functions may be added, Subsection 5.3.1 starts with some examples. Characteristics of these management functions will be presented in Subsection 5.3.2. To distinguish these management functions from other management functions, Subsection 5.3.3 defines the concept of protocol management. Subsection 5.3.4 discusses the relation between protocol management and service management.

5.3.1 Examples

The management functions that can be identified during the implementation phase, are needed to support the primary protocol functions. In this subsection some primary functions of a hypothetical network layer protocol will be considered to identify examples of management functions.

Assume the network layer protocol performs the following primary functions:

- Forwarding. In routers the forwarding function determines the outgoing link over which packets should be forwarded. The implementation of this function can make use of forwarding tables (Figure 5.5). The destination address which is contained in each packet, serves as input to this table. The table is searched, until an entry is found that matches this destination address. This entry associates the destination address with an outgoing link.

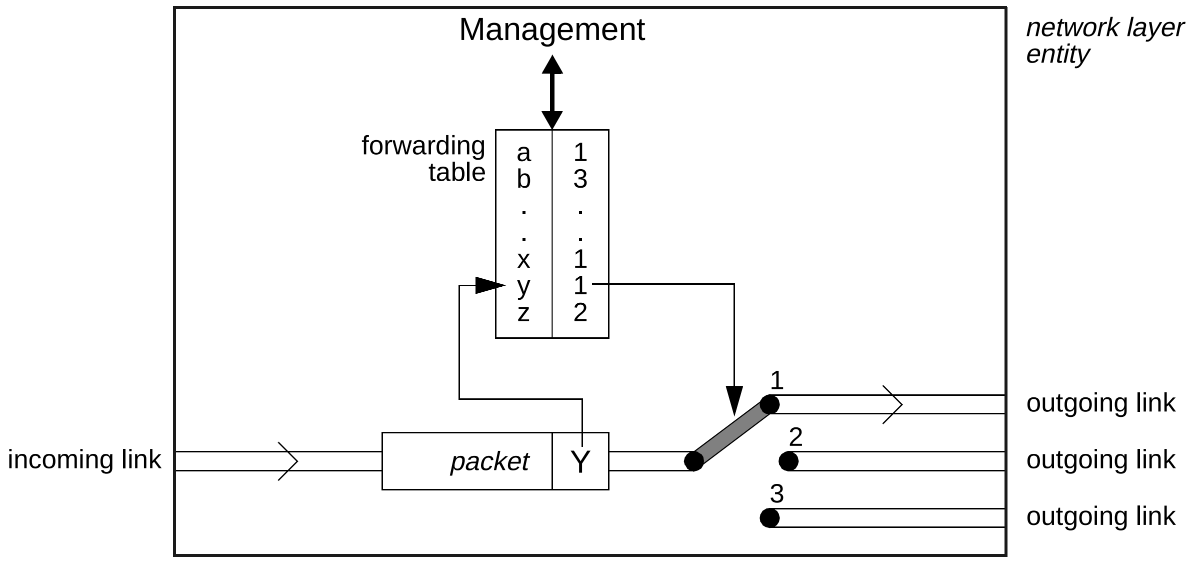


Figure 5.5: Forwarding within a network layer entity

Protocol definitions usually do not specify how to initialize and maintain the forwarding table. Nevertheless initialization and maintenance of this table is necessary to allow the network to become and remain operational. How this should be done, is often considered to be a management problem.

- Flow control. A good example of a flow control mechanism, is the window mechanism. This mechanism uses a ‘maximum window size’ parameter, which must be set by management. How this should be done, is usually not specified by the protocol definition.
- Congestion control. In case of congestion, it may be a good idea to drop certain packets. When and which packets should be dropped, is usually not specified by the protocol definition. It is the responsibility of management to solve this question.
- Segmentation and reassembly. The segmentation function may be activated in case the size of a network layer PDU exceeds the maximum that is supported over an outgoing link. The implementation of this function may be based upon a table that contains for each outgoing link the maximum size that is supported. Initialization and maintenance of this table may be considered as a management function; protocol definitions usually do not specify this function.
- Error correction. A possible error correction mechanism is retransmission. Examples of variables that may be used by this mechanism, are the time-out variable and the variable that indicates the maximum number of retransmission attempts. Setting these variables is usually considered to be a management responsibility.

The examples demonstrated the need to extend primary protocol functions with functions to *initialize* and *modify* variables. It may also be necessary, however, to introduce management functions to *obtain* the values of protocol variables.

> *Example:* Users may request that packets be transferred over the links that show the lowest error rates. In this case management, thus the functions that maintain the forwarding table, must be able to obtain error values for each possible link.

5.3.2 General characteristics

The management functions that are added to support the primary protocol functions have the following general characteristics:

- They can be identified as part of the design process of the primary protocol functions, thus during the *implementation phase*.
- They affect the operation of *protocol entities* (this is a consequence of the fact that the primary protocol functions are performed by protocol entities). In fact, the management functions initialize, modify and observe information within these entities.

- The management functions belong to the same protocol layer as the primary protocol functions. This is a logical consequence of the fact that for the definition of these management functions detailed knowledge of the primary protocol functions is needed. Such knowledge is not available at higher or lower protocol layers.
- The management functions can be described in a way that abstracts from possible realizations. It is therefore possible to standardize these management functions.

As opposed to service management, protocol management deals with problems that relate to the internal structure of the provider. These problems may be much more complex then the problems service management is faced with.

5.3.3 Definition of protocol management

To distinguish the management functions that can be identified in the implementation phase from other management functions, the term 'protocol management' is introduced.

> *Definition:* Protocol management is that part of management that is responsible for the proper operation of a particular protocol. Protocol management functions support the primary protocol functions. Protocol management functions allow the initialization, modification and observation of entities within a protocol layer.

Most of the current management literature is on protocol management. Protocol management is defined by OSI, TMN and Internet.

5.3.4 Relation with service management

The implementation relation that exists between service and protocol functions is shown in Figure 5.3. An interesting question is whether a similar implementation relation exists between service management and protocol management: should protocol management functions be regarded as the implementation of service management functions or not?

To answer this question, two more general questions will be considered:
- How can primary service functions be implemented?
- How can service management functions be implemented?

Implementation of primary service functions

Primary service functions may be mapped upon primary protocol functions, but also upon protocol management functions. This can be explained by means of the following example.

Consider the primary service function that describes how to exchange user data between SAPs. During the implementation of this function, forwarding functions may be introduced. Such forwarding functions can be considered as primary protocol functions. As explained earlier, forwarding functions may make use of forwarding tables. To initialize and maintain these tables, protocol management functions must be added too.

The implementation of primary service functions thus involves primary protocol functions as well as protocol management functions.

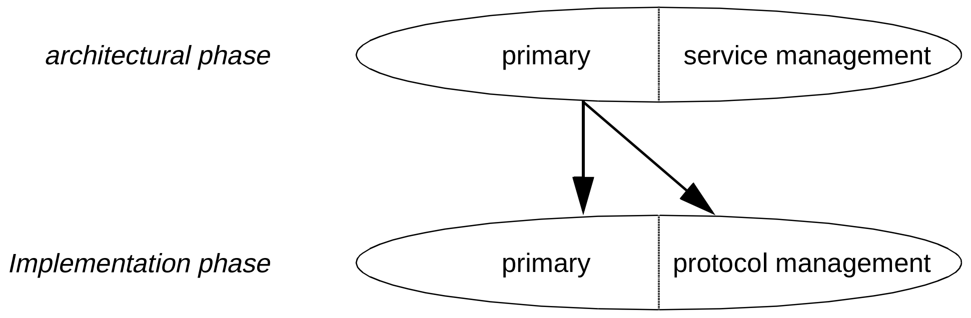


Figure 5.6: Implementation of primary service functions

Implementation of service management functions

Service management functions can also be implemented in different ways. To demonstrate this, two examples will be given:

- In the first example, the service management functions will be mapped upon primary protocol functions.
- In the second example, the same service management functions will be implemented in a different way and mapped upon protocol management functions.

The service management function that will be used for both examples, is modification of SAP addresses. Consider the case of a service provider with three SAPs: a, b and c (Figure 5.7). Assume the address b should be changed into x.

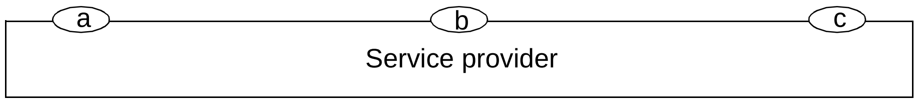


Figure 5.7: Service provider with three SAPs

First example:
Suppose the service provider can be decomposed into three protocol entities (one per SAP) plus an underlying service provider with broadcast capabilities (Figure 5.8). This structure allows for a straightforward implementation of the service management function: before b changes the SAP address, it broadcasts a special *Check* PDU to verify the uniqueness of the new address. If the other entities do not raise objections, b carries out the actual address change and notifies the other entities of this change via (for instance) a *Commit* PDU.

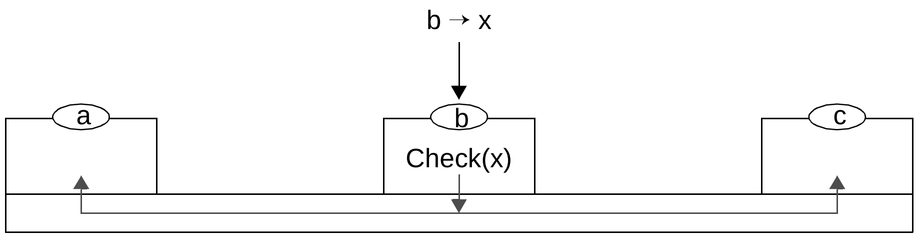


Figure 5.8: Implementation involves a 'Check' PDU

With this kind of implementation, the protocol function that takes care of the address change can be developed in isolation from other protocol functions. It will therefore be pointless to associate this address change function to any of the primary protocol functions; the function that changes the SAP address is not needed *to support* any of the primary protocol functions. This implies that the function that changes the SAP address does not satisfy the requirements of *protocol management* (see page 84); the function should therefore be regarded as a primary protocol function.

Second example:
Suppose the service provider can be decomposed into three End Systems (ES) plus one Intermediate System (IS). Each ES is connected to the IS via a point-to-point link. The IS performs a forwarding function and uses a forwarding table. In case the address b is changed into x, the forwarding table should be adjusted. As discussed previously, modification of the forwarding table may be considered as a protocol management function.

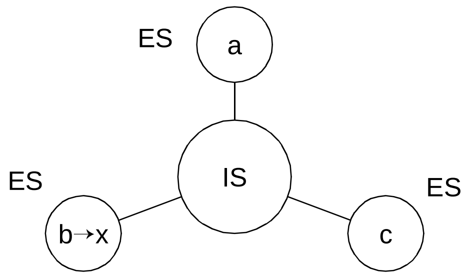


Figure 5.9: Implementation involves modification of the forwarding table

In this case, the implementation of the service management function involves the addition of protocol management functions.

Conclusion:
The two examples above indicated that service management functions can be implemented as primary protocol functions, as well as protocol management functions (Figure 5.6). Thus there is not necessarily a one to one relationship between service management and protocol management functions.

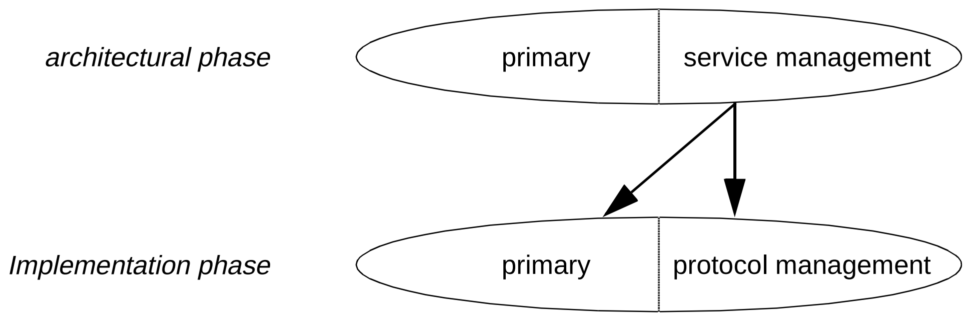


Figure 5.10: Implementation of service management functions

5.4 Management issues in the realization phase

To allow observation, initialization and modification of the components that realize individual network systems, management functions may be needed. To give an idea of the realization components that should be managed, Subsection 5.4.1 starts with some examples. The examples are followed in Subsection 5.4.2 by a discussion of some general characteristics of these management functions. In Subsection 5.4.3 the concept of element management will be defined.

5.4.1 Examples

Individual network systems can be realized in hardware and / or in software. Both realization forms introduce their own management problems.

Hardware implementations require management functions to observe, initialize and modify hardware modules.
Items that are observed in many hardware implementations, are power lines. The idea behind this is that changes in power consumption may indicate failure of hardware components. Since hardware failures are more likely with increasing temperatures, several hubs already include functions that read the current temperature and inform the manager in case this temperature exceeds a pre-defined level.
A common initialization task is to allocate interrupt vectors and I/O addresses.
Modifications may involve replacement of hardware boards, but also the update of firmware. Flash memories are nowadays widely used to accommodate such firmware changes.

Software management also involves observation, initialization and modification. In case network software is implemented on top of a multi-processing operating system (e.g. UNIX), the following items may be observed: process status, process priority, CPU time, foreground memory, swap space etc.
Some of these items require some initial assignment and may also be modified after the system has become operational. Examples of these are priority and memory use.

5.4.2 General characteristics

A primary protocol may be realized by different manufacturers in different ways. Realizations are thus vendor specific. This implies that also management of these realizations will be vendor specific. As a result, standardization should be restricted to some common parts, such as how to structure and transfer management information. The decision which items can be managed (e.g. the contents of the MIB), can only be taken by the individual manufacturers. Agreement upon such MIBs is unlikely, since the differences in these MIBs allow manufacturers to distinguish their products and thus obtain a higher share of the market.

It is interesting to note that the IETF has recognized the need to develop enterprise specific MIBs. For this purpose, the IETF has included a dedicated branch in the naming tree. A special name for this form of management has not been defined by the IETF, however.

5.4.3 Definition of element management

To distinguish the management functions that can be identified during the realization phase from other management functions, the term ‘element management’ will be used.

> *Definition:* Element management is that part of management that is responsible for the proper operation of the realization components that are used within individual network systems. Element management functions include the initialization, observation and modification of these physical components.

The concept of element management has also been defined by TMN; the ISO and Internet management architectures did not define this concept. In TMN, the concept is called: ‘network element management’.

5.4.4 Relation with protocol management

Analogous to service and protocol management, there need not necessarily be a one to one relationship between protocol and element management. Primary as well as protocol management functions may therefore be realized as primary, but also as element management components.

5.5 Relevance for the operational phase

In the previous sections, three classes of management functions were identified: service, protocol and element management. Each of these classes relates to one of the three stages of the step-wise design process. In this section it will be demonstrated that the distinction between service, protocol and element management may not only be relevant during the design phase, but that this distinction may in some cases also be useful during the operational phase. In this section it is assumed that a layered architecture has been used for the design of the network.

To explain how service, protocol and element management can be applied in the operational phase, a structured approach to fault detection will be presented. It should be noted that this approach is not intended to replace traditional fault management approaches, but to *supplement* traditional approaches[1]; it is for instance still desirable to monitor the operation of the crucial network components on a regular basis.

This structured approach to fault detection can be illustrated as follows:

1) Use service management to find the protocol layer that causes the problem.
2) Use protocol management to find the (sub)system within that layer that causes the problem.
3) Use element management to find the realization component within that system that causes the problem (Figure 5.11).

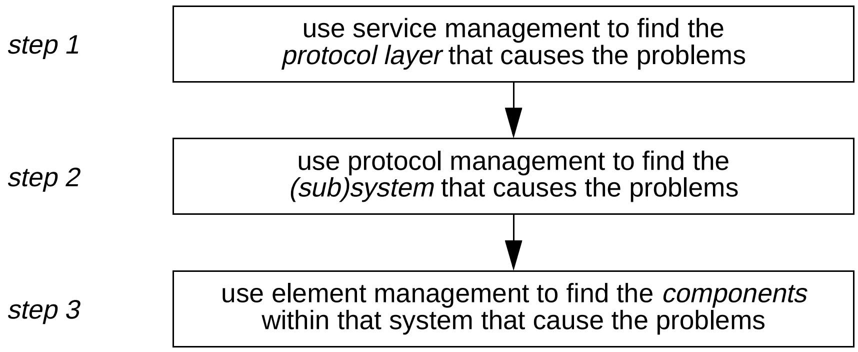

step 1
use service management to find the *protocol layer* that causes the problems
step 2
use protocol management to find the *(sub)system* that causes the problems
step 3
use element management to find the *components* within that system that cause the problems

Figure 5.11: Structured approach to fault management

Step 1: use service management

According to the structured approach, the manager should first trace the protocol layer that causes the problem. This can be done by checking the behaviour of each of the underlying service providers in succession. Such checking can be performed without knowledge of the provider's internal structure; it can therefore be based upon service management. If the outcome of the check shows that the provider performs well, the problem must be caused by a higher layer protocol.

The provider that is checked first (see note below), may be the one that is located immediately below the user's application. The outcome of this check may be:

1. Several other fault management approaches have been described in literature (e.g. [35], [38], [64], [67] and [68]).

- The provider performs well. Lower level service providers need therefore not be investigated and the problem must be caused by the protocol layer immediately above this provider. Step 1 can be closed.
- The provider does not perform well. The problem is thus located within this service provider. Step 1 repeats, but this time the next lower service provider will be checked.

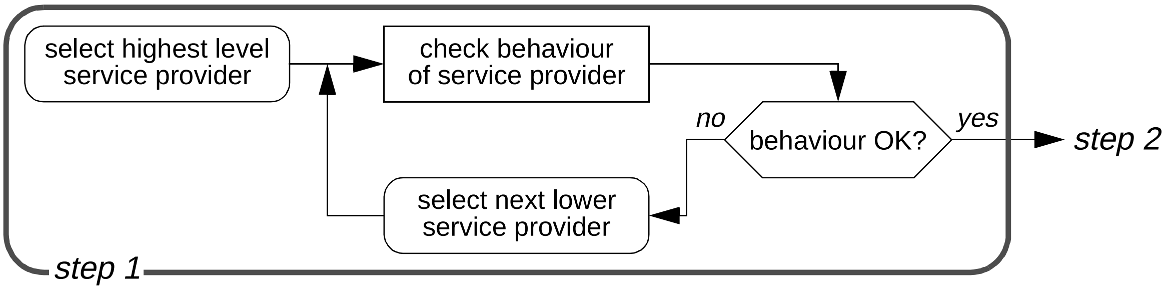

select highest level service provider
check behaviour of service provider
no
behaviour OK?
yes
step 2
select next lower service provider
step 1

Figure 5.12: Find protocol layer that causes problems

Figure 5.12 shows the successive checking of underlying service providers. The loop stops after a service provider is found that performs well. The protocol that uses this underlying provider must be the one that caused the problem and will be investigated in step 2.

Note: step 1 starts with the service provider immediately below the user's application, and not the lowest service provider. This can be understood by realizing that additional knowledge may be required to determine the lowest (e.g. physical and data link) service providers that are used by the application. Obtaining this knowledge may not be easy, since it requires insight in the network's topology and an understanding of the algorithm that is used to forward user data. The approach that requires the least amount of knowledge is therefore the one that start from the highest level service provider, thus the one located immediately below the application.

Step 2: use protocol management

After the protocol *layer* that causes the problem has been determined, the specific *entity* that causes the problem should be determined. In case the protocol is a point-to-point protocol, only two entities must be checked. In case of a network layer protocol, all entities between source and destination must be checked. To find these entities, it may be necessary to consult the forwarding tables or use possible 'route recording' functions.

Step 3: use element management

After the *entity* has been found that causes the problem, the *components* that realize this entity must be checked. To perform such check, knowledge of the specific realization must be available.

Example

Consider the case of an e-mail problem. Figure 5.13 shows the successive steps that *can* be performed by the manager to solve such problem. The figure assumes that the problem is caused by a malfunctioning component within one of the network routers.

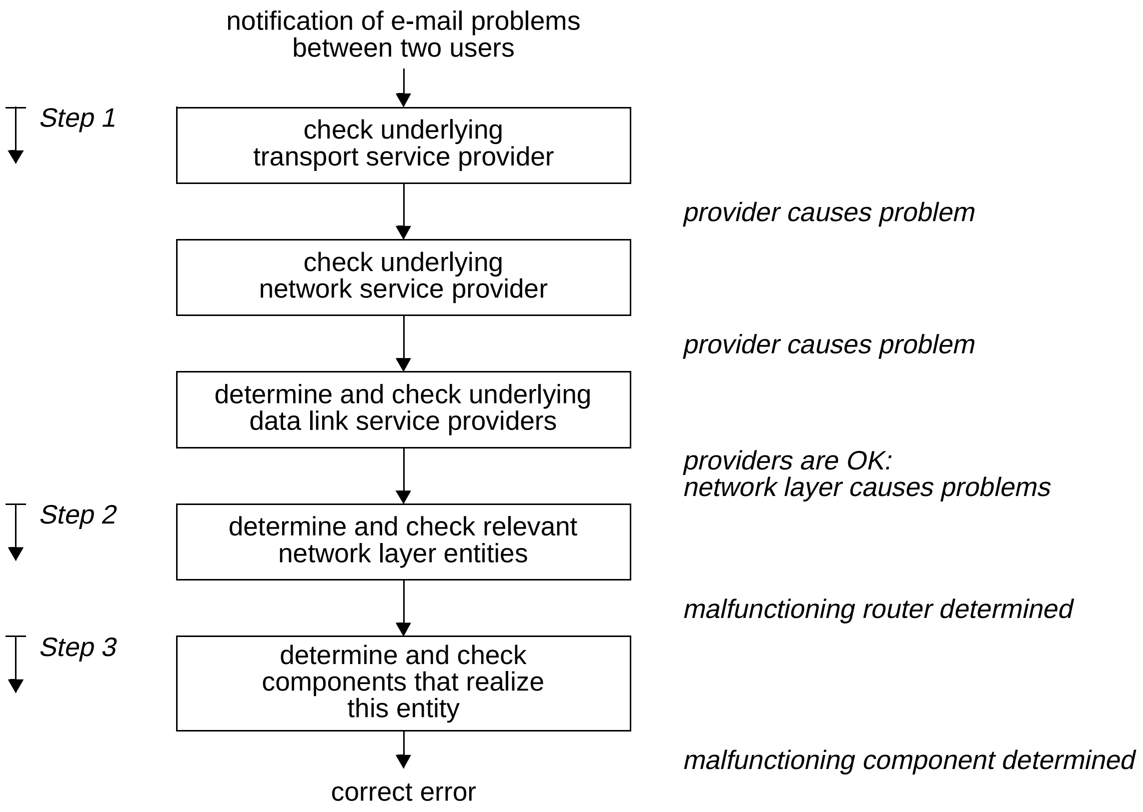

notification of e-mail problems between two users
Step 1
check underlying transport service provider
provider causes problem
check underlying network service provider
provider causes problem
determine and check underlying data link service providers
providers are OK: network layer causes problems
Step 2
determine and check relevant network layer entities
malfunctioning router determined
Step 3
determine and check components that realize this entity
malfunctioning component determined
correct error

Figure 5.13: Example of fault management

OSI and Internet management do not recognize the idea of service management. As a consequence, these approaches do not support step 1 of our structured approach. Instead, the current OSI and Internet MIBs confront managers immediately with all details of the protocols (see also Subsection 4.4.2).

5.6 Conclusions

This chapter showed that management problems could be identified within all stages of the step-wise design process. This conclusion is interesting, since the idea to define management functions at service level has not been recognized by the OSI and Internet management groups.

In this chapter, three classes of management functions were identified: service, protocol and element management. The distinction between these classes allowed us to improve our understanding of (Figure 5.14):

- Which management issues arise at which stage of the design.
- Which management issues are visible to the network users, and which issues require knowledge of the provider's internal structure.
- Which management issues are vendor specific.
- Which management issues may, and which management issues may not be standardized.

	design step	visible to user	vendor specific	standardization
service	architecture	yes	no	yes
protocol	implementation	no	no	yes
element	realization	no	yes	no

Figure 5.14: Differences between service, protocol and element management

6: Design of management functions

6 Design of management functions

This chapter discusses the design of management functions. The objective of this discussion is to show that management functions can be developed together with primary functions as part of the same design process. For this purpose it is assumed that the design is performed in a cyclic fashion, which implies that a *'cyclic design model'* can be applied.

The structure of this chapter is as follows. Section 6.1 gives a short introduction to the cyclic design model. Section 6.2 through Section 6.4 apply this model to explain when service, protocol and element management functions should be elaborated during such design. Since the design of element management functions is the easiest to explain, it will be discussed first. Service management will be discussed last.

In Section 6.5 the case is treated that during the development of management functions additional management problems may appear. To resolve these problems it may be necessary to introduce additional management functions that manage the initial management functions.

Section 6.6 illustrates that it may not always be possible to complete the definition of all management functions during the design phase. In several cases it may be possible, however, to reconsider the unresolved management problems as part of the design of future generations of network systems. To include the design of these management functions, the cyclic design model will be extended to incorporate the development of such future generations. In fact, the model that emerges may be considered as a combination of the life cycle model and the cyclic design model.

Section 6.7 provides the conclusions.

6.1 Cyclic design

To provide a common basis for the remainder of this chapter, a short introduction to the cyclic design model will be given. Further information on this model can be found in literature [6][7][34][104].

The idea of cyclic design is to distribute design problems over a sequence of design cycles, allowing to exploit design experience obtained in previous cycles while performing the next.

Figure 6.1 shows an example of a cyclic design process. In this example, three design cycles are performed in sequence. During the first design cycle only a subset of the user requirements will be considered. During the second design cycle additional requirements will be considered, and during the last design cycle all requirements will be taken into account. The experience that is obtained in each of these cycles, is used as input to the subsequent cycles. In this example, each cycle comprises all phases of the step-wise design process: architectural, implementation and realization phase. The realization that satisfies all user requirements, will only be available after the last design cycle is completed; the earlier design cycles did not cover all user requirements and produced what may be called ‘prototypes’.

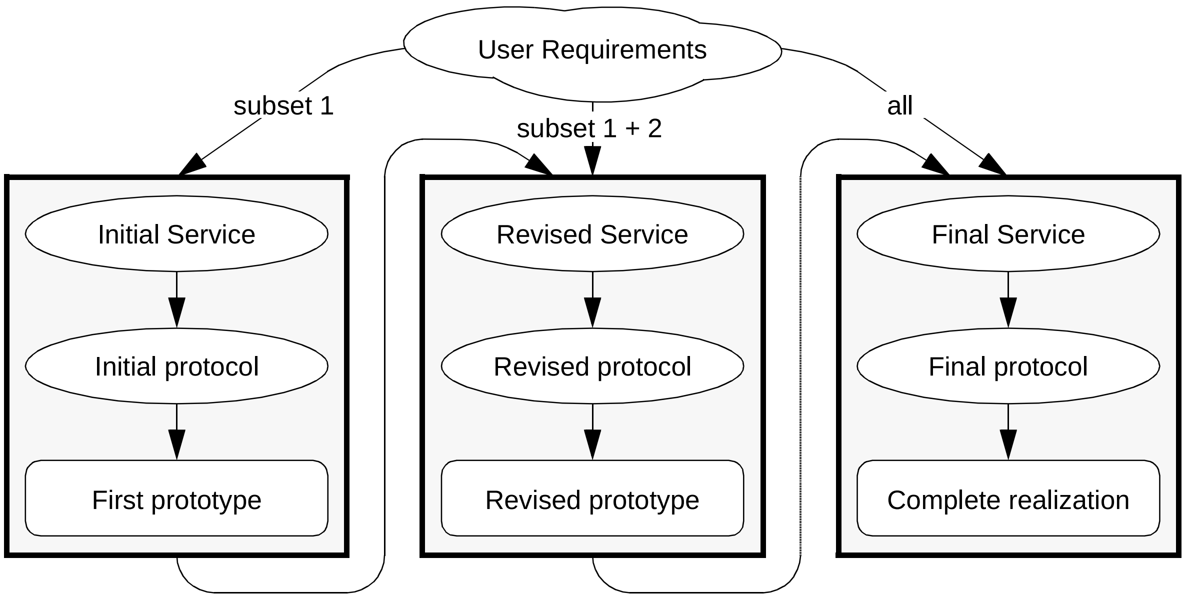


Figure 6.1: Cyclic design

A good strategy in cyclic design is to start with those requirements that are expected to have the strongest impact on the system’s structure. Such strategy ensures that most of what has been achieved in the first cycle(s), can be reused in later cycles. Usually this means that the designer should start with the development of primary, i.e. data transfer functions; development of management functions should be postponed until later cycles.

> *Example:* In the first cycle the designer elaborates how to exchange data between systems. To reduce the complexity of this cycle, the designer neglects for instance QoS requirements. After completion of the first prototype, the designer detects that the prototype is unable to deliver all data: data sometimes get lost. In the next cycle the designer therefore adds retransmission functions. Although the new prototype behaves better, some data still get lost. The designer therefore adds in subsequent cycles fault management functions, which should be used to detect and possibly correct persistent faults.

At first glance, cyclic design looks inefficient with respect to design time and manpower. Two remarks should therefore be made:

- It is not necessary to consider all three phases of the step-wise design process during each design cycle. This is particularly true for the first cycle(s), in which the designer will primarily be interested in getting rough insight in the

system's structure. Realization phases cost a lot of time, but do not contribute very much to this insight. It may therefore be a good idea to skip some of these realization phases.
- Cyclic design allows several design groups to proceed in parallel. The following scenario is for instance conceivable (Figure 6.2). An architectural group develops the initial service definition. After its completion, an implementation group uses this definition to develop the initial protocol definition. In the mean time the architectural group starts to develop a revised service. After completion of the initial protocol, a realization group uses this definition to develop the first prototype. In the mean time, the implementation group continues and develops a revised protocol. The experience obtained in the first implementation cycle will also be used by the architectural group for the development of a better service, etc. The ESPRIT/PANGLOSS project applied this approach effectively and obtained good results with it [6].

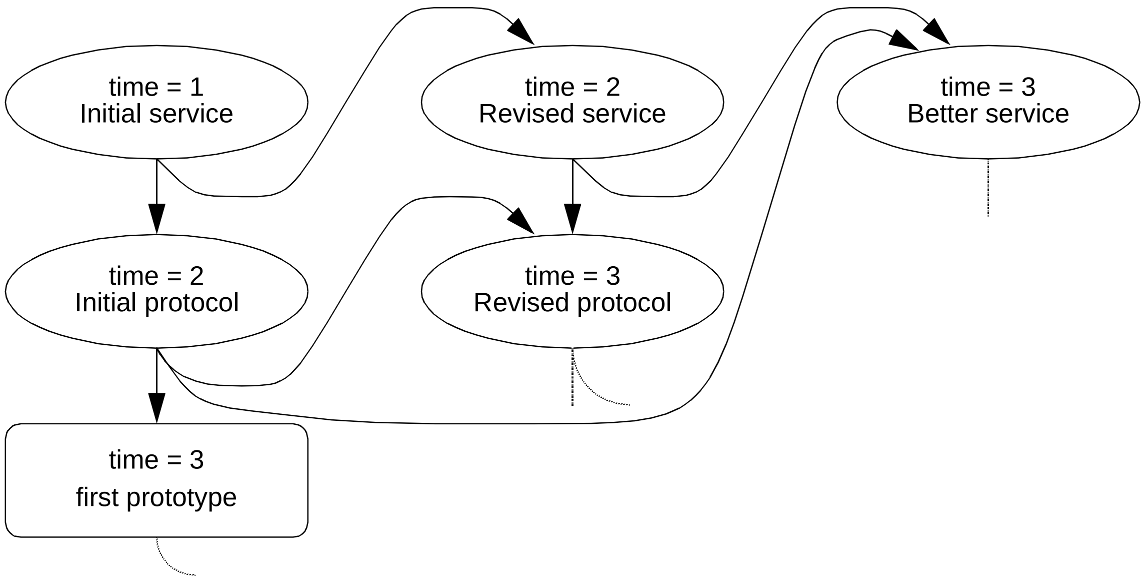


Figure 6.2: Parallel activities in cyclic design

6.2 Development of element management

As explained in Section 5.4, the need for element management functions appears during the last phase of the step-wise design process: the realization phase. In this section this phase will be discussed separate from the architectural and implementation phase. The starting point for this phase is the protocol specification. In this section it is assumed that the cyclic design model will be applied to transform this protocol specification into a realization.

Consider the example of a protocol specification that includes flow control and retransmission functions. Assume the protocol should be realized in software. During the first cycle, the designer's main problem is to find a good realization structure. For this purpose, the designer considers the most important protocol functions first, in this example the flow control and retransmission functions (Figure 6.3). Although buffers will be needed for both functions, the designer need not worry during this first cycle about the amount of memory needed for these buffers. Just to be sure, the designer may decide to allocate a large amount of memory.

