Architecture for Dynamic and Secure Group Working

Ivan Đorđević

SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Department of Electronic Engineering
Queen Mary, University of London
United Kingdom

June 2004

To my Family

Acknowledgements

I would like to thank my supervisor Dr Chris Phillips, for all his help, continuous encouragement, and belief in me during my PhD. Also, to the academic members of staff in the Department of Electronic Engineering, Queen Mary, whom I have been interacting with, for their comments and suggestions. In addition, I am thankful to Dr Theo Dimitrakos from CCLRC, for numerous technical discussions.

My PhD has been partly supported by EPSRC, Grant GR/R97733/01, which I would like to acknowledge.

I am also grateful to everyone in the Department for making my days at Queen Mary memorable, during both relaxing and working hours.

Finally, my love and gratitude goes to my family, especially my parents Živkica and Radiša, and my sister Jelena, for their love and care during the years of my study in London.

Abstract

The main motivation for this research comes from an examination of existing approaches to group working within distributed communication systems. The overall contribution is the development of a novel hybrid architecture for secure corporate communication, supporting dynamic closed user groups operating across a distributed and non-secured infrastructure. It employs protection mechanisms that are enforced at the end-user terminal but can be controlled from a scalable administration control plane.

The architecture comprises a population of client terminals and one or more administrator nodes. The formation and maintenance of groups is controlled by the administrator(s). This is achieved by using a signalling protocol with encrypted messages that contain certificates to provide authorisation and authentication. Clients liase with the administrator to request joining or creating a working group. The group members are issued with certificates specific to the group. An administrator is also able to set policy constraints, regulating the actions permitted by individual group members. This is enforced by management and policing functionality present within each client machine's operating system.

Once the group is established, communication between the group members can take place using peer-to-peer mechanisms, preceded by the authentication process based on certificates. The administrator only needs to be contacted if there is a change of group structure or operational parameters. This provides one form of scalability. Furthermore, rather than providing a single administrator, multiple administrators can be used to each manage a cohort of clients. This is particularly beneficial if cross-organisational groups are to be formed. Using a proxy mechanism, an administrator may permit a selection of its clients to join groups managed by other administrators, whilst ensuring all interaction with the remote administrator is channelled through the clients' local one in order to support policy management and vetting functions.

In addition to a detailed assessment of the relative operational performance of the architecture, various security features are proposed and evaluated, particularly relating to the access control and authorisation of clients' actions. The complete system constitutes a distributed firewall, where the administrator configures the policy rules, either individually or through agreement with other administrator nodes, if inter-organisational groups are to be formed. The firewall instantiation on each client machine, containing only the set of policies relevant to that client, is used to enforce the actions corresponding to the detected authorisation level.

Table of Contents

List of Figures

List of Tables

Glossary

3-DES	Triple- Digital Encryption Standard
AA	Attribute Authority
AC	Attribute Certificate
AES	Advanced Encryption Standard
AH	Authentication Header
ARPANET	Advanced Research Project Agency Network
ARQ	Automatic Repeated Request
CA	Certification Authority
CAS	Community Authorisation Service
CPU	Central Processing Unit
CRL	Certificate Revocation Lists
CUG	Closed User Group
DAC	Discretionary Access Control
DES	Data Encryption Standard
DH	Diffie-Hellman (cryptographic technique for key generation and agreement)
ESP	Encapsulation Security Payload
GSI	Grid Security Infrastructure
GSM	Global System for Mobile Communications
HTTP	Hyper Text Transfer Protocol
IBE	Identity-Based Encryption
ID	Identity; the unique identifier that distinguishes different entities in a given domain
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPSec	IP Security Protocol
ISP	Internet Service Provider
IPv6	Internet Protocol version 6 (with enhanced capabilities compared to its predecessor IPv4; both are now in use on the Internet)
ITU	International Telecommunication Union
JXTA	short for “Juxtapose, as in side by side”; P2P Project of Sun Microsystems
KDC	Key Distribution Centre
LA	Local Administrator
LAN	Local Area Network
MAC	Mandatory Access Control
OCSP	Online Certificate Status Protocol
OPNET	Optimum Network Performance

OS	Operating System
P2P	Peer-to-Peer
PGP	Pretty Good Privacy
PKC	Public Key Certificate
PKI	Public Key Infrastructure
PKIX	IETF Standard for Digital Certificates
PMI	Privilege Management Infrastructure
PLMN	Public Land Mobile Network
RADIUS	Remote Authentication Dial In User Service
RBAC	Role-Based Access Control
RB-RBAC	Rule-Based RBAC
RFC	Request for Comments, proposals for Internet standards
RM	Remote Group Manager
RML	Remote Members List
RNG	Random Number Generator
RSA	(Rivest/Shamir/Adleman) asymmetric encryption and authentication system
SDSI	Simple Distributed Security Infrastructure
SEM	Semi Trusted Mediator
SP	Service Provider
SPKI	Simple Public Key Infrastructure
SSL	Secure Socket Layer
SSP	Security Service Provider
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Trusted Third Party
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
UUCP	UNIX-to-UNIX Copy Program
VA	Validation Authority
VO	Virtual Organisation
VOMS	Virtual Organisation Membership Service
VPN	Virtual Private Network
WAN	Wide Area Network
X.509	ITU Standard for Digital Certificates

Chapter 1 Introduction

1.1 Motivations and Objectives

The main motivation for this research comes from looking at existing solutions for providing flexible and scalable distributed collaborative working, and especially the way security is implemented and supported within them.

With the rapid development of computer networks and Internet-based communication, there is an increasing trend of using these functionalities and new developing technologies for business-related purposes. Even as recent as 1990, most desktop computers were considered to be stand-alone machines. Inter-networking of computers on a global scale has changed the perception of computers as stand-alone machines, and the continuing exploitation of the Internet has witnessed the development of new working patterns including nomadic computing and the formation of extranets. Within this context, protection of dynamic group working environment is one of the essential aspects. With the current solutions, this is provided by protecting campus boundaries with a firewall and using encryption techniques to interconnect remote sites, while assuming that all employees located behind the firewall are trusted. However, a recent survey on security breaches in corporate e-business environments has demonstrated that 35% of respondents did not know if an attack came from inside or outside of the company network [1]. Also, the annual survey by FBI and US Computer Security Institute [2], for several years now continuously reports that significant number of computer security breaches originate inside organisations and are made by insiders, be them employees, business partners, subsidiaries or 3rd party suppliers.

In addition, lack of a common standard for e-business applications results in employing security solutions on top of the communication infrastructure, in many cases only after potential or exploited vulnerabilities are revealed on an already developed system [3]. The general drawback with such an approach is that adding security mechanisms at that stage of the system development severely undermines the performance and flexibility of the system. New means for group collaboration, communication, and large-scale, cheap dissemination of information, through enhancements of Virtual Private Networks, Peer-to-Peer infrastructures, and increased use of digital certificates infrastructures, are trying to address some of these problems. For example, recent EU initiative (see [4]) aims to address the requirements for next generation collaborative working environments. It emphasises on the need for such middleware technology to support dynamics, mobility, and security and privacy of users, collaboration through un-trusted domains, as well as management support for inter-organisational processes and processes across the disciplines.

The main objective of this research was to develop a framework that can provide a new distributed, secure working environment based on robust and efficient network mechanisms, allowing for dynamic collaboration groups without topological constraints. By bringing together aspects of peer-to-peer and client-server communication model, with digital certificates for distribution of security policies and distributed firewalls for policy enforcement, a novel hybrid architecture has been developed. The architecture offers a unified approach for supporting communication and security within cross-domain dynamic distributed collaborative groups, through: flexible and scalable way of the communication, robustness and dynamics of the group management, and security of the overall environment (both that of the users and of actual communication).

1.2 Contribution of the Thesis

The main contribution of the thesis is a design of hybrid architecture for secure and distributed collaborative working. A novel combination of peer-to-peer and client-server communication models is proposed for enabling dynamic closed user groups that are non-dependant on geographic topology or administrative domains. The architecture includes:

- A framework for dynamic and scalable security management of closed user groups, enabled through usage of digital certificates for authentication and authorisation.
- Separation of the security policy deployment model into centralised policy definition and distributed end-entity enforcement of the security policy.
- A detailed description of a set of security protocols for supporting authenticated and confidential communication within the architecture, which consists of a:
  - Hierarchical protocol for communication between administrators and their clients within an administrative domain.
  - Peer-to-Peer protocol for communication between different administrators across the administrative domains.
  - Peer-to-Peer protocol for supporting communication between the clients within a closed user group, which is totally transparent to the administrative domains and can be used equally to support intra- or inter-domain group interaction.

In addition to a detailed survey and evaluation of the security support for current distributed dynamic collaboration environments, and dissemination of the author's research findings, the contribution of the work reported in this thesis is:

- A detailed description of the functionalities and requirements for the proposed architecture, as well as possible directions to further develop and improve the architecture of Closed User Groups.
- A simulation model of the above architecture and a performance evaluation of the communication protocol in a large-scale system setup. In particular, overheads introduced

by security mechanisms and certificate manipulation have been examined through a range of different scenarios.

1.3 Structure of the Thesis

The main part of the report consists of eight chapters, including the introduction, the discussion and the conclusion. Each chapter begins with a brief description of its scope and ends with a brief summary of its outcome.

Chapter 2 reviews current approaches in providing information security, with the particular emphasis on their suitability for protecting dynamic distributed systems. The main paradigms for supporting distributed collaborative communication, and the state-of-the-art with respect to the security features deployed are reviewed in Chapter 3.

Chapter 4 introduces a hybrid architecture developed during the author's PhD research. It discusses functionalities and requirements of the architecture, and gives the details of the security protocols.

The simulation model, developed to evaluate the proposed architecture, is described in Chapter 5. Chapter 6 then discusses validation and verification of the simulation model and analyses the performance results obtained.

Chapter 7 provides a discussion and evaluation of the work, and also considers future work placing the architecture in a larger context. Conclusions are given in Chapter 8.

The author's publications and references are provided at the end of the thesis.

Chapter 2 Information Security Overview

Increased capacity of communication links and computational power of machines are stimulating new applications and roles of computers in every day life, having a major influence on the requirements for achieving *information security* within an organisation. Firstly, before computer connectivity and networking had been introduced, computers were solitary machines protected only by password and/or encryption of confidential data on the hard-disc [1]. However, since the widespread development of digital telecommunication networks in last several decades, protection of the communication has been becoming increasingly important.

From this perspective, studying security can be divided into two major aspects: *computer security*, which deals with controlled access to information using protected hardware and software, and *communication security*[2], which deals with protection of information using encryption techniques while information is stored or transmitted over an unprotected medium [5].

The aim in applying information security is to establish a *security perimeter*, defined as "the boundary of the domain (introduced by space or logical architecture of the system) in which security policy or security architecture applies" [11]. Current approaches for providing overall information security are a combined usage of several features: cryptographic algorithms (for protection of data), authentication (for authorisation of the communicating entities), and devices such as firewalls and intrusion detection systems (IDS), placed on the borders of the corporate network to protect the enterprise intranet and regulate traffic flows[2] [3]. Also, the emerging trends for nomadic computing and distributed services and applications are setting new requirements for more comprehensive and sophisticated management of security policies and information exchange, involving for example digital certificates or message time-stamping.

This chapter reviews the above technologies, and examines their applicability for securing distributed and dynamic inter-organisational infrastructures.

2.1 Basic Security Requirements and Threats

A computer system can be considered as a set of resources available for use by the authorised users. Maintaining security of the system and at the same time obtaining secure communication requires several aspects to be considered, which altogether define information security [3],[6]:

[1] In this context, a number of dumb terminals connected to a minicomputer etc. is considered a solitary system.
[2] Sometimes, communication security has been referred to as network security.

- *Confidentiality* - the ability to protect transmitted messages in a way that the intended recipients know what was being sent, while unintended parties cannot observe source and destination, content, frequency, length or other characteristics of the traffic.
- *Data Integrity* - the property of ensuring that data is transmitted from source to destination without modification in message contents (e.g. duplication, insertion in, destruction of the packet stream etc.).
- *Message Authentication* - the property of knowing that the data received is the same as the data that was sent and that the claimed sender is the actual sender.
- *Entity Authentication (identification)* – confirmation of the identity of an entity.
- *Non-repudiation* - the property that enables sender or receiver to prove the transmission of the message if the other party denies it. Thus, recipient can prove that the particular sender actually did send a certain message; also, sender can prove that the message has been received by a particular recipient.
- *Authorisation* – conveyance (to another entity) of official permissions to do or be something.
- *Access Control* – ability to limit and control the access to resources to the privileged entities. This includes authentication and authorisation of the entities involved.
- *Availability* – of (authorised) communicating parties or stored information.

There is a number of general techniques and security mechanisms that address various aspects of information security. However, depending on the type of system/ interaction that needs to be protected, some of these security requirements may be more difficult and/or more important to achieve than others. If not designed and implemented properly to fit the system they aim to protect, each of the above security requirements could represent an obstacle to the system functionality, and more importantly a potential vulnerability that can be exploited through a security attack.

A security attack can be defined as "any action that compromises the security of information owned by an organisation" [3]. More specifically (according to [7],[8],[9]), it can be defined as a formulation or execution of a plan to carry out a threat, where:

- *A threat* is an action or event performed by an attacker that exploits system vulnerability in a malicious way (i.e. that jeopardizes any aspect of information security or breaches system' security policy).
- *A vulnerability* is a known or suspected flaw in the hardware, software, or operation of a system that can be used to compromise any of the system's security requirements.
- *An attacker* is a malicious entity (normally a human user), which can be either authorised or non-authorised.

Generally, attacks on computer systems can be divided into passive and active attacks [3],[10]. *Passive attacks* involve interception of the communication without modifying the data (eavesdropping). Therefore, it is very difficult to discover them and the emphasis is more on prevention rather than detection. *Active attacks*, on the other hand, involve modification of the data, either by modifying the original or by creating a false data stream, or modifying stored information. Both of these types of attacks can be performed either from *inside* or *outside* of security perimeters, which is another important classification of computer attacks [11]. Some of the most common (types of) attacks, based on the way they are carried out are listed below [3],[8],[9],[12],[13]:

- *Denial of Service*: technique of disrupting system by denying use or degrading the service. In this case, the attacker is not so much trying to reach the secret data as to prevent or delay the system operation, by re-directing the traffic or flooding the system with false data streams.
- *Social Engineering*: the attacker tries to persuade someone in an organisation to disclose sensitive access-control information, such as user IDs and passwords.
- *File manipulation:* creation, removal or modification of data.
- *Brute Force*: searching for password or username by trying every possible combination.
- *Meet-in-the-middle*: this attack assumes that a 'third party' is involved in communication between two entities, where none of them is aware of third party's presence. The attack is performed in manner that third party presents itself as entity B when communicating to entity A, and vice-versa.
- *Sniffing (passive wiretapping)*: eavesdropping the traffic flow on the communication link in order to gain knowledge of information it contains.
- *Session Hijacking*: the attacker attempts to take control over one side of an existing (authenticated) connection. Since authentication generally takes place only at the start of a connection, this will allow the attacker to fully masquerade as the other side without further security checks.
- *Spoofing (identity forgery)*: the attacker masquerades as a trusted/authorised entity in order to gain access to a system.
- *Viruses, Trojan Horses and Worms*: a program or piece of software with a potentially malicious function that runs on the host machine without the owner's knowledge. Some of them may replicate themselves, consume resources destructively or propagate through the system. This is the most common computer security threat, since viruses and other types of malicious code can be easily shared through data exchange, particularly by receiving infected document from a known party (e-mail attachments are the source in 80% cases [14]).

2.2 Basics of Secure Communication

This section gives a brief reviews of the general properties of current cryptographic techniques. A more detailed overview and in-depth description of various mechanisms can be found in [3],[6].

2.2.1 Symmetric Cryptography

Symmetric cryptography, also known as conventional or single-key encryption was the only cryptography technique used before the invention of public-key encryption in 1976.

The main property of the symmetric encryption technique is that the same key that is used to encrypt the data is also used for decryption and recovering the clear text (Figure 1a). Encryption and decryption are inverse processes, usually based on mathematically complex non-linear permutations. Frequently used algorithms, such as DES and 3-DES (Data Encryption Standard), AES (Advanced Encryption Standard) [15] etc., have been widely studied for their strength and security. However, a way of breaking them with significantly less effort than a brute-force attack has not been found.

For a good algorithm, it must be impractical to decrypt a message on the basis of known ciphertext and the algorithm itself, meaning that the security depends only on the secrecy of the key used, which would normally need to be exchanged by some out-of-band means (i.e. non-electronically, face-to-face). However, key management is a big drawback of the symmetric cryptography approach. Since both the sender and the receiver must share the same key, efficient key distribution is very difficult to achieve in large environments without compromising any of the keys.

2.2.2 Public-Key Cryptography

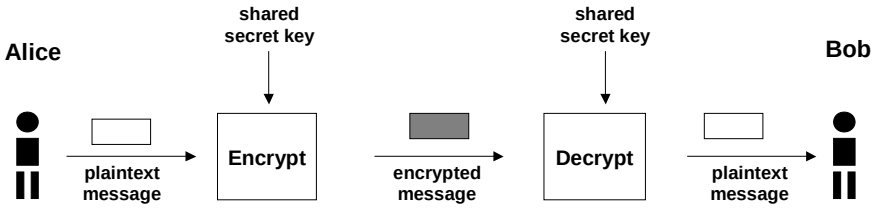
Public-key cryptography (also known as asymmetric encryption) was first proposed in 1976 [16]. A year later the first practical scheme was proposed (RSA cryptosystem [17]), and there have been several more realisations of public key systems since [3],[6].

An important property of public-key cryptography is that the key used for encryption cannot be used to decrypt the data; a different key is needed to recover the clear text. This key pair is called a public key and a private key, respectively. They are uniquely related through a specific mathematical operation that may differ depending on the scheme used [3]. The essential property of public key cryptography is that it is mathematically infeasible to compute the private key if the public key is known. This enables two parties (e.g. Alice and Bob) involved in the

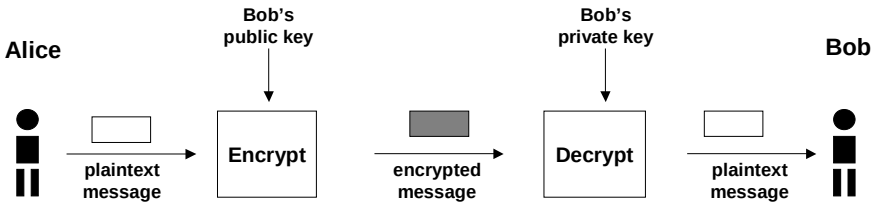
[3] For example, RSA uses factoring of prime numbers as its basis, while some other approaches are based on a discrete logarithm, elliptic curves, etc.

communication to freely exchange their public keys, whilst keeping their private keys to themselves.

If Alice wants to send the confidential message to Bob, she encrypts it with Bob's public key. The message can be decrypted only with Bob's private key (Figure 1b). Since Bob is the only one possessing this private key, the message secrecy is assured.



a) Confidentiality with Symmetric Encryption



b) Confidentiality with Asymmetric Encryption

Figure 1: Encryption/Decryption Process: a) Symmetric; b) Asymmetric

A major obstacle to the wider use of asymmetric cryptosystems is computational speed when compared to symmetric encryption. For example, [3] reports that it takes about 1000 times longer to encrypt the same data with RSA (an asymmetric algorithm) than with DES (a symmetric algorithm). Therefore, practical applications tend to use symmetric keys for the encryption of the actual data, whereas the asymmetric encryption is normally used for authentication and exchange of secret (symmetric) keys.

In addition, both schemes (as explained in Figure 1) are prone to the famous meet-in-the-middle attack. The issue is somewhat similar in both schemes, as the emphasis is on the authentication of the entities, rather than the confidentiality of the communication; therefore sharing of a secret key in the symmetric scheme, and exchange of public keys in the asymmetric scheme are analogous problems. The problem is that the entities involved do not have any assurance in each other's identities. Through impersonation, a third party can stand 'in the middle', presenting

itself as Bob to Alice (and vice versa), intercepting and potentially modifying all the communication, while Alice and Bob are not aware of the deception.

In order to provide authentication for the communicating entities, a form of message signing is introduced.

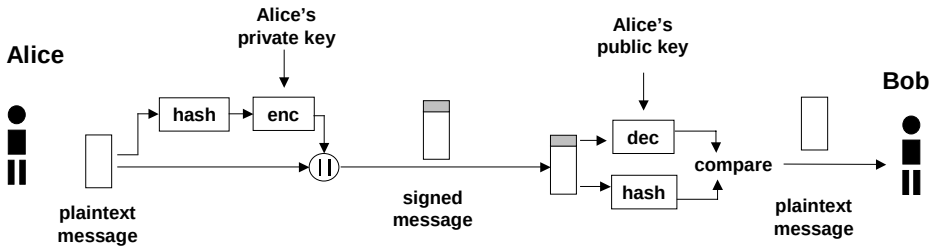
2.2.3 Digital Signatures and Hash Functions

Public-key cryptosystems provide a mechanism for message signing by their very nature. In a same way that message sent by Alice can be encrypted with Bob's public key (and upon receipt decrypted with his private key), it is possible for Alice to encrypt the message with her private key. Upon receipt, Bob will be able to decrypt the message using Alice's public key, but so will anyone else who possess Alice's public key. Therefore, this approach does not provide confidentiality, but it assures that the message has been sent by Alice (since she is the only one in possession of her private key). This operation is called *digital signature* and it forms the basis for authentication of entities and message origin. It is possible to achieve authentication by using symmetric encryption only – if a symmetric key is shared *only* by the communicating entities; this, however, assumes secret distribution of keys, a requirement which is avoided by using public-key cryptography. There are systems for secure communication over the public network that use only symmetric encryption (such as Kerberos system developed at MIT [18]), but that normally involves a rather complex authentication procedure and does not scale well in large, inter-organisational environments [4].

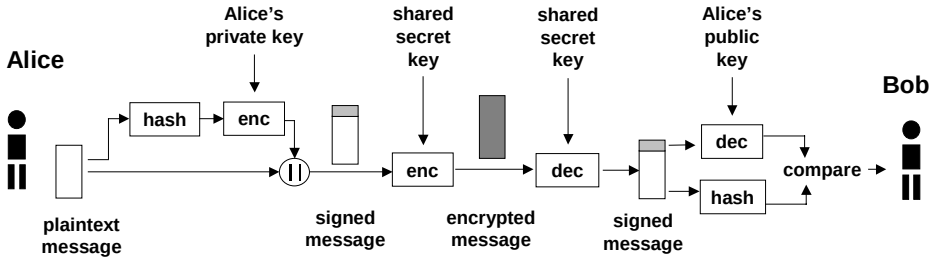
In order to make a digital signature with public-key scheme more practical, and if confidentiality is not needed, the encrypting operation is applied only to a small part of the message called the *hash*. A hash is created when a one-way function is applied to the whole message, giving a fixed-size string from the message of any size. For example, if Alice wants to send an authenticated message to Bob (Figure 2a), she would encrypt only the hash value with her private key, add it to the original plaintext message and send to Bob. Upon receipt, Bob decrypts the hash using Alice's public key, and at the same time applies hash function to the rest of the message (i.e. the original message). If those two values match, he is assured that the message originates from Alice.

If confidentiality is needed in addition to authentication, Alice could encrypt the signed message with Bob's public key. In practice, however, encryption of full message would be normally performed with a symmetric encryption, using previously distributed symmetric key (e.g. through mechanism in Figure 1b). This process is depicted in Figure 2b.

[4] Motivated by rapid development of supproting infrastructures for public-key cryptohraphy, current advancements in Kerberos system are considering use of public-key cryptohraphy for initial authnetication [19].



a) Authentication with Asymmetric Encryption



b) Authentication and Confidentiality

Figure 2: a) Digital Signature with Public-Key Scheme; b) Authentication and Confidentiality through combined use of Asymmetric and Symmetric Encryption

The issue with digital signatures is the trustworthy distribution of public keys, since a genuine copy of the sender's public key is required by the recipient. For example, Bob may receive a public key from someone that he thinks is Alice, but in electronic environment he has no means to check whether it is really Alice or someone impersonating her. He will receive messages with a valid signature, and he can send encrypted messages that only 'Alice' will be able to decrypt and read, but none of the above techniques provides a means to relate someone's identity to the keys he or she presents. This problem has been addressed through digital certificates.

2.3 Digital Certificates and Supporting Infrastructures

2.3.1 Authentication

In order to increase the confidence in the authenticity of the public keys, the concepts of public-key certificate and Public-Key Infrastructure (PKI) were introduced [20],[21]. A public key certificate is a statement, digitally signed by the authorised entity, which confirms authenticity of someone's public key[5]. PKI is a supporting security architecture that facilitates publishing and management of public-key certificates.

[5] The first notion of digital certificate dates from 1978, from Loren Kohnfelder's bachelor's thesis in electrical engineering from MIT [22]: "Diffie and Hellman introduce a central authority known as the Public File. Each individual has a name in the system by which he is referenced in the Public File. Once two communicants have gotten each other's keys from the Public File they can

The main property of the public key certificate is that it relates one's public key with the additional information (which in some form identifies the entity – key owner). This relationship is achieved with a digital signing of the body of the certificate, which consists of a number of fields. At creation (*certification*), when all the data of interest is put together, the whole certificate is digitally signed by the issuer (by hashing and encrypting with the private key, as explained in the previous section). This additional piece of information represents the digital signature, and together with the rest of the data forms the certificate. Therefore, the certificate effectively represents the signed (but not encrypted) message. If one needs to *validate* the certificate, one first needs to obtain the public key of the certificate issuer, and then to perform the validation, generally following the process depicted in the right-hand part of Figure 2a. Any of the public-key cryptography algorithms could be used for building a PKI. However, the RSA system is particularly suitable, due to a property that lies in the very nature of the RSA algorithm. In any PKI system, signing of the certificates (done with a private key) is performed only once, whereas the validation (done with a public key) is performed many times. The underlying property of the RSA algorithm is that a 'public-key operation' takes much less time than a 'private-key operation', and indeed much less when compared to the corresponding public-key operations performed with Diffie-Hellman or elliptic curve encryption [23].

The processes of *certification* and *validation* are two basic operations common to all PKIs [24]. However, current PKI systems can be divided in two main groups based on the certification process, namely which entity is authorised to sign the certificate.

In the first approach, known as PGP (Pretty Good Privacy, [25]), users would normally act as their own issuing authority. The PGP framework [26] specifies a protocol for message exchange, encryption algorithms, and certificate fields of which the main ones are: certificate holder's public key, the certificate holder's information, and digital signature(s). Initially, at the certificate creation, the certificate owner (e.g. Alice) puts together her public key and some form of her identity (e.g. Alice Smith, alice@hotmail.com, etc.), and signs it with her own private key. This self-certification is one of the main PGP properties, aiming to remove certificate hierarchy and introduce flexibility in the certificate-supported communication. The scheme also allows extension of signatures, which is its unique property: if Bob interacts with Alice and has confidence that Alice's PGP certificate reflects Alice's real identity, he can also sign her certificate, additionally verifying it. Furthermore, while signing it, he can associate his own expression of confidence in Alice and the validity of her certificate. This approach aims to create a so-called 'Web-of-Trust', without a strict certification hierarchy, using 'chain of

securely communicate. The Public File digitally signs all of its transmissions so that enemy impersonation of the Public File is precluded." (Since primary reference was unavailable, the citation was taken from [42].)

authenticators', where trust is built through the interactions between the entities, and is reflected through the numbers of signatures and levels of trust associated with each of them [27]. PGP was originally developed for securing e-mails, and operates successfully in small, informal surroundings. However, in addition to scalability, it has some security concerns for its practical usage in commercial situations; there is no entity (even the certificate owner) ultimately responsible for issues related to certificate validity, actions performed, etc. This is in strong conflict with the environment where business interests are protected through a "well-defined contract with loss responsibilities and fines" [28].

The other approach to PKI, X.509 defined by ITU (International Telecommunication Union) [20] and PKIX of IETF (Internet Engineering Task Force) [29],[6] introduces the hierarchical relationship between the certificate verifiers and public-key certificate holders (owners of public-private key pair). Users' public keys are created and signed (verified) by a recognized entity called Certification Authority (CA). This type of infrastructure has several important features which are discussed in the remainder of this section as follows (for more details, please see [24],[30],[31]):

Certificate Verification process consists of creating and signing a certificate. It is performed by the CA, and it can be done in two ways. In the first approach, the user creates its own public-private key pair, and delivers the public key to the CA, which creates the corresponding certificate and signs it. The benefit of this approach is that it does not jeopardize the user's private key by sending it over the unsecured network. In the other approach, the CA creates key pair for the user; it also creates and signs the public-key certificate and delivers it to the user. The drawback of this scenario is that user's private key also needs to be delivered, and in a secure way; the potential benefit (in a corporate environment) is that the private key is also known to the CA. For example, if Alice is dismissed, the organisation can still access her work-related files by obtaining her private key from the CA (the feature called *key escrow*). However, both of the approaches assume initial mutual authentication between two parties, which ideally needs to happen off-line.

CA Arrangements and Certificate Delegation. Ultimately, each PKI has its source of the certification path, known as a *root CA*, which is self-certified and which is implicitly trusted. In order to avoid the situation where one CA would need to verify the certificates of all the users (on a global scale), the root CA can certify other CAs, by *delegating* them a right to issue certificates. This effectively creates a chain of certificates, where the last CAs in hierarchy can issue certificates only to users, but cannot certify other CAs. CAs are normally arranged in a top-down hierarchy, where parent CAs can certify only their children. When communication

[6] These two approaches are essentially the same. PKIX has initially adoopted ITU X.509 standard, and develops it further for the particular use on the Internet.

between two users takes place, they need to validate the certificates of all the CAs that separate them in order to verify each other's public keys (a so-called *certificate path validation*). Therefore, scalability of the PKI depends on the length of the certificate chain. In order to decrease the length of the certificate path, *cross-certification* between the CAs of different branches (on the same or different levels of hierarchy) is supported, which on the other hand may add to the complexity of the certificate path discovery. An additional drawback is that the trust of the whole PKI is put in the root CA.

Certificate Validation is a process of examining whether the specific certificate is valid at the time of usage. Two methods for validation are supported in the PKI: online validation (certificate is validated every time it is used), and offline validation (certificate contains validity period, defined at the creation time).

Certificate Revocation is a method for revoking the certificates that are no longer valid (e.g. if user's private key is stolen, if any details in the certificate are changed, etc.). The revocation mechanism is closely related to the supported validation method. Although a number of approaches have been proposed, this is commonly recognized as an open problem in the PKI research. The most common revocation techniques are:

- Certificate Revocation List (CRL) [32]. This is the most widely used approach. It is a list which contains all the revoked certificates where the validity period has still not passed. It is periodically signed and issued by the CA in general, but the CA may delegate this responsibility to other authorities (validation authority – VA). This approach requires user to download the whole CRL if it wishes to validate a particular certificate. In large environments CRLs can become very long, for which reason the alternative Δ-CRLs were introduced. Δ-CRL is effectively the list containing the updates from the previous posting of a CRL. The common problem with revocation lists is that a certificate becomes revoked only when it is on the list, and not when its validity actually stops. Therefore, the frequency of updating CRLs is obvious trade-off between the scalability and security.
- Online Certificate Status Protocol (OCSP) [33]. This enables timely information regarding the revocation status of a certificate. To validate the specific certificate, a user sends a request to validation authority (VA), which sends back signed response. This approach eliminates CRLs. However, since the request is sent at every user-to-user communication, performance is an obvious concern. One of the proposed improvements is to have several VAs, which brings security concerns, since the VA's private key is multiplied over a number of entities (since every response needs to be signed).

There are approaches that aim to solve the problem of certificate revocation. For example, [34] introduces an entity called the SEM (semi-trusted mediator) that works in conjunction with the CA. The private key of each user is split in two parts, of which one is kept at the SEM, and another with the user. If the user's certificate is revoked, it is not granted by SEM to either encrypt or sign the message. However, this mechanism has some scalability issues since user

needs to query the SEM for decryption of every message received and for signing of every message sent. A way of solving this is to replicate SEMs, which on the other hand brings security considerations, especially in large, inter-organisational environments, as the authors suggest themselves.

A public key certificate contains a number of fields. There are several versions of PKIX/X.509 certificates. The latest one, version 3, supports the following fields in the certificate [32][7]:

- *Version* (i.e. 1, 2 or 3)
- *Serial Number* (must be unique per CA)
- *Issuer* (the field that uniquely identifies the CA)
- *Validity Period* (contains time of issuing and time of expiration; global reference time needs to be agreed or specified within the certificate)
- *Subject* (the field that uniquely identifies the key owner)
- *Subject Public Key Info* (value of user's public key and identifier of the encryption algorithm with which key is used)
- *Extensions* (this field is fundamental distinction to v01 & v02 which do not have it); this can contain a number of things, such as: the additional attributes about certificate owner, constraints on certification path, reference to CA's public key (corresponding to the private key used to sign the certificate), reference to CRL where the certificate should appear if revoked, etc.
- *Signature Algorithm* (reference to a specific algorithm used by the CA to sing the certificate with its private key)
- *Signature Value* (CA's digital signature, with which CA certifies the validity of the information in the rest of the certificate).

X.509 is a result of a significant effort over the period of last 10 years, and as such represents a comprehensive infrastructure for supporting trust and security of the communication over the insecure Internet. However, one of the main pitfalls of the paradigm is a requirement to be achievable on the global scale. It has proved highly impractical (and perhaps impossible) to create globally unique identifiers of certificate holders. Another obvious issue is legal aspect and responsibilities of the CAs and other parties in case of breach of the agreement, a way to pursue corrective actions, etc., especially on the global, international scale. For example, there is a number of successfully operating CAs [38], but they are normally grouped in different PKIs that conform the law of different countries, company policies, etc. This goes so far that Verisign [39], one of today's most recognized commercial CAs, explicitly states in the certification

[7] Section 4 of [32] describes these in detail, and Appendix C of the same document lists some examples of PKIX v03certificates.

contract that it is exempt from all the responsibilities regarding the certificate, accuracy of its data, and its usage (see point 5 of [40]).

It is a common observation that achieving a global PKI is currently not feasible, but that it can be very useful approach for managing and supporting security within closed/isolated environments.

The certificate infrastructures described in this section assume the systems based on RSA encryption method. One of the common problems for RSA-based systems is key generation: due to a large keys (1024 or 2048 bits) the process is time-consuming, and it also requires a key to be stored on an electronic medium, since it would not be possible to remember it. A recent proposal [35], based on elliptic curve cryptography, offers a practical solution to this problem. It is called Identity-Based Encryption (IBE), and it has been developed particularly to secure e-mail communication [36]. It allows any string to be used as a public key[8] (in particular e-mail addresses). It employs the mechanism where the 'master secret' is kept at the central server (presumably a CA), and for every interaction a private key is calculated based on the public key and the master secret. Similar to the previous scheme, the central server can refuse to grant a decryption key. However, the mechanism has some performance concerns, firstly since user needs to query server for decryption of every message received, and secondly since the private key needs to be computed every time. Also, since a public key of a user can be chosen as a string that relates to a user's identity, it multiplies the number of keys used (e.g. is it alice@hotmail.com or alice@yahoo.com ?), makes typing errors a significant problem (e.g. Alicia instead of Alice), and makes key revocation more complex (if the key is related to the user's privileges or e-mail address) [37]. This is still new approach and does not interoperate with current PKI systems. It has been mentioned here for clarifying its existence, rather than as a serious consideration within this research.

2.3.2 Authorisation

The main motivation for developing public key certificates and PKI systems was to enable secure and controlled authorised access to the resources in a structured way. The main shortcoming of authentication certificates is their lack of expressiveness, which limits their ability to support the requirements for several levels (most commonly hierarchical) of authorisation. Since they bind owner's public key to its name (or some other form of identification), they can normally express which entity is permitted access, but do not specify what level of access should be allowed. Version 3 of the X.509 authentication certificate improves this to some extent, by allowing certain freedom in defining the 'extensions'

[8] Therefore the name of the scheme.

contained in the certificate. However, there are a number of approaches developed in the last five years, specifically addressing the problem of authorisation via certificates.

The SPKI (Simple Public-Key Infrastructure) certificates [41],[42] were initially developed to address the problem of globally unique names in X.509 authentication certificates. Instead of trying to find a unique name for the user and then relate it to a public key via certificate signature, SPKI certificates consider public keys themselves to be globally unique user identifiers. (This approach is supported by the very basic assumption behind cryptography, that the encryption keys are unique[9] - otherwise, what is the use of them anyway?).

Therefore, SPKI certificates contain the issuer's public key (that identifies certificate issuer) and the subject's public key (that identifies key owner). Including the actual names of certificate issuer and owner within the certificate is completely optional. The rest of certificate fields are: authorisation rights, validity period, and boolean 'delegation' flag. This structure enables the certificate issuer-key to grant certain privileges to a subject-key, and also to specify whether these privileges can be further delegated by the subject-key. All the information is signed by the private key corresponding to the issuer's public key. A subject-key (certificate holder) can further delegate the authorisation contained in the certificate by acting as an issuer [43].

The consequence of this approach is that the last subject-key in the delegation chain needs to present all the certificates in order to claim certain authorisation rights. By the very nature of the system, certificate chains can become very long, therefore making validation process time-consuming. SPKI system allows chain 'reduction' in two ways, both of them resulting in one certificate with intersected authorisation rights and validity period. For either of two approaches, the whole certificate chain needs to be presented by a subject: 1) if presented to the first issuer, it can create new certificate and sign it with its private key; 2) if presented to a server, the server can create new certificate and sign it with issuer's private key. The first approach assumes that the original issuer needs to be located, and also increases its communication burden. The second approach assumes that a server is a type of secure server that keeps all the private keys, in addition to providing services, which may not be acceptable in a fully distributed environment.

Although the SPKI approach gave a possible solution to unique identifiers (via public keys), it created a problem of relating a public key to a real identity. The problem was addressed by taking the approach of a *local namespace* proposed in SDSI scheme (Simple Distributed Security Infrastructure) [44], and eventually two schemes merged into SPKI/SDSI [45], an experimental project of the IETF SPKI working group [46]. Effectively, the original 'SPKI

[9] For example, [3] reports estimations made in 1996 that 430-bit RSA key could be obtained by brute-force attack in 500 MIPS-years (milion instructions per second).

certificate' is split into two: name certificates and authorisation certificates. Through name certificates, an issuer can introduce new entities into its local namespace, by relating their public keys to some locally defined identifier and signing it with its public key. Authorisation certificates preserve all the features of already described 'SPKI certificates', with the additional flexibility of defining certificate owner either through its public key, or through an identifier defined within certificate issuer's local namespace. An additional capability of name certificate is that it enables the issuer to define links between different identifiers from its local namespace. Therefore, it can define an 'abstract' identifier and relate it to the 'real' ones. This provides straightforward way of grouping, which is the unique feature of this certification system.

The SPKI/SDSI approach was developed to operate in distributed environment without defined hierarchy. However, applying it to a corporate, inter-organisational environment creates several concerns. The delegation of authorisation certificates introduces scalability, but once delegated, the issuer completely looses control on how far the authorisation will propagate; it cannot track the chain, and the only way to gain such information is if the whole chain is presented to the issuer for reduction[10]. Also, the way authorisation rules are defined depends only on the certificate issuer, and in the system where anyone is allowed to create such certificates (or modify the existing ones) it may be difficult to achieve a common representation. Finally, the scheme does not support certificate revocation at all, and completely relies on short lifetime of the certificates.

Another approach that addresses the problem of authorisation via certificates is developed within IETF' PKIX working group [29]. Some of the requirements for authorisation certificates have been addressed in the independent proposal called 'Smart Certificates' [47], which suggests modifications and enhancements of X.509 public-key certificates. Most of these are covered by IETF's Attribute Certificates (AC) [48], which are fully compatible with version_2 of X.509 public-key certificates and further extend their functionalities. The paradigm of X.509 attribute certificate and supporting infrastructure is known as Privilege Management Infrastructure (PMI).

The main difference between a public-key certificate and an attribute certificate [11] is that AC does not contain public-key. Instead, link between the owner's PKC and AC is provided through a dedicated field within AC. Therefore, AC is considered to be valid only if presented with valid PKC. In this sense, AC can be seen as more powerful way to grant certain attributes

[10] The obvious improvement could be to specify the length of chain, but such a feature has not been introduced in any version of SPKI/SDSI scheme.

[11] Further on, if abbrevations PKC and AC are used, they will refer to IETF's PKIX X.509 public-key and attribute certificate, respectively.

to an entity, which would be otherwise contained in the *extensions* field of PKC. Such an approach has several advantages:

- Firstly, the validity period of authorisation rights is not necessarily the same as the validity of someone's public key (e.g. if Alice has access to Bob's files while they are working on the project, this privilege may need to be revoked after the project is finished; however, Alice would still remain the employee of the organisation, identified by her PKC).
- Secondly, the issuer of PKC may not be eligible to define authorisation policy – separation of certificates enables separation of the certificate issuing authorities. The authority issuing ACs is known as *Attribute Authority* (AA). As stated in [48], the AA and CA (Certificate Authority) should not be the same entity.
- Finally, authorisation-related information is very often confidential; the separation between PKC and AC enables entities first to establish a secure connection (by authentication each other via PKCs), and then to make a specific request (by presenting the AC). Some of the attributes can be encrypted by AA before the attribute certificate is signed. This feature is optional, and if used it is targeted for set of predetermined recipients.

The attribute certificate consists of the following fields:

- *Version* (v2)
- *Holder* - the field that uniquely identifies the holder of the authorisation rights, through one of the following means: reference to holder's public key certificate, entity name (as specified in the *subject* field of its public key certificate), or reference to an object (such as entity's public key)
- *Issuer* (the field that uniquely identifies the AA)
- *Serial Number* (must be unique per AA)
- *Validity Period* (contains time of issuing and time of expiration; global reference time needs to be agreed or specified within the certificate)
- *Attributes* – contains information about certificate holder. Examples of common attributes (but not limited to) are: service-specific authentication, identity to be charged (for the service), group membership, role (of the AC holder). If the certificate is used for authorisation, this field will normally contain set of privileges, as a sequence of attributes.
- *Extensions* – similar to public key certificate, this field can contain reference to AA's public key (corresponding to the private key used to sign the certificate), reference to CRL where the certificate should appear if revoked, etc. However, this field would not contain authorisation-related data, as it may be the case in PKC's 'extensions' field.
- *Signature Algorithm* (reference to a specific algorithm used by the AA to sing the certificate with its private key)
- *Signature Value* (AA's digital signature, with which AA certifies the validity of the information in the rest of the certificate).

General framework for management of ACs and PKCs is to the large extent analogous. However, there are few distinctions. According to [48], the use of AC chains is not recommended. Also, it is possible to issue short-lived ACs without revocation. Validation of ACs is typically performed when the appropriate PKC is presented in addition (or beforehand). The link between two certificates, defined at the AC's creation, enables recipient to relate them and authenticate the sender before the authorisation can take place.

2.3.3 Access Control and Trust Management Systems

Once authorisation certificates are used to allow (or constrain) a users' access to the resources, the question of what the attributes (or authorisation rights, in case of SPKI/SDSI) represent becomes apparent. This is closely related with the way access-control policy of the system is defined. Traditionally, two approaches were used: *Discretionary Access Control*, DAC (specific authorisations, either as permissions or prohibitions, are defined for each user) and *Mandatory Access Control*, MAC (security level are defined for users and objects, and strict hierarchy defines relationships among them). The approach of *Role-Based Access Control* (RBAC) started emerging in the beginning of 1990ies, and it has been proved since that DAC and MAC are special cases of RBAC. Current state-of-the-art in the area (see Sandhu, [50]) suggests that the Role-Based Access Control (RBAC) is currently (and will remain in the foreseeable future) a dominant model for access-control [12].

The biggest improvement of RBAC is more abstract way of treating security policy. Instead of specifying all the accesses each user is allowed to execute, access authorisations are specified for roles. Each role, defined as 'set of actions and responsibilities associated with a particular working activity' [49], is more generic and includes many users (also, one user can hold a number of roles in various contexts). Specification of users' authorisations is divided in two independent parts: one to assign users to roles, and another to assign access right to the roles. Therefore, it was a natural approach to include roles as (some of) the attributes within attribute certificates [52]. During the communication, by associating role-attribute with the access control mechanism at recipient, the appropriate privileges are retrieved and authorisation is granted. If, for example, Alice's role changes, she needs to obtain new attribute certificate (AC) that describes her new role. On the other hand, if an organisation's policy changes, privileges associated with each role (or some of them) will change. Consequently, an *access-matrix* at each

[12] However, access control is only one aspect of role-based policy management. Comprehensive policy specification languages, such as Ponder [51], allow complex relationships to be expressed within the policy deployment model, addressing aspects such as authorisation, delegation, information filtering, and refrain policies. One can argue that the appropriate role-based model can capture these aspects via attributes suitable for certificates, but that discussion, no matter how attractive, goes beyond the scope of this thesis.

of the access points needs to be updated [13]. However, Alice (and other users) do not need to obtain new certificates. This has obvious advantages over the approach where particular privileges are specified within the authorisation certificate (as proposed in SPKI/SDSI [14]), in which case changes in the policy structure would force re-issuing of Alice's certificate.

RBAC was originally developed to enable more flexible policy management in closed environments, where security administrators assign roles manually to the users, based on predefined user's responsibilities in the organisation. Work in [53], and subsequently in [54] discusses the problem of automated 'mapping of users to roles'. In [54], Rule-based RBAC (RB-RBAC) is proposed, a mechanism that allows automatic assignment of roles to users, based on the credentials they present. In [53] the authors propose credentials that can be carried as authorisation attributes within the certificates. Different services in a distributed system may need only a subset of the privileges associated to a role in order to grant access. If the role is described as a set of rules, it can be 'decomposed' into several certificates. At the access request, only the certificate(s) with relevant rules can be presented in order to claim a privilege, therefore revealing only a minimum of the authorisation rights needed for access. For example, Alice's role as a 'senior manager' may include privileges to sign expenses, read administrative files, and receive emails. On the other hand, Bob's role as 'secretary' may include privileges to read administrative files, print, and send emails. With RBAC, Alice and Bob would need to present a certificate confirming their full role, regardless of what action they want to take. With RB-RBAC, these two roles could be broken into three rules. For reading administrative files, both Alice and Bob would have a certificate with the same rule. For other actions, they would have a certificate each, with different rules. Their roles would be described with two certificates each, but they would need to present only one in order to gain access to any service they are authorised to use. This makes it more difficult for a malicious entity to discover their role and full privilege rights.

Recently, several infrastructures for supporting access control through credential-based authorisation in distributed environments have been proposed. Most representative of these infrastructures, commonly known as *trust management systems*, are KeyNote [55], PERMIS [56], and Akenti [57]. Essentially, they consist of:

- A policy language for defining security policy by means of: access rules to the resources, and privileges of users.
- An authorised entity ('root of trust', 'source of authority', etc.) that verifies access policy to the resources and privileges of users, and prescribes it by means of digitally signed piece(s)

[13] Access-matrix is logical representation of relationships between users and access objects, based on which relationships between roles and privileges can be derived. As pointed in [49], more optimal implementation methods exist in practice.

[14] The author does not argue that it is not possible to apply role-based approach onto SPKI/SDSI scheme, but rather that it may be very difficult to maintain coherent and consistent definition of roles in such a distributed system without introducing some degree of hierarchy.

of information (e.g. authorisation certificates). The two types are essentially different, as they carry *requirements* to access the resources or *credentials* associated with users.

- A mechanism ('compliance checker', 'policy decision point', etc.) that, upon user's request, combines relevant policies with user's credentials and compares it against the request. It is responsibility of this mechanism to gather all the relevant information, part of which may be presented by a user, resource, or stored elsewhere. Based on the outcome, the access control point is advised to grant or deny request.

Although the general approach is very similar, the above-mentioned systems fundamentally differ in the way their policy language operates, and also at the level of implementation [58] [15]. However, as already stated, discussion on the capabilities of policy languages, and policy deployment model in general, goes beyond the scope of this thesis. The motivation here is to discuss the main architectural characteristics of trust management systems, as they are powerful mechanisms to support secure and authorised interactions in distributed environments.

KeyNote [55] takes SPKI-like approach, by relating authorisation rights to a public key and allowing delegation of the authorisation. It does not specify how credentials are created, nor prescribes any certification system to be used. It does not give details of where and how policy information is kept, nor what type of authentication should be used. That is left to the installer, while the system relies on the common representation of simple expressions of policy language to match requirements and credentials and make a decision. A unique entity called 'root of trust' initially defines policy, and grants certain privileges (which is effectively a portion of the policy) to the users. In the same way, users can further delegate (any part of) privileges they possess. This loose arrangement removes nearly all hierarchy and makes it difficult to deploy more structured role-based approach [16]. This also makes it difficult to make more detailed parallel with PERMIS and Akenti, which are much more specific proposals.

The general architectures of PERMIS [56] and Akenti [57] are very similar. Both infrastructures build on top of PKIX X.509 Privilege Management Infrastructure; they are developed to support RBAC policy authorisation, and do not support delegation of privileges. Central authority defines policy and stores it in the 'policy' certificate: in case of PERMIS, a whole policy is stored in one certificate, whereas in the case of Akenti, different certificates may be issued for different resources, containing the relevant part of access policy. This information is stored at public directories (of which there can be several). Another type of certificate supported is the authorisation certificate, created by authorised entities for users, and carrying users' role and

[15] Note that there is no uniform implementation of KeyNote, since [55] outlines only general approach, and leaves a lot (perhaps too much) freedom to the developers.

[16] In order to be able to do so, more elaborate expression of relationships and constraints is needed. As evaluated in [57], KeyNote policy language is not explicit enough to support that.

additional privileges [17]. Both approaches deploy the pull model, where authorisation certificates are stored in public directories (which also store certificate revocation lists), and are 'pulled' by a policy decision point on as-needed basis. Akenti explicitly requires authentication via X.509 public-key certificate (only strong authentication) and relies on external Certification Authority for that purpose, whereas PERMIS is 'authentication agnostic' and accepts both public-key certificate and username/password authentication.

Both schemes allow resources to be controlled by several authorities, in which case all the relevant policy-certificates need to be gathered and evaluated. In any case, the particular decision is made by a policy decision point at the run-time. Such an approach assures up-to-date policy information, but still assumes that centralised point(s) need to be contacted at every user's request, which may create a bottleneck in large environments or disrupt a service (if some of the central points are not available) [18]. Another concern of this approach is a requirement for the systems' policy and users' credentials to be stored in public non-secure directories, which creates an attractive target, either for Denial-of-Service, or intrusion with the aim of stealing policy information.

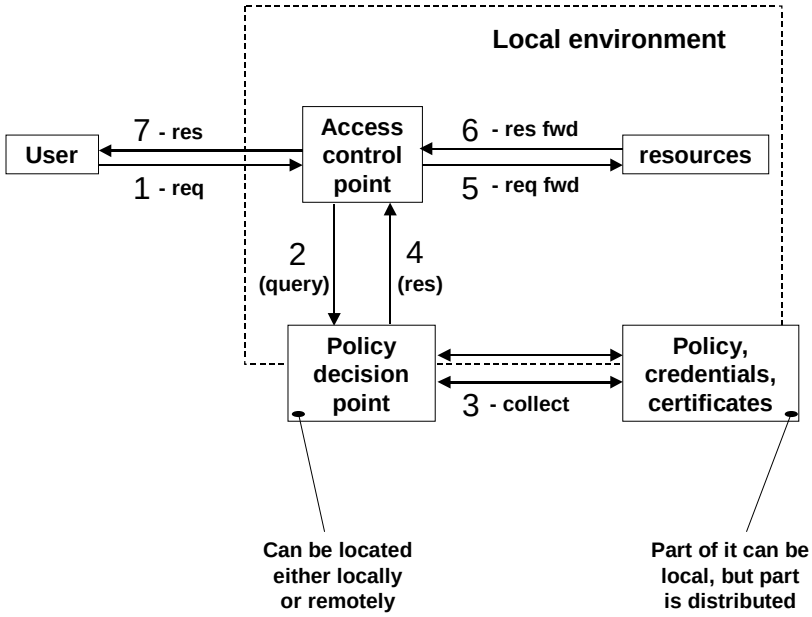


Figure 3: Typical Architecture of Trust Management Systems

[17] The 'additional privileges' are interpreted different in two approaches, due to (optional) capability of role inheritance. Again, this is due to difference in the policy language expressions. (See [58] for more details.)

[18] Note that this is a similarity with the SEM [34] (Semi-Trusted Mediator) and IBE [35] (Identity-Based Encryption) approaches described in previous Section 2.3.1. Of course, trust management systems are much moe complex systems that provide access control, and not only authentication.

Trust management systems (Figure 3) offer a new attractive approach for managing authorisation in a distributed inter-organisational environment. However, they are more suited for client-server oriented architectures, where access of a large number of users to a few resources needs to be controlled by one or more authorities. They do not scale well in an environment such as peer-to-peer networks or collaborative communities, where every user may represent a potential resource, and vice versa.

2.4 Host Security

So far, the most of this chapter deals with the establishment of secure communication between the entities over a non-secure, distributed network. The other important aspect of information security is protection of the resources and entities from malicious intruders or non-authorised actions. For that purpose special devices and techniques are developed: Firewalls and Intrusion Detection Systems (IDS). They represent the means and the location of the actual enforcement of security policy.

A firewall is defined as a device (or collection of devices) placed between two networks, that has the following properties [59]:

1. All traffic from inside to outside (and vice-versa) must pass through the firewall.
2. Only authorised traffic (as defined by the local security policy) is allowed to pass.
3. The firewall itself is immune to penetration.

In a traditional corporate environment, where 'inside' is considered everything within corporate LAN (Local Area Network), the first requirement is relatively simple to satisfy, by placing the firewall on the edge of LAN, and assuring that it is the only access point (so-called 'choke-point') to the external, non-secure network (e.g. the Internet). The accomplishment of the third requirement takes the approach of installing as little software as needed for functioning of the firewall (which depends on the experience of security administrator), or preferably in using some of the proprietary devices, such as Cisco, CheckPoint, etc. In addition to reducing the possibility of misuse, this also assures the maximum firewall performance on the network [60].

Although there is a lot that can be said about the above, perhaps the second requirement is the most interesting, since that is the functionality that distinguishes different types of the firewall. Three main approaches in the firewall design exist [59]:

- **Packet-Filtering Gateway** works by inspecting (and discarding) packets based on their source or destination address, or application port number. The administrator makes a list of (un)acceptable machines and services, creating an enforcement policy that adds up to and overrides one of the following default policies [3]: that is either '*what is not explicitly permitted is prohibited*', or '*what is not explicitly prohibited is permitted*'. The filtering can

be done either for incoming packets, outgoing packets, or both. Discarded packets are not kept, and the decision is normally based only on the content of the current packet. An enhancement of packet filtering is *stateful-inspection* firewall [61]: besides looking at the individual packet contents, it also inspects the attributes of the multi-packet flows. However, this type of firewall requires more administrative setup (the connection table has to be built to track individual packet flows). In general, the advantage of a packet-filtering firewall is its simplicity and performance (they are normally very fast). The drawbacks are difficulty of setting up firewall rules correctly and lack of authentication.

- **Application-Level Gateway (proxy)** contains special purpose code for each desired application, effectively simulating the effects of the application [61]. When data is transferred from the source to the destination, a proxy gateway stands in the middle of the protocol exchange, establishing two connections: one between the source and the proxy and one between the proxy and the destination. Though the proxy seems to be transparent from the point of view of the communicating parties, it is capable of monitoring and filtering any specific type of data, such as protocol commands, before sending it to the destination. This type of firewall has the ability to perform the authentication function, and also to audit and log all the traffic. The main disadvantage is the additional processing overhead introduced. Also, only installed applications are supported, which makes it very expensive to implement and maintain.
- **Circuit Gateway** controls the flow of data at the TCP/UDP layer, and it can be implemented as a stand-alone system or as a special function of application-level gateway [3]. It does not permit end-to-end connections. It sets up two TCP connections, one between itself and inside user, and one between itself and outside user. Once the connections are established, TCP segments are copied without examining the content. The security function consists in examining which connections are allowed [3]. This type of firewall is typically used for outgoing connections, and only if inside users are considered trusted (measures like limiting the duration and frequency of usage, allowed addresses of outsiders, and even authentication of inside users before connection establishment takes place [59]). It does not have large performance overheads, and enables logging number of bytes and the TCP destinations.

Depending on the size of the organisation and protection needed, different firewall configurations are possible. Various types of firewalls can be combined to provide protection across multiple layers of a protocol stack; as well as to provide different levels of protection according to geographical regions, such as the use of a DMZ (de-militarised zone) which commonly offers access to a corporate LAN for semi-trusted users (e.g. dial-up remote employees). Although firewalls are normally designed as separate hardware machines, software applications known as *personal firewalls* (or desktop firewalls) are available since recently [61].

They are used to provide the protection to increasing number of home users. Similar to a normal firewall, they screen traffic according to pre-configured and user-configured rules, and can be combined with an anti-virus scanner (causing it to activate and screen inbound traffic the moment it reaches the target host).

Although firewalls provide strong and important protection to the organisational LAN, they have a number of potential drawbacks. The firewall 'choke-point' tends to create a performance bottleneck and congestion points in the network; in order to avoid it, inside users can setup dial-up connections, creating completely unprotected 'backdoor' entrance for an attacker. End-to-end encryption, often used to secure traffic between the hosts residing in different organisational sites, cannot be inspected by the firewall. Also, new applications use more executable fragments embedded inside the transmitted data (Word macros, JavaScript, etc. [9]) that can be easily used maliciously, not only against the recipient but also against all the entities within the firewall perimeters [19]. Finally, firewalls do not protect from the malicious insiders, which is a significant concern to organisations. [20]

Firewall systems are used as a preventive security measure. In order to detect the attacks that pass the firewall (or are initiated by inside users), Intrusion Detection Systems (IDS) are deployed within the secured part of the network/system, as the second line of defence. IDS can be defined as a security system that monitors and analyses computer system events and network traffic for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorised manner [11],[61].

Functionally, there are two types of IDS [3],[61]: *misuse detection* (mainly used in current industrial products), and *anomaly detection* (subject to significant research). Misuse detection (also known as signature-based) works by matching the current occurrences in the system against the defined attack patterns (so-called attack signatures, which capture the patterns of known attacks). The problem with this type of IDS is a large number of *false negatives* (the actual malicious action is not detected), since a minor modification in the way attack is carried out creates different pattern. Anomaly detection (also, heuristic-based) correlates the current behaviour patterns to already defined profiles of legitimate behaviour (normally defined by the statistical analysis of the users' behaviour in the system, or recently from state-based models of the protocol specifications [64]). The problem with this approach is a large number of *false positives* (legitimate action that is classified as a malicious), since any deviation from the adopted pattern of users' behaviour is flagged as an intrusion. Learning mechanisms and

[19] [62] points some of the firewall-related concerns, perhaps in unconventional, but in a rather straightforward and consise way.

[20] For example, a recent survey on security breaches in corporate e-business environments has demonstrated that 35% of respondents did not know if an attack came from inside or outside of the company network [1]. Also, [63] reports that equal number of attacks originate from outside as well as from inside the organisations.

updates on the behavioural patterns are normally used to try to minimize this. For best results, intrusion detection systems combine both approaches.

These types of IDS can be implemented as: passive-listening network hardware devices (*network-based IDS*), dedicated software that monitors events on a single machine (*host-based IDS*), or centrally controlled collection of monitoring sensors (either software or hardware) placed at various point inside the organisational LAN (*distributed IDS*) [65]. Similar to firewalls, the structure of the protected system and the level of security required can dictate any suitable combination of the above. Since they are expensive to deploy and maintain, and require human administrator to monitor the alarms, IDS are normally placed to monitor high-security servers of the inner LAN, or the exposed side of the firewall.

Recent research efforts are trying to develop more distributed IDS, following the paradigm of peer-to-peer networking [66], or hierarchical semi-centralised distributed systems [67]. Indra [66] proposes associating pluggable daemons with peer hosts, which should provide monitoring and reporting of the network activity. The challenges yet to be addressed are: the process of evaluation of the gathered information, assessment of the trustworthiness of the entities sharing the data, and the details of the communication protocol for collection and distribution of logged data and alarms. EU project SAFEGUARD [67] develops an agent-based architecture for increasing the survivability of the management networks for large complex critical infrastructures. It proposes various types of agents for distributed monitoring and reasoning on different levels of the system hierarchy, aiming to provide an automated response and self-healing of the system. The author has been collaborating with the SAFEGUARD participants from within the Electronic Engineering Department at Queen Mary, University of London. The initial screening **[BIG]** has indicated that their approach may suitably complement for the novel distributed architecture that is described in Chapter 4.

2.4.1 Distributed Firewalls

The approach of distributed firewalls, proposed by Bellovin [68], aims to address the protection of organisational environment when users are distributed across the Internet. In such an environment, usage of traditional firewall does not offer protection, since ‘inside’ users are not concentrated within the geographical boundaries covered by the firewall. With a distributed firewall approach, security policy is still centrally defined by the security administrator, who is not necessarily topologically the ‘local’ administrator. Enforcement, however, takes place on each endpoint, by each individual host that participates in a distributed firewall (Figure 4). Authentication between the administrator and the users needs to be provided (the original proposal suggests using some form of certificates), and all the communication can be (optionally) encrypted. There are several benefits of such a distributed approach [68]:

- Security does not depend anymore on restricting the network topology. The security perimeter can easily be extended to safely include remote hosts and networks (e.g., telecommuters, extranets).
- If a single user were compromised, that would not create vulnerability for the whole system.
- End-to-end encryption is made possible without sacrificing security, since the distributed firewall is effectively the end point.
- Policy rules are distributed on as-needed basis: the end hosts maintain and enforce only the policy that is relevant for their communication. This does not require each host/firewall to maintain the complete set of policies, which may be very large for large networks.
- It eliminates a single firewall 'chokepoint'. This is a benefit both from the performance and availability point: throughput is no longer limited by the speed of the firewall, and there is no a single point of failure that can isolate an entire network.
- Also, it can be deployed behind the traditional firewall, providing a second layer of defence.

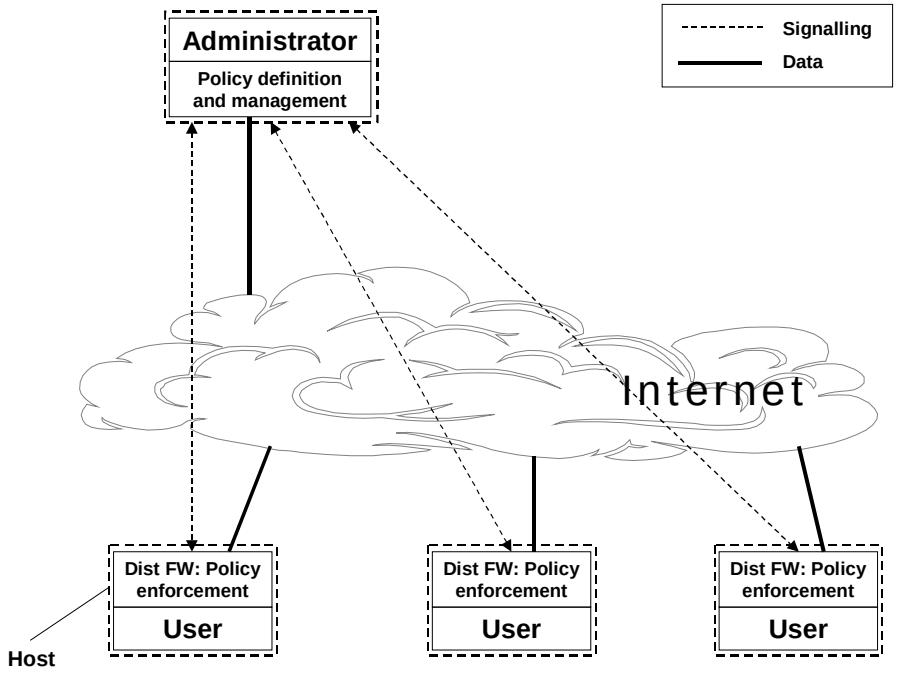


Figure 4: Distributed Firewall Paradigm

A suitable mechanism for defining and distributing firewall policy, and the end-host enforcement mechanism are two important issues that need to be considered in order to allow practical use of distributed firewalls.

Further work of Bellovin and his colleagues [69],[70], as well as an independent approach (so-called *micro-firewalls*) proposed in [71], focus on the implementation of the security enforcement mechanism at each distributed firewall instance. The approach taken is to modify

the kernel of host's operating system in order to accept dynamically defined rules for access control. These proposals inherit the KeyNote approach in policy definition, allowing self-signed certificates and privilege delegation. Although details of enforcement mechanisms are provided, the schemes do not address communication mechanism further than pointing a need for authenticated exchange of credentials. As already discussed in the previous section, the loose arrangement of the KeyNote approach removes nearly all hierarchy and makes it difficult to deploy a more structured role-based approach, which may be needed to support a corporate-oriented way of working and inter-organisational collaborations.

Work on micro-firewalls [71] and related research known under the same name [72], suggest integration of a host-based intrusion detection mechanism with a micro-firewall implemented in the Linux kernel. The main motivation is to provide a monitoring mechanism at the same point (and time) with the firewall-based security enforcement. The idea is to provide feedback to the collection points, where such data can be analysed and a corresponding response action taken. The scheme suggests use of agents for collecting the intrusion reports and distributing the updated policy, but without detailed elaboration on how collected data is correlated and used for deriving suitable policy updates. The approach is very similar to the SAFEGUARD architecture mentioned in the previous section [67], with SAFEGUARD focusing more on information correlation and decision making, instead of middleware based monitoring.

The architecture summarised in [73] takes different approach in exploiting the distributed firewall concept. It proposes an *autonomic distributed firewall* implemented as a network interface card. Such a device, placed at the endpoint (between the host machine and the network connection) is controlled from the centralised location by a common protocol. The card stores cryptographic keys, packet filtering rules, and a list of group members (for supporting encrypted group communication). Policy is created and/or updated manually by the administrator behind the centralised policy server, and distributed to the users [74]. The proposal indicates that communication between different servers is supported, but authentication mechanisms or protocol details are not specified. The communication between users, who are members of the same group, is encrypted with the symmetric key provided by the server. In general, authentication for both user-server and user-user communication is based on the encryption of the request with the symmetric key embedded in the hardware interface. This provides limited sender authentication (any traffic encrypted with the appropriate key is accepted), and relies on the strength of the encryption key and physical protection achieved with the hardware. The main argument for the advantages of such hardware-based approach is separate memory and processing power for the traffic monitoring, as well as tamper-proof hardware which provides

protection from the malicious user[21] or executable code from the application level. This also introduces the main limitation of the scheme. Since it operates on a network level, functionality of the network interface card is limited to a packet filtering; packets are either allowed or denied based on the specific protocol, port number or IP address. However, once access is permitted, it is out of control of the network interface card, and the sender's actions still may need to be authorised by some other means (e.g. role-based access control).

An OS kernel-based firewall implementation is advantageous compared to packet filtering mechanism, since it enables more thorough traffic inspection, and access control based on presented credentials. Alternative approaches to achieve controlled software execution have been reported, such as profiling of nominal software execution as described in [75], or based on the system call policies as proposed in [76]. These approaches, although they have not been applied for security of distributed environment, demonstrate the availability of a number of the dynamically definable/modifiable mechanisms for controlling the software application.

2.5 Summary

This chapter has reviewed approaches for providing information security relevant to this research. Traditional mechanisms for weak authentication and firewall-protection of sites fail to provide good security for the emerging applications, both in terms of flexibility and more fine-grained controlled access to the resources.

New approaches, such as digital certificates for authentication and authorisation, and role-based access control are therefore being developed and improved in order to accommodate more dynamic systems. Mechanisms such as distributed firewalls and distributed intrusion detection are being developed for protection of distributed entities, and a suitable communication mechanism is needed to provide distribution of the security policy.

The next chapter reviews the main paradigms for supporting distributed communication, and the state-of-the-art with respect to the security features deployed.

[21] However, a malicious user can always remove such a device.

Chapter 3 Functionalities and Security of Distributed Collaborative Systems

The implementation of groupware has come a long way since the original reliance on email delivered via UUCP [22]. Motivated by commercial applications, further research in the area of network protocols and security mechanisms has led to the development of Virtual Private Networks [77] that offer a consistent security model but are rather inflexible. Secure multicast [90] and peer-to-peer networking [104] were introduced to meet the requirements for more scalable and flexible group-oriented working environments [23]. Finally, increased power of desktop computers and the rising trend for pervasive computing is witnessing the emergence of frameworks such as Grid computing [122] and Web Services, aiming to facilitate comprehensive services in a secure, yet ubiquitous environment [123].

All of the above approaches will be discussed in subsequent sections, with particular focus on the security mechanisms as one of the biggest challenges these architectures are facing.

3.1 Virtual Private Networks

One of the first solutions to offer a secure communication network for distributed corporate sites was Virtual Private Network (VPN). VPN can be defined as a communications environment in which access is permitted only within a defined community of interest [77]. It is a logical network that uses underlying network infrastructure to connect the 'community of interest'. This means that VPN does not have its own physical infrastructure.

There are two main motivations for building VPNs [77]. Using public infrastructure and ISP services gives much more cost-effective and flexible communications infrastructure compared to the dedicated leased lines. Another motivation lies in the privacy of communication, where the characteristics and integrity of communication services within one closed environment are isolated from all other environments that share the common underlying infrastructure.

Three main architectural types of VPN deployment can be distinguished [78]:

- *Intranet (site-to-site or LAN-to-LAN) VPNs* allow private networks to be extended across the Internet or other public network service in a secure way, facilitating secure communications between internal departments and branch offices of a single company. In Intranet VPNs the primary technology requirements are strong data encryption to protect

[22] An acronym for UNIX-to-UNIX Copy Program, UUCP is a protocol for transferring files, news, and mail, and executing remote commands between machines.

[23] Mobile ad-hoc networks share the similar aims, but having somewhat different requirements imposed by its specific setup. That area has not been surveyed since the requirements for the node mobility goes beyond the scope of this work.

sensitive information; reliability to ensure the prioritisation of critical applications, database management, and document exchange; and scalable management to accommodate the rapidly growing number of new users, new offices and new applications.

- *Remote access VPNs* support mobile and telecommuting employees with dial-up connection (or increasingly, permanent broadband line) to connect to the organisation's intranet across the Internet or other public network service in a secure way. One of the main requirements is strong authentication in order to verify remote and mobile users' identities in an accurate and efficient manner. On the management side, remote access VPNs require centralized management and a high degree of scalability to handle the vast number of users accessing the VPN.
- *Extranet VPNs* allow secure connections with business partners, suppliers and customers for the purpose of e-business. It is effectively a combination of previous two deployment types. Extranet VPNs require an open, standards-based solution to ensure interoperability with the various solutions that the business partners might implement. Equally important is traffic control to eliminate bottlenecks at network access points and guarantee quick delivery and response times for critical data.

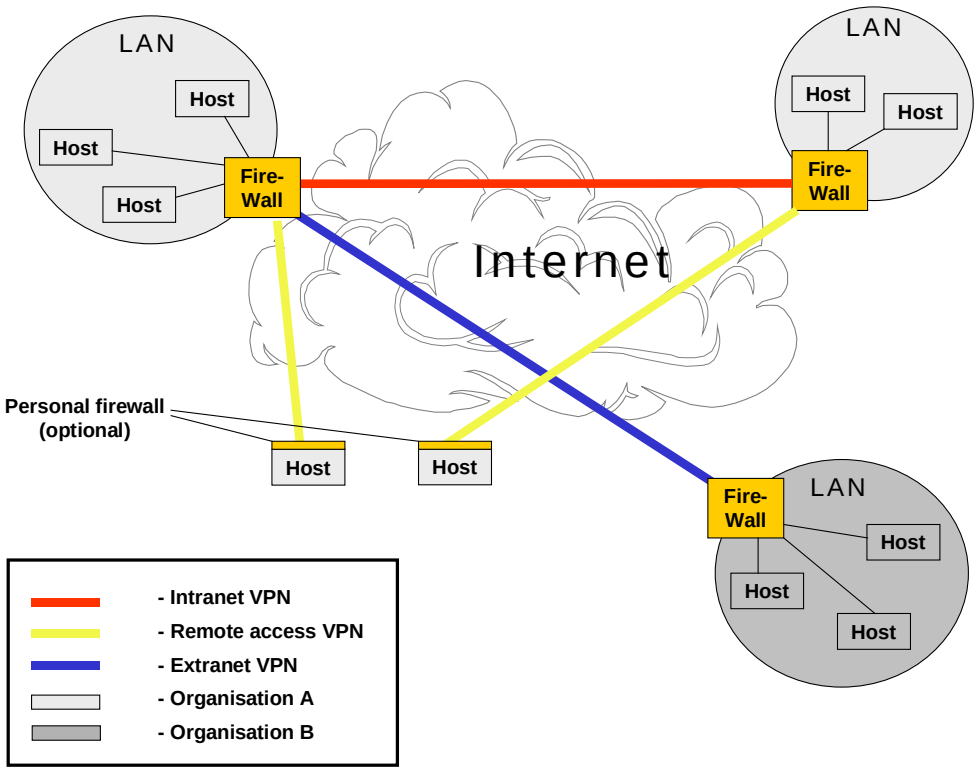


Figure 5: Different Deployment Types of Virtual Private Networks

Intranet and Extranet VPNs are normally established as long-term infrastructures, where the basic formation changes mainly to include new LANs (or to exclude the existing ones). On the other hand, remote access VPNs are much more dynamic, where a connection between remote

user and central site is established on as-needed basis (normally via VPN Gateway, responsible for client's authentication).

One of the most common methods for implementing authentication services for remote VPN users is RADIUS server (Remote Authentication Dial In User Service) [80]. Within a single LAN, it consists of one secure central location for storing all the information about users, their passwords and access privileges (RADIUS Server) and several remote access servers (known as NAS – Network Access Server) that remote user connects to. For protection against eavesdropping, the NAS, acting as the RADIUS client, encrypts the user's authentication password before it sends it to the authentication server. If the primary security server cannot be reached, the security client or NAS device can route the request to an alternate server. When an authentication request is received, the authentication server validates the request and then decrypts the data packet to access the user name and password information. If the user name and password are correct, the server sends an Authentication Acknowledgment packet, and an authentication key (or signature), identifying itself to the security client. Once the NAS receives this information, it enables the necessary configuration to allow the user the access rights to network services and resources. If at any point in this login process all necessary authentication conditions are not met, the security database server sends an authentication reject message to the NAS device and the user is denied access to the network. The RADIUS server itself can act as a client (effectively as a proxy) at another RADIUS server (or other kind of authentication server) in order to retrieve authentication data of a user from another administrative domain. This permits two or more administrative entities to allow each other's users to dial in to either entity's network for service. Experience has shown that RADIUS servers can suffer degraded performance and data loss when used in large-scale systems, partly since they do not include provision for congestion control [81].

A typical approach for facilitating extranet VPNs, enriched with the usage of public-key certificates for authentication (instead of Kerberos system, for example) is described in a recent publication [82]. It introduces a trusted third party that acts as a certification authority. For the organisations to participate in extranet VPNs, their administrators first need to register and obtain a public key certificate from a trusted third party (TTP). Also, the administrators specify the policies for inter-organisational communication and forward them to the TTP. The TTP, after receiving policies from different organisations, relates the relevant ones. For example, if organisation A wants to communicate with B and C it defines its policy constraints towards them; in a similar way, B and C define their policy towards A. The TTP then creates combined A/B and A/C policy, both of which are sent to A, but only the corresponding ones to B and C. Once compiled, the policy is signed by TTP's key and sent back to the security server of the organisation, which is effectively a firewall of the Intranet. If Alice (from organisation A) wants

to communicate with Bob (who belongs to B), a connection is established in two steps: Alice first authenticates herself to her local security server, and then the server from organisation A performs mutual authentication with a server at organisation B, which then contacts Bob. Once the link is established, all the communication between Alice and Bob goes via both security servers A and B, which effectively act as firewall proxies. The added value is that relationships are previously agreed (through the TTP), which increases trustworthiness of the entities. The pitfall of such a scheme is the separation of the relationships between different organisations: Alice is accountable only to firewall A, and all that firewall B considers is the policy relationship towards firewall A. Eventually, all the privileges are enforced properly (Alice's by server A, and organisation A's by server B), but the artificial layer created by proxy servers unnecessarily burdens the communication. Ideally, Alice and Bob should be able to communicate directly; however, this problem is not solved by VPNs, but with some other architectures as it will be described in the following sections.

The scenarios where a remote user attempts to create an extranet VPN relationship are avoided in practice due to complex policy requirements (typically, a user would need to use dial-in VPN service to its Intranet, and then to connect to the external organisation via already defined extranet VPN links).

From the technological perspective, VPN deployment takes a number of the approaches in order to accommodate different network protocols and service requirements. As noted in [79], predecessors of VPNs appeared in the 1970s, in the form of privately operated network devices connected over a carrier's dial-up or dedicated leased lines. Since the emergence of VPNs in 1990s a range of existing as well as purpose-built protocols have been used for VPN deployment [77]. One of the common ways for creating a VPN is *tunneling*. A tunneling protocol encapsulates the data packet in another packet with the additional header that provides routing information to enable the encapsulated payload to securely cross the network. The entire process of encapsulation and transmission of packets is called tunneling, and the logical connection through which the packets travel is known as a *tunnel*. In general, VPN implementation method depends to the large extent on the underlying transport mechanism. Different literature sources give somewhat different VPN taxonomy [77],[78],[79],[87]. However, they agree that the VPN architecture can be generally divided into three main types, depending on the level of protocol stack where they are implemented:

Link Layer VPNs share a common switched public network infrastructure (such as Frame Relay or ATM networks), while the VPNs have no visibility of each other. This is commonly achieved by creation of the 'virtual circuit' between the sender and the recipient, available only for the duration of the connection. The main characteristics of this approach are their flexibility and cost-effectiveness, but also scaling limitations and complexity of configuration

management. Also, data confidentiality is very difficult to achieve, especially in heterogeneous environments.

Network Layer VPNs are usually based on the Internet Protocol (IP). Using IPSec (IP Security Protocol, [83]) tunnel mode, the original IP packet is augmented with the additional fields (AH - authentication header and/or ESP - encapsulation security payload) for authentication and (optional) encryption, and a new IP header is added. The new IP header is used to route the IP packet through the Internet. The receiver removes and discards the IP header, processes and removes the additional security headers, and processes the original IP packet in the usual way. The advantage of this approach is that full separation and high security of the communication can be achieved. A disadvantage is processing overhead introduced by the encryption/authentication, as well as the vulnerability of the tunnel at the end points (normally the firewall or VPN gateway), where extra headers are stripped off and packet is visible in its original form.

Transport and Application Layer VPNs are mostly based on the use of encryption, aiming to provide privacy and data integrity between two communicating applications. The commonly used TLS/SSL (Transport Layer Security / Secure Socket Layer) protocol effectively consists of two protocols [84]; handshaking is used to negotiate the session (normally by some means of authentication, e.g. certificates), and record protocol is used for transmission of data encrypted with symmetric key. The protocol provides full confidentiality and message integrity. However, it requires modifications to the application programs and support of additional infrastructure such as certificates.

In general, VPN deployment has two conflicting requirements: flexibility and security. Preserving flexibility of communication (such as mobility of users) requires easy VPN implementation, but also more careful security considerations. On the other hand, mechanisms that allow simple and easier deployment (at the lower level of protocol stack), set more challenging security requirements [87].

The main benefit of using VPNs is that a Wide Area Network (WAN) environment can be constructed at lower cost, as dedicated and long distance lines are replaced with local connections to the network/service providers and shared long distance connections. Also, virtual private networks, particularly Internet-based, allow greater flexibility when deploying nomadic computing, telecommuting and branch office networking. On the other hand, there are several weaknesses of VPNs, most of them related to the scalability and security [85]:

- VPN Gateways [24], centralized points of VPN traffic, represent the single points of failure, potential traffic congestion points, and are at the same time an obstacle to the dynamics of user connectivity and their direct communication.
- Higher levels of security (such as strong authentication and data encryption), introduced to ensure integrity and confidentiality of data impact on the performance both on the user and server side in increased protocol header overhead and authentication latency.
- Finally, although VPNs offer protection to the network communication, protection of the users at corporate sites has to be achieved through other means, such as firewall. Bearing in mind that protocols encapsulated by VPN tunnels can effectively bypass the network security policy enforced at the local firewall, this raises other security issues [86].

There are efforts attempting to enhance the existing concepts of VPNs, especially from the security viewpoint. Perhaps the most significant is the IETF's attempt to unify the underlying protocol by using IP and its embedded IPSec mechanisms (i.e. for IPv6) [88],[83]. However, the problem of a scaleable authentication mechanism and security of the communication once it has left encrypted VPN tunnel, still remains.

3.2 Secure Multicast for Group Communication

Multicasting is a technique that enables a single packet transmission to be sent to one or more destinations or to a group. Typically, group membership is dynamic (i.e. members can join or leave at any time). In such an environment, a secure multicast protocol must provide group membership control, secure key distribution, and secure data transfer [89]. Typically, this is achieved by distributing the group key only to the participants in a multicast session (control of the group membership), and using that key for encryption of the multicast traffic (secrecy of exchanged data). In addition, in a dynamic environment, where members can freely join and leave, new members need to be prevented from acquiring the data sent before they have joined (so-called backward secrecy) or after they have left (so-called forward secrecy) [90].

The above requirements are normally addressed by refreshing the group key (also known as *rekeying*) whenever a member has left the group, or even when a new member joins. The main issues related to the key distribution are that it needs to be done in a secure way (i.e. the key is disclosed only to the group members), and that the mechanism is scaleable (ideally, it should not depend on the size of the group – known as the '1 affects N' property [89]). In this respect, a number of different schemes for key management and distribution have been proposed. A comprehensive overview of different techniques is given in [90]. According to [90], the approaches for group key management in multicast can be divided in three main classes:

[24] VPN Gateways are normally implemented as a part of firewall infrastructure at the edge of organisation's Intranet.

- *Centralised*, where a single entity is in charge of the group control. Ultimately one entity is responsible for generating a group key and maintaining the group membership, which makes them relatively easy to implement but puts a significant burden on the single group controller (also known as the *Key Distribution Centre* - KDC). However, there are number of methods for communicating the new key to the group members, aiming to achieve better scalability. Examples of this approach are the simple pair-wise scheme (where KDC shares different key with each of the members, and uses these keys to distribute the group key), or mechanisms based on a hierarchical tree (where each group member possess a number of keys, based on its path in binary (or n-ary) tree to the root) [91].
- *Decentralised*, where a large single group is divided into a number of subgroups, and each of them is managed independently via dedicated subgroup controller. A typical example is the Iolus framework [92]. In Iolus, a large group effectively consists of a number of subgroups, where each subgroup is controlled by independent Group Security Agent (GSA), and these are managed by a top-level Group Security Controller (GSC). For a subgroup it manages, the GSA maintains subgroup key and pairwise keys with each member. For its parent group, the GSA maintains a subgroup key and a pairwise key with the GSC. Therefore, GSAs are in charge of 'translating' data from/to their subgroups (by re-crypting the data). Similar approaches have been proposed (see [90] for overview) to address the issues of timely group key distribution (either through synchronised or periodic rekeying, or by introducing a central entity which is in contact with subgroup controllers), or for removing trusted third parties (i.e. preventing subgroup controllers to obtain the group key)[25]. In decentralised schemes, the problem of key management is shifted to smaller groups by delegating it to a number of KDCs. Therefore, this approach tends to be more scaleable, but normally requires synchronised key distribution among sub-groups and typically makes data transfer between the members of different sub-groups more complex.
- *Distributed*, where group members perform key generation themselves: either by all of them contributing to the key, or that being done by one of the members. The scalability problem of this type of key agreement lies either in the number of message transactions needed in order to derive the group key, or in the computation needed to be performed by each of the group members or a group leader [96]. However, this approach may be preferable for ad-hoc communities, as it does not require an explicit group manager (i.e. KDC).

Regardless on the type of multicast groups (with respect to the group key management scheme), there are two important issues related to the group security: authentication and access control.

Commonly, authentication provided with a key management is recognised as a group authentication, meaning that participating entities can authenticate each other as a group

[25] Please note that terminology may differ in various approaches. However, concepts are fundamentally the same.

member, but not necessarily on the level of an individual user (so-called, sender authentication [93]). Group authentication is naturally provided by encrypting multicast data with a group key which is (by definition) known only to the valid group members. However, authentication of each entity (if needed) is far more complex. In [94], usage of authentication tokens is proposed, but still not on the level of a single entity and it does not address nomadic users [26]. The approach described in [95] suggests introducing a third party to issue public-key certificates for group members, for certifying a group, a member's identity and a validity period. In [96], a mechanism based on Diffie-Hellman agreement (see [6]) is proposed, which (being developed for distributed ad-hoc communities) avoids centralised authority but introduces significant communication overhead.

The second issue, of access control, is normally concerned with admission to a group, i.e. whether or not an entity is a valid member of a group [94]. This is provided through timely update and distribution of a group key. However, this does not provide any means of constraining actions of an entity once it is allocated to a group, or for defining more fine-grained group policy other than inclusion/exclusion. Work reported in [97],[98] tackle this problem to some extent, by proposing a policy negotiation during the group establishment (typically, policy is proposed by group controller/initiator). However, it mainly focuses on prescribing policies such as: type of encryption to be used, mechanism for key distribution, key lifetime, etc., which may also be a determining factors on whether an entity should be allowed into a group.

Recent publications by IETF address the above issues of authentication and access control. In [99], use of asymmetric cryptography for signing multicast traffic at the level of IPSec is suggested. Security architecture framework described in [100], addresses decentralised multicast groups. It introduces the additional entity called 'policy server', as the highest entity in multicast group hierarchy. Policy servers (of which there can be several) maintain multicast security policy and communicate it to group controllers. This information is used by group controllers to make a decision on allowing an entity to enter the group, which may also take some form of authorisation. However, [100] points out the difficulty on agreeing the policy rules if group members reside in different administration domains, and is less specific in this regard.

3.3 Peer-to-Peer Networks

The initial development of the Internet was based on the peer-to-peer principle. Nodes connected to ARPANET [27] were mainly academic institutions with equal rights, and which did not need protection from each other. The widespread increase in network connectivity in 1990s

[26] Eligibility of participating in a group is determined based on the IP subnet addresses, which is included in the authentication token.
[27] Acronym for Advanced Research Projects Agency Network (ARPANET), developed by the US Department of Defense.

has seen the introduction of a client-server model in order to accommodate a large number of users with slow network connections and low processing power of personal computers. Typically, in a Client-Server system a dominant computer (server) offers services to a number of other computers with less control (clients). Clients can communicate with other clients only through the Server [28]. The Peer-to-Peer (P2P) approach is essentially different from this model.

The re-emergence of P2P networks (initially with Napster [101] in 1999) has resulted in a number of approaches being developed to support some of the features recognised as a peer-to-peer [102]. For that reason, the definition of P2P is still not uniform. Peer-to-peer can be defined as a distributed network architecture where participants share part of their own hardware resources, in order to jointly provide the services and the content offered by the network. The participants have a direct access to each other, without passing other nodes. Each peer (node) in the network is at the same time a resource provider, as well as resource requestor – therefore the term *servent* is often used to describe the nodes in P2P network (i.e. each node is both ***serv***er and cli***ent*** at the same time) [103],[104].
However, it is very difficult to make an architecture that fulfils all of the above requirements and that is practical at the same time. In general, two main types of P2P architectures can be recognized among a range of different solutions:

- Pure peer-to-peer: all peers act like servents; removing of any arbitrarily chosen peer would not cause any loss or disruption of network service.
- Hybrid peer-to-peer: a central entity is necessary to provide part of the network service (normally routing information, i.e. directory services).

Since most of the P2P networks are developed targeting a specific service, their architectures differ, and sometimes is difficult to distinguish to which type they belong. Most commonly, the nodes require special software in order to engage in the network.

Napster [101], developed for sharing of mp3 files, is a hybrid P2P architecture: clients request the name of the file from the server (see Figure 6a). The server (which maintains the directory of registered clients and the files they share) replies to the client the matching results, which includes the IP address and name of the corresponding host, file size, bit rate, etc. The next step is the actual file transfer between the clients, which occurs directly without server's intervention. Seti@Home [105] is another example of hybrid P2P architecture. It is a project at University of California at Berkeley, to support processing of signals from space captured with

[28] Virtual Private Networks and current Business-to-Consumer e-Commerce solutions are examples of such a model. On the other hand, the SNMP protocol uses client-server architecture in the opposite way, with one client and a number of servers in a normal operational mode; since clients act as 'administrators', there may by a functionality to support direct client-client communicate with each other. However, direct interactions between servers are not possible.

the radio telescope [29]. The approach taken is to distribute parts of the captured signal to the home/office computers for the analysis. The architecture is similar to that of Napster, with the difference that clients do not interact directly – it is recognised as a P2P network since the clients share part of their own resources.

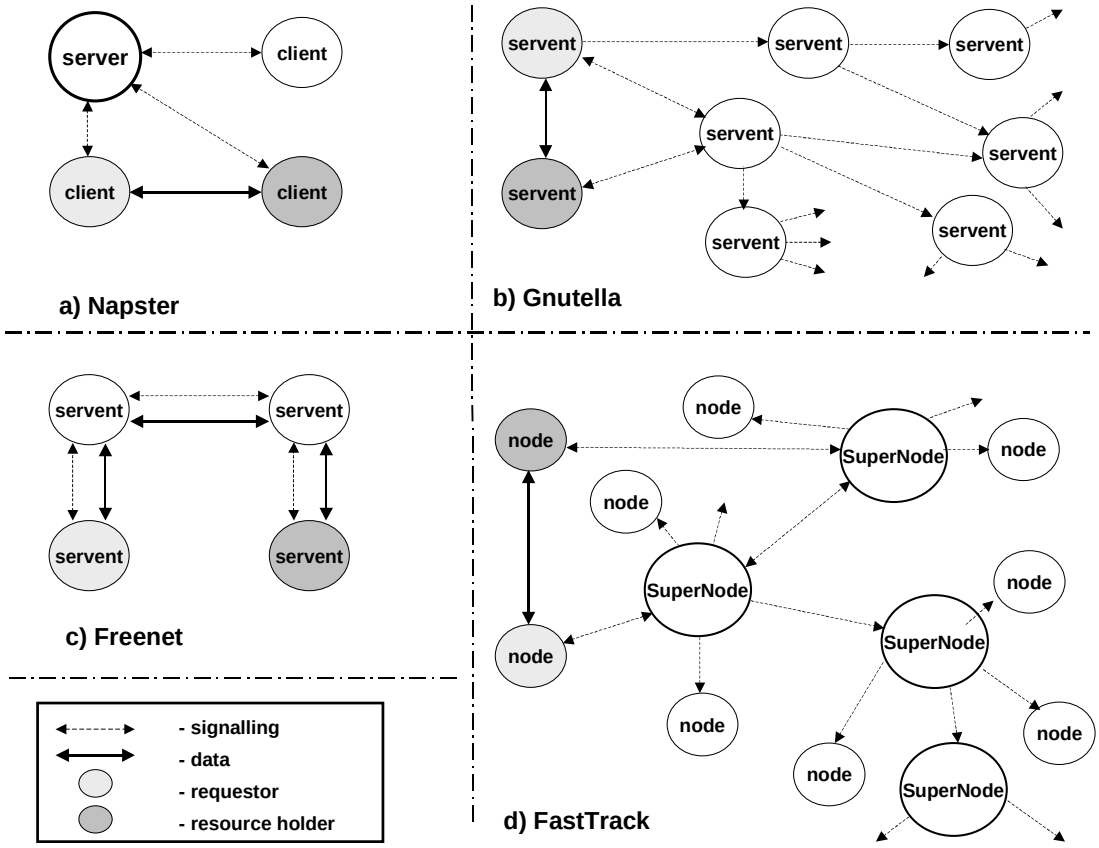


Figure 6: Basic Peer-to-Peer Architectures

Gnutella [106] is an example of pure P2P architecture, developed to overcome legal obstacles that caused Napster's termination (see Figure 6b). The file search is initiated by a peer's broadcast request to its neighbours, which forward it further until the file is found or the packet expires (based on pre-defined time-to-live and maximum number of hops). The information about the peer hosting the file travels the same way, whereas the actual data transfer occurs directly through creation of a new session. Freenet [107], another pure P2P architecture, takes somewhat different approach. It uses point-to-point protocol for propagation of the initial request, until the file is found or the request expires. The file is downloaded through the same path, being also duplicated and stored at the each of intermediate nodes (see Figure 6c). Each peer keeps the list with names of the files and the corresponding peers from which it came. Also, for the most popular files (i.e. most frequently requested), the actual data (file content) is

[29] SETI stands for Search for ExtraTerrestrial Intelligence. As a curiosity, the processing time of Seti@Home accumulated since 1999 corresponds to nearly 2 million CPU years.

kept as well. In brief, files are searched according to the unique identifier they are described with (created by applying a hash to the content or name of the file). If a peer cannot answer the request, the request is forwarded to the peer from its list that has the closest identifier to the requested one. It has been demonstrated that this approach operates in a small-world fashion, with less than 10 hops [104]. The main motivation for the development of Freenet is to provide the anonymity: since requests are forwarded only to the neighbours, a requestor can only intuitively guess the distance from the original source of file (according to the number of hops), but a direct contact is never made.

One of the significant observations made from the analysis of Gnutella and Freenet is clustering in a small portion of nodes with high Internet connection that share a large number of files, and the majority of the nodes which (effectively behaving like clients in a client-server model), only request and download the data [108]. FastTrack [109] takes this to its advantage, and defines two types of servents in its network: SuperNodes and Nodes, which is decided by the software based on the host's Internet connection. Typically, each Node in the network is assigned one SuperNode, which it sends the requests to. SuperNodes exchange requests among themselves according to the Gnutella protocol. Once the file is located, the requestor establishes direct connection for the download (see Figure 6d). The approach in having two layers of peers assures more scaleable and reliable routing than in Gnutella; therefore, this P2P model can be considered as 'mixed' between pure and hybrid P2P. Jabber [110] operates similarly to FastTrack, only the distinction between the servers and clients is clearly made. However, clients are able (and encouraged) to communicate directly if they have any prior knowledge of each other. Jