Configuration Detection and Verification in Computer Networks

Yuko Murayama

Thesis submitted for the degree of PhD

Department of Computer Science
University College London
Gower Street, London WC1E 6BT

Abstract

This thesis identifies a problem which we have termed *Configuration Detection*, and proposes a solution to it. The problem is: how can one learn what objects exist and where those objects are in a given environment? We consider the problem in a computer network environment, which is a collection of various hosts and routers connected together in a management domain.

As internetworking has become popular in both wide area and local area networks, the growth in the number of network objects has been substantial and dynamic. In such an environment, the maintenance of knowledge of the current configuration of the network has become a practical problem.

We identify that the real problems in configuration detection are inconsistency and invalidity. Inconsistency arises because a host can be attached to the network without registration; invalidity is caused by there being no verification of the address announced by a host. These shortcomings could lead to network-level threats such as unauthorised tampering with routing and control, unauthorised use of resources, unauthorised traffic generation, and unauthorised disclosure of information.

We suggest the use of a procedure for the authorisation of network addresses to solve the network level threats. We analyse the flow of an address and suggest where and how authorisation over the network takes place. The flow is verified by using the formal analysis methods suggested recently by Burrows, Abadi, and Needham.

Finally the applicability of the model to other types of information than network addresses is examined.

Acknowledgements

This thesis would not have been possible without the support from my supervisor, Peter T. Kirstein.

I would like to thank Cathy Wittbrodt and Peter Williams for their criticism and encouragement.

I am grateful to Graham Knight, Robert Cole, Zonghou Ma, George Pavlou, James Malcolm, Jon Crowcroft, Ian Wakeman, Denis Timm, Raphael Carbonell, Nigel Chapman, Mike Roe, Steve Kille, Paul Barker, and Søren-Aksel Sørensen all at UCL.

Russel Winder, as postgraduate adviser, provided me with the $\TeX$ system for writing this thesis, and encouraged me constantly to write up.

The people outside the College gave me a lot of help as well. I thank Craig Partridge, Mike Burrows, and Radia Perlman. Peter Hendrickson helped me in getting a copy of David Moon's Chaosnet report.

I am grateful to many people on the tcp-ip list, whose discussions were the source of my research development.

I also thank my friends in the department for their support; those include Gordon Joly, John Andrews, Irshad Buchh, and Ping Hu. Ping Hu explained to me the birth registration scheme in Wuhan, China, as he had done this for his son not long ago.

My parents, Keisuke and Saeko Murayama, believed in my determination and gave me a tremendous amount of support — including the financial aid, which was essential for my study here.

Thank you all.

for

the delightful days at `uk.ac.ucl.cs`

Contents

List of Figures

List of Tables

'*Mon ami* – you know my suspicious nature! I believe nothing that anyone says unless it can be confirmed or corroborated.'

'That's right, old boy,' I said affectionately. 'A throughly nice, trustful nature.'

' "He says," "she says," "they say" – Bah! what does that mean? Nothing at all. It may be absolute truth. It may be useful falsehood. Me, I deal only with *facts*.'

Agatha Christie, *Dumb Witness* (1937)

Chapter 1

Introduction

1.1. Overview

Just as the workings of our society are based on information, so also are the operations of computer networks. We call a system whose operation is based on information an information-based system. How well the system works depends on the *credibility* of the information used.

Computer networks have evolved to the stage where they now provide users with access to a wide range of information resources. Information has long been treated as data in computer networks, and much research has been carried out on how well networks can carry data from one end to another. Shannon identified this level of research as the engineering level [Sha49]. This thesis is concerned with one quality of the semantic level of such information; is the information being transmitted actually correct?

The most significant aspect of computer network evolution in the past decade has been its growth. The open architecture of networks [ISO84] [CeC83] and packet switching technology [CeK78] have contributed not only to the development of single networks, but also to their interconnection; the latter is called internetworking. Earlier work on layering concepts for networking [Pou85] provided a great contribution to the success of interconnection of networks; networks are now connected together within an organisation and across organisations. As a result, the size of networks has grown so easily and unexpectedly that existing management mechanisms now have difficulty coping, being originally designed to manage only a limited number of objects. One of the problems is knowing what objects exist and where they are located; this is a particularly crucial problem if the objects and their locations are important for network operations. We term this problem *configuration detection* in this thesis.

In this thesis we are concerned particularly with network objects such as hosts and routers within an organisation, and the configuration information relating to their location, such as names, addresses, and routing information. Two types of systems have been traditionally responsible for location information; network management systems and information systems. A network management system maintains the location information which is used for management activities of the network environment, such as fault detection and isolation, the installation and removal of a nodal system, and status monitoring. The location information in the management systems has hitherto been managed manually by skillful and knowledgeable operators. This scheme works so long as the size of the network is limited. However, with the growth experienced in wide area and local area networks in the eighties, one cannot hope to continue to cope with the volume and complexity of the information by such manual means.

An information system has been responsible for providing each participant of network operations with the location information. Name servers and directory services are examples of the static type of information systems, whereas the routing information exchange system [Ros82], the address resolution system [Plu82], and the distributed bridge learning system [IEE87] are of the dynamic type. The former have coped with the growth-in-size problem by introducing hierarchically decentralised management [Moc87b], [CCI88b (X.500)], [ChM89], so that the size of local object tables remains manageable — this is the direction that network management systems are following. The latter have adjusted to the new conditions by introducing dynamic learning mechanisms.

Whichever approach is taken, existing systems face two problems: the inconsistency and invalidity of information. Inconsistency concerns the difference between the current and registered configuration; it is caused by the fact that a host can be added to or removed from the network without this fact being reported. Invalidity concerns the misconfiguration of a host or the incorrectness of the registered information; it is caused by the fact that there is no verification mechanism either for the configuration details of a host, or for the registered information. These shortcomings could lead to various threats at various levels. The application level threats in such circumstances are unauthorised access to other hosts, masquerade, unauthorised access to servers, and unauthorised disclosure of information; the network level threats are unauthorised tampering with routing and control data, unauthorised use of resources, unauthorised traffic generation, and unauthorised disclosure of information. The former would be best countered at the application level by authentication and access control. In this thesis, we are concerned particularly with the network level threats, and propose a scheme which will counter some of them: unauthorised tampering with routing and control, unauthorised use of resources, and a particular type of unauthorised traffic generation caused by a chain reaction.

Our solution is the use of certificates for network addresses in information systems; we propose the use of a certificate during an address resolution operation to counter network threats within a subnet. We propose the procedures on how the certificate for a network address can be issued and maintained. Our scheme includes adding two processes to the current address information flow; one is the off-line registration of the address and the other attributes of a network host with an authority during information generation, the other is the configuration verification of the host when it announces itself to a network. Upon successful verification, an authority will issue a certificate for the network address which will be registered in the relevant information systems. Network operations make use of these information systems. This way, only confirmed information can appear in the network operations. In addition, we need a mechanism to detect host removal and revoke its certificate. We show an example of use of a certificate in the Ethernet Address Resolution Protocol operation.

Later in this thesis, we examine the extensibility of our scheme by applying it to types of information other than network addresses resolution information, which will be critical to network operations. They include routing information, time information, trusted peers, application level addresses, and keys for cryptosystems. Finally we show that our scheme is a credible distribution mechanism for configuration information.

The fundamental contribution of this thesis identifies a new problem in networking and presents a solution. Both the problem and solution are extensible. The thesis should be considered as a stepping stone to research in this new area.

1.2. Thesis Organisation

The thesis organisation is as follows. In Chapter 2, the historical differences in research and development of two types of systems which hold location information are examined: the first type consists of information systems such as name servers and directory services; the second type consists of network management systems. From this perspective, related work on information systems and network management systems is discussed. Chapter 3 presents a configuration management model, through which we identify the relation between information systems and network management systems. The model is described in terms of configuration information flow and the associated functional modules which handle the flow. As a result, each module requires the knowledge of relationships between information items and objects. The question of how one can obtain the knowledge leads to the problem of *configuration detection.* Chapter 4 discusses configuration detection in computer networks and identifies two perspective problems: inconsistency and invalidity. We describe the precise threats caused by these shortcomings. Chapter 5 presents our solution to some of the threats described in Chapter 4 — an authorisation system which will issue a certificate to a verified address of a host. The protocol presented is analysed using the formal method recently introduced by Burrows, Abadi, and Needham [BAN89] in Appendix E. Chapter 6 examines the extensibility of our scheme described in Chapter 5 suggesting the use of certificates for types of information other than network addresses. Chapter 7 presents the conclusions of the thesis. Appendix A gives a review of existing configuration detection methods, which will be referenced in Chapter 4. Appendix B describes the birth registration in China, which was consulted when the registration scheme in Chapter 5 was produced. Appendix C presents the model of addressing in communication systems, which analyses the addressing procedures and is referenced in Chapter 5 and Appendix E. Appendix D gives the detailed operations for information maintenance, whose overview will be given in Chapter 5. Appendix E gives the formal analysis of the trustworthy address flow scheme described in Chapter 5. Finally, Appendix F includes the statistics of the current ARP service time, which will be used in Chapter 5.

Chapter 2

Related Work

2.1. Overview

In Shoch's terms [Sho78] one needs to have the knowledge of the following for computer networking activities;

what objects exist,

where they are, and

how one can reach them.

We refer to this knowledge *configuration information* in the rest of this thesis. Network monitoring systems have to know which objects to monitor, where they are, and how they can be perceived from the monitors [Mur85b], [HHS83]. Management systems require the same knowledge but more crucially, as they need to exercise control over objects. For an electronic mail system, when one sends a message, the least one has to know is who could be the receiver and its address. The routing function needs to know what intermediate systems are available to what destinations.

Traditionally there are two types of systems which are responsible for the maintenance of configuration information: one is concerned with the maintenance of information items which are associated with the network objects such as hosts and routers, and which are used for network operations; the other is concerned with the maintenance of hosts and routers themselves. We call the former *information systems*, and the latter *network management systems*. Name servers and directory services are information systems, whereas network monitoring and control systems are network management systems. Both types of systems hold configuration information. However, historically, they have been developed and discussed separately.

In the following sections, we analyse such systems as they really exist or are specified. We look particularly at the credibility of the configuration information held. In each we discuss the credibility of three types of operations; information registration, maintenance, and look-up.

The chapter is organised as follows. The next section presents two information systems, the Domain Name System and the Directory system. Section 2.3 presents various network monitoring and management systems. Section 2.4 presents the work of the standards bodies in the area of network management. Section 2.5 presents an example of the integration of information and network management systems, from MIT's Athena project. Section 2.6 presents the conclusions of the chapter.

2.2. Information Systems

2.2.1. Overview

Information systems provide the information items required for network operations. As examples of information systems, we describe the Domain Name System (DNS) and the Directory Services specified in the X.500 series of recommendations. Other examples include the routing information exchange protocol suites and the address resolution protocol systems which may also be classed as information systems whose operations are distributed. Nevertheless, in this chapter we discuss only name-server type systems such as DNS and the Directory Services, as these have become more generalised.

DNS provides the names associated with hosts in the Internet environment in addition to other attributes. The administration of the information base is decentralised in a tree-structured manner. The X.500 Directory Services recommendations are specified by the International Telegraph and Telephone Consultative Committee (CCITT) and the International Organisation for Standardisation (ISO). The standards are intended to facilitate the name and other resource information services in the Open System Interconnection (OSI) environment.

Both systems originated from the need for the translation from names of network nodes to their network addresses; however, the information held by the systems has subsequently been generalised greatly. This has blurred the distinction between so-called name servers and the data bases of network management systems.

In the following two subsections, we describe DNS and the Directory Services.

2.2.2. The Domain Name System

The Domain Name System (DNS) is a distributed information system for information about domains in the DARPA Internet [Moc87b] [Moc87a]. The information relating to a particular domain includes a canonical name (if any), name servers, the names and addresses of hosts and their hardware/software types, network services available on the hosts, and a mail relay for the domain or a particular host address.

Domains in DNS are hierarchical administrative units of the naming scheme. For instance, our department has a domain name, `cs.ucl.ac.uk.`, where `cs` stands for Department of Computer Science, `ucl` for University College London, `ac` for the UK Academic Community, `uk` for the United Kingdom; here `uk` is the top level domain, and the last dot is the root of the name tree. Other top level domains under the root are `edu` for universities in the U.S.A, `com` for companies, `mil` for military organisations, and `gov` for U.S. government agencies. Non-U.S. countries have their name abbreviations as a top level domain name — though there is a movement to have US domains also include "US" as a top level domain. The top-level and second-level domains must be registered with the Network Information Center at SRI, California, U.S.A. [Sta87] — though there are plans to introduce a mechanism for European sites to be registered in Europe.

A DNS database is divided into *zones*. Zones are distributed over more than one name server, so that a failure of a particular name server does not affect the availability of zones information. Name servers do not have to reside in the zones they support. Zones may be viewed as "cuts" in the tree-structured name space; some adjacent domains of the tree are grouped together to form a zone.

A zone file contains the information about the resources of all the domains within the zone.

The method by which local resources are registered, such as how a host is related to a zone file, is the responsibility of the local management in a zone.

There is a special set of servers, *root servers*, which hold the information on top-level and second-level name servers. When a user enquires about a foreign site, and there is no locally-cached information, the local site name server asks one of the root servers for advice; the reply will supply the information on which foreign name server holds the information on that site. The local site name server then will send a query to that foreign name server. The answer will be cached locally for future reuse.

Thus there are three types of information held locally at each site; 1) local site information, 2) a list of root servers, and 3) foreign site information. The information items on local sites and a list of root servers are refreshed periodically; however the foreign site information will be deleted when a timeout counter for a given information item has expired [Lot87]. Moreover, the local site information is held in a set of name servers for the zone to which the site belongs. One of them is declared primary, and stores the master file as the definitive source of information. The others are declared secondary, and periodically refresh their local caches by checking the versions of the information against the primary source.

From the information *registration* point of view, the above information items are usually based on a file of information edited manually by a human operator. The source of local site information is the local master file. The root server information is partly from the local master file; the list of the current root servers is obtained from one of the root servers. The foreign site information is from the foreign master file, which again is edited manually by a human operator. Thus the reliability of local information depends on the security of the local system environment and the competence of the data entry skills of the operator. The reliability of foreign information is similar to the local information, as it too depends on its own local environmental security and human operator. It also relies heavily on information obtained from the root server, because it is the root server which introduces the foreign name server to the local site. No authentication takes place either for a root server or a foreign name server. There is no confirmation on whether the host associated with a name has been configured on the network or not.

During the *maintenance*, removal and refreshing of information items are controlled by a timeout mechanism. Refreshing activities are performed only by the secondary name servers. However, once again, there is no authentication or access control on these secondary servers. For the *look-up* operation, there is no authentication or control over the access of which users and sites may request information.

2.2.3. The Directory

The Directory [CCI88b (X.500)] is an information system, particularly intended to be used by applications operating in the Open System Interconnection environment [ISO84(IS 7498)] for information look-up. It is similar to DNS described in the previous section, however, its service is more general than merely name resolution. Information is distributed and maintained in a network of Directory Service Agents (DSAs). Look-up operations by users are handled by Directory User Agents (DUAs). An enquiry is passed from a DUA to the DSA, which holds the information item, possibly through some intermediate DSAs.

As in the DNS, the information in the Directory is structured as a tree, according to the hierarchy of organisations. The tree is called the Directory Information Tree (DIT). The DIT is fragmented into naming contexts, which are similar to zones in DNS. A DSA maintains an arbitrary set of naming contexts which may be any part of the DIT. In the DNS, root servers have the knowledge of the location of those fragments (i.e. zones) that contain the top-level and second-level nodes of the name tree; hence, they are always the default reference. By contrast, DSAs chain the operations to get to the final target entry of the DIT. A DSA, therefore, keeps a routing table which maps a fragment (i.e. naming context) to the DSA which holds that fragment.

Thus there are two types of information in a Directory context; one is the information which is structured as the DIT and is queried by the user, and the other is used for maintenance of the Directory. The first type of information is input and accessed by the users of the Directory through DUAs. The second type of information is input by some management authority agent and used solely by the Directory agents, i.e. DSAs and DUAs. In this thesis, we are concerned only with the first type of information.

The credibility of the information registration, maintenance and look-up operations is not mentioned directly in the standard; however, generic mechanisms for the authentication of application entities are presented, viz. simple authentication and strong authentication.

Simple authentication uses the user name and its password. There are three modes of operation, the unprotected mode, and the two protected modes. In the unprotected mode, the user name and its password are transmitted in a clear text, hence, the operation is not secure against interception and subsequent improper reuse. One of the protected modes uses a one-way function to maintain the integrity of the user name and its password; the other protected mode uses two one-way functions — i.e. the output of the first one-way function is an input of the second one-way function — to protect the information transmitted.

Strong authentication uses a digital signature computed by means of a public key cryptosystem as well as a hash function. Three schemes are suggested, viz. one-way, two-way, and three-way authentication schemes according to the number of transactions involved. The three-way authentication scheme is required to authenticate strongly the two sites involved; however, the scheme has been criticised because interception by a third party is still possible [BAN89]. Although a correction has been suggested, the current standard still contains this flaw.

Some applications of the above schemes are suggested in the standard. One application of the authentication schemes is to perform authentication at the beginning of the DUA/DSA session, or during the establishment of the association between two DSAs. Another application is to authenticate the originator of a request, a DUA or a DSA, and the originator of the result, a DSA, for each operation.

Upon the successful authentication, access control can be carried out for the authenticated user. Access control is left as a local design matter in the standard; however, an annex of the standard provides a suggestion of access control categories and protection items. Protection could be provided for user access to the following entities:

- an entire subtree of the DIT
- an individual entry

- an entire attribute within an entry
- selected instances of attribute values

The following access categories for a protected item could be provided:

- to be detected
- to be compared
- to be read
- to be updated
- to be created and deleted
- to be modified

Information is stored into the Directory by the authenticated user, who holds a creating access right, through DAP, but there is no way guaranteeing the correctness of the information. For example, suppose a name and an address of a network host are input to the Directory by the user. Either the address or the name might have typographical errors; or the host might be misconfigured. Therefore, using the Directory as a implicit registration service is not trustworthy, as correctness constraints are neither considered nor imposed. It is possible that the host is removed or changes its location, even though its name and obsolete address are still stored in the Directory. The look-up operation is again through DAP; authentication and access control can be exercised. However, the user may look-up an obsolete or incorrect address of the host without this being realised — until the address is used. Moreover, the cryptographic information used for authentication such as credentials (e.g. keys and certificates) is generated by an off-line Certification Authority. It is not specified how those credentials are input securely into the Directory in the first place.

2.3. Network Management Systems

2.3.1. Overview

In this section we describe the following network management systems: the Automated Network Management System, of Bolt Beranek and Newman Inc. (BBN), NetView of

International Business Machines Corporation (IBM), and the Monitoring and Control systems developed at the Computer Science Department, University College London (UCL-CS).

BBN has been involved in monitoring and control of various networks in the U.S.A. including the ARPANET. As a result of experience in this area, the Automated Network Management (ANM) System has been designed and implemented.

IBM has its own network architecture, Systems Network Architecture (SNA). The management facilities were designed originally for their SNA products, however, the recent management facility, NetView, now fits into a more heterogeneous network environment.

UCL-CS has long been involved in internetworking, particularly in the interconnection of the U.S.A. and the U.K. Monitoring activity has been essential to support the internetwork services.

In the following, we discuss systems developed by the above organisations.

2.3.2. The Automated Network Management System at BBN

BBN has long been involved in the design, evolution, operation and maintenance of various networks such as the ARPA Network [RoW70], the Atlantic Satellite Network [KBC79], MILNET , and the Packet Radio Network [JuT87]. It has been in charge of monitoring and controlling not only the subnetworks, but also the DARPA Internet gateways which connected the subnetworks to form the DARPA portion of the Internet [HHS83]. The heterogeneity and complexity of the Internet was inevitable due to the different natures of the underlying subnetworks. This led to the design of a network management system which could monitor and control both subnetwork nodes as well as the Internet gateways, with features of an expert system [WBB85].

The BBN Automated Network Management (ANM) System has two levels of operation [FLN88]; at the lower level, basic management services such as object control and information retrieval are facilitated, and on a higher level, analysis of the collected information is carried out using expert systems. The lower level management services are provided by a network of Distributed Management Modules (DMMs) which are conceptually similar to the DSAs of the Directory; each DMM is responsible for the control of several network objects such as gateways, packet switching nodes, packet radios, and hosts, and holds their information. When a request for information or control of a network object is received by a DMM from the higher level (client level) application or from an operator, it will perform the operation locally, or it will forward the request to another DMM.

Two types of management protocols are used; one is for information retrieval and the other for requests for control [CCN87]. Neither of them deals with the addition or deletion of an object and its information. Indeed, the information about network object location is currently managed manually.

The information on an object's attributes is input manually. Once the information is stored, those parameters can be tuned remotely to control the associated object. Objects can be polled on request. All control is carefully constrained; it is enciphered, and may be exercised only from specific Network Management Centres.

2.3.3. IBM's NetView

NetView [Kan88] is a network management system product from IBM. It provides the users of IBM's network system, SNA, with network management tools.

SNA [Mei87] started with a centralised tree structure mainly for remote terminal access to a host computer, when it was first introduced in 1974. It has grown first to a network of hosts, and then to an internetwork of SNA networks and non-SNA networks.

The earlier management tools of SNA were driven by the requirements of the IBM service sector when the multi-host SNA evolved, rather than from the users [Sal83]. As the users' network structures became more complicated, and the users' skill in networking advanced, the demand for the network management tools grew [RoM88]. Several management products were released to meet this need, and were subsequently integrated to form NetView. The feature of Generic Alert [Moo88] and the introduction of NetView/PC [ACG88], which provides an interface to non-SNA products, made possible both the management of non-SNA products and the distribution of management.

NetView and NetView/PC provide management services at the application level, making use of management services provided by the lower-level Network Addressable Units. They offer a set of customisable options and application interfaces available for customisation. The management structure is centralised. There is a focal point which is the central point of control and runs NetView or NetView/PC. Other SNA nodes are connected at entry points through which the management information is passed to and from the focal point. There is a service point which runs NetView/PC and may be connected to non-SNA systems.

Four areas are addressed in the IBM network management: problem management, change management, configuration management, and performance management. Among these, change management and configuration management are of interest to us. Change management is concerned with the planning, distribution, installation, and tracking of configuration changes. Configuration management, according to their definition, is concerned with the generation and maintenance of a configuration database that contains a knowledge of all physical and logical network resources and their relationships.

The configuration information is input by network operators at the focal point of control, and then distributed to each site. A password is used for operator authentication together with access control; however, the credibility of the information itself is not verified. Alternatively, in the case of a local area network environment, the configuration of hosts is learned dynamically. The configuration report server at each local subnetwork collects the reports of changes, and informs the service point, a multi-LAN manager, of the current configuration [WiM88]. How much integrity would be maintained for the configuration information is not clear to us due to the limited amount of documentation readily available on this system.

2.3.4. The systems at UCL-CS

2.3.4.1. Overview

UCL-CS has been active in internetworking in both wide and local area networks. As we used to provide the internetworking services between the networks in the U.S. and the U.K., we started monitoring at the user application level. The relevant monitor was called the Catenet Monitor. Soon after, a local monitor, the Status and Alarm System (SAS) was created; this later incorporated the Catenet Monitor, as well as another monitor of our service ports towards the other U.K. sites. These experiences led a team to design and create the prototype INCA OSI management system.

In the following subsections, we describe the Catenet Monitor, SAS, and the INCA network management system.

2.3.4.2. The Catenet Monitor

The Catenet Monitor [Col82] [CoK82] was aimed at monitoring the availability of gateways on the path to a number of hosts in the U.S.A. and elsewhere over the Catenet — which became the DARPA portion of the Internet later. Availability was determined by means of sending a probe packet to each gateway and analysing the responses.

A pair of connectionless "request and reply" packets at the network level were used as a probe. A probe is composed of a pair of Internet Control Message Protocol (ICMP) [Pos81b] packets; i.e. the monitor sends an ICMP Echo Request to a gateway and waits for an ICMP Echo Reply sent back from it.

A list of gateways to be monitored is included in the monitoring program; hence, the program had to be recompiled when the list required any change. Although the compile procedure does validate the input data to a certain extent, the configuration information is still error-prone. The system worked as long as the number of gateways to be monitored was manageable. No access control was exercised explicitly for monitoring information look-up. The program ran on a specific machine, and anyone who had access to that machine could have run the program.

2.3.4.3. The Status and Alarm System (SAS)

The SAS [KMW84] was created originally in order to monitor the departmental local networks, in which a number of systems for computing, networking, and internetworking are scattered over interconnected local area networks [CDH84]. The main purpose of the SAS was to detect any failure of local systems as soon as possible.

Like the Catenet Monitor, it made use of probes to monitor the local hosts over the Cambridge

Ring [JNT82]. Again, a pair of the connectionless request and reply type of protocol packets was used as a probe. The Single Shot Protocol was used for the probe operation. Later, the SAS integrated two more probe systems; one was the Catenet Monitor Probe as presented in the previous subsection [Mur85a], the other was the probe system operating at a level higher than the network level. The latter was called the Service Probe [MuM85]; it examined the availability of the service hosts in the Department which provided external users in the U.K. with network services, such as electronic mail, file transfer, and remote terminal access, to the U.S. The probe made use of a pair of open and closed sequences of a connection-oriented transport protocol through a generic transport service interface, the Clean and Simple interface [Ben80] [BrH81].

The system maintained a configuration file which contained the physical equipment names and the services to be monitored. A service on each physical system was monitored periodically following the order of the configuration file. If a system was down which was not scheduled, an alarm message would be sent to the system managers by means of electronic messages. The status information was shown to an arbitrary user on a UNIX host by using a command, status — e.g. `status 44c` (requested the status of the PDP 11/44 processor called the 44C machine).

Like the Catenet Monitor in the previous section, the probes own configuration information was maintained by human input, and therefore was error-prone. Access control for information look-up was done passively through the account management; i.e. anyone who had an account on the UNIX systems at UCL-CS had access to the status information.

2.3.4.4. The INCA network management system

The INCA network management system is a prototype OSI Network Management System developed at UCL under the ESPRIT Project INCA [Kni89].

The prototype system [KPW88], [PKW89] is composed of several physical subsystems operating over the Ethernets, and management functionalities are distributed over the subsystems. Each managed subsystem has a management agent and the agent reports to a set of manager processes of open and close connection events. The reports are maintained in a log file. It is planned that the result of hourly analysis of logs could be stored in a Directory. There is a management process which displays the current status of the subsystems. The process maintains management level connections to each subsystem it monitors and detects the crash of a subsystem through the status of the connection; a crash can be detected because there should be a periodical synchronising transaction between two peers on the transport level during a connection. The subsystems communicate with manager processes over the ISO Common Management Information Protocol (CMIP) [ISO89b(DIS 9596)].

The knowledge of the configuration of subsystems is not learned dynamically; it was a policy that the implementation of the prototype should be as simple as possible, and dynamic learning was not a goal. Thus the initial knowledge of the subsystems depends heavily on human input. While the maintenance of the knowledge is done by the monitor process, neither authentication nor authorisation is exercised for the communication between a managed subsystem and a manager process. No access control is exercised at the moment for look-up of the information items.

2.4. Integrated security in information and management systems

2.4.1. Overview

In this section, we describe an environment in which the information system and the management system are integrated. We look at the specific example of the Athena Project environment. The Athena environment is concerned with application-level security rather than at the network level which is the primary interest of this thesis. Nevertheless, we present it here, because there is no equivalent for network-level information.

2.4.2. The Athena Project environment at MIT

Project Athena at the Massachusetts Institute of Technology (MIT) started in 1983 as an experiment to explore innovative uses of advanced computer technology in the educational environment [BLP85]. The perspective of the project was the evolution of personal computers, workstations, and network technology.

The Athena system [Tre88] is composed of personal computers, workstations, and host computers, connected together by Ethernets as subnetworks. The subnets are linked together by a high-speed fiber optic spine network. The users are mainly students, who use personal computers or workstations to access the system. The system operations are decentralised into servers, such as the authentication server [MNS87], the name server [Dye88], the mail server, the network file store server, the remote virtual disk server, the notification server [DEF88], and a server for the servers, the service management system [RJL88]. The servers run on host computers, dedicated personal computers and workstations.

In order to manage the environment in a coherent way, the service management system (SMS) is responsible for centralised management of authoritative information and automated information distribution. Each server receives the information items required for its service operations from the SMS. The SMS maintains information such as servers, their locations (i.e. machines), mail lists, the members of the mail lists, user information, and access control lists.

The user accesses information through servers such as the name server. This server, in turn, contacts the SMS for information access through the SMS library. Alternatively local access is possible by using the database administration program at the SMS. The verification and validation of an information item is exercised either by the SMS or by the application which has called the SMS during addition of the information item, depending on the type of information.

One application is the registration program for new users. The records of registered students are input off-line to the SMS by means of magnetic tapes, which were input originally manually and contain students attributes such as names and associated ID numbers. The password registration takes place on-line through the program (Registrar). Registrar asks a new user for name and student ID number for authentication. Then the user can input the proposed principal identifier and password. The principal identifier is further examined with the currently registered identifiers. If it is not in conflict with any other, then the identifier and the password are registered.

The authentication scheme for registration is simple; it relies on the strong assumption that the users are students and everyone will register at a certain stage. If any spoofing action takes place by using a new user's name and ID number, it will soon be discovered by the authentic user when he/she eventually registers.

When the SMS updates the information base in each server, it sends its authenticator to the server in the beginning of the update session; thus the receiver can be sure of the identity of the information provider, the SMS. A checksum is added to the information sent for data integrity purposes. Each server including SMS has its own access control list. Access control at each server is exercised over the user previously authenticated by the authentication server, Kerberos [SNS88].

2.5. Conclusion

Network management systems are responsible for monitoring and controlling network objects, such as hosts and gateways. The object configuration information such as name and location is the key input to the management operation. The integrity of source information has depended heavily on the skill of the trusted operators of the management systems. This scheme worked because the number of managed objects was limited. As internetworking becomes popular in local and wide area networks, the size of the objects and the complexity of network structure increase. The maintenance of configuration information becomes significant compared to the maintenance of objects themselves.

On the other hand, the information systems have been coupled tightly with network operations, but coupled loosely with the managed objects. As more network operations have tried to make use of the information systems to derive the basic information, the information the latter hold has become more general from the original name-and-address mapping. The diversity of information items they now hold has become so widespread, that the cooperation of so-called name servers and network management systems is an inevitable requirement in networking.

One approach is the generalisation of name servers as the Directory System. The latter's specification is intended to provide a resource information directory function which network managers and operators can request the necessary information items.

Another approach is the integrated architecture practised in the Athena environment. The idea is to provide a central management point of information. From there, various network operation servers including name servers can receive the necessary information items. It is similar to NetView where, however, the information is for the purpose of management only.

These information systems and network management systems are vulnerable to errors in the registration information, because no verification takes place. The SMS of the Athena Project has good provision for information verification and validation. However, when it comes to a specific information item such as user information, the authentication of a user at registration is weak as verification takes place before the authentication server can get involved; a user is authenticated by presenting the name, the account name for computers, and the password. How securely those parameters are maintained up until the moment of registration is up to the user. The Athena structure provides good and robust information maintenance and look-up operations, however, it does not provide for integrity of the original information. Perhaps, for the user information, verifying more personal attributes of a user than merely the name and password may prevent a deceiver from masquerading.

The information held by the information systems is coupled tightly with network operations but coupled loosely with objects, because it is used by network operations but it is not verified with the status of objects — e.g. an object might have been removed. On the other hand the information held by the network management systems is coupled loosely with the network operations but coupled tightly with the objects, because it is not used by network operations but it is used for the management of objects. Moreover, the current management systems may miss an object which is attached to the network without registration in the management systems. These considerations would lead to inconsistency — a situation where an object is attached to the network but not yet known to the rest of the network, or an object is known to the information system but has not been installed yet or has been removed. Together with the possibility of invalidity — misconfiguration of objects or incorrect information in the information systems — we would face the application level and network level threats described in Chapter 4. Among the possible threats, we will deal with some of the network level threats caused by the incorrect network addresses, viz. unauthorised tampering, unauthorised use of the resource, unauthorised traffic generation by an intentional network storm. In this thesis we deal with the protection of the internetwork within an organisation from these network level threats, proposing a network address registration system which maintains the integrity of the network addresses; this system is a configuration information system which is tightly coupled with both operations and objects.

Chapter 3

Configuration Management in Computer Networks

3.1. Overview

In this chapter, we describe the activities concerned with the configuration management of computer networks.

The ISO standard in network management defines five areas, viz. fault management, configuration management, performance management, accounting management, and security management [ISO89c]. In this thesis, we are concerned with only the single aspect of configuration management. Note that configuration management in this context is different from the traditional discipline called configuration management used for software version control [BHS79]; that is concerned with how to maintain the revisions of hardware and software modules during system development.

Configuration management of computer networks is concerned with the configuration changes which are the location-related and availability-related changes of network objects. The management activities deal with these changes.

In this chapter, we define network objects, configuration changes and the activities needed to deal with these changes. We present our model of configuration management, and discuss a problem arising. The next section describes network objects. Section 3.3 discusses configuration changes from a network-wide viewpoint. Section 3.4 presents the configuration management operations needed to deal with these changes. Section 3.5 presents a model of configuration management. Section 3.6 describes an empirical trial of the design of a Directory subtree for the storage and retrieval of management information, and a related problem area. Section 3.7 concludes the chapter.

3.2. Network objects

The network objects with which we are concerned in this thesis are hosts and routers in packet-switched computer networks; in ISO terms, hosts are conceived as end systems, whilst routers are intermediate systems. Hosts run applications that use each other's resources to perform user level tasks. They are connected to a network and inter-communicate over the networks. A router is connected to more than one network; it forwards an incoming message originating from one host to another router, or delivers it to the destination host if this is connected to the same subnet as the router.

3.3. Configuration changes

Configuration changes of the network objects can be specified from a network-wide perspective being related to both the location and availability of the objects in networks. Location-related changes comprise these situations when:

- a new object is introduced into a network.
- an object is removed from a network.
- an object is moved to another network.
- an object is replaced by another object (including the case of object version change).

Availability-related changes comprise these situations when:

- an object becomes operational and is ready to accept a request to process network packets.
- an object becomes unavailable due either to failure, or because an intermediate object has become unavailable so that the object is not reachable.
- an object becomes unavailable due either to performance deterioration, or the congestion of the incoming request queue.

These changes are performed by a set of operations; the local change of an object, recognition of the local change, the subsequent changes of the objects which are affected by the change, and the update of the associated information. In the next section we will discuss these operations.

3.4. Configuration management operations

3.4.1. Overview

Configuration management has to deal with the network-wide changes described in the previous section. We identify two types of change operations; the first performs local changes at each object site, and the second performs network-wide changes. The latter includes the steps of the recognition of the local changes, the subsequent change to other objects which are affected by the original change, and the network-wide information control.

Although an object is considered as a logical entity — e.g. a host or a router, in reality it is composed of a set of software and/or hardware modules. The local change operation deals with such physical objects as these whose function, as a whole, is perceived as a single logical entity; the physical objects are a subnetwork interface, network software, and parameters for network operations which may be specified in a configuration file, or may be set with physical switches.

The recognition of the local changes is required by the network management system and by the other network objects through the information system. The management system may decide to pursue further changes on other objects, if required. The information system is responsible for distributing the updated information about the changed objects, so that other objects in the network learn about the changes.

In the following subsections, we discuss each of these types of operation. The next subsection discusses local configuration change operations. Section 3.4.3 describes the recognition of the local changes. Section 3.4.4 discusses subsequent change operations. Section 3.4.5 describes information control.

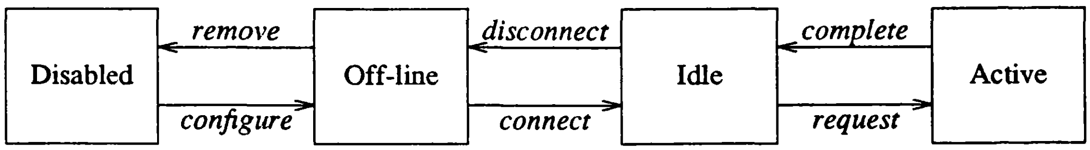


Figure 3.1: An object's state transition diagram

3.4.2. Local configuration change operations

The configuration changes presented in Section 3.3 are performed firstly at each host by means of configuration change operations; the operations are *configure*, *remove*, *connect*, *disconnect*, *request*, and *complete*. Figure 3.1 shows the object state transition by these operations.

The state "Disabled" indicates that the host is not configured. In the "Off-line" state, the network hardware and software are installed, the operational parameters are set, but the host is not in operation. In the "Idle" state, the host is operational over the network and ready to process a network packet. In the "Active" state, the host is actually processing a network packet.

The operations in Figure 3.1 are as follows:

- *configure*: the object is configured — i.e. the network interface and the network software are installed and the operational parameters are set.
- *remove*: the object is removed from the environment — i.e. the network interface, the network software, and the associated parameters are removed.
- *connect*: the object starts the network operations.
- *disconnect*: the object is disconnected from the network — i.e. the network process has been stopped.
- *request*: the object accepts a send request for a network packet from the transport layer process or receives a packet over the network from another host.

- *complete*: the object finishes dealing with the packet.

Using these operations, we will describe the network-wide changes once again. During the introduction of a new object, we need the configure and connect operations. For the removal of an object, we need the *complete*, *disconnect*, and *remove* operations. For the *complete* operation, the manager has to wait until all the current application-level transactions finish. Alternatively the management process at the object could stop all transactions, and send abort notices to the peer objects of those transactions, if required. Moreover, in this situation, although the object's state is idle, the object is not ready to send and receive any further packet because it is in the process of being disconnected; this is the 'quiescent' state defined by Kramer and Magee [KrM90]. An object is quiescent if it is not currently engaged in any ongoing transactions that it initiated, it does not initiate new transactions, it rejects any new transaction requests, and no other transactions have been or will be initiated by other objects which require service from this object.

Moving an object is a combination of a removal change and the introduction of the same object at a different location. The replacement of an object is composed of a removal and the introduction of a different object at the same location.

An object becomes operational when it is connected. It is unavailable when it is disconnected. It becomes unreachable when an intermediate object has become unavailable; this could be viewed as the object being itself unavailable. Therefore, the connect operation includes not only boot-up but also the fact that the path from the object is made available. Also, the Off-line state can indicate not only that object itself is unavailable, but also that the object is not reachable because the intermediate object is unavailable.

3.4.3. Recognition of configuration changes

The location and availability changes can be discovered in several ways as follows:

- by off-line reporting from the person who installed or removed the object.
- by receiving a report over the network from an object when it started operation.

- by an enquire-and-reply protocol initiated by a manager agent on the network — a reply is made by the object or its representative agent.
- the objects are discovered and observed passively by others on the network either by monitoring packets sent by those objects.

3.4.4. Changing other objects

If the configuration of an object is changed, such a change may affect other objects. This situation requires that knowledge of the relationships between the objects exists, in order to see which objects are affected. Such knowledge may exist either in a central manager site, or can be distributed over several sites. It could even be distributed to each object; i.e. the change is broadcast or multicast to a set of objects and each object would decide whether it needs to change itself. Such a self-configuring feature might be attractive for a distributed operation environment.

3.4.5. Information control

Configuration information of the objects stored both in the network management systems and the information systems, has to be updated when any configuration change occurs. If the information and network management systems are coupled tightly with the objects' local configuration files, the update operation is trivial. The update of a local file may be propagated automatically to the network management systems and the information systems; alternatively the update may be made at just one of those systems, and local sites import the change. If these systems are coupled loosely with the objects, the update operation is required in each system. In either case, the knowledge of the relationships between information items spread over different systems (i.e. how one update can potentially affect other information items) may be required for consistency. As in changing other objects, the knowledge could be distributed directly to each object, such that an update can be broadcast; it is up to the receiving system to modify information items upon the receipt of the update.

3.5. Configuration Management Model

3.5.1. Overview

Driven by the discussion in the previous sections, we present our configuration management model.

The next section presents the participants of our configuration management model. Section 3.5.3 describes the facilities required for local change operations. Section 3.5.4 presents the facilities required for the network-wide change operations. Section 3.5.5 describes the operation and information flow in our model. Sections 3.5.6 through 3.5.9 present the information held by each participant of configuration management.

3.5.2. The participants

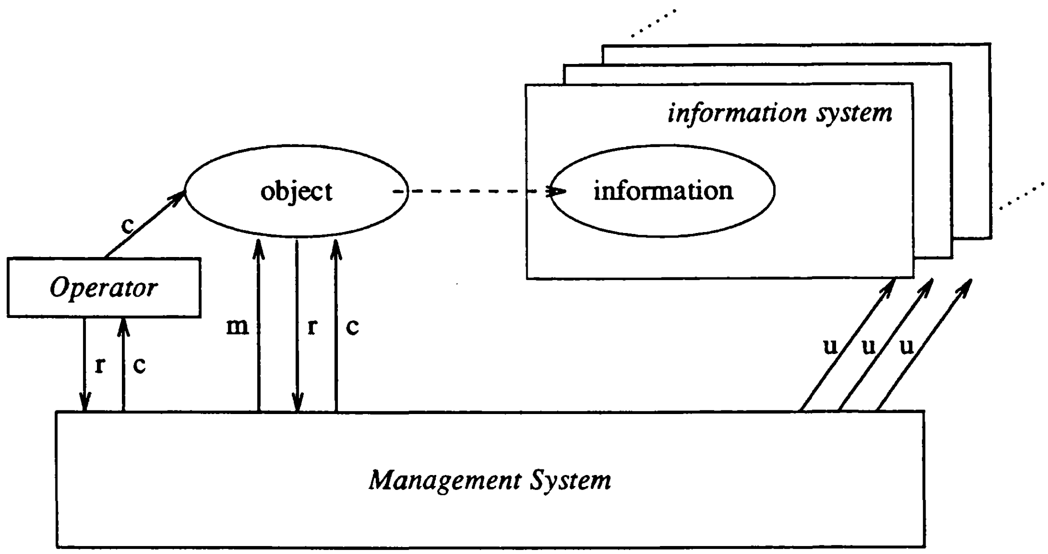
In our model, the participants of configuration management are a managed object, an operator, the management system, and the information system. Figure 3.2 shows our configuration management model.

An object is a managed entity with interfaces to the operator for local settings and to the network management system. An operator is a person who is in charge of the hardware and software maintenance of an object. The management system may ask the operator to perform the necessary local changes; in this case the operator may report the completion of the change back to the management system. The local change operations to the object may be done entirely locally by the operator; alternatively it could be done by the management system over the network.

The management system is responsible for recognising the local change of an object, for invoking subsequent changes to other objects which are affected by such a change, and for updating the relevant information items maintained by the information system.

The information system is responsible for accepting the update from the management system and ensuring the necessary update operations to all necessary information items. The other object sites will learn of the change through the information system.

We presume that all these operations would be performed over the network, except for the



Legend:

c : control
m : monitor
r : report
u : update

Figure 3.2: A network object configuration management model

local change operation performed by the operator.

3.5.3. The facilities required for the local change operation

The local change operations require the following facilities to control the object:

- to install and remove a set of software or hardware entities
- to set the parameters for network operations
- to boot up
- to lead the object to the quiescent state; i.e. to force object not to complete the current transactions, not to initiate new transactions, and prevent the others from initiating any transaction to the object
- to shut down the object

The above operations can be performed locally; in this case, the management interface to the object is provided only locally. However, where remote control is preferred, a network management interface is clearly required.

To change the object into the quiescent state, the manager may need to have access to the information system, so that it can inform the other sites on the network not to initiate new transactions. Alternatively, this can be done by forcing the object not to reply to any request; any site which tries to initiate a transaction will simply get no reply and it will abandon the attempt after the expiry of a timeout.

3.5.4. The facilities for the network-wide change operations

For the recognition of a local change, we require interfaces from the manager either directly to the objects or to the operators; the interface to the object is used for monitoring, the interface to the operator for receiving a report of a change. Monitoring facilities include unsolicited or solicited reporting mechanisms from the objects themselves, and from traffic monitors which watch traffic to detect changed sites.

The knowledge base of the relationships of the objects, which is used for subsequent change operations, should be maintained by the manager. Subsequent changes include further local change of other objects as well as updating information associated with these changes.

For the information control, the manager should maintain the knowledge of the relationships between the information items in the different information systems, which require consistency. According to the knowledge, the manager can propagate the update.

3.5.5. The operation and information flow

For a configuration change, the management system may send a request for a local change operation to an object; alternatively it issues the request to an operator who performs the change locally. For remote control of the object, the management system may send the request with a unified identification of the object's parameters. In this case, the object maps the unified parameters into the object-specific parameters. This is the approach made in the network standardisation activities [ISO89e], [PaT87a], [Ros90], [LaB90]. The request from the management system may carry the information items such as the *object ID*, the *operation*

ID, and the *parameters for the operation*. A *parameter for the operation* is composed of a *parameter ID*, and a *parameter value*. Examples of these parameters to be controlled or examined, are protocol operation addresses, an address mask of a hierarchical address, a maximum transmission size of the protocol packet, and so on.

For recognition of local changes, there are four cases as follows:

- An operator sends a report of change to the management system.
- An object sends an unsolicited report to the management system.
- The management system queries the object and the object replies.
- The management system detects the change by observing network traffic.

The report from the operator or the object includes the *object ID*, the changed *parameter ID*, and the new *parameter value*. After the change recognition, the management system sends an update request to the relevant information systems. The update request includes the *object ID*, the *operation ID*, the *information item ID*, and the new *value*. The *operation ID* may be one of addition, deletion, and modification. The information systems could be the Directory and name servers, a set of network nodes which participate in a routing information exchange protocol, and the other network management systems.

Changing one object may invoke further necessary local changes of the associated objects and the updates to the relevant information.

3.5.6. The information in the management system

The management system has to maintain the following information:

i. the objects to be managed, the configurable parameters and valid values, and the allowed operations on those parameters.

ii. what objects exist currently, and the latest status of their configuration parameters, so that it can detect change.

iii. the relationships between objects, in order to specify the consequent changes — e.g. when an IP entity is added, the associated protocol entities such as routing protocol entity for the subnetwork ought to be installed as well.

iv. a set of the information systems, their locations, and the information types which they hold

3.5.7. The information maintained by the operator

The operator must know the information listed in the previous subsection. In addition, the operator has to know the physical location of the object, the attributes of hardware and software such as vendor, product type, and version number, the contact telephone number for repair, the information about service contract, and the history of the object; i.e. installation date and failure records [MuC88].

3.5.8. The information in a managed object

The information in a managed object includes the attributes of the object itself, and the attributes to customise the group operation protocol entity for the object. For example, the information in a gateway in the DARPA Internet Protocol (IP) environment [Mur87], [Bra89] and [BrP87], includes the followings:

i. The attributes of the object
   - the IP address of the object,
   - the version number of the IP specification which the object uses

ii. The attributes of the subnetwork and network
   - the subnetwork mask
   - the maximum transmission unit size for the subnetwork
   - a default routing table
   - the address resolution table for the subnetwork if desired

iii. The attributes controlling object operation
   - options for some particular operations such as a support for source routing, record routing, and time stamp options

The subnetwork mask is the mask to filter the subnetwork address out of the IP address. This mask is used to see if the destination address is on the same subnetwork or not when a packet is being sent. The default routing table can be loosely specified because when the outgoing packet is sent to one of the routers specified in the table and if there is a router with a better route for the packet, the host is informed of the location of the preferred router by the default router. The address resolution table translates the IP address into the underlying subnetwork address. The table could be maintained dynamically — e.g. the Ethernet Address Resolution Protocol [Plu82], but even in this case strong enforcement of a particular translation can be exercised by specifying it in the configurable table.

For management purposes we may also need the following specifications:

- Error types
- Error levels for logging and its frequency
- Event types
- Event levels for reporting to the manager and and its frequency
- Manager's ID and its location

3.5.9. The information in the information systems

An information system holds the information items looked up by the network objects; these information items are relevant to network operations, such as name-and-address translation and routing.

In the DARPA Internet, for name-and-address translation, a name server holds the information items such as the names of the registered network entities, their network addresses, and the higher protocol entities associated with the network entities together with the systems information in which those entities reside [HSF85]. The latest Domain Name System [Moc87b] includes more information such as the mail relays to be used [Par86].

A network of routers may maintain the following information [McQ74], [ISO89a]:

- next hop routing tables
- neighbour hosts and gateways (i.e. adjacency matrix)

- measured quality of service characteristics of a link
- a part or full map of the network topology

The neighbour hosts and gateways are the ones which are on the same subnetwork link as router so that they can be reachable without any other intermediate router.

A routing information exchange protocol is responsible for maintenance of some or all of the above information. For example, the topology information is learned through the information exchanged by the protocol in the original ARPANET routing algorithm, whereas a second ARPANET routing algorithm [MRR80] assumes that each router has the previous knowledge of the topology information [Per85].

For management purposes an information system must know the management system, from which it receives updates. It may also have to know its users for access control.

3.6. A design exercise of a Directory for the management information

3.6.1. Overview

As we argued in Chapter 2, the current network systems lack communication between the network management systems and the information systems. Our configuration management model proposes how they could possibly communicate from the network management side to the information systems. Another idea for coupling of these two tightly could be that the network management system uses an information system to store the information required for management. We therefore tried to incorporate some of the information required by the operator into an experimental Directory following the ECMA specification [ECM85], which was developed under an ESPRIT project, THORN (THe Obviously Required Name-server); the first objective of the THORN project was an experimental use of a Directory, and the second objective was the replication of the information base among the Directory Service Agents [HGK86]. The ECMA specification was relatively stable at the time of trial whilst the X.500 specification was still far from completion.

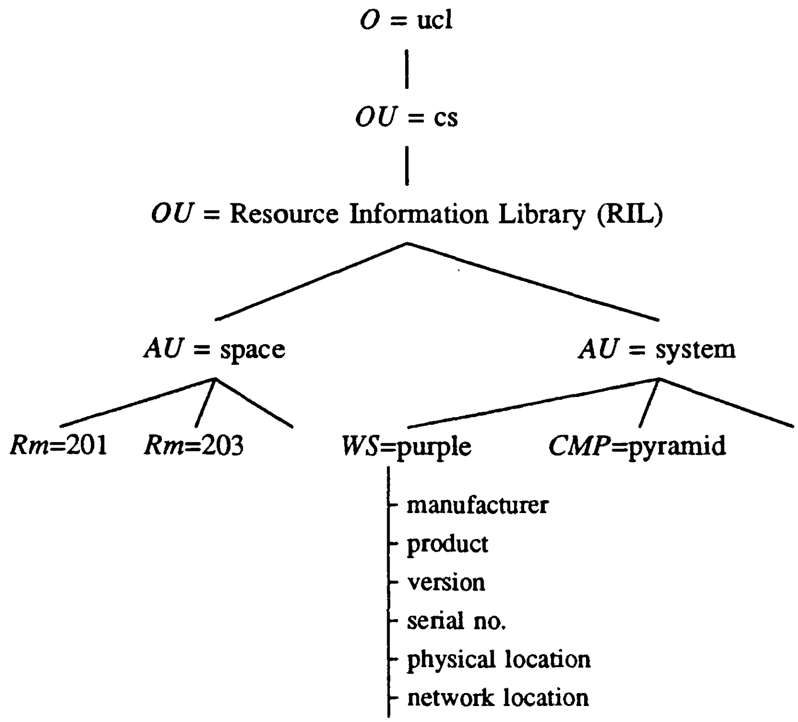
The Directory was used for the personnel and application service information in our department. The service information included electronic mail-related information such as postmaster and mail protocol, and other network service information such as file transfer protocol. Therefore, it seemed meaningful to see how the management information could fit in the Directory structure.

3.6.2. The structure of the Directory

The Directory was structured as a tree; the top level was country, the next level was composed of organisations such as University College London (UCL), and the third level was composed of organisational units such as departments. The path name from the root to the node in the Directory which held the information about a particular object will give the hierarchical name to the object; our department was named as follows: the country name was Great Britain (GB), the organisation name was UCL, and the organisational unit name was Department of Computer Science (CS). The personnel information was under the subtree of the Department of Computer Science.

However, we created another subtree under our department to differentiate the management information part from the personnel information part. Figure 3.3 shows this subtree.

We designed a subtree named Resource Information Library (RIL) under which were architectural units (AUs); a space AU and a system AU. Under the space AU, we specified physical location unit, i.e. rooms, and under the system AU, there were various types of systems on the network such as multi-user support systems with non-intelligent consoles, ones with intelligent consoles (i.e. workstations), personal computers, non-intelligent end systems, and intermediate systems (i.e. routers). Each room node had the names of the systems in the room. Each system node had information such as name, manufacturer, product, version, serial number, physical location which is a pointer to the room node, and the network location which was a network interface address. As a result, the management information about one of the hosts, say A, was reached from the root through the following paths: GB, UCL, CS, RIL, system, and A. This was a mixture of hierarchical naming and the hierarchical file store concept as in the UNIX system. It would have been nicer if we had located the management information on the same level as the personnel information, so



Legend:

O : Organisation
OU : Organisation Unit
AU : Architectural Unit
Rm : Room No.
WS : Work Station
CMP : Computer

Figure 3.3: The Resource Information Library Subtree

that the information on System A could have been held in the node named GB.UCL.CS.A.

3.6.3. Creation of a subtree of the Directory

In order to create the RIL subtree of the Directory, we needed two operations: one for creating a basic structure of subtree, and another for inputting individual nodal information. The former was done by preparing two basic files of the specification of the tree architecture; one was for node type definitions, and another for property definitions for each node. The Directory system read this file and created the subtree space.

The latter was to input the actual information about rooms and systems following the subtree structure — e.g. one of our systems would have an entry like the system name was purple, manufacturer was SUN Inc., product name was SUN, the network address was 128.16.5.1 and so on.

3.6.4. Problems arisen from the design exercise

Throughout the exercise, we faced the three problems as follows:

1. incorporating network configuration in a tree-structured directory
2. classification of systems; according to the administrative domains or the system types.
3. input of information

In the beginning we tried to store network configuration information; i.e. what subnetworks connect what hosts. The configuration itself was a graph structure, and the Directory is a tree. Using the alias feature which relates one entry to another by means of pointer, we could possibly express the configuration of the network. As a result, we would need a lot of pointers across the tree and this would complicate the Directory structure. Therefore, we decided to store only host system information.

Another problem was the design of the system subtree. The system subtree could be divided according to the projects to which the systems belong. Another way to divide the subtree could be according to the classification of the type of the systems. We took the latter because the former classification would be only significant during the lifetime of a project; a system belonged to one project would be used later by another project.

Before we completed creating this subtree for the operator, we realised the third problem, the amount of information we had to type in for each entry in the Directory; for each host we had to input seven attributes, and there were hundreds of hosts. It was not only time-consuming but also error-prone. The management system might not rely entirely on this manually input management information. We realised that it would make life easier if at least the name and address of currently existing hosts were learned through the network.

In the previous section, we have listed the amount of information required by each participant of the configuration management. The problem is that we are neither sure where exactly the information at each participant comes from nor how credible it is.

We realised the need for a credible acquisition mechanism for the management information. This lead us to look at a problem of *configuration detection*; i.e. how can we acquire the very basic information required for configuration management, a current list of existing objects and their locations? This will be examined in detail in the next chapter.

3.7. Conclusion

In this chapter, we have introduced the activities required of configuration management of the network objects and presented our configuration management model in which the information systems are attached closely to the network management system. The configuration information in each participant in the model has been presented.

An empirical trial of design of a directory subtree for the local systems operator lead us a question; how one can learn the existence of the objects and their locations. We call the problem *configuration detection* in this thesis. It will be discussed fully in the next chapter, and we will seek for a solution of the problem in the rest of the thesis.

Chapter 4

Configuration Detection

4.1. Introduction

This chapter presents a novel problem which we term *Configuration Detection.* It is the method by which one can learn the current configuration of the objects of interest in a given environment. In this thesis, we are interested in the objects such as hosts and routers in interconnected local area networks within an organisation.

Configuration Detection in data networks has not been previously considered to be a serious problem, because the size of the networks (in terms of the number of nodes) was small, heterogeneity of the host systems was manageable and the Systems Manager did not need extra tools to keep track of the managed objects and their locations. With the evolution of open architectures of protocol stacks and standardised network interfaces including widely distributed network software, internetworking has become popular. As a result, the size of network has grown and the heterogeneity has become unmanageable. One way to cope with the management of this environment is to have a dynamic learning system, e.g. a routing information exchange protocol, and the Ethernet Address Resolution Protocol; another is to divide the management domain into smaller subdomains in a hierarchical manner. Whatever approach is taken, we see two problems: *inconsistency* and *invalidity.*

Inconsistency is the state of difference existing between registered information and actual configuration; it is caused by the fact that installation and removal of hosts can be done without registration, or vice versa. *Invalidity* includes the incorrectness of registered information and the misconfiguration of a host; it is caused by the fact that there is no verification mechanism. Indeed, configuration detection has become a problem because of these two underlying factors.

In this chapter, we examine the consequences and threats from inconsistency and invalidity, and highlight the particular ones dealt with in this thesis. The next section describes the problem of configuration detection. Section 4.3 discusses the consequences of inconsistency. Section 4.4 describes the consequences of invalidity due to inadvertent errors. Section 4.5 presents the security and integrity threats posed by deliberate intent to deceive. Section 4.6 examines these potential consequences and threats from the shortcomings identified in the earlier sections. Section 4.7 discusses a solution to the particular threats. Section 4.8 concludes the discussion of the chapter.

4.2. Problem Description

Network activity without knowledge of the configuration is just like climbing a mountain without a map. *Configuration Detection* is defined as the acquisition of the knowledge of the current configuration.

There could be two possible ways for ascertaining the configuration information:

- to enquire from a registered configuration information base what the configuration should be
- to discover the objects through the network, ascertaining what the configuration really is (see Appendix A for various methods)

Ideally, both of the above would be used together; one requests from the information base the registered information, and compares this with the actual configuration. One mechanism would be to use a combination of application-level management protocol messages and broadcast ping (see Appendix A, section A.5.6). Our scheme is composed of two sessions: a network management transaction phase and the broadcast ping (note that the term, ping, indicates a pair of the ICMP Echo Request and associated Reply session). In the first phase the properly registered hosts are disabled from answering the ICMP Echo Request, and in the second phase a broadcast ping takes place. This way, we can detect the hosts which are not registered but exist on the network. The scheme is as follows:

1. A configuration detector (CD) sends a network management protocol

message to each registered host. The message tells the host to turn off the ICMP Echo feature. The host should acknowledge a successful operation to CD.

2. CD executes a broadcast ping. Given that all the registered hosts are turned off, only unregistered hosts would reply. We make the assumption that the number of unregistered hosts is limited enough not to flood the network.

During this procedure, we have to overcome two problems; inconsistency and invalidity. Inconsistency is the difference between the registered information and the real configuration; the registered information may not be up-to-date, or the configuration of a host may yet to be done. It is caused by the fact that an object can be added to, and removed from the network without registration. Invalidity means that the registered information may be incorrect, and an object may have been misconfigured. It is caused by the fact that there is inadequate validation, verification, and guarantee both for the registered information and the configuration of a host.

We categorise the inconsistency caused either by incorrect registered information or misconfiguration as invalidity. In our definition the difference between inconsistency and invalidity is as follows: inconsistency indicates that either the information or the configuration of a host was once true but now obsolete, whereas invalidity indicates that either the information or the configuration of a host has never been true.

In the following sections, we explore the possible consequences of inconsistency, invalidity from inadvertent errors, and threats from intent of deliberate deceit.

In order to make our discussion simpler, we presume that the objects of interest are hosts and routers in a specific environment, the Internet Protocol (IP) over a subnet, the Ethernet, belonging to one organisation, and that their IP network addresses are the information of concern. In the rest of the chapter, we mean an Ethernet address by a subnet-level address, whilst we mean an IP address by just an address or a network address.

4.3. Consequences of inconsistency

The difference between the registered information and the real configuration, is caused by a configuration change without a corresponding registration change, or vice versa. The following shows the possible inconsistent cases:

- the latest configuration status of a host has *not* been registered, but it has been
    - added — the configured address X with no registered address (Case I)
    - changed — the configured address Y with the registered address X (Case II)
    - removed — no configured address with the registered address X (Case III)
- the latest configuration status of a host has been registered, but it is
    - not added — no configured address with the registered address X (Case IV)
    - not changed — the configured address Y with the registered address X (Case V)
    - not removed — the configured address X with no registered address (Case VI)

The removed cases include the situations that hosts are brought down due to failure. Cases IV through VI can be treated in the same way as Cases I through III; Cases I and VI are identical, so are II and V, and III and IV. In the following discussion of inconsistency, we will look into Cases I through III.

In Case I a host is attached to a network, but not known to others on the network; the available resource of the host, such as processing and storage resources, cannot be used by the others.

In Cases II and III a host address is registered as X, but changed to another address Y, or not yet attached to the network; its users are denied services.

The problem with inconsistency is that these cases would coexist. Imagine that a host address X has been changed to Y, and a newly introduced host is now configured with X. A client uses the server on the address X, assuming that it is the original host while the new host may provide a similar, but slightly different, service. The client may not realise the difference until there is any significant problem in using the result of the service.

The situation can get more complicated with the possibility of invalidity — e.g. in the above example, a deceiver can come up with a host configured with the address X, masquerading the authentic host which has changed its address into Y. We will examine the inconsistency in the presence of invalidity in the following two sections, firstly from inadvertent error, and secondly from deliberate deceit.

4.4. Consequences of invalidity caused by inadvertent errors

In this section we discuss the consequences of invalidity caused by inadvertent errors. We exclude the situation that both the registered address and the configured address have the same incorrect value. In other words there is always inconsistency between the registered address and the configured address, when invalidity occurs.

Under the condition of an inadvertent error, a registered address could be incorrect, or a host could be misconfigured; a system may malfunction causing it to appear as if it were misconfigured with a different address — e.g. dropping a bit.

The invalidity of the registered address caused by inadvertent errors and the misconfiguration of a host could cause the generation of unnecessary traffic, inadvertent masquerading, denial of services, and packet regeneration. Unnecessary traffic could be generated towards a foreign site, if the address of a local host is registered as a foreign address. Inadvertent masquerading would occur if the currently registered address of a host is used by another while the authentic host has not been configured yet. Denial of service could happen if a host is misconfigured with the same address as another. Some misconfigured hosts and routers with the same subnet level address could invoke the regeneration of packets which could become a network storm. Those which can be detected sooner or later by the users of the hosts are not serious. However, inadvertent masquerading, some cases of denial of service, and packet regeneration could be serious; some of them would not be easily detected at the user level. Also, the scale of the resulting effect might be large, and the management activities may have difficulty in identifying the trouble maker. These cases will be explained below.

If a new host is configured with the registered address of a local host which is not configured, other hosts will mistake the new host for the authentic one. If the authentic host is a server, users on the other hosts will try to connect to the new host and may get denial of services. If the new host offers a similar service at the same service port, the users would get the incorrect service without noticing. This would also happen if the registered information was not correct — i.e. if a host address was registered as Y while the host was configured with the address X. As described in Chapter 2, there is no verification of information stored in the DNS and the Directory. Inadvertent masquerading by mistaking a host as another — be it caused either by inconsistency or invalidity — could waste a user's time because he may be getting a different result from the wrong host — e.g. he may compile source code on the wrong machine. If the authentic host with the correct address X was foreign, the name of the foreign host would be bound to the local address Y; the new local host would receive the traffic from foreign sites. If this new local host provided a similar service to the authentic one, the users might not realise the fact.

If a new host is configured with a local address which is neither registered nor configured, the host can function without being noticed. However, later, if a new legal host is configured with this address, the original host will face denial of network services when the new host starts operating; alternatively, the new host will face denial of services when the original host reinitiates a connection to a server. As the original host is not registered, the network manager has neither knowledge of it nor access to it, so it will be hard to isolate the fault.

Moreover, the host which is not registered could cause another host difficulty in management, when it was misconfigured with a duplicate subnet level address; this could cause a network storm by a chain reaction as is explained below.

Another type of serious misconfiguration is that host is configured with both the router's or server's subnet level address, (i.e. a duplicate subnet level address), and the forwarding capability which is usually for routers. The amount of the traffic destined to the authentic router or server will be doubled; the misconfigured host will receive all the packets destined to the router or server and forward them to the router or server. If the misconfigured address is that of a router, packets may travel beyond the organisational networks causing unnecessary traffic over the inter-organisational networks; this could be serious as the the resources of other organisations are consumed in vain. Likewise, the packets destined to the misconfigured host will be forwarded by the router. If the address is that of a server, the traffic towards the server may cause congestion at the server's interface.

A network storm could be created when three or more routers, or misconfigured hosts with the forwarding capability, are configured with the same subnet level address with different network level addresses. The following sequence was identified by Perlman [Per91]. Suppose there are three routers, A, B, and C; note that this could happen also by having hosts misconfigured with a forwarding capability. A packet is sent out with the destination X which should be forwarded to the router C. The packet is also received by A and B as they have the same subnet level address. As the destination network level address is neither A nor B, they forward to C; the packet forwarded by A is also picked up by B as well as C, and B then forwards it, it will be picked up by A and forwarded again. In this way, each packet sent to one of these three routers will be multiplied by two others until they finally expire at the timeout specified in the time to live (TTL) field of each packet.

Generally, if there are n routers and misconfigured hosts with the same subnet level address, a packet sent to one of them will be regenerated by the other $n-1$ routers and hosts for the first round; one of the n routers is a correct router for this packet and would forward it off the subnet. From the second round on, each of those $n-1$ regenerated packets will generate $n-2$ others. In this way, packets are generated exponentially. Eventually, the network is flooded with messages, and the only way to halt the traffic is to remove the misconfigured routers and hosts. The number of regenerations, S, is expressed as follows:

$$S = (n-1) + (n-1)(n-2) + (n-1)(n-2)^2 + \ldots + (n-1)(n-2)^{t-2}$$

$$= (n-1)\sum_{i=1}^{t-1}(n-2)^{i-1}$$

which may be expressed as follows:

$$S = \frac{(n-1)\{(n-2)^{t-1}-1\}}{n-3}$$

Either a large n or t would be enough to cause the network to be flooded; the TTL could be specified as large as 255 — the maximum TTL, as the TTL field is one octet long. Surprisingly, some of the popular systems in our Departmental network are sending and receiving the Network File Store maintenance packets with the TTL value of 255.

This effect could be invoked easily by the misconfiguration of a host with a subnet level broadcast address, because all routers take packets sent with the subnet level broadcast address, and try to forward them. This is called "a chain reaction" coined by Manber [Man90] to characterise the phenomenon that a design failure in a computer or in a network protocol will propagate throughout a network, causing a breakdown of the entire network.

4.5. Threats from deceit

This section considers the consequences of invalidity from the perspective of deliberate deceit; a deceiver uses his own host to mount various attacks. We presume that the deceiver has no authorised access to the system which holds registered information; however a passive attack, such as unauthorised information disclosure, may be possible by observing traffic passing through a network.

The following threats are possible:

i. Application level threats

1. **Unauthorised access to hosts**
2. **Masquerading**
3. **Unauthorised access to application servers**
4. **Unauthorised disclosure of information**

ii. Network level threats

1. **Unauthorised tampering**
2. **Unauthorised use of the resource (resource stealing)**
3. **Denial of services by unauthorised traffic generation**
   a. **a storm by a chain reaction**
   b. **victimising a host by unsolicited messages**
4. **Unauthorised disclosure of information**

For **unauthorised access to hosts**, i-1, a bogus host may masquerade as another with a registered address of a host which is not configured, or whose address is registered incorrectly. Alternatively the host may interfere in the transaction over the network between a name server and a user, or between a local name server and a foreign name server, by answering with a bogus host address faster than the authentic name server, or by means of redirecting traffic, network threat ii-1. A deceiver may well have superuser access on the bogus host, and can gain unauthorised access to the resources and information on other hosts which trust the bogus host and its superuser. Access control information may be modified, so that, later, the deceiver can gain proper access to any one of those trusted hosts; it is also possible that the proper users may be denied service, or a bogus host can be introduced as a legal server into the master zone file for the DNS — normally only a superuser is allowed to modify this file.

For the **masquerading** attack, i-2, a deceiver masquerades as a server on the same bogus host as above; i.e. a malicious process may masquerade as a genuine service. It may gain information from the client process, or even provide the client sites with the incorrect information.

For **unauthorised access to application servers**, i-3, using a bogus host, the deceiver masquerades as a client, gaining unauthorised access to the servers.

For **unauthorised disclosure of information**, i-4 by observing the traffic, one can see the information carried between hosts.

In the network level attack, ii-1, **unauthorised tampering**, network packets can be redirected to a bogus host, giving it control over the network packets, by misuse of a routing information exchange protocol including an unsolicited management message like an ICMP Redirect message. A bogus host might be configured with an unassigned address. It initiates a routing information exchange protocol, and other routers start forwarding packets to this bogus host by mistaking it as a new router. Alternatively two bogus routers can be used in such a way that one is configured with the address of the existing local router, and the other with an unassigned address. These bogus routers never appear in the local subnet operations, so that it will not be detected by the authentic one. The first bogus router sends an unsolicited network level management message, an ICMP Redirect message, to the victim hosts indicating that further messages should be sent to the second bogus router. The other hosts correct their routing tables by replacing the authentic router with the second bogus router. In both instances, the deceiver gains control over traffic which should have gone through the authentic router. It will drop some of the packets, and forward the rest to the authentic one.

Another way to redirect packets is possible if there exists inconsistency, e.g. a router is removed without notifying the hosts which use the router. A bogus host pretends to be the authentic router by answering ARP Requests from victim hosts; the victims then forward packets to the deceiver. Alternatively its source address is changed to the deceiver's address, and the packet is resent to another possible router if any; in this way the deceiver can intercept the transaction between the victim and the destination hosts.

For **unauthorised use of the network resource**, ii-2, a deceiver may steal network bandwidth without being noticed, by using an unassigned local address. A deceiver may gain unauthorised access to network services, by communicating with another bogus host.

Unauthorised traffic generation has two aspects; one is to provoke a chain reaction in a network; the other to produce unsolicited packets. A ii-3a attack uses a chain reaction. A deceiver configures a bogus host X with a network address which is not in use, and with a subnet level broadcast address. The deceiver produces an IP packet destined for itself, and broadcasts it out to the subnet. The packet is regenerated by the routers and misconfigured hosts with the forwarding capability described in the previous section. The deceiver could make use of a subnet level address other than the broadcast address; however, in this case, he needs to have more than three hosts configured with a forwarding capability, all of which have the same subnet level address.

For a ii-b traffic generation attack, a traffic generator can be configured by a deceiver with any address, foreign or local, and then it can be used for attacking a site, or a group of sites. A bogus host will never be detected and go on attacking from time to time by changing its own address. The generator can produce hundreds of request packets to victim hosts. The victim hosts will be unable to process the quantity of packets, and consequently the users will be denied network services due to overloading. This could lead to denial of distributed services such as the Network File Store, due to congestion.

Unauthorised disclosure of information, ii-4, is learning of some of the registered addresses by unauthorised observation of traffic.

4.6. Evaluation of threats

4.6.1. Overview

In this section, we analyse the threats according to three factors, viz. the impact, the mechanisms required for both the detection of an attack and protection from the associated threat. In the next subsection, we examine the impacts from threats. Section 4.6.3 describes the detection of attacks and the protection from threats, and Section 4.6.4 evaluates threats.

4.6.2. Impacts from threats

Among the threats, the serious ones are those which would let a deceiver have control over important resources, such as confidential information, access control information, and other information crucial to the application and network operations. In this sense, the application level threats are all serious so long as hosts hold such confidential or crucial information worth attacking — e.g. a student may be interested in the examination papers being prepared by the lecturers, or an employee may be interested in the employer's administrative information.

Among the network level threats, unauthorised tampering is serious. A deceiver can have control over the packets which may carry the information used for application and network operations; unauthorised tampering could also lead the application level threats if there is no protection at the application level for the integrity of the information being transmitted — use of ordinary checksum such as the one for a transport layer protocol would not work because the bogus router could calculate the checksum of the tampered packet using the well-known algorithm. The problem with redirecting at the network level is that even if there was an application level protection, just dropping a packet would be harmful; if the packets of a database update message or a routing information exchange protocol message were dropped, an information item in a database may never be updated, or an available network would be considered as unreachable — this will lead to application level denial of services.

Traffic generation is a problem as it can cause denial of services — e.g. application operations over the network such as the Network File Store, Remote Procedure Call, and name server operations may time out and this would cause the cessation of the application level operations. In particular, traffic generation caused by a chain reaction is serious because all the routers on a subnet have to be disconnected to cure the problem. Unauthorised disclosure of information at the network level would cause no harm as far as intra-organisational networks are concerned unless the deceiver is a spy from outside the organisation, who sees traffic by attaching his own system; if this is the case, the organisation which wants to hide its network structure, could well have a security system at the user level — such as user access control to the building which has networks.

Compared to the above threats, network level resource stealing is less serious; the LAN bandwidth is usually big enough so long as the unauthorised use of the network level resources does not interfere with the authorised distributed operations such as NFS — which has a retry interval of 0.7 to 30 seconds. Therefore, an overload of the LAN over such a duration would not disturb the user application, although the user may face performance deterioration — a minor denial of services. However, it could cause a serious management problem; if any of those illegal hosts is misconfigured with a subnet level address the same as some others, or with a subnet level broadcast address — this could cause a network storm;

network-wide fault isolation would be difficult because the host is not known to the network manager.

The seriousness of information disclosure at the network level, i.e. disclosure of network addresses, depends on organisational policy — e.g. a military organisation would not like to disclose its network architecture.

4.6.3. Possible detection and protection mechanisms

Authentication and access control of servers and clients at the application level will solve many of the previously described problems — e.g. the application level threats, i-1 through i-3. By trapping unsuccessful authentication and access control at each application, attempts at deceit may be detected.

Detection of the **unauthorised disclosure of information** attack at either the application or network level (i-4 and ii-4) is impossible, because the deceiver only observes the traffic but may not communicate with others over the network. However, it can be prevented by encryption at each level — i.e. the encryption of all the packets of concern. At the network level, for example, by observing packets on the network, one can see which network addresses are in use. This could be prevented at the user access level; these could be physical access controls to the building, and application level gateways which would translate the real address into another, so that the structure of the local networks would be kept from being disclosed to an outsider. It could also be avoided by the encryption of all packets at the network level.

Unauthorised redirection at the network level could be detected and prevented by the authentication and access control at each router as well as at each host. However, routing is a complicated operation by itself. Incorporating security features would make the operation more complex. For this reason, their incorporation would not be desirable in intra-organisational networks, although it may be necessary in the inter-organisational environment. Alternatively, unauthorised redirection can be prevented at the subnet level — i.e. the prevention of the problem in a smaller domain. Encryption could be used to prevent a deceiver from joining the network without knowing a key. Ideally we would encrypt entire subnetwork protocol packets; alternatively, we could encrypt specific management ones — e.g. the ARP packets. Without knowing the cryptographic attributes for the ARP operation, one could not join the network operation easily. Although unsolicited packets could still be sent out by the bogus host, no ordinary host, which uses the ARP to maintain the network-to-subnetwork mapping table, could send any packet to the bogus host. Even if the bogus router sends a redirect message to the other hosts to introduce another bogus host as a new router, other hosts cannot forward the packets to the new router because its subnet level address would never be learned. However, this solution is vulnerable when the registered host is misconfigured, or when a deceiver lurks among those who know the ARP key. This will also prevent **unauthorised use of the network resource**.

To counter **unauthorised traffic generation** caused by a chain reaction, each router can filter the incoming ARP packets so as not to accept the ARP information with a subnet level broadcast or multicast address.

No exact solution would prevent unauthorised traffic generation by unsolicited messages. However, locating the subnetwork on which a deceiver is operating may be possible. Each site monitors the load on the network interface, or the retry counts at higher layer operations. If one site observes that its interface is congested, the host should send an alarm to its console; it cannot send a message because the interface may be busy. If the deceiver is operating beyond a subnet and the attack lasts a long time, by examining the load on the subnet interfaces in each router, or by using a subnet level traffic analyser, one may be able to detect the subnet that the deceiver is using; one cannot, however, tell from which host a deceiver is sending packets, since the deceiver's packets include a fake source address instead of the real source address of his host.

Any attack using a foreign address as a source address can be solved at an application level gateway most rigorously. The gateway controls the application level traffic between the local organisation and the other organisations; it authenticates a server on one side of the gateway and a client on the other, and access control is exercised. Traffic generation attacks on foreign sites can be eliminated in this way as well.

4.6.4. Evaluation of threats

The application level threats, i-1 through i-4, are serious as long as the information being accessed is important. They are protected best by authentication, access control, and encryption at the application level.

Unauthorised tampering by unauthorised redirection at the network level is serious because it causes denial of services at the application level, no matter how strong the protection mechanisms are provided at the application level. Unauthorised redirection should be prevented by having counter measures either at the network level or the subnetwork level. Network level protection, however, does not protect the network from a deceit which makes use of inconsistency. The subnetwork level protection also counters a relatively minor threat, resource stealing. It also provides protection from unauthorised traffic generation caused by a chain reaction; however, it is prevented more easily by having a filter of incoming ARP packets at each router.

Unauthorised traffic generation by unsolicited messages is serious; it may cause denial of services at the application level, and it cannot be prevented. Detection of the location of the attacker is possible but only to the extent determining the subnet on which the deceiver is operating. It is difficult to identify the host being used by the deceiver, as the generated packets may not include the real address of the host.

Information disclosure at the network level can be a serious problem in an inter-organisational environment. In this thesis we are not concerned with disclosure of network address information since our interest is the intra-organisational networks.

4.7. Our solution

Authentication and/or access control at the application level would solve many problems which are obstacles for configuration detection; there would then be little point in pursuing a deceit at the application level as a deceiver can benefit little. Inadvertent masquerading would be discovered easily by the user because the bogus server cannot be authenticated.

Among the serious network threats, traffic generation by a chain reaction is solved easily by filtering the ARP packets. Traffic generation by unsolicited messages is difficult to protect against, but it may be possible to determine the subnet on which deceiver is operating.

In this thesis, we try to protect the network from unauthorised redirection, ii-1, on the subnet level as well as the misconfiguration of a router or host with an improper subnet level address and the forwarding capability. Our idea is to solve the problem within a subnet by the certification of network and subnetwork address mapping. When a host starts up, it has to go through a registration procedure. During registration, the host configuration is verified. Upon a successful registration, a host is given an authorised token. This implies that a host which is misconfigured would not get the token. Given the list of certified hosts, configuration detection is trivial. The token is a certified address mapping, which should be used in the ARP operation, allowing only authorised hosts to appear in the ARP table at each host. This prevents an innocent host from sending a packet to a bogus host, because the innocent hosts use ARP to locate the deceiver's server.

This measure does not prevent the generation of unsolicited packets, ii-3b, or traffic observation, ii-4. It does not cope with a set of unregistered hosts communicating with each other without using ARP, either.

4.8. Conclusion

In this chapter, we have introduced *Configuration Detection*: how one can learn the current configuration of objects in a given environment. We looked particularly at the Internet IP over the Ethernet environment, where the open architecture of the network would allow anyone to bring up a host without the registration of its address.

With the presence of inconsistency and invalidity, configuration detection becomes problematic. Inconsistency is the difference between the registered information and the actual configuration, whereas the invalidity is the incorrectness of both the registered information and the host configuration. The consequences from inconsistency and invalidity from inadvertent errors, and the threats from intent of deceit, were explored and examined. Many of them can be solved by authentication and/or access control at the application level. However, network level threats need a network level solution. Among the network level threats, unauthorised tampering and traffic generation are serious; the former will be dealt with in this thesis.

Our solution solves these problems within each subnet, by providing authorisation of network and subnet level address mappings. In the next chapter, we will present such an authorisation system as a solution to these network level threats.

Chapter 5

A Trustworthy Information Flow for Addresses

5.1. Overview of the chapter

One current problem in computer networks is that a host can appear in the network and announce an arbitrary address. This can give rise to inconsistency and invalidity conditions; these leading to the network level threats described in the previous chapter. In this chapter we present a preventative solution; verification of the host configuration and authorisation of the host address. We analyse the flow of address information, and suggest where and how the authorisation of an address takes place. The flow is examined using formal methods suggested recently by Burrows, Abadi, and Needham [BAN89] — details are given in Appendix E.

We look specifically at the Internet Protocol (IP) over the Ethernet environment in this chapter.

The chapter is organised as follows. The next section discusses the current address information flow, and presents our proposal for address information flow. Sections 5.3 through 5.5 describe the detailed operations of our address information flow; object generation, information generation, and information registration are presented in Section 5.3, information systems are discussed in Section 5.4, and information look-up is presented in Section 5.5. Section 5.6 describes the characteristics of our scheme. Section 5.7 presents the state transition diagram of a host involved in our address flow scheme. Section 5.8 analyses our scheme. Section 5.9 discusses the maintenance of timers clocks in the network. Section 5.10 describes the required functions of our authorisation system. Section 5.11 discusses architectural issues. Section 5.12 describes viability. Section 5.13 describes how the proposed scheme applies to the current Ethernet Address Resolution Protocol system. Section 5.14 presents the chapter conclusion.

5.2. The address information flow

We observe the flow of a network address in the Internet IP over the Ethernet environment in the following example. Firstly, it is decided to install an IP host or router into the network. We call this *object generation*; the object is an IP host or router. Secondly, the address of the host is allocated, and the address is configured into the object at the point of the installation. We call this stage *information generation*. Thirdly, the host starts network operation, and thus makes its address known to the network. We call this activity *registration over the network*. Fourthly, the address becomes known to others on the network. This distribution is done either by using a network information repository, such as a name server, or by dynamical learning operations such as the Ethernet Address Resolution Protocol. We call such an information repository and the dynamic learning system, *information systems*. We also call the other hosts' address learning activity, *information look-up*. The other hosts are *information users*. Figure 5.1 shows the flow of addressing information.

Our goal is to protect the network addresses from several network level threats, viz. unauthorised tampering, resource stealing, and the network storm type of unauthorised traffic generation. What is missing in the current address flow scheme is that there is no mechanism for verifying the credibility of an address, or the binding between host and address. The credibility of addresses can be verified at any stage — e.g. *information generation*, *registration over the network*, information maintenance at *information systems*, and *look up*. We take the approach that the credibility is verified strictly by *registration over the network*. The reason for this is: that for the subnetwork level addresses, such as the Ethernet level addresses and the network part of an IP host address, the Naming Authorities (NAs) who allocate addresses are possibly in a different domain from the host's domain. This may make it difficult to generalise how much the operations are error-free, both for the authorities and for the host, i.e. the *information provider*. Neither is it practical for another host, acting as the *information user*, to verify credibility every time it looks up the addresses of others.

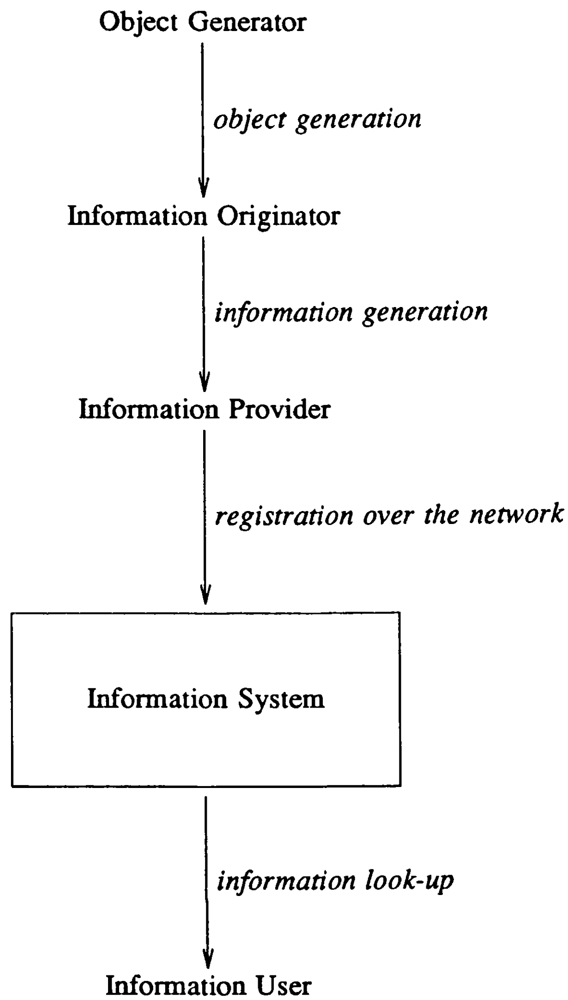


Figure 5.1: The address information flow

Thus, the accuracy check at the time of *registration over the network*, where the information item first comes into the network, seems the most reasonable way.

In our model of address flow, we add two operations to the current scheme; one is off-line registration, before the installation of a host, and the other is the verification of configured addresses, during registration over the network. A host address is registered with a Certification Authority (CA) before it is configured into the host. Later, during registration over the network, the configured address is verified by the management system using the information previously registered off-line.

In the following sections, we describe the detail of these operations. Section 5.3 describes object generation, information generation, and information registration. Section 5.4 describes information maintenance. Section 5.5 describes information look-up.

5.3. Object generation through information registration

5.3.1. Overview

Trustworthy address registration requires the rigorous synchronisation of the object generation, its address generation, and the address registration operations. It is our claim that the synchronisation can take place by means of *confirmation* of the association between the address and the host once the host is installed. Confirmation is carried out by matching the address claimed by a host with the address from a previous off-line registration.

The activities involved in object generation through address registration are listed below:

1. Resource Admission (*object generation*)
2. Naming (*information generation*)
3. Certificate Registration (*information generation*)
4. Configuration (*information generation*)
5. Configuration Confirmation and Registration (*registration over the network*)

During Resource Admission, local environmental policy is used to make the decision of whether or not to allow a host to be configured. An address is allocated to the host during the Naming process. The address and associated cryptographic system attributes of the host are registered off-line during Certificate Registration. If the registration succeeds, the registration ID is issued by the Certification Authority (CA). During Configuration, the host is introduced to the network. During Configuration Confirmation, the host uses the network to register its address using the registration ID; verification of configuration takes place which ensures that the host is bound to the address which was assigned during Naming, and then the verified address is registered with information systems.

The strict scheme of birth registration in China introduced in Appendix B, was referenced for developing the above scheme.

In the following subsections, we introduce a scheme for object generation through to address registration. The next section describes object generation. Section 5.3.3 discusses address generation. Section 5.3.4 presents address registration over the network.

5.3.2. Object generation

The object of our concern is an IP host or router. By an IP host or router, we mean an Internet IP process associated with an Ethernet interface. The object in our terms, therefore, includes hardware and software for Ethernet operations as well as the software for the Internet IP operations. These hardware and software reside in one physical location called a *system* in the following discussion. The address of concern is the IP address, and we assume that the Ethernet address has been defined and configured by the vendor of the Ethernet interface card.

At object generation, resource admission takes place; an object manager, i.e. a local network manager, requests the installation of an IP host, and the request is examined by the organisational administrator. Permission for installation of an IP host may be requested with a suggested location — i.e. to which subnet the host might be attached. The request is evaluated against current organisation-wide resource management and network planning policies. If the permission is granted, the administration and management domain for the IP host is then determined, together with the access control policy. As a result of the examination procedures, a "permit" is issued to the object manager by the organisational administrator. With the permit, an Internet IP address associated with the Ethernet interface can be requested. The permit includes the location, i.e. the subnet to which the host should be attached, and the access rights of the host.

5.3.3. Information generation

5.3.3.1. Overview

Information generation includes naming, certificate registration, and configuration. The *information originator* is the Naming Authority of an IP address.

In the subsequent three sections, we look into each step in the process.

5.3.3.2. Naming

Once the installation of an IP host is permitted, an address will be allocated. Naming is the management of the address space. An IP address is composed of three parts; a network number, a subnetwork number, and a local host addresses (see Appendix C for detail). Addressing is decentralised in a hierarchical way. A network number is allocated by the Defense Data Network (DDN) Network Information Center (NIC) at SRI International [RSR88]. A subnetwork number is assigned by the network manager who administers the address space for the particular network number. A local host number is assigned by the local network manager.

For instance, the allocation of a host address is composed of the following processes:

i. to learn the range of host number space for the subnet to which the host will be attached

ii. to select a host number from the free addresses in the host number space learned at i

iii. to produce a host address by concatenating the Subnetwork address and the host number

Note that the Ethernet interface address is already assigned by the manufacturer of the interface card, though it is possible to change this address. In the case that we specify our own Ethernet address, we need our own organisation number from the Naming Authority (for a manufacturer number), which is the Institute of Electrical and Electronic Engineers, Inc.

The verification of whether the address has been already allocated to another host and whether the network and subnetwork parts of the address are correct, could be checked by the Naming Authority's internal mechanisms. In the proposed scheme, a more stringent verification takes place much later during registration over the network.

5.3.3.3. Certificate Registration

The allocated address is registered together with the cryptographic system attributes and the description of the host system. The information given here will be used later for configuration confirmation when the host starts registration over the network.

In our scheme, we assume an asymmetric two-key cryptographic system for network operations; a secret key and a public key. Since we are interested in the maintenance of integrity of an address more than the confidentiality of an address, we shall use a secret key to encipher, and the public key to decipher [NeS78].

An IP host associated with an Ethernet card is registered with the following attributes by the Certification Authority:

- the permit issued during object generation
- a verification ticket which shows the existence of the hardware and software for the host — e.g. documentation of the delivery of the system from the vendor which includes an Ethernet interface address.
- the allocated IP address
- the location of the host; this includes the management domain, the Ethernet address, and the name of the system into which the IP software and the Ethernet interface card are installed.
- the description of the IP host system, including the attributes of the system, and the detail of the application services that it offers through this network interface.
- the cryptographic system attributes; the public key of the host
- the certificate of the requester

Authenticity of the permit and the verification ticket are examined. Also the authenticity and credibility of the requester is verified. The correct location and access host information in the permit is used to decide what network operation keys are required — e.g. the Ethernet ARP key. Some of those keys could be given to the host later during the registration over the network. The host description information is used to decide the security class of the host system. The security class is included in a security label given to the host later during the registration over the network. The security label will be used for information maintenance operations.

On a successful registration, a random number called a *Registration ID* is given to a host for the future registration confirmation. The Registration ID acts as a session key for registration during the Configuration Confirmation Phase. The Certification Authority's public key may be given to the object at this time as well. A key or a pair of secret and public keys for network operations such as the Ethernet Address Resolution Protocol operation could be also given. With this kind of key, authorised entities can encrypt and decrypt the packets of a certain operation.

5.3.3.4. Configuration

Configuration activities include the installation of hardware and software into the system, and setting configurable parameters used by an IP host.

The IP and Ethernet software and an Ethernet interface card are installed into a host machine with the following information:

- the registration ID
- the secret and public keys of the host
- the Ethernet and the Internet IP addresses
- the cryptographic keys for particular network operations such as the secret and public keys for the Ethernet ARP operation
- the public keys and locations of the trusted CA, managers and monitors.

When the system is booted up, it reads the above information from the specified location within the system, e.g. a configuration file, and the IP process and its associated Ethernet interface become operational. In the proposed scheme, when an IP host is booted up for the first time, just after the completion of its configuration, it has to go though registration over the network. The registration ID and the host's secret key will be needed to complete this registration.

Configuration is an off-line system generation process during which the hardware and software components of the host are installed into a system. Configuration confirmation and registration, explained in the next section, are on-line processes in which the notarisation of addresses is carried out over the network.

5.3.4. Registration over the network

5.3.4.1. Overview

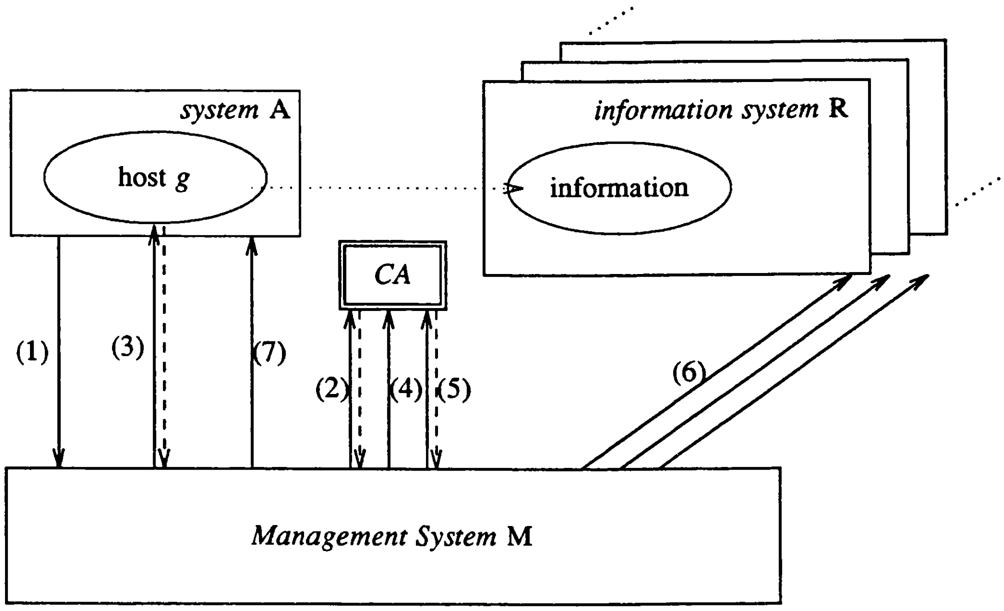
Registration over the network includes configuration confirmation, and the registration of host information. The *information provider* is the host which starts registration over the network. We are less concerned with the authentication of the information provider, than with the authentication of the claimed information item — i.e. if an information item is *correct*, it does not matter who has provided it. At configuration confirmation the *credibility* of the IP address and associated Ethernet address is examined. Upon a successful verification, the authorisation of the IP address and the Ethernet address mapping is carried out. The authorised mapping is then propagated by using the Ethernet ARP operation by each host.

In the following two sections, we discuss configuration confirmation and registration of the authorised address mapping.

5.3.4.2. Configuration Confirmation

The purpose of configuration confirmation is two-fold: the first is to verify the credibility of configuration (e.g. binding of an allocated address to the host); the second is to validate the allocated address.

When the host is installed with the IP software and the Ethernet interface, it has to go though configuration confirmation in order to get the IP and Ethernet address mapping authorised. The registration ID is only known to the host and the Certification Authority (CA). After verification and validation, authorisation takes place. Authorised address mapping is a pair of IP and Ethernet addresses signed by the CA. It is through the ARP operation, that other hosts learn the authorised address mapping of the new host. In this case, the ARP operation



Legend:

(1) : A initiates the registration of g
(2) : M makes a request to CA for the information associated with registration ID
(3) : M verifies that g is operational.
(4) : M reports to CA that g is operational
(5) : M asks CA for notarisation
(6) : M distibutes the updates to the Rs
(7) : M gives the notarised address to A
——→ : request or report
←- - - - : reply

Figure 5.2: The flow of configuration confirmation and registration

is regarded as an information system. In our scheme, the authorised information could be stored in a database such as a directory system, and the other hosts will learn the address of the new host from the directory. In the latter case, the directory is an information system.

In the following we introduce a protocol for configuration confirmation as well as registration. Figure 5.2 shows the protocol flow.

The configuration confirmation sequence is as follows:

1. A host uses the network to inform the manager of the completion of its

configuration using the registration ID

2. The manager gathers the host information, including the associated registration ID, which was registered at Certificate Registration.
3. The configuration of the host is verified using the previously registered information
4. Upon successful verification, the manager recognises the addition of the host.
5. The host address is authorised
6. The authorised address is registered by the information system.
7. Any network operation group key such as the one for the Ethernet ARP operation may be given to the host

The detailed packet exchange of the configuration confirmation protocol is described in Appendix E, Formal Analysis.

In steps 1 through 3, authentication of a registration session for a host is carried out. A registration ID is used as a shared secret between the system A and the CA. However, knowing the registration ID is not enough to authenticate the session. Only if the system is recognised also as being configured correctly in step 3, will the session continue.

In step 4 above, additional information about the new host is recognised. The associated information is certified in 5, and then passed to the information system in 6. The authenticator and the security label of the host, issued at 5, will be used later at information maintenance — to authenticate the host, and the security label used to determine the security class of the host. Also, unauthorised information could be stored with the authenticator and the security label of the host.

Later in 7, the host system will receive the certificates and group keys such as the one for the Ethernet ARP operation (see Section 5.13 for the detailed operations.) The group operation keys are the appropriate encryption and decryption keys for the group operations such as address resolution protocol operation or routing information exchange operation. A group

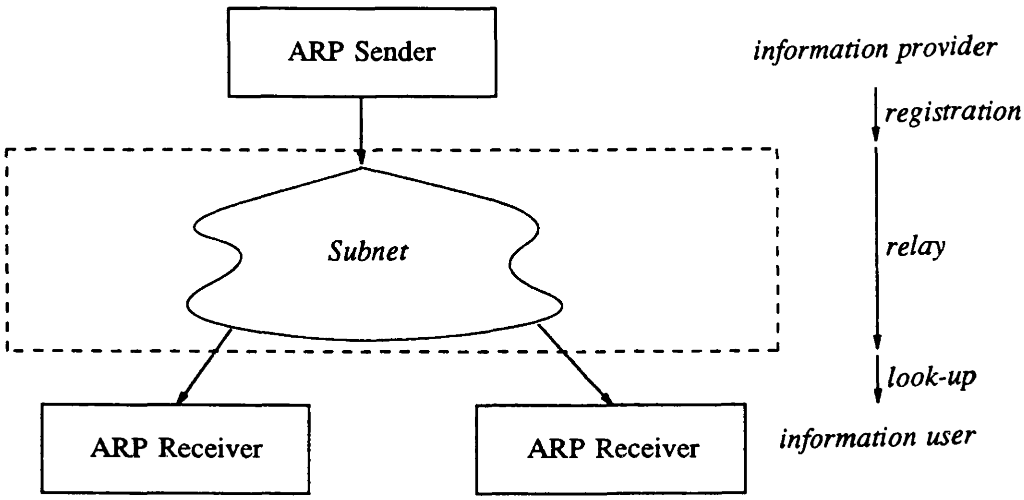


Figure 5.3: a view of the ARP system as an information system

key could be considered as a tool to tune the dissemination of information flow. One group key could be the one for the Ethernet ARP operations. We view the Ethernet ARP system as follows. The operation of sending out Ethernet ARP packets is a part of the *registration channel*; the information flow (6) in the configuration confirmation phase. The Ethernet ARP system is an information system. Receiving the ARP packets and learning from them is information look-up. Figure 5.3 shows our view.

5.4. Information maintenance

5.4.1. Overview

The *information system* is responsible for maintenance of information. Because of our particular interest in network addresses, we are concerned more with integrity than with confidentiality. Changes to objects should be detected and the associated information items have to be updated [MKM88]. An information update mechanism should maintain integrity of addresses.

In the previous section, we have presented a strong verification scheme for the addition of a host and its address at registration. At an *information system*, we apply a policy for the removal of a host. This could be done by a timeout mechanism, which requires subsequent re-registration. This method is employed by the current Ethernet ARP system, in which each host exercises timeout of the ARP cache entries. However, the removal of a host could cause inconsistency. During the period between when the host is removed and the timeout expiration, a host impersonating the authentic one could cause network level threats.

Another idea is to enforce the deletion of address associated with inactive hosts, which ensures a current status report from the host systems. The policy could be enforced by monitoring hosts and by regular reporting from hosts. In the following sections, we describe the detection of host removals and the deletion of the associated addresses.

In the next two subsections we describe the detection of changes and update of an address respectively used in our system.

5.4.2. Detection of changes

For detection of a change, a strong policy enforcement on objects is required to prevent a host from being removed or from modifying its address without being detected.

To detect host removal, hosts are monitored periodically. If the monitoring result reveals that a host system has not responded for a while, the removal should be verified. The verification could be done by sending a control message to the host to execute some test and send back the result. The verification messages could be sent through an alternative path if any exists. Using this scheme, it is expected that the network management server at each host system is the last application to be removed.

If the verification reveals that the host is not responding, the host system is considered down. If the host is down for some time, a drastic action should be taken to conclude that the host has been removed — unless a systems manager confirms that the host system is down temporarily for a certain reason. This *timeout rule* will encourage a host system to report its down situation. The systems managers would make best effort to report it by all possible means. They must be serious in reporting the scheduled down time, because otherwise the object might be isolated from the rest of the network by the removal of the registration.

Monitoring of host status can be done by sending out probes. The monitoring traffic can be reduced by watching the active network addresses, and sending a probe only to the addresses which have not appeared on the network for some specified interval. Alternatively the hosts can be set to send a spontaneous report on a regular basis — with active probing used only if no report is received.

Removal is confirmed using the verification result. When the removal is confirmed, the associated information will be updated as described in the next subsection.

The modification of an address could be reported by the host. The authorised address should be removed and a new registration is required for the new address to be authorised.

As a result of the above discussion, the following tools are required:

1. A reporting mechanism from a host to the manager
2. A monitoring mechanism
3. A verification mechanism for the manager to verify the host
4. A mechanism to synchronise the recognition of host changes and an update of the associated information

The authenticators and security labels of hosts are used in the above operations. See Appendix D for these detailed operations.

5.4.3. Update of an address

Update of an authorised address is examined after a change of a host or its address is detected. The update requires two steps: to invalidate the formerly registered information, and to add the new information. The CA confirms the removal of the object using an *Expired* message, which will be distributed to all relevant information systems. On receipt of the *Expired* message, an information system removes the associated information items. The expired message includes a time stamp to prevent replay. In the Ethernet ARP system, we can add a third packet type, the Expired packet, which could be sent by the management system.

We need the following mechanisms for host address maintenance:

1. Verification and confirmation of removal

2. A mechanism to distribute an Expired message to information systems which hold host information

The detailed operations are outlined in Appendix D.

5.5. Information look-up

5.5.1. Overview

Information look-up occurs when the *information user* retrieves host addresses from an *information system.* The look-up operation is concerned more with confidentiality than with integrity of the looked-up information in general. Note that the information maintenance operation explained in the previous section is more concerned with integrity. As we are trying to protect network addresses of the hosts within an organisation from threats such as unauthorised tampering, resource stealing, and network storms, we are concerned with the integrity of addresses. From this point of view, confidentiality of the addresses within an organisation would not matter very much in our scheme; observing the network traffic, one can see the available addresses, but we are not interested in its protection. However, in inter-organisational communication, some disclosure of host addresses is not preferred. In these cases, it is probable that the addresses of private hosts and routers may need to be hidden from the global set of subjects, but be visible only to the local subjects. For example, some sites are invisible to other sites on the Internet. Those sites include some MILNET sites, and various commercial organisations (e.g. most of those at Digital Equipment Corporation (DEC), IBM, and Sun Microsystems Inc. (SUN)) [Now90]. Alternatively, the management systems in different organisations which cooperate with each other for some management activities like fault isolation, might exchange local address information but may wish that the information is seen only by mutually trusted management systems. These situations lead to a need to consider the confidentiality of the addresses. In this section, we discuss some environments in which addresses require protection from unauthorised look-up attempts.

If a user, i.e. a host, is not authorised to look up a particular set of addresses, one has to make sure that it cannot access it directly or indirectly through any other intermediate routes of the information flow. Policy enforcement is required, so that an authorised user could not act as an intermediate agent for users who are not authorised for access to the addresses. This is control over address information dissemination.

A network address can be easily disseminated, because it appears as the source address of a network packet in inter-organisation network operation. In order to prevent such undesirable dissemination, we need a "fire wall" between the organisation and the outside world; by a fire wall, we mean a gateway which exercises access control over the traffic going through from one of its interfaces to another. We could exercise this on two levels — the network level and application level. At the network level, physical reachability of site can be controlled for each network level packet based on the source and destination addresses. A network level gateway at a border between an organisation and the outside could have a mapping table that maps a real network address to a fake address. The destination address of an incoming packet and the source address of an outgoing packet would be translated to the fake address, to hide the internal addresses of the organisation. Alternatively, an application level gateway could be used to communicate with the outside, so that only the network address of that gateway would be visible.

Extra care is required for network level access control. The error messages of network level operations often include the original network packet [Pos81b]. If an erroneous situation occurred for the incoming packet from the outside, the error reporting message would include a real address of the destination, and would be sent to the source host in the outside.

Access control requires authentication of a subject. Some addresses may have to be known only to hosts on a particular subnet. In this respect, authentication is required to identify which access control group a host belongs to. In our scheme, a group key given during configuration confirmation could be used for authentication of membership of a group. We can use the authenticator, security label and certificates obtained at registration to authenticate a given network host.

We need the following mechanisms for information look-up if the protection from unauthorised information disclosure is required:

1. authentication and access control; the user must be authenticated and its access to an information item should be controlled.
2. the maintenance of confidentiality; the information item should be visible by only the requester.

The detailed operations will be presented later in Appendix E of Formal Analysis.

5.6. Characteristics of the scheme

In our scheme, information verification is carried out during the registration over the network when a host starts the operation. In this way, an address is coupled tightly with the host. Authentication of the information provider, i.e. a host, is not necessary so long as we can believe the information item itself. The idea of a certificate came from this perspective.

A registration ID could be considered as a shared secret for the authentication of a host. However, it is our primary aim that the ID is used for the synchronisation of the registration. In other words, by showing the ID the host specifies which item it wants to register.

Our main goal has been to maintain the integrity of information, as the registration operation and information maintenance of information systems do not preserve confidentiality. However, when the information exchange should need confidentiality, each packet can be encrypted using the receiver's public key. Alternatively a specific key could be used to encrypt all the management transactions.

Our definition of a CA is somewhat different from that specified in the X.509 standard [CCI88a]. The CA in the proposed scheme is responsible for off-line registration of host information as well as the on-line process of registration over the network. The Certification Authority (CA) does issue certified information, similar to that described in X.509, though our CA does so only if an information item has been verified recently. This makes our CA more authoritative than the CA in X.509, because it means that the information certified by our CA has not only been certified by the CA, but also been verified. The detailed functions will be discussed later.

5.7. The state transition of a host

In this section, we observe the state transition of a host. States of network objects during a configuration procedure are discussed by ISO [ISO89d] and Sloman [Slo84]. The former draws a distinction between administrative states and operational states. The latter provides more detailed operational states. We simplified the above two and created our own model. Our state model has two administrative state classes and four operational state classes. Figure 5.4 shows our state transition diagram.

The administrative state classes are as follows:

- **Registered**: the information about a host has been registered with the Certification Authority (CA).
- **Not Registered**: the host is not registered; hence, it is not known to the CA.

Operational state classes are as follows:

- Disabled: the host is not configured.
- Off-line: the host is installed physically, however, the network interface has not be activated yet.
- Idle: the host is operational over the network.
- Active: the host is processing a request for a network operation.

Combining the above two types of classes, we have the following states:

- **Registered** Disabled: the host has not been configured yet in practice, but its configuration information is registered with the CA.
- **Registered** Off-line: the host is installed with its configuration information being registered with the CA.

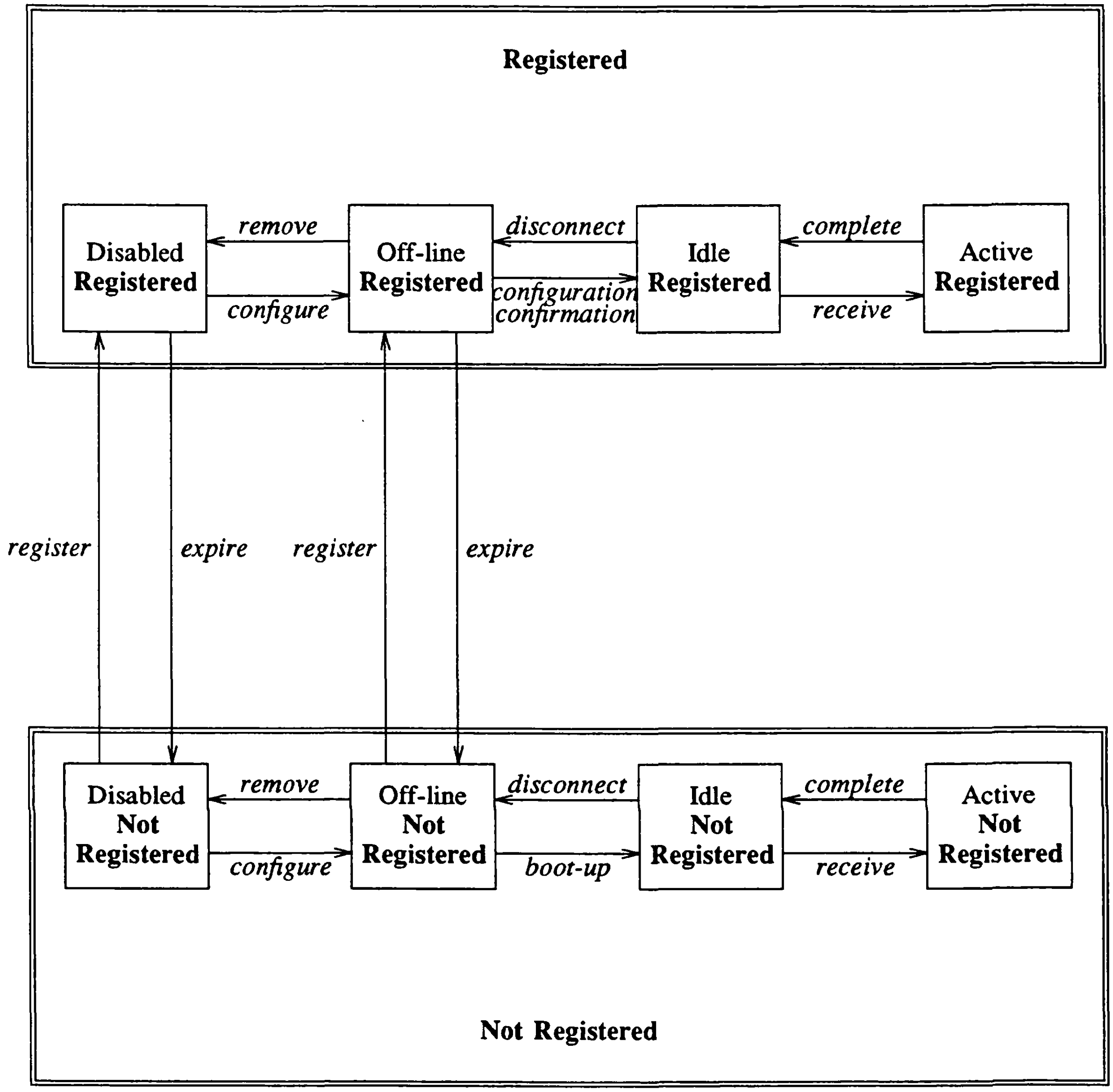


Figure 5.4: State transition diagram

- **Registered** Idle: the address of the host has been authorised by the CA and it is ready for a secure network operation.
- **Registered** Active: the host is dealing with a request for a secure network operation at the authorised address.
- **Not Registered** Disabled: the host is not configured, and its information is not registered by the CA.

- **Not Registered** Off-line: the host is installed, but its information is not registered by the CA.
- **Not Registered** Idle: the host is ready for insecure network operation.
- **Not Registered** Active: the host is dealing with a request for a insecure network operation.

Arrows in the figure indicate operations as follows:

- *register*: register the information about a host in the CA off-line
- *expire*: expire the validity of certified information about a host in the CA
- *configure*: install the subnetwork interface, the network software, and set the related parameters.
- *remove*: remove the host from the environment.
- *configuration confirmation*: this is done only at the very first boot-up. It verifies the configuration of the host and authorise its address.
- *boot-up*: the host starts operating with the configured parameters.
- *disconnect*: disconnect a host from the network. The host may be under going a maintenance operation, and will soon return to the network; in this case, the management system, M, will be informed of the scheduled disconnection, so that the host will not be recognised as removed and the associated information will not be expired.
- *request*: receive a request to send a network packet from a transport layer process or receive an incoming network packet from another host.
- *complete*: finish processing a packet.

The starting state of a host is Disabled and Not Registered. When the host has completed Resource Admission and Naming, it goes through Certificate Registration, and then the host will be in the Disabled and Registered state.

The host may in reality be configured before registration; the host's state will be Off-line and Not Registered. Later when it is registered by the CA, an extra configuration operation is required to give the host a Registration ID. Then the host will be in the Registered and Off-line state.

Although ISO's state model excludes the situation that a host is configured without admission from the manager, in an open network environment this is possible. If a host is configured without admission and registration, it will be in the state, Not Registered and Off-line. The host can be booted up and join the network, but it cannot participate in secure network operation because its address is not authorised. It goes into the state, Idle and Active, and its administrative state is always Not Registered.

A registered host will go through Configuration Confirmation when it is booted up for the first time. The host will be operational on the network, and be ready to process network packets; i.e. Idle and Registered. When it receives an incoming packet, it will go into the Active and Registered state. When it finishes processing the packet, it will go back into the Idle state again awaiting the next event.

Although a host appears as inactive when it is Idle, it is ready for network operations including management operations. In this state, the manager can send a probe and see if the host is operational as described earlier in the information maintenance section.

When a host is disconnected or removed, the host is regarded as removed. The associated information is expired, and the host will move to the Not Registered state; either Off-line or Disabled.

5.8. The proof of the address flow

We have examined the address flow in our scheme using the formal method introduced by Burrows, Abadi, and Needham [BAN89] in Appendix E. The examination showed that if the address information was certified, it would be delivered to the user without being tampered.

In order to make the maintenance of address integrity more trustworthy, the formal analysis suggested that the Naming Authority (NA) — the local manager who assigned the host address — should be informed by the CA during the host registration over the network that the host is announcing some address; let us call it X_a. Moreover, the NA has to reply to the CA confirming that X_a is indeed the allocated address for that host. This suggests that we could facilitate this with a specialised communication mechanism between the CA and NA. Alternatively, the latter can be done by letting the NA sign the address when it allocates and passes it to the Object Manager. The signature of the NA should be recognised by the off-line CA during Certification Registration. Then, during or after Configuration Confirmation, the CA can inform the NA of the completion of correct address configuration. This implies that the NA may not know at this point that the correct address has been indeed configured to the intended host; however, the NA will now be informed of it, eventually, by the CA. This issue highlights the large amount of trust placed upon the CA.

5.9. Time stamps

In a message exchange, there are two ways to ensure that a message cannot be replayed. One is use of nonce, and another uses time stamps. The word "nonce" means literally "used only once" and is used in the computer network security literature as a random number unique for each transaction so that it cannot be replayed. The counterpart of nonce sends back the reply which usually shows that the message is indeed the reply to the previous message. As the uniqueness of nonce is only known to the generator site, i.e. the initiator of the transaction, it requires more than the use of time stamps to ensure that the message was not sent at any time before the current exchange of messages. A time stamp can be known to both sites as unique and recently produced providing that the clocks at the two sites are synchronised. It can be done by the counterpart including another nonce generated by the counterpart in the reply as well as the initiator's original nonce; then the initiator of the first message sends back the third message containing the counterpart's nonce, which will assure the counterpart that the first message was sent recently.

Our protocol suite uses time stamps. We require each participant's clock to be synchronised to the other's. In the proposed scheme, time stamps are needed to ensure the freshness and uniqueness of each message. For such a requirement, the synchronisation algorithm does not need to maintain the absolute accuracy of each clock, but rather relative consistency [DeS81] [MaO85]. We need the synchronisation of a logical clock at each site [Lam78] instead of synchronisation of the physical clock [Mil89]; a logical clock is set to the time based on the consensus of participants of synchronisation, whilst the physical clock is set according to some standard time. We need temporal synchronisation of the manager site, the CA, and the information systems. A host could get the initial time value from the manager at the end of Configuration Confirmation; i.e. at the same time as a host receives its certificate. Afterwards, the hosts and the manager could periodically synchronise their clocks. Time synchronisation between these agents should use nonces rather than time stamps to ensure secure synchronisation, because this way we will avoid the paradox that time depends on security for authenticated time, and the security depends fundamentally on time [Ste91]; otherwise, by replaying stale messages, a deceiver could resynchronise time and thus replay other management transaction messages, causing an information system to accept, for example, a stale certificate.

5.10. Required functions

In the earlier sections, we described the procedure for trustworthy information flow by introducing two agents, the management system and the CA together with the information system. In this section, we decompose the functionality of these agents. In the next two sections we discuss in detail the management system functions and the CA functions respectively.

5.10.1. The management system functions

The management system is responsible for the procedures of registration over the network and for information maintenance. The following functions are required by the management system;

- *Function 1:* detecting configuration changes by monitoring or receiving the report from a host.
- *Function 2:* verifying the host configuration
- *Function 3:* controlling a host remotely or locally.
- *Function 4:* requesting the notarisation of a host address upon successful

verification.

- *Function 5:* disseminating information items to the relevant information systems when a host status changes.

Among the above functions, the verification mechanism for network-level configuration needs to be distributed to each subnetwork, i.e. each Ethernet. Moreover, such a verification mechanism should have the capability of logging each layer operation for an incoming packet from a specified host, so that the configuration of the host can be verified by examining its layer operability. The other functions can reside anywhere in a management domain; they could reside in one system for the aggregation of control.

The following information is required by the management system to perform the above functions:

- the public keys and locations of the CAs and the information types over which each CA has authority.
- a verification tool to be used for each host type
- a set of operations it can exercise on the hosts, hosts' configurable parameters, and valid values of the parameters
- a table of configured hosts, and the information systems which hold the information items about those hosts
- a table of information systems, their locations, and their associated information types

5.10.2. The CA functions

The CA is responsible for authorisation procedures. First it deals with the off-line registration of a host; i.e. Certification Registration. Second it authorises the address of the host over the network. The following functions are required in the CA:

1. to maintain the registration IDs and the associated host information whose source was off-line.
2. to provide the registered information to the trusted manager.

3. to notarise the address once its association to the right host is verified by the trusted manager.
4. to interface with the Naming Authority

The following information is required by the CA:

- a table of a registration ID, the associated object name, the attributes of the object, the location, whether it is configured or not, and whether it is notarised or not
- a list of information types which it can notarise

The list of configured object could be maintained dynamically by the CA on receipt of the verification report from the management system at the Configuration Confirmation.

The CA is responsible for the management of registration IDs. It has a knowledge of the association between a registration ID and the information item which is to be registered. The management system knows over what information items the CA has an authority. Conversely the CA knows over what information it is authoritative for. The host sites and the management system believe that the Naming Authority has been or will be informed by the CA of the fact that a host is announcing the allocated name when the CA has notarised the binding.

5.11. Architectural issues

In this section, we examine how the functions of the management system identified in Section 5.10.1 could be distributed over networks in an organisation.

The detection mechanism (Function 1) should be replicated, so that a host change can be detected reliably. The verification mechanism (Function 2) should be replicated in each subnet, since the verification needs the subnet level operations. Other functions can be either centralised or distributed.

One of the decentralised approaches to accommodate those functions could be as follows. The *detector* D is responsible for detecting configuration changes, as well as for verification of the host configuration. The *controller* C is responsible for controlling objects remotely or locally. The *information manager* IM is responsible for dissemination of updates of information items to the information systems. The detector function should be replicated in each subnet, because it exercises the verification at the subnet level.

We described the Certification Authority (CA) as an agent. However, it may well be that there are more than one, each of which notarises a different type of information — e.g. one for the Ethernet and IP address mappings, and another for authenticators and security labels.

Therefore, the configuration of a decentralised manager system may include several detectors, each with the controller and the information manager, and some with the CAs and information systems.

5.12. Viability of our scheme

5.12.1. Overview

Our scheme introduced earlier in this chapter assumes that the functions of the management system and the CA are all operational. When those functions are decentralised and distributed over the network, communication failures result in a reachability problem between subsystems, hence, the subsystem's functions may not be available.

For fault-tolerance, a conventional solution is to keep redundant resources [SiM90]. Another solution is to provide a way in which the proposed trustworthy information system could work with a partial set of agents, possibly for a limited duration of time.

There are two kinds of redundancy; one kind replicates functions at different locations, the other requires redundant paths between subsystems. The former needs the management of replication, but those replicated subsystems would have a group (i.e. multicast) address over the network, so that path control would be done through the normal network operation. The latter needs control of the routing path between agents, but it does not need replication management. A preferred scheme may include both.

Another approach towards fault-tolerance would be to implement some adaptability in the information system, so that it would operate without some of the functions. The aim of our scheme was to identify the necessary functional agents which make the network information trustworthy; the absence of any of them would leave dubious information. Adaptability allows some dubious information to remain for a while.

In the following subsections, we discuss the viability of our scheme with respect to providing redundancy and adaptability. The next subsection presents the partial systems beginning with the minimum functions required for operation. Sections 5.12.3 and 5.12.4 examine applicability to the UCL-CS and the Internet respectively.

5.12.2. Partial systems

In this section, we examine the operation of partial systems in which only limited functions are available. First we define the minimum system; then we discuss a partial system which has the minimum system functions with the CA. Third we describe the minimum system with the verification function and the CA.

5.12.2.1. The minimum system

In this section, we define the minimum system. For a specified limit of time this system allows a host to register its address without confirmation. Addresses registered during this periods are dubious. No certificate will be produced, because no verification takes place.

A modified registration scheme for viability is as follows. At configuration confirmation, if any function of the management system is not available, the host registration stops. The consequence is that either a host cannot start registering, as the port of the management system which receives the registration request from a host is not available, or the on-going registration is never completed. Therefore, we need a timeout mechanism at the registering site. After the specified duration of time has elapsed, the host site can go into dubious mode. In this mode, the site can report to another port of the management system and pass a limited number of information items which are stored in the relevant information systems. The management system accepts such information items, and examines whether they will conflict with any of the registered information. It is this verification of inconsistency, which makes the raw information not fully dubious. For this purpose, the management system has to keep a list of configured hosts and their registered information; the partially registered information itself is stored in a certain information system. If there is no conflict, the management system passes the list to the relevant information systems, attaching the "dubious" mark and the recommended timeout value. It is up to an individual information system to decide whether it accepts the dubious information, or discards it. Some of them might use the recommended timeout value for the information.

In this modified scheme, the minimum that a system requires is a redundant port or path to receive a report from a host (Function 1 in Section 5.10.1), and a function to disseminate information items to the relevant information systems (Function 5 in Section 5.10.1). By excluding the CA from the minimum system configuration, we allow both the authentic site which might have been misconfigured, and the maliciously spoofing sites to start operating for a while. The individual information systems and their users have the final decision on whether to take such a risk.

For the sake of the minimum system operation, we make the hierarchy of the information items clear. Certificates are the most trusted way to rank information. The second rank is done by using the information registered with the authenticators and the security labels. The address registered without any of them in the emergency registration above, is least trusted; we call it a *raw* address. When the trustworthy information system has a problem, and a minimum system configuration is being used, the information items claimed by the host sites are either in the second rank or raw. An information item can be replaced only by another at the same or higher rank. For instance, if a network address X_a has been certified to a host at a site A, any attempt to change it is impossible with the minimum system configuration, because neither authorised Expired message nor new certificates can be produced. Hence, no one can expire the certified information in the minimum system. If X_a is declared with an authenticator, then it can be replaced with another address with the same authenticator. If X_a is declared as raw, anything can replace it.

As authorisation cannot take place in the minimum system configuration, the certificates can be expired only by timeouts. When a timeout of an authenticator is performed, the associated information items, which have been registered with the authenticator should be deleted as well. However, as long as the authenticator is valid, the information items without notarisation can be modified.

Those ranks of credibility can be enforced in terms of timeouts as well; i.e. the certificates have longer timeouts than the information items with an authenticator and a security label. Second rank information items have longer timeouts than the raw information. The information with an authenticator and a security label have timeouts less than a day. The timeout of the raw information may be twenty minutes. Those timeouts can be recommended by the management system before the information is distributed to the information systems. The information systems may set its own timeouts, perhaps based on those recommended timeouts.

5.12.2.2. The minimum system with the CA

If the CA is available with the minimum system configuration described in the previous section, at least the cryptographic attributes of the site, such as the public key, would be passed to the management system for authentication of the host site; in this case, only the authenticated sites can be information providers. However, even if the site was authenticated, it would not be certain whether the site was configured correctly because the verification function is not available. Hence, the credibility of the host address information passed to the information manager would still be uncertain. If there was an off-line report that a host is removed, the CA could issue an Expired message about the host information.

5.12.2.3. The minimum system with the verification function and the CA

In the partial system, the verification of the host configuration can be carried out together with the host information which was registered previously with the CA off-line. Consequently the CA issues certificates. However, without the control function which sets the host parameters (Function 3 in Section 5.10.1), a host can have neither certificates nor network operation keys. Perhaps, the management system could act as a publisher — an agent who answers the ARP request for other hosts; i.e. the management system replies to the ARP request for this host. In this case, the management system should have representative subsystems on each subnet, so that each subsystem can join the ARP operation. This subsystem could be used to disseminate address information (Function 5 in Section 5.10.1) to the ARP system.

This partial system can produce Expired messages, so that information maintenance is performed.

5.12.3. UCL-CS

In the UCL-CS domain, there are 3 subnetworks; the main subnetwork has about 100 hosts over three Ethernets connected by two MAC level bridges. Another has 12 hosts, and the third one has 22 hosts. The Internet IP protocol is running over the Ethernet Protocol. In this environment, a subnetwork is one or more Ethernets connected by one or more bridges, which participate in the address resolution operation which translates Internet IP addresses into Ethernet addresses.

There are three uses for our computer networks, viz. teaching, research, and administrative purposes. In such environment, the use of sharing of limited resources to achieve reasonable performance is more important than protection of resources. However, some protection is provided for students.

The machines for undergraduate students are located on a separate subnetwork from other networks and are connected to other networks via a subnetwork level router. The router has no entry for foreign network addresses outside our department, so that no student can have network level access to those sites. The hosts used by lecturers, as well as the ones for research and administration, were once located on the same subnetwork. A distributed file system, the Network File Store (NFS), is used so that the users can access their files from any host to which they have access. Authentication of the users by each host is done by passwords and their access to hosts is controlled; an exception is personal computers which can be accessed by anyone.

For the purpose of teaching, sharing resources and information outweighs protection; a lecturer may prepare assignments in a file at his directory to be read by students. The same file server is used by lecturers and students. A student can look at the files managed by the lecturers if file permissions permit. Since we are using the UNIX operating system, protection of each file is controlled by the owner of the directory on which the file resides. There was an incident in which an examination paper was stored in a file, which a lecturer left accessible by his students. This could have been avoided if lecturers had another file server which was not accessible by students. However, while the file server is being accessed over the subnetwork, one can see the traffic which is sent in a clear text. Using a personal computer on the subnetwork, students can see the traffic on a subnetwork even if they have no access to that subnetwork. Although this means is tiresome, it allows one to use one's machine as explained, as "unauthorised access to a host" in Chapter 4. Attempts to watch traffic have been prevented partly because students have not enough knowledge of networking and systems to do so. Nevertheless, nowadays examination papers are prepared totally off-line — networked office automation is not permitted.

Information on research projects or administration is also under the similar threats. In this case, people are more knowledgeable about networking and systems. However, since the sharing of the limited resources and information as well as the provision of reasonable performance is vital for research, no application and network level protection have been provided. Because research hosts are heterogeneous, and are used for experimental purposes, the management of the subnetwork may encounter various kind of failures. For example, a project sought to implement DEC protocols on a SUN workstation. This development required the Ethernet address of the SUN to be changed into one whose manufacturer's part indicated DEC. Luckily enough, the new address did not conflict with the other DEC hosts in the department. In spite of those unpredictable experimental situations which could lead to a fatal situation such as a network storm, the network manager is required to continue providing network service.

Administrative information requires confidentiality; however, the hosts on which administration is carried out, are on the same subnetwork as many research hosts. The situation above, in which research work and administrative information maintenance share the same subnetwork without protection, is not desirable.

A solution could be implemented as follows. For administrative work including the preparation of examination papers, application level protection could be provided, i.e. authentication and encryption for each transaction. However, a network level attack, that would delay administrative work by redirection and dropping of packets, is still possible. Moreover, the manager of the research subnetwork wishes to avoid misconfiguration which could cause a fatal impact on networking. On the other hand researchers wish to avoid the heavy overhead needed for protection. We can modify our scheme, so that only routers are certified objects and in the ARP operation, each host holds two tables, viz. one table for certified ARP information and the other for non-certified information; during address resolution, an IP process first looks up the former, and if there is no entry for the IP address in question, it looks up the latter. However, all hosts go through a registration scheme when they are configured for the first time which allows misconfiguration to be detected. Only routers are given certificates. They use their certificate for the ARP operation so that the overhead of maintenance of certificates and secure ARP operation can be minimised. The timeout of a certified ARP entry could be longer such as several hours, so that they are only removed either due to timeout or by the Expired message. In this way, the secure ARP operation at each site would not be executed frequently. Routers should have a sanity check to prevent an ARP entry which contains the broadcast subnet address, but this is not compulsory — let us believe that people are not so antisocial as to cause a network storm by a chain reaction; note that a chain reaction due to misconfiguration would be prevented by the verification during registration over the network. Moreover, if it was so desired that the hosts used for administrative work should be hidden completely from the rest of the network, then they could be located on a separate subnetwork with a network level router serving as a fire wall. Within the subnetwork, our original secure ARP operation is carried out, so that only certified ARP information is accepted; i.e. all legal hosts have certificates. From the architectural point of view, we need Functions 1 and 2, detection of a change and verification, in each subnet. As our definition of a subnetwork is a network-link address resolution domain, bridged Ethernets are considered as one subnetwork. Therefore three subsystems are needed in order to carry out Functions 1 and 2, each of which is located in each subnet. In case of bridged Ethernets, it is very difficult to verify on what Ethernet a host resides. This is because the bridges act transparently to network operations, and an Ethernet address may not relate to a location at all [DaP81]. To verify the location of the host more precisely, an additional feature is needed. Such a feature could be either a verification mechanism placed on each Ethernet, each of which does link level verification of nodes, or more intelligent bridges, which a management system can query the nodes reachable through each port of a bridge.

Other management functions and the CA can be located anywhere. They can be placed on one host, or spread over the network. Alternatively, they can be placed on a private Ethernet, which could be attached to one of the subnetworks by a bridge or a network router. This isolates most of the intra-manager communication. The management system and the CA could have information used for management operations, located locally. Alternatively, it can be placed in a common information base.

If any manager function were not available, the ARP system would operate with a partial system; by a partial system, we mean a system with some but not all of the required functions of our authorisation system. If it is operating with the minimum system or the minimum system with the CA, the new certificate is not produced. A newly installed router has to join the ARP operation with raw information; it advertises its Internet IP and Ethernet addresses without any certificate. An existing router which uses its certificate and wants to change its Internet IP address, can advertise the mapping of new address and its existing Ethernet address. In this case, the old certified mapping remains in the ARP table at each site because no Expired message is issued and the certified mapping has a longer timeout. The new mapping can be introduced and accepted into the table of non-certified ARP entries. There is an implication in this case that because the old certified entry remains, it will be possible that a packet addressed to the old IP address is delivered to the router. Then the router had better accept it, otherwise, there would be a deceit.

Because no verification can take place with the minimum system, there must be a constraint that a new host or router can have only an address, which has not been assigned to any other host. This maintains the consistency of the addresses; i.e. no illegal address can appear on the network. This can be done by a new host or router that listens to the traffic for a while and learns the existing IP and Ethernet mappings. It then starts operation if and only if there are no conflict with its address.

If a minimum system with a verification function and the CA are available, certificates can be produced, however, a router cannot receive the certificate when the control function of the management system is not available. In this case, the manager subsystem could act as a publisher for the router, and disseminate the certified address resolution packet. A new host are verified and it does not need a certificate, so that it can start operation.

5.12.4. The Internet

In the Internet, the gateways are connected by subnetworks to form a global internetwork in a hierarchical way; we use the word "gateway" instead of router in this subsection, because those routers are likely to connect different type of subnetworks, and so are traditionally called gateway. There is a backbone network of the core gateways, which is considered as an autonomous system itself. Other autonomous systems, such as campus-wide internetwork and an institute-wide internetwork, are connected to a core gateway through a stub gateway at each site. By the Internet, we include the operations both in the intra-autonomous system, and in the inter-autonomous systems; i.e. the operation between the core gateways, and the operation between the core gateways and the stub gateways.

The Internet includes various types of networks. The end systems in different autonomous systems should understand that the link between them cannot necessarily be trusted; unauthorised tampering, masquerading, denial of services, and unauthorised disclosure of information are all possible. Using end-to-end measures at the application level, such as authentication, access control, encryption, or digital signature, the communication between end sites becomes more secure. However, the possible network level threats, such as unauthorised use of network resources, traffic generation, and unauthorised redirection are still possible. Indeed, compared to the intra-organisational networks, an inter-organisational network is more concerned with the network level threats, because it is this level of service that the network offers. Note that in the inter-organisational network, we do not discuss disclosure of network addresses, because this is only of concern to particular organisations, and it should be solved within those domains — perhaps by providing the translation of internal network addresses into fake addresses. Among the threats, unauthorised use of network resources and redirection of packets are serious to the inter-organisational network. Traffic generation is difficult to stop in the Internet, because it needs higher fire walls than the network level gateways.

It is difficult to stop traffic generation, because the packets would have legal addresses — only the detection of the network, from which the packets are originated, would be possible. Even if there is authentication and access control for each packet, by replaying the legal packets within an organisation or in the inter-organisational network, one can easily generate traffic; one would argue that each gateway checks the packet ID, however, those gateways are basically state-less, i.e. they have no record on which packets they forwarded. What is required would be the higher level gateways, so that the network level traffic would be stopped at that gateway.

Unlike intra-organisational local area networks, the Internet includes wide area networks, some of whose costs are charged. The network operation is based on agreements between member organisations. In such an environment, unauthorised use of the network is most undesirable. The access control in the Internet is not required per end system but per network or per organisation, i.e. per autonomous system. Authentication of the source network or organisation for each packet would be required for access control.

Unauthorised redirection of packets can be achieved by intercepting routing information exchange protocol operations either between intra-autonomous systems or between inter-autonomous systems. Interception can be done either physically, by cutting a link between gateways and inserting a deceiver's own system, or by misuse of the routing information exchange protocols. The former could be applied to a point-to-point link; the deceiver's system may bypass the traffic between authentic gateways and may drop some packets which make particular networks unreachable. Moreover the deceiver can introduce new networks by attaching this information to the authentic routing information exchange protocol packets; later routing to those new sites is done by the deceiver's system — this is unauthorised use of network resources. The deceiver's new site can communicate with its counterpart in a remote location over the network which the site is not authorised for use.

In the latter case, i.e. misuse of the routing information exchange protocols, a deceiver can misuse either one of two such protocols: one for the gateways, another for the gateways of the subnetwork which connects those gateways. For example, it can be achieved by making use of the neighbour acquisition part of the Exterior Gateway Protocol (EGP), introducing a new bogus gateway. The bogus gateway will advertise the networks, which have been advertised by the authentic stub gateway, with less distance, so that the packets to those networks will be redirected to the bogus gateway; it may forward some of them to the authentic stub gateway, changing the source addresses of the packets into its own address, so that the return traffic, if any, will come back to the bogus gateway. Alternatively the intercepted packets may be dropped or delayed causing denial of services. Moreover, by misuse of intra-autonomous routing protocol operations, a deceiver can inject false network reachability information via a bogus gateway to a stub gateway which, in turn, advertise this information to core gateways; this would cause a convergence of network traffic to that site. This could also happen if a stub gateway malfunctions and thinks that it could be reachable to any network with a very short distance.

The neighbour core gateways for each core gateway have been specified manually — it may be done remotely by making use of a management protocol, so that a deceit is almost impossible, providing the management operations are protected from tampering. The neighbour core and stub gateways might be configured manually as well, although EGP does not specify this and dynamic neighbour acquisition would still be possible.

Unauthorised use of network resources could be solved by applying our registration scheme for each organisation. Then an organisation administers each packet going out of its domain, perhaps by giving a ticket or signing the packet. In the Internet each gateway checks the packet's authenticity compared to the list of authorised organisations or networks. This is the approach taken by the Visa scheme [Est85], and policy routing [Cla89]. Our solution to the two threats is to provide a rigorous routing information exchange, so that gateway would never learn an unauthorised site. The solution still allows unauthorised sites to send packets to authorised sites; however, because the reply to the unauthorised site can never be delivered back, it would not be terribly useful for unauthorised sites. Each packet does not have to be authenticated, so that the work at each gateway is lighter than other schemes which require each gateway to check each packet. This is a similar approach taken by the NSFNET [Bra9], however, their scheme does not allow the stub gateway to introduce a new site — i.e. with no dynamic learning; instead, the EGP is used only for reachability check of authorised networks. Our scheme is to assign an authorised list of possible networks; a gateway can introduce only networks which it is allowed to introduce. We introduce our solution by applying our trustworthy address flow scheme in the following.

We try to protect the Internet from false routing information, and incorporate a protection mechanism within a routing exchange between core gateways as well as in the one between a core and a stub. In applying our scheme, we presume that the Certification Authority and the management system belong to the autonomous system of core gateways, i.e. the core system.

The core gateways are located over the subnetworks within the core system — i.e. the backbone network. The verification function of the management system could be distributed to each subnet. The other management functions can reside anywhere within the core system. When a core gateway is configured to a subnetwork, it will be verified and given certificates, i.e. it follows our configuration confirmation scheme. The certificates include authenticators and another used for for network routing information exchange. An authenticator of the core gateway is issued for each of its subnetwork interface, and would include an Internet IP and subnetwork level address pair and the public key of this gateway. Another type of certificate includes the Internet IP address of a core gateway and a list of network addresses, which it can possibly advertise to the neighbour core gateways. One of the former would be used for the communication with a stub gateway later.

At the routing information exchange between core gateways, each core attaches its authenticator and the certified list of networks which it can possibly announce to the protocol packet which is signed by the gateway's secret key. The certified list of networks can control the routing from the administrative point of view, because even if a network is reachable through more than one core gateway, only one of them with the certificate can advertise.

For the inter-autonomous system operations, i.e. the routing information exchange between a core and a stub, the core will show its authenticator and sign each packet with its secret key. A stub will show both its authenticator and the certified list of networks which it can possibly announce. The management system would like to verify the stub when it starts, but it may be difficult as the subnetwork between the core and the stub may well belong to the stub's autonomous system. It can be verified only relatively through the core possibly by making use of the record route facility of the Internet Protocol gateways [BrP87]. At the Configuration Confirmation Phase of the registration of the stub, the manager subsystem sends an enquiry-on-configuration packet to the stub asking to return the reply with the record route option, so that the manager subsystem can be sure that the stub is reachable through the particular core gateway; the core gateway could sign the reply packet with its secret key.

When a stub gateway starts operating, it has to report to the management system in the core system through the core, and configuration confirmation is carried out; same as a core gateway, it will be given two certificates, an authenticator and the certified list of networks. An authenticator, in this case, would include the Internet IP address of the stub, the number of the autonomous system, which it represents, and the associated core gateway. The other certificate includes the autonomous system number, and a list of network numbers, which can possibly be advertised by the stub gateway. Using the authenticator in neighbour acquisition, the core gateway understands that the stub is an authorised one. Each routing information packet could be signed by the stub's secret key and accompanied by both certificates. Alternatively the certified list of networks possibly announced by the stub could be kept at the associated core, and the stub can send anything in a clear text; sanity checks can take place at the core gateway to filter out unlikely information.

In the presence of a communication failure, the above operation has to be performed with a partial system. If it is the minimum system, a newly installed core gateway cannot get any certificate, and it can advertise only raw information to peer core gateways for a while. As the second type of certificate, which shows what networks the gateway can advertise, can be only assigned by the CA, which is not a part of the minimum system, the new gateways may advertise anything. Each peer has to verify the received raw information with the entries of the routing table, which are with the certificates. If any conflict is detected, the information is discarded. If not, the information is accepted with a short timeout. The existing gateways cannot change the certified capability; it cannot advertise an unexpected network.

Under the minimum system, for a new stub gateway, the associated core gateway may not trust such a gateway, however, it may believe the reachable information from the stub to a certain extent. This is because a certified list of networks which the core can advertise to other cores is assigned to the core by the CA, and the certificate shows what networks the core would hear from the stub as reachable sites; i.e. the certified list tells the core the possibility of incoming information. Therefore if the core receives the raw reachability information from the stub, which includes such possibly networks, it would accept but with a short timeout.

With the minimum system with the verification function and the CA, new gateways can be verified and the CA issues the certificates, although they cannot receive the certificates. The communication failure is likely within the management system, so that neither a core nor a stub can receive anything from the management system. Therefore, the core and stub gateways operate in an insecure mode as in the previous case above.

In both cases, the second type of certificate for control over advertisement of reachable networks, is not verified information, but assigned information. When it is used, it acts as a filter. We can see a similar example in the NSFNET [Bra88]. Its internet hierarchy has three layers; the backbone (a core system), the regional (a stub gateway domain), and the campus networks. When a new campus network wants to join, it has to have a bilateral agreement with one or more of the regional networks, which will represent it. The agreement is passed to the backbone, so that the backbone can maintain the policy database [Rek89]. Those transactions take place all off-line. At each receipt of network reachability from the regional network, the core system validates it with the policy database. No verification of the configuration of a gateway is exercised.

There is one drawback of this scheme. It is not easy to verify the existence of a newly introduced network, where a stub's autonomous system exercises a strong access control over the dissemination of its network addresses; i.e. the translation of real addresses into fake addresses takes place at the gateway. In this case, it is required to have a management level gateway in each autonomous system, which communicates one with another to exchange information for verification.

Whatever verification methods would be taken, the verified information would be distributed not only to the gateways, but to the appropriate information system such as the Domain Name System. Then another problem arises. The current DARPA portion of the Internet has a separate network information center, which is another management domain for the registered information. We think that it is this arrangement that makes the Internet operation distrustful; i.e. a network host can start operating without registration. If the network operation center and the network information center belong to one domain, the life would be easier, because the verification of the network host done by the operation center, can affect directly the registered information base at the network information center. This has been practiced in the NSFNET, which learned the lessons from the experiences of the DARPA Internet.

5.13. In case of malfunction

The malfunction of either the management system or the CA would be detected by the other. If the management system malfunctions, it would report the CA about the confirmed configuration of a different address of the host. In this case, the CA would detect it, because it compares the reported address and the address which was registered off-line. On the other hand, if the CA issued a certificate of an incorrect address, the management system could check it and detect the incorrectness before it would distribute to the host and the information systems.

If both of them malfunction, then the host would detect the incorrectness of the certificate, and should report to the network manager.

5.14. Application to the current Ethernet ARP system

5.14.1. Overview

In this section we show how in practice we can incorporate the ideas of trustworthy information flow to the current Ethernet ARP system.

5.14.2. The current Ethernet ARP system

The Ethernet Address Resolution Protocol (ARP) [Plu82] was originally created for the Chaosnet environment [Moo81] to resolve dynamically the internet address to the subnet address [Pat89].

The protocol is connection-less, and consists of two types of packets: one for request and another for reply. When an internet packet has to be sent out, the internet protocol operation process looks in the internet and subnet address resolution table for the internet address of the destination. If the search fails, the host will broadcast an ARP Request packet over the subnet specifying its own address mapping as a source as well as the destination's internet address. The appropriate host answers with an ARP Reply packet which is unicast to the requester. While not included in the specification of the protocol, in practice all the hosts in the subnet learn the requester's internet and subnet address combination by the broadcast ARP Request packet; however, only the target host whose Ethernet address was requested would create a new entry for the newly learned combination, and others would update the entry if a host has the entry for the requester. This way, the number of transactions is reduced. It is this dynamic learning which caused the fact that any host can start operation with any internet address.

5.14.3. Modification to the current system

Notarisation of source address by the CA uses a *public key system* for integrity [DiH76], [RSA78], [NeS78], [Hel87]; a secret key is used for encryption, and an associated public key for decryption.

A certificate contains the Ethernet and Internet IP addresses; it is time-stamped and encrypted by the CA's secret key. A *certificate* of the sender is attached to an ordinary ARP packet.

The ARP packet is further encrypted by a group key, the ARP operation key for a particular subnet.

Our ARP packet is composed of the following fields; the bold-faced parts are additional to the original packets:

- (N-1)-level subsystem type (e.g. Ethernet Protocol)
- (N)-level subsystem type (e.g. the Internet Protocol)
- the length of an (N-1) address (e.g. 6 octets for an Ethernet address)
- the length of (N) address (e.g. 4 octets for an Internet IP address)
- operation type (e.g. request or reply)
- **the group type** (e.g. a subnet ID)
- the (N-1) address of the sender
- the (N) address of the sender
- the (N-1) address of the target
- the (N) address of the target
- **the ID of the Certificate Authority (CA)**
- **a certificate signed by the CA with the following contents:**
    - **the (N-1) address of the sender**
    - **the (N) address of the sender**
    - **time stamp of the CA**

Note that the (N-1) address field of the target in an ARP Request packet is not checked. For the Ethernet ARP, an ARP packet is packed in an Ethernet frame. An ARP Request packet is broadcast and its (N) and (N-1) addresses may be learned by the other hosts on the network. An ARP Reply packet is sent to the requester and the sender's (N) and (N-1) addresses are learned only by the requester.

The address mapping information learned from the ARP operation would be saved in the host's cache, either as a certificate or a decrypted clear text form depending on the security requirements of the host. Moreover, a secret key for the ARP operation is shared by all the members of a group. A group is one address resolution domain which would be one subnet. In this way, a host who does not know the secret key cannot join the ARP operation.

We could use a private key system for this as well; however, using the public key system seems more flexible for future applications. For instance, the secret key which we use in the ARP, could be used for other network operations than the address resolution, to ensure that a packet comes from a certain subnet.

5.14.4. The implementation of an enhanced Ethernet ARP

5.14.4.1. Overview

The purpose of the implementation was to see how the idea of modification of the current Ethernet ARP will work. We have implemented a sender and multiple receivers of the ARP packets. Because the Ethernet ARP codes are not accessible to the users, the ARP operation was emulated on the higher level. The Ethernet operation was emulated at the User Datagram Protocol (UDP) layer over the Internet IP subnet broadcast. Our ARP used a specific UDP port. Figure 5.5 shows this.

Three types of hosts were used for this experiment. One was a sender of an *ARP Request* and another received and learned the information. The third type of hosts learned by listening the *ARP Request* with no or partial ARP cryptographic system information.

RSA was used for both certificate and the ARP operation. The ARP operation could use DES, however, for future use I decided to use the public key cryptosystem. In some application, it could be useful to show that a packet is indeed from a certain group of the network.

5.14.4.2. The ARP Procedure

First we assume that the certificate for a host has been generated by the certification authority and given to a host at the registration or boot time.

An address resolution requester constructs an ARP request packet with the certificate of its address information; i.e. its IP and Ethernet addresses. From the first field to the group

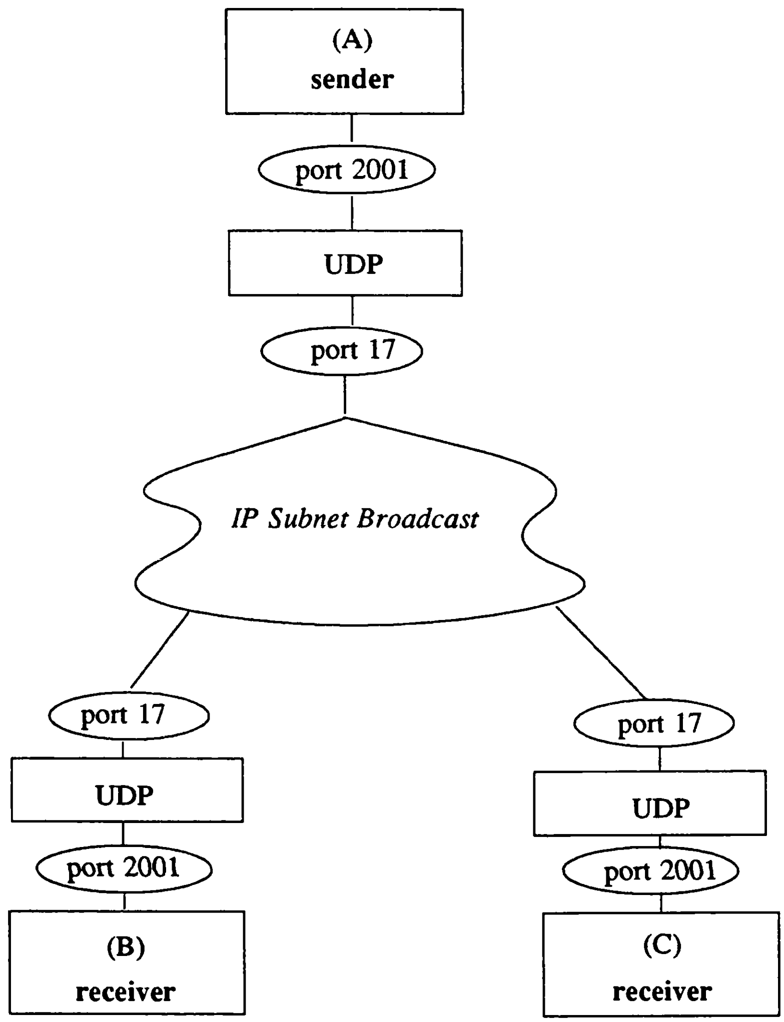


Figure 5.5: The experimental environment

ID field are clear text. The rest of the packet is encrypted by the secret key for the ARP operation group — e.g. the subnet operation key or the key for the particular ARP group operation.

A receiver examines the operation code. If it supports, the ARP Request with a Certificate, it proceeds in order to see the group ID. If it has the knowledge of both cryptosystem the group uses and the group key, it can decrypt the rest of the packet. Then the receiver learns the sender's address mapping. With the group key we can obtain somewhat loose data integrity.

If the receiver is concerned with the accuracy of information, it further decrypts the certificate with the certification authority's public key. The decrypted addresses can be compared with the sender's addresses. If they match, the address mapping information about the sender will be stored in the ARP table.

Another possibility is not to encrypt with the group key at all. The receiver would be only concerned with the information in the certificate. In this case, the first sender's address fields in the packet would not be used by the receiver for learning.

5.14.4.3. The packet size

An original ARP packet is 28-byte long. Our scheme adds more fields such as **subnet ID** and **certificate**. In our implementation, the length of the subnet ID field is set to 8 bytes.

The certificate of the sender contains the CA's ID, the 4-byte Internet IP address, the 6-byte Ethernet address, and the 14-byte time stamp as a clear text. As a result of the encryption by the CA, it becomes 64-byte long.

Thus our ARP packet size before the encryption with the ARP key becomes 100 bytes. The 100-byte packet is further encrypted and finally becomes 144-byte long as explained in the next subsection.

5.14.4.4. Encryption and decryption

The RSA cryptosystem is used for both ARP packet encryption and certificate production.

For the certificate, our application provides information integrity. Thus the information is encrypted by the CA's secret key and passed to a host at the configuration confirmation phase. It is decrypted by the CA's public key at each receiver of the ARP packets. For the ARP packets, we also implemented a facility to maintain integrity. In other words, a packet is encrypted with the ARP secret key at the sender and decrypted with the ARP public key at the receiver.

The encryption procedure is as follows. An original ARP packet is divided into the header part and the rest. The header is kept as a clear text and the rest of the data is divided into 64-byte data blocks with padding of zeros where appropriate. Encryption is carried out for

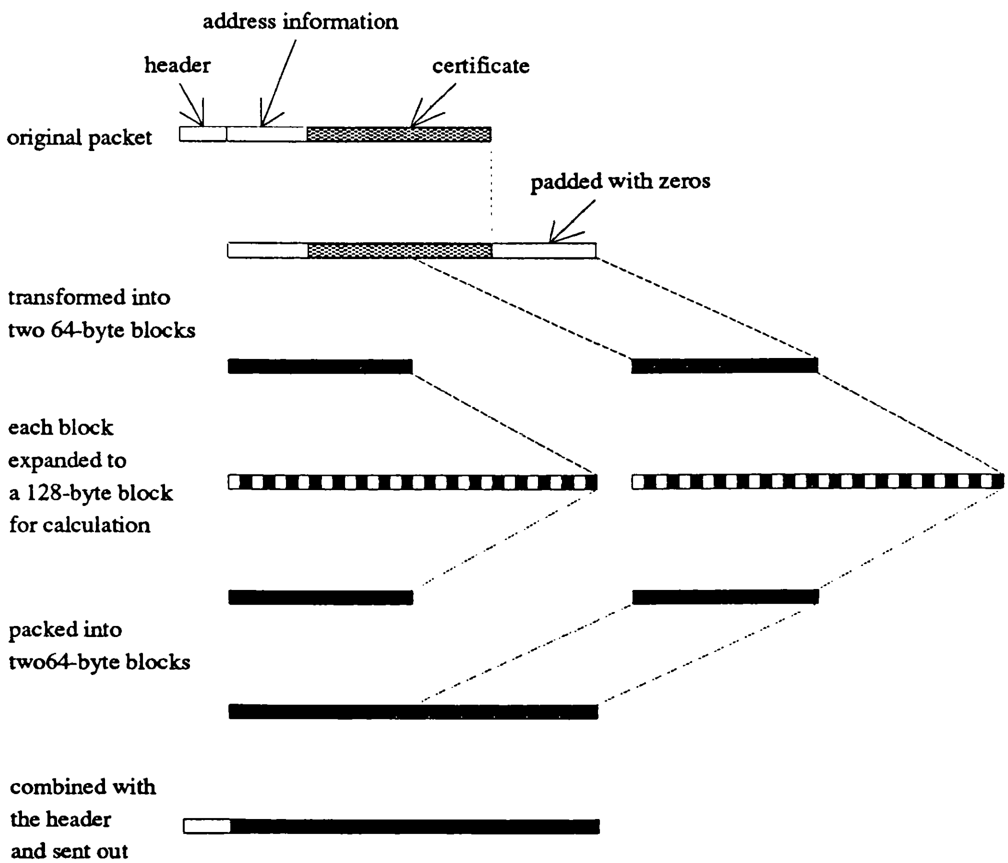


Figure 5.6: the data structure flow for encryption

each 64-byte block. The final result is a clear text header and encrypted 128 bytes of data, totally 144 bytes are sent out. The receiver decrypts the packet in an opposite way. Figure 5.6 shows the data structures flow.

5.14.4.5. Key generation and distribution

In our implementation, keys were generated by one central agent and distributed over the network to each host by means of the UNIX copy command under which was the Network File Store. No encryption was used for this operation.

In the real use of our scheme, the key generator might be located in each host, or there could be an agent. If key generation is done by an agent, the distribution of the key may be done by making use of off-line media such as disks and tapes. Alternatively, distribution is done over the network. In the latter case, a secure distribution protocol between a host and the key generator is needed.

5.14.5. Extra cost required for the secure ARP

5.14.5.1. Overview

Using secure ARP for the interconnected LANs within an organisation, requires all the agents of our authorisation mechanism, viz. the management system and the Certification Authority (CA). In the following subsections, we describe those functional modules required for the secure ARP. The next subsection lists the required modules and their sizes if known. Section 5.14.6.3 will report the delay imposed by the encryption and decryption. Section 5.14.6.4 will discuss the frequency of the operation.

5.14.5.2. Required space for the functional modules

In Section 5.10, we listed the required functions in the management system and the CA. In this subsection, we decompose those functions into the detailed functional modules required for our authorisation system. Figure 5.7 shows the location of the functions.

Through the implementation of the secure ARP described earlier, we have implemented some of these required functional modules as well as similar ones. In the following, we list the modules and their sizes where they are known. The sizes in this subsection are based on objects compiled under the SUN 4/60 systems.

In the CA, the following functional modules are required:

- Off-line
    i. to generate a Registration ID
    ii. to input host information associated with a Registration ID.
- On-line
    i. to secure interfaces to the manager system and to the Naming

Authority (NA)

ii. to compare the host information from the manager system with the one registered off-line

iii. to generate certificates

The secure interfaces to other agents, the management system and the NA are a set of request and reply messages, and the size would be 40 K bytes; our implementations of the secure ARP shows that an encrypted request/reply transaction will take 10 K bytes with the security routine library which is about 30 K bytes. We have implemented one of the above functional modules, which generates a certificate of an IP and Ethernet address mapping; the size is about 5 K bytes with the ISO Development Environment (ISODE) library which is around 90 K bytes. Our certificate generator only uses the routine for the time stamp of the ISODE library which includes many other unused routines. Since those libraries are expected to be shared by other processes, only one copy of the libraries is required in a system. The module of comparison would be around 20 K bytes. The off-line functional modules would have a total size of 30 K bytes. Therefore, a CA system would require less than 200 K bytes. If the system where the CA resides was managed securely enough, so that a Trojan Horse attack to the shared library would be prevented, the ISODE and security library could be shared with other processes in that system; in such a system, we only need an additional 66 K bytes for a CA capability.

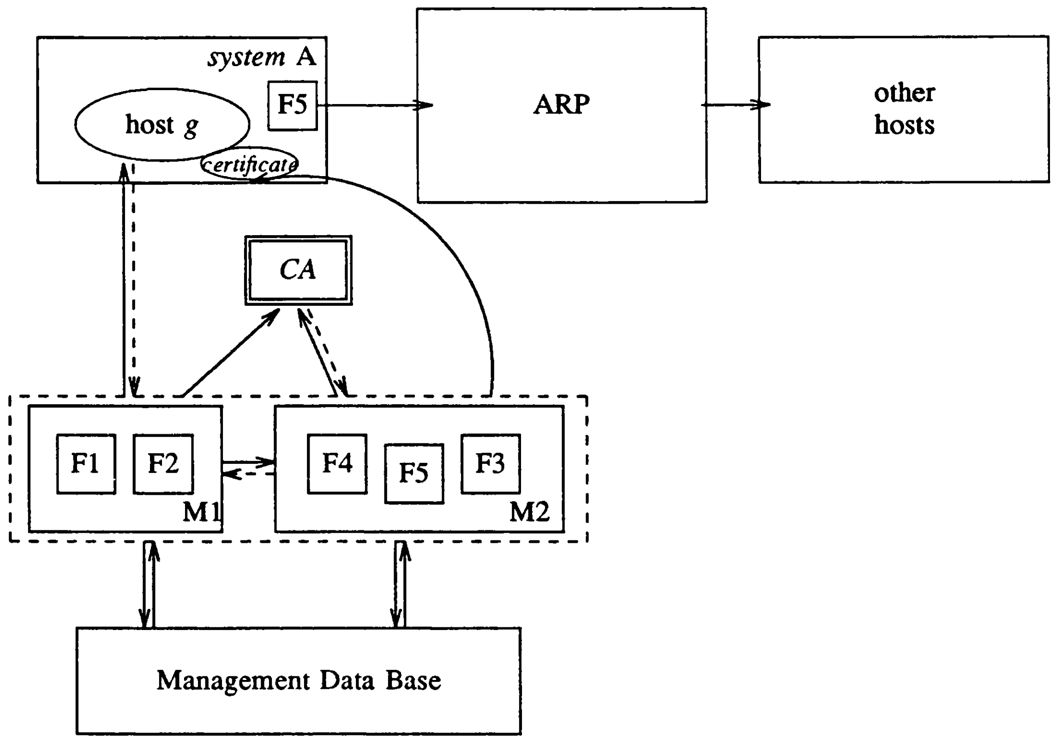
As we described in Section 5.10.1, the management system requires the following functions: detection of changes, verification, control, request for notarisation, and dissemination of certificates. In the following each function will be decomposed of functional modules.

Detection of changes requires the modules to do the following:

i. to monitor traffic and to produce a list of active hosts

ii. to get a list of registered hosts — we could ask for the Directory, then we need the Directory Access Protocol (DAP) module

iii. to receive a report from hosts — we could use the event reporting mechanism of a management protocol.



Legend:

M1: a management subsystem at each subnet

M2: a management subsystem at each subnet
F1: detection of changes

F2: verification

F3: control

F4: notarisation

F5: distribution of a certified address

——➤ : request or report

◄- - - - : reply

Figure 5.7: The locations of the functions

iv. to provide an interface to the network management personnel

v. to compare the monitoring and reported results to produce the list of inactive hosts which would be used for verification

The verification function can be decomposed into modules to do the following:

i. to read a Registration ID or the list of inactive hosts

ii. to ask the CA for the host information with a Registration ID

iii. to consult the management data base (e.g. a Directory) for the information about a registered host — we could use the DAP module.

iv. to verify a host possibly making use of management protocol request and reply messages; we need a capability to record Ethernet level header information for this transaction

v. to produce a report of an added host or a removed host, and send it to the CA

The request for notarisation function requires the module to get a certificate for a host from the CA.

The controller function is required:

i. to access a newly issued certificate for a host and assign it to the host possibly by making a use of a SET operation of a management protocol.

ii. to interface to the network management personnel

The distribution function is required to distribute a certificate to the information systems including the management data base. In our application to the secure ARP, one of those information systems is the ARP protocol operation; in this case a functional module of distribution is located in each host who would participate in the ARP operation. Moreover, we require the distribution function for updating the management data base which could be a Directory; in that case, the DAP module could be used.

The Directory protocols and OSI management protocols are becoming available in many hosts. If there are already present, then the extra cost we require for the management systems would be to implement less than a half of the above listed modules: to produce a list of inactive hosts (Function 1), to verify and produce a report of added or removed hosts (Function 2), to get certificates (Function 4), and to distribute them (Function 5). Amongst these, we know that it takes about 40 K bytes including the security routine library for the "get certificates" operations, i.e. request and reply. The size of other modules in the management system would be the order of about 30 K bytes. The total size of the management system would be about 110 K bytes apart from the DAP and the management protocol modules, with 30 K bytes each for Functions 1 and 2, and 10 K bytes for Functions 4 and 5 with the security library 30 K bytes; if the system the management functions was secure enough, and the security library could be shared with other processes in that system, the overhead of the management functional modules would be about 80 K bytes. However, as we describe later, some of them should reside in each subnet. If the management system functions are decentralised or distributed and each subsystem will require the secure communication, an extra 10 K bytes for a secure request/reply transaction module is needed per subsystem, providing the subsystem already has the security routine library.

The distribution function (Function 5) at each host requires 50 K bytes including the library; as a local host environment may not be necessarily secure enough for the security library to be shared, the functional module of distribution in each host should be self-contained module and should include its own copy of the security library routines — therefore, it should be as big as 50 K. Each host also requires to have a process to remove the certified entry when the expired message is received; this process could share the security library with the distribution functional module, and the total size of this process could be 30 K bytes. Therefore, the extra 80 K bytes should be required at each host for the secure ARP process.

The management system includes the subsystems which are located in each subnet and which perform the functions of monitoring and verification of host configuration. The rest of the management system functions such as remote controller of the hosts and the notarisation requester as well as the CA are located anywhere within the organisation.

5.14.5.3. Delay for a secure ARP

The delay incurred by the processing of encryption, decryption and filtering at the ARP operation is as follows. With our workstation — a Sun 3/50 with no special hardware to assist the encryption and decryption, in average, it takes 4.55 seconds to encrypt a 64-byte-long block of data, and 0.41 seconds to decrypt it. That is, an encryption of ARP data (two 64-byte blocks) takes 9.10 seconds. It takes 0.82 seconds for the receiver to decrypt the ARP packet, and a further 0.41 seconds to decrypt the certificate.

We also observed on a SUN 4/60 that it takes 2.71 seconds to encrypt a 64-byte data, and 0.25 seconds to decrypt it; on a SUN4/65 system, 2.17 seconds to encrypt, and 0.20 seconds to decrypt.

5.14.5.4. Frequency of the ARP operation

In theory, if we had n hosts on a set of Ethernets interconnected by bridges, at most $n(n-1)$ pairs of ARP Request and Reply transactions would be required in the beginning. If we had 100 hosts, 9900 transactions would be carried out in the beginning of the network operation. This is improbable, because an ARP Request packet would be sent out only, if (a) the address resolution was required, and (b) there was no entry in the cache of the ARP table. The ARP table is filled with the address mappings learned from the incoming ARP packets, which would be a broadcast ARP request or an ARP reply. The ARP reply is not broadcast but destined only to one host. For a broadcast ARP Request packet, all the hosts will refresh the timeout of the entry for the sender if the entry already exists; a new entry of the ARP table for the sender will be created only at the host to whom the request is made. Note that the target host is specified in the ARP Request packet, but that packet is enclosed in an Ethernet frame with the Ethernet broadcast address in the destination address field. With this cache scheme, it requires $n(n-1)/2$ pairs of request and reply transactions for every host to know of everyone else, because the both parties of one transaction will cache each other's address mapping. Again if we take an example of 100 hosts in an environment, it requires 4950 pairs of transactions. Presuming that the timeout is set to 20 minutes, the average interarrival time of an ARP request received by all the hosts would be about 242 milliseconds if all the hosts on the network need to know each other within the timeout. However, we show below that this is a worst case analysis; many less interactions are observed in a typical UCL experiment.

Falaki and Sorensen [FaS91] observed that in our departmental network with four file servers and about 120 hosts, 80 percent of the traffic is due to a file server and client communication; the rest is mostly due to the peer host traffic. A set of peer hosts are the ones which belong to the same domain such as the same project or the same function such as for a teaching purpose. It is very likely that the ARP table entries are required mostly for either the server-client pair or peer-host communication. Figure 5.8 shows the observed transition of the number of entries in ARP tables at different hosts, three servers, one of which was a monitor, one student work station and seven research work stations. The observation took place on a Thursday afternoon for one and a half hours with a ten-minute interval. Server 2 is also a monitor, so that it has as many entries as it has to monitor. The work stations require only several entries; occasional increases in size are all due to the user execution of the "rusers" command which we will explain later. Our observation supports Falaki's claim that a host only communicates with a limited number of other hosts.

We have produced a simulator of a cache of ARP entries to count the necessary number of transactions; a randomly chosen host broadcasts an ARP request for a server or one of its peer hosts, and receives a reply until all the hosts have required entries. The simulation result shows that with 100 hosts, of which 5 are servers and each of which has 4 peers and communicate with 2 out of 5 servers, 341 pairs of Request and Reply transactions are required for every host to have the required entries in their ARP table. Note that only the request packets are broadcast, so that they are picked up by all the hosts on the network, whereas the reply packets are picked up only by the destination hosts. Once an ARP Table entry is created for an address, it will be updated by listening to all the broadcasted ARP Requests through the net. Another simulation result shows that once the entries are created, 165 pairs of transactions are required to keep updating the ARP table at each host in the above examples. Therefore, even if there is a timeout, e.g. 20 minutes, by this updating mechanism, the entries which are used frequently are retained in the table. Using our secure ARP system, the timeout for certified entries would be several hours. Therefore, in the previous example of 100 hosts with 5 servers and 4 peers for each host, 341 pairs of transactions would be carried out once in several hours, as distinct from the 4950 worst case analysis.

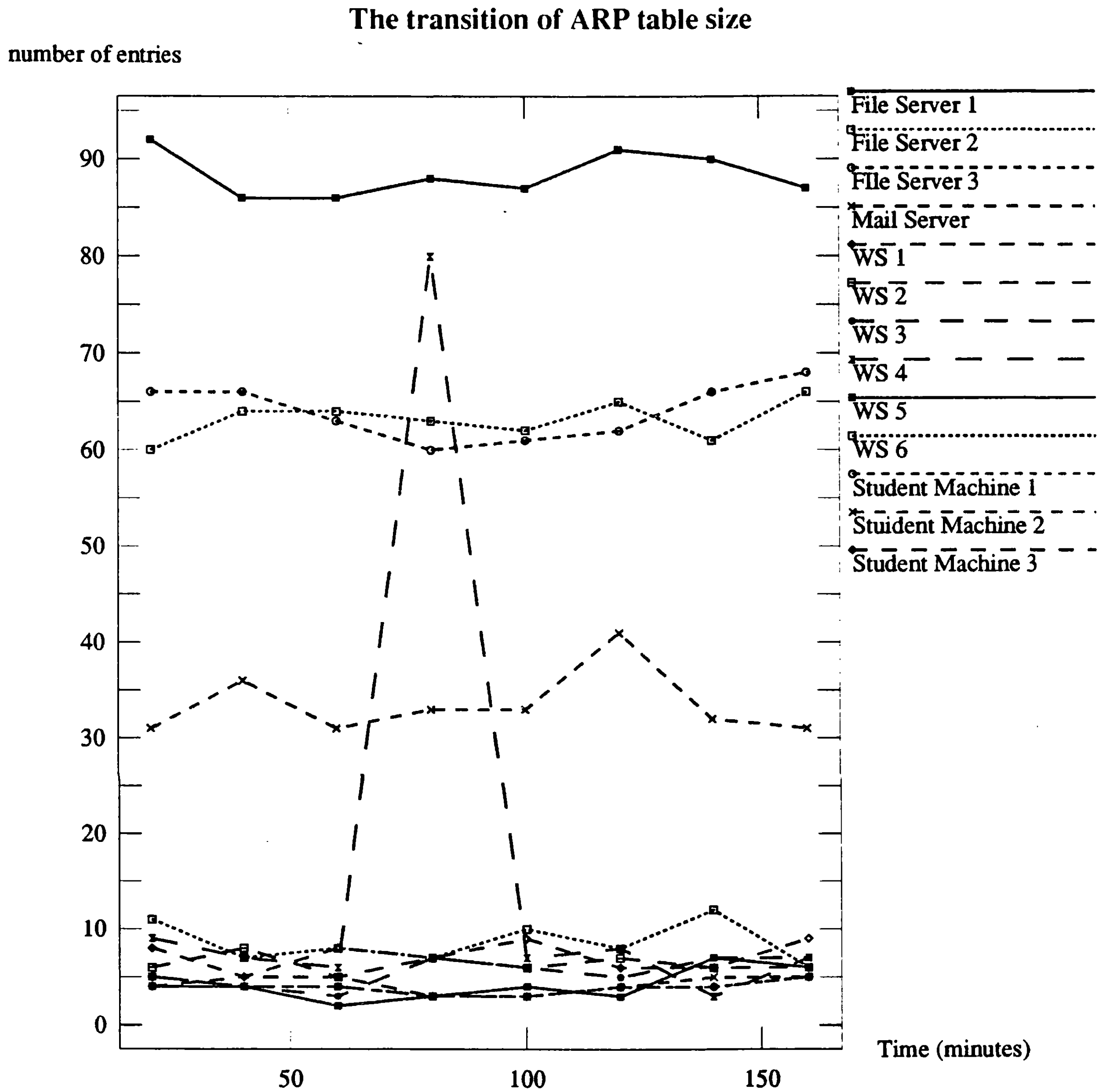


Figure 5.8: ARP Table Size Transition

We monitored the ARP request traffic in the end of October for one week. In the following we take an example from one of the days, Thursday 24 October 1991; the other days showed more or less similar pattern of traffic. Figure 5.9 shows the cumulative frequency of the packets. Unfortunately because of static configuration of the servers, about half the 10002 observed ARP packets were due to the servers calling for the hosts which were not configured in the network; this should have been fixed at the network routing level. We omitted this irregular traffic, and considered only the traffic without calls to dead host. The occasional jumps of the number of arrivals were due to the execution of a user command called "rusers", remote users, to see the users currently logged in each host on the network; upon an execution of the command, the user host broadcasts a `ruser` request over the Ethernets and every other host tries to reply with some small delay, therefore, the hosts which have no ARP entry for the user host will send an ARP Request asking the Ethernet address of this user host. If the user host is neither a server nor well-known to many hosts, the number of the ARP requests becomes very big. Some bursts were also due to monitors; however, the number of packets is relatively small, because the server is well-known, and most of the hosts have an ARP entry for the server. Our observation shows that this average number of the burst is about 60 packets within the order of 100 milliseconds. In order to examine the average interarrival time of the ARP requests, we differentiate those bursts from the normal traffic which was drawn as the third line in the Figure 5.9. In the following, we discuss the analysis of the normal traffic, and then we will discuss about the treatment of the burst packets.

We have removed the burst traffic from the data. 2648 ARP Request packets are received over about 6 hours. About 150 pairs of transactions were carried out within the timeout of the ARP cache — 20 minutes. Our simulation result of necessary number of updating the ARP cache with 100 hosts with 5 servers and 4 peers — the similar situation as UCL.CS — was 165 pairs of transactions. The fact that the real number of transaction is less than the calculated number is that not all the peer hosts do communicate each other in our departmental network. The least square regression result for the interarrival time of the packets to have an exponential distribution did not fit very well with the observation as shown in the Figure 5.10. Therefore, we use the statistics derived from the samples in the following discussion. We found that the average arrival rate of ARP Request packets would be at most 0.06 packets per second from the sample; the mean interarrival time is about 8 seconds. The exponential distribution derived from that sample by the least square regression gives the rate of 0.16 packets per second — i.e. the average interarrival time is about 6

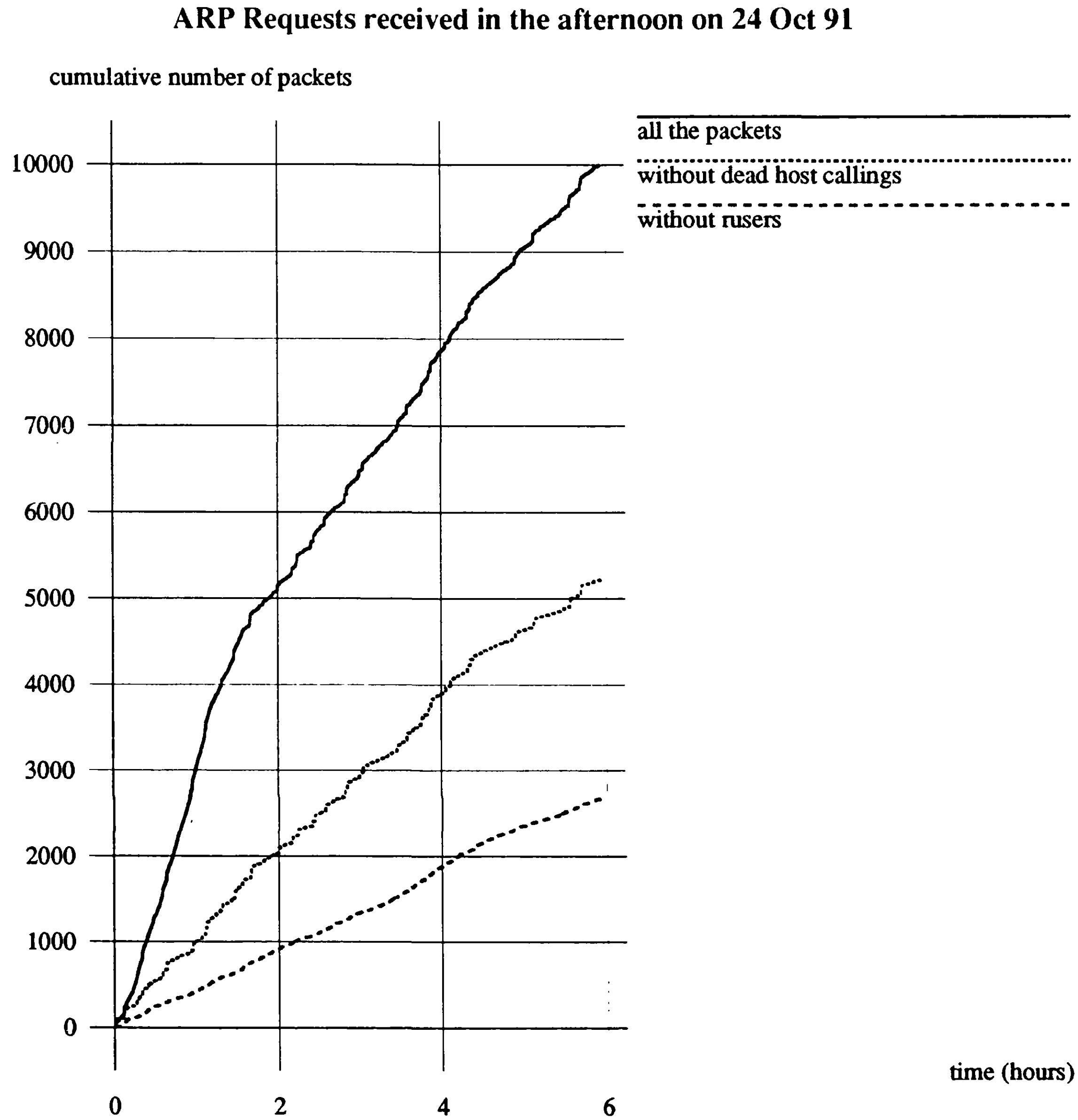


Figure 5.9: ARP Request Cumulative Frequency

seconds. Also, we observed that the average number of ARP Reply packets destined to a host was about one percent of the total number of packets — about 15 to 20 reply packets per weekday afternoon on average.

With the current ARP system, the process of incoming packet takes the order of several hundred microseconds on a SUN 4/60; we derived this number simply by subtracting the

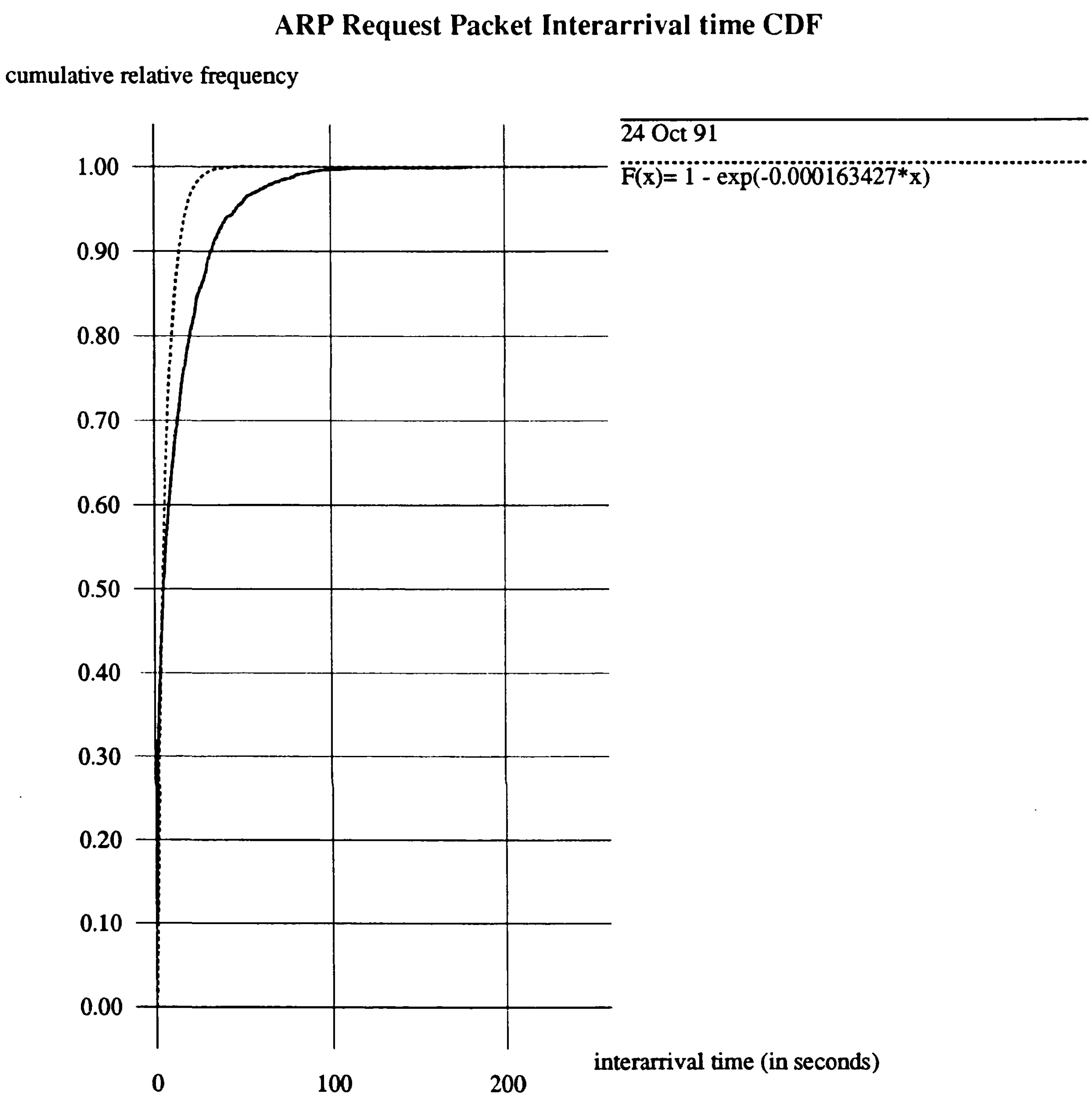


Figure 5.10: ARP Request packets CDF

time when an ARP Request packet is observed by the monitor from the time when the associated Reply packet is observed. This simple calculation gives us roughly an idea of the processing time of caching information into the ARP table at a host, because when a host receives an ARP Request, it will update the ARP table according to the information in the packet first and then send the request if the request is made to this host. The measurement result is presented in Appendix F. Adding our encryption and decryption procedures, the incoming packet processing time will need additional 0.75 seconds on a SUN4; 0.50 seconds for decryption of the packet and 0.25 seconds to decrypt the certificate. As the decryption procedures are the same for any packet, we could consider that this decryption time is deterministic. The incoming packet process could vary because if there is no entry for the learned address, a new entry may be created and the reply will be sent back if the request is made for this host. However, the variation of this procedure is the order of some hundreds microseconds and it is negligible compared to the total service time, because the decryption process takes the time of the order of seconds. As our samples show the average incoming ARP Request packet interval is several seconds which is more than the average system service time of our secure ARP.

In this way, the learning the address mapping from the broadcast ARP Request packets would work for normal packets, and we could discard most of the burst packets. However, when a host wanted to communicate with another host and it had no entry in the ARP table for the destination, i.e. the destination address did not appear in the broadcast ARP packets so far, the source host would have to send an ARP Request and receive a Reply. A sending host has to take 5.42 seconds (2×2.71 seconds) to encrypt the request packet; note that the certificate was already encrypted by the CA. This is not practical in the current environment because the application at the source host would have to wait that long unless the encryption is done very quickly by hardware if possible. For the sake of maintenance of information integrity, we may only need the certificate attached to the original ARP packet, and encryption for the whole packet may not be necessary, so that during the ARP operation it is required only to decrypt the certificate — no encryption is necessary; the generation of a certificate, which takes longer time, is done long before the use of the certificate. Indeed, a receiver could ignore the original sender address part but takes the certificate part for learning; the idea is that if the information is authorised as a *truth*, it does not matter who says that.

Although in a particular environment like our departmental network, the interarrival time is long enough to handle the incoming packets, it still takes about 0.25 seconds to get a trustworthy entry of the ARP table. When a host has to communicate with another host whose address is not in the original host's ARP table, the IP process on the original host will have to wait 0.25 more seconds than the time taken by the current ARP system. If it is not acceptable, we could provide two ARP tables; one as same as the current ARP table, and another with certified entries. When the incoming packet is received, the packet is copied to the queue to the secure ARP procedure which will decrypt the packet and cache it to the table of certified entries. Meanwhile the clear text part of the packet is processed as the current system does, managing the ARP table of addresses from the clear text part of the packet. At address resolution for an IP address, the certified table is looked up first, so that the certified information is taken first. Therefore, in real time, if there is no entry, the IP process will get the non-certified mapping first, and after 0.25 seconds, the certified entry for the same destination will be available; in other words, during the decryption time, if the need is urgent, the IP process can go on with the communication with the untrustworthy mapping for 0.25 seconds if it wishes. If the IP process requires the secure ARP mapping for the destination whose entry is not yet in the table, it will have to wait but only once in several hours — the timeout of a certified entry.

rate(packets/300 msec)	frequency
0	34604
1	1211
2	131
3	23
4	15
5	15
6	9
7	4
9-61	29

Table 5.1: ARP Request arrival rates on 24 Oct. 1991 without calls to dead hosts

rate(packets/300 msec)	frequency	
0	34710	
1	1143	
2	123	
3	15	(8 monitor)
4	5	(5 monitor)
5	1	(1 reboot)
6	1	(1 reboot)
7	1	(1 reboot)

Table 5.2: ARP Request arrival rate on 24 Oct. 1991 without rusers

Table 5.1 shows another kind of statistics, the number of arrivals per the service time for our secure ARP Request packet processing which we set 300 milliseconds. Considering the decryption time for a certificate and the current delay between an ARP Request and its associated reply, we presume that setting the service time as 300 milliseconds is more than enough for decrypting a certificate and updating the certified ARP table. The data includes the rusers traffic but not the calls for the hosts which were not available; therefore, it represents the normal situation in our departmental networks. The majority was zero to 2 packets, occasionally several packets arrived within a 300 millisecond interval. The high arrival rate intervals — i.e. 9 to 61 packets per 300 milliseconds — were due to the rusers traffic. When we removed the rusers traffic which, we observed, occurred several times a day, we had the rate distribution as in Table 5.2; those intervals with several packets were all due to rebooting a host, and some of them were for a network monitor to probe each host, so that it needed address resolution of the other hosts. Those ARP Requests due to rebooting, monitoring, and the bursts from rusers and calls to dead hosts previously mentioned, carry the source addresses which are not of interest to every host. They could be discarded by most hosts but the one which is the target. At the target host, the burst could be dropped at the certificate decryption queue whose size could be set to two, but the clear text part of each packet could be processed as the current system does, so that reply can be sent out within the delay of about some hundred microseconds. The reason for the queue size of two for the certificates handling process is that a reply packet will be sent back to the requester in the order of several hundred microseconds after the request is made; the requester may receive its own request packet as well on some implementation of Ethernet.

5.14.6. Conclusion of the section

Even if the address mapping information is loaded statically from the Directory at boot time, a certificate could be used for the confirmation of the credibility of the information.

One drawback of use of a certificate is an additional mechanism required to maintain its versions, so that stale certificates would not be accepted. This is done partly by timeout at each site; the CA regenerates a valid certificate with a new time stamp and an authentic site receives its renewed certificate from the management system, meanwhile the expired certified entry in the ARP table at each site is cleared. Moreover upon removal of a host, one needs the trustworthy distribution of the Expired message to all the sites in the subnet as well as the process at each site which exercises a sanity check on incoming ARP certificates with the list of expired ones. The Expired message can be sent at the ARP level, creating the third type of ARP packet — the ARP Expire packet. Alternatively it could be sent to each site using a management protocol.

For our authorisation system, we require about 110 K bytes (in terms of a SUN4 system) apart from the Directory protocols and the common management protocols. Most of the management functions of our authorisation system are used only once when a host is configured to the network. After that, all that is required is the monitoring function which detects the removal of the registered hosts; upon the detection of a removal, the other management functions and the CA are required to produce and propagate an Expired message associated with the certificate of the removed host.

At each host, about 80 K bytes (in terms of a SUN 4 system) are required for the additional features to the current ARP system; in addition, another ARP table of the certified entries would be 20 bytes per entry therefore $n \times 20$ bytes where n is the number of hosts of interest — the certified entry list does not have to hold all the hosts but only holds the host of interest. The process to another ARP table with the certified entries would take less than 300 milliseconds, presuming that an ARP packet consists of a clear text with an attached certificate — i.e. no encryption of a whole packet is carried out. The measurement result in our departmental Ethernets shows that occasional bursts of arrivals of the ARP Request packets, from which one may update the host addresses mappings, are caused by a particular network operation, rusers, and those packets include the source addresses which are not of interest for a host for its normal communication. Hence, the burst packets should be discarded by the process of the certified ARP table, but should be cared by the process of the non-certified ARP table as the current system does. With an extra 80 K bytes of memory and 300 millisecond of delay to have a certified ARP table entry, we can avoid the risk of unauthorised redirection; note that 300 millisecond delay could be interpreted as that the secure address mapping would not be ready until 300 milliseconds later, and that a host might operate insecurely for 300 milliseconds if this is acceptable. As for the frequency, with the previous example of 100 hosts with 5 servers and 4 peers for each host, it requires less than 2 minutes (341×300 milliseconds) to process 341 secure ARP Requests at each host once in several hours. This figure could be acceptable in our departmental environment.

Occasional burst requests observed in our departmental network were of no interest to the normal network operations, hence, they could be processed only with their clear texts and no certified entries would be created. However, if they are of interest, the queue for the certificate decryption should be long enough to hold a burst.

If one wants to justify to have our authorisation system in a network, the following checks should be made:

- to examine the seriousness of the network redirection or other network threats which cannot be solved in other feasible ways
- to examine the possibility to have the extra resources for our authorisation system and the extra functions in the current ARP system.
- to estimate the required number of ARP requests, and to examine if the total delay would be acceptable or not.

Upon a satisfactory result of the above examinations, one could use our authorisation system.

If there was hardware to encrypt and decrypt in the order of one or two milliseconds, then the service time of the secure ARP would be the order of two or three milliseconds at longest on a SUN4 system. In that case, the secure ARP would have to maintain only one table with certified entries — the second table with entries derived from the clear text part of the packet would not be required.

5.15. Conclusion of the chapter

The primary design goal of internet protocols has been emphasised to be connectivity and reachability [CeC83], [Pou85] over the various types of subnetworks rather than access control. The Internet Protocol is a successful example in this respect. The consequence is that the internet protocol environment is insecure. In a traditional network information systems such as name server and routing information exchangers, an information item has been claimed without verification. There are many pitfalls where anyone can break in [Bel89]. There are many chances that network operators can make hazard mistakes — e.g. a typing mistake in setting up a network address. Incidents have been almost avoided, partly because of the implied mutual trust in the research internetwork community, and partly because of the size of the internetwork has been manageable for an operator to input information items correctly.

As the size of an internetwork grows and the diversity of nodal system architecture becomes significant throughout both inside and between single organisations, some control is required to protect the network from both malicious and innocent incidents. The original Core Gateways of the ARPANET and MILNET were operated by BBN, and access control prevented manipulation of routing tables. As the Internet becomes more diverse, more general techniques are needed. Both the Visa scheme [Est85] [EMT88] and the Policy Routing [Cla89] [EsT89] are based on the idea that the control would be exercised at a border gateway between an organisation and the inter-organisational network.

Our scheme, which would make registered information trustworthy, is based on the philosophy that any incident could be prevented from going beyond the subnet. The subnet is an address resolution domain, and would be a part of an organisation. The idea is to solve the problem within an organisation.

The very reason for those incidents is that a network operation is reliant on addresses. The current problem in networks is that there is no way to know how trustworthy the addresses in use are. We regard a network as an information-based system. How trustworthy the information-based system operates, depends on how trustworthy the information, i.e. an address, is.

Information systems are responsible for the addresses used in the networks. We presented the flow of address information. The organisational administrator, an object generator, generates the logical existence of the host. An address associated with the host is produced by the the Naming Authority (NA), an information generator. An address is incorporated into a host at configuration. The host, an information provider, introduces its address to information systems at registration over the network. An information system could be a network operation such as an Ethernet ARP system, or an information repository like an X.500 Directory. Finally the other hosts learn the address of a new host from information systems by the look-up operation.

Our approach is to make address flow trustworthy. In fact, the security of the information maintenance and look-up operations have been a popular topic and could be found elsewhere [Den83], [FSW81], [ClW87], [MNS87], [CCI88b (X.500)]. What is new in our scheme is to incorporate the verification of the host configuration as well as to apply security techniques to the registration, maintenance, and look-up of addresses.

Host state transitions were shown, and the address flow was proved formally, using the logic introduced by Burrows, Abadi, and Needham, which was originally intended for analysis of authentication protocols.

The required functions in the management system and the CA are described. The viability of the scheme was discussed with introducing the partial systems in case of system failure. Two examples were taken and examined; UCL-CS and the Internet. UCL-CS is an example of intra-autonomous system. The Internet domain was discussed by extending the model to the inter-autonomous systems environment.

Finally we applied the use of certificates to the existing Ethernet Address Resolution operation to demonstrate how our scheme would fit in the current system. An interesting discovery was that the decryption of the certificate takes much less time than encryption. The implication is that once the certificates are issued at registration, the use of the certificates in the network operations could be feasible. In fact, it does not matter how an address resolution takes place. Our system takes 110 K bytes for the management system apart from the popular tools such as the ones for directory access and for management communication; each host requires to have 80 K bytes additional to the current ARP system. The delay imposed by processing the certified ARP table would be less than 300 milliseconds on a SUN4 system. Presuming that the interarrival time of the ARP packets is several seconds, this delay would not be so significant compared to the average interarrival time on our network, although real time application operation would suffer. We could avoid this by having two ARP tables at each host, one learnt from a clear text ARP packet and another learnt from a certificate; this delay implies that a host would go on network operations insecurely for 300 milliseconds. Once a certified entry is obtained, it will last for several hours. This indicates that the frequency of the secure ARP operation is mostly done in the beginning of the several hour interval, and only the monitoring and expired message operations would be necessary in the rest of the interval.

In the next chapter, we will examine the extensibility of our scheme by applying it to other types of information than network level addresses.

Chapter 6

Extensibility of the Trustworthy Information Flow

6.1. Overview of the chapter

In the previous chapter we presented the trustworthy flow model for network addresses, proposing the use of a certified mapping between the network address and the associated subnetwork address in the ARP operation. In this chapter we examine the extensibility of the model by applying it to types of subnetworks other than the Ethernet, and to other types of information which are sensitive to network operations besides network addresses. We suggest the use of a certificate for routing, time, and the other network configuration information, as well as some application level information, including keys for authentication.

The chapter is organised as follows. The next section discusses the applicability of the certified address resolution information to other types of subnetworks. Section 6.3 describes the use of certificates in name resolution. Section 6.4 presents the use of a certificate in routing on the Internet. Section 6.5 describe the use of a certificate in time synchronisation on computer networks. Section 6.6 discusses another type of configuration information, the maximum transmission unit (MTU). Section 6.7 describes trusted peer information. Section 6.8 describes the use of certificates for authentication keys. Section 6.9 presents the conclusions of the chapter.

6.2. Address resolution information

In the previous chapter, we described how the integrity of address resolution information is assured by making use of a certificate, which provides the mapping between an Internet IP address and an Ethernet address, in the Ethernet Address Resolution Protocol (ARP) operations. In this section we describe the history of ARP which gives a reason why it could be used in a variety of subnetworks, and then discuss the applicability of this type of certificate to other types of network than the Ethernet.

ARP was designed originally for a local network system, the Chaosnet [Moo81]. Chaosnet was developed in 1975 by the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, under Project MAC. It was intended to be used as the internal communication medium of a shared file system, the Lisp Machine system. Chaosnet was composed of hardware and higher level protocols. The hardware level protocol was similar to the Ethernet, but each interface was allocated a time slot based on its hardware address. At the higher level protocol, the software was similar to the TCP/IP, but designed to interconnect the networks within a campus. The higher level address was composed of the network address and the hardware address; hence, the address resolution was simply algorithmic. Later, the hardware level was replaced by the Ethernet, over which the software level protocol ran; therefore, an address resolution protocol was needed to translate the software level protocol address into an Ethernet address. As the original goals of the Chaosnet were simplicity and performance, security issues were relegated to the end-to-end level.

The ARP architecture is designed to work in any types of networks and subnetworks which support broadcast. Indeed, ARP is used for Internet IP address resolution on various types of subnetworks such as IEEE 802.3, 802.4, and 802.5 local area networks [PoR88], the Fiber Distributed Data Interface (FDDI) network [Kat90], and the Switched Multi-megabit Data Service network (SMDS) (a packet switched data service that provides LAN-like performance and features over a metropolitan or wide area [PiL91]).

The use of ARP in these network environments allows the same network level threats as those of the Internet IP over the Ethernet environment which were described in Chapter 4. Therefore, the scheme presented in the previous chapter is applicable to those types of subnetworks as well.

In an ISO connectionless mode internet [ISO86(DIS 8473)], there is a reporting mechanism between the end systems and the intermediate systems to learn each others network address and subnetwork address mapping. This learning mechanism consists of a mixture of ARP-like and Internet Control Message Protocol (ICMP)-like functions, such as ICMP Re-direct which could cause a change of routing information in an end system. We could apply the use of certificate for the pair of the network and subnetwork addresses.

6.3. Name resolution information

Name resolution can be considered as address resolution at a higher level. In the Internet [CeC83], the name of a destination application is translated into an address, which consists of a port number, a transport protocol ID, and a network address. In an Open System Interconnection (OSI) environment, it is a translation from an application entity title to the Presentation address, (the tuple of selectors, each of which identifies a layer service access point as follows: (P-selector, S-selector, T-selector, (list of N-selectors)), where P, S, T, and N stand for Presentation, Session, Transport, and Network layer services respectively [ISO84]). The translation of the application names is done by a name translation system such as the Domain Name System or a Directory System, introduced in Chapter 2. However, the integrity of information stored in these systems is not necessarily maintained. Application level authentication can prevent unauthorised access to these systems, but it cannot guarantee the correctness of information about configuration of the application entities, e.g. the misconfiguration of a port number or transport protocol ID. Using this loophole, an deceiver could perform higher level re-routing. Imagine that the misconfigured application is a server, the deceiver can masquerade as the server by using the proper port. An incoming request to the proper port would be received by the deceiver, and it will then send the request to the misconfigured authentic server. Therefore, the deceiver could be a moderator of incoming requests; it may drop, delay, and replay the incoming requests, even if there is authentication at the application level.

We can certify name translation information to prevent higher level re-routing. During configuration confirmation, the application address is verified and the server is given a certified address. Those certified addresses can be stored in the name server database for secure application level communication.

Intra-organisational translation could use the organisational Certification Authority (CA) for producing certificates. For inter-organisational translation, a central CA might not scale well because the CA would have to produce certificates for all the local applications in all organisations. Presuming name translation is performed by a distributed name server such as the Domain Name System (DNS) or the X.500 Directory System, there would be two ways to use certificates. One is to decrypt certificates, when the name and address mappings are stored in the distributed database: then the information is kept as clear text in the database. This is a straight forward application of our intra-organisation registration scheme, but it is assumed that all the name resolvers in the DNS or Directory Service Agents (DSAs) in the Directory System in different organisations trust each other. Another way is to store the certificates without being decrypted and the user application would decrypt them using the certification path during look-up. A certification path is defined in the X.509 Directory standard as an ordered sequence of certified public keys held by objects in the Directory Information Tree (DIT) which, together with the public key of the initial object in the path, can be processed to obtain that of the final object in the path. In short, it is a sequence of public keys, each of which is certified by the next trusted CA in the path. In practice, a certificate for the name and address of a local application A, produced by a local CA in one organisation, would be obtained by another application B in a different organisation, through a distributed name server via that application's Directory entry. Then B would obtain a certification path from its site to A's CA, and verify A's certificate. This allows that all the name translation agents in different organisations do not have to trust directly one another. It is based on a chain of trust established between organisations, in which a CA in one organisation would trust another CA in another organisation which would trust another CA, etc.

As the DNS and the Directory evolve and may include untrustworthy sites, the second approach might be more applicable. Whichever method is taken, the notarisation of name and address mappings within an organisation will be required.

6.4. Routing information

By manipulating the routing information exchange protocol operation, a deceiver can reroute packets and obtain control over them; packets may then be dropped, delayed, or replayed. Whatever application level protection is provided, this could cause unauthorised denial of services at the application level. Moreover, time critical operations such as network time synchronisation will suffer from this threat. The use of a certificate in address resolution operation provides protection at the subnetwork level from network level threats. Alternatively, protection from re-routing can be exercised at the routing information exchange level on each router. In particular, inter-organisational routing cannot be protected at the subnet level, but only at the routing information exchange level.

In general, routing is based on distance vector or link states [Per85]. In the previous chapter we discussed the use of certificates for the distance vector information exchanged by the Exterior Gateway Protocol (EGP) [Ros82] in the Internet; each router is given a certificate which shows for which networks the router can possibly advertise reachability.

In link state-based routing, each router must maintain complete knowledge of topology. For this, we could use certified topology information; i.e. a relation between a node and its neighbours. This information is distributed by flooded routing updates, including state information such as delay of each link, to the neighbours [MRR80]. We cannot apply the authorisation scheme to the link state information. The link state information should be as near real time as possible; it would not be practical to have the state of a router verified from the network manager site, and to have a certificate issued for the verified state. We can authenticate the information provider, by having the authenticator of the source router attached to the routing information exchange packet. As a result, no bogus router can participate, as it would not have the certified authenticator.

Other state information which may require authentication is network monitoring information. If a router is observed as down, re-routing at the application level for electronic mail, file transfer, and remote terminal access can be carried out by the network managers. A bogus router may impersonate a victim router simply sending out "port not available" information. To prevent this, management and monitoring level state information could be accompanied by the certified authenticator of the sender.

As the Internet has grown in size and complexity, it is seen more as a connection of autonomous administrative organisations than merely as a connection of networks. The strict topological restriction of EGP — a tree topology — does not fit very well with the interconnection of the autonomous organisations, which would allow a mesh topology [BrE90]. Besides, more administrative control is required for inter-organisation routing; i.e. access control over the transit packets through an organisation is required. From this perspective, policy based routing has emerged. Among the proposed policy routing protocols, we look at one of them, the Border Gateway Protocol (BGP) [LoR90]. BGP is categorised as distance vector hop-by-hop routing similar to EGP. The BGP routing information updates include policy information such as the preferred Administrative Domain (AD) paths to reach each network and the next hop router. There are two types of certificates we could apply. One is to certify the attachment of the next hop router to the AD; another is to certify each AD. This certificate would not be used in the BGP packets; it is to be used for the distribution of Policy Terms [Cla89] which describe the policy of each AD, and is outside the scope of the BGP specification. A Policy Term (PT) for an AD would include information such as source and destination ADs, path constraints, quality of services, time restriction to use the AD, user class, authentication and security requirements of the AD, as well as charging and accounting policies [BrE90]. When an AD requests permission to be connected to the rest of the network, it could go through the registration procedure. It would be given a certificate for those constraints, and also the BGP router of that AD can be given its certificate as well. A certified PT will be distributed to all other ADs. Based on the certified PT, a BGP router can produce a policy routing table and verify the route advertised in a BGP routing packet.

Unlike the case we discussed about the Internet in Chapter 5, future autonomous domains have no hierarchy but are on an equal level. The link between the organisations would be managed by the cooperation of both sides. In such an environment, our authorisation system agents, i.e. the management system and the CA, would be distributed in each organisation; each CA is then responsible for the associated organisation and communicates with other CAs in other organisations with mutual trust. A certificate produced by a CA in one organisation, would be decrypted in another organisation by making use of certification path as described in Section 6.3. Alternatively, the certification path could be attached to the certificate, but this may not be practical, because a certification path from one organisation to another should be learned only once, providing a CA would not change its public key often.

6.5. Time information

Synchronised time is required for network and higher level communication, such as network-wide measurement, distributed processing [Lam78], network monitoring, and authentication protocols using time stamps. A deceiver could have control over those operations, attacking time synchronisation by masquerading as a server, modifying, replaying, delaying, or blocking synchronising packets [Bis]. Most of these attacks can be avoided by application level authentication and access control, providing that those authentication protocols do not depend on time stamps — they can use nonce. However, replay, delay, and block attacks can be avoided at the network level. The scheme offering protection from re-routing which was presented in the previous section could be used.

Another way to protect time information from threats is to filter the incoming information by using a certificate for the relevant information. One approach to filtering the incoming information is to check the reliability of a peer by use of a certificate for the latest verified time of the peer. Time servers are verified occasionally. As in the link state information in routing described in the previous section, it will not be practical to certify time information in every synchronisation packet. However, it could be done once in a while, such as when a time server is booted up, which might be once a day; alternatively a time server's clock would be verified more often, perhaps once every few hours, and then a certificate issued. The CA could issue a certificate for the clock of the time server including the time server's ID, its public key, and a time stamp by the server's clock. To verify a time server, the management system requires contact with the peer time servers.

Another approach in a specific operation, the Network Time Protocol [Mil89] used in the Internet, is to provide a certificate for a filtering parameter. The protocol uses a preconfigured parameter, the maximum filter dispersion, to limit the dispersion of the *offset values* of a peer; the *offset* value shows the estimated difference between the host clock and the peer clock. At the moment, the same value of the maximum filter dispersion is used for all the peers. This parameter can be chosen for each peer and certified by the CA to prevent bogus data from the peers. The certificate can be attached to each synchronisation message, or distributed to relevant parties in the beginning of the protocol operation. The integrity of time information can be maintained by this kind of sanity check using the certified parameters.

The dilemma in providing time synchronisation through the network is that it should be authenticated for secure synchronisation, yet the authentication itself may require the synchronised time stamp! This was observed in the Kerberos system [Ste91]. In our scheme, during configuration confirmation, a host does not need to use its own time stamp. The local time on the registering host is initialised at the end of confirmation. A certificate for the time protocol process, which will be used as an authenticator in periodical synchronisation, is then issued.

6.6. Other configuration information in the network layer

There are other network configuration parameters which could be certified. Among the parameters, the Maximum Transmission Unit (MTU), and subnetwork mask can be certified. Type of service is defined for each network; MTU specifies the maximum size of a packet allowed on the subnetwork. The subnetwork mask is used to filter the network number out of the destination address of a packet at routing to decide which router the packet should be forwarded.

Each subnetwork has a different MTU, and if a packet size is larger than the MTU in a transit subnetwork, the packet must be fragmented. Fragmentation allows a source host to send out a packet without knowing the MTUs on various subnetworks on the path to the destination, and the actual transmittable packet size to be decided as latest as possible. On the other hand, since the fragmented packets are reassembled eventually at the destination, fragmentation increases the number of packets, and hence, the network resources required for packet processing such as buffers and processing time which are allocated on a packet-by-packet basis on routers along the rest of the path to the destination. Loss of a fragment causes a higher layer connection-oriented protocol process to retransmit the whole original packet.

Considering these drawbacks, Kent and Mogul suggested the avoidance of fragmentation [KeM87]. From this perspective, the path MTU discovery scheme [MoD90] has been proposed to learn the minimum MTU throughout networks on the path to the destination; fragmentation in the transit networks is then avoided by setting the network packet size to the learned minimum MTU size. The scheme is that a host sends a test packet to the destination, whose size is set to the MTU of the nearest router indicating the prohibition of fragmentation; when the packet arrives at the network where the packet is too big to go through, the router at the network sends an error message with the MTU size of that network back to the source host.

There are at least two ways in which a deceiver could misuse the discovery protocol. One is that a deceiver might tell a host to use a very small MTU, so that the host will send many more packets than it would really have to. The other is that by sending a big MTU, a deceiver could redirect the traffic to a bogus router, if the MTU was used as a cost in routing. The former would cause performance degradation because it would consume valuable network resources both in the routers and in hosts — network resources are allocated on a packet basis. The latter would cause unauthorised access to packets.

The former attack is as follows. A bogus router can send the source host an error message with an extremely small MTU size, say 64 octets when the real size is considerably large; the IP over X.25 networks have a 576-octet long MTU, and the Ethernet's MTU is usually 1500 octets. If the host wanted to send 2000-octet long data, after learning the bogus minimum MTU as 64 octets, it would send 32 packets whereas it would only be necessary to send a few packets — e.g. 4 packets if the subnetworks are X.25.

The latter attack is as follows. A bogus host or router could then have control over the routing by pretending it can offer a route with a big MTU; the learned MTU could be used by a source host to decide a better route. Moreover, the MTU of a silent router who does not send this error message could not be learned. A more active discovery scheme is needed in order to learn the real minimum MTU. The active scheme would record the certified MTUs of the networks over the path to the destination and back to the source.

The subnetwork mask is another parameter which is important for routing. By setting the subnetwork mask on a host to an appropriate value, the legal but malicious router, i.e. the router that has an entry in the routing table of a host, could gain control over the packets which, otherwise, are not supposed to go through that router, even though the routing information at the host was maintained securely. Alternatively, detecting a misconfigured subnetwork mask on a host, a bogus host masquerading as a router sends a Redirect message to the host — the bogus router can then have control over the local traffic which would never go through any router if the subnet mask was not misconfigured.

The MTU and subnetwork mask can be certified during the configuration confirmation phase, and the certificates can be assigned to the host. Moreover, the access right to modify those parameters should be given to an authentic management process; authentic management processes are given their authenticators when they start operating.

6.7. Trusted peers

Another type of configuration information which may need to be certified is the identity of trusted peers. In routing information exchange, one has to know which host or router would be a neighbour to send and receive update messages. Application level servers, such as time servers, mail servers, and name servers, have to know the trusted peer servers with which to exchange messages. In a network management system, a management agent has to know other trusted management agents. Having the incorrect peer information, one would trust a deceiver; authentication does not work, because a deceiver can be authenticated as a legal host but not as a trusted peer.

Using our scheme, the trusted peers for a protocol operation are passed to a host in the form of certificates at the end of configuration confirmation. In the case of dynamic discovery of peers, a peer can attach its authenticator at that protocol level to a peer acquisition message. An authenticator includes the protocol ID, version number, any protocol specific configuration information and the protocol level address; the address may include an application port number, a transport protocol ID, and a host's address. The protocol specific information in the Network Time Protocol, for instance, is the number of levels called "stratum" which shows where in the hierarchy of servers the host is classified; a top level server is supposed to be connected to a very accurate physical clock, whereas the lower level servers are synchronised to the higher level servers' time.

6.8. Cryptographic attributes

Authentication at any level depends on the key used to authenticate a peer process; this requires that the key be associated correctly with the requisite process. This requires very great care in the original registration of the key and its association with the Certification Authority. Alternatively, the key pair may be generated by the CA; then great care must be taken in distributing the secret key to the correct entity.

Another example is in the proposed Simple Network Management Protocol authentication framework [Gal91]. Each site in an administrative domain is supposed to have a local database about authentication scheme attributes of the peer SNMP entities to authenticate the peer before each SNMP transaction. The database includes information such as the authentication protocol, its relevant keys, and the keys for confidentiality if required. Using our scheme, those parameters can be distributed to each site in a secure manner when a host is configured to the network.

Therefore, our scheme in Chapter 5 can be used to register and maintain those keys used in authentication. A protocol process is an object in this case. The associated information is its private or public key to be used by an authentication server, or by the peer principals. When the application process is installed on a host, the host will go through the registration procedure. The certificate of the key is either stored in the repository such as the directory or distributed to the authentication servers.

6.9. Conclusion

The scheme we have developed in this thesis is a secure distribution mechanism for network configuration information and keys, secure in a sense that the integrity of information is maintained.

We have applied the use of certificates to the information items which are static but essential for network operations; we have listed those information items in this chapter. They are address resolution information, name resolution information, routing information, network and time protocol parameters, trusted peer information, and cryptographic attributes. By certifying them when a host is configured to the network, one can be sure of the current network configuration. Any invalidity in these information items could cause the network and higher level threats. The most serious threats are those to address resolution, name resolution, routing information, network mask, trusted peers, and cryptographic attributes; a deceiver could have control over packets in various layers, and in case of cryptographic attributes, all application level threats listed in Chapter 4, viz. unauthorised access to hosts, masquerading, unauthorised access to application servers, unauthorised disclosure of information, are all possible.

Time information is important, but the protection of the above information will protect time synchronisation operations as well to great extent. A network-level configuration information item, MTU, can be considered less critical; a false MTU causes more fragmentation. Certifying critical information would give a network manager some degree of security in an evolving environment.

Chapter 7

Conclusion

7.1. Summary of the thesis

In this thesis we have identified the problem of configuration detection in computer networks and discussed the integrity of configuration information used by hosts and routers, mainly of network addresses, and later in Chapter 6 extended it to other types of information, such as routing information, time synchronising attributes, trusted peers, and cryptographic attributes.

Configuration information such as name, address, and routing information is vital to network operations including management operations. Historically, two types of systems have been developed for the maintenance of such configuration information; one is the *information system*, and the other is the *network management system*. Information systems are responsible for maintenance of the information used for network operations. Network management systems are responsible for the maintenance of the network objects. Both types of systems maintain the configuration information about network objects yet they have been traditionally treated separately. However, the generalisation of the information items managed by information systems, together with the growth of the networks in size, has blurred the distinction between these two types of systems.

We have presented how those two types of systems could cooperate together in configuration management activities. We argue that configuration management in computer networks deals with configuration changes. These changes are presented from a network-wide perspective — the location and availability-related changes of network objects. The changes are performed by a set of operations: local change operations at an object site, recognition of a local change, subsequent change operations to other objects which are affected by the original change, and update of associated information. We presented a model of configuration management identifying the required agents and then facilities. The agents required are the network object, its operator, the management system, and the information system. The object is controlled locally by an operator and remotely by the management system. The management system is responsible for recognition of local changes, deciding subsequent changes, and updating information. Through the information systems, the other objects in the network learn a configuration change. As an exercise we tried to incorporate the management information required by the operator into an information system — the Directory. Through this empirical exercise, we realise the need for a mechanism to learn of the currently existing objects in a network — *configuration detection.*

In a given environment one needs to know which objects exist and where they are located. We call the correct acquisition of this knowledge, *Configuration Detection*(CD). In this thesis we were concerned with the computer network environment and network objects such as hosts and routers. CD is observed to be a significant problem when the environment has evolved such that its size in terms of the number of objects is large. However, we argue that the real problem is that CD suffers from *inconsistency* and *invalidity.* Inconsistency arises when there is a difference between the information about an object and its actual current state. It often happens currently that objects can be attached or removed without updating the information about them across the environment. In contrast, invalidity indicates the incorrectness of the registered information and the misconfiguration of a host. It could be due to inadvertent errors or deliberate deceit. It is caused by the fact that there are inadequate validation, verification, and guarantees both for the registered information and the configuration of an object. Inconsistency indicates that the information about the objects was correct at one stage, whereas in the case of invalidity, either the information or the object configuration has never been correct. These shortcomings could lead to threats, to the continued working and privacy of the system. The application level threats considered were unauthorised access to hosts, masquerade, unauthorised access to application servers, and unauthorised disclosure of information. The network level threats considered were unauthorised tampering, unauthorised use of resources (resource stealing), unauthorised traffic generation, and unauthorised disclosure of information. In this thesis we were concerned particularly with the network level threats and especially, unauthorised tampering by unauthorised redirection of packets.

We devised an authorisation system for the address resolution information to protect the network environment from unauthorised redirection of packets caused by incorrect host addressing. A host is required to go through a robust registration process and receive a certificate notarising the network and subnetwork address mapping which will be used for subsequent dynamic address resolution operation; this guarantees that only certified information will be learned by other network hosts.

We described this procedure in terms of the address information flow between the participants in the configuration management model presented in Chapter 3. We added further agent, the Certification Authority (CA), to the original model in Chapter 3. The address flow is as follows. An address is produced and maintained through a set of procedures, viz. object generation, information generation, information registration over the network, information maintenance, and information look-up. Object generation is the admission of a host. Information generation is the naming of the host and the installation of the hardware and software with the name and other parameters to form an operational host. Information registration over the network is to boot the host and make it available to the other hosts on the network. An information system maintains the address and other data attributed to the host and passes them eventually to the information user. In our example environment of the IP over the Ethernet network within a single organisation, information maintenance could be done by an information store such as a Directory or a name server; alternatively it could be done by a dynamic learning system like the Ethernet Address Resolution system and a routing information exchange system. What is missing in the current address flow is the verification of credibility of the address announced by a host. We incorporate a verification feature by having a further two procedures at the beginning of the address flow; the off-line registration of the name and cryptographic attributes of the host during information generation, and the configuration confirmation during the registration over the network. Upon successful confirmation, a certificate of the mapping between the network and subnetwork addresses is issued by the CA and passed to the user through the information system. The information system maintains the certificates. This requires periodic monitoring of the host to detect its removal and expiring appropriate certificates.

We employ a public-key cryptosystem in the production of certificates; i.e. a certificate is encrypted by the CA using the CA's own secret key and decrypted by the user using the CA's public key. We also use a public-key cryptosystem for the maintenance of integrity and authentication of all transactions between the agents of our authorisation system.

The procedures were verified by means of formal analysis using the logic introduced by Burrows, Abadi, and Needham [BAN89].

The viability of the model was discussed in terms of partial systems in case of communication failures. The minimum system was defined as the management system with limited functions plus the information systems. Note that a CA is not included in this mode, so that no certificate will be produced. With the minimum system, one can do registration and maintenance operations in an untrusted way for a limited period. Two other types of partial systems were introduced; the minimum system with CA, and the minimum system with the verification function and CA. Application to the UCL-CS domain and the Internet were examined. Application to UCL-CS was direct and simple. However, for the Internet, we needed to expand the design notions from considering an intra-autonomous system to inter-autonomous systems where we needed to introduce mechanisms for inter-domain trust. We introduced management level gateways between autonomous systems, which would communicate with one another to exchange information required for verification.

The address flow model has been applied to the Ethernet Address Resolution environment where the availability of hosts is learned dynamically, hence, is error-prone, owing to the fact that a device can easily announce an unauthorised address. We showed how a certificate scheme and encryption could be incorporated in the current Ethernet ARP system. Eventually we concluded that we only need authorisation and would not need authentication of the information provider. If an information item was authorised as a truth, it would not matter who claimed that item. The concept of notarisation was backed up quantitatively by the fact that encryption takes ten times as long as decryption in the RSA public-key cryptosystem. In other words, once an information item is authorised, it only has to be decrypted at the ARP operation time. The use of a certificate scheme would make the modification of the Ethernet Address Resolution operation simpler and more robust than using any other encryption-based scheme such as the authentication of a sender, which would require the encryption of packets at the sender, and the acquisition of keys of different hosts at the receiver besides decryption of the packets. In our scheme, the change of address would be managed by means of monitoring and distributing expired messages; i.e. monitoring the availability of hosts and sending expiry messages relating to removed hosts.

The use of certificates in the Ethernet ARP system was justified in terms of extra cost of resources such as memory space, delay, and frequency, incurred by the additional processes. We showed that in our departmental network the average interarrival time of the ARP Request packets which were received by all the hosts was much longer than the service time of secure ARP. Therefore, updating ARP entries with the incoming ARP certificates would be manageable. However, it takes still the order of hundreds milliseconds to process a certificate by software in current systems. The source host may suffer by the delay when there is no ARP entry for a destination host, because it has to know the mapping in real time by sending an ARP Request and processing the associated reply. In that case, we could have two ARP tables, one with entries learned from a clear text part of an incoming ARP packet and another with certified entries. A network process which prefers quicker resolution would take an insecure entry from the former table, while the security conscious network process would wait until the certificate is processed. This situation happens only at the beginning of the several hours interval by setting the longer timeout, several hours, for the certified entries of the ARP table. If there was hardware to decrypt in the order of one or two milliseconds, this extra ARP table would not be needed.

Finally we examined the extensibility of our scheme presenting some subnetwork types other than Ethernet as well as listing other types of information than network addresses to which we could apply the use of certificates; apart from address resolution information, they are name resolution information, routing information, trusted peers, and cryptographic attributes. Indeed, what we have created for the authorisation of an information item, is a secure distribution mechanism for network configuration information; secure in a sense that the integrity and credibility of the information is maintained.

7.2. Contribution of the thesis

This thesis has presented a novel problem which we termed *Configuration Detection*, and provided a solution — an authorisation system. Our main concern was with the computer network environment. We looked particularly at the protection of information critical to network operations from network level threats caused by inconsistency and invalidity, both of which are the shortcomings of current computer networks.

We devised an authorisation system for such critical information. The main emphasis fell upon the registration scheme. The novelty of the scheme lies in the fact that the verification of the configuration information takes place at host registration. As a result of successful verification, a certificate is issued by an authority. Information items are passed to the information systems in certified form, so that the user of the information can be sure of its credibility. In fact, the system we devised is a secure distribution mechanism for information which requires integrity and authenticity.

The security of information systems has been researched widely in terms of data integrity and access control. However, very few looked at the veracity of the information itself. We argue that it is not useful to hold incorrect information in a very secure system. The approach taken in this thesis is to look at the information flow and make it trustworthy in the sense that the credibility of the information is assured. We have applied an authorisation scheme using a cryptographic technique for this. The use of cryptographic techniques in information flow over the computer networks is nothing new. Many applications tend to believe incorrectly that authentication of the source also guarantees the credibility of information that the source has sent out; this is certainly not true. What we need is authorisation by a trusted party, not the source. The authorisation should not be exercised blindly. It should be based on a careful verification and validation mechanism of the information items.

Configuration Detection is a rich problem with much potential in its solutions depending on the objects to be discovered and the nature of the environment. Whether our solution is applicable for any other environment than computer networks is yet to be seen. What we have presented is a very small fraction of this area. None the less, this thesis has laid a stepping stone for the evolution of the research in this new and exciting area.

7.3. Future research

The problem and solution which we have presented both have much room to evolve. We describe several possible topics which could be researched in the future.

The information flow concept could be developed further. There is potential especially for the dissemination of trust of the authority and in information availability.

The use of the certificate scheme is a well-understood way to obtain trustworthy information exchanges. However, there is a price. The central registration scheme would contain a drawback in a distributed processing environment where a more light-weight scheme would be better appreciated. Also, how we could replicate and distribute the trust is an important research area. For the inter-organisational networks, the U.S. Privacy Enhanced Mail (PEM) approach [KeL89] is useful here. In this approach, the security services are signed locally but use policy-driven certification authorities and cross certification to determine the level of credence to put into remote authentications. In cross certification, the certification authorities certify each other by bilateral agreement — this would shorten the certification path, because the certification path would not have to go through a strict CA hierarchy. For the intra-organisational networks, security labels might be useful as well. The security label shows the security characteristics of the source site which announces an information item, so it is up to the receiver whether the information item is trustworthy. This method might be more suitable in a distributed processing environment than the certificate scheme. We need further research into the practical use of security labels with some other attributes, such as probability of the event.

The availability of information is another important topic. When an information item is changed, the propagation delay of the update to the look-up user site is problematic. A good example is a routing information exchange environment. If one node changed its address, it is a problem how fast or appropriately this information can be disseminated.

Moreover, formal specification of information flow is poor at the moment. The Burrows, Abadi, and Needham notation [BAN89] has made the design of authentication protocols much easier; this is their great contribution. However, what we need in the analysis of information integrity is a way to express the state transition of the information items. One idea is as follows. If an Information Generator, NA, gives an information item, X, to the Information Provider, A, the flow can be expressed as follows:

$$NA \rightarrow A : (X)NA$$

A then tells the authority, CA, of the information item:

$$A \rightarrow CA : ((X)NA)A$$

In this way we could express how the information item has been passed through from one agent to another; i.e. a history trace of the information item. The notation has the possibility of expressing the filtering operations.

Trustworthy registration does not prevent hosts from joining silently and talking to each other with their own protocol. This could happen where the subnetwork has an open architecture. We call this phenomenon *resource stealing* because it takes some of the available bandwidth of the subnet. *Resource stealing* disturbs other registered network operations indirectly by deteriorating performance. We may need a mechanism for the detection of these stealers. This could be done by observing low level traffic and examining if any illegal protocol operation is going on.

The scheme was originally designed for an intra-autonomous system environment. We need further research for the inter-autonomous systems. The main problem is the distribution of trust. Possibly we need a management level gateway in each autonomous system. These management level gateways exchange information, verifying objects jointly. One of them belongs to the super manager autonomous system which manages the internet as a whole. The specification of the requirements and functions of such management level gateways is certainly a future research topic.

References

[Acc83] M. Accetta, "Resource Location Protocol," RFC887 , December 1983.

[ACG88] M. Ahmadi, J. H. Chou & G. Gafka, "NetView/PC," *IBM Systems Journal* Vol.27, No.1 (1988), pp.32–44.

[BLP85] E. Balkovich, S. Lerman & R. P. Parmelee, "Computing in Higher Education: The Athena Experience," *Communications of the ACM* Vol.28, No.11 (November 1985), pp.1214–1224.

[Bel89] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite," *ACM Computer Communication Review* Vol.19, No.2 (April 1989).

[BCW90] A. Ben-Arzi, A. Chandna & U. Warrier, "Network Management of TCP/IP Networks: Present and Future," *IEEE Network* Vol.4, No.4 (July 1990), pp.35–43.

[Ben80] C. Bennett, "Clean and Simple," Internal Note 979, September 1980.

[BHS79] E. H. Bersoff, V. D. Henderson & S. G. Siegel, "Software Configuration Management: A Tutorial," *IEEE COMPUTER* (January 1979).

[Bis] M. Bishop, "A Security Analysis of Version 2 of the Network Time Protocol NTP: A Report to the Privacy and Security Research Group," Technical Report PCS-TR91-154, Dept of Mathematics and Computer Science, Dartmouth College.

[BrH81] R. Braden & P. Higginson, "CLEAN AND SIMPLE INTERFACE UNDER MOS," Internal Note 1054, February 1981.

[BrP87] R. Braden & J. Postel, "Requirements for Internet Gateways," RFC1009, June 1987.

[Bra89] R. T. Braden, ed., "Requirements for Internet hosts - communication layers," RFC1122, October 1989.

[Bra9] H. W. Braun, "NSFNET routing architecture," RFC1093 , February 1989 .

[Bra88] H-W Braun, "The new NSFNET backbone network," *Connexions; The Interoperability Report* Vol.2, No.12 (December 1988), pp.6–9.

[BrE90] L. Breslau & D. Estrin, "Design of Inter-Administrative Domain Routing Protocols," Technical Report 90-8, 1990, Computer Network and Distributed Systems Laboratory, Dept. of Computer Science, University of Southern California.

[Bro89] D. Brownell, "Dynamic RARP Extensions for Automatic Network Address Acquisition," RFC draft, September 1989.

[BAN90] M. Burrows, M. Abadi & R. Needham, "Rejoinder to Nessett," *ACM Operating Systems Review* Vol.24, No.2 (April 1990), pp.39–40.

[BAN89] Michael Burrows, Martin Abadi & Roger Needham, "A Logic of Authentication," DEC Systems Research Center Technical Report No.39, February 1989.

[CCI88a] CCITT & ISO, "Recommendations X.500 series; The Directory - X.509 (ISO 9594-8) Authentication Framework;," March 1988.

[CCI88b] CCITT & ISO, "Recommendations X.500 series; The Directory - X.500 (ISO 9594-1) Overview of Concepts, Models and Services; X.501 (ISO 9594-2) Models; X.509 (ISO 9594-8) Authentication Framework; X.511 (ISO 9594-3) Abstract Service Definition; X.518 (ISO 9594-4) Procedures for Distributed Operation; X.519 (ISO 9594-5) Protocol Specifications; X.520 (ISO 9594-6) Selected Attribute Types; X.521 (ISO 9594-7) Selected Object Classes," March 1988.

[CCN87] R. Callon, K. Corker, M. Nodine, J.Ong, M. Stillman & J. Westcott, "Disciplines for Effective Network Management," *Proc. of IEEE Military Communications Conference* Vol.1 (1987), pp.1.4.1–1.4.8.

[CFS89] J. Case, M. Fedor, M. Schoffstall & J. Davin, "A Simple Network Management Protocol (SNMP)," RFC1098, April 1989.

[Cas88] J. D. Case, "Personal Message, Subject: Configuration Detection," July 1988.

[CDF88] J. D. Case, J. R. Davin, M. S. Fedor & M. L. Schoffstall, "Introduction to the Simple Gateway Monitoring Protocol," *IEEE Network* Vol.2, No.2 (March 1988), pp.43–49.

[CeC83] V. G. Cerf & E. Cain, "The DoD Internet Architecture Model," *Computer Networks* (July 1983).

[CeK78] V. G. Cerf & P. T. Kirstein, "Issues in Packet-Network Interconnection," *Proceedings of the IEEE* Vol.66, No.11 (November 1978), pp.1386–1408.

[ChM89] D. R. Cheriton & T. P. Mann, "Decentralizing a Global Naming Service for Improved Performance and Fault Tolerance," *ACM Transactions on Computer Systems* Vol.7, No.2 (May 1989), pp.147–183.

[Cla89] D. Clark, "Policy Routing in Internet Protocols," RFC1102, May 1989.

[ClW87] D. Clark & D. Wilson, "A Comparison of Commercial and Military Computer Security Policies," *Proc. of IEEE Symposium on Security and Privacy* (April 1987), Oakland, California.

[Col82] R. Cole, "User Level Availability Monitoring," 1982.

[CDH84] R. Cole, T. Daniel, P. Higginson, S. Kille, P. Kirstein, A. Lawson, P. Lloyd & R. Moulton, "Network Interconnection Facilities at UCL," Internal Note 1588, May 1984, presented to the ICCC 84 at Sydney, Australia, September 1984.

[CoK82] R. Cole & S. Kelly, "A Catenet Probe, Reporting and Monitoring System," University College London, Internal Note 1281, May 1982.

[Com88] D. Comer, *Internetworking with TCP/IP; Principles, Protocols, and Architecture*, Prentice-Hall, 1988.

[CrG85] W. J. Croft & J. Gilmore, "Bootstrap Protocol ," RFC951 , September 1985.

[DIX82] DEC, INTEL & XEROX, "The Ethernet: A Local Area Network, Data Link Layer and Physical Layer Specifications," AA-K759B-TK, November 1982, Version 2.0.

[DaP81] Y. Dalal & R. Printis, "48-bit Absolute Internet and Ethernet Host Numbers," *Conf. Proc. of IEEE Seventh Data Communication Symposium* (1981).

[Dee88] S. E. Deering, "Multicast Routing in Internetworks and Extended LANs," *Conf. Proc. of SIGCOMM '88 Symposium at Stanford, California* (August 1988).

[DEF88] C. A. DellaFera, M. W. Eichin, R. S. French, D. C. Jedlinsky, J. T. Kohl & W. E. Sommerfeld, "The Zephyr Notification Service," *Proc. of USENIX Winter Conference* (February 1988), Dallas, Texas.

[Den83] D. Denning, *Cryptography and Data Security*, Addison-Wesley Publishing Company, January 1983, ISBN0-201-10150-5.

[DeS81] D. E. Denning & G. M. Sacco, "Timestamps in Key Distribution Protocols," *Communications od the ACM* Vol.24, No.8 (August 1981), pp.533–536.

[DiH76] W. Diffie & M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory* Vol. IT-22, No.6 (November 1976).

[DoD83] Department of Defense, U.S.A (DoD), "Military Standard - Internet Protocol," MIL-STD-1777, August 1983.

[Dye88] S. P. Dyer, "The Hesiod Name Server," *Proc. of USENIX Winter Conference* (February 1988), Dallas, Texas.

[ECM85] ECMA, "OSI Directory Access Service and Protocol," TR32, December 1985.

[Est85] D. Estrin, "Access to Inter-organization Computer Networks," MIT/LCS/TR-345, August 1985, a Ph.D. Thesis, Laboratory for Computer Science, Massachusetts Institute of Technology.

[EMT88] D. Estrin, J. Mogul, G. Tsudik & K. Anand, "Visa Protocols for Controlling Inter-Organizational Datagram Flow: Extended Description," TR-88-50, 1988, Computer Science Department, University of Southern California, Los Angeles, California 90089-0782.

[EsT89] D. Estrin & Gene Tsudik, "Security Issues in Policy Routing," *Proc. of IEEE Symposium on Security and Privacy* (May 1989), Oakland, California.

[FaS91] S. O. Falaki & S-A Sorensen, "Traffic measurement on a local area computer network," UCL Research Note RN/91/58, July 1991.

[FLN88] M. Feridun, M. Leib, M. Nodine & J. Ong, "ANM: Automated Network Management System," *IEEE Network* Vol.2, No.2 (March 1988), pp.13–19.

[FSW81] E. B. Fernandez, R. C. Summers & C. Wood, *Database Security and Integrity*, Addison-Wesley Publishing Company, 1981, ISBN 0-201-14467-0.

[Fin84] R. Finlayson, "Bootstrap loading using TFTP," RFC906 , June 1984.

[FMM84] R. Finlayson, T. Mann, J. Mogul & M. Theimer, "A Reverse Address Resolution Protocol," RFC903, June 1984.

[Gal91] J. Galvin, "SNMP Administrative Model," Internet-Draft, Aptil 1991.

[GoS86] I. S. Gopal & A. Segall, "Dynamic Address Assignment in Broadcast Networks," *IEEE Transaction on Communications* Vol. COM-34, No.1 (January 1986), pp.31–37.

[Hal60] P. R. Halmos, "Naive Set Theory," 1960, The University Series in Undergraduate Mathematics.

[HSF85] K. Harrenstien, M. Stahl & E. Feinler, "DOD Internet Host Table Specification," RFC952, October 1985.

[Hau86] B. Hauzeur, "A Model for Naming, Addressing, and Routing," *ACM Transactions on Office Information Systems* Vol.4, No.4 (October 1986), pp.293–311.

[Hel87] M. Hellman, "Commercial Encryption," *IEEE Network Magazine* Vol.1, No.2 (April 1987), pp.6–10.

[Hew87] Hewlett-Packard, "Operating Manual For The HP 4971S LAN Protocol Analyzer," Manual Part No. 18211-90011, March 1987.

[HHS83] R. Hinden, J. Haverty & A. Sheltzer, "The DARPA Internet: Interconnecting Heterogeneous Computer Networks with Gateways," *COMPUTER (IEEE)* Vol.16, No.9 (September 1983), pp.38–48.

[HiS82] R. Hinden & A. Sheltzer, "THE DARPA INTERNET GATEWAY," RFC 823, September 1982.

[HGK86] C. Huitema, S. Goss, S. Kille, G. Michaelson & A. Foster, "THORN Year 2 General Specifications," INRIA-10 Rev.2, October 1986, THe Obviously Required Name-server (THORN).

[IEE87] IEEE, "Draft IEEE Standard 802.1: Part D, MAC Bridges," Revision C, August 1987, IEEE Project 802 Local and Metropolitan Area Network Standards.

[ISO84] ISO, "International Standard ISO 7498 Information processing systems - Open Systems Interconnection - Basic Reference Model," ISO 7498, 1984.

[ISO86] ISO, "Protocol for Providing the Connectionless-mode Network Service," ISO 8473, 1986, Draft Standard.

[ISO89a] ISO, "Information technology — Telecommunications and information exchange between systems — OSI Routeing Framework," ISO/TR 9575, 1989.

[ISO89b] ISO, "Information Processing - Open System Interconnection - Management Information Protocol Specification - Part 2: Common Management Information Protocol," DIS 9596, 1989, Draft International Standard.

[ISO89c] ISO, "Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management framework," INTERNATIONAL STANDARD ISO/IEC 7498-4, 1989.

[ISO89d] ISO, "Information processing systems - Open Systems Interconnection - Systems Management Part 2: State Management Function ," International Standard ISO/IEC DP 10164-2, January 1989.

[ISO89e] ISO, "Information Processing - Open System Interconnection - Systems Management Overview," ISO/DP 10040, September 1989, Draft Proposed International Standard.

[JNT82] Joint Network Team (JNT), *Cambridge Ring 82 Protocol Specifications*, the Computer Board and Research Councils, Nov. 1982.

[JuT87] J. Jubin & J. J. D. Tornow, "The DARPA Packet Radio Network Protocols," *proceedings of the IEEE* (January 1987).

[Kan88] D. Kanyuh, "An integrated network management product," *IBM Systems Journal* Vol.27, No.1 (1988), pp.45–59.

[Kat90] D. Katz, "Proposed standard for the transmission of IP datagrams over FDDI networks," RFC1188, October 1990.

[KeM87] C. A. Kent & J. C. Mogul, "Fragmentation Considered Harmful," *Proceedings of SIGCOMM'87 Workshop* (August 1987).

[KeL89] S. Kent & J. Linn, "Privacy Enhancemenet for Internet Electronic Mail: Part II — Certificate-Based Key Management," RFC1114, August 1989.

[KBC79] P. T. Kirstein, C. Bradbury, R. C. Cringle, H. R. Gamble, R. Binder, D. McNeill, J. Cole, R. Bressler & D. L. Mills, "SATNET Applications Activities," *Conference Record; National Telecommunications Conference (NTC'79), Washington, DC* Vol.3 (November 1979), pp.45.5.1–45.5.7, IEEE Catalog Number 79CH1514-9.

[KlK77] L. Kleinrock & F. Kamoun, "Hierarchical Routing for Large Networks: Performance Evaluation and Optimization," *Computer Networks* Vol.1, No.3 (1977), pp.155–174.

[Kni89] G. Knight, "The INCA Network Management System," *Connexions; The Interoperability Report* Vol.3, No.3 (March 1989), pp.27–32.

[KPW88] G. Knight, G. Pavlou & S. Walton, "An OSI Network Management System," November 1988, presented at ESPRIT Technical Week, Brussels.

[KMW84] Graham Knight, Zonghou Ma & Steve Wilbur, "The University College Status and Alarm System," University College London, TR104, September 1984, 395/UCL/85/11.

[KrM90] J. Kramer & J. Magee, "The Evolving Philosophers Problem: Dynamic Change Management," *IEEE Transactions on Software Engineering* Vol. SE-16, No.11 (November 1990), pp.1293–1306.

[LaB90] L. LaBarre, "OSI Internet Management: Management Information Base," Internet-Draft, May 1990, the OSI Internet Management Working Group.

[Lai88] W. S. Lai, "Packet Forwarding," *IEEE Communication Magazine* Vol.26, No.7 (July 1988), pp.8–17.

[Lam78] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications od the ACM* Vol.21, No.7 (July 1978), pp.558–565.

[Lot87] M. Lottor, "Domain administrators operations guide," RFC1033 , November 1987.

[LKV85] W. M. Loucks, W. I. Kwak & Z. G. Vranesic, "Implementation of a Dynamic Address Assignment Protocol in a Local Area Network," *Proc. 10th Conference on Local Computer Networks* (October 1985), sponsored by IEEE.

[LoR90] K. Lougheed & Y. Rekhter, "Border Gateway Protocol (BGP)," RFC1163 , June 1990.

[Man90] U. Manber, "Chain Reactions in Networks," *IEEE COMPUTER* Vol.23, No.10 (October 1990), pp.57–63.

[MaO85] K. Marzullo & S. Owicki, "Maintaining the Time in a Distributed System," *Operating Systems Review* Vol.19, No.3 (July 1985), pp.44–54.

[McQ74] J. M. McQuillan, "Adaptive routing algorithms for distributed computer networks," BBN Report No. 2831, May 1974, Ph. D. Thesis, Harvard University.

[MRR80] J. M. McQuillan, I. Richer & E. C. Rosen, "The New Routing Algorithm for the ARPANET," *IEEE Transactions on Communications* Vol. COM-28, No. 5 (May 1980), pp.711–719.

[Mei87] A. Meijer, *Systems Network Architecture: a tutorial*, Pitman, 1987, ISBN 0-273-02842-1.

[Mes88] the tcp-ip mail list Messages, "Subject: ICMP's and IP source address," September 1988, discussion at tcp-ip@sri-nic.arpa on Broadcast Ping .

[MNS87] S. P. Miller, B. C. Neuman, J. I. Schiller & J. H. Salzer, "Kerberos Authentication and Authorization System," December 1987, M.I.T. Project Athena Technical Plan, Section E.2.1.

[Mil89] D. L. Mills, "Internet time synchronization: The Network Time Protocol," RFC 1129, October 1989.

[Moc87a] P. Mockapetris, "Domain names - implementation and specification," RFC1035 , November 1987.

[Moc87b] P. Mockapetris, "Domain names - concepts and facilities," RFC1034 , November 1987.

[MoD90] J. Mogul & S. Deering, "Path MTU Discovery," RFC1191, November 1990.

[MoP85] J. Mogul & J. Postel, "Internet Standard Subnetting Procedure," RFC950, August 1985.

[Moo81] D. Moon, "Chaosnet," A.I. Memo No.628, June 1981, Obtainable from Artificial Intelligence Laboratory, Publications, Massachusetts Institute of Technology, 545 Technology Square, Room 818, Cambridge, MA 02139, U.S.A..

[Moo88] R. E. Moore, "Utilising the SNA Alert in the management of multivendor networks," *IBM Systems Journal* Vol.27, No.1 (1988), pp.15–31.

[Mur85a] Y. Murayama, "The Status Report by the CN Probe and the SAS Probe," University College London, Internal Note 1804, August 1985, .

[Mur85b] Y. Murayama, "An Introduction to the Status Alarm System," Internal Note 1667, January 1985.

[Mur87] Y. Murayama, "The information Available in the DARPA Internet Layer," Internal Note 2214 Rev.1, November 1987, 395/UCL/87/42.

[MuC88] Y. Murayama & R. Carbonell, "Network Management; its requirements (1) The Required Configuration Information for Systems' Maintenance," Internal Note 2245, March 1988.

[MKM88] Y. Murayama, G. Knight & J. Malcolm, "The Scope of Network Configuration Management," in *Issues in LAN Management: Proceedings of the IFIP TC6/W6.4A Workshop on LAN Management, West Berlin, 2-3 July 1987*, I. N. Dallas & E. B. Spratt, eds., North-Holland, 1988, 131–146.

[MuM85] Y. Murayama & Z. Ma, "The Service Probe," University College London, Internal Note 1850, November 1985, 395/UCL/86/13.

[NeS78] R. Needham & M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM* Vol.21, No.12 (December 1978), pp.993–999.

[Nes90] D. M. Nessett, "A Critique of the Burrows, Abadi, and Needham Logic," *ACM Operating Systems Review* Vol.24, No.2 (April 1990), pp.35–38.

[Now90] Bill Nowicki, "Personal Message, the discussion about control on dissemination of local address information," February 1990, Message-Id: 9002052134.AA29114@Legato.COM.

[Par86] C. Partridge, "Mail Routing and the Domain System," RFC974, January 1986.

[PaT87a] C. Partridge & G. Trewitt, "HEMS VARIABLE DEFINITIONS," RFC 1024, October 1987.

[PaT87b] C. Partridge & G. Trewitt, "THE HIGH-LEVEL ENTITY MANAGEMENT SYSTEM (HEMS)," RFC 1021, October 1987.

[Pat89] M. Patton, "Personal Message: Addressing in the Chaosnet," May 1989, the discussion at the tcp-ip@nic.ddn.mil .

[PKW89] G. Pavlou, G. Knight & S. Walton, "A Guide to the UCL Network Management System," Internal Note 2397, July 1989.

[Per85] R. Perlman, "Hierarchical Networks and the Subnetwork Partition Problem," *Computer Networks and ISDN Systems* Vol.9, No.4 (April 1985), pp.297–303.

[Per91] R. Perlman, "Personal Message, the discussion about the doubled traffic cause by the misconfiguration of a host," August 1991.

[PiL91] D. M. Piscitello & J. Lawrence, "Transmission of IP datagrams over the SMDS Service," RFC1209, March 1991.

[Plu82] D. Plummer, "An Ethernet Address Resolution Protocol," RFC826, November 1982.

[Pos81a] J. Postel, "INTERNET PROTOCOL - DARPA Internet Program Protocol Specification," RFC791, September 1981.

[Pos81b] J. Postel, "Internet Control Message Protocol," RFC792, September 1981.

[PoR88] J.B. Postel & J.K. Reynolds, "Standard for the transmission of IP datagrams over IEEE 802 networks," RFC1042, February 1988.

[Pou85] L. Pouzin, "Internetworking," in *Computer Communications. Volume 2: systems and applications*, W. Chou, ed., Prentice-Hall, 1985, Chapter 15.

[Rek89] J. Rekhter, "EGP and policy based routing in the new NSFNET backbone," RFC1092 , February 1989.

[Rey88] J.K. Reynolds, "BOOTP vendor information extensions," RFC1084 , December 1988.

[RSA78] R. Rivest, A. Shamir & L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM* Vol.21, No.2 (February 1978), pp.120–126, MIT Laboratory for Computer Science and the Department of Mathematics.

[RoW70] L. G. Roberts & B. D. Wessler, "Computer network development to achieve resource sharing," *AFIPS Conf. Proc.: 1970 Spring Joint Computer Conference* Vol. 36 (May 1970), pp. 543–549.

[RSR88] S. Romano, M. Stahl & M. Recker, "Assigned Numbers," RFC1062, August 1988.

[RoM88] D. B. Rose & J. E. Munn, "SNA network management directions," *IBM Systems Journal* Vol.27, No.1 (1988), pp.3–14.

[Ros90] M. Rose, ed., "Management Information Base for Network Management of TCP/IP-based internets: MIB-II," RFC1158 , May 1990.

[Ros82] E. C. Rosen, "Exterior Gateway Protocol (EGP)," RFC827, October 1982.

[RJL88] M. A. Rosenstein, D. E. Geer, Jr. & P. J. Levine, "The Athena Service Management System," *Proc. of USENIX Winter Conference* (February 1988), Dallas, Texas.

[SUN88] SUN, "Network Bootstrap Loading," *Sun Technology* (Summer 1988).

[Sal83] T. P. Sallivan, "Communications Network Management enhancements for SNA networks: An overview," *IBM Systems Journal* Vol.22, No.1/2 (1983), pp.129–142.

[ScS80] M. Schwartz & T. E. Stern, "Routing Techniques Used in Computer Communication Networks," *IEEE Transactions on Communications* Vol. COM-28, No.4 (April 1980), pp.539–552.

[SeR84] L. J. Seamonson & E. C. Rosen, "STUB Exterior Gateway Protocol," RFC 888, January 1984.

[Sha49] C. Shannon, *The Mathematical Theory Of Communication*, The University Of Illinois Press: Urbana, 1949.

[Sho78] J. Shoch, "Inter-Network Naming, Addressing, and Routing," *Conf. Proc. of IEEE COMPCON Fall 1978* (1978).

[SiM90] A. D. Singh & S. Murugesan, "Fault-Tolerant Systems," *Computer* Vol.23, No.7 (July 1990), pp.15–17, Guest Editors' Introduction.

[Slo84] M. Sloman, "Part 2 of Final Report of COST 11 BIS Local Area Network Group," October 1984.

[Sol85] K. R. Sollins, "Distributed Name Management," MIT/LCS/TR-331, 1985, Technical Report, Laboratory for Computer Science, Massachusetts Institute of Technology.

[Sta87] M. K. Stahl, "Domain administrators guide," RFC1032 , November 1987.

[Ste91] J. G. Steiner, "Personal Message: an enquiry about time synchronisation," May 1991.

[SNS88] J. G. Steiner, B. C. Neuman & J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," January 1988, presented at Winter USENIX 1988, Dallas, TX.

[Sun77] C. Sunshine, "Interconnection of computer networks," *Computer Networks* Vol.3, No.1 (January 1977), 175–195.

[Tre88] G. W. Treese, "Berkeley UNIX on 1000 Workstations: Athena Changes to 4.3BSD," *Proc. of USENIX Winter Conference* (February 1988), Dallas, Texas.

[WBB85] J. Westcott, J. Burruss & V. Begg, "Automated Network Management," *Proc. of IEEE INFOCOM '85* (March 1985).

[WCM89] S. Wilbur, J. Crowcroft & Y. Murayama, "MAC Layer Security Measures in Local Area Networks," Research Note RN/89/43, May 1989, presented at the LANSEC 89 Workshop held at EISS, Karlsruhe, April 1989.

[WiM88] M. Willett & R. D. Martin, "LAN Management in an IBM Framework," *IEEE Network Magazine* Vol.2, No.2 (March 1988), pp.6–12.

Appendix A

A Review of Configuration Detection Methods

A.1. Overview

This appendix is referred by Chapter 4 in which we mention two general methods of configuration detection; one is to enquire from a registered information base, the other query the objects through a network. Here we review some of the latter methods used to find different types of objects in computer networks.

Configuration detection has been practised in various network operations. In principle, the following four types of method are used:

1. manual input
2. listening to the traffic
3. unsolicited reports
4. query

In each case a Configuration Monitoring Entity (CME) gathers information from manual input in case of type 1 or from incoming packets. Although amongst the above, type 1 is the most primitive way, it is also surprisingly the most popular way in network and systems management.

In type 2, the CME listens for a certain type of protocol packets passing through the network; the existence of the protocol entities are deduced from the source addresses of the packets. Type 3 is a kind of listening; here the reporting packets are destined explicitly to the CME for the learning purpose. In type 4 the CME learns of the existence of objects by means of a request and reply type of protocol.

In the following subsections, we review examples of use of the above methods for the different types of network objects. The next section describes an example of type 1 configuration detection in a switch which connects a set of terminals and some computers. Section A.3 discusses a type 2 detection system, which takes examples of the HP Ethernet analyser and an IEEE802 type bridge. The following two sections present activity examples of types 3 and 4; section A.4 provides an insight into routing operations, particularly in the DARPA Internet and the ISO Connectionless Internet, and section A.5 reports the use of various network management protocols, such as the Reverse Ethernet Address Resolution Protocol, the Bootstrap Protocol, the Resource Location Protocol, the Simple Network Management Protocol, and the Internet Control Message Protocol.

A.2. Learning by manual inputs

A.2.1. The Departmental Terminal Access Network

The Terminal Access Network (TAN) at UCL-CS consists of a data switch with serial links to terminals and host computers. The objective of the network is to divert a substantial proportion of the terminal traffic from the Departmental LANs, which are Ethernets. The TAN has a gateway to the Departmental backbone LAN via a terminal multiplexor; this provides a terminal user with an alternate path to a host.

When a new terminal is added to the network, the following procedures take place:

1. install the connection between the data switch and a terminal.
2. register the new connection to the data switch
3. update a file of information about the configuration of the data switch

Installation includes determining a port in the data switch. Each port is identified by a module number and the port number in that module. In fact, the switch port plays a role of the Subnetwork Point of Attachement (SNPA) in the TAN. Selecting a port, therefore, decides automatically the subnetwork address of the terminal.

The switch would not recognise a new connection, even if it exists physically, until the switch is configured so by a human operator from the console station. Registration information includes address, attributes of the communication link, and accessibility to the host session level services from the newly connected terminal.

Updating the information file has a different function, namely for a human operator to keep track of the physical configuration. The information includes the attributes of the interface in the switch (like the choice of DTE/DCE of the port of the switch), and the attributes of the destination, to which the interface is connected (such as a socket identifier in a room to which the terminal is attached), This information cannot be obtained from the console of the switch, but is required for systems maintenance. The information is often out of date; partly this is due to a human input and is used by a human operator only, but mainly because it is not required during actual network operation — only during failure conditions or when reconfiguration is needed. If the object added is a host or more intelligent equipment, we will need a similar procedure at the end site as well.

A remote user from the outside the department can have access to some hosts. In that case, the object added is a user, and this must be registered at the data switch, otherwise the user cannot be recognised by the switch and the access will be rejected in the log-on phase.

The current configuration information of the switch can be interrogated at the console station, which is attached to a special port of the switch. If the console were more intelligent than a terminal (or printer), and had another interface to some network, then it could export this information to the remote manager. Providing the end sites had similar intelligence, we could probably produce the full detail of the connections of the network: i.e. which end system is connected to which switch port.

Some attributes of a terminal connected to the switch can be varied, and so can be detected only dynamically on a session basis by a host. This information is partially input by the user of the terminal at the session, and partially inferred automatically (e.g. speed). While theoretically the attribute information could be associated with the port, this is not done in practice; a user at the same terminal, who initiates another session, has to specify the terminal type again even with the same host — though in many hosts a preferred default terminal is associated with the user login unless overridden at the start of the session.

A.3. Learning by listening

A.3.1. The HP LAN Protocol Analyzer

The HP 4971S, a LAN protocol analyzer, detects the addresses of Ethernet stations by listening to the Ethernet frame level traffic [Hew87]. It extracts the source and destination addresses from a frame and creates the table of the stations. As the analyzer listens to the traffic on the Ethernet to which it is attached, it can detect only the stations on the other sides of the Media Access Control (MAC) level bridges as long as they send and receive the frames to and from the local Ethernet.

By specifying the filter, the higher level protocol packets, such as DARPA IP packets, can be derived. Thus the table of the higher level protocol entities can be created. At the MAC level, such a procedure cannot be used directly; there is no way to know which station is on which Ethernet, as the bridges and repeaters are transparent. Only the higher level protocol entity addresses indicate the physical Ethernet, and thus can be used to provide the information. In case of the DARPA Internet protocol, separate subnet addresses [MoP85] could be assigned to each physical net. In our departmental LANs, however, the subnetting is done according to administrative domains or access control groups. There is therefore no association with physical Ethernets.

A.3.2. The Ethernet Address Resolution Protocol

It is not specified in the protocol specification [Plu82], but, in practice, a host on the Ethernet learns the DARPA IP and Ethernet mappings by observing a source address of an Ethernet ARP Request packet which is broadcast.

A.3.3. The IEEE 802 MAC level bridges

A Media Access Control level bridge specified in the IEEE standard [IEE87] observes the source addresses of transit packets, and learns to which link the bridge should forward packets. In this way, the forwarding data base is created dynamically.

A.4. Network Layer Routing

A.4.1. Overview

Configuration detection has been a fundamental discipline for routing in packet-switched networks. Detected objects are network layer entities in packet switches and links between them. The means are provided to detect those objects, such as neighbour acquisition and routing update protocol.

Routing operations include routing control and traffic forwarding. Because of our interest in configuration detection, the latter is relevant. Routing control consists of the following activities [Lai88], [ISO89a]:

1. routing information collection (e.g. measurement)
2. information distribution
3. route calculation and maintenance

It is the first two above, i.e. routing information collection and information distribution, with which a node in a network acquires the knowledge of the network configuration. From routing information collection, a node can see the available nodes near by, and from information distribution, it is informed of the reachable destinations.

In the following subsections, we describe these configuration detection aspects of routing operations. The next subsection presents routing information exchange in the DARPA Internet as an example of gateway-gateway configuration detection. Subsection A.4.3 and Subsection A.4.4 describe the ISO and DARPA IP protocols respectively for a host to find available gateways.

A.4.2. The Internet

The Internet has developed out of the original DARPA Internet, but has broadened its scope over the years. Much of the argument in this section is based on the DARPA Internet portion, which was strongly managed. The current growth is more democratic in nature. There is now a range of routing and management procedures, so that there is a serious inter-operability and systems management problem.

The DARPA Internet gateways originally used the Gateway-Gateway Protocol (GGP) for routing information exchange [HiS82], while the gateways were relatively homogeneous in terms of their hardware and software, and their number was relatively small. As more networks have joined the Internet, the number of gateways connecting them has increased. The heterogeneity of their software and hardware has increased as new gateways often have radically different software [SeR84], and they come from a wider range of vendors [PaT87b]. The GGP was neither flexible enough to cope with the size and heterogeneity of the Internet gateways, nor specified sufficiently tightly enough for multivendor inter-operability.

The Exterior Gateway Protocol was specified to remedy this situation. In this scheme, the Internet System is regarded as being composed of a number of Autonomous Systems, each of which has a separate administration on routing. In other words, each Autonomous System has its own routing scheme which may differ from that in another Autonomous System. One Autonomous System could make use of others as intermediate hops to the final destination of the traffic. The system is loosely configured as a federated system [Sol85]; its architecture is influenced strongly by the federal nature of the funding agencies. Thus the Defense Research portion of the Internet System is divided in a hierarchical manner, with an Autonomous System of specific gateways on top; these are called core gateways and managed by BBN exclusively. Other Autonomous Systems are connected to the core gateways as child nodes by means of stub gateways. Core gateways exchange routing information by GGP internally and now using another protocol; a core gateway and a stub gateway exchange the information by EGP.

The functions of EGP includes the followings [Com88]:

1. acquisition of tactically selected neighbours
2. testing neighbour reachability
3. advertising network reachability to neighbours

The protocol does not specify which system is to be configured as neighbour; this needs human intervention. The human decision maker, therefore, is supposed to have a knowledge of neighbours. Neighbour reachability is tested by Hello and I-Heard-You messages exchanges. By advertisement of reachable networks from neighbours, a gateway can obtain the Internet structure outside its Autonomous System. An Autonomous System can obtain its internal configuration by an appropriate internal routing information exchange protocol.

In regard to configuration detection, EGP gives a mechanism to provide a knowledge of networks and gateways to other Autonomous Systems. Intra-Autonomous System configuration has to be detected by other mechanisms, such as human input or another protocol used in that Autonomous System. Other Autonomous Systems are also connected in with quite different protocol suites. For example the collection of the NSFNET backbone and the Regional Networks, funded partially by the National Science Foundation, have a similar philosophy — but based on a core managed by MERIT Inc., which is a membership consortium of some universities in the State of Michigan, U.S.A., with support from IBM and MCI. The resulting interoperability problems are currently being resolved.

A.4.3. A subnet of an ISO Connectionless mode Internet

In an ISO connectionless mode internet, intermediate and end systems use the connectionless mode protocol [ISO86(DIS 8473)] for internetworking operation. Routing information is exchanged by different protocols. For example, there is one for use between some of the intermediate systems in the Internet. There is another for use between the end systems and the intermediate systems [ISO 9542] in a subnet, which allows a system to know about the existence of other systems. In this protocol, an end system multicasts a Hello message to all the intermediate network entities periodically, and so does an intermediate system to all the end systems. A Hello message from an end system carries the information on available network ports (NSAPs) for each subnet port (SNPA) in the end system. A Hello message from an intermediate system conveys the information on the network entity title of the intermediate system. The current configuration of a subnet can be detected by listening to those Hello messages. A system can listen to those messages by enabling its multicast address ports. This suite is used by some of the Autonomous Systems in the Internet.

A.4.4. The Gateway Discovery Protocol

Gateway discovery, a mechanism to enable a host to locate at least one operational router for the traffic destined to the outside of the subnet where the host resides, is a problem. Discovery is done by use of the specified pair of ICMP messages, a *Gateway Query* and a *Gateway Report.* At boot time, a host multicasts a *Gateway Query* and routers reply with *Gateway Reports.* This is a type 4 learning. Moreover, the routers periodically multicast unsolicited *Gateway Reports.* By listening to those reports, a host can learn the operational gateways. This is a type 3 learning.

A.5. Use of network management protocols

A.5.1. Overview

Network management has become of great interest to the internetworking community [PaT87b] [Kni89]. This is due to the growth and increasing heterogeneity of the Internet environment. Each domain manager wants to manage the network objects in a unified way. The Simple Network Management Protocol (SNMP) was specified from this perspective. Another more powerful suite called the Common Management Information Protocol (CMIP) [ISO89b(DIS 9596)] or the CMIP Over TCP/IP (CMOT) [BCW90] is the subject of international standardisation. It is less complete at this time than SNMP, but parts of it are used also in the Internet.

Many protocols are available for the management of the specific objects. Some of them are intended for a configuration detection purpose. In this section, we introduce use of those existing protocols for detection of objects.

A.5.2. The Reverse Ethernet Address Resolution Protocol

The Reverse Ethernet Address Resolution Protocol (RARP) was produced for a disc less workstation to acquire the knowledge of its own network address [FMM84]. It is used for bootstrap operations [Fin84], [SUN88]. When a system starts operating, it sends a RARP request with its link address by multicast to a specific group of servers, or most probably by broadcast. A server replies to the requester with the requester's network address. The recent extension of this protocol, the Dynamic RARP (DRARP), is a superset of RARP and designed for dynamic addressing [Bro89]. The server will provide the requester with a temporary network address which lasts for an hour, if there is no registered address previously assigned for the requester. In either protocol, a system has to know the link address.

A.5.3. The Bootstrap Protocol (BOOTP)

The Bootstrap Protocol is designed for a disc less client host to obtain the knowledge of its configuration information for the network operation [CrG85] The information includes host's network address, the address of a server, and the location and name of the boot file. With the extension, various servers' information including the information on the gateways can be obtained [Rey88]. Using BOOTP is a type 4 learning; it is essentially a request and reply protocol. The request is broadcast.

A.5.4. The Resource Location Protocol

The Resource Location Protocol (RLP) is for a host to find out the network addresses of the servers [Acc83]. It is essentially a request and reply protocol. Thus the host can exercise type 4 learning of servers' addresses. The host has to have a previous knowledge of what servers exist.

A.5.5. The Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) is produced to unify the management transaction between a manager and a managed object [CFS89]. It is possible to use the protocol so that a network operation centre to inquire of gateways about their configuration and their nearest neighbours [Cas88], [CDF88]. This way, the operation centre learns the network map of the gateways.

A.5.6. Broadcast Ping

Ping is a program which sends out an ICMP Echo Request to a certain destination and receives an ICMP Echo Reply. In this way, we can keep track of active sites. An interesting aspect is that we can specify the network broadcast address as a destination, so that one can learn the existing objects with zero knowledge [Mes88]. At present there is a problem in this scheme. Because of an implementation bug, some hosts reply back to the broadcast address; from these hosts, the network would be flooded by the requests. Some distributed operations, such as the Network File Store operation, would have to wait longer than their time-out limits. The whole network operation, therefore, is likely to become dead-locked. However, if no such hosts existed with this bug, and the network is relatively quiet, it would be a strong tool for type 4 configuration detection.

A.5.7. Try-and-see Ping

The try-and-see ping is an idea presented to the tcp-ip mail list by Wood [Mes88], who had opposed the broadcast ping because of its disturbance of the network by flooding. The idea is to learn the existing network objects with zero knowledge by sending an ICMP Echo Request to all the possible addresses. For the detection of non-existing objects, the time-out mechanism is used. According to our experiment, it takes constantly around 40 minutes to exhaust 255 addresses, no matter how many hosts actually exist. That is to say, for any subnet with a 255-address space, it would take 24 hours to try 36 subnets. Obviously it does not scale very well.

This is a type 4 technique, and it could be exercised on any layer by employing any other request/reply type of protocol including an application-level management protocol such as SNMP and CMIP.

Appendix B

The birth registration in Wuhan, China

In this appendix, we examine birth registration, which is the most basic registration in our society. Many other trusted operations are based on the birth certificate, so that the reliability is vital. A typical example is obtaining one's passport, which will be used for further trusted operations. In particular, we look at the birth registration scheme at Wuhan, a city in China. Their registration scheme includes a resource management issue.

It has to be noted that the procedure introduced here is merely an example, and may be slightly different from the ones in any other cities. Figure B.1 shows the registration in Wuhan; the details are as follows:

1. When a woman has become pregnant, she reports it to an official in the company for whom she is working.

2. The official then produces a *request form for permission* from the government for the mother to give birth. The request form is sent to the local government department.

3. According to the government policy (e.g. one child per parents), the government will give or refuse the permission to give birth to the mother. The policy here would be based on the management of resources such as food. Birth without the permission may result in some fine or wage cut. The following discussion assumes the permit is issued. The permit is issued under the mother's name.

4. When the mother has given birth at a hospital, they pass the mother, or possibly the father, a *registration form verified by the hospital.*

5. Before the registration, the name of the baby has to be decided.

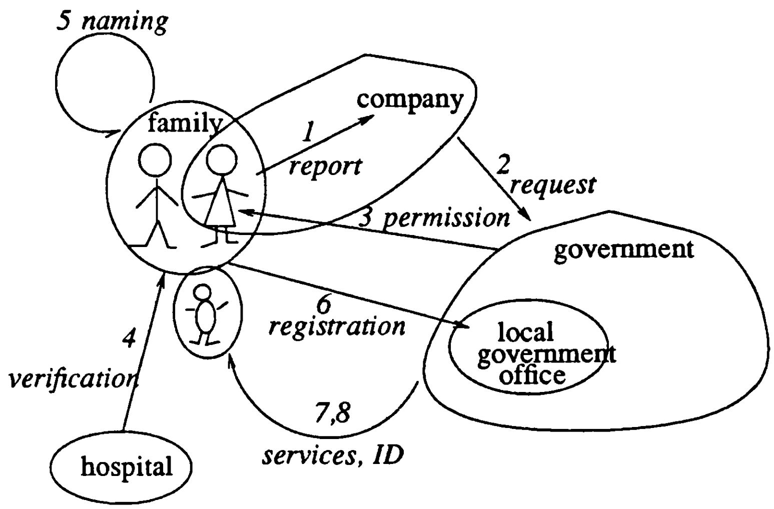


Figure B.1: Birth registration in Wuhan, China

6. A parent goes to a local police station where the registration takes place. The parent has to provide the followings:
   - the document giving permission to give birth under the mother's name
   - a *registration form verified by the hospital*
   - the name of the baby
   - the certificates of both parents which can be obtained by the appropriate police office where they most recently registered.
7. On a successful registration, the baby has an entry in the family registration booklet, and will become entitled to services such as food tickets and education opportunities.
8. Later at the age of somewhere between 16 and 18, the child is given an ID card with a unique number with the photograph. The ID number consists of the location code of the police station where the registration takes place, the date, and a serial number; an even number for girl, and an odd number for a boy. The card will be reissued at the age of 35.

Step 1 through step 3 are the Resource Admission Phase. Step 4 is the Configuration

Phase and step 5 is the Naming Phase. Step 6 could be considered as both the Certificate Registration Phase and the Configuration Confirmation Phase. The permit ticket works both as a permission from the administrator and as a registration ID.

Another interesting aspect is how information integrity is maintained. In our birth certificate example, well-formed transactions may be carried out in the government by observing the balance between issued permit tickets and registrations with and without the tickets. Separation of duty is performed by the fact that the birth is certified *absolutely* by the parents by declaring the baby as a family member, and *relatively* by the hospital who witnessed the birth; hence, they can verify the exact relation between the mother and the baby.

Accuracy check on an information item such as the name is trivial, because it is usually the parents who both assign the name and register it. In other words, the information originator is the same as the information provider.

Appendix C

The model of addressing in communication systems

In this appendix, we analyse the addressing procedures by modeling them using set theoretical tools [Hal60], and see where a possible flaw lies. The concept of addresses in our model could be extend to names as Hauser asserted [Hau86].

The appendix is organised as follows. The next section presents the concept of the group, on which our addressing model is based. Section C.2 discusses the overview of addressing activities. Section C.3 introduces the *set theoretical* tools, which will be used to build up the model in the subsequent sections. Section C.4 describes address space management. Section C.5 discusses binding and unbinding an address to and from a group. Section C.6 describes membership management, and Section C.7 discusses address maintenance.

C.1. A concept of group

In order to discuss address management, we need some consideration of the nature of communication systems. What we have enjoyed so far is basically the communication between two sites [Sha49]. However, we can look at networking differently as an association of more than one entity. We call this association a **group**.

A *network* could be perceived as a group of entities like hosts, routers, and gateways which understand the same protocol. A network group is somewhat stable and lasts until the technology of the next generation takes over. On the other hand, a *multicast group* is a group which is more dynamic in terms of membership than a *network*, and lasts as long as it has a member [Dee88]. An *(N)-connection* in ISO's Open System Interconnection Reference Model [ISO84] is also a group; it is very dynamic, in the sense that it lasts a relatively short time — though at least as long as the *association* of (N+1)-entities is required to exist.

In this respect, a host or gateway can be considered as a **group** of network entities. A network entity is regarded as a network operation process associated with a network interface. A host might have one entity, or could have more entities as a multi-homed host. If a multi-homed host forwards the network traffic which is not destined to itself, it is a router or gateway. A router forwards the traffic to and from the same type of links, while a gateway forwards the traffic to and from the different type of links.

Where the grouping is exercised according to organisational management domains, different protocol layer entities, which share the same operating system domain and whose interface to each other is through inter-process communication, are considered as a system group.

Each group has its own structure of members. The Internet IP group has a hierarchy of networks, subnetworks, and hosts and gateways. The addresses are inherited from the parent entities: that is, a network entity has the children of subnetworks; a subnet entity has the children of gateway and host entities. The attributes of a parent entity are inherited by its child entities. For instance, a network address is passed to a subnetwork, and then both are inherited by a host or a gateway. The Ethernet group has another hierarchy, as they have manufacturers on the top level. An individual system such as a minicomputer, work station, and a personal computer (PC) is a system group. A system group may have a tree structure, which has hardware components and software components of the system as children nodes. The system groups may be the children of a certain organisation. This organisation might have some parent organisation and so forth. The idea is that those system groups would well be the subtrees of a Directory of information [CCI88b]. In this hierarchy, each node of the Directory tree, such as an organisation, a system, and a component has a name as an identifier.

In our addressing model, we look at a network layer, particularly assuming the Internet IP environment [DoD83], [Pos81a]. An address is assigned to a network group object such as network, subnetwork, and host. Since multi-homed hosts, routers, and gateways are not really *addressable* on their own by one Internet IP address, we treat them differently from *addressable* group objects. They are considered as a set of group objects, hosts. A group object has members which are network entities. A network entity could be considered as a process of network operation associated with a network interface. For convenience, we use the term gateways including the meaning of routers in further discussion.

C.2. Overview of addressing activities

Addressing activities can be classified into the following aspects:

- Address space management
- Binding and unbinding an address to a group object
- Management of groups and their memberships
- Address maintenance

Address space management is concerned with what addresses are available to be assigned. It includes the following activities:

i. learning the range of address space from which one can assign an address for an object in an environment.

ii. maintaining the *free* addresses

iii. selecting an address from the *free* address space for a group object

Where an address is recycled, it is required to release the binding of an address from the associated entity and to return it back to the *free* address space.

In fact, the popular terms, *address assignment* and *unassignment*, have two implications. One is to select an appropriate address from the address space for a given group object; another is to bind the selected address to the group object in question. That is, the group object would be notified of the selected address by some means. On *unassignment*, the address has to be unbound from the object first, and it has to be returned to the *free* address space. In our discussion, the selection of address is considered as a part of address space management and binding is considered as another activity.

Management of groups and their memberships is the control over binding network entities together and producing an *addressable* group object. A network entity in our context is a protocol machine in a host, eg. an Internet IP process associated with an Ethernet interface in a system.

Address maintenance includes keeping track of current *active* bindings of addresses to group objects and making a decision to unbind them.

In the context of centralised dynamic address assignment, addressing has been analysed to have the following four properties [GoS86], [LKV85]:

- *Uniqueness:* No two stations have the same assigned address.
- *Assignment:* Unless the address space has no more *free* address, eventually an object will be assigned an address.
- *Consistency:* The manager's view of address assignment status is consistent with that of each object except for transient situations.
- *No frivolous unassignment:* The address should be unassigned only when it is unavoidable. As long as the object is active on the network its address should remain the same. Should reassignment be performed, it should end up with the same address.

We regard that any addressing system is more or less dynamic, in the sense that any network routing scheme can be considered as adaptive [ScS80]. We will discuss the addressing activities with the above properties. *Uniqueness* is discussed in address space management. *Assignment* is examined both in address space management and in binding and unbinding management. *Consistency* and *no frivolous unassignment* are discussed in address maintenance.

C.3. Constructing the model with the set theoretical tools

C.3.1. Overview

The most attractive point in applying set theoretical tools to addressing procedures would be that we have to define clearly each set of objects and their relations. This process helps us to realise what objects are of concern, and analyse the addressing activities systematically.

Using set theoretical tools to express addressing is not new. Gopal and Segall [GoS86] presented the first attempt to formalise the addressing activities by the concepts of sets. We have developed this further by incorporating the concept of a set of *group* objects.

In the following subsections, we present the basics tools of our model. The next subsection presents the required sets of objects. Subsection C.3.3 presents the functions between the sets. Subsection C.3.4 discusses the time indexes for the subsets.

C.3.2. The sets of objects

Suppose we have a set of entities, E. An entity, $e \in E$ (i.e. e belongs to E), could be an Internet Protocol machine associated with a network interface in a *host* or *gateway*. A *power set*, $\mathcal{P}(E)$, is a set which contains all the possible *groupings* of entities in E. If E has entities, e_0 and e_1 then $\mathcal{P}(E)$ includes the *subsets*, such as $\emptyset$, { e_0 }, { e_1 }, and { e_0 , e_1 }. ($\emptyset$ denotes the *empty set*. A name, *power*, comes from the fact that the number of possible subsets of a finite set with n elements is 2^n — i.e. if E has n elements, $\mathcal{P}(E)$ has 2^n elements.) $\mathcal{P}(E)$ also includes E itself as a subset which includes all the elements of E.

We define another set, G, which includes *addressable* objects. An *addressable* object is a group of entities in E. G holds *group* objects whose members are elements of $\mathcal{P}(E)$.

Now we need one more set, address space A. A holds all the possible addresses for a certain protocol operation. Let us assume it to be the Internet Protocol address space for the moment. As the addressing in the Internet is *network relative* addressing, we can *partition* A into network address spaces [DaP81]. There exists another *partition* for logical groupings, the multicast group address space, however, we discuss this part later. A network address space can be further *partitioned* into subnet address spaces. (The set theoretic word, *partition* means disjoint *subsets* in a set.)

An address in A has a correspondent value in G. If the value is a non-empty entity, g, in G (i.e. $g \in G$ and $g \neq \emptyset$), then the address has been assigned to the group entity, g.

A group entity, $g \in G$, has a correspondent value in $\mathcal{P}(E)$. If the value is a non-empty entity, x, in $\mathcal{P}(E)$ (i.e. $x \in \mathcal{P}(E) - \{\emptyset\}$), then the group has a member or members.

C.3.3. The functions and the subsets

Now we define functions between the sets, A, G, and $\mathcal{P}(E)$. The operations are described in terms of functions. Firstly we consider *passive* operations, which would not change any state of sets.

Group is a function which maps from A to G. It takes an address as an argument and returns an entity in G, g. If the address is not bound yet to any group entity, it returns $\emptyset$.

$$Group\colon A - \{\emptyset\} \to G$$
$$Group(a \in A) = \begin{cases} g & \text{if } a \text{ has been bound to } g; \\ \emptyset & \text{if } a \text{ has not been bound to any group object.} \end{cases}$$

Using $Group$ we can specify a *set* of addresses that have been bound to some group objects in G.

$$Assigned = \{a \in A\colon Group(a) \neq \emptyset\}$$

Another *passive* function is **Member** which maps from G to $\mathcal{P}(E)$. It takes a group entity as an argument and returns the corresponding entity in $\mathcal{P}(E)$ which is a subset of E. The idea is **Member** returns a set of members which belong to a given group. The formal notation is as follows:

$$Member\colon G \to \mathcal{P}(E)$$
$$Member(g \in G) = \begin{cases} x \in \mathcal{P}(E) & \text{if } g \text{ has any member;} \\ \emptyset & \text{if } g \text{ has no member yet.} \end{cases}$$

Using those *passive* functions, we can express networking and subnetworking as follows:

$$net_0 \in G$$

$$subnet_0 \in G$$

$$Member(subnet_0) \subseteq Member(net_0)$$

A host, $host_0$ would be expressed as follows:

$$Member(host_0) = \{e_0\}$$

If a group object has at least one member, we call it *configured*. $Configured$ is a subset of G.

$$Configured = \{g \in G\colon Member(g) \neq \emptyset\}$$

G has, in fact, many interesting subsets such as $Nets$, $Subnets$, $Hosts$, $Gateways$, and $Multicasts$. $Nets$ is a set of all the *network* type group objects in G. $Subnets$, $Hosts$, $Gateways$ are sets of *subnetworks*, *hosts*, and *gateways* type group objects, respectively. $Multicasts$ is a set of *logical* group objects. Member entities of a $Multicast$ type group may be spread over the different networks, whereas members of a $Subnet$ group object may be located within one network. All those $subsets$ belong to $\mathcal{P}(G)$.

Another *passive* function, **Address** $maps$ from a set of group objects to the address space, A. The formal notation is as follows:

$$Address\colon G - \{\emptyset\} \rightarrow A$$
$$Address(g \in G - \{\emptyset\}) = \begin{cases} a \in A & \text{if } g \text{ has any address;} \\ \emptyset & \text{if } g \text{ has no address yet.} \end{cases}$$

As a function can only have one value for one argument, we assume that no more than one address would be assigned to one group object. That is, there should be no *alias* address. Since we are only concerned with network level addressing at the moment, we believe that this restriction is reasonable in practice.

Using $Address$ we can specify a set of group objects that were claiming the addresses whether or not they were officially assigned:

$$Claiming = \{g \in G\colon Address(g) \neq \emptyset\}$$

Figure C.1 shows the simple conceptual view of the sets and functions defined so far.

C.3.4. Indexing with time

The group objects are not *static* in the sense that they change their memberships from time to time. For instance, an entity may be added to a network. In order to express those dynamics, we will index the subsets of G with an element of time space, $t \in T$. A network, $UCLnet$,

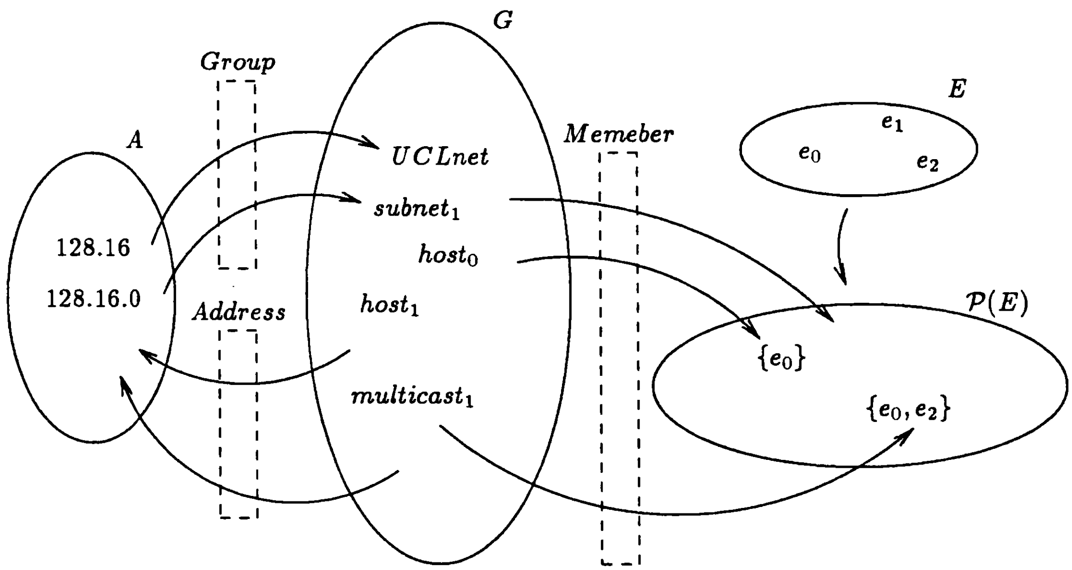


Figure C.1: the initial view of the model of addressing

at the time, t_i would be expressed as $UCLnet_{t_i}$. When the network has a new member, e_j at the time, t_{i+1}, the configuration change can be denoted as follows:

$$UCLnet_{t_{i+1}} = UCLnet_{t_i} \cup \{e_j\}$$

Alternatively if e_j is a *host*, the change can be expressed as follows:

$$host_{k,t_{i+1}} = host_{k.t_i} \cup \{e_j\}$$

$$UCLnet_{t_{i+1}} = UCLnet_{t_i} \cup host_{k,t_{i+1}}$$

Likewise, when the network has lost one of the members at the time, t_{i+1}, the configuration change can be denoted as follows:

$$UCLnet_{t_{i+1}} = UCLnet_{t_i} - \{e_j\}$$

or

$$UCLnet_{t_{i+1}} = UCLnet_{t_i} - host_{k,t_i}$$

If a network interface is removed from a *host*, it is expressed as follows:

$$host_{k,t_{i+1}} = host_{k,t_i} - \{e_j\}$$

Functions such as *Group*, *Address*, and *Members*, are, in fact, sets themselves of ordered pairs, as follows:

$$Group = \{(a, g): a \in A \text{ and } g \in G\} \text{ and } Group \subseteq A \times G$$

$$Address = \{(g, a): g \in G \text{ and } a \in A\} \text{ and } Address \subseteq G \times A$$

$$Member = \{(g, X): g \in G \text{ and } X \in \mathcal{P}(E)\} \text{ and } Member \subseteq G \times \mathcal{P}(E)$$

Thus, we can index them with an element of time as well. For instance, a group object associated with an address $a \in A$ at the time t_i is $Group_{t_i}(a)$. Member of a group, $g \in G$, at time t_i is $Member_{t_i}(g)$.

Finally, we define a subset of G, *Requesting*, which includes group objects which have requested for addresses but have not been assigned yet. As long as $Requesting \neq \emptyset$, addressing activities are required.

C.4. Address space management

C.4.1. Overview

In this section, we look at the address space in more detailed manner, and analyse how its management is carried out on request for a new address. Note that we are looking at the addressing on a network layer or on a link layer in the first place. Releasing an unused address is another important issue of the management, and will be discussed later in Section C.4.8 Address maintenance.

The next subsection discusses addressing schemes. Subsection C.4.3 describes the address allocation operation.

C.4.2. Addressing Schemes

A network layer address is composed of objects called numbers. The address space, A, includes those numbers. Addressing is to pick up numbers, concatenate them, and produce an address. An address itself is a number and an *element* of A.

There have been two addressing schemes; one is *relative* addressing and another is *absolute* addressing [DaP81] or *physical* addressing [Com88].

Relative addressing implies that an address for a network entity is decided in a hierarchical way. If it is hierarchical according to the network structure, an address is composed of objects, such as a network number and a local number. A local number is only unique in that network domain. A local number would be further divided into a subnetwork number and a host number. A host number is again only unique in the subnetwork domain. This type of addressing is *network-relative*. The Internet IP addressing [RSR88] is an example of *network-relative* addressing.

Absolute addressing would be that an address is a unique number in an address space. The Ethernet addressing could be an example of this scheme, however, it actually is produced in a hierarchical manner — relative not to a network but to a vendor [DIX82].

The most significant characteristic of *absolute* addressing is that wherever a network interface is moved to, the address remains same. On the other hand, a *network-relative* address indicates a kind of location of an entity; the address has to be changed when the entity moves to a different location.

Network-relative addressing has a property that each component of an address, such as network address, subnetwork addresses, and host addresses, is mapped to a specific type of group object, such as network, subnetwork, and host. A formal notation for this is as follows:

$$Group(A.nets) \subseteq Nets \cup \{\emptyset\}$$

$$Group(A.subnets) \subseteq Subnets \cup \{\emptyset\}$$

$$Group(A.hosts) \subseteq Hosts \cup Gateways \cup \{\emptyset\}$$

Similarly, if any network object, $net \in Nets$, has an address, the address is one of the network addresses, $a \in A.nets$. It is true for $Subnets$ and $Hosts$ as well.

$$Address(Nets) \subseteq A.nets \cup \{\emptyset\}$$

$$Address(Subnets) \subseteq A.subnets \cup \{\emptyset\}$$

$$Address(Hosts \cup Gateways) \subseteq A.hosts \cup \{\emptyset\}$$

Note that as far as addressing is concerned, there is no difference between $gateways$ and $hosts$. They are equally treated as $hosts$, as each network interface of a $gateway$ is addressed as a $host$.

According to a classical conception [Sho78], *an address indicates where a resource is and a route tells us how to get there.* An address could be independent from a routing scheme as in *absolute* addressing. This is the case in a small-size network and in a broadcast network. The size of a network may grow by interconnecting some subnetworks via store-and-forward nodes such as routers, and gateways. As the size of a network reflects directly on the size of a routing table [Sun77], the routing is structured naturally in a hierarchical manner [KlK77]; for a large network the routing table at each node may be based on clusters at a certain level such as network, and not on the whole nodes.

To ease this scheme, hierarchical addressing, namely *relative* addressing may be introduced. In this case, addressing is no longer independent of routing. It is a routing policy that decides an addressing.

If the routing hierarchy is based on the network structure, the addressing would be *network-relative*; that has been traditionally the hierarchical addressing. On the other hand, if the routing hierarchy is based on a region where a united access control policy could be exercised [Cla89], an address could be *region-relative*. Indeed, an addressable entity, eg. a region or a network, is a unit of access control on the network level. The *addressability* provides basic *reachability*. *Accessibility* is the result of the access control over the reachable entities. Therefore, the *practical reachability* is derived by accessibility.

In the further discussion, we only look at the *network-relative* addressing as an example. *Absolute* addressing could be seen as identical to intra-domain addressing of *network-relative* addressing.

C.4.3. Address allocation

The allocation of an address includes the following operations:

i. learning the range of address space from which one can assign an address for an object in an environment

ii. selecting an address from the *free* address space for a group object.

It takes the following steps, for example, in the Internet IP addressing scheme:

1. *Network Address Allocation:*

    i. learn a Network Class of a Network object, $g \in Requesting$ at the current time, t_i, which needs an address

    ii. learn the Network Number range for a given Network Class (The Internet IP uses an 32-bit address for $hosts$. As it is *network-relative*, a network part is the first 8 or 16 or 24 bits for class A or B or C network address [RSR88])

    iii. choose a Network Number from the part of the range which has not been assigned to anything yet

    iv. produce a *Network Address* by concatenating the Network Class and the Network Number.

2. *Subnetwork Address Allocation:*

    i. learn the Subnetwork Number range for a given *Network Address*

    ii. choose a Subnetwork Number from the part of the range which has not been assigned to anything yet

    iii. produce a *Subnetwork Address* by concatenating the *Network address* and the Subnetwork Number.

3. *Host Address Allocation:*

    i. learn the Host Number range for a given *Subnetwork Address*

    ii. choose a Host Number from the part of the range which has not been assigned to anything yet

    iii. produce a *Host Address* by concatenating the *Subnetwork address* and the Host Number.

We formalise the host address allocation in the following. We define new $functions$ to specify various ranges;

$$NumberSpace: A \rightarrow \mathcal{P}(A)$$

$$NumberSpace(x \in A) = Y \subset A$$

where $Member(g \in \text{the range of } Group(Y)) \subseteq Member \circ Group(x)$

$$HostNumberSpace = NumberSpace \mid A.subnets$$

$NumberSpace \mid A.subnets$ means that the *domain* of $function$ is restricted to the address space for subnets in A, $A.subnets$.

Host address allocation, for instance, can be formalised as follows:

i. $\exists s \in A.subnets$ for a given group object, $g \in Requesting_{t_i}$

ii. the Host Number range is $HostNumberSpace(s) \subset A$

iii. choose $n \in Concatenate(s, HostNumberSpace(s)) - Assigned_{t_i}$ where $Concatenate$ is a $function$, $Concatenate$: $A \times A \rightarrow A$ to produce an address from two numbers.

iv. a host address, $a \in A$ is obtained by $a = Concatenate(s, n)$.

A new address can be taken, if and only if a group object has not had an address yet. A new number is taken in (ii) above, if and only if there is a free number in $HostNumberSpace(s)$ to be assigned to this type of group object. This condition guarantees the address to be *unique* in the address space. The formal notation for this is as follows:

$\exists a \in A$ for a given $g \in Hosts \cap Requesting_{t_i}$

if and only if

$a \notin Assigned_{t_i}$;

and

$$Concatenate(s, HostNumberSpace(s)) - Assigned_{t_i} \neq \{\emptyset\}$$

C.5. Binding and unbinding an address to a group object

A selected address, $\exists a \in A$, is bound to a group object, $g \in Requesting_{t_i}$, which has requested for an address. Binding operation is formally described as follows:

$$Group_{t_{i+1}} = \left(Group_{t_i} - \{(a, \emptyset)\}\right) \cup \{(a, g)\}$$

At the time, t_{i+1}, $Group(a)$ maps to g. It implies $Assigned_{t_{i+1}} = Assigned_{t_i} \cup \{a\}$.

$$Address_{t_{i+1}} = \left(Address_{t_i} - \{(g, \emptyset)\}\right) \cup \{(g, a)\}$$

Unbinding operation is described in a similar way:

$$Group_{t_{i+1}} = \left(Group_{t_i} - \{(a, g)\}\right) \cup \{(a, \emptyset)\}$$

At the time, t_{i+1}, $Group(a)$ maps to no group object, $\emptyset$. It implies $Assigned_{t_{i+1}} = Assigned_{t_i}$ $- \{a \in A.nets\}$.

$$Address_{t_{i+1}} = \left(Address_{t_i} - \{(g, a)\}\right) \cup \{(g, \emptyset)\}$$

However, unbinding is somewhat more complicated because there must be a decision making on what condition an address should be unbound. This aspect is further discussed later in the address maintenance section.

A group object, $g \in Requesting_{t_i}$ has to be guaranteed to be assigned an address at the time, t_k, in the future eventually. That is, for $\forall g \in Requesting_{t_i}$, $\exists t_k$, where $i < k$, such that $g \notin Requesting_{t_k}$ and $Group_{t_k}(a) = g$.

C.6. Management of groups and their memberships

Joining a group in operation is a commitment of belonging to a certain level of access control criteria [Cla89]. A router between groups exercises access control. A group may be a host, a subnetwork, a network, or a multicast group. The decision making of grouping, therefore, is a policy enforcement.

When an entity, $e_j \in E$, joins a group, $g \in G$, it has a membership of the group. This can be described formally as follows:

$$Member_{t_{i+1}}(g) = Member_{t_i}(g) \cup \{e_j\}$$

This implies $Configured_{t_{i+1}} = Configured_{t_i} \cup \{g\}$ if g had no member at the time, t_i.

If a network interface, e_j, joins a network, net_2 in operation, it commits to belong to a host, $host_0$, a subnetwork, $subnet_1$, and a network, net_2, as follows:

$$\text{for all } g,\ \forall g \in \{host_0, subnet_1, net_2\},$$

$$Member_{t_{i+1}}(g) = Member_{t_i}(g) \cup \{e_j\}$$

$$\text{where for all } t,\ \forall t \in \{t_i, t_{i+1}\},$$

$$Member_t(host_0) \subseteq Member_t(subnet_1) \subseteq Member_t(net_2)$$

Similarly when an entity, e_j, leaves a group, g, it will loose a membership of g.

$$Member_{t_{i+1}}(g) = Member_{t_i}(g) - \{e_j\}$$

If the group, g, has no member as a result, it implies $Configured_{t_{i+1}} = Configured_{t_i} - \{g\}$.

A group object, $g \in G$, which has an assigned address but has not been *configured* yet, is expected to have members some time in the future. If it took too long to do so, it might be possible that the address would be unassigned.

A group object, which has been configured but has no address assigned, will join $Requesting$ at some stage, t_i.

C.7. Address maintenance

Address maintenance is concerned with bindings between the address space and group objects. Main activities include making decision of *unassignment* of an address and avoiding *inconsistency* of bindings. To do so, the following operations are required:

i. detecting the unused addresses and returning them to the *free* address pool

ii. control over the addresses in use

An address could be placed into the *free* address pool where addresses were recycled. The address should have been unbound from any group object and no group object is claiming the address any more. The following formal notation is trivial from the definition of $Assigned$:

$$Assigned_{t_{i+1}} = Assigned_{t_i} - \{a \in A\}$$

if and only if

$$Group_{t_{i+1}}(a) = \{\emptyset\};$$

and

$$a \neq Address(\forall g \in Claiming_{t_{i+1}})$$

Where unassignment of an address is required, it should not be *frivolous*. That is, unassignment is only performed when the life time of a group object is observed to be over. Therefore, it needs to keep track of life time of each group object. In a dynamic address scheme [GoS86] [LKV85], group objects are monitored by polling and when it is perceived as *dead*, its address is unassigned. Unassignment is also performed on request from a group object. In a preassigned addressing scheme, on the other hand, unassignment would be required at a reconfiguration time. Reconfiguration is a change of binding between G and $\mathcal{P}(E)$ in terms of our formal model. A configuration change can be detected by the following fact:

$$Member_{t_{i+1}}(g) \neq Member_{t_i}(g)$$

where $Member_{t_{i+1}}(g) - Member_{t_i}(g)$ denotes a set of newly added entities to a group object, g, and $Member_{t_i}(g) - Member_{t_{i+1}}(g)$ shows a set of entities which disappear between times, t_i and t_{i+1}. The latter is further formalised to express the detection of *removed* entities as follows:

$$Member_{t_i}(g) - \left(Member_{t_{i+1}}(g) \cup Member_{t_{i+2}}(g) \cup \ldots \cup Member_{t_{i+k}}(g)\right)$$

where the duration of time $t_{i+k} - t_i$ is long enough to confirm the removal of the entities. On the detection of the removal, the addresses of the entities are unbound firstly, and then the addresses are returned to the free address space.

Another aspect of address maintenance is to avoid *inconsistency*. Here *consistency* means that a group object is claiming the exact address which has been assigned to it. *Inconsistency* occurs where a group object is claiming a different address from the assigned one. From this perspective, control over the addresses in use is required. The following operation reveals the status of *consistency* from the manager's viewpoint:

$$Address\left(Group(a \in Assigned_{t_i})\right) = \begin{cases} a & \text{if consistent;} \\ \emptyset & \text{if binding has not completed yet;} \\ else & \text{if inconsistent.} \end{cases}$$

From the network's viewpoint, the following operation verifies whether the claimed address is properly assigned or not:

$$Group_{t_i}\left(Address_{t_i}(g)\right) = \begin{cases} g & \text{if consistent;} \\ \emptyset & \text{if } g \text{ claims } a \text{ without proper assignment;} \\ else & \text{if } g \text{ claims another group's address.} \end{cases}$$

Appendix D

The detailed operations for information maintenance

In this appendix, we describe the detailed operations for information maintenance explained in Chapter 5.

The appendix is organised as follows. The next section describes the operations for the detection of changes. Section E.2 presents the operations for update of an address.

D.1. Detection of changes

As a result of the above discussion, the following tools are required:

1. A reporting mechanism from a host to the manager
2. A monitoring mechanism
3. A verification mechanism from the manager to a host
4. A mechanism to synchronise the recognition of host changes and the update of the associated information

These operations can be spontaneous or on request. If the operation is on request, a report message has to have a report ID associated with the request. Using the manager processes which we used in a registration protocol at Configuration Confirmation, the detailed above operations could be expressed in the following.

(1) Report from the host site, A, to the manager, M, of a status change

A report from the host site may be issued spontaneously or on request. The report is sent to the manager as follows:

A $\rightarrow$ M: [$Authenticator_A$, $SecurityLabel_A$,{ current $Certificates$ for A, status, T_A }SK_A]

Status could be *Removed*, or *Down*. *Type of entity* could be a particular network protocol entity, or it could be *all* in case of the removal of the whole system at the site. In case of *Down* message, it can also contain the information on a scheduled up time.

Depending on the security label of the system, the reported information might be further verified at the *semantic level* by the manager as explained later in (4) or some other reliable mechanism.

For removed, the manager and the controller examine it, and the manager reports to CA and the information manager. The information manager asks CA for the notarisation of the *Expired* message. It is the information manager which is responsible to propagate it to the relays.

(2-a) Strong monitoring

Strong monitoring is a probe mechanism. The manager, M, sends a probe packet to an object site, A, and the site replies as follows:

M $\rightarrow$ A: [$Authenticator_M, SecurityLabel_M, \{T_M\}SK_M$]

A $\rightarrow$ M: [$Authenticator_A, SecurityLabel_A, \{T_M + 1, T_A\}SK_A$]

$SecurityLabel_M$ ensures that the manager is authorised to monitor the object site, A. The time stamp of M uses for associating the reply with the request. The time stamp of A in the reply would be used for calculating the delay which could be used to deduce the processing delay at A and the delay over the network.

(2-b) Passive monitoring (an observer)

The manager observes the network by picking up sender's address of a particular protocol packet and keeps track of active addresses. It is essential that the packet has $Authenticator$ and $Certificate$ of the sender. Ideally observed packets need to be sent out from all the active objects frequently, so that inactive addresses are detected timely and as correctly as possible. To this respect, a regular reporting mechanism from objects would be useful.

(3) Verification from the manager, M, to the host site, A

The manager keeps an eye on network objects either by sending some kind of probe or by watching *active* addresses. If it finds anything unusual, it will verify the information. For instance, it may find an entity not responding to a probe or it may find inconsistency in the binding of an address to an object.

Verification is exercised after the examination of the reporter's security label as in (1), or after monitoring as in (2-a) or (2-b) above. The verification methods could vary according to systems' types. An example would be to ask the object site for some information items, which could be done by the manager as well. Another method might be to tell the object site to execute some loop-back test at a network interface, and send back the result to the controller as follows:

M $\rightarrow$ A: [{ Operation=loop-back test, network interface ID, T_M }SK_M]

A reply would be as follows:

A $\rightarrow$ M: [{ the result of the test, $T_M + 1$, T_A }SK_A]

(4) Reporting from the manager, M, to the relay

This will be explained in the next section.

D.2. Update of an address

We need the following mechanisms:

1. Confirmation of removal by CA
2. A distribution mechanism of an Expired message to relays which hold host information

The protocols for the above are as follows:

(1) Confirmation of removal by CA

The manager, M, asks the confirmation of removal of an object to CA as follows:

M $\rightarrow$ CA:[{ object site ID, Removal, $T_M\}SK_M$]

where object site ID is $Authenticator$ or $Certificate$ of the object site. As CA is also reported from the detector of the change, CA does may not require the confirmation operation. However, if the request of notarisation comes and CA has not received the change report from the detector, it asks the detector to confirm the change. On a successful confirmation, the certificate is issued and sent to the manager.

CA $\rightarrow$ M:[{ object site ID, $Expired, t_{CA}\}SK_{CA}$]

(2) A distribution mechanism from the information manager to a relay, R

The information manager distributes the update as follows:

M $\rightarrow$ R: [{ *Update operation*, A's Directory name,

A's security label, $Authenticator$, Information item ID, new value $\}SK_M$]

Update operations include modify and delete. If it is delete, no new value is included, but the *Expired* message from CA is attached.

Appendix E

Formal analysis of the address flow

In this appendix, we examine our scheme of address flow introduced in Chapter 5 by the formal method introduced by Burrows, Abadi, and Needham [BAN89]. Their attempt is novel because the protocol flow is examined in terms of the knowledge obtained by each participant of the transaction; this simplifies the proof of authentication protocols. Our use of the method is to examine the integrity of information flow in the protocol. The method is suitable for our needs because it simplifies the flow of information. Traditional ways to express those authentication protocols do not provide the proof mechanism for the correctness of a protocol; the use of other methods such as a state transition diagram would make the proof procedure unnecessarily complicated for our purpose.

Although the recent argument by Nessett [Nes90] showed that the logic missed out the confidentiality aspect intentionally for simplifying the process [BAN90], it is no problem with our application of the logic because the primary concern of our scheme is to maintain information integrity.

In the next subsection, we introduce the notation. Section F.2 describes the goal of our analysis, Section F.3 describes rules, and Section F.4 gives the assumptions. Sections F.5 through F.11 analyse each step of our scheme; viz. Resource Admission, Naming, Certificate Registration, Configuration, Configuration Confirmation, and information look-up. Section F.12 gives the conclusion of this formal analysis.

E.1. Introduction to notation

$A \models X$: means that A believes that X is a true information item.

$A \Rrightarrow X$: means that A has a jurisdiction over X.

$\sharp(X)$: originally means that the information item, X, is generated recently. We modify this definition into that the production of X has been validated and verified recently; i.e.X is perceived as correct at a recent time but it is not clear whether its generation was recent or not.

$A \triangleleft X$: A sees X.

$A \mid\sim X$: A once said X.

We introduced a new notation as follows, which indicates the information item has been notarised by a trustful authority:

$\langle\langle X \rangle\rangle_C$: X is certified by the authority C.

In the following subsections, we also use the following abbreviations:

- NA : a Naming Authority
- m : the Administrator and the Object Manager
- CA : the Certification Authority which has an off-line agent and an on-line agent (i.e. on-line means that the agent is on the network).
- M : the management system over the network
- R : the information system
- $F.in$: the incoming filter
- $F.out$: the outgoing filter
- B : the user of information, which is another object
- g : an object
- A : the system, to which the object is configured
- Reg : the registration ID

- X_a : the name of the object
- X_d : the description of the object
- $X_c(Z)$: the cryptographic system attributes of Z
- T_Z : the time stamp of Z

E.2. The goal of analysis

As the notation which we use was originally intended for authentication, the successful result of a protocol between two principals, A and B, is that both principals know the shared secret, and each knows that the other knows it as well. The notation for this result is as follows:

$$A \models X \text{ and } A \models B \models X$$

and

$$B \models X \text{ and } B \models A \models X$$

However, our goal is somewhat different, in that we need to ensure an information item on an object, g, e.g. a network address, X_a, originated by the *information originator*, NA, and verified by CA, is assigned to the *information provider*, A, as is intended, and arrives at the *information user*, B, eventually. Therefore, the successful result should be as follows:

$$(1)\ NA \models X \text{ and } (2)\ NA \models A \mid\sim X$$

$$(3)\ CA \models NA \models X \text{ and } (4)\ CA \models (A \mid\sim X)$$

$$(5)\ A \models NA \models X \text{ and } (6)\ A \models X$$

$$(7)\ B \models CA \models X \text{ and } (8)\ B \models NA \models X$$

$$(9)\ B \models X$$

We rewrite the above goals with more precise information items using the following notations:

- *Bind(X_a)*= g means that the address, X_a is assigned for the object, g
- *Claim(g)=X_a* means that g is claiming X_a

Then the goals are expressed as follows:

$$(1)\ NA \models Bind(X_a){=}g,\ (2)\ NA \models Claim(g){=}X_a$$

$$(3)\ CA \models NA \models Claim(Bind(X_a)){=}X_a,\ (4)\ CA \models Claim(Bind(X_a)){=}X_a$$

$$(5)\ A \models NA \models Claim(Bind(X_a))=X_a,\ (6)\ A \models Claim(Bind(X_a))=X_a$$

$$(7)\ B \models CA \models Claim(Bind(X_a))=X_a,\ (8)\ B \models NA \models Claim(Bind(X_a))=X_a$$

$$(9)\ B \models Claim(Bind(X_a))=X_a$$

E.3. The rules

The following rules are defined.

The first rule is that if A sees the information item, X, encrypted with B's secret key, A believes that B once said X as follows:

$$\frac{A \models PK_B, A \triangleleft \{X\}SK_B}{A \models B \mid\!\sim X}$$

The second rule is that if A believes X is uttered only recently and B once said X, then A believes that B has said X, recently as follows:

$$\frac{A \models \sharp(X), A \models B \mid\!\sim X}{A \models B \models X}$$

The third rule is that if A believes that B has jurisdiction over X then A trusts B on the truth of X as follows:

$$\frac{A \models B \mapsto X, A \models B \mid\!\sim X}{A \models X}$$

E.4. The assumptions

We have the following assumptions:

Assumption 1:

$$CA \models \sharp\ (Reg)$$

Assumption 2:

$$CA \models NA \mapsto Bind(X_a)= g$$

Assumption 3:

The timers in all the processes of concern are synchronised.

For $x \in \{\ M, R, CA, A\ \}$,

$$x \models (\sharp\ (T_M), \sharp\ (T_R), \sharp\ (T_{CA}), \sharp\ (T_A))$$

Assumption 4:

$$\{ A, M, R \} \mid\equiv CA \mid\Rightarrow (Reg, X_a, X_d, X_c)$$

Assumption 5:

$$\{ A, M, R \} \mid\equiv CA \mid\Rightarrow \{ Claim(Bind(X_a))=X_e \}$$

That is, if

$$\{ A, C, D, M, R \} \mid\equiv CA \mid\sim x, \text{ where } x \in \{ Claim(Bind(X_a))=X_e \},$$

$$\text{then } \{ A, M, R \} \mid\equiv x.$$

Assumption 6:

$$CA \mid\equiv \{ A, M, R \} \mid\equiv CA \mid\Rightarrow \{ (X_e) \}$$

Assumption 7:

$$\{ A, M, R \} \mid\equiv NA \mid\equiv Claim(Bind(X_a))=X_e,$$

$$\text{if and only if } \{ A, C, D, M, R \} \mid\equiv CA \mid\equiv Claim(Bind(X_a))=X_a.$$

Assumption 8:

$$CA \mid\equiv M \mid\Rightarrow \text{(change mode, object ID, information items)}$$

i.e. M has an authority to declare the change and it indicates that the verification has done.

We also assume that relevant public keys are distributed previously to the manager and the information system.

In the following subsection, we analyse each phase formally.

E.5. Resource Admission

The following is known in this phase.

$$m \mid\Rightarrow g$$

$$m \mid\Rightarrow \sharp(g)$$

means that an Object Manager or an Organisational Administrator, m, is widely known as having jurisdiction over the existence of the object, g.

E.6. Naming

The following is known in this phase.

$$NA \mapsto X_a$$

A naming authority, NA, has jurisdiction over the name for g, X_a.

The protocol is as follows:

Message 1 $m \rightarrow NA$: $[g]$

Message 2 $NA \rightarrow m$: $[Group(X_a) = g]$

The manager, m, applies to NA for an address for object g with Message 1. NA then allocates an address for g. However, from our original assumption of the scheme, it is not clear that NA would have a verification mechanism for its own operations. We therefore assume that no one is sure of $\sharp(X_a)$.

NA provides m with an address for g with Message 2 over the *generation channel*. All m knows is that X_a is allocated by NA for an object, g as follows:

$$m \models NA \mapsto (Group(X_a) = g)$$

and

$$m \models NA \mid\sim (Group(X_a) = g)$$

E.7. Certificate Registration

The protocol is as follows:

For a system object,

Message 1 $m \rightarrow CA : [\langle\langle P\rangle\rangle_{CP}, \langle\langle V\rangle\rangle_{CV}, X_a, \langle\langle L\rangle\rangle_{CL}, X_D, \langle\langle m\rangle\rangle_{CM}, X_C(PK_A), T_m]$

The parameters are:

- $\langle\langle P\rangle\rangle_{CP}$: the permit from the *Organisational Administrator*
- $\langle\langle V\rangle\rangle_{CV}$: the verification ticket which verifies the existence of object
- X_a : the name obtained from the Naming Authority
- $\langle\langle L\rangle\rangle_{CL}$: the previously notarised location ID (e.g. an Organisation ID)

- X_d : the description of the object
- $\langle\langle m \rangle\rangle_{CM}$: the certificate of the Manager, m
- $X_c(A)$: the cryptographic system attributes including its public key for the system A, to which the object, g belongs.
- T_m : the time stamp of m

It all depends on how much CA believes those information items certified by the different authorities. On a successful registration, a registration ID is given as follows:

Message 2 $CA \rightarrow m : [Reg.ID, t_{CA}]$

or Certification Authority's public key may be given to the object at this time as well as follows:

Message 2 $CA \rightarrow m : [Reg.ID, X_{CA}, t_{CA}]$

where X_{CA} is the cryptographic system attributes for the Certification Authority on the network.

If the object is a *system* object , the Certification Authority decides the Security Class for it.

Message 3 $CA \rightarrow: [X_S]$

where X_S denotes a security class. We show this as a message but it is not passed to anyone. It shows that X_S is created.

The protocol analysis is as follows:

$$CA \models \sharp T_m$$

$$CA \models \langle\langle X \rangle\rangle \text{ if } CA \models \sharp X$$

$$CA \models CP \mapsto P$$

$$CA \models CV \mapsto V$$

$$CA \models CL \mapsto L$$

$$CA \models CM \mapsto m$$

We assume that each certified X, $\langle\langle X \rangle\rangle$, has been issued by an appropriate trustful authority with a trustful time stamp. Therefore, the Certification Authority always believes the certified information as follows:

$$CA \triangleleft \langle\!\langle X \rangle\!\rangle_{CX}$$

then

From Message 1,

$$CA \models (\langle\!\langle P \rangle\!\rangle_{CP}, \langle\!\langle V \rangle\!\rangle_{CV}, \langle\!\langle L \rangle\!\rangle_{CL}, \langle\!\langle m \rangle\!\rangle_{CM})$$

If the time stamps in those certificates, $\langle\!\langle P \rangle\!\rangle_{CP}$, $\langle\!\langle V \rangle\!\rangle_{CV}$, $\langle\!\langle L \rangle\!\rangle_{CL}$, and $\langle\!\langle m \rangle\!\rangle_{CM}$, are considered recent compared to T_m, then

$$CA \models (P, V, L, m)$$

and then CA believes that the object described as X_D with its cryptographic attributes X_C is possibly named X_a, is located at L, and managed by m.

$$CA \models (g, X_D, X_C)$$

$$CA \triangleleft (g, X_a)$$

We cannot say $CA \models X_a$ unless NA has its strong verification mechanism and guarantees $\sharp(X_a)$.

From Message 2, the *Object Manager*, m, sees *Reg.ID*:

$$m \triangleleft Reg.ID$$

where we assume that

$$m \models CA \mapsto Reg.ID$$

$$m \models CA \mapsto \sharp(Reg.ID)$$

E.8. Configuration

Message 1 $m \rightarrow A : [Reg.ID, X_a, X_c(A), X_d, X_m]$

That is, the Manager, m, sets the parameters in the object, g; the parameters are the Registration ID, g's name, X_a, and its cryptographic system and other attributes, $X_c(A)$ and X_d. X_m is the information on managers, such as the public keys of the management system, CA, and the relevant information systems. The information includes their names, locations, and cryptographic system attributes.

From Message 1,

$$A \triangleleft Reg.ID$$

$$A \triangleleft X_a$$

$$A \triangleleft X_c(A)$$

$$A \triangleleft X_d$$

$$A \triangleleft X_m$$

The important point is that a network object, g, sees those parameters for network operations, however, it has no idea that the parameters are true. A strong belief between m and A such that:

$$A \models X, \text{ if } A \triangleleft m \models X$$

would be only possible in a very secure and trustful environment.

The current problem in network operations is that the objects claim the information item which they only learn from its managers without verification.

E.9. Configuration Confirmation

In the following the protocol sequence is summarised, and analyse formally. We will use a formal notation following that used in a Kerberos paper [SNS88] with the key notations introduced by Needham and Schroeder [NeS78]. We denote *Reg* as a registration ID. The protocol packet exchange is as follows.

Message 1:

A informs M of the completion of its configuration with the registration ID.

A → M: [g's ID, { Reg } PK_M]

The above says that A sends a packet [...] to M. {...} PK_M indicates that the field {...} is encrypted by the public key of M. Encryption with the receiver's public key provides the means of confidentiality, whereas encryption with sender's secret key would provide the means of authentication [NeS78].

Message 2.1:

M asks CA for the expected information to be registered with the registration ID.

$M \rightarrow CA$: [g's ID, {{ Reg, T_M } SK_M } PK_{CA}]

Message 2.2:

CA will reply with the attributes of the object.

$CA \rightarrow M$: [g's ID, {{ Reg, X_s, X_d, $X_c(A)$, $T_M + 1$ } SK_{CA} } PK_M]

M verifies the configuration of g over the network. We do not specify the verification method, however, the use of g itself is mandatory; i.e. the verification transaction uses g to see its operability and usability. One way of carrying out the verification would be to ask A for the attributes values, and A answers as follows:

Message 3.1:

$M \rightarrow A$: [Enquiry, { g, IDX_s, IDX_d, T_M } SK_M]

Message 3.2:

$A \rightarrow M$: [Reply, {{ XX_s, XX_d, $T_M + 1$ } SK_A]

If M perceives that $PK_A \in X_c(A)$, $XX_s = X_s$, and $XX_d = X_d$, M sends the following packets, otherwise the procedure stops.

Message 4:

M reports the confirmation of the addition of the new host g to CA.

$M \rightarrow$ { CA } : [{ Add, g, X_s, X_d, $X_c(A)$, T_M } SK_M]

M requests CA for notarisation.

Message 5.1:

$M \rightarrow CA$: [{ Request, g, X_s, T_M } SK_M]

Message 5.2:

$CA \rightarrow M$: [{ Reply, g, Certificates, $\langle\langle X_s \rangle\rangle_{CA}$, $T_M + 1$ }SK_{CA}]

A certificate for the mapping between internet and subnet addresses could be as follows:

{ { subnet address, protocol address } SK_G, version } SK_{CA}

where SK_G is a secret key of subnet G. A key could be a private key, but we regard it as a pair of public and secret keys, which we will use as a capability for look-up.

Certificates include the security label and an authenticator.

The security label is as follows:

$$\{\text{ A's Directory name, security class, } version_{SecurityLabel}\}SK_{CA}$$

For an authenticator, we could employ an idea suggested by Wilbur [WCM89] that an authenticator includes system A's public key. The idea is subtle; having A's authenticator one does not need to look-up A's public key from any other place such as the Directory or any other information base. The format of an authenticator is as follows:

$$\{PK_A, \{subnetaddress_1\}SK_A\ ,\ version_{Authenticator}\}SK_{CA}$$

Versions can be time stamps, nonces, or any other appropriate uniquely assigned numbers. If time stamps are used, there must be a global clock in the network; synchronisation is not required but there is the need of maintenance of the clock.

Message 6:

M distributes the certificate and other information item associated with g to a relevant information system, R.

$M \rightarrow R$: [{ Add, g,$\langle\langle\ X_a\ \rangle\rangle_{CA}$, T_M } SK_M]

Since the information has been previously certified, it does not matter who the *information provider* is, hence, authentication of the sender is not required. However, for the sake of the authentication of the update operation ID, we encrypt the packet with the manager's secret key.

The added object information is used by the information system to update its access control list, because the object will be a new user of the information system for information look-up.

Message 7:

M assigns certificates to g. The relevant group keys are assigned as well.

$M \rightarrow A$: [{ Set, g,$\langle\langle\ X_a\ \rangle\rangle_{CA}$, $X_c(group)$, T_M } SK_M]

Protocol analysis is as follows.

From Message 1,

$$D \triangleleft Reg$$

From Message 2.1,

$$CA \mid\equiv M \mid\sim Reg$$

From Message 2.2,

$$M \mid\equiv CA \mid\sim (Reg,\ X_a,\ X_d,\ X_c(A)),$$

and

$$M \mid\equiv CA \mid\Rightarrow (Reg,\ X_a,\ X_d,\ X_c(A))$$

hence, by Rule 3,

$$M \mid\equiv (Reg,\ X_a,\ X_d,\ X_c(A))$$

After the receipt of Message 3.2 which is a reply to the request in Message 3.1,

$$\mathrm{M} \mid\equiv \mathrm{A} \mid\sim (\mathrm{Claim(g)}=\mathrm{XX_a},\mathrm{XX_d})$$

Then M checks locally whether $XX_a=X_a$ and $XX_d=X_d$ (and the success of decryption of the packet shows $PK_A \in X_c(A)$) or not. If so, the configuring X_a with the object g is verified as follows:

$$M \mid\equiv A \mid\sim (Claim(g)=X_a)$$

From Message 4,

$$CA \mid\equiv M \mid\equiv (A \mid\sim (Claim(g)=X_a)\ )$$

Also,

$$CA \mid\equiv M \mid\Rightarrow (A \mid\sim (Claim(g)=X_a)\ )$$

hence,

$$CA \mid\equiv (A \mid\sim (Claim(g)=X_a)\ )$$

On receipt of Message 5.1 of the request of certificate from M, CA remembers that it already knew the followings:

$$CA \mid\equiv (Reg,\ Claim(g)=X_a)$$

and from Message 4,

$$CA \mid\equiv (A \mid\sim (Claim(g)=X_a)\ )$$

Thus *CA* could issue certificate straight away. However, here *CA* checks the consistency of addressing by looking through a list of addresses which *CA* has certified. *NA* will be informed if there is any double-allocation of an address. Moreover, after the validation there needs to be a transaction (probably off-line) between *CA* and *NA* to confirm each other that the allocated name, X_a, is indeed usable as follows:

$$CA \rightarrow NA: \langle\langle Claim(g)=X_a, T_{CA} \rangle\rangle_{CA} \qquad (5.1.add1)$$

If *NA* perceived that $Bind(X_a)=g$, then it sends the following reply:

$$NA \rightarrow CA: \langle\langle T_{CA}+1 \rangle\rangle_{NA} \qquad (5.1.add2)$$

From (5.1.add1),

$$NA \models CA \mid\sim (Claim(g)=X_a)$$

From (5.1.add2), we can see that *NA* admits the followings:

$$NA \models Bind(X_a)=g \qquad (1),$$

and

$$NA \models Claim(g)=X_a \qquad (2)$$

As *CA* already knew that $Claim(g)=X_a$, it deduces the following:

$$CA \models NA \models (Claim(Bind(X_a))=X_a) \qquad (3)$$

CA issues a certificate for $Claim(Bind(X_a))=X_a$ in Message 5.2. From Message 5.2 as *CA* has sent the certificate, it shows that

$$CA \models Claim(Bind(X_a))=X_a \qquad (4)$$

$$M \models CA \models Claim(Bind(X_a))=X_a,$$

and from Assumption 5,

$$M \models Claim(Bind(X_a))=X_a$$

On issue of the certificate, *CA* should expire the *Reg*. Also, the timeout limit should be set in *Reg* in case that for some reason the registration would never be invoked.

From Assumption 6,

$$CA \models M \models CA \models Claim(Bind(X_a))=X_a$$

From Message 6,

$$R \models CA \models Claim(Bind(X_a))=X_a$$

From Assumption 7,

$$R \models NA \models Claim(Bind(X_a))=X_a$$

From Assumption 8,

$$R \models Claim(Bind(X_a))=X_a$$

From Message 7,

$$A \models M \mid\!\sim (Set, g, \langle\langle Claim(Bind(X_a))=X_a \rangle\rangle CA, X_c(group))$$

and

$$A \models \Rightarrow (Set, g, \langle\langle Claim(Bind(X_a))=X_a \rangle\rangle_{CA}, X_c(group))$$

hence,

$$A \models (Set, g, \langle\langle Claim(Bind(X_a))=X_a \rangle\rangle_{CA}, X_c(group))$$

$$A \models CA \models Claim(Bind(X_a))=X_a \text{ and } CA \Rightarrow Claim(Bind(X_a))=X_a$$

hence,

$$A \models Claim(Bind(X_a))=X_a \tag{6}$$

and from Assumption 7,

$$A \models NA \models Claim(Bind(X_a))=X_a \tag{5}$$

Now that A receives the certificate, $A \models Reg\_ID$ and it knows that the registration has been completed.

E.10. Information maintenance

We assume that the *information system* forwards transparently the accepted information items to the *look-up* users.

E.11. Information look-up

The sequence of packet exchanges is as follows:

Message 1) $B \rightarrow R$: B, Request, $\{ \langle\langle Auth_B \rangle\rangle_{CA}, A, ID_X, T_B \} SK_B$]

After the examination of B's access right, the answer will be sent back to B.

Message 2) $R \rightarrow B$: [$\{B, \{A, ID_X, Certificate(X), t_R\}SK_R\}PK_B$]

Protocol analysis is as follows.

From Message 1,

$$F.in \models B \models \text{(Request to R,}A, ID_X)$$

From Message 2,

$$B \models R \models \langle\langle Address(Group(X_a)) = X_a \rangle\rangle_{CA}$$

and

$$B \models CA \models Address(Group(X_a)) = X_a \tag{7}$$

As $B \models CA \Rightarrow Address(Group(X_a)) = X_a$,

$$B \models Address(Group(X_a)) = X_a \tag{9}$$

From Assumption 8 and above,

$$B \models NA \models Address(Group(X_a)) = X_a \tag{8}$$

By now the goals from (1) through (9) are all proved.

E.12. Conclusion of the Annex

In this Annex, we have analysed the protocols for the generation and registration channels through to the look-up channel. We have proved how the integrity of the information, an address, is maintained throughout the channels. The proof was presented rather redundantly. This is because it was the first attempt to use the formal notation and logic originally intended for proving authentication protocols, to prove the integrity of information flow.

We have managed to use the logic system to prove the goals; however, we have found a difficulty in this use. In an authentication protocol, the number of the parties involved in the operation would be usually at most three; two sites and an authority. The generation and registration channels involve more than three agents, and at information generation there are multiple paths for information flow; one goes though to object set up; another goes to CA. We have no clear way to differentiate an address, X_a, which is set to the object at the Configuration Phase, from the one, which is in the knowledge of CA. In this sort of situation, we may need some more explicit way to express how an information item comes through. Apart from that, our trial of use of the logic has shown that it is useful when one wants to express the flow of information on the semantics level.

The proof showed that if the address information was notarised, it would be delivered reliably to the user.

Appendix F

Measurement of delay to process ARP Requests

In this appendix, we show the measurement result of processing delay for ARP Request packets. They are used by Chapter 5. We wished to measure the actual service time for ARP Request, however, as the ARP operation was done in the kernel in a system, we measured the time simply by subtracting the time an ARP Request was observed from the time the associated ARP Reply was observed. This include queueing delay at Ethernet level and the delay for preparation of a reply packet as well as the processing delay for the request packets. The monitor was located on the same Ethernet as the host on which an "rusers" command was executed, so that reply packets can be observed by the monitor; note that the reply packets are not broadcast, so that one cannot see them over a MAC level bridge which would not forward the packets local to one Ethernet.

Table F.1 shows the satistics of the measurement. The unit of the data is microseconds. The 95 percent C.I. means the T confidence interval for the mean of the population; since the sample did not indicate that the population was a normal distribution, we only can say that the mean may fall *approximately* in this interval. Q1 is the first quarter percentile, and Q3 is the third quarter percentile; three quarters of the data was less than 422 microseconds. Figure F.1 shows the histogram of the data. Since we would like to see a rough figure of service time compared to the encryption and decryption time, we would not go further the analysing this data. Some of the data which are longer than 600 microseconds might be due to the queueing delay at the Ethernet level.

Sample size	329
Mean	426.64
Median	387.00
Standard Deviation	125.17
Minimum	318.00
Maximum	959.00
Q1	363.50
Q3	422.00
SE Mean	7.84
95.0 percent C.I.	(413.06, 440.22)

Table F.1: The statistics of the measured ARP Service Time

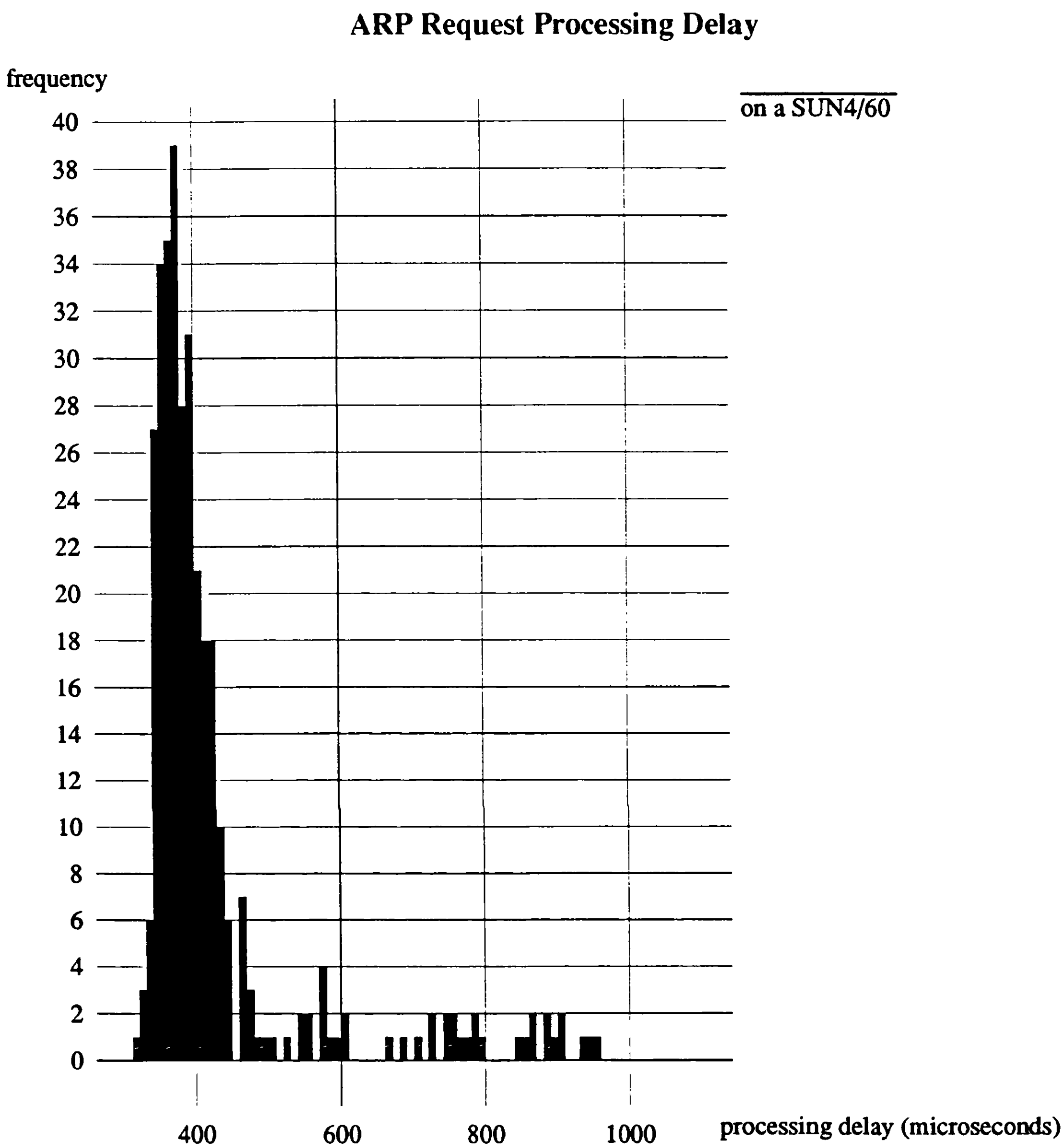


Figure F.1: The service time for an ARP Request packet