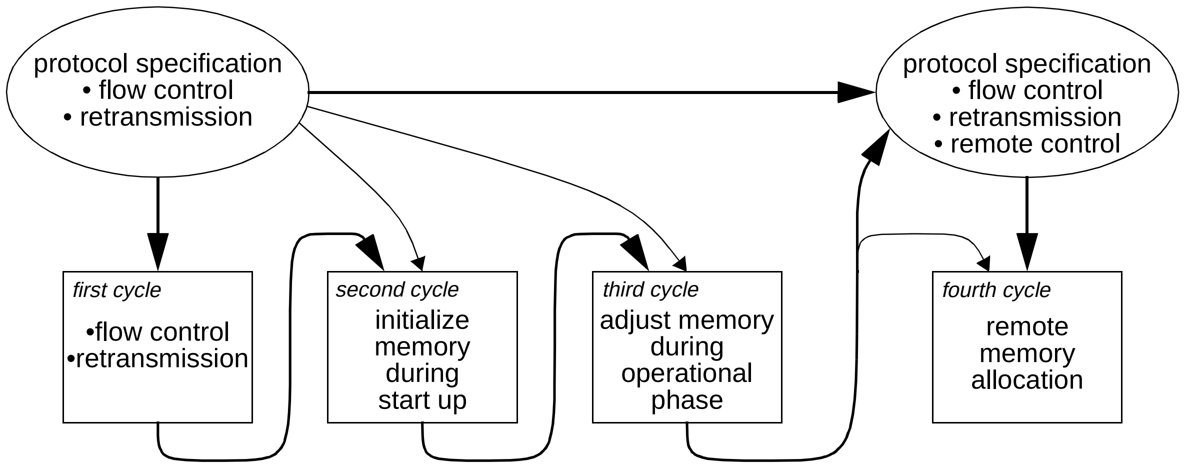


Figure 6.3: Development of element management

After completion of the first prototype, the designer may discover that these buffers account for 50% of all memory capacity. To reduce this percentage, the designer may decide to add a function that allows a manager to determine the required amount of memory during the start up. In case the system is expected to handle many messages (e.g. routers and file servers), the manager allocates large amounts of memory; in other cases (e.g. End Systems) the manager allocates small amounts of memory.
As discussed in Section 5.4, this function may be regarded as an element management function. Such function will be identified during the design of the primary functions and should be elaborated during a next cycle of the design.

A problem with the type of element management as described above, is that memory allocation is static: the amount of memory allocated during system's start-up can not be changed after the system has become operational. In a next cycle the designer may therefore decide to add the possibility to allocate during the operational phase memory dynamically (Figure 6.3). This type of management will be particularly useful in combination with a facility to monitor the actual memory usage. Such management function should therefore be added too. As a result, the element manager becomes able to fine-tune memory allocation based on operational requirements.

A final step may be to include a possibility to monitor memory usage and allocate memory from a *remote* place (Figure 6.3), which involves the definition of special element management interactions. Since these management interactions change the external behaviour of the various systems, the initial protocol is thus extended.

The example in this section showed that element management functions can be *identified* during the first design cycle(s) as part of the realization process of the primary functions, and can be *elaborated* during the subsequent design cycles (Figure 6.4). To allow the initiation of element management functions from a remote place, it may be necessary to extend the initial protocol specification. This implies that the introduction of element management functions can have repercussions for later *implementation* cycles. The need to introduce element management functions can usually not be deduced from the service definition: it will therefore not be possible to address element management functions during the architectural phase.

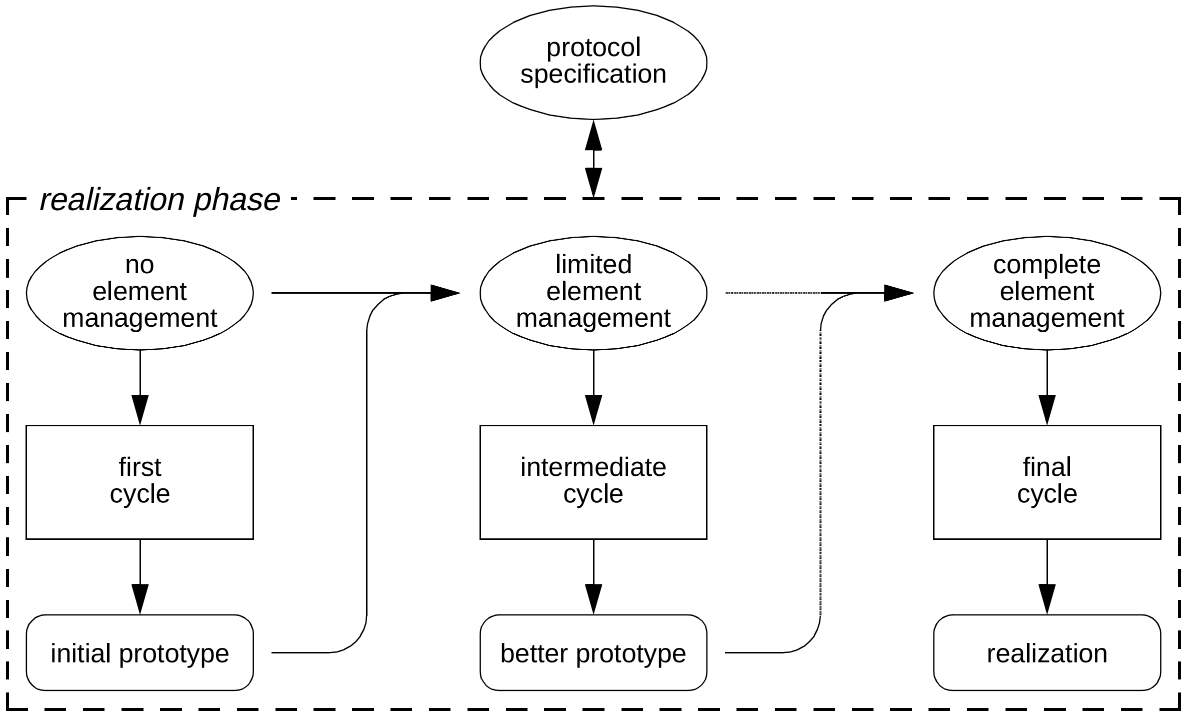


Figure 6.4: Element management and cyclic design

6.3 Development of protocol management

The question *when* to elaborate protocol management functions will be discussed now. The scope of this discussion will be restricted to those management functions that can be completely defined before the operational phase starts; protocol functions elaborated after the start of the operational phase will be discussed in Section 6.6.

Protocol management functions should be elaborated during the implementation phase, and not in the architectural or realization phase. This can be understood as follows:

- To define protocol management functions, knowledge must be available about the primary protocol functions for which the management functions are intended [111]. Such knowledge about protocols is not available in the architectural phase, which implies that protocol management functions can not be defined during the architectural phase.
- To perform protocol management functions, it will generally be necessary to exchange management information between protocol systems. To allow sys-

tems from different vendors to interoperate, all information exchanges must be defined during the implementation phase. This implies that also protocol management functions should be elaborated in the implementation phase, and not in the realization phase.

A sensible implementation approach is to determine the main structure of the provider first. In case the design is performed in a cyclic fashion, this implies that the primary protocol functions should be defined during the initial design cycle(s). While elaborating these functions, protocol management functions may be *identified* (see also Section 5.3). To reduce the complexity of the design, the *elaboration* of protocol management functions may be postponed until later design cycles.

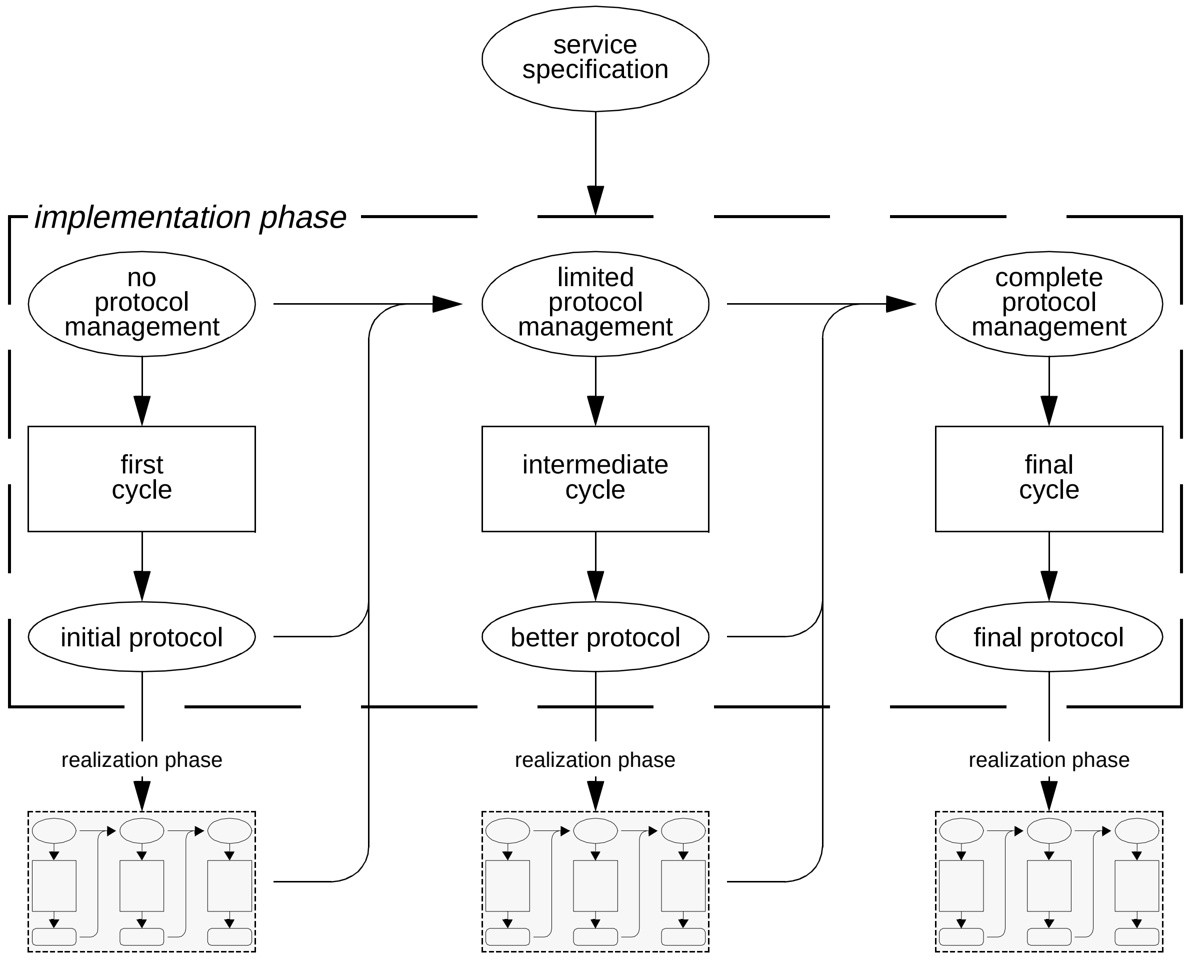


Figure 6.5: Protocol management and cyclic design

This process is shown in Figure 6.5. In this figure the first cycle is used for the development of an initial protocol definition, which includes functions that can directly be derived from the service definition (such as those to exchange user data). Although the need for protocol management functions is already apparent during this first cycle, such functions will not be elaborated during this first cycle. Instead, protocol management functions are gradually added within the subsequent design cycles. The result of each of these cycles is a 'richer' protocol[1].

Example

The development of the OSI connectionless network layer standards provides a nice example of the cyclic addition of management functions [75][77][1]. During the first cycle (Figure 6.6), the ConnectionLess Network Protocol (CLNP) was defined. This CLNP standard [46] defines PDU structures and identifies the functions needed to exchange user data. Examples of such functions are: routing[2], addressing, segmentation and congestion control. The CLNP standard does not define the details of these functions; elaboration of such details is left to subsequent cycles [75]. After completion of the standard and after different manufacturers realized several prototypes, a second cycle was started. During this cycle the problem of how to exchange routing information between End System (ES) and Intermediate System (IS) was resolved. The result of this cycle was the ES-IS routing standard (ISO 9542 - [48]). A third cycle was started to resolve the problem of how to exchange routing information between Intermediate Systems within a single routing domain. This resulted into the definition of the IS-IS intra domain routing standard (ISO 10589 - [55]). The problem how to exchange routing information between Intermediate Systems within different routing domains was addressed in a fourth cycle; this resulted in the definition of the IS-IS inter domain routing standard (ISO 10747 - [57]). In later cycles ES address assignment and group address (multicast) extensions were tackled.

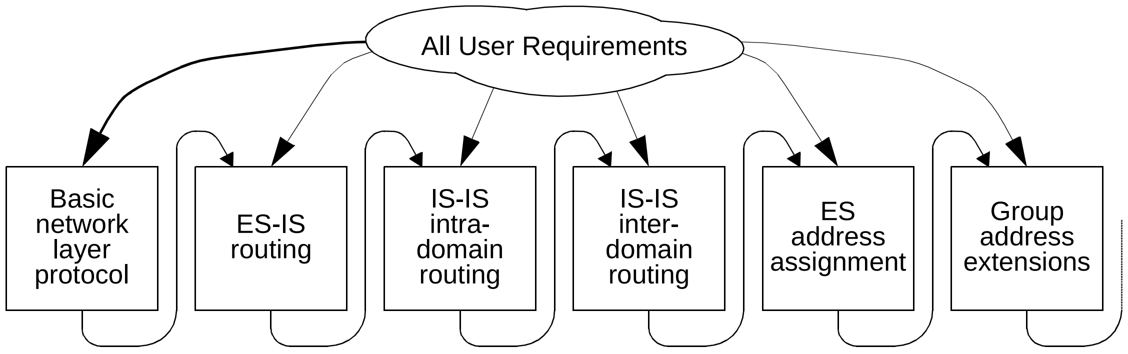

All User Requirements
Basic network layer protocol
ES-IS routing
IS-IS intra-domain routing
IS-IS inter-domain routing
ES address assignment
Group address extensions

Figure 6.6: Cyclic development of the OSI network layer protocol

Feedback from previous realization exercises

Figure 6.5 shows that experiences obtained in previous realization exercises may have consequences for successive implementation cycles. This leads to two possible consequences.

1. In [75] this enrichment process is called ‘horizontal refinement’. Note that many do not consider the *original* protocol to be *refined*, but *new* (management) protocols to be *defined*. This view emerges if the term ‘protocol’ has become a byword for ‘standard’.
1. See also Figure 1.7 on page 9.
2. Do not confuse routing and forwarding. Routing is responsible for the initialization and maintenance of routing tables. Forwarding is responsible for the selection of outgoing links and the actual transmission of packets over such links. A forwarding function uses, but does not modify the routing table [49].

In the first place it is possible that previous realization exercises show that certain parts of the protocol are difficult to realize. The implementor can use this knowledge and redefine the protocol parts in question. Of course the new protocol should still be a valid implementation of the original service definition.

In the second place it is possible that previous realization exercises show a need for remote element management. As discussed in the previous section on page 98, this may result in the definition of *additional* protocol interactions to support the initialization, observation and modification of particular realization constructs.

6.4 Development of service management

During the start of the design the emphasis will in many cases be on user data aspects and not on management. This implies that the first questions the designer will be faced with, are questions like:

- Should the provided service be connection-oriented or connectionless?
- Should the network support a fixed or variable data size?
- Should the network support a fixed or variable delay?

It is likely that the answers to these questions have a major impact upon the network's internal structure.

> *Example:* To a large extent, the choice between a circuit switched network with out-of-band signalling and a packet switched network with inband signalling will be determined by the answers to the above questions.

In case the design is performed in a cyclic way, it may be a good approach to elaborate user data issues during the first design cycle(s) and service management issues during later cycles (Figure 6.7).

The first cycle(s) will be used by the designer to transform the requirements concerning the exchange of user data into a service definition. This definition specifies the service primitives that can be used to exchange user data, the relationship between these service primitives and the parameters that belong to these primitives. An example of such service definition is the standard for the OSI network service. This definition does not yet include service management issues, such as how to assign addresses or closed user groups (see also Section 5.2). These issues are not expected to determine the network's main structure and can therefore be elaborated during subsequent cycles.

Figure 6.7 shows that new design cycles do not only consider service management issues that can be derived from the user requirements, but also issues that came up during previous realization and implementation exercises. From this form of feedback the designer for instance learns which system failures are most likely to occur. This knowledge can be used in new design cycles to

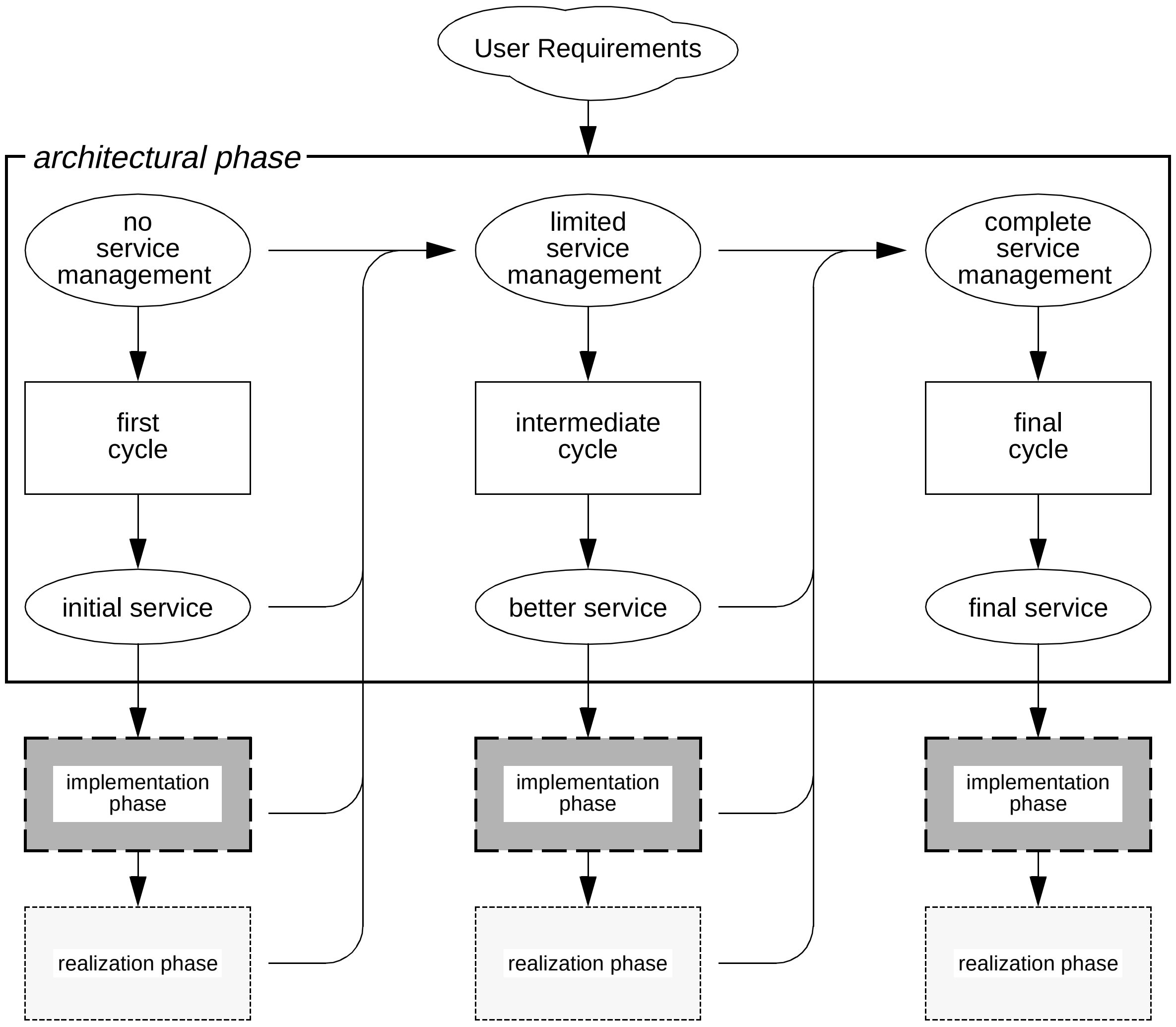


Figure 6.7: Service management and cyclic design

decide which fault information should be presented to the service users and which fault management functions should be added.

The sequence in which service management issues will be addressed, is decided by the designer. The designer usually starts with those requirements that are expected to have the largest impact on the network's structure. It may be expected that service management issues that directly follow from the user requirements (such as the assignment of addresses and closed user groups) will be tackled before service management issues that can only be identified after some implementation or realization experience has been obtained (such as fault management).

In real life designs, service management functions will not always be addressed during the architectural phase. Instead, elaboration is sometimes postponed until the end of the realization phase. This may be a bad policy, as can demonstrated by the following example.

Example: In real life designs, the problem of SAP address assignment is sometimes tackled as tail piece of the design. In such cases, ad-

dress assignment functions are added by the various manufacturers just before the realization phase completes. As a result, address assignment will be system dependent and management interactions between systems will not be defined. Without such interactions, the network can not check for duplicated addresses. Since uniqueness of SAP addresses must still be guaranteed, the burden is put upon the (human) manager, who must take special measures during the operational phase.
In case the problem of address assignment would have been resolved during the architectural phase, a special service primitive could have been defined to set or modify SAP addresses. This service primitive could be defined in such way, that successful execution would only be possible in case the proposed address was unique. To support this primitive, the implementor should define protocol functions that check uniqueness of the proposed address. Since these functions are added during the design phase, special measures will not be needed during the operational phase.

6.5 Meta-management

Section 6.2 through Section 6.4 discussed the development of management functions. In a number of cases the complexity of these management functions will be such, that these functions themselves should also be managed. To denote this 'management of management', we introduce the term 'meta-management. Which meta-management functions should be added can only be determined during the elaboration of the 'normal' management functions. Meta-management functions must therefore be elaborated in later design cycles as normal management functions (Figure 6.8).

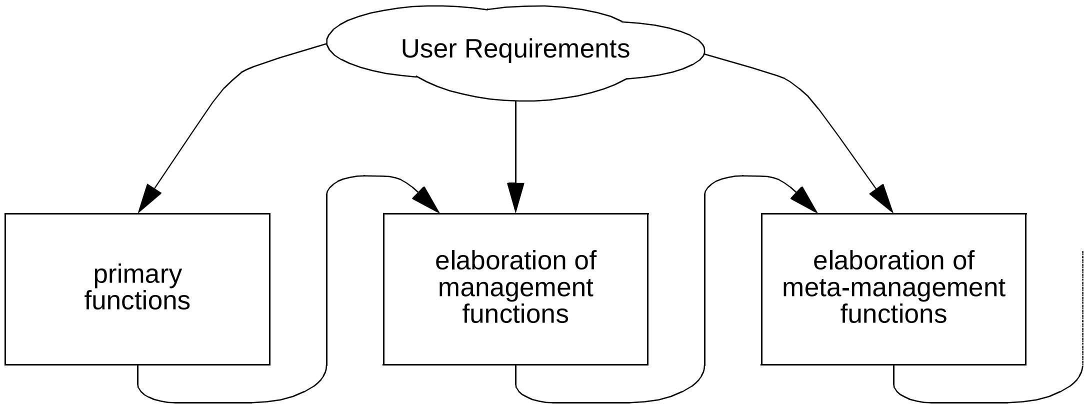

User Requirements
primary functions
elaboration of management functions
elaboration of meta-management functions

Figure 6.8: The addition of meta-management to the design

The idea of meta-management can be illustrated by means of an example. The example that is taken, is the one of TCP [84] and SNMPv2 (see Section 4.2).

The ellipse at the bottom of Figure 6.9 shows the primary functions, as defined by the TCP protocol. Management of these functions can be accomplished by

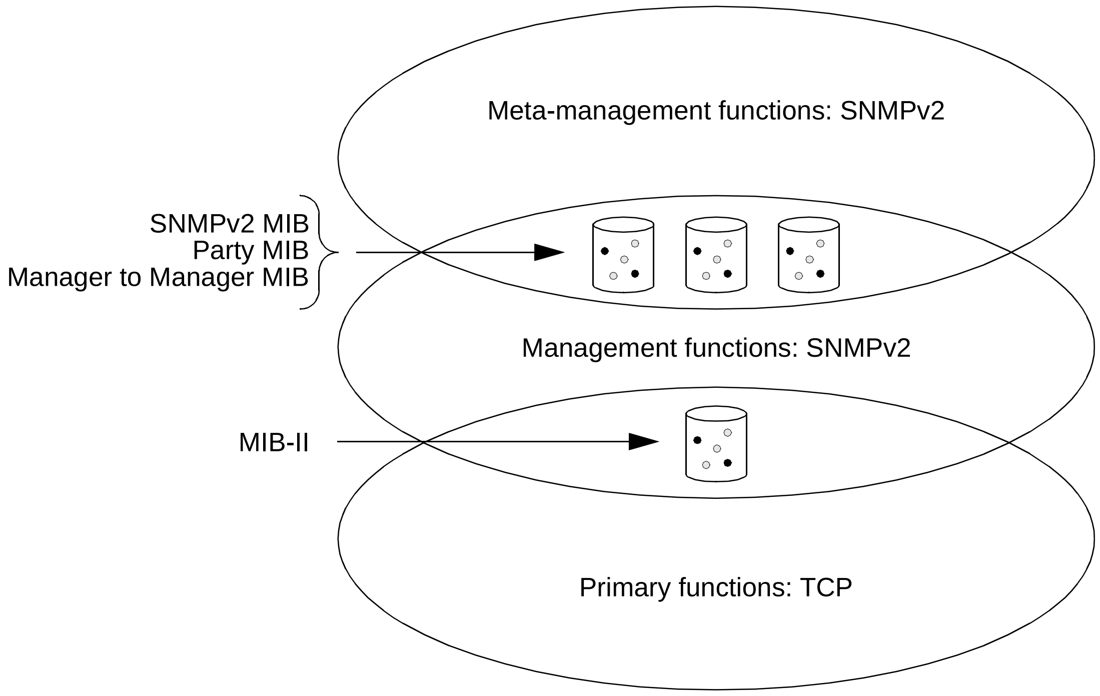


Figure 6.9: Primary functions, management and meta-management

reading and modifying certain TCP variables. The variables that are available to management are defined as part of a separate standard: the MIB-II [94]. Managers may read and write MIB-II variables across a network. The rules describing how to transfer MIB-II values and the policy that may be used to assign to different managers different MIB-II access rights, are defined by the SNMPv2 standards (the ellipse in the middle of Figure 6.9).

The operation of the SNMPv2 functions depend upon various SNMPv2 variables. Examples of such variables are:

- authentication keys
- encryption keys
- access control tables.

The values of these variables are determined and maintained by meta-management functions (ellipse at the top of Figure 6.9). For this purpose, these variables are organized as part of three distinct MIBs: SNMPv2 MIB [99], Party MIB [98] and Manager to Manager MIB [100]. The values in these MIBs are *used* by the normal management functions; they are *controlled* by the meta-management functions. Some of these meta-management functions are also defined by the SNMPv2 standards.

6.6 Cyclic design versus life cycle

The previous sections discussed the cyclic addition of management functions and the growth of functionality with each successive cycle. After completion of several cycles, the realization at our disposal already meets most of the user requirements. Still additional cycles will be needed to include the remaining functionality. As compared to the first cycles, completion of these additional cycles may be time consuming for the following reasons:

- The designer may not yet have sufficient experience to solve all remaining issues (see also page 3). This is not surprising if we realize that traditionally most research is invested in basic functions (such as data transfer), and not (yet) in derived functions (such as management).
- The number of details that must be addressed in the later cycles is, as compared to the earlier cycles, usually much higher.

Example: It is reasonable to assume that the size of a specification has some relationship to the number of details. If we take the example of the OSI network layer (page 101), we see that the specification that was made during the first cycle had a size of 51 pages. During the second cycle 31 pages were added. During the third cycle 122, and during the fourth cycle 107 pages were added. The number of details that must be addressed in later cycles is thus much higher than the number of details that must be addressed in the first cycles (Figure 6.10).

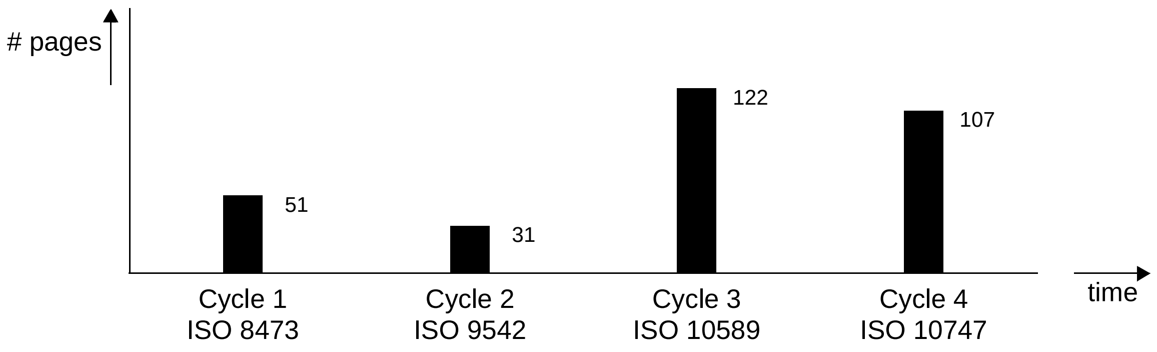


Figure 6.10: Increasing size of OSI network layer standards

Addressing all tedious little details takes a lot of time. From a financial and competitive point of view it is therefore understandable that a wish exists to use the ‘incomplete’ realizations that have been developed thus far. The designer should anticipate this and allow quick and dirty solutions for these tedious little details; the perfect solutions should be postponed until later cycles.

A possible strategy is to let human operators perform the remaining management functions after the start of the operational phase (explicit management). In the specific case of the network layer (Figure 6.6) this means for example that some of the routing functions will be performed by persons. Since human beings can not be at multiple places at the same time, management should be performed in a centralized way.

6.6.1 From explicit to implicit management

The idea of human operators performing management functions was already discussed in Subsection 1.3.1. In that subsection it was shown that explicit management has several deficiencies, such as being expensive, causing a relatively high number of errors and showing poor response times. Explicit management should therefore not be seen as the ultimate solution. After the network becomes operational the designer should therefore continue his job and improve the design!

An obvious step is to gradually automate the functions performed by the human manager. This implies that there will be a shift from *explicit* management to *implicit* management. We may say that the manager functions that were previously realized in 'brainware', will step by step be realized in hard- and software.

This process can be explained in terms of the cyclic design model. Figure 6.11 shows, for instance, the additional design cycle that is performed during the operational phase to improve the manager system. Although the figure shows only a single cycle, it can be expected that in reality multiple cycles will be needed. Note that the additional cycle does not comprise a new architectural and implementation phase; only the *realization* of the manager system will be improved. This is because the changes should be restricted to the *manager system*; the *managed systems* should not be affected. The protocol should therefore remain the same; there is no need for an additional architectural and implementation phase.

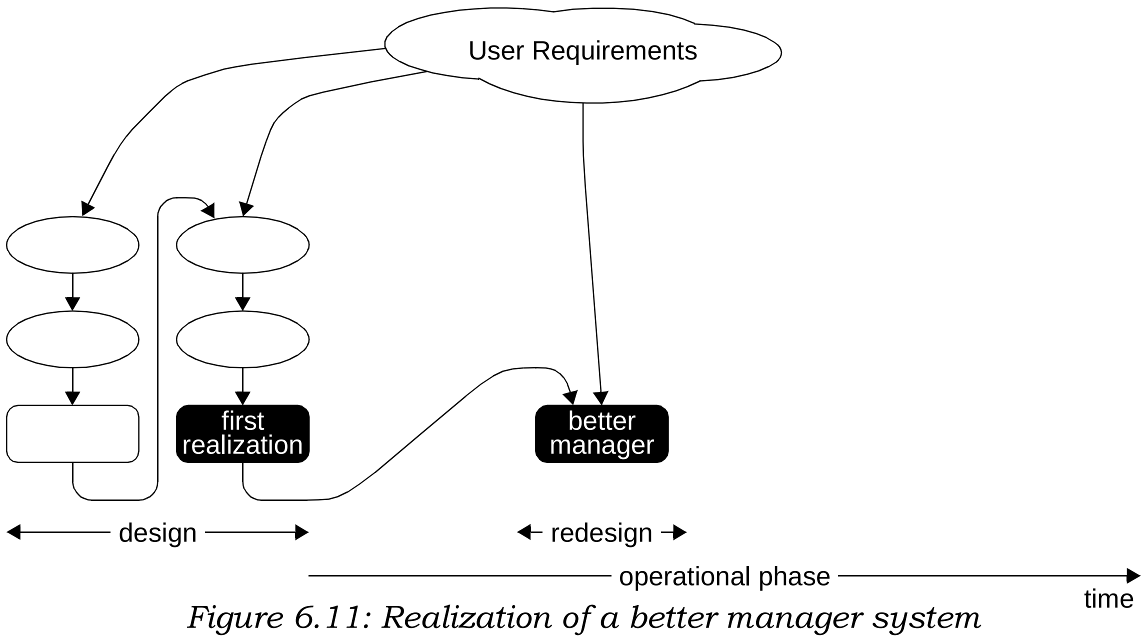


Figure 6.11: Realization of a better manager system

It is important that already during the design phase there is some awareness that this shift from explicit to implicit management is likely to happen. To enable future modifications the designer may for instance decide to develop the manager system in the form of a *management platform* that can easily be extended with new management applications.

Management platform

Many of the manager systems that are on the market today are actually management platforms (Figure 6.12). To a certain extent these platforms can be compared to MS-DOS PCs: they can both perform elementary functions without additional software but they both become more powerful after special application software has been added. Such application software may be developed by independent vendors, who have particular expertise in certain problem areas.

Manufacturer	Management platform
HP	Openview
IBM	Netview 6000
SUN	SunNet Manager

Figure 6.12: Some existing management platforms

A typical management platform consist of the following components:
- A user interface (based on X11 or MS-Windows) to enter management commands and display management information.
- Communication software (such as SNMP or CMIP).
- A database system to log management information.
- Programming interfaces to add management software at a later stage. The programming interfaces should be 'open', which means that software may be written by different vendors.
- A 'script' facility to combine multiple management commands.

It may be expected that future management platforms will also include some kind of 'expert system'. Such systems allow a gradual shift from explicit to implicit management; several papers have already been written on this subject [11][39][40][68].

6.6.2 From centralized to distributed management

Centralized management is initially needed to ensure that human operators will be able to perform their job from a single location (or just a small number of locations). However, as management functions are automated step by step, the need to perform management in a centralized way slowly disappears. In Subsection 1.3.2 it was explained that centralized management has several deficiencies (e.g. single points of failure, potential performance bottleneck). To solve these deficiencies, it is likely that at certain moments in time (after the shift from explicit to implicit management) the decision is made to distribute some of the management functions. There is thus also a shift from centralized to distributed management too (Figure 6.13).

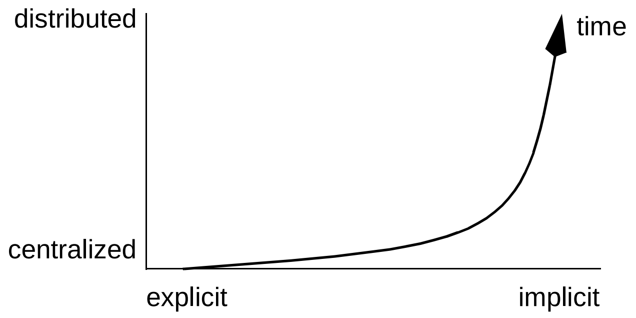


Figure 6.13: From explicit & centralized to implicit & distributed management

A good example to demonstrate the shift from centralized to distributed management is the development of routing protocols for datacommunication networks. Initially the contents of forwarding tables was determined at a central place. Presently this function is implemented in a distributed way and the contents of the forwarding tables can be determined by the various routers within the network. A number of routing protocols have been standardized to define the management interactions between these routers (see also page 9).

The cyclic design model can also be used to illustrate this development (Figure 6.14). At the right side of the figure the design cycle is shown in which (parts of the) management functionality is distributed over the various systems. Such distribution requires a modification of existing protocols, which means that (as opposed to the redesign of the manager) an additional implementation and presumably also architectural phase will be needed.

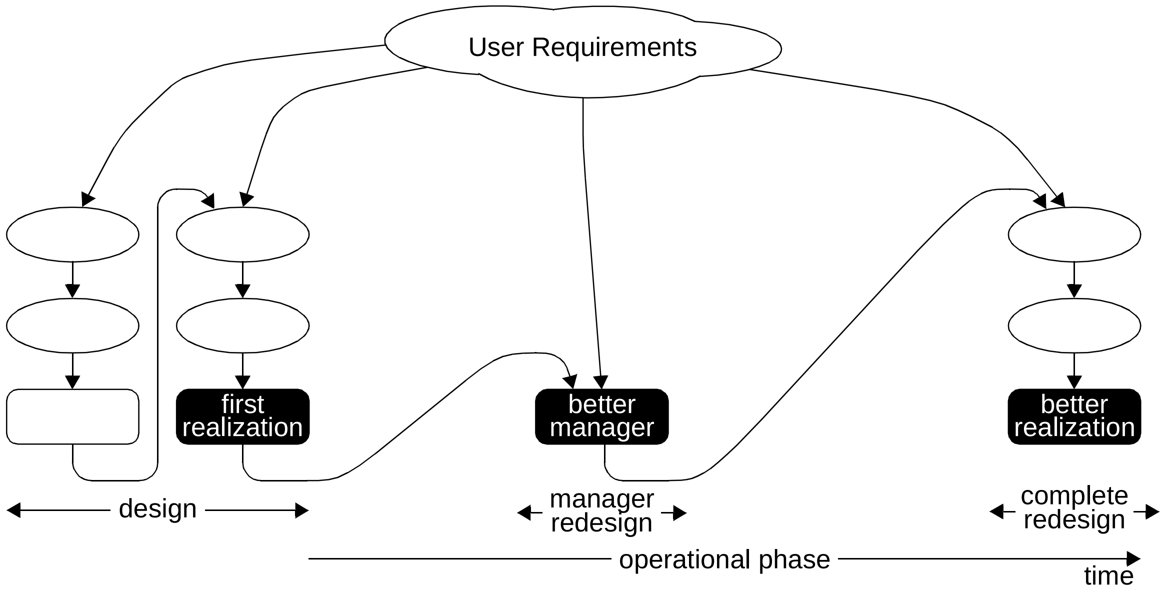


Figure 6.14: Additional design cycle to distribute management functions

This kind of redesign affects many network systems. Since not all systems may have been written down, the frequency at which these cycles are initiated may not be to high.

6.7 Conclusions

The cyclic design model can be used to illustrate the design of (service, protocol and element) management functions. According to this model, primary functions will be developed in the first design cycles. As part of the definition of primary functions, management functions will be *identified*. To keep the complexity of each design cycle within reasonable bounds, the *elaboration* of management functions will be postponed until later cycles.

The complexity of management functions may be such, that also the management functions should be managed (Section 6.5). To denote this 'management of management', the term *meta-management* is introduced.

The designer may not always be in the position to get complete grip of all management functions before the start of the operational phase (Section 6.6). In such case the designer should take a number of decisions to allow a human operator to perform the remaining management functions during the operational phase. One of these decisions is to select the explicit and centralized management approach. After the designer has obtained additional experience, this decision may be adjusted and step by step the approach may be changed into an implicit and distributed way.

The cyclic design model can also illustrate the various redesigns that may be needed because of this change in the operational phase. Whereas each cycle in the design phase results into a new *prototype*, each cycle in the operational phase results into a new system *generation*.

7: An alternative management architecture

7 An alternative management architecture

The management architectures that were presented in part I of this thesis can be characterised by the fact that they only consider management functions, and not the primary functions that should be managed. In this chapter an alternative architecture will be presented that shows how *primary functions should be extended with management functions*. This extension should be performed during the additional design cycles and results in enhanced service and protocol specifications. These enhanced specifications *integrate* primary and management functions.

The view that management functions should be seen as extensions to the primary functions, implies that it should be possible to define both kind of functions in terms of a *single* set of architectural *concepts* and *rules*. In this thesis it is decided to use the concepts and rules of the OSI Reference Model; this decision is made because OSI concepts and rules are, as compared to others, relatively well accepted and documented.

An important contribution of this chapter are the *models* that show how to apply these concepts and rules to explain the case that management is performed from one or more remote locations.
The following models will be presented:
- Section 7.1 develops models for service management.
- Section 7.2 develops models for protocol management.
- Section 7.3 develops a model for element management.

A number of concluding remarks will be provided in Section 7.4.

7.1 A service management architecture

In this section several models will be presented that show how existing services can be extended to include service management. To demonstrate that service management can be expressed in terms of the same architectural concepts and rules as normal services, a number of examples will be given.

Figure 7.1 shows the part of the service design process in which existing services are extended to include service management. During the early design cycles the focus will be on the development of primary service functions.

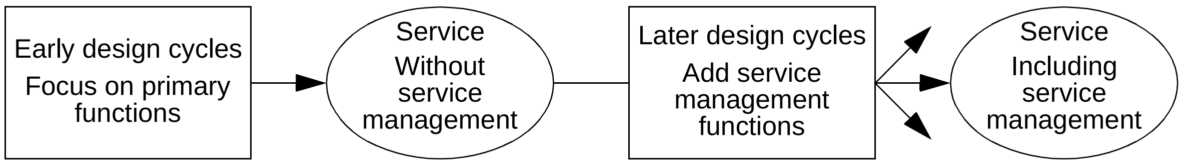

Early design cycles
Focus on primary functions
Service
Without service management
Later design cycles
Add service management functions
Service
Including service management

Figure 7.1: Extending a service to include service management

Figure 7.2 shows the service model that guides the design of these primary functions. It shows a service provider, a number of service users (the four clouds) and SAPs. These SAPs represent the parts that users and provider have in common; the P at each SAP indicates that only the execution of *P*rimary service functions is supported.

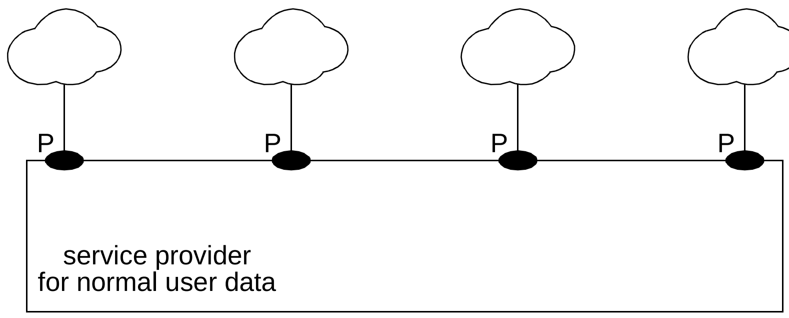


Figure 7.2: Starting point: no service management yet

The addition of service management has consequences for the interactions between users and provider. *This implies that the behaviour of the provider, as well as the behaviour of users will be adapted.* The effect of service management addition can thus not be confined to the service provider alone!

How to distribute service management functionality

The model of Figure 7.2 is the starting point for the later design cycles in which the service management functions will be added. In these later cycles design decisions must be made regarding the distribution of this service management functionality[1]. The following decisions can be made:

- Distribute all service management functionality in an equal way over all users. Subsection 7.1.1 discusses this approach.
- Centralize all service management functionality. In this case all service management interactions should be executed from a single SAP. The designer may even decide to introduce a dedicated service management SAP. Centralized service management functionality is discussed in Subsection 7.1.2.
- The third possibility is the hybrid approach in which the two previous approaches are mixed. The hybrid approach allows execution of a subset of the service management interactions from all SAPs; the remaining service management interactions have a special status and are restricted to a dedicated SAP. Subsection 7.1.3 discusses this case.

7.1.1 A model for distributed service management

The idea behind equal distribution of service management, is to give all service users the same set of service management facilities. In literature this idea is called ‘Customer Network Management’ [1][117]. A potential problem associated with this approach, is that interference between managing users can not always be avoided. To reduce such problem, the designer should put restrictions upon the set of service management interactions.

1. The advantages and disadvantages of distributed management are discussed in Section 1.3 and will not be repeated in this chapter.

Example: Users should for instance be allowed to change their own address, but they should not be allowed to change the address of someone else. In case a user wants to change his own address, a check should be made to ensure uniqueness of the new address.

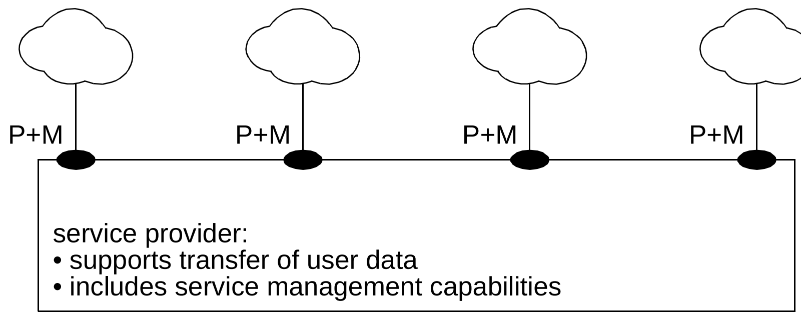

P+M
P+M
P+M
P+M
service provider:
• supports transfer of user data
• includes service management capabilities

Figure 7.3: Model for distributed service management

Figure 7.3 shows the model that emerges after service management functionality has been added in an equal way. As an implication, all SAPs support the same set of service management interactions. In fact, the model is similar to one that was used during the early design cycles (Figure 7.2); addition and equal distribution of service management does not modify the structure of the initial model.

The result of service management addition is a change of user and provider behaviour. This change must be reflected in the service definition, which will be discussed in the next subsection.

7.1.1.1 Impact on the service definition

The effect of adding service management can be reflected in a service definition as follows:

- Existing service primitives may be extended with special service management *parameters*.
- Special service management *primitives* may be defined.
- The *relationship* between service primitives may be modified.

Parameters

To demonstrate the addition of service management may involve the attachment of new parameters to existing service primitives, the example of QoS monitoring (page 79) will be used.

Consider the case of a connectionless service design. During the early cycles of the design the designer may assume that it will be possible to implement a service that guarantees the QoS as negotiated between user and provider as part of the *DataRequest*. Since this negotiated value is known at user level, there is no need to include a QoS parameter in the *DataIndication* (Figure 7.4).

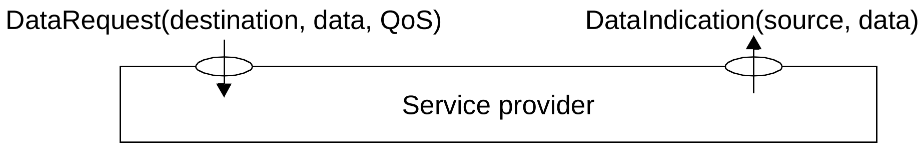

DataRequest(destination, data, QoS)
DataIndication(source, data)
Service provider

Figure 7.4: No service management

After the first prototypes have been developed, the designer may discover the initial assumption, which says the provider always delivers the negotiated QoS, does not hold. The designer may now decide to add during a subsequent design cycle a QoS parameter to the *DataIndication* primitive (Figure 7.5). This parameter expresses the delivered QoS. This parameter may be important to service management: its value can for instance be forwarded to the initiating user and inspected for accounting purposes.

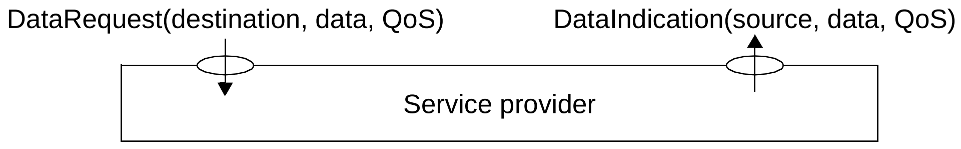

DataRequest(destination, data, QoS)
DataIndication(source, data, QoS)
Service provider

Figure 7.5: A parameter is added for service management purposes

Primitives

The introduction of service management may also involve the definition of new service primitives. An example of a possible service management primitive is the *error-report*. This primitive may be added in the later design cycles to inform the user about permanent and transient provider errors. The *error-report* primitive may carry parameters to indicate the error cause. Such parameters may be used by the service user to decide if and when new service attempts should be made (Figure 7.6).

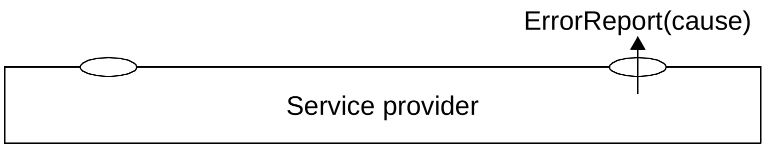


Figure 7.6: ErrorReport primitive

Another service management example is the reachability test. This example can[1] be modelled in terms of two service primitives: *TestReachabilityRequest* and *TestReachabilityIndication* (Figure 7.7). The request primitive includes the address of a remote SAP (e.g. B); the indication primitive includes the requested reachability information. Both primitives occur at the same SAP (e.g. A).

1. Some designers prefer to model the reachability test at a higher abstraction level in terms of a single *TestReachabilityIndication* primitive. For the service implementor however it may be easier to have two primitives that map directly upon underlying PDUs.

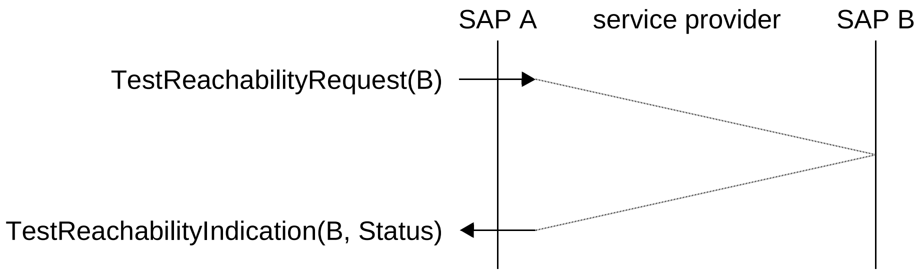

SAP A
service provider
SAP B
TestReachabilityRequest(B)
TestReachabilityIndication(B, Status)

Figure 7.7: Reachability test

Relationship between service primitives

Another possible effect of adding service management, is a modified relationship between service primitives. This modification should have the nature of an adjustment, and not a complete change of the relationship as defined during the early design cycles.

An example that demonstrates such modification is multicasting to a group. Suppose the designer has extended the original service definition with a management interaction to remove members from a group. The execution of such interaction has consequences for subsequent multicasts. There must therefore be a relationship between multicasts and the service management interaction.
This relationship is shown in Figure 7.8. Assume the group has initially three members: X, Y and Z. As shown at the top of the figure, a multicast from A will be received by all three members of the group. The management request that removes Y from the group is shown in the middle of the figure. After execution of this request, subsequent multicasts will no longer be received by Y. The conclusion can therefore be drawn that the relationship between primitives has been changed.

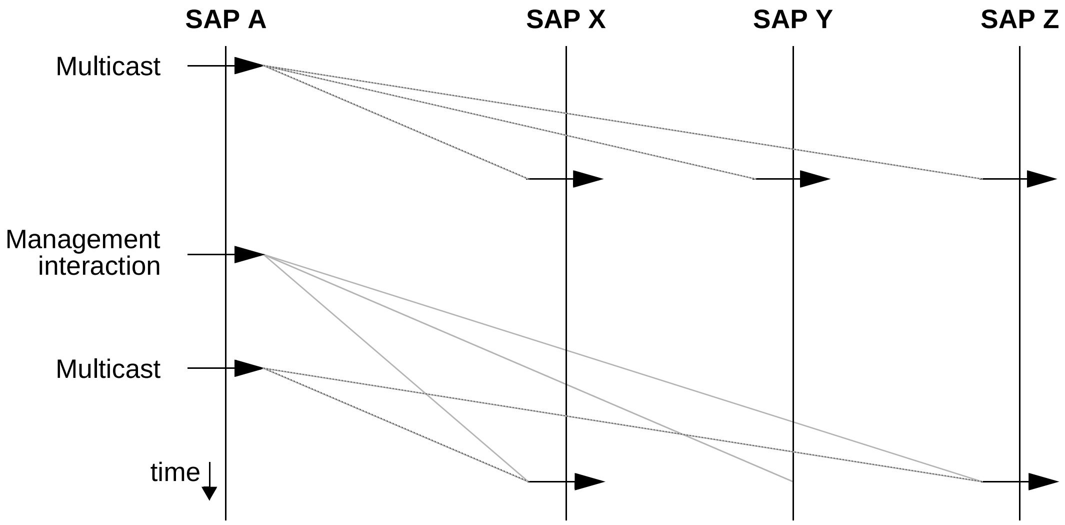


Figure 7.8: Refinement of primitive relationship

7.1.1.2 Difference with primary service definition

Although there are no principle differences between the design of primary and the design of management functions, it is possible to find some pragmatic differences between service primitives as defined in the early design cycle(s) and primitives as defined in the later cycles.

In the early design cycles, primitives and parameters are intended to carry user data from one user to another[1]. This implies that these primitives:

- Should carry address and data parameters.
- Should be defined in pairs: the occurrence of a request / response at one SAP should lead to an indication / confirm at another SAP.

The service management primitives and parameters which are defined in the later design cycles are not intended for user data transport. This implies that these primitives:

- Need not include address and user data parameters.
- Need not be defined such that the occurrence of a primitive at one SAP leads to execution of a similar type of primitive at another SAP.

Graphical representation

The effects of adding service management are graphically represented in Figure 7.9. The figure shows a single user plus parts of the service provider. The service user performs two kind of functions: primary functions for the exchange of user data and service management functions. The area that is shared between the user functions and the provider, is the SAP. To illustrate the possibilities of service extension that were discussed on the previous pages, the SAP has been divided into several parts:

- The primitives that belong to part I do not contain any service management functionality. These primitives have been defined during the early design cycles and required no modification after the addition of service management.

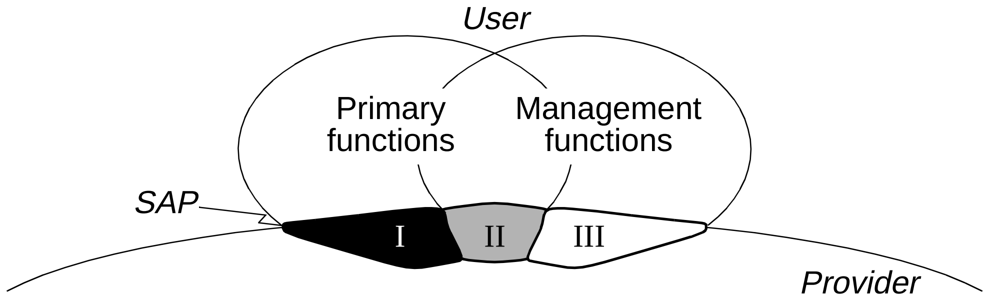


Figure 7.9: SAP can be divided into three parts

1. This statement is correct in case of communication services, but too simplistic in case of application services. Note however that the emphasis of this thesis is on management of communication networks.

- The primitives that were extended with service management *parameters* (see page 115), belong to part II. The explanation for this is that these primitives perform primary as well as service management functions.
- The primitives of which the *relationship* to other primitives were refined (see page 117), belong to part II. The explanation for this is that these primitives perform primary functions, but the relationship between these primitives has been modified because of service management reasons.
- The new *primitives* that were added to the design for service management purposes (see page 116), belong to part III. The explanation for this is that these primitives do not perform any primary functions.

7.1.2 A model for centralized service management

The choice between distributed and centralized service management is made by the designer and follows from the user requirements. The centralized approach is followed in case a requirement exists that service management interactions should be confined to a single location. The purpose of such a requirement may be to prevent unauthorized service management access or to avoid the interference that may occur in case of multiple managing users. Expressed in service concepts, centralized service management implies that all management interactions should be executed at one single SAP. This may be an existing SAP, or a new service management SAP. Both possibilities will be discussed in Subsection 7.1.2.1.

The model for centralized service management is simple and is presented in Figure 7.10. The model is similar to the one of Figure 7.2, except that the SAP at the right supports service management interactions. This support is indicated with the letter M; the brackets surrounding the P indicate that primary interactions may, but need not be supported at this SAP.

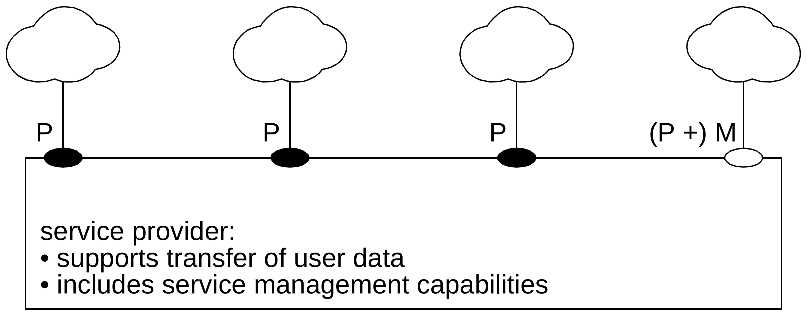

P
P
P
(P +) M
service provider:
• supports transfer of user data
• includes service management capabilities

Figure 7.10: Model for centralized service management

7.1.2.1 Impact on the service definition

If a service is extended to include service management, the service definition must be modified. In Subsection 7.1.1.1 the possible modifications for the case of distributed service management were discussed. Since all of these modifications may also occur in the case of centralized service management, they are

not repeated in this subsection. Instead, an additional modification, which is specific for centralized service management, will be discussed: the service may be extended with a special management SAP.
This subsection gives also a further example to show how the sequence of primitives can be refined after the service has been extended with service management.

Special service management SAP

After the designer has decided to confine service management interactions to a single location, the designer is confronted with the question whether the responsibility for service management should be allocated to an *existing* user or not.
In the case of public networks, the solution will often be to make a special user responsible for service management. This user usually belongs to the organization that owns the service provider. In the remainder of this text the term ‘service manager’ will be used to denote this special kind of user. To connect the service manager, the designer *adds* a dedicated service management SAP to the provider (Figure 7.11). This addition must be reflected in the new service definition.

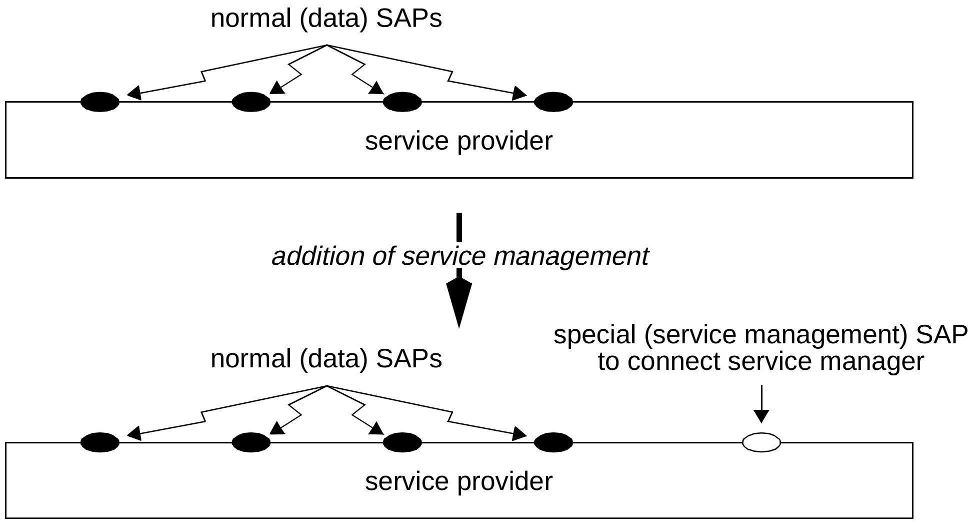

normal (data) SAPs
service provider
addition of service management
normal (data) SAPs
special (service management) SAP
to connect service manager
service provider

Figure 7.11: Introduction of special service management SAP

The designer may also decide that one of the *existing* users becomes responsible for service management. This solution is commonly found in private networks. In analogy with the terminology that is used in the world of operating systems, this user can be seen as a ‘super-user’: super-users have all the rights of normal users, plus the additional right to initiate service management interactions.

In the service definition the idea of a super-user can be reflected in two different ways. First, the designer may decide to *modify* the behaviour that is possible at the existing SAP between provider and super-user. In addition to the primary interactions that were defined during the early design cycles, this SAP also supports the execution of special service management interactions.

Second, the designer may decide to introduce a dedicated service management SAP. The super-user will now be connected to the service provider via two SAPs: one for normal data interactions and one for management (Figure 7.12).

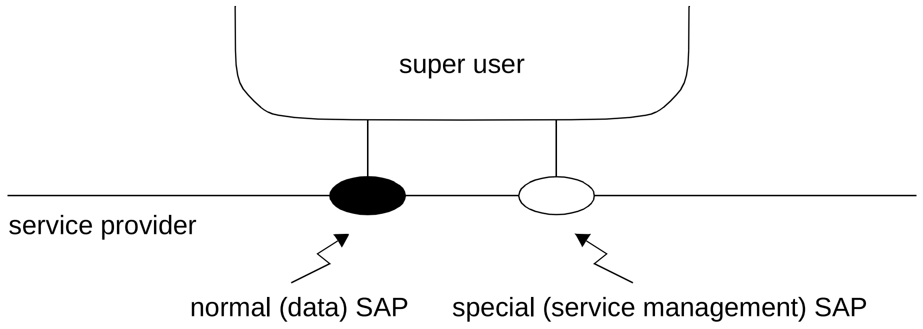


Figure 7.12: 'Super-user' connected via different SAPs

Relationship between service interactions

The service management primitives must be related to existing data primitives. Such new relationship must always be defined; without such relationship service management would not have any impact on the primary functions.

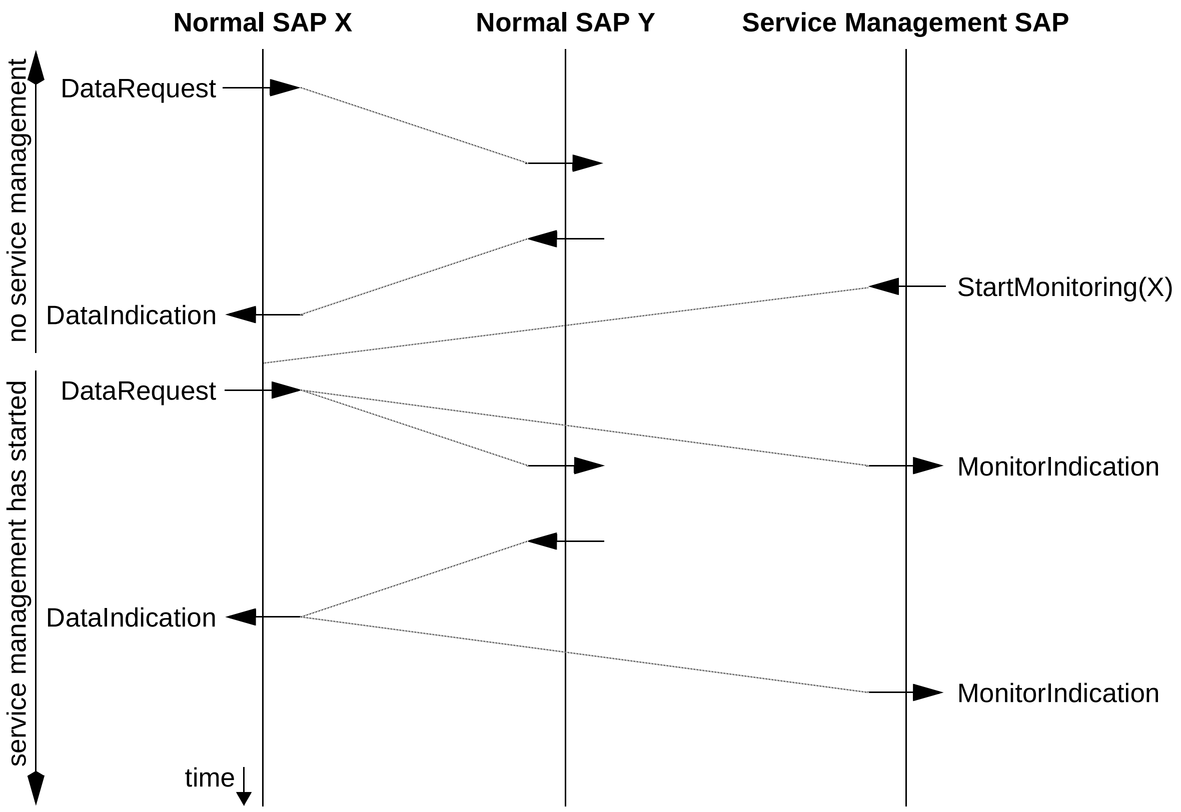


Figure 7.13: Possible effect of a service management SAP

Figure 7.13 gives an example in which service management primitives, executed at a dedicated service management SAP, are related to normal data primitives. In this example, the *StartMonitoring* and *MonitorIndication* primitives are related to normal *DataRequest* and *DataIndication* primitives at a normal SAP. Such monitoring primitives can for instance be defined as part of a security management scheme. The *StartMonitoring* primitive includes as parameter the address of the SAP that should be monitored. After execution of this primitive, all data that is send and received at SAP A is offered as part of the *MonitorIndication* at the service management SAP.

7.1.2.2 The service manager

The service manager of our model for centralized service management (Figure 7.10) can be compared to the service managers as defined elsewhere [20][81].

In essence, our model presents a *functional model* that can be used by the designer during the architectural phase. The goal of our model is to show the division of functionality between service users and service provider. According to our model, the service manager should be considered as a service *user*. This is in line with our definition of service management (Subsection 5.2.3).

Models as described elsewhere can often be considered as *organizational models*. The goal of such models is to show the division of responsibility between organizational entities. These models may be particularly useful during the operational phase.
Applied to telecommunication (telecom) networks [106], organizational models show the division of responsibility between the *telecom company* and the *users of the telecom network*. Because the telecom company provides to its users a telecom service, the terms ‘telecom company’ and ‘service provider’ are often considered as synonyms [36]. Since service management will usually be performed by the telecom company, organizational models may describe service management as part of the service provider. This explains the difference between the term ‘provider’ as used in the telecom world and as used throughout this thesis (Figure 7.14).

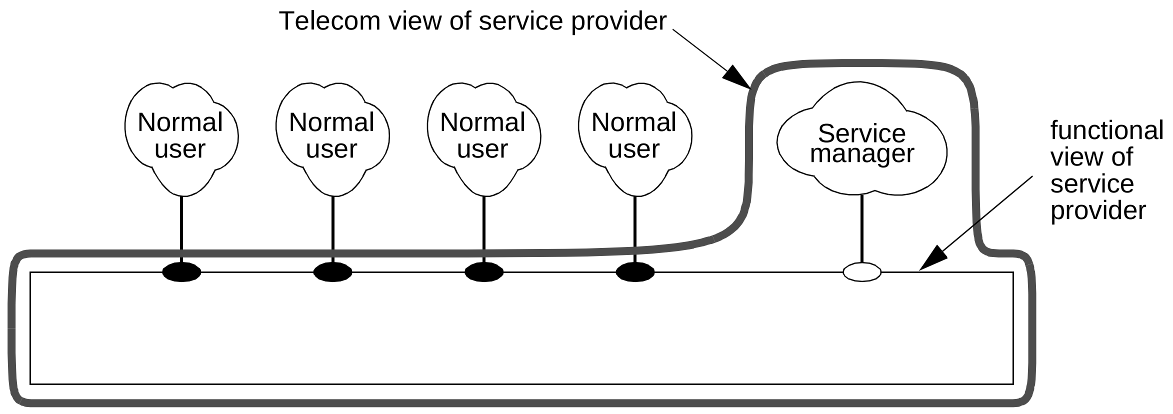

Telecom view of service provider
Normal user
Normal user
Normal user
Normal user
Service manager
functional view of service provider

Figure 7.14: Different views of service provider

It is possible to redraw the organizational model such that it resembles the functional model. This can be accomplished by moving the service management SAP from the top to the side of the service provider. The resulting model (Figure 7.15) may be attractive for its ability to clearly distinguish between service manager and normal users. It demonstrates that service users consider the service manager as part of the organization that provides the 'service'. Since the semantics of sideways connected SAPs has not been defined, Figure 7.15 should not be considered as a functional model.

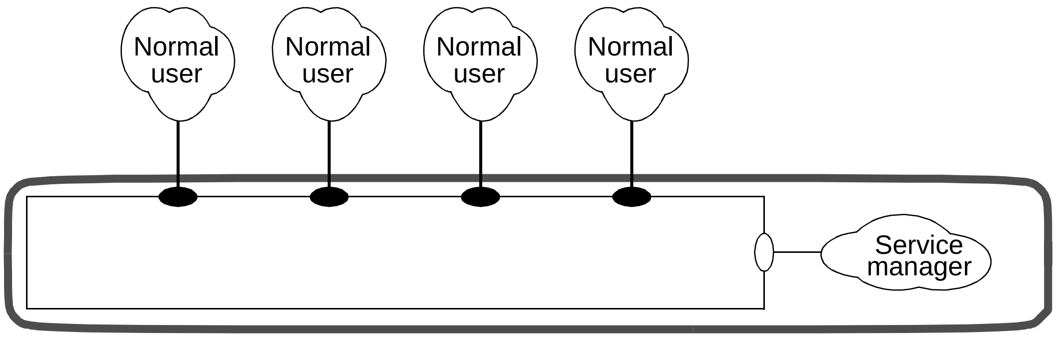


Figure 7.15: Service management is performed by the telecom organization

7.1.3 A model for hybrid service management

Distributed service management (Subsection 7.1.1) has, just as centralized service management (Subsection 7.1.2), certain advantages and disadvantages:

- Distributed service management allows users to take immediate action in case they experience problems with the service provider or in case they change their demands. With distributed service management it may be difficult to avoid interference in case different users want to initiate conflicting management actions. Also security is a potential weak point of distributed service management.
- With centralized service management it is much easier to avoid conflicting management actions and to guarantee a high level of security. The drawback of centralized service management, however, is that users who experience problems or who change their demands should always contact the service manager. This may be inefficient.

Example: Assume a NFS client gets no responses from a NFS server. To determine the problem cause, the client may be interested in the results of a reachability test, since such test will unveil whether the server application or the underlying UDP service causes the problem. In this example the best approach seems to distribute service management and allow each user to initiate a reachability test.

Example: Most networks allow the connection of new users and the removal of existing users. Depending on the particular design, connection and removal of users may involve service management interactions. Since unauthorised connection or removal attempts should be avoided, the right to initiate these service management interactions may be restricted to a single user. In this example the best approach seems to centralise service management.

The idea behind hybrid service management is to combine the advantages and avoid the disadvantages of the distributed and centralized approach. This can be accomplished by introducing different sets of service management interactions. Some of these interactions, such as the reachability test, may be initiated by all users, while other interactions, such as connection and removal of users, may only be initiated by a single central service manager. In the hybrid approach all users get service management capabilities, but some users get more capabilities than others.

The model for hybrid service management is shown in Figure 7.16. In this model the M' indicates that the associated SAP allows only a limited set of service management interactions to be executed; the M'' is associated with the central service management SAP and indicates that this SAP allows execution of the complete set of service management interactions.

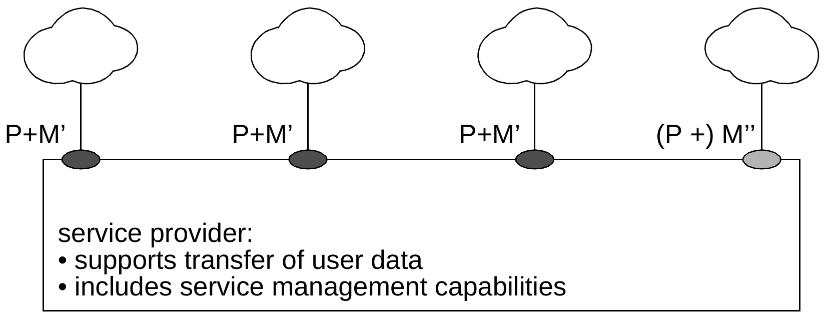

P+M'
P+M'
P+M'
(P +) M''
service provider:
• supports transfer of user data
• includes service management capabilities

Figure 7.16: Model for hybrid service management

Although the figure suggests the existence of two sets of service management interactions (at SAP M' and M''), it will generally be possible to define more than two sets. In case of a large public network, it may for instance be a good idea to define three different sets:

- One set of service management interactions that may only be executed by a single central service manager. This manager possibly belongs to the organization that owns the public network. The set of service management interactions may include interactions to add and remove network users.
- Another set of service management interactions that may only be executed by a small number of service managers. Each company that is connected to the public network may have their own service manager. The set of service management interactions will be smaller then the previous set and limited to the company to which the service manager belongs. The service manager may for instance create and delete group addresses for use within the company.
- The set of service management interactions that may be executed at all remaining SAPs. This set includes for instance reachability tests.

Hybrid service management is commonly found in real networks. The distributed and centralized approaches which were discussed in the previous subsections, should be seen as exceptional forms which are useful for explanatory purposes.

The addition of hybrid service management involves the same changes to the service definition as discussed in the previous subsections. These changes are therefore not repeated here.

7.2 A protocol management architecture

This section discusses the addition of *protocol* management functions to primary protocol functions. Subsection 7.2.1 demonstrates that protocol management can be expressed in terms of the same architectural concepts as used to express primary functions. Subsection 7.2.2 presents a functional model that shows how to perform protocol management from central locations; this model is described in terms of entities and underlying services. Subsection 7.2.3 presents the related physical model; this model shows how entities can be mapped upon subsystems.

7.2.1 Architectural concepts for protocol management

Within the OSI community protocols are usually described in terms of:
1) Individual protocol interactions.
2) The relationship between individual protocol interactions.
3) The set of protocol entities.

The idea that protocol management should be seen as an extension to the original protocol, implies that the addition of protocol management should have impact on the terms mentioned above.
The impact on the individual protocol interactions is discussed in Subsection 7.2.1.1; the impact on the relationship between protocol interactions is discussed in Subsection 7.2.1.2. In case the addition of protocol management leads to the definition of new (protocol management) entities, it becomes feasible to perform management from a central location. Discussion of this possibility will be postponed until Subsection 7.2.2.

7.2.1.1 Individual protocol interactions

The addition of protocol management may have the following consequences for individual protocol interactions:
- New fields may be added to existing PDUs.
- New PDUs may be defined.

New PDU fields

First an example is given to demonstrate that the addition of protocol management may involve the addition of special fields to existing PDUs. The example taken is the one of token ring, as defined by IEEE and ISO [47].

The token ring standard defines how data can be exchanged between stations that are connected to a ring. A station that has data ready for transmission, waits until it receives a free token. After reception of such token, the station captures the token and starts transmitting. The same station that puts the PDU on the ring is also responsible for the PDU's removal. There is some time elapse between putting a PDU on the ring and removing the PDU from the ring. During this time the responsible station may run into problems (it may for instance be switched off). As a result, the PDU will not be removed from the ring.

To detect such problems, a special management field has been added to the PDU. This field is the so-called 'monitor bit' and is used as follows. The transmitting station puts the PDU on the ring and sets the monitor bit to zero (Figure 7.17).

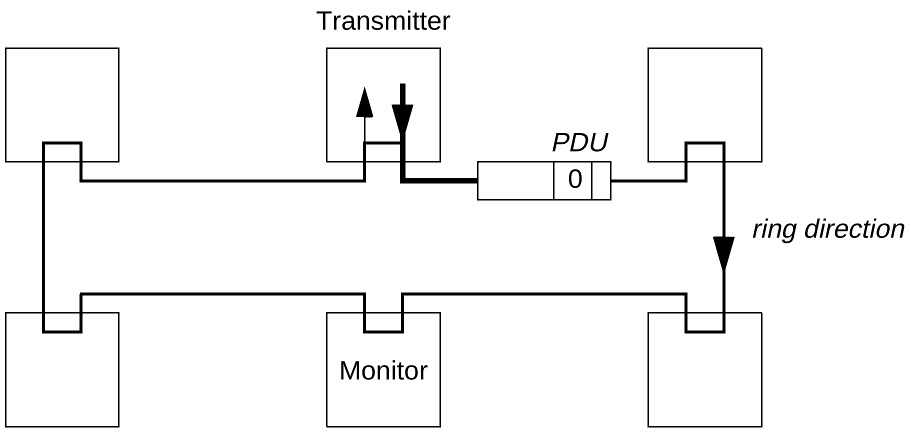


Figure 7.17: Token ring: transmission of PDU

The PDU passes all stations connected to the ring, including a special monitor station. This station is responsible for detection and correction of all kinds of errors[1]. As the PDU passes, the monitor checks the monitor bit and sets the bit to one (Figure 7.18).

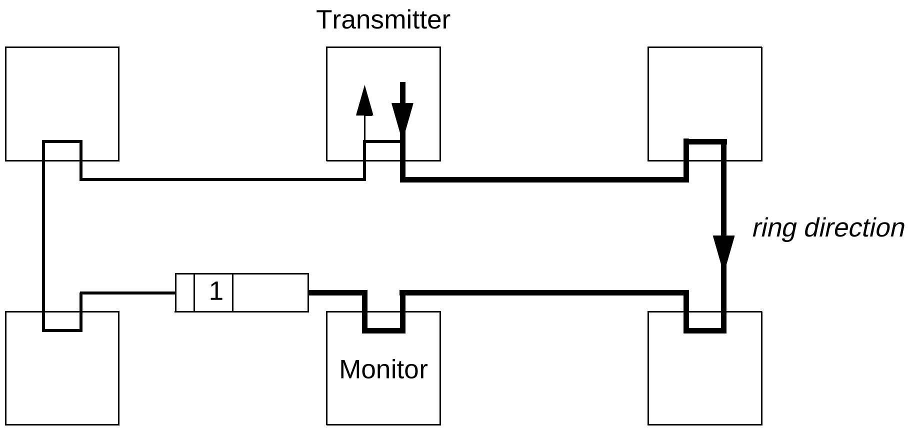


Figure 7.18: Token ring: monitor sets monitor bits

The PDU continues its travel along all stations connected to the ring. If an error occurs that results in the PDU not being removed from the ring, the monitor receives the PDU for a second time. The monitor inspects the monitor bit

1. There is only one station in the ring that performs monitoring functions. This station may also participate in the exchange of normal data PDUs.

and sees that the bit has already been set. The monitor concludes that an error has occurred and starts error correction procedures.

New PDUs

The same example can also be used to demonstrate that the addition of protocol management can involve the definition of new PDUs. The example explained that the correction of errors was the responsibility of the single monitor station. There is a chance however that also the monitor station breaks down. In such case the tasks of the monitor will be assigned to one of the other stations (the standard defines that every station must be able to perform the monitor role). To detect monitor break down, the monitor periodically transmits special *Active Monitor Present* PDUs. All stations within the ring check whether these periodic transmission occur. If the *Active Monitor Present* PDU fails to appear, all stations within the ring start a complex recovery mechanism. As a result, the station with the highest address takes over the monitor responsibilities.

Additional examples that demonstrate the need for new management PDUs can be found in the routing standards. The ES-IS routing standard [48], for example, defines *Hello* and *Redirect* PDUs.

Hello PDUs are used to distribute configuration information. The receiver of such PDUs knows that the station that generates these PDUs is alive and reachable. If necessary, the receiver adds a new entry to its forwarding table. If *Hello* PDUs are no longer received from certain sources, it is concluded that these sources can not be reached any more. As a result, forwarding tables will be updated.

Redirect PDUs are transmitted from intermediate systems to end systems. They are generated in case an intermediate system detects non-optimal routes. The PDUs contain the address of better intermediate systems that are closer to the destination and can also be reached by the end system. Upon receipt, the end system changes its forwarding table (which possibly contains a single entry).

7.2.1.2 New relationship between protocol interactions

Another possible effect of adding protocol management is a modification of the PDU sequence. Such modification may be needed to express the influence of management functions upon the operation of the primary functions. The nature of the new sequence will generally be an adjustment and not a complete change of the original sequence; deviations of the original sequence will only be displayed in abnormal cases (e.g. system failure). In many cases the effect of protocol management can be described in terms of additional *constraints* upon the normal PDU sequence.

> *Example:* If station X no longer receives *Hello* PDUs from station Y, X may stop the transmission of data PDUs to Y. The protocol definition

that results after the inclusion of management, should describe this modified relationship.

Also in case the addition of protocol management does not involve the definition of new PDUs or PDU fields, it is possible that the original PDU sequence will be modified. This can be demonstrated by the example of transparent bridges [52]. These bridges include filtering tables which are used by the primary protocol functions to determine whether incoming PDUs should *not* be forwarded over certain outgoing ports (= LANs). Without protocol management, the tables remain empty and incoming PDUs will be forwarded to all other LANs. With protocol management, the tables will be filled and PDUs will only be forwarded over selected LANs (Figure 7.19). Protocol management puts thus additional constraints upon the PDU sequence.

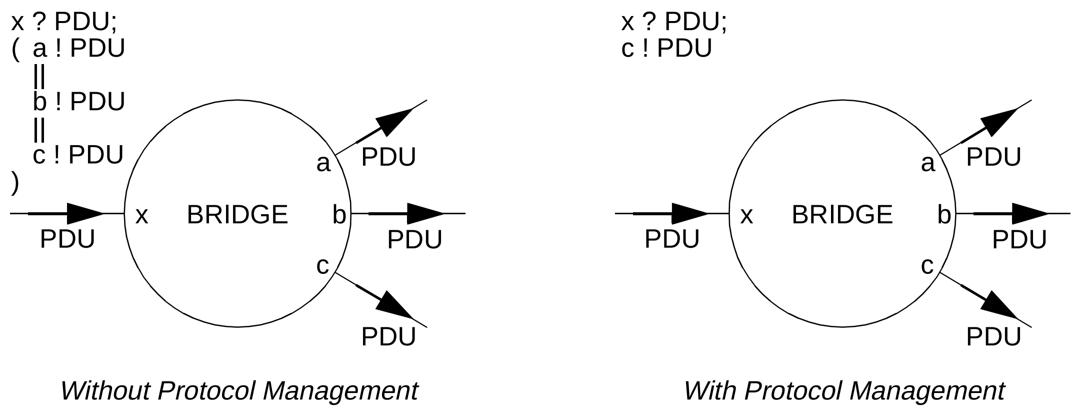


Figure 7.19: Transparent bridge

Let us explain how these filtering tables are used by the Primary Protocol Functions (Figure 7.20). After a PDU arrives on port x, the destination address is extracted from the PDU and used as input to the filtering table. If the filtering table contains the destination address, the table holds the outgoing port (LAN) over which the PDU must be forwarded. If this port is not the same as the one over which the PDU was received, the PDU will actually be forwarded. If this port is the same as the one on which the PDU was received, the PDU will be discarded. If the filtering table does not contain the destination address, the PDU will be forwarded over all possible ports, except the one over which the PDU was received. In case of empty tables, the incoming PDUs will thus be forwarded over all other LANs.

It is the task of management functions to initialize and maintain the filtering table. This task is performed as follows. After a PDU is received on port x, the *source* address is extracted from the PDU. This address is used to initialize and maintain the filtering table. The idea is as follows: if PDUs *from* a specific address are received on port x, port x must also be used to forward PDUs *to* that specific address. After reception of a PDU, the filtering table is consulted to check if the PDU's source address has already been included. If this is not the case, the address and port are added to the table and a timer is started that allows the entry to be timed-out (and removed). If the address has already

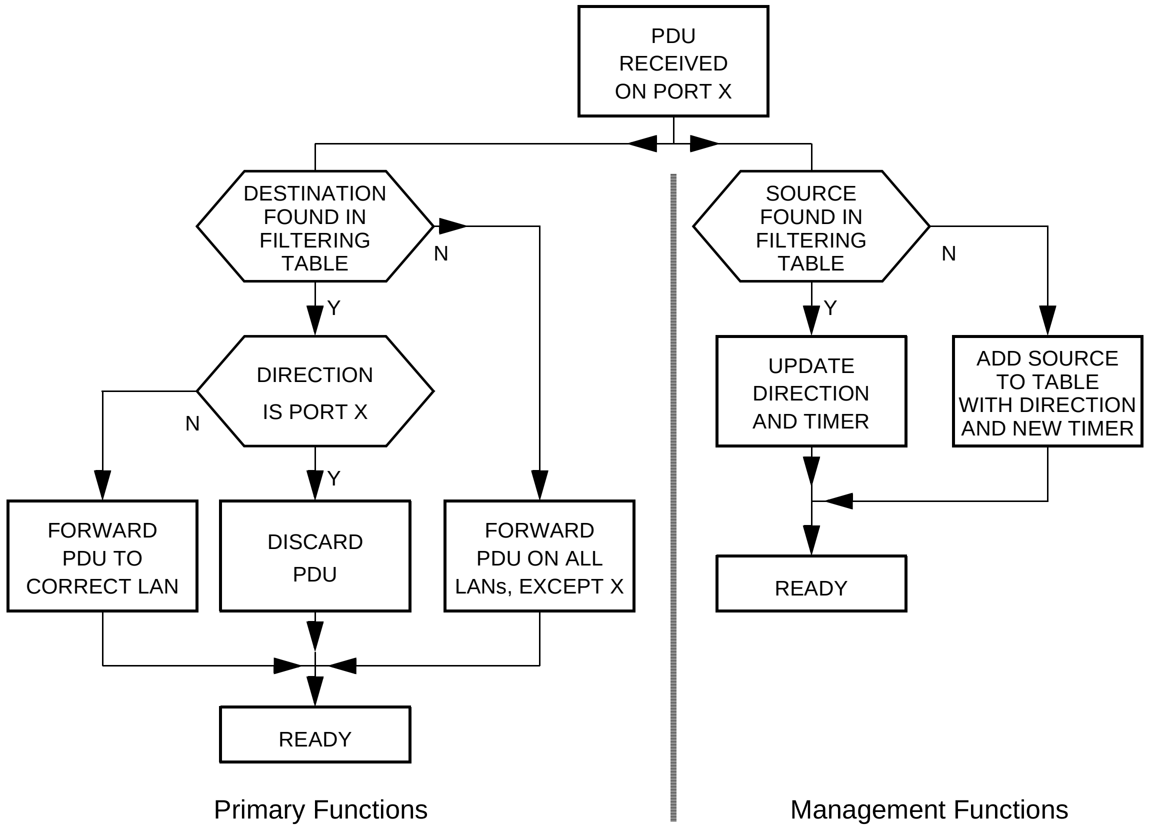

PDU RECEIVED ON PORT X
DESTINATION FOUND IN FILTERING TABLE
N
Y
DIRECTION IS PORT X
N
Y
FORWARD PDU TO CORRECT LAN
DISCARD PDU
FORWARD PDU ON ALL LANs, EXCEPT X
READY
SOURCE FOUND IN FILTERING TABLE
N
Y
UPDATE DIRECTION AND TIMER
ADD SOURCE TO TABLE WITH DIRECTION AND NEW TIMER
READY
Primary Functions
Management Functions

Figure 7.20: Transparent bridge: primary & management functions

been included, the port information will be replaced with the port number over which the last PDU was received. Also the timer will be restarted.

7.2.2 A functional model for remote protocol management

This subsection develops a number of models that show how protocol entities can be managed from one or more remote locations. The models will be developed in a number of steps. First the outcome of the initial design cycle(s) will be rephrased. Next the addition of protocol management functionality will be discussed. Since this functionality need not be distributed equally, special attention will be paid to centralized management (Subsection 1.3.2). Finally it will be shown how management interactions can be mapped upon underlying services and a functional model for protocol management will be presented.

Outcome of the early design cycles

During the initial design cycle(s) the focus is on the protocol's primary functions. These functions handle the transfer of user data; management issues will generally not be addressed. The outcome of the initial cycle(s) are a number of protocol entities plus the interactions that take place between these entities (the protocol). The terms *'data entities'* and *'data interactions'* are used to denote this outcome. Data interactions are carried out by normal data PDUs.

Figure 7.21 shows such outcome for layer N. The P within the data entities indicate that the entities perform *P*rimary functions.

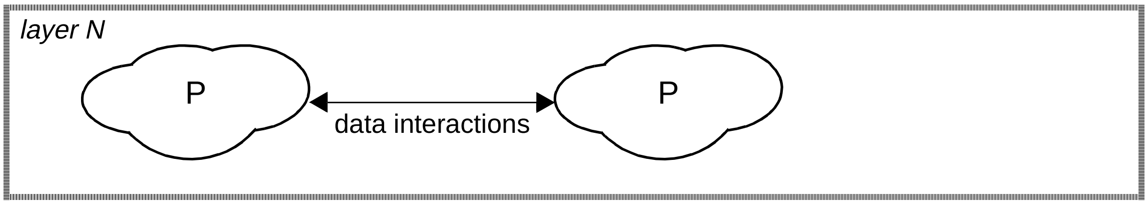


Figure 7.21: Outcome of early design cycles

Addition of management functionality

During the later design cycles management functionality will be added. This new functionality may or may not involve the introduction of new (management) entities.

First the case will be considered in which no new management entities will be introduced. Here the added management functionality will completely be performed by the existing data entities. This is illustrated in Figure 7.22. The figure shows that the data entities from the previous figure are now engaged in two kind of functions: *P*rimary functions (P) and *M*anagement functions (M)[1]. The interaction between both entities is carried out by the normal data PDUs, augmented with new management PDUs and / or management PDU fields (see also Subsection 7.2.1).

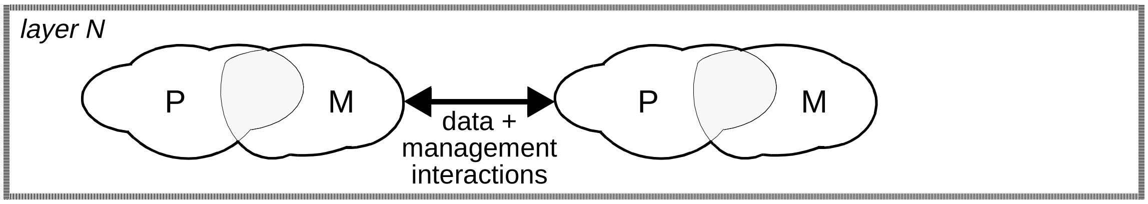


Figure 7.22: Addition of management functionality to existing entities

The figure shows that there is some overlap between Primary and Management functions. This overlap is drawn as a grey area and indicates the relationship between Primary and Management functions. It comprises those parts of the Primary functions that can be managed; in many cases these parts contain parameters and tables. In terms of Internet management such part represent the MIBs.

Now the case will be considered that new management entities are introduced. In such case the management functionality is performed by data entities *as well as* one (or more) management entities. Figure 7.23 shows the example in which a single management entity (indicated by M'') has been added. This entity 'controls' the primary part (P) of each data entity via the associated management part (M'). To enable such control, management interactions must be

1. Entities are enclosed by thick lines. According to the OSI definitions, entities may be engaged in multiple functions [43]. Functions are enclosed by thin lines.

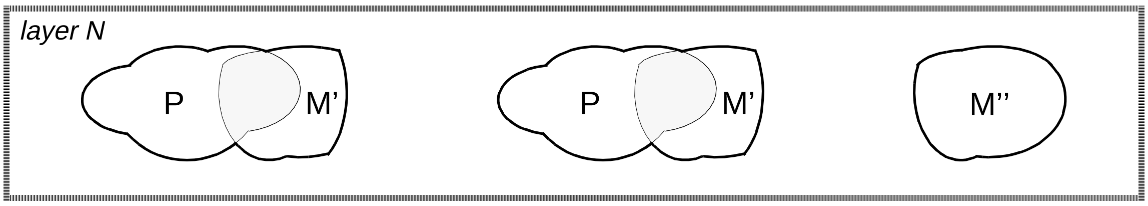


Figure 7.23: Addition of new management entity

exchanged between M'' and all M's. Note that data interactions will still be exchanged between both Ps (the interactions are not shown to keep the figure surveyable).

The part of the Management functionality performed by the normal data entities is usually different from the part of the Management functionality performed by the management entities. In case of Internet management this is for instance clearly formulated by their 'fundamental axiom', which says that "the impact of adding network management to managed nodes must be minimal, reflecting the lowest common denominator" [101][102]. According to this view most of the management functionality must be performed by dedicated management nodes. This difference in functionality is also depicted in Figure 7.23: the functionality within the data entities has a different form and name (M') then the functionality within the management entity (M'').

An interesting part of Figure 7.23 is that the management entity is part of the *same* protocol layer as the data entities. This is a logical consequence of the view that protocol management should be *added* to the layer that is managed[1]. It is the only possible way for data and management entities to exchange PDUs (in case management PDUs were exchanged between entities in different layers, the layering concept would be violated).
Although the idea that data and management entities reside within the same layer seems straightforward, the OSI management architecture does not model management this way. This is because the developers of this architecture confused abstraction levels (see also Subsection 2.3.1).

Centralized management

Dedicated management entities can be added to a design in case management decisions must be taken from special locations. The advantages and disadvantages of concentrating management functionality were already discussed in Subsection 1.3.2; the term *centralized management* was introduced to denote this kind of management. Centralized management is shown in Figure 7.24. The management interactions take place between the management entity (M'') and the management functionality within the data entities (M'). The data interactions take place between the primary functionality (P) within the data entities. Note that centralized management does not allow direct management interactions between data entities; changes in data entities can only propagate

1. Note that this way of modelling is in agreement with the general characteristics of protocol management, as discussed in Subsection 5.3.2.

to other data entities via the central manager. Interactions between management entity and data entities are usually initiated by the management entity.

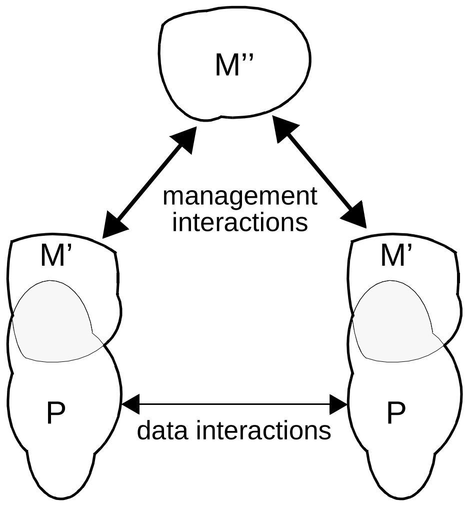


Figure 7.24: Centralized management

Instead of a single management entity, a whole hierarchy of management entities may be defined. Such hierarchy may be useful to enable different management domains and policies to be defined [107][108]. If the scope remains protocol management, all entities still reside within the same layer as the data entities. Note that this management hierarchy should not be compared to TMN's functional hierarchy (Section 3.4), where the relationship between different types of management (service, protocol and element management) is expressed.

In the remainder of this text the focus will be on the single manager approach; the idea of hierarchies of protocol managers will not further be discussed.

The functional protocol management model

The final step in the development of a functional model for protocol management, is the mapping of protocol interactions upon underlying services. As seen before, two different kinds of interactions exist: data interactions and management interactions. These interactions may be treated by the designer in different ways.

A first possibility is that the designer decides to use for both kinds of interactions a single PDU type. In this case management information will be put in special fields of data PDUs. This was already discussed on page 125, where the example was given of the monitor bit of token ring.

Since management information will be conveyed as part of normal data PDUs, the *same* underlying service will be used for the transfer of data as well as management information. This possibility can be compared to OSI's layer operation. Figure 7.25 shows this possibility; in case no separate management entity has been introduced, the entity M'' at the right can even be removed.

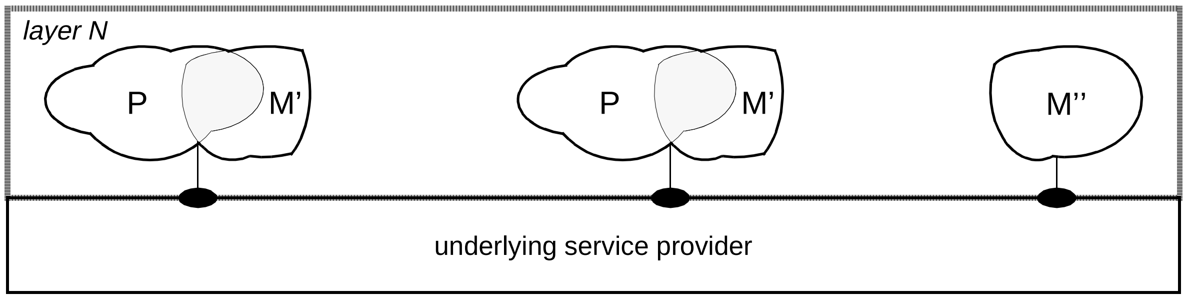


Figure 7.25: Data and management interaction use same underlying service

Figure 7.25 can also be used to model the case in which the designer uses *different* PDUs for data and management, but maps both kinds of PDUs upon the *same* underlying service. This can be compared to OSI's layer management.

The most interesting case, however, is that the designer decides to use for both kinds of interactions different PDU types and map these PDU types upon *different* underlying services. Such decision can be made for various reasons.

One reason is to guarantee that user data and management information will be transferred independently of each other. Such independent treatment ensures that management information exchanges can not be hampered by potential problems in the service provider for user data. Even in case of faults or congestion, management information will still arrive in time.

Another reason might be that management interactions put different demands upon the underlying service as compared to data interactions. This can be expected in case of management of the lower protocol layers. The differences may be such, that it will not even be possible to use a single underlying service for data and management PDUs. Examples of such differences are:

- Data unit size. In case of for instance ATM, data PDUs have fixed lengths of 53 bytes: 48 for user data and 5 for protocol control information. PDU sizes for management are usually of variable length and much longer.
- Quality of Service. At data link level it may be acceptable that some data PDUs get lost. It may not be acceptable, however, that management PDUs get lost. There may also be differences with respect to throughput and delay. As opposed to management PDUs, the exchange of data PDUs may require a high throughput and a fixed delay (e.g. ISDN B-channel).
- Reachability of managed entities. There may be a desire to manage several data link entities from a central location. In general these data link entities may be connected to multiple physical service providers. According to the OSI Reference Model, data link entities connected to one physical service provider can not interact (at data link level) with data link entities connected to other physical service providers. In case of meshed topology networks, no central location can be found where all physical service providers meet. To allow the exchange of management information between the centralized manager and the various data link entities, introduction of a special service provider may become inevitable.

The analysis of realization consequences of introducing a special service provider for the exchange of management information will be postponed until Section 8.2. A model that illustrates this special service provider is shown in Figure 7.26; the model can be used to explain all types of protocol management discussed so far. In the remainder of the text the term *functional reference model for protocol management* is used to denote this model.

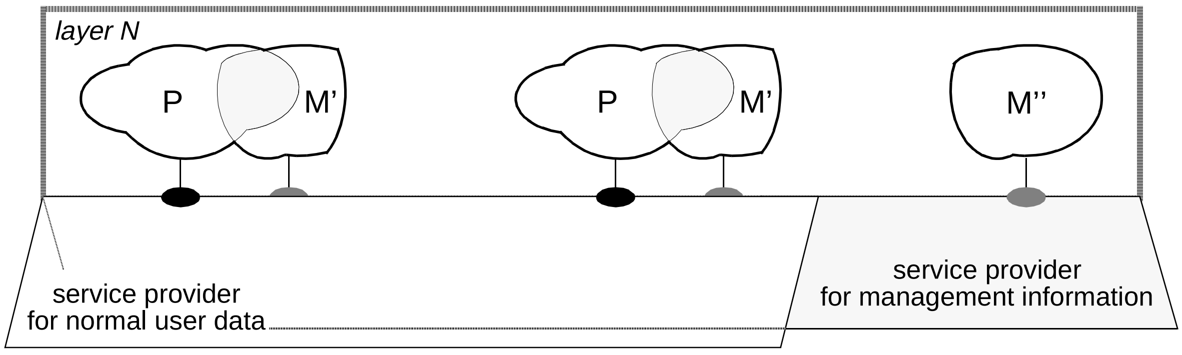

layer N
P
M'
P
M'
M''
service provider for normal user data
service provider for management information

Figure 7.26: Functional reference model for protocol management

7.2.3 A physical model for remote protocol management

The physical protocol reference model can be derived from the functional model by mapping functional entities upon subsystems[1]. The special management entity M'' (Figure 7.26) can be implemented in different ways:

- The management entity can be implemented together with a data entity. There is thus one subsystem that includes a management plus data entity.
- The management entity can be implemented as a separate subsystem.

First the case is treated the management entity is implemented in a subsystem together with a data entity. This subsystem is called the *management subsystem* and shown in Figure 7.27 as the rectangle at the right. All other data entities are implemented as separate subsystems; the term *data subsystems* is used to denote these subsystems (the rectangle at the left). The figure does not show the underlying service(s) used by the various subsystems: it is possible that management information is exchanged over the same underlying service as user data, but it is also possible that management information is exchanged over a dedicated service.

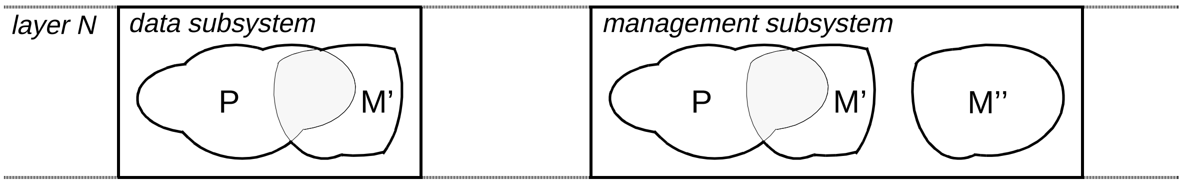


Figure 7.27: Management entity is implemented together with a data entity

1. According to the OSI definitions, single subsystems may consist of multiple entities [43].

The following example illustrates the feasibility of this approach.

> *Example:* The TCP/IP protocol suite defines a function to determine the network (IP) address during start-up [85]. This function is called the 'Reverse Address Resolution Protocol' (RARP) and operates as follows. As part of the start-up procedure, each data entity assembles a RARP request containing a unique number identifying the sending entity. This number usually is the same as the underlying MAC SAP address[1]. After the request has been assembled, it will be broadcasted over the underlying MAC service. Although the request is received by all other entities, only the management entity reacts. This entity is called the 'RARP server' and uses the unique number to determine the associated network address. This address is returned as part of a response PDU.
> A RARP server is usually implemented in software and requires a limited number of resources (CPU time and memory). Introducing a separate system to implement such server is needless and would be a waste of money. In the case of RARP, the management entity is thus implemented in combination with a data entity.

Next the case is treated the management entity is implemented as a separate subsystem. Also in this case a distinction is made between data and management subsystems. Usually only a limited number of management subsystems exist. As opposed to the previous case, management subsystems will not perform primary functions (Figure 7.28).

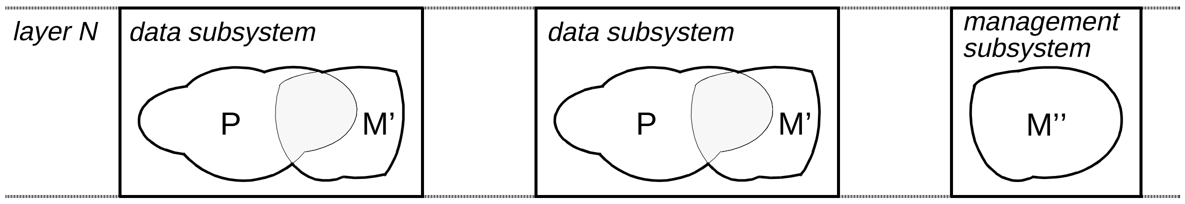


Figure 7.28: Management entity is implemented as separate (sub)system

The introduction of a dedicated management subsystem can be attractive. It is a way to concentrate most of the management functionality at a single location. Centralization of management functionality can be attractive for several reasons; it allows for example human beings to perform (some parts of) the management functionality (see also Section 1.3).

The physical protocol management model

Figure 7.29 is a further development of Figure 7.26 and illustrates the *physical reference model for protocol management*. It shows a special service provider for the exchange of management information between M'' and the various M's.

1. RARP is primarily used in LAN environments on top of the MAC service. Manufacturers of MAC chips guarantee that each chip includes a unique MAC address. Each MAC service supports broadcasting.

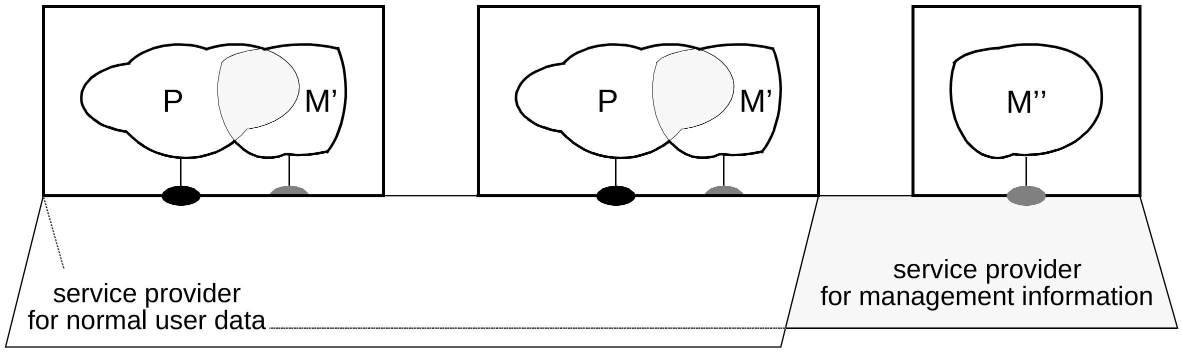


Figure 7.29: Physical reference model for protocol management

7.3 An element management architecture

The objective of the realization phase is to structure individual subsystems by mapping protocol functions upon realization components. The internal structure of a subsystem can be described in many ways, and is dependent upon the choice of the realization platform. In case of software solutions on top of single tasking operating systems (e.g. MS-DOS), the structure can be expressed in terms of functions, procedures and / or objects. In case of multitasking platforms (e.g. UNIX), the structure can be expressed in terms of processes and Inter Process Communication (IPC). In terms of hardware, realization structures can be expressed in terms of boards, CPU’s, memories and backplanes.

In fact there are many possibilities for realization structures. It is important to recognize that these possibilities are the same as with stand alone systems: network systems can be realized by the same components as stand alone systems. As a consequence, there is no fundamental difference between element management and management of stand alone systems. It must therefore be possible to describe both in terms of the same concepts.

Given the fact the scope of this thesis is on *network* management (thus management of *distributed* systems), these general concepts will not be discussed here. Instead, the remainder of this section concentrates on the *distributed* implementation of element management functionality.

7.3.1 A model for remote element management

The starting point of the realization phase is the protocol definition. Although a protocol definition describes the behaviour of all subsystems, there is no requirement to realize the different subsystems in the same way. In fact, different manufacturers will develop different realizations. As a result, also different element management solutions will evolve.

To develop the remote element management model, the individual subsystem of Figure 7.30 will be examined. The externally observable behaviour of this subsystem is defined at protocol level. The protocol does not define the subsystem's internal structure; from a protocol point of view the subsystem can be considered as a black-box. Such black-box interacts with its environment via his underlying SAP(s). At these SAP(s), SPs will be executed. The data part of these SPs contain the PDUs directed towards the subsystem.

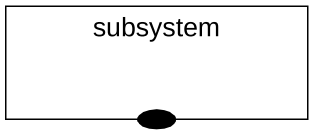


Figure 7.30: Individual subsystem

Example: A possible structure of a router subsystem is shown in Figure 7.31. The router connects three underlying data links and uses for each link a dedicated CPU and memory board. An advantage of this structure is the maximum of parallelism and absence of active central components; the fact the routing table should be scattered over all boards is a disadvantage. A high speed internal bus connects the three boards. All components may be realized within the same cabinet.

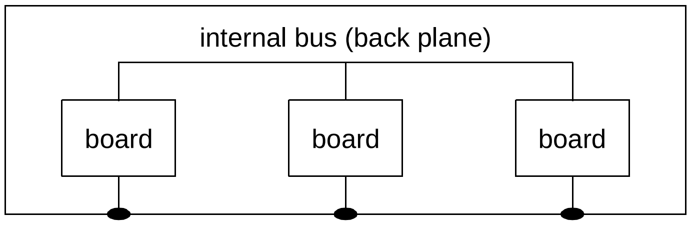


Figure 7.31: Example of the internal structure of a router subsystem

Element management functions are needed to supervise the various realization components. Which element management functions should be added, must be determined during the realization phase.

As first step in the development of an element management model, the element management functions will be realized at the same location as the components that must be managed (Figure 7.32). This implies that the components that realize the primary functions, reside at the same location as the components that take care of their management. Both kind of components are part of a single subsystem.

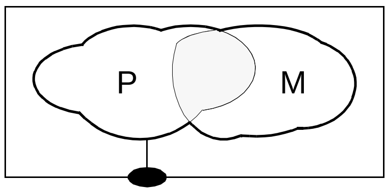


Figure 7.32: Addition of element management

The figure shows two kind of entities within the subsystem: P entities to satisfy the primary protocol functions and M entities to perform element management functions. Drawing these entities does not imply a specific realization structure: both entities are only drawn to indicate that *different* kind of functions should be performed.

The figure shows that only P entities require interaction via an underlying SAP. The fact that M entities do not require a connection with an underlying SAP, can be understood after one realizes that the element management functions reside at the same location as the managed components and no need for management interactions with external entities exists. M functions will therefore not be connected to an underlying SAP.

In some cases it may not be feasible to realize all M functions within a single cabinet. Such case is for instance if parts of the M functions are performed by a human manager, as will be illustrated by the following example.

> *Example:* Figure 7.33 shows a particular solution for adding element management to the router of the previous example. In this solution, a special Element Management (EM) board has been introduced within the router cabinet. This EM board communicates with the other boards via the internal bus. Connected to the EM board are sound alarms, indication lights and buttons. These sounds, lights and buttons realize the interaction with the human manager outside the cabinet.

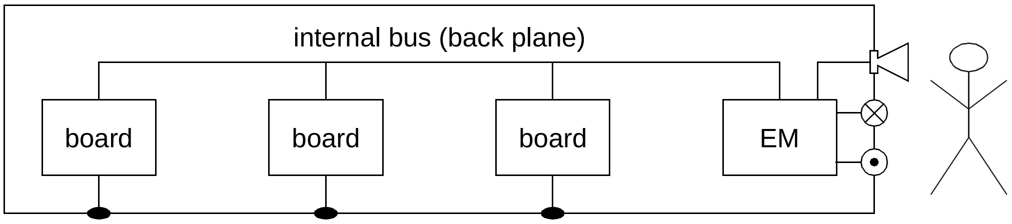


Figure 7.33: Human manager manages router components

Since cabinets may be placed in special rooms and cellars, it may be desirable for the human operator to manage the router from a remote place (Figure 7.34). The rectangle at the left shows the P entity and parts of the original M entity are realized within a single cabinet. The rectangle at the right indicates that the remaining parts of the M entity are realized at another location. In

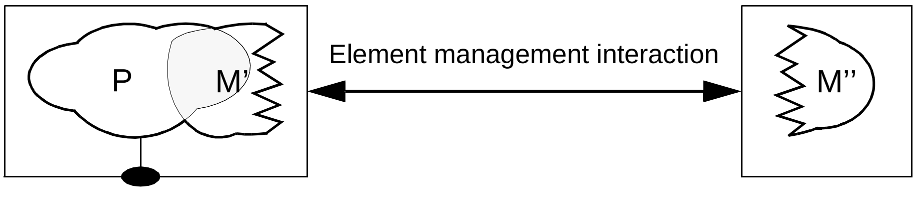


Figure 7.34: Remote element management

order to cooperate, element management interactions are needed between both locations, which ask for some additional functionality in both Ms.

> *Example:* Figure 7.35 illustrates this solution for the previous router example. The sound alarms, indication lights and buttons reside at the same place as the human manager. A communication line is needed to connect the equipment at both locations.

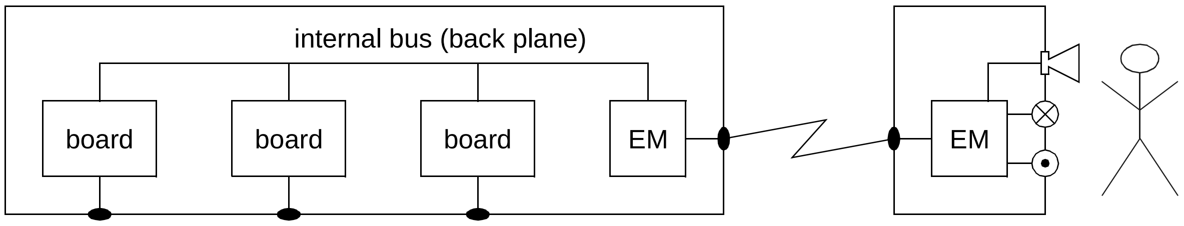


Figure 7.35: Some element management parts reside at another location

The final step in the development of the remote element management model, is the addition of a service provider for the exchange of element management information. This is shown in Figure 7.36.

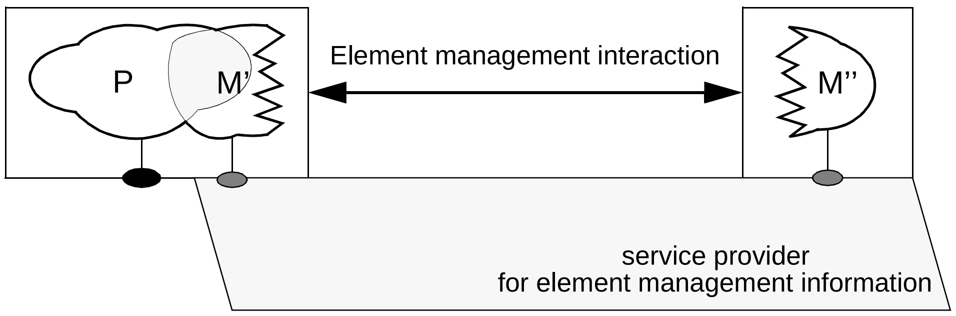

P
M'
Element management interaction
M''
service provider
for element management information

Figure 7.36: Model for remote element management

Difference between protocol and element management model

The model for remote element management has a strong resemblance with the model for remote protocol management. Still there is an important difference, which becomes apparent if we consider management of multiple systems from a central location.

In case of protocol management, a *single* management entity can be used to control *multiple* protocol subsystems. This entity relates protocol management interactions with one protocol subsystem to protocol management interactions with other protocol subsystems. If the centralized management entity receives for example an alarm from one protocol subsystem, it changes the routing table in another subsystem.

In case of element management, *multiple* management entities will be needed to control all protocol subsystem realizations. For each realization, a dedicated centralized element management entity will be needed. Since the realization of a subsystem has no functional relationship with the realization of other subsystems, there will not be a *relationship between the various element management entities* either. Of course the various element management entities may still be located within a single cabinet. This does not imply a functional relationship, however. Figure 7.37 illustrates this difference.

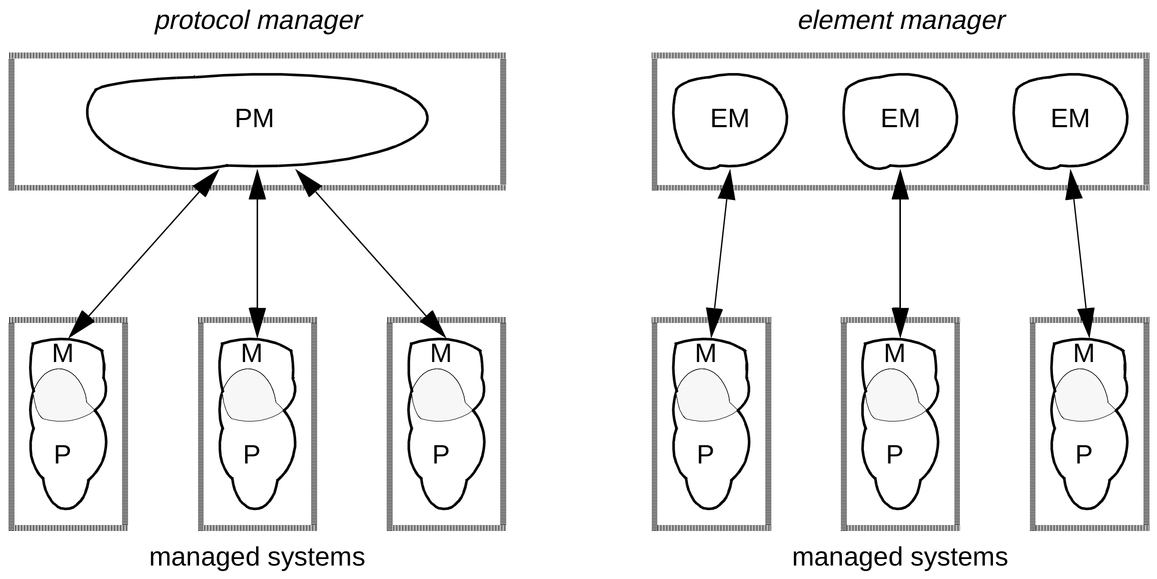


Figure 7.37: Difference between protocol and element management

7.4 Concluding remarks

There are several parts of OSI, TMN and Internet management that can be related to the functional model for protocol management of Subsection 7.2.2. Discussing this relationship is useful for the following reasons:

- Current work on management can be placed in a better framework. As a result, the understanding of network management may be improved.
- By using existing parts, the applicability of the functional model for protocol management can be demonstrated without having to define yet another set of management protocols and services.

After the various relationships have been considered (Subsection 7.4.1 until Subsection 7.4.3), an advantage of the models of the previous sections will be presented (Subsection 7.4.4).

7.4.1 OSI management

The way in which general OSI concepts are used in this chapter to develop a number of management models is interesting, since the architectural models as produced by SC 21/WG 4 do not apply these concepts in a *consistent* way (see also Section 2.3). In fact, the models as presented in this chapter may be considered as *alternatives* or *replacements* of (the communication oriented parts of) the OSI Management Framework [44] and the Systems Management Overview [53]. Since the models of this chapter show how primary functions can be extended with management functions, they can also be regarded as *extensions* of the OSI Reference Model [43].

As shown in Figure 7.38, it is possible to relate several parts of OSI management to the functional reference model for protocol management as presented in this chapter. The figure shows two managed entities (one at the left and one in the middle) plus one manager entity (at the right).

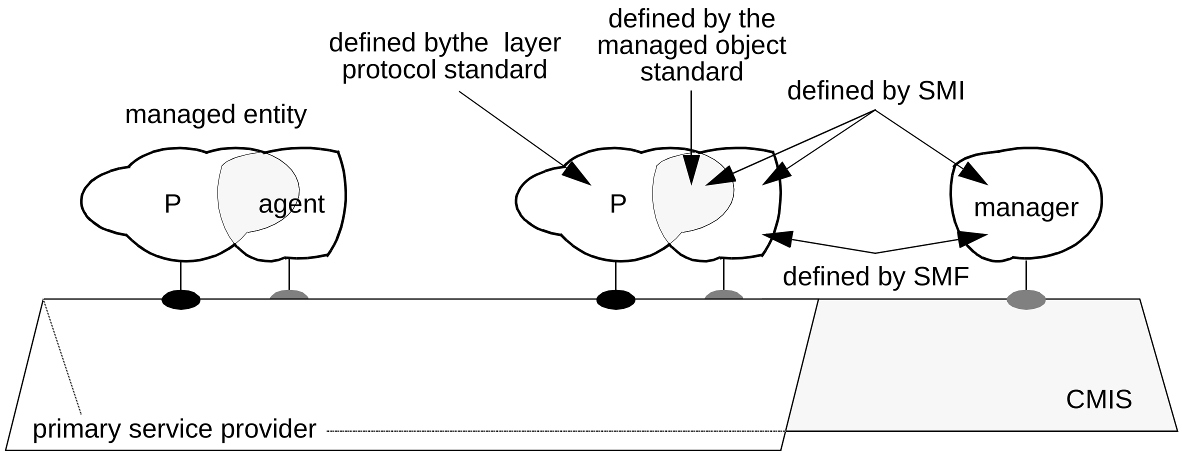


Figure 7.38: Relationship to OSI management

The two managed entities together perform the primary functions (P) of the ‘layer protocol’. ISO has defined for each layer a separate ‘layer protocol’ standard (e.g. the transport protocol standard for the transport layer [45]). These protocol standards define a number of variables and commands (indicated by the dotted areas within the managed entities), which can be used for management purposes. In terms of OSI management, these variables and commands represent the ‘managed objects’. It is important to understand that the ‘layer protocol’ standards do not define the name and syntax of these objects, nor do they define the operations that may be performed and the notifications that may be emitted by these objects. For this purpose, ISO has defined special *managed object standards* (e.g. ISO 10737 for the transport layer [56]). General rules explaining *how* objects should be described, are defined by the Structure of Management Information (SMI) standards (ISO 10165). These rules should not only be known to describe managed objects, but also to interpret management information within the manager entity. Some SMI standards define ‘generic management information’, which in terms of this thesis can be considered as meta-management information. Generic information is located in the right most part of each managed entity, which also performs the Systems Management Functions (SMF - ISO 10164). These functions will be activated by the manager, the manager must therefore also understand the SMF.

The primary functions within the managed entities use the primary service provider (e.g. the network service provider) for the exchange of their normal user data. Management information between manager and managed entities is transferred over the Common Management Information Service provider (CMIS).

It should be noted that the model of Figure 7.38 is a *functional model*, which does not show the mapping of functions upon implementation components. For this reason the primary service provider and the CMIS provider are shown as separate providers[1]. In later phases of the design these providers should be mapped upon implementation components. While performing such mapping, the designer *may* decide to use the *same* implementation components for both service providers (such decision may be taken to keep the implementation costs low). In such case an implementation relationship will be introduced between both providers. This relationship will be further discussed in Section 8.2.

The OSI management group has also introduced an ‘object oriented’ approach for management. In many of its management standards, this object oriented approach plays a dominant role. The management models as defined in this chapter do not say anything about object orientedness. As will be explained in Chapter 9, it is possible to extend our architecture to include this approach.

7.4.2 TMN

As opposed to the other management approaches, TMN has defined similar concepts as service, protocol and element management. Instead of ‘protocol management’ TMN uses the term ‘network management’, instead of ‘element management’ TMN uses the term ‘network element management’.
Although the TMN recommendations identify the need for service and element management, they do not define dedicated models for both forms of management.

The functional reference model for protocol management can be used to illustrate TMN function blocks and reference points. Figure 7.39 shows as example the OSF and NEF function blocks, as well as the q_3 reference point. Since TMN has adopted many OSI management standards, the figure may be refined to include these standards. The resulting figure becomes equivalent to Figure 7.38.

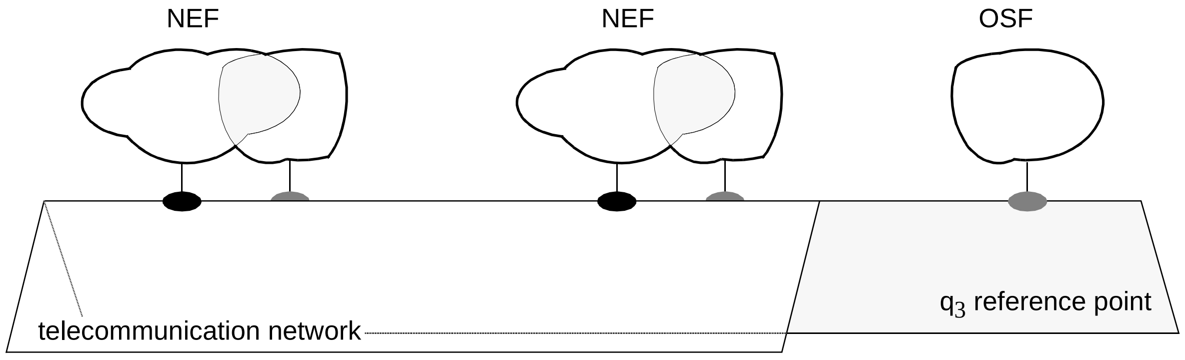


Figure 7.39: Relationship to TMN

1. This is an important difference with the models as defined by OSI management. See also Subsection 2.3.2.

7.4.3 Internet management

The Internet management standards can be related to the functional reference model for protocol management in a similar way as the OSI management standards. As example, the TCP transport protocol will be considered [84]. The variables of TCP that can be managed by a remote manager, are defined as part of the MIB-II [94]. This MIB not only defines the variables that belong to the transport layer, but also variables that belong to some of the other layers. The structure of MIBs is defined by the SMI [90]; the dotted area within the transport layer entity must therefore be described according to this SMI. The SMI must also be known by the manager entity (Figure 7.40).

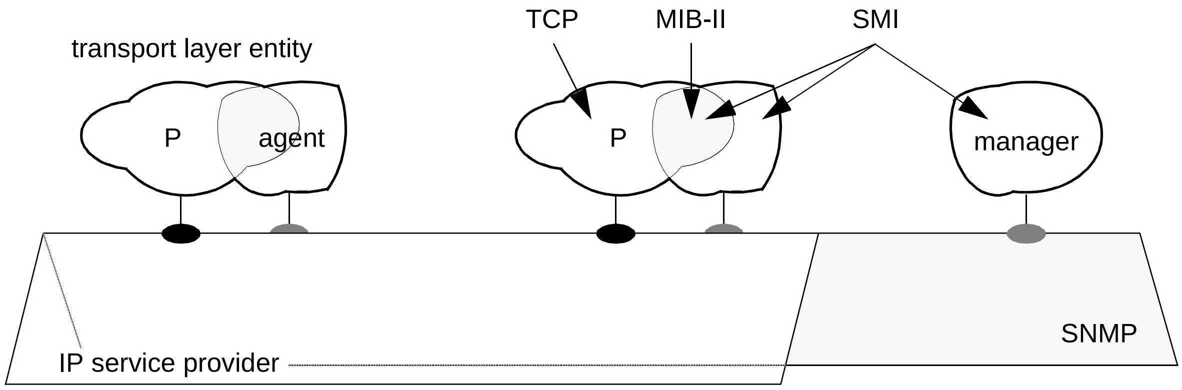


Figure 7.40: Relationship to Internet management

It may be surprising that Figure 7.40 does not show SNMP [92] as the protocol between managed and managing entity, but as part of their underlying service. This is because an important part of SNMP deals with functions like the Basic Encoding Rules, which should be considered as general transfer functions and not as management specific functions.

There are some parts of the SNMP protocol, however, that require knowledge of the MIB structure and are therefore specific for management. These parts may therefore be regarded as part of the protocol between managed and managing entity. Examples of such parts are the Get-Next PDU and the view concept as defined by SNMPv2 [97].

7.4.4 Advantages

As opposed to other management approaches, this thesis proposes to model management functions in conjunction with primary functions. An advantage of this approach, is that also the relationship between both kind of functions will be modelled. As a result, it becomes possible to determine the effect of management upon the primary functions. The models as presented in this chapter can thus be used to construct simulation tools that help managers to predict the effect of management operations.

At this moment there is already a great demand for simulation tools. It may be expected that this demand will increase, particularly if the current situation lasts in which managers are confronted with thousands of different managed object types. Because of this large number, managers may be unsure which objects should be modified to obtain a desired result. For such managers, tools may be helpful allowing them to simulate management actions in advance.

Many of the current simulation tools are hand-made and unique for specific types of networks. Their development is expensive and their maintenance requires a lot of effort. Automatic development of tools would be a step in the right direction. The models as presented in this chapter can in principle be used for such development.

Such automatic approach uses the protocol and service specifications of both primary and management functions, as developed during the design phase. These specifications should be written in a formal language (e.g. LOTOS) and are read by a general purpose simulation tool (Figure 7.41). This simulation tool can basically be the same as the ones used by the network designers during the design phase for the simulation of normal network protocols (e.g. Lite [28]). Because management operations will be initiated during the operational phase, it will be possible to prolong the use of such simulation tools up to the operational phase.

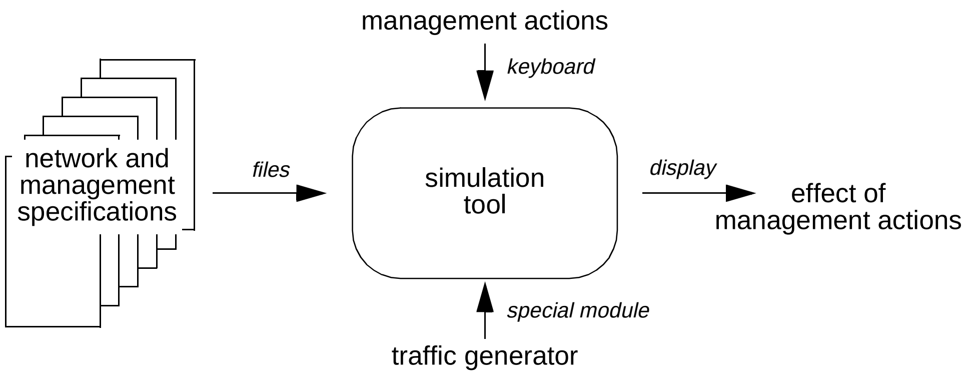


Figure 7.41: Simulating the effect of management actions

At this moment it is difficult to judge the feasibility of this approach. Important factors determining this feasibility are the evolvement of formal techniques and the level in which such techniques will be accepted.

8: Management information service providers

8 Management information service providers

Chapter 7 has presented various architectural models showing how designs can be extended to include management. Some of these models introduce special service providers for the exchange of management information (e.g. Figure 7.26). The purpose of this chapter is to discuss these service providers. To distinguish between these service providers and the service providers supporting the primary functions, the following terms will be used:

- Management Information Service Provider (MISP): the special service provider that has been introduced in the later design cycles to support the exchange of management information. The term 'MISP' is chosen for its resemblance with OSI terms (e.g. CMIS and CMIP)
- Primary Service Provider (PSP): the service provider that has been introduced in the early design cycles to support the exchange of user data on behalf of the primary functions.

It is not the intention of this chapter to develop service *definitions* for the MISPs; service primitives, parameters and a relationship between these service primitives will therefore not be proposed. Instead, the objective of this chapter is to discuss:

- The relationship between the various MISPs that may be introduced by the designer (Section 8.1).
- The relationship between the MISPs and the PSPs (Section 8.2).
- The question whether introduction of additional service providers is specific for management, or that also other design issues may require the introduction of multiple underlying providers (Section 8.3).

8.1 Several management information service providers

A good approach to design networks is to spread the various network functions over a number of functional layers. A first step in such design is to decompose the N-service provider into a N-protocol and (N-1) service provider. During the initial design cycles, the focus of this decomposition will be on the primary

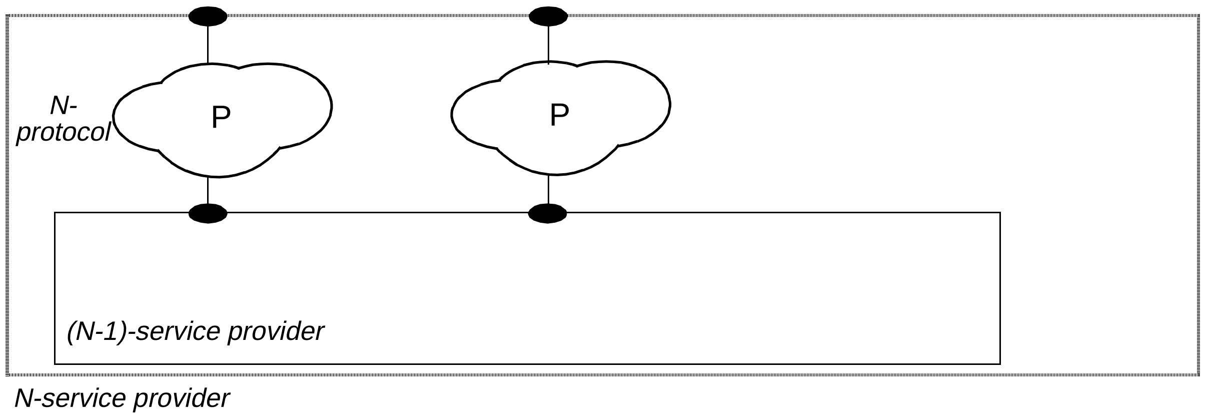


Figure 8.1: Decomposition of the N-service provider

functions and not on management issues. The model showing this decomposition is drawn in Figure 8.1; it is equivalent to the models generally found in textbooks on computer networks.

8.1.1 Addition of management

After the primary functions of the N-protocol and (N-1)-service have been defined, management will be added. This addition is performed during the later design cycles and involves:

- The extension of the N-protocol with protocol management functions.
- The extension of the (N-1)-service with service management functions.

N-protocol management

In Section 7.2 several possibilities were discussed to add protocol management to a design. One of the possibilities was to perform protocol management from a remote location. Given the importance of this possibility, a specific 'functional reference model for protocol management' was developed (Figure 7.26). This model showed how protocol entities performing primary functions could be managed from a central location. To allow the central manager to access and control the primary functions, the protocol entities were extended with some limited form of management functionality. This management functionality was called 'agent' functionality. The central manager performed the major part of the management functionality and could be considered as master; the managed entities performed only a limited amount of management functionality and could be seen as slaves. To enable the exchange of management information between manager and agents, a dedicated MISP could be introduced.

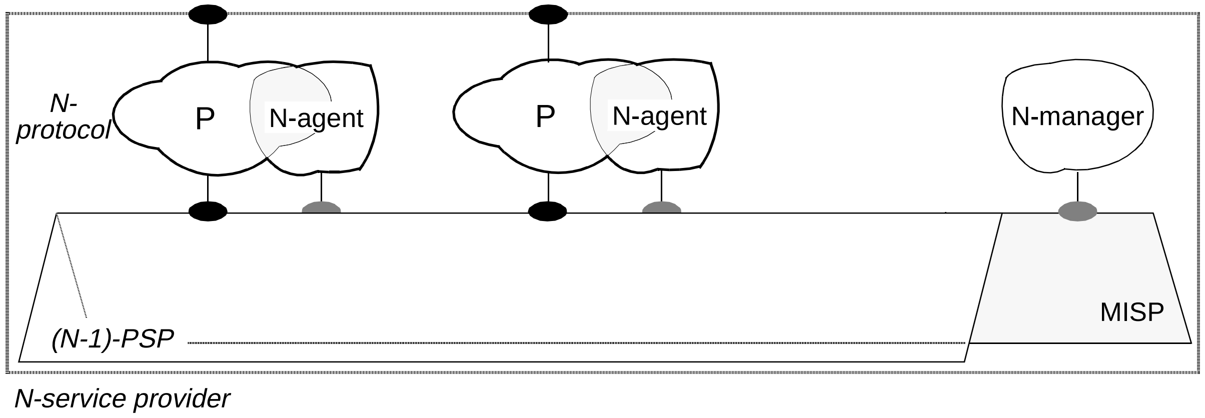


Figure 8.2: Addition of N-protocol management

Figure 8.2 is a further development of Figure 8.1 and shows the addition of N-protocol management. It is actually the same as the 'functional reference model for protocol management' (Figure 7.26). It shows two kinds of underlying services:

- The (N-1)-service provider, which supports the exchange of user data on behalf of the primary functions. To express this support for the primary functions, the name (N-1)-PSP will be used.
- The MISP, which supports the exchange of management information between manager and agents.

(N-1)-service management

It is important to understand that the addition of management to the decomposed structure of Figure 8.1 will not be restricted to the N-protocol layer; also the (N-1)-service provider must be extended with service management functions (note that Figure 8.2 called this service provider the (N-1)-PSP). The models resulting from this extension have been discussed in Section 7.1; they did not recognize a need to introduce special MISPs for the exchange of service management information. For the purpose of this section it is therefore not necessary to proceed this discussion of (N-1)-service management.

8.1.2 Decomposition of the (N-1) service provider

After management functionality has been added to the N-protocol and the (N-1)-PSP, the layered design approach continues with decomposing the (N-1)-PSP. The fact that the PSP, and not the MISP is decomposed, is a logical consequence of the fact that primary aspects should be addressed first; decomposition of the MISP may be postponed until later design cycles.

The decomposition of the (N-1)-PSP yields a (N-1)-protocol and a (N-2)-service provider (Figure 8.3). Again the designer focuses upon the primary functions and ignores, for the time being, the way these functions should be managed. As a consequence, the (N-1)-protocol definition includes only primary func-

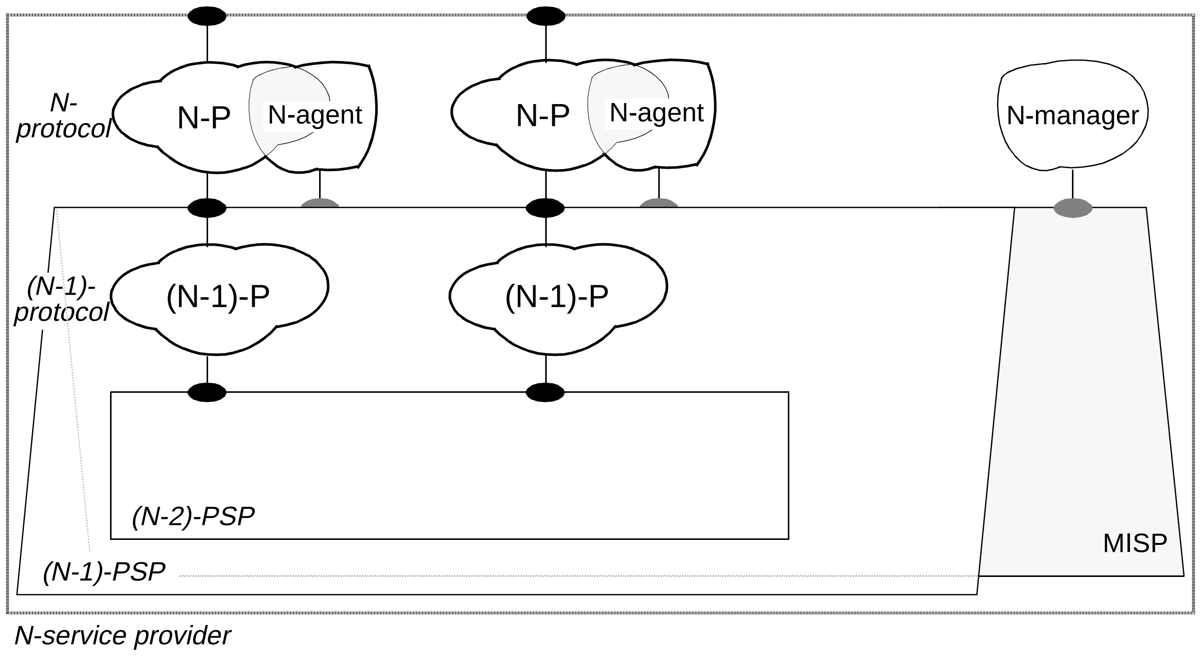


Figure 8.3: Decomposition of the (N-1)-service provider

tions and the (N-2)-service supports only the exchange of user data on behalf of the protocol entities performing these primary functions. The (N-2)-service can therefore be regarded as a (N-2)-PSP; there is no need to introduce another MISP yet.

8.1.3 Addition of management

After the (N-1)-PSP has been decomposed, the designer should consider the problem of managing the resulting structure. Again, addition of management should involve (N-1)-protocol, as well as (N-2)-service management. The introduction of (N-1)-protocol management encompasses the definition of (N-1)-agents, a (N-1)-manager and a new MISP. To emphasize the difference between the MISP already identified in the previous decomposition and this new MISP, Figure 8.4 calls the first MISP-A and the second MISP-B. As discussed previously, (N-2)-service management has no further consequences for this model and need not further be discussed.

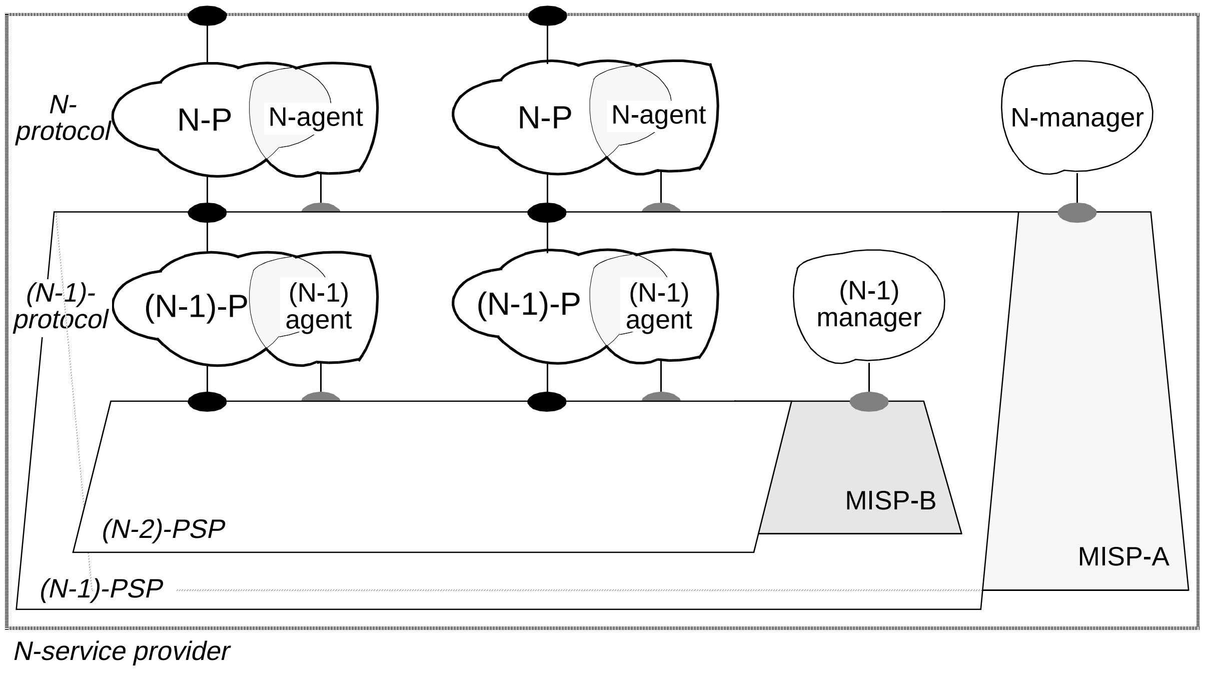


Figure 8.4: Addition of (N-1)-protocol management

8.1.4 Integration of MISPs

Each time a PSP is further decomposed, a new MISP may be introduced. The number of MISPs that may be added, is limited by the number of protocol layers that must be managed. It is likely that in case of management of an OSI protocol stack the designer introduces more MISPs then in case of management of an Internet protocol stack. As discussed in Section 7.3, it is also possible that additional MISPs will be required to support element management. The designer may thus be confronted with a large number of MISPs.

Fortunately, many of these MISPs satisfy similar requirements. This may be understood by considering the following examples:

- The requirements for the exchange of management data belonging to a data link layer sliding window mechanism, are the same as those belonging to a transport layer sliding window mechanism.
- The requirements for the exchange of counter values indicating the number of bits received by a physical layer entity, are the same as those indicating the number of open application connections.
- The requirements for the exchange of MAC addresses are the same as for the exchange of X.400 address.

The fact that many MISPs satisfy similar requirements, implies that many MISPs provide the same kind of service. In case we abstract from the primary functions and concentrate upon the management functions, it is even possible to represent and implement these MISPs as a single integrated service provider. Figure 8.5 shows such representation. The large service provider supports various groups of users: a group consisting of N-agents plus a N-manager, another group consisting of (N-1)-agents plus a (N-1)-manager etc. Since there is no communication between the various groups, the groups may perform different protocols and can be considered as closed user groups.

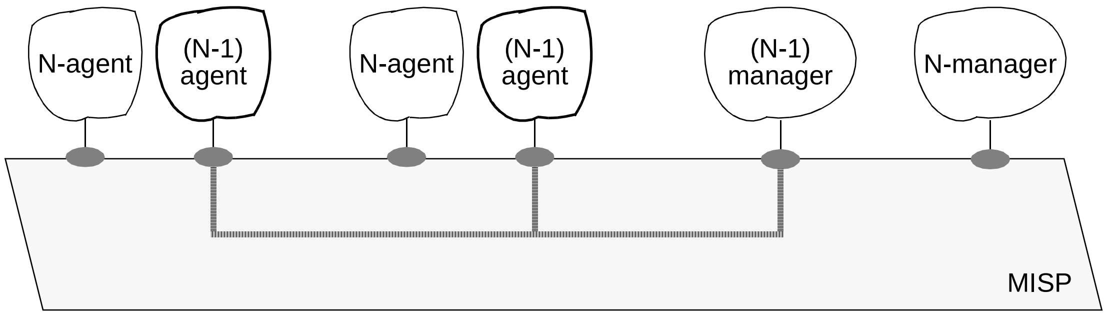


Figure 8.5: Single service provider for management information exchanges

In many cases the MISPs should support a service similar to OSI's presentation service. In such cases a natural conclusion may be that the managers and agents perform application layer interactions. The flaw of this conclusion is that it completely ignores the primary functions and concentrates upon the derived management functions.
A better approach seems to base the naming of interactions upon their primary functions; interactions between a transport layer manager and transport layer agents should for instance be considered as transport layer interactions, and not as application layer interactions.

Although it is possible to give many more examples to demonstrate the similarity in MISP requirements, it should not be concluded that it will *always* be a good idea to require for *all* management information exchanges the same kind of underlying MISP. This becomes apparent when existing management solutions are analysed. Such analysis shows that it may be better to distinguish between several types of MISPs:

- A type that is commonly used supports the point-to-point communication between manager and agent. This type is useful in case of centralized management. The required service resembles the OSI presentation service.
- A type that supports *broadcast* communication between manager and agent. This type may be useful in case the exact address of the manager or agent is unknown. A good example provides the Reverse Address Resolution Protocol (RARP), as discussed on page 135.
- A type that supports transaction processing and thus commitment control. Such type may be useful to allow a manager to change management information in *multiple* agents as an atomic action. It guarantees that either all management information will be changed, or none will be changed [29][116].
- A type that supports broadcast communication between entities which possess equal management capabilities. All entities communicate with each other, which implies that a master-agent relationship does not exist. A good example is provided by the intra-domain IS-IS routing protocol.

> *Example:* The intra-domain IS-IS routing protocol [55] uses a link state algorithm. This algorithm prescribes that network layer entities periodically send 'link state PDUs'. Whenever an entity generates such PDU, it puts in this PDU a list of all its neighbours, plus the cost to reach each neighbour. The link state PDU must be broadcasted to *all* other network layer entities in the network (thus not only the neighbours). In this way, each entity receives from all other entities neighbour information. This information allows each entity to build a topology map of the network, which in turn can be used to adjust the forwarding table.
> The fact that link state PDUs should be broadcasted over the entire network, has important consequences for the MISP. MISPs that perform well in master-agent environments, may not be good enough to satisfy the needs of the link state algorithm.

8.2 Primary versus management service providers

In the models that have been presented thus far, a distinction was made between PSPs and MISPs. From a conceptual point of view, the PSPs and MISPs are independent (see for example page 133). This conceptual independence does not imply however that PSP and MISP *must always* be realized with different physical components. Separate realizations are of course possible, but it may also be possible to use the *same* physical components for the realization of PSP and MISP.

> *Example:* The idea to use the same realization component for conceptual independent things, has many real world examples. Consider for example a word processor and a spreadsheet application. Both applications are conceptually independent, although they may use the same CPU, operating system calls, display, file system etc.

> Even in case the word processor and the spreadsheet applications are installed on different computers, there may be an undesired dependency when both computers get their power from the same electricity lines.

Existing management architectures follow different approaches with respect to the question of separated versus shared realizations. An example of separate realizations is provided by TMN, examples of shared realizations are provided by OSI and Internet management.

Shared realizations are attractive because they are generally cheaper then separate realizations. Their major disadvantage is the undesirable dependency introduced during the realization phase. Since PSP and MISP use the same physical components, failure of a component may impact both PSP and MISP. The bizarre effect of this may be that management performs well in case the primary functions perform well, and management breaks in case the primary functions break. Management may thus not be available at times it is needed most. Therefore shared realizations are less suited for fault management purposes.

In case of OSI management, this undesired dependency is taken for granted and no measures are taken to avoid potential problems. In case of Internet management, this dependency is not taken for granted. This is demonstrated by statements like "when all else fails, network management must continue to function, if at all possible" [101]. The Internet management group therefore decided to use the most primitive MISP still satisfying the requirements of management interactions: UDP. UDP provides a connectionless transport service and its realization requires less components in comparison to its connection-oriented counterpart TCP. UDP is thus less susceptible for problems then TCP.

8.3 The idea of special management service providers

This section discusses the question whether the introduction of special underlying service providers is characteristic for management, or that also other design decisions may lead to the introduction of new underlying service providers. The question is interesting, since it gives a better idea whether the addition of management should be regarded as a distinguishable part of the design or not.

A good example which may be helpful to answer this question is ‘out-of-band signalling’. The term ‘out-of-band signalling’ was originally introduced in the telephony world to indicate "that signalling information was conveyed outside the band of frequencies normally used for messages" [41]. Signalling information is needed to establish and release connections, but may also be used during connections to modify, for example, characteristics of the connection.

With the arrival of networks for mobile communication, signalling information may even be exchanged independent of connections. The term ‘out-of-band signalling’ is presently used in a more general sense, to indicate that user data and signalling information are treated differently during their transport through the network.

The reason user data and signalling information are treated differently, is because they have different requirements.

In the world of digital telephony, user data is conveyed as 8 bit samples that are offered to the network every 125 µs. To keep these samples in sequence and to maintain synchronization, the transfer delay should be fixed. Loss and corruption of samples will be annoying, but surmountable.

As opposed to the isochronous character of user data, signalling information has an asynchronous character. Signalling information is offered to the network at irregular intervals, and does not have a fixed length. Loss and corruption of signalling information can not be tolerated, but strict delay requirements do not exist.

The difference in these requirements may be expressed in the design by the introduction of different underlying services. The service used to transfer user data between switches should have an isochronous character and a data length of 8 bits. The service used to transfer signalling information should have an asynchronous character and a variable data size.

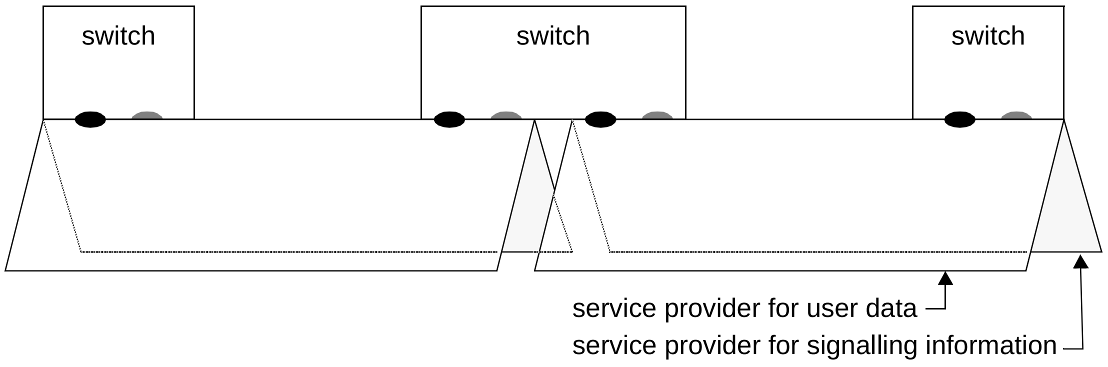

switch
switch
switch
service provider for user data
service provider for signalling information

Figure 8.6: Out-of-band signalling

The model for out-of-band signalling is given in Figure 8.6. Although this model has been described in literature (e.g. [104][76]), it has not been incorporated into OSI standards.

This example of out-of-band signalling shows that design problems that are not related to management, may also require the introduction of multiple underlying service providers. The conclusion must therefore be that the idea to introduce multiple underlying service providers is not exclusive for management; this idea can therefore not be used to make a clear distinction between management issues and other design issues.

9: Management protocols

9 Management protocols

This chapter provides an analysis of the management interactions taking place between managing and managed systems. These interactions have been modelled in Chapter 7 and allow remote managers to obtain management information and initiate management operations; the term ‘management protocol’ will be used to denote these interactions.

The analysis provided by this chapter may be helpful to improve the understanding of management protocols as well as management information models. With this understanding, it may be easier for readers to compare existing management protocols (e.g. SNMP and CMIP) and judge their fundamental differences.

An investigation of current management literature indicates that three major approaches towards management protocols can be identified [110][114]. These approaches differ with respect to the semantics of their management interactions. The approaches are:

- The ‘Variable Oriented’ (VO) approach: in this approach the manager operates on *variables* within managed systems. Variables can directly be read and modified via commands like *Get* and *Set*. Possible variables are: routing tables, window sizes and error counters. SNMP provides the best example of the VO approach (Chapter 4). Also early versions of OSI management used this approach (Chapter 2).
- The ‘Command Oriented’ (CO) approach: in this approach the manager issues imperative *commands*. Each command has well defined semantics. Possible commands are: reset protocol machine, use other gateway (redirect) and close all connections. Routing protocols and token ring management provide good example of this approach.
- The ‘Object Oriented’ (OO) approach: in this approach the manager operates on so-called managed *objects*. Managed objects have attributes and behaviour. The manager sends operations to, and receives notifications from the object. OSI management is the best example of this approach (Chapter 2).

The VO approach will be discussed in Section 9.1; the CO in Section 9.2. These sections discuss for each approach the characteristics and impact on the design process. Section 9.3 discusses the OO approach, and demonstrates that this approach may be considered as a mixture of both VO and CO approach.

9.1 The Variable Oriented Approach

With the Variable Oriented (VO) approach, managed systems make accessible to their managing systems all the variables that reflect and control the behaviour of their primary functions. From a manager's point of view, these variables are put into a structure which is called the MIB (Figure 9.1).

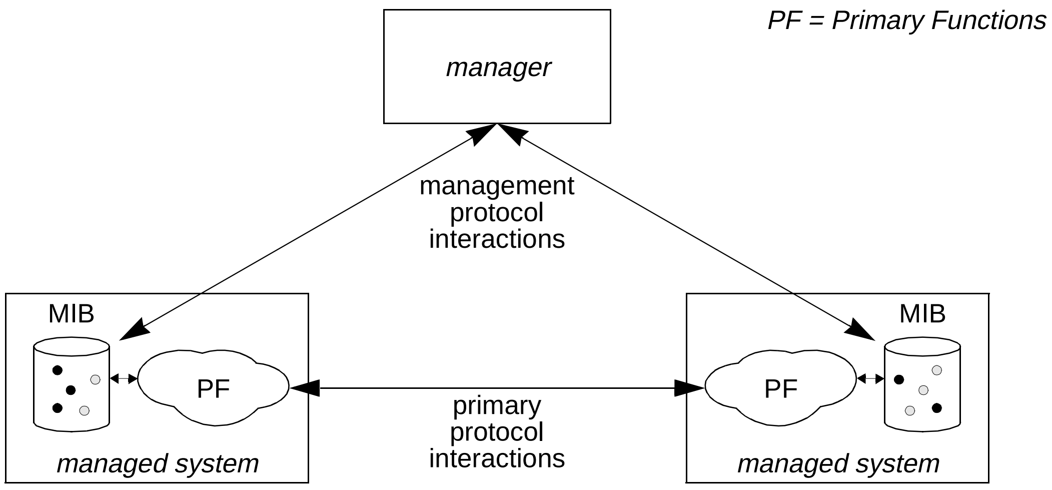


Figure 9.1: Variable oriented approach

> *Example:* the address of a system is an example of a simple, and the routing table an example of a complex variable. Both variables can be read and set by management. There are also variables that can be read but not be set by management. An example of such a variable is a counter for CRC errors.

In the VO approach, managers do not specify the *goals* of management operations, but the *means* to reach these goals. Management interactions are thus defined at a relatively low level of abstraction. This low level of abstraction has led to the introduction of the term 'remote debugging'. Usually there are a large number of variables and a small number of management PDUs. In its bare form, only two PDU types are needed: *GET* PDUs to read and *SET* PDUs to modify variable values. In general, VO management may be inefficient with respect to the use of communication and transmission resources (bandwidth).

> *Example:* Assume the goal of a management operation is to reset a system. To reach this goal, the manager should issue for each relevant variable a *SET* PDU. Efficiency may be improved by allowing a single PDU to include multiple *SET* requests.

9.1.1 Management functions

The VO approach can be characterized by the fact that most management functions will be performed by the managing systems and only a few by the managed systems.

The functions performed by the managing systems may be considered as the 'management intelligence'; these functions are sometimes called 'management applications'. They monitor the variables within the various managed systems and decide when and how to modify which variables. They initiate *GET* and *SET* operations.

> *Example:* The application within a managing system monitors for example the CRC counters within managed systems and concludes that the quality of the transmission line between two systems has dropped below an acceptable level. The application decides to take the line out of service to allow repair. Before taking the line out of service, the application determines which forwarding tables should be adjusted to route traffic around the problem area (Figure 9.2).

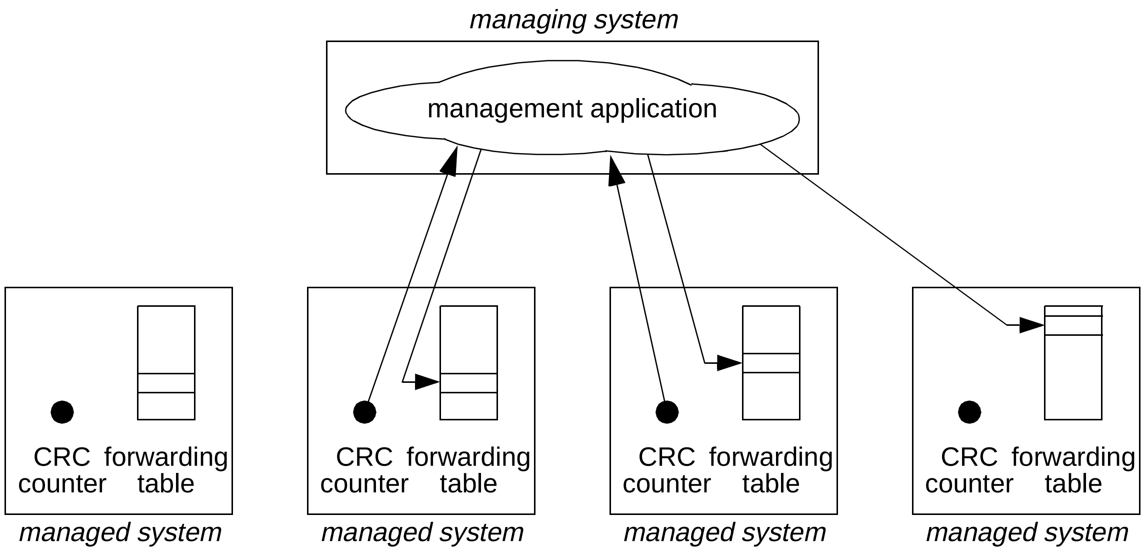


Figure 9.2: Managing system performs management applications

Another characteristic of the VO approach is that at least two PDUs should be defined: one to read variables (this type is usually called *GET*, although the name *READ* may also be used) and another to set variables (this type is usually called *SET*, although the names *WRITE* or *PUT* may also be used).

Sometimes additional PDU types are defined, such as *GetNext* to get the next variable that is contained within the MIB or *GetBulk* to get multiple variables.

The semantics of *GET* and *SET* remain the same for all the variables within the protocol layers that can be managed. *GET* and *SET* can even be used to manage other things than networks. In some publications these PDUs are therefore called 'polymorphic' commands [37].

Although most management functions can be performed by the managing systems, some functions must still be performed by the managed systems.

A function that should always be performed, is a function that inspects the management PDUs to determine which variables should be accessed. To support this function, a naming tree may be defined to uniquely identify each variable. Since managed systems may include large numbers of management variables, the realization of this function may require the introduction of hash-tables [26].

Managed systems should also perform some *communication oriented functions*. To facilitate for instance the use of an abstract syntax, these functions may include encoding rules (e.g. BER). To prevent unauthorized managers from reading and modifying variables, these functions may include access control and authentication. All of these functions can be considered as MISP internal protocol functions; they should be performed by managing as well as managed systems (Figure 9.3).

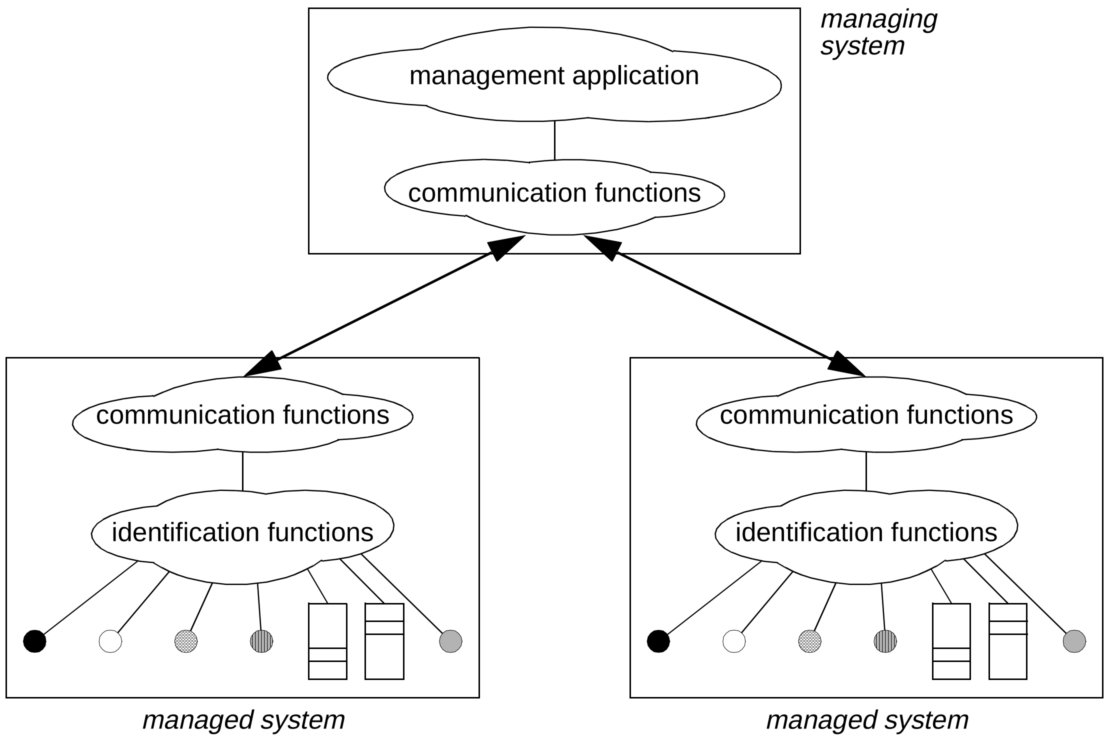


Figure 9.3: Different kind of management functions

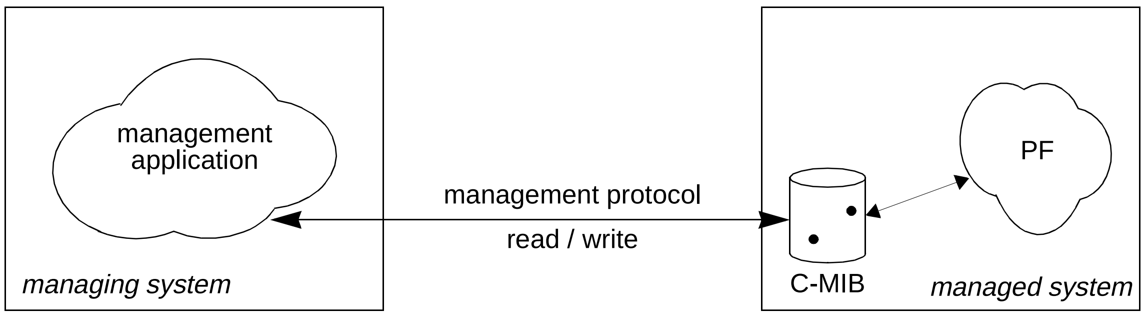
9.1.2 Management variables

Two groups of management variables can be identified:

- Control variables: these variables *control* the primary protocol functions.
- Information variables: these variables *contain information* about the protocol's primary functions.

Control variables

Control variables have a direct impact upon the operation of the primary functions (Figure 9.4). By modifying these variables, the management application is able to change the behaviour of the primary functions. *GET* commands are used to obtain current control variable values, *SET* commands are used to change these values. Most control variables should be readable and writable.



PF = Primary Functions
C-MIB = MIB containing control variables

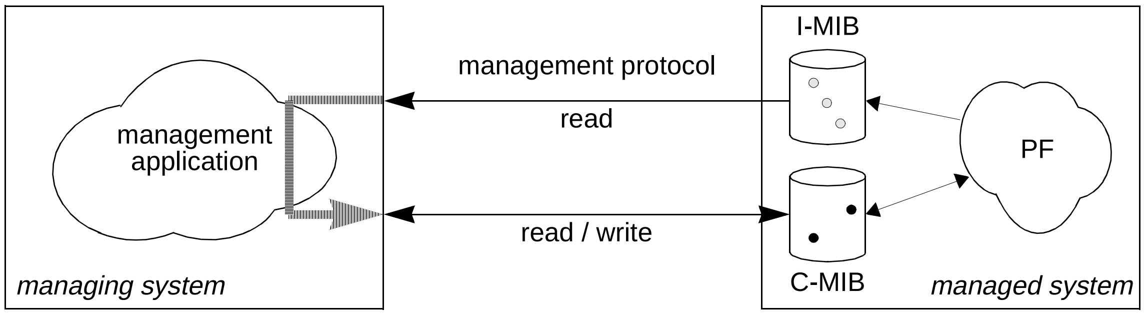
Figure 9.4: Control variables

Example: An example of a control variable is the address variable of a system. Primary functions operate directly upon this variable: after reception of a data PDU this control variable is read to determine if the data PDU has reached its destination. The forwarding table, the maximum window size parameter and time-out parameters provide other examples of control variables.

Information variables

The management application must make intelligent decisions with respect to the question when to modify which control variable. To make these decisions, the management application needs a certain amount of information. Part of this information can be obtained by reading control variables. In many cases however, these variables may not provide sufficient information: additional *information variables* may be needed (Figure 9.5).

To obtain information about the operation of primary functions, the management application reads these information variables. Modification of these variables does not make sense, since the operation of the primary functions does not depend upon their values. Information variables need therefore only be readable; they are used by the management application to determine when to modify which control variable.



PF = Primary Functions
I-MIB = Part of MIB containing Information variables
C-MIB = Part of MIB containing Control variables

Figure 9.5: Information and control variables

Example: the following variables are examples of information variables: a variable that counts the number of incoming PDUs, a variable that counts the number of transmission errors and a variable that indicates how long the system is already active.

The SNMP MIBs (Figure 4.9) define more then 2500 variables. Of these variables, more then 75% represent information variables. Of all information variables, more then 50% represent counters.

9.1.3 Impact on the design process

The definition of management variables and applications can be performed in a certain sequence (Figure 9.6).

The control variables influence the behaviour of the primary functions and can be identified as part of the definition of these functions ①. Control variables can thus be identified during the first design cycle(s).
After the control variables have been identified, they should be structured into a MIB. This involves the assignment of names, syntax and access capabilities (control variables have usually read / write capabilities).

After the control variables have been identified and structured into a MIB, the designer may develop the management applications ②. These applications represent the management intelligence and can be implemented as read and write operations upon control variables. Applications will be located within the managing systems. As compared to the identification of management variables, the design of management applications may be difficult. This task may be performed during the later design cycles.

During the design of management applications, the need to add information variables becomes apparent ③. These variables should be added to the managed systems and related to the primary functions ④. Also these variables should be structured into a MIB; this time only read capabilities will be required.

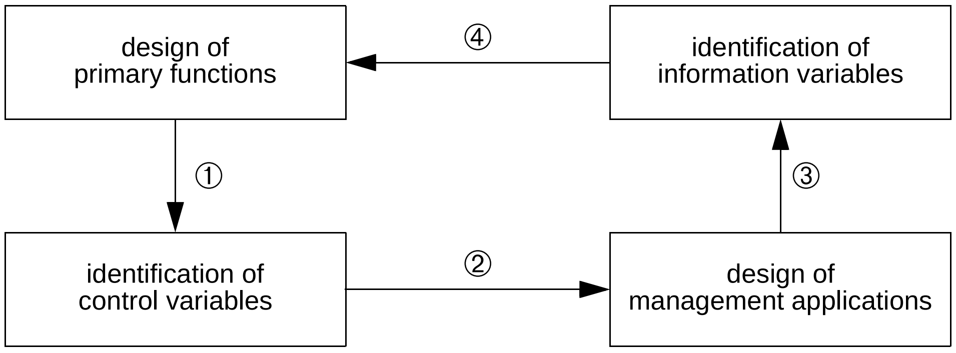

design of primary functions
④
identification of information variables
①
③
identification of control variables
②
design of management applications

Figure 9.6: Successive design steps for VO management

The communication oriented functions can be implemented as soon as a stable MISP definition becomes available. In practice, application and communication oriented functions can be implemented in parallel.

Design of management applications during the operational phase

As compared to the managing systems, there may be many managed systems. Replacing all managed systems with new generations will be expensive and should be avoided whenever possible. The designer should therefore try to define and implement all required management variables *before the start of the operational phase.*

Given the relatively small number of managing systems, it may be relatively inexpensive and therefore acceptable to modify or replace (parts of) these systems with new generations during the operational phase. As discussed in Section 6.6, the designer should anticipate this possibility and develop *management platforms* to which management applications will be added as soon as they become available. The design of management applications should thus continue *after the start of the operational phase* (Figure 9.7).

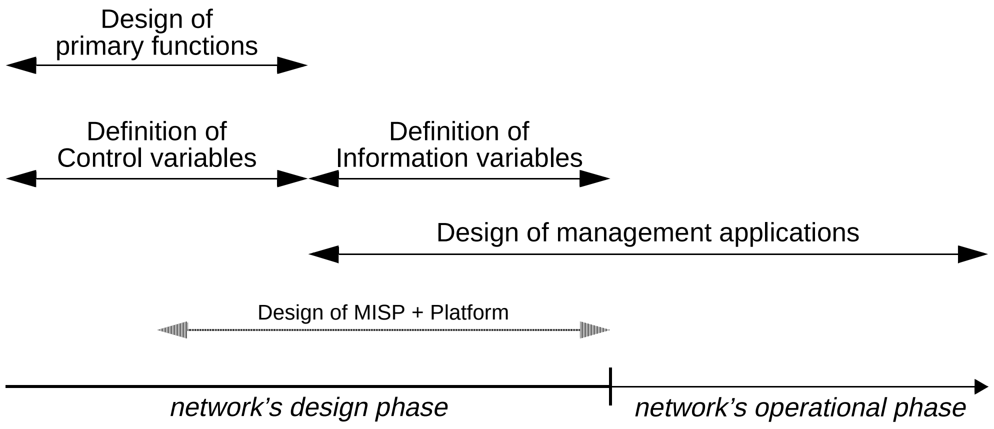

Design of primary functions
Definition of Control variables
Definition of Information variables
Design of management applications
Design of MISP + Platform
network's design phase
network's operational phase

Figure 9.7: Design of management applications during operational phase

A problem with this approach is that on the one hand all management variables should be defined *before* the start of the operational phase, but on the other hand the need to introduce information variables can only be determined as part of the design of management applications, which may be *after* the start of the operational phase.

The only way to meet this problem is to *guess* during the design phase which information variables may be needed by future management applications. Real life examples indicate that most designers remain at the safe side and introduce information variables for many possible items. As a result, a large number of information variables will be defined (as mentioned on page 162, more then 75% of all SNMP MIB variables represent information variables).

Human managers may have problems to understand the fine grained details of all these information variables. It is therefore conceivable that these managers make mistakes.

9.1.4 Strengths and weaknesses

An important characteristic of the VO approach is that it allows to commence the operational phase before the design of all management applications has been concluded. VO management is thus well suited for explicit management.

One of the characteristics of the VO approach, is that most management functions are performed by the managing systems and only a few by the managed systems. The management functions that are performed by the managed systems are usually easy to implement and have little impact upon the system's structure and price. If failures occur within managed systems:

- It is unlikely that the management functions within these systems cause the problem.
- It is likely that these management functions will continue to function well.

The VO approach may thus well be suited for fault management.

One of the weaknesses of the VO approach, is that it may be inefficient with respect to bandwidth. Other weaknesses are:

- Many system resources (e.g. CPU time and memory) may be required to implement the intelligence within the managing systems. Such systems may therefore be expensive.
- The management interactions have little expressive power.

Given the unequal distribution of management functionality and the low level of management interactions, the VO approach is better suited for centralized management than for distributed management.

A final weakness is that the VO approach, in its pure form, may be insecure. Without extra measures, unauthorized managers may be able to take over the network. To improve security, it may be necessary to include in managed systems authentication and access control functions. These functions increase the complexity of the managed systems[1] and thus the chance of errors. Besides, the authentication and access control functions must also be managed: meta-management may therefore be necessary.

9.2 The Command Oriented Approach

With the Command Oriented (CO) approach, managing systems monitor, initialize and modify the behaviour of managed systems by issuing imperative commands (such as 'reboot', 'reset' or 'tell me the error rate'). In the CO approach, managers specify the *goals* of management operations; they do not specify the means to reach these goals (as was the case in the VO approach). As a consequence, the variables that reflect and control the behaviour of the primary functions need not be visible to the managing systems.

1. SNMPv2 is an improved version of SNMP and contains authentication and access control functions. SNMPv2 is far more complex than SNMPv1.

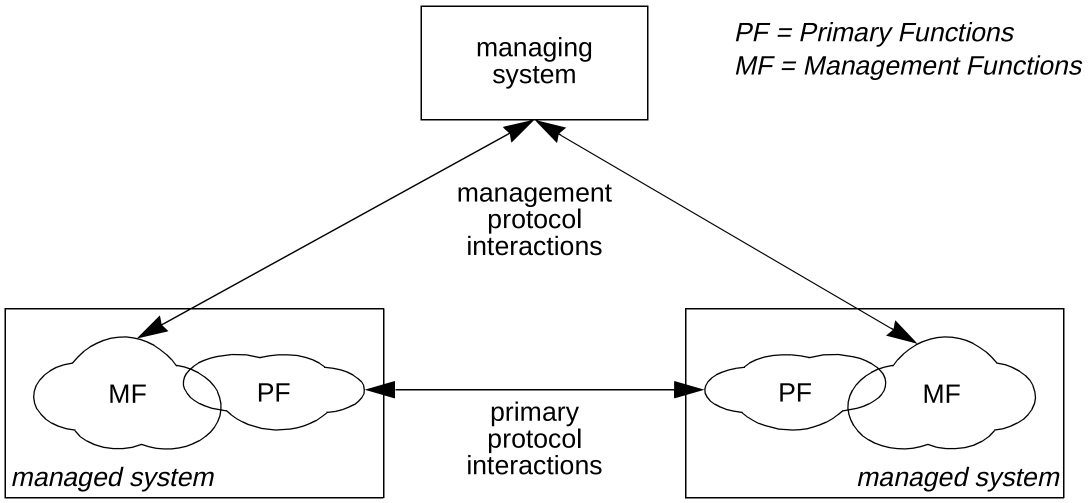

managing system
PF = Primary Functions
MF = Management Functions
management protocol interactions
MF
PF
PF
MF
primary protocol interactions
managed system
managed system

Figure 9.8: Command Oriented management

The imperative commands are defined at a high level of abstraction and have well defined semantics. Management goals may be achieved by issuing single management PDUs. CO management may thus be efficient with respect to bandwidth.

A characteristic of the CO approach, is that a large number of management PDUs may be defined. Many of these PDUs will be applicable to a single protocol layer.

9.2.1 Examples

The CO approach can be used for distributed and implicit, but also for centralized and explicit management. The routing management standards developed by ISO and the IETF (Figure 1.7) provide good examples of distributed and implicit management; token ring and DQDB management provide good examples of centralized and explicit management. Token ring management will be discussed as an example.

Token ring management

The token ring standard specifies a hierarchical form of management: a system manager on top, intermediate managers in between and the normal ring stations at the bottom. All primary functions are performed by the normal ring stations (Figure 9.9).

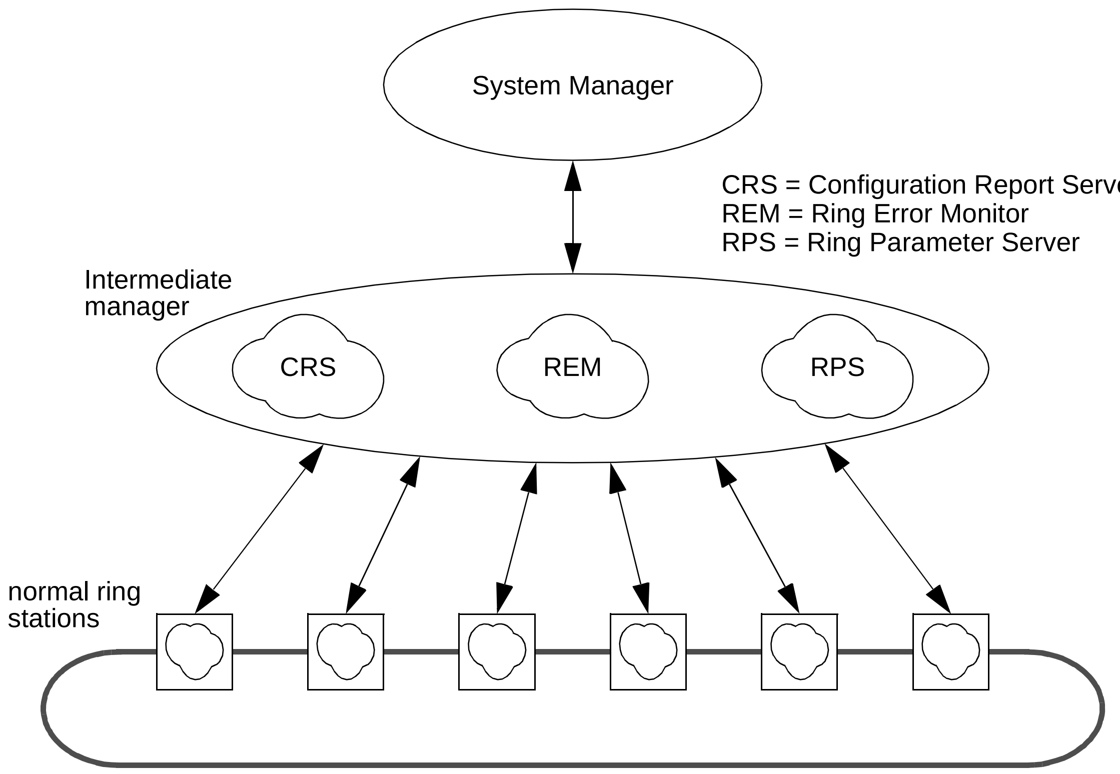


Figure 9.9: Token ring management

This example will be limited to the management interactions between the intermediate manager level and the normal ring stations. In token ring management there are three kinds of intermediate level manager functions[1]:

- the Configuration Report Server (CRS).
- the Ring Error Monitor (REM).
- the Ring Parameter Server (RPS).

The following imperative management PDUs may be exchanged between the normal ring stations and the CRS, REM and / or RPS.

name	description
Remove Ring Station	Sent by the CRS to a specific ring station causing unconditional removal
Request Ring Station Address	Issued by RPS, REM or CRS to obtain the address(es) of a specific station
Report Ring Station Address	Sent by a station as a response to the RPS, REM or CRS
Request Ring Station Attachments	Issued by RPS, REM or CRS to obtain information on the functions active in a specific station
Report Ring Station Attachments	Sent by a station as a response to the RPS, REM or CRS
Request Ring Station State	Issued by RPS, REM or CRS to obtain state information of a specific station

Figure 9.10: Management commands associated with normal ring stations

1. CRS, REM and RPS functions may be implemented as part of normal ring stations (not necessarily the same), but also as special stations (which should be connected to the ring).

name	description
Report Ring Station State	Sent by a station as a response to the RPS, REM or CRS
Request Initialization	Issued by the station that has just entered the ring. It informs the RPS that it has been inserted and wants new parameters
Initialize Ring Station	Issued by the RPS as a response to a previous Request Initialization.
Change Parameters	Sent by the CRS to set parameters
Report Error	Sent by a station to the REM
Report SUA Change	Sent by a station to the CRS when a change in station's Stored Upstream neighbour Address is detected

Figure 9.10: Management commands associated with normal ring stations

One of the normal ring stations additionally performs the so-called 'monitor' functions. Imperative management PDUs may also be exchanged between RPS, REM, CRS and the monitor.

name	description
Report Active Monitor Error	Sent by the active monitor station to the REM
Report Neighbour Notification Incomplete	Sent by the active monitor station to the REM.
Report New Active Monitor	Sent by a new active monitor to the CRS

Figure 9.11: Management commands associated with monitor

This example shows that, for the case of token ring, fifteen PDU types have been defined for management of MAC entities. It is likely that for management of a complete system (which includes seven layers) more than hundred different PDU types will be needed. Compare this to the two different PDU types that are required by the VO approach (*GET* and *SET*)!

9.2.2 Management functions

The CO approach can be characterized by the fact that, as compared to the VO approach, more management functions are performed by the managed systems and less by the managing systems. With this approach, the translation of a management request into a number of changes on variables will be performed by the managed, and not by the managing system.

> *Example:* Consider a reset. In the VO approach, the intelligence within the managing system determines which variables within the managed system should be reset. In the example of Figure 9.12 these are four variables: V1 - V4. After these variables have been determined, the manager issues four *SET* commands. In the CO approach, this intelligence will be implemented within the managed system themselves. This time only a single reset command need to be exchanged between managing and managed system.

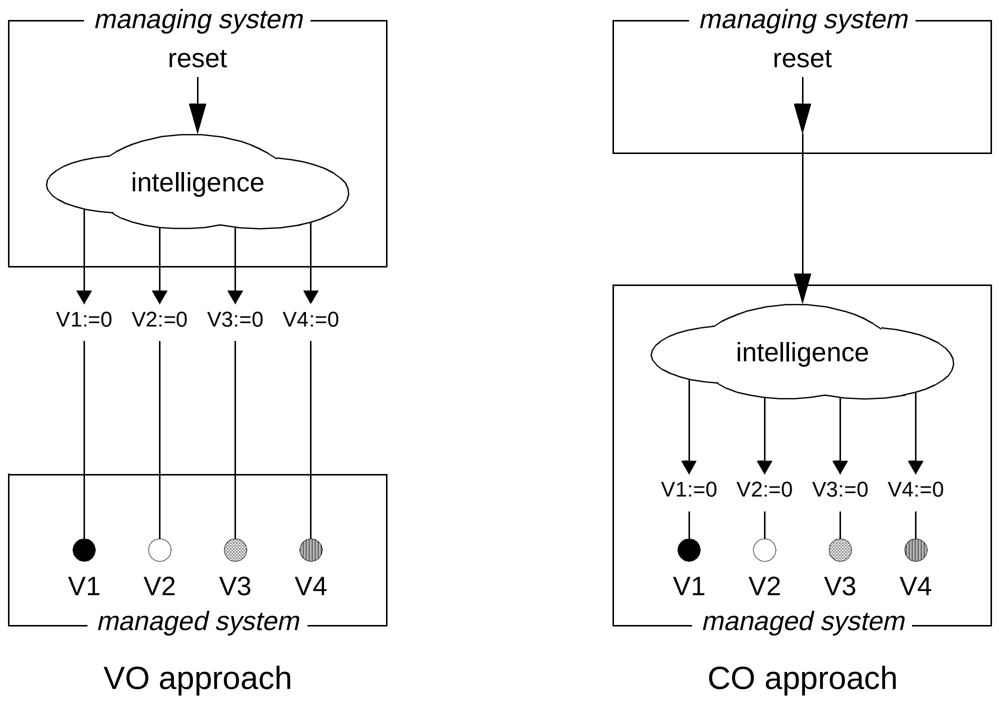


Figure 9.12: Reset according to VO and CO approach

9.2.3 Impact on the design process

Since more management functions are performed by the managed systems and less by the managing systems, the difficulty of implementing managed systems increases and the difficulty of implementing managing systems decreases. The management functions within the managed systems may even grow to a level in which these functions must themselves be managed. This requires the introduction of meta-management. Real world examples demonstrate that these meta-management functions may themselves be defined according to the VO approach.

> *Example:* The OSPF routing standard of the Internet world defines a number of CO management interactions [95]. The routing entities which exchange these interactions are relatively complex and may themselves be managed. This meta-management is based upon VO interactions. A special OSPF meta-management MIB has been defined that includes approximately hundred variables [96].

Given the large number of managed systems, it may be impractical to add new management functions to these systems after the start of the operational phase. Besides, the well defined semantics of management PDUs make it difficult for managing systems to perform management operations that were not anticipated during the design phase. As compared to the VO approach, more management problems must therefore be resolved during the design phase and there is less freedom during the operational phase.

9.2.4 Strengths and weaknesses

The CO approach can be used for distributed as well as centralized management.
A strong point of the CO approach is its power of expression; a single management interaction may be sufficient to obtain the desired effect. The CO approach is efficient with respect to bandwidth.

The fact that a relatively large number of management functions is performed by the managed systems, implies that the chance of failures within these systems should not be neglected. CO management may therefore be less suited for fault management.

As compared to the VO approach, more management problems must be resolved during the design phase; there is less freedom during the operational phase. This is a weak point of the CO approach.

9.3 The Object Oriented approach

In the previous sections two management approaches were presented: the VO and the CO approach. The intention of this section is not to develop a third approach, but to relate the Object Oriented (OO) approach of OSI to both previous approaches. It should be noted that this section interprets the OO concepts in the specific sense of OSI management, and not in a more general sense as may be found in literature.

In OSI's OO management approach, managed systems contain a number of 'managed objects'. These objects contain attributes and show some kind of behaviour.The manager sends operations to, and receives notifications from these objects (see also Subsection 2.2.1). Some operations result in changes of attribute values; other operations result in some kind of behaviour.

9.3.1 Relation with VO and CO approach

Whereas the emphasis in a VO approach is on the *information* kept within managed systems, the emphasis in a CO approach is on the *interactions* between managed and managing systems. As will be demonstrated in this section, OSI's OO approach combines characteristics of the two other approaches and can be considered as a hybrid form.

"Attributes are properties of managed objects that are expressible in terms of a value" [115]. Managers may use *M-GET* primitives to obtain attribute values, and *M-SET* primitives to modify such values.
By reading and modifying attribute values, the manager becomes able to observe and change the behaviour of primary functions. Attributes may be compared to the variables of the VO approach.

Next to the use of attributes, OO management provides a second mechanism for observing and changing the behaviour of primary functions: *M-ACTION*. "*M-ACTION* primitives are concerned with performing operations on managed objects other than those associated with the manipulation of attribute values" [115]. The *M-ACTION* primitives can be seen as the vehicle to perform imperative commands, such as defined by the CO approach.

M-GET, *M-SET* and *M-ACTION* demonstrate that the OO approach can be used in a VO as well as a CO sense; the OO approach may in fact be considered as a synthesis of both approaches. The next subsections discuss some of the most important OO concepts and identify for which goals these concept have been defined. The purpose of these subsections is to show that these goals may also be achieved with other approaches.

9.3.2 Classes and inheritance

Managed objects are structured into classes. The properties (attributes, operations, behaviour and notifications) of each object are determined by the class definition to which the object belongs. Classes may be considered as *descriptions*, while managed objects may be considered as class *instantiations*. Multiple managed objects may belong to the same class.

New class definitions can be obtained by extending existing classes with new properties. The term *specialization* is used to denote such extension. Objects that belong to a new class, show all properties of the original class plus all new properties; the new class *inherits* all properties of the existing class. The new class is a *subclass* of the existing class; the existing class is a *superclass* of the new class. It is possible that subclasses include the properties of multiple superclasses: the term *multiple inheritance* is used to denote this possibility.
As opposed to some other object oriented variants, OSI's OO approach only allows *strict inheritance*. This means that subclasses inherit *all* properties (and not just a subset) of their superclass(es).

Classes and inheritance are primarily meaningful during the design phase; they have limited use after all objects have been defined and the operational phase has been entered.

The concepts of classes and inheritance have been introduced [115]:
- To produce clear and consistent managed object definitions.
- To encourage the controlled re-use of specifications.

Although these goals are important, it should be recognized that they can also be achieved in other ways. In a VO approach these goals may, for instance, be accomplished via data types. In such approach the designer starts with the definition of basic types. These types may be included in more detailed data types, which in turn may be included in again more detailed types. This allows the construction of a hierarchy of data types; this hierarchy may be compared to the inheritance hierarchy.

ASN.1 supports, just as many programming languages, the creation of data types and the possibility to develop complex data types from simple types. Figure 9.13 shows some language constructs which can be used to create complex data types.

ASN.1	C	Pascal
sequence	struct	record

Figure 9.13: Language constructs to define complex data types

9.3.3 Containment

OSI's OO approach allows objects to contain one or more smaller objects. This principle is called 'containment'. Application of this principle is possible in a recursive fashion, which implies that also these smaller objects may contain objects. In this way it is possible to define a containment hierarchy, which may be drawn in the form of a tree[1].

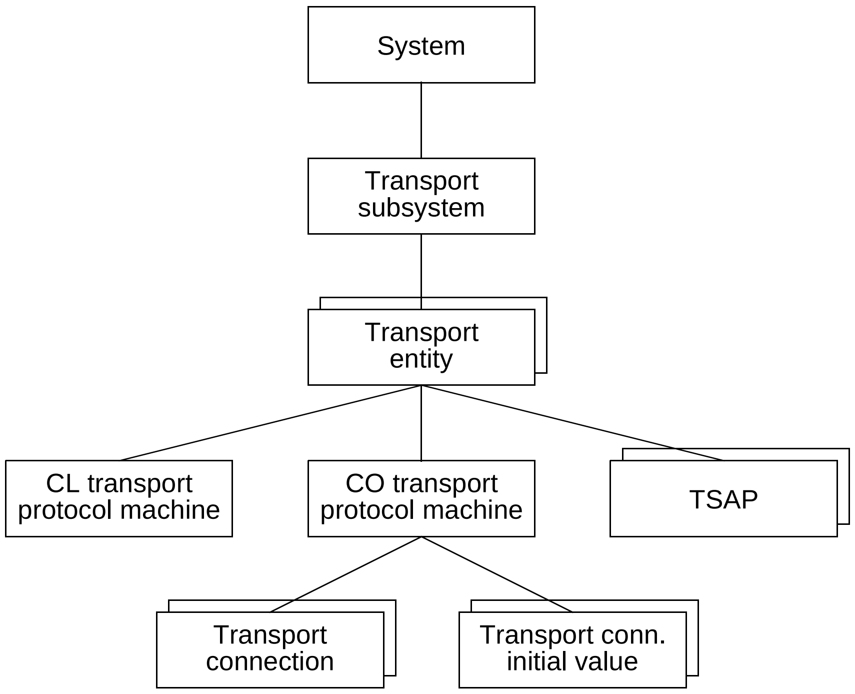

System
Transport subsystem
Transport entity
CL transport protocol machine
CO transport protocol machine
TSAP
Transport connection
Transport conn. initial value

Figure 9.14: Example of a containment tree

Example: Figure 9.14 shows the containment tree of the transport layer [56]. This tree includes seven objects that are specific for the transport layer. These objects are all contained within the general 'system' object, which also contains managed objects for other lay-

1. The containment hierarchy should not be confused with the inheritance hierarchy.

ers. The figure shows for instance that the 'transport entity' managed object contains managed objects for the:
- Connectionless transport protocol machine.
- Connection oriented transport protocol machine'.
- Transport service access points.

In OSI's OO approach containment trees are used for two purposes: naming and multiple object selection (via scoping). It should be noted however that naming and multiple object selection can also be realized in other ways. SNMP for instance demonstrates that unique names can also be assigned without introducing containment trees and the *GetBulk* operator of SNMPv2 shows that also in case of a VO approach it is possible to operate on multiple variables with a single request.

9.3.4 Encapsulation

A characteristic of the object oriented approach is that the information within each managed object is shielded from its environment (information hiding). To manipulate the information that is contained within an object, the outside world must use special management *operations* and *notifications*. This characteristic is called encapsulation and offers a number of advantages [65][110]:
- The operations and notifications may be described at a relatively high level of abstraction, which makes errors less likely and standardization easier.
- The integrity of the (information within the) object can be guaranteed. It is for instance possible to protect the information from unauthorized access and to enforce the correct sequence of management operations.

Encapsulation is not specific for the OO approach, but can also be found elsewhere:
- A service defines the external behaviour of a distributed system. A service does not unveil a distributed system's internal structure to its environment (the users). The internal operation is thus encapsulated.
- A protocol defines the external behaviour of a (sub)system. The internal structure of the (sub)system is not visible to the environment; the internal operation is thus encapsulated.

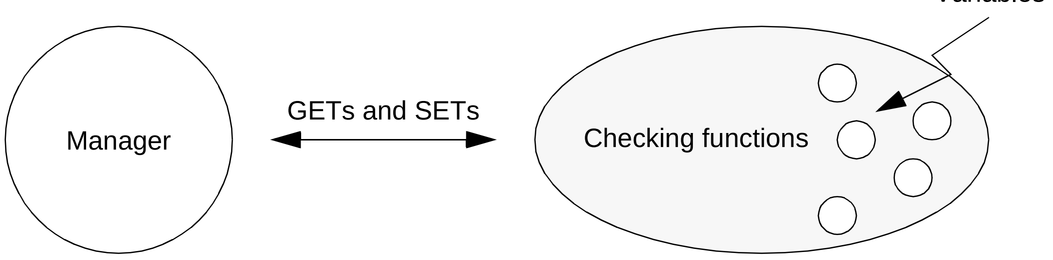


Figure 9.15: Checking functions to 'encapsulate' variables

In its pure form, VO management can not prevent unauthorized access (page 164) nor can it avoid sequencing problems. In practice, designers may therefore decide to introduce special functions within the managed systems that perform some initial checks before they allow the *GET* and *SET* commands to proceed. These special functions are needed to avoid the problems as mentioned above and may be seen as some kind of ‘shell’ surrounding (or encapsulating) the variables (Figure 9.15).

To demonstrate this, some examples from Internet management are given:
- With SNMP, managed systems include special functionality to check whether it is possible to perform all *SET* requests that are contained in a single management PDU as an atomic action. If one of these requests can not be honoured, the other requests of the management PDU will not be performed too.
- With SNMPv2, extra functionality has been included to ensure the origin of the management request and to restrict variable access to certain managers (Subsection 4.2.2).
- With SNMPv2, special functionality has been included to check whether a PDU that contains a command to reset the ‘authentication clock’ also contains a command to change the authentication key[1]. A request to reset the authentication clock will not be honoured if there is no associated request to change the authentication key.

9.3.5 Allomorphism

The final OO principle that will be discussed, is allomorphism. "Allomorphism is the ability of a system containing a managed object to act as if the managed object were a member of another managed object class" [115]. To realize allomorphism, it may be necessary to add a special parameter to the management interaction. In case of OSI, this parameter is called the ‘object class’.

Allomorphism may be useful to realize the following requirements[65]:
- It should be possible to update a MIB [110].
- It should be possible to extend a MIB with proprietary features.
- It should be possible to customize MIB views for individual managers.

It should be noted that allomorphism is just one mechanism to realize these requirements and that the VO approach allows the use of similar mechanisms to realize these requirements. With SNMPv2 it is for instance possible to satisfy these requirements via:
- The introduction of new variables.
- The use of the ‘context local entity’ parameter.
- The use of the ‘view’ mechanism.

1. ‘Authentication clocks’ are used to detect whether messages have been captured from the network and resent at a later time by unauthorised users (message replay).

Conclusions

This thesis motivates that in principle there is no difference between the design of primary network functions and the design of network management functions. Both kinds of functions are strongly interwoven and can be described in terms of a single set of architectural concepts and rules; an integrated architectural model can be developed which includes both kind of functions. Such model shows the relationship between primary and management functions and may be used to construct management simulators (Chapter 7).

The management architectures of the ISO, ITU-T and IETF do not describe management functions together with primary functions and do not provide integrated architectural models.

ISO’s management architecture violates the idea of layering, by allowing management entities in the application layer to directly manipulate objects in the underlying service provider. Another deficiency of this architecture is that fault management may be problematic because the fault management functions implicitly rely for the exchange of their management information upon the correct behaviour of the managed functions (Chapter 2).

In TMN such implicit dependence between managed and management functions does not exist; to exchange management information TMN has defined a separate Data Communication Network (DCN). A deficiency of TMN is that its architectural concepts are not always properly defined (Chapter 3).

The IETF has not produced a separate standard to define its Internet management architecture. Instead, the IETF has concentrated upon the development of management protocols and MIBs. Many MIBs have already been defined, unfortunately no adequate structure has been proposed to organize the variables contained within these MIBs (Chapter 4).

This thesis demonstrates that primary functions can be developed together with management functions in a cyclic fashion. The first cycle(s) will generally be used to design the primary functions and identify which parts of these functions should be managed. To define the details of the management functions, subsequent cycles will be needed. The complexity of the management functions may be such, that also these management functions should be managed. The designer may perform a number of additional design cycles to develop these ‘management of management’ (meta-management) functions (Chapter 6).

Management functions can be structured by distinguishing the functions defined in one phase of the design process from functions defined in other phases. The following structure is proposed:

- Service management functions, for management functions introduced in the architectural phase.
- Protocol management functions, for management functions introduced in the implementation phase.
- Element management functions, for management functions introduced in the realization phase.

As opposed to the service and protocol management functions, element management functions are linked to practical realizations and manufacturer dependent. As a consequence, element management functions will not be standardized (Chapter 5).

Two basic types of management protocols can be identified: Variable Oriented (VO) and Command Oriented (CO). With VO management, the emphasis is on the management *information* within the managed systems. With CO management, the emphasis is on the management *interaction* between the managing and managed systems. VO management is better suited for centralized and explicit management, whereas CO management is better suited for distributed and implicit management. OSI's Object Oriented (OO) management protocol can be regarded as a mixture of VO and CO management (Chapter 9).

Further Research

It may be interesting to map the SNMPv2 protocol upon the architecture that has been described in this thesis. The purpose of such exercise should be to get a better understanding of SNMPv2. As a result, proposals can be presented to improve this protocol.

Practical experience is needed to understand the applicability of the service management, protocol management and element management concept. Special MIBs should therefore be developed, as well as special management applications that take advantage of these MIBs.

It seems worthwhile to build a management simulator (Subsection 7.4.4). The effort to obtain a realistic prototype should not be neglected however, since such simulator requires as input formal descriptions of both primary as well as management functions.

References

[1] Aidarous S.E., Proudfoot D.A., Dam X.: "Service Management in Intelligent Networks", IEEE Network Magazine, January 1990

[2] Autrata M., Strutt C.: "DME Framework and Design", in: Network and Distributed Systems Management, Chapter 23, Addison-Wesley Publishing Company, 1994

[3] Ben-Artzi A., Chandna A.,Warrier U., "Network management of TCP/IP Networks: Present and Future", in: IEEE Network Magazine, page 35-43, July 1990

[4] Black U.D.: "Network Management Standards - The OSI, SNMP and CMOL Protocols", McGraw-Hill Series on Computer Communications, 1992

[5] Blaauw G.A., Brooks F.P.: "Computer Architecture", lecture notes (draft), University of Twente, Department of Computer Science, Enschede, The Netherlands, August 1985

[6] Bogaards K., Pires L., Pras A., Schot J.: "The Pangloss method", Proceedings of the Esprit Conference, Elsevier, 1988

[7] Bogaards K.: "A Methodology for the Architectural Design of Open Distributed Systems", Ph.D. thesis, University of Twente, 1990, ISBN 90-9003554-0

[8] Bootman S., Shabana M.: "Generic building blocks for the Telecommunications Management network", Globecom - IEEE Global Telecommunications Conference & Exhibition, 1988, Conference record Volume 1, page 163-167

[9] Boyd R.T., Brodrick K.J.: "Operational Support Systems for the future Local Network", BT Technology Journal, Vol. 7, No. 2, April 1989, page 136-150

[10] Brooks Jr. F.P.: "The mythical man-month", Addison-Wesley publishing company, 1975

[11] Brugnoni S., Bruno G., Manione R., Montariolo E., Paschetta E., Sisto L.: "An Expert System for Real Time Fault Diagnosis of the Italian Telecommunications Network", in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 617-628, North-Holland, 1993

[12] Carl-Mitchell S., Quarterman J.S.: "Building Internet Firewalls", Unixworld, Febr. 1992, page 92-102

[13] Case J.D., Davin J.R., Fedor M.S., Schoffstall M.L.: "Introduction to the Simple Gateway Monitoring Protocol", in: IEEE Network, page 43-49, March 1988

[14] Case J.D., Davin J.R., Fedor M.S., Schoffstall M.L.: "Network management and the design of SNMP", in: ConneXions, the Interoperability Report, page 22-26, March 1989

[15] Case J.D., Davin J.R., Fedor M.S., Schoffstall M.L.: "Internet network management using the Simple Network Management protocol", in Proceedings of the 14th IEEE Conference on Local Computer Networks, page 156-159, Oct. 1989

[16] Case J.D., Davin J.R., Fedor M.S., Schoffstall M.L.: "Keeping it simple (network management)", in Unix Review, Vol. 8, No. 3, page 60-66, March 1990

[17] Cassel L.N., Partridge C., Westcott J.: "Network Management Architectures and protocols: Problems and Approaches", IEEE Journal on Selected Areas in Communications, Vol. 7, No. 7, page 1104-1114, September 1989

[18] CCITT: "Recommendation E.410 - Telephone Network and ISDN - Quality of Service, Network Management and Traffic Engineering - International Network Management - General Information", Geneva 1992

[19] CCITT Blue Book: "Recommendation M.30, Principles for a Telecommunications Management Network", Volume IV - Fascicle IV.1, Geneva 1989

[20] CCITT: "Recommendation M.3010, Principles for a Telecommunications Management Network", Geneva 1992

[21] CCITT COM IV-42-E, Question 23/IV: "Draft Recommendation M.30 - Version R1", November 1990

[22] CCITT COM IV-61-E, Question 23/IV: "Draft Recommendation M.30 - Version R4", August 1991

[23] Cerf V.: "Network management comes of age in the Internet", in: ConneXions, the Interoperability Report, page 2, March 1989

[24] Cheswick W.R.: "Firewalls and Internet Security", Addison-Wesley, 1994

[25] CMIP Run!; "New APIs for Management Data", Vol 3, No 2, 2nd Q '94, page 11

[26] Comer D.E., Stevens D.L.: "Internetworking with TCP/IP - Volume 2: Design, Implementation and Internals", Prentice Hall International Editions, 1991

[27] Data Communications: "The Price of Failure", 21 June 1991, page 11

[28] Eertink H., Wolz D: "Symbolic Execution of LOTOS Specifications", in: Proceedings of the 5th International Conference on Formal Description Techniques, FORTE '92, ed. Michel Diaz, R. Groz.

[29] Eldering A.M.: "Realizing Management Transactions", Msc-thesis University of Twente, Enschede, The Netherlands, 1994

[30] Elmer-DeWitt P.: "Ghost in the machine", in: Time, 29 January 1990

[31] Embry J., Manson P., Milham D., "An Open Network Management Architecture: OSI/NM Forum Architecture and Concepts", in: IEEE Network Magazine, page 14-22, July 1990

[32] Embry J., Manson P., Milham D., "Interoperable Network Management: OSI/NM Forum Architecture and Concepts", in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 29-44, North-Holland, 1991

[33] Fernandes F.P.L.B., Monteiro E.H.S., "An Application Process Framework for Advanced Communications Management", in: Proceedings of the fifth RACE TMN Conference, 1.3/2 page 1-15

[34] Ferreira Pires L.: "The Lotosphere Design Methodology: Basic Concepts", Final deliverable, LOTOSPHERE project, ESPRIT Ref: 2304, March 1992

[35] Frontini M., Griffin J., Towers S.: "A Knowledge-Based System for Fault Localisation in Wide Area Networks", in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 519-530, North-Holland, 1991

[36] Geihs K., Mann A.: "ODP Viewpoints of IBCN Service Management", Technical Report N0. 43.9104, IBM Deutschland GMBH, Heidelberg

[37] Gering M.: "CMIP versus SNMP", in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 347-359, North-Holland, 1993

[38] Goldman J., Hong P., Jeromnimon C., Louit G., Min J., Sen P.: "Integrated Fault Management in Interconnected Networks", in: Proceedings of the IFIP TC6/WG 6.6 First International Symposium on Integrated Network Management, page 333-344, North-Holland, 1989

[39] Goodman R.M., Ambrose B.: "A Hybrid Expert System / Neural Network Traffic Advice System", in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 607-616, North-Holland, 1993

[40] Goyal S.K.: "Knowledge Technologies for Evolving Networks", in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 439-461, North-Holland, 1991

[41] Graham J.: "The Penguin Dictionary of telecommunications", Penguin books, 1985

[42] Halsall F., Modiri N.,An Implementation of an OSI Network Management System", in: IEEE Network Magazine, page 44-53, July 1990

[43] ISO 7498: "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", Geneva, 1984

[44] ISO 7498-4: "Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework", Geneva, 1989

[45] ISO 8073: "Information Processing Systems - Open Systems Interconnection - Connection oriented transport protocol specification", Geneva, 1992

[46] ISO 8473: "Information Processing Systems - Data Communications - Protocol for providing the Connectionless-mode Network Service", Geneva, 1988

[47] ISO 8802/5: "Information Processing Systems - Local Area Networks - Token Ring Access Method and Physical Layer Specification", Geneva, 1987

[48] ISO 9542: "Information Processing Systems - Data Communications - End System to Intermediate System Routeing Information Exchange Protocol for use in conjunction with the Protocol for the Provision of the Connectionless-mode Network Service", Geneva, 1988

[49] ISO TR 9575: "Information Processing Systems - Data Communications - OSI Routeing Framework", Geneva, 1990

[50] ISO 9595: "Information Processing Systems - Open Systems Interconnection - Common Management Information Service Definition", Geneva, 1990

[51] ISO 9596: "Information Processing Systems - Open Systems Interconnection - Common Management Information Protocol", Geneva, 1991

[52] ISO 10038: "Information Processing Systems - Data Communications - MAC Bridges", Geneva, 1993

[53] ISO 10040: "Information Processing Systems - Open Systems Interconnection - Systems Management Overview", Geneva, 1992

[54] ISO DIS 10165-1: "Information Processing Systems - Open Systems Interconnection - Structure of Management Information - Part 1: Management Information Model", Geneva, 1993

[55] ISO 10589: "Information Processing Systems - Data Communications - Intermediate to Intermediate System Routing Information Exchange Protocol for use in Conjunction with ISO 8473", Geneva, 1992

[56] ISO 10737: "Information Processing Systems - Open Systems Interconnection - Specification of the elements of Management Information relating to OSI Transport layer Standards", Geneva

[57] ISO 10747: "Information Processing Systems - Data Communications - Protocol for Exchange of Inter-Domain Routing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs", Geneva

[58] ISO/TC 97/SC 16 N1719: "OSI Management Framework - Fourth Working Draft", October 1983

[59] ISO/TC 97/SC 21 N382: "Procedures for Management Information Service Standardization", February 1985

[60] ISO/TC 97/SC 21 N975: "OSI Management Framework - Seventh Working Draft", November 1985

[61] ISO/TC 97/SC21 N1388: "Proposal for further consideration of the OSI Management Architecture", Egham, September 1986

[62] ISO/TC 97/SC21 N3003: "Summary of Voting on DIS 7498-4, Information Processing Systems - Open Systems Interconnections - Basic Reference Model - Part 4: Management Framework", August 1988

[63] Jander M.: "SNMP: coming soon to a network near you", Data Communications, November 1992, page 66-76

[64] Jordaan J.F., Paterok M.E.: "Event Correlation in Heterogeneous Networks using the OSI Management Framework", in: Proceedings of the IFIP TC6/WG

6.6 Third International Symposium on Integrated Network Management, page 683-695, North-Holland, 1993

[65] Klerer S.M.: "System Management Information Modelling", in: IEEE Communications Magazine, page 38-44, May 1993

[66] Kobayashi Y., "Standardization Issues in Integrated Network Management", in: Proceedings of the IFIP TC6/WG 6.6 Symposium on Integrated Network Management, page 79-90, North-Holland, 1989

[67] Lewis L.: "A Case-based Reasoning Approach to the Resolution of Faults in CommunicationS Networks", in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 671-682, North-Holland, 1993

[68] Lor K. E.: "A Network Diagnostic Expert System for Acculink Multiplexers Based on General Network Diagnostic Scheme", in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 659-669, North-Holland, 1993

[69] Masahiko Matsushita: "Telecommunication Management Network", NTT Review, Vol. 3 No. 4, July 1991, page 117 - 122

[70] McCloghrie K., Rose M.T.: "Network management of TCP/IP based internets", in: ConneXions, the Interoperability Report, page 3-9, March 1989

[71] Milham D.J., Willetts K.J.: "BT's Communications Management Architecture", in: Proceedings of the IFIP TC6/WG 6.6 Symposium on Integrated Network Management, page 109-116, North-Holland, 1989

[72] Murrill B.:"OMNIPoint: An Implementation Guide to Integrated Networked Information Systems Management", in: Proceedings of the IFIP TC6/WG 6.6 Third International Symposium on Integrated Network Management, page 405-418, North-Holland, 1993

[73] Nakakawaji T., Katsuyama K., Miyauchi N., Mizuno T.: "MINT: an OSI Management Information Support Tool", in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 845-855, North-Holland, 1991

[74] Network Management Forum: "Discovering Omnipoint - a common approach to the Integrated Management of Networked Information Systems", 1993

[75] Pras A.: "Network Management: an Alternative View", in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 109-117, North-Holland, 1991

[76] Pras A.: "An Architecture for a Distributed IS-PABX", in Local Communication Systems: LAN and PBX II, Proceedings of the IFIP TC6/WG6.4 Second Int. Conf. on Local Communication Systems, Palma, Spain, June 1991

[77] Pras A.: "Netwerklaag-routering", in Handboek Telematica, Samson, the Netherlands, June 1992

[78] Pras A.: "TMN - Introduction and Interpretation", TIOS 92-16, Memoranda Informatica 92-40, University of Twente, The Netherlands, 1992

[79] Pras A., Togtema J.: "SNMPv2 at Twente University", The Simple Times, February 1994

[80] RACE CFS A150: "General Terminology", Common Functional Specifications, document 11, Issue B, December 1991

[81] RACE CFS H400-H411: "Network Management - Services", Common Functional Specifications, document 7, Issue B, December 1991

[82] RACE CFS H404: "Provisioning services in TMN", Common Functional Specifications, document 7, Issue B, December 1991

[83] RFC 768: "User Datagram Protocol", Postel, J.B., August 1980

[84] RFC 793: "Transmission Control Protocol", Postel J.B., 1981

[85] RFC 903: "Reverse Address Resolution Protocol", Finlayson R., Mann T., Mogul J.C., Theimer M., June 1984

[86] RFC 1021: "High-level Entity Management System (HEMS)", Partridge C., Trewitt G., October 1987

[87] RFC 1022: "High-level Entity Management Protocol (HEMP)", Partridge C., Trewitt G., October 1987

[88] RFC 1028: "Simple Gateway Monitoring Protocol", Davin J., Case J.D., Fedor M., Schoffstall M.L., November 1987

[89] RFC 1052: "IAB recommendations for the development of Internet network management standards", Cerf V.G., April 1988

[90] RFC 1155: "Structure and identification of Management Information for TCP/IP-based internets", Rose M.T.; McCloghrie K., May 1990

[91] RFC 1156: "Management Information Base for network management of TCP/IP-based internets", McCloghrie K.; Rose M.T., May 1990

[92] RFC 1157: "Simple Network Management Protocol (SNMP)", Case J.D., Fedor M., Schoffstall M.L., Davin C., May 1990

[93] RFC 1189: "Common Management Information Services and Protocols for the Internet (CMOT and CMIP)", Warrier U.S., Besaw L., LaBarre L., Handspicker B.D., October 1990

[94] RFC 1213: "Management Information Base for network management of TCP/IP-based internets: MIB-II", McCloghrie K., Rose M.T., March 1991

[95] RFC 1247: "OSPF version 2", Moy J., July 1991

[96] RFC 1253: "OSPF version 2: Management Information Base", Baker F., Coltun R., August 1991

[97] RFC 1441: "Introduction to version 2 of the Internet-standard Network Management Framework", Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[98] RFC 1447: "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)", McCloghrie K., Galvin J., April 1993

[99] RFC 1450: "Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)", Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[100] RFC 1451: "Manager-to-Manager Management Information Base", Case J., McCloghrie K., Rose M., Waldbuster S., April 1993

[101] Rose M.T.: "The Simple Book - Second edition", Prentice-Hall International Editions, 1994

[102] Rose M.T., "Network management is Simple, you just need the right framework", in: Proceedings of the IFIP TC6/WG 6.6 Second International Symposium on Integrated Network Management, page 9-25, North-Holland, 1991

[103] Saal H.: "LAN Downtime: Clear and Present Danger", in Data Communications, 21 March 1991

[104] Schot J.: "The role of Architectural Semantics in the formal approach of Distributed Systems Design", Ph.D. thesis, University of Twente, 1990, ISBN 90-9004877-4

[105] Sinderen M.J. van: "On the design of application protocols", Ph.D. thesis, University of Twente, 1995, ISBN 90-365-0730-8

[106] Skubic J., Burwall H., "Service Management Architecture", in: Proceedings of the XIII International Switching Symposium, Vol. VI: page 155-160, May 1990

[107] Sloman M.S.: "Domain Management for Distributed Systems", in: Proceedings of the IFIP TC6/WG 6.6 Symposium on Integrated Network Management, page 505-516, North-Holland, 1989

[108] Sloman M., Twiddle K.: "Domains - A Framework for Structuring Management Policy", in: Network and Distributed Systems Management, page 433-453, Addison-Wesley Publishing Company, 1994

[109] Smith C., Milham D.J., Mulcahy C., "OSI Systems Management", in: BT Technology Journal, page 27-38, April 1990

[110] Stallings W.: "SNMP, SNMPv2 and CMIP - The Practical Guide to Network Management Standards", Addison Wesley

[111] Steege J.W. ter: "Alternatives for the OSI Systems Management Framework", Msc-thesis University of Twente, Enschede, The Netherlands, 1991

[112] Tucker J.: "A Common Approach to Managed Objects", in: Proceedings of the IFIP TC6/WG 6.6 Symposium on Integrated Network Management, page 159-165, North-Holland, 1989

[113] Vissers C.A., Ferreira Pires L., Quartel D.A.C.: "The Design of Telematics Systems", lecture notes, University of Twente, Department of Computer Science, Enschede, The Netherlands, October 1994

[114] Warrier U.S., Sunshine C.A., "A Platform for Heterogeneous Interconnection Network Management", in: Proceedings of the IFIP TC6/WG 6.6 Symposium on Integrated Network Management, page 13-24, North-Holland, 1989

[115] Westgate J.: "Technical Guide for OSI Management", NCC Blackell, 1992

[116] Worp S.B. van der: "Distributed Transactions in OSI Management", Msc-thesis University of Twente, Enschede, The Netherlands, 1993

[117] Yoshida M., Kobayashi M., Yamaguchi H., "Customer Control of Network Management from the Service Provider’s Perspective", in: IEEE Communications Magazine, page 35-40, March 1990

Glossary

The terminology that is used within the network management community is poorly defined. This easily leads to misinterpretations.

This glossary includes a number of terms that have the potential to be misunderstood. For each of these terms the interpretation is given that is valid for this thesis. A number of terms are used in the sense of OSI: these terms are marked with ^{OSI} and enclosed between double quotes ("...").

Architecture

The complete set of *architectural concepts*, the rules to combine these concepts plus possibly *architectural models*.
The term architecture is also used in a more restricted sense to denote the functional structure of a distributed system as it appears to its users. In this sense the term architecture denotes the outcome of the architectural phase and is used in contrast to the terms *implementation* and *realization*.

Architectural concepts

The building blocks that are used to create architectural models. Examples are: entity, PDU, protocol, SAP, service, SP, system.

Architectural model

A specific structure of architectural concepts. An example is the OSI Reference Model.

Distributed system

The complete communication network, including all parts that perform communication functions.

Entity ^{OSI}

"An active element within a subsystem".
A single entity may be engaged in several *functions*.

Framework

The same as the first interpretation of *architecture*: the complete set of *architectural concepts*, the rules to combine these concepts plus possibly *architectural models*.

Function ^{OSI}

"A part of the activity of entities".

Implementation

The outcome of the implementation phase. An implementation unveils the internal structure of the network in terms of *(sub)systems* and protocols. The term implementation is used in contrast to the terms *architecture* and *realization*.

Primary functions
The term ‘primary’ functions is introduced in this thesis in contrast to ‘management’ functions. Primary functions may be considered as the ‘normal’ network functions that satisfy the primary user requirements. Management functions are the functions that ‘support’ and ‘control’ the primary functions.

Protocol OSI
"A set of rules and formats (semantic and syntactic) which determines the communication behaviour of *entities* in the performance of *functions*".

Realization
The outcome of the realization phase. A realization shows the internal structure of a single *system*. The term realization is used in contrast to the terms *architecture* and *implementation*.

Reference model
See *architectural model*.

Service OSI
"A capability of a layer and the layers beneath it, which is provided to entities at the next higher layer at the boundary between these layers."

Subsystem OSI
"An element in a hierarchical division of an open system which interacts directly only with elements in the next higher or the next lower division of that open system."
The intersection of a layer and system.

System OSI
A single node within the network. Examples are: routers and terminals.

Index

Abbreviations

ACL	- Access Control List
AFNOR	- Association Française de Normalisation
API	- Application Programming Interface
ASE	- Application Service Element
AT&T	- American Telegraph and Telephone company
ATM	- Asynchronous Transfer Mode
BER	- Basic Encoding Rules
BGP	- Border Gateway Protocol
BT	- British Telecom
CCITT	- Comité Consultative Internationale de Telegraphique et Telephonique
CFS	- Common Functional Specification
CLNP	- ConnectionLess Network Protocol
CMIP	- Common Management Information Protocol
CMIS	- Common Management Information Service
CMOL	- Common Management Over LLC
CMOT	- Common Management Over TCP
CO	- Command Oriented
CORBA	- Common Object Request Broker Architecture
CPU	- Central Processing Unit
CRC	- Cyclic Redundancy Check
CRS	- Configuration Report Server
DCF	- Data Communication Function
DCN	- Data Communication Network
DIS	- Draft International Standard
DME	- Distributed Management Environment
DP	- Draft Proposal
DQDB	- Distributed Queue Dual Bus
EGP	- Exterior Gateway Protocol
ES	- End System
ETSI	- European Telecommunications Standards Institute
FCAPS	- Fault, Configuration, Accounting, Performance and Security
FDDI	- Fibre Distributed Data Interface
FTAM	- File Transfer And Management
IAB	- Internet Activities Board (until 1993 Internet Architecture Board)
IBC	- Integrated Broadband Communication
IBCN	- Integrated Broadband Communication Network
ICMP	- Internet Control Message Protocol
IEC	- International Electrotechnical Commission
IEEE	- Institute of Electrical and Electronics Engineers
IETF	- Internet Engineering Task Force
ILM	- Intermediate Level Manager
IN	- Intelligent Network
I/O	- Input / Output
IP	- Internet Protocol
IPC	- Inter Process Communication
IPX	- Internetwork Packet Exchange
ISDN	- Integrated Services Digital Network

ISO	- International Organization for Standardization
ITU	- International Telecommunication Union
ITU-T	- ITU Telecommunication Standardization Sector
JTC	- Joint Technical Committee
LAN	- Local Area Network
LAPB	- Link Access Protocol Balanced
LLA	- Logical Layered Architecture
LLC	- Logical Link Control
LM	- Layer Manager
MCF	- Message Communication Function
MD	- Mediation Device
ME	- Managed Element
MF	- Mediation Functions
MIB	- Management Information Base
MIS	- Management Information Service
MISP	- Management Information Service Provider
NAU	- Network Attachment Unit
NE	- Network Element
NEF	- Network Element Function
NFS	- Network File System
NM Forum	- Network Management Forum
NNI	- Nederlands Normalisatie Instituut
OMG	- Object Management Group
OMNI*Point*	- Open Management Interoperability Point
ONA	- Open Network Architecture
OO	- Object Oriented
OS	- Operations System
OSF	- Open Software Foundation
OSF	- Operations System Function
OSI	- Open Systems Interconnection
OSPF	- Open Shortest Path First
PC	- Personal Computer
PDU	- Protocol Data Unit
PF	- Presentation Function
PSP	- Primary Service Provider
QA	- Q Adaptor
QAF	- Q Adaptor Functions
QoS	- Quality of Service
RACE	- Research and development in Advanced Communications technologies in Europe
RARP	- Reverse Address Resolution Protocol
REM	- Ring Error Monitor
RFC	- Request For Comment
RIP	- Routing Information Protocol
RM	- Reference Model
RPS	- Ring Parameter Server
SAP	- Service Access Point
SC	- Sub-Committee
SG	- Study Group
SGMP	- Simple Gateway Monitoring Protocol

SIG	- Special Interest Group
SMAE	- Systems Management Application Entity
SMDS	- Switched Multi-megabit Data Services
SMF	- Systems Management Functions
SMI	- Structure of Management Information
SMO	- Systems Management Overview
SNA	- Systems Network Architecture
SNMP	- Simple Network Management Protocol
SONET	- Synchronous Optical Network
SP	- Service Primitive
TCP	- Transmission Control Protocol
TG	- Task Group
TLM	- Top Level Manager
TMN	- Telecommunications Management Network
TP	- Transaction Processing
TR	- Technical Report
TSAP	- Transport Service Access Point
UDP	- User Datagram Protocol
VAS	- Value Added Service
VO	- Variable Oriented
WG	- Working Group
WS	- Work Station
WSF	- Work Station Functions
XMP	- X/Open Management Protocol
XOM	- X/Open OSI-Abstract-Data Manipulation

Samenvatting

Netwerkmanagement heeft tot taak de werking van communicatienetwerken te controleren en te verbeteren, alsmede het netwerk aan te passen in het geval de eisen van de netwerkgebruikers veranderen. Management betreft het initialiseren, observeren en modificeren van de netwerkfuncties. Voor het verrichten van management zijn speciale functies nodig, die in dit proefschrift ‘managementfuncties’ worden genoemd. Om managementfuncties te kunnen onderscheiden van de normale netwerkfuncties die de primaire gebruikerseisen realiseren, wordt in dit proefschrift eveneens de term ‘primaire functies’ geïntroduceerd.
Managementfuncties kunnen op een handmatige wijze worden uitgevoerd door personen (dit noemen we ‘expliciet management’), maar ook op een geautomatiseerde wijze via hard- en software modules (‘impliciet management’). In het geval managementfuncties handmatig worden uitgevoerd, zal het merendeel van deze functies worden verricht vanaf speciale managementsystemen die zich bevinden op een beperkt aantal lokaties. Indien managementfuncties automatisch worden uitgevoerd, kan het beter zijn een groot deel van deze functies over het hele netwerk te distribueren en te implementeren als onderdeel van de gemanagede systemen.

Architecturen voor netwerkmanagement bieden de ontwerper de mogelijkheid managementfuncties op een hoog abstractieniveau te beschouwen en een goed beeld te ontwikkelen van de te ontwerpen managementservices en -protocollen. In het kader van dit proefschrift wordt aangenomen dat dergelijke architecturen uit de volgende componenten bestaan:

- een verzameling architecturele concepten,
- regels die aangeven hoe deze concepten gebruikt moeten worden, en
- modellen die de toepassing van deze regels en concepten demonstreren en hierdoor het ontwerp van een specifieke klasse van systemen vereenvoudigen.

Alle bestaande managementarchitecturen, dus ook die van de ISO, ITU-T (voorheen CCITT) en de IETF, zijn ontwikkeld nadat het ontwerp van de primaire functies was afgerond. Een dergelijke aanpak getuigt van een specifieke zienswijze betreffende de rol van management en nodigt uit tot het toepassen van verschillende architecturele concepten voor het ontwerp van primaire functies enerzijds en het ontwerp van managementfuncties anderzijds. In dit proefschrift wordt een alternatieve aanpak voorgesteld, waarin geen principieel verschil wordt gemaakt tussen de ontwerpeisen voor primaire functies en de ontwerpeisen voor managementfuncties. Beide soorten eisen worden geïntegreerd in één enkel ontwerpproces dat gebruik maakt van één architectuurmodel.

Dit proefschrift bestaat uit twee gedeelten. In deel 1 (hoofdstuk 2 - 4) wordt de ‘state of the art’ van de drie belangrijkste netwerkmanagementarchitecturen

besproken. Deel II (hoofdstuk 5 - 9) laat zien hoe een alternatieve (geïntegreerde) netwerkmanagementarchitectuur ontwikkeld kan worden.

Hoofdstuk 2 analyseert de ISO-managementarchitectuur, zoals deze is vastgelegd in het 'OSI Management Framework' en de 'Systems Management Overview'. Het toont aan dat er, ondanks het vele onderzoek dat op het gebied van ISO-management is verricht, een aantal tekortkomingen van deze managementarchitectuur nog steeds niet is opgelost.

De managementarchitectuur van de ITU-T staat bekend als het 'Telecommunications Management Network' (TMN) en wordt in hoofdstuk 3 besproken. De naam van deze architectuur geeft reeds aan dat deze architectuur primair bedoeld is voor management van telecommunicatie- (b.v. telefonie) netwerken. In feite beschrijft TMN meerdere kleinere architecturen:
- een functionele architectuur,
- een fysieke architectuur,
- een informatie-architectuur, die veel ideeën van ISO management bevat,
- een architectuur van 'logische lagen' (logical layered architecture), inclusief een 'verantwoordelijkheidsmodel' (responsibility model).

Om de korte termijn managementproblemen van het Internet het hoofd te kunnen bieden, heeft de IETF in 1988 het 'Simple Network Management Protocol' (SNMP) gedefinieerd. Hoofdstuk 4 bespreekt deze Internet-managementaanpak. In tegenstelling tot de ISO en de ITU-T heeft de IETF geen afzonderlijk document geproduceerd waarin de gebruikte managementconcepten staan beschreven. De reden hiervoor is dat de IETF gebruik maakt van concepten die reeds beschreven zijn in (verouderde versies van) het 'OSI Management Framework' document, en deze concepten bovendien beschouwde als zijnde vanzelfsprekend. In 1992 begon de IETF met de ontwikkeling van een tweede versie van het SNMP protocol (SNMPv2). In dit protocol zijn een aantal nieuwe concepten geïntroduceerd die helaas nauwelijks worden uitgelegd en daarom moeilijk te begrijpen zijn.

Om managementfuncties beter te kunnen structureren wordt in hoofdstuk 5 een mogelijke classificatie van deze functies voorgesteld. Het hoofdstuk laat zien dat reeds tijdens de verschillende fases van het ontwerpproces over managementfuncties moet worden nagedacht.

In hoofdstuk 6 wordt uitgelegd hoe primaire en managementfuncties op een geïntegreerde wijze ontworpen kunnen worden. Er wordt aangetoond dat het niet altijd mogelijk is het ontwerp van alle managementfuncties af te ronden voordat de operationele fase begint. Dit hoeft geen probleem te zijn, mits er maatregelen worden genomen opdat de netwerkoperator de ontbrekende managementfuncties tijdens de operationele fase handmatig kan verrichten. De ontwerper dient intussen verder te gaan met de ontwikkeling van de overgebleven managementfuncties, zodat in toekomstige generaties van de netwerksystemen deze functies op een automatische wijze kunnen worden uitgevoerd.

Een alternatieve managementarchitectuur, die zowel primaire als ook managementfuncties bevat, wordt in hoofdstuk 7 ontwikkeld. Dit hoofdstuk geeft een aantal voorbeelden die aantonen dat beide soorten functies gemodelleerd kunnen worden met behulp van de regels en concepten zoals die in het OSI Referentie Model zijn beschreven. Omdat er verschillende klassen van managementfuncties zijn en managementfuncties op diverse manieren gedistribueerd kunnen worden, worden in hoofdstuk 7 meerdere modellen ontwikkeld. Al deze modellen bevatten managementprotocollen, alsmede onderliggende service providers die de uitwisseling van managementinformatie ondersteunen. De managementprotocollen worden in hoofdstuk 9 besproken en ingedeeld in twee basistypes: ‘variabel georiënteerd’ en ‘commando georiënteerd’. Hoofdstuk 8 bespreekt de service providers zoals die voor het vervoer van managementinformatie kunnen worden gebruikt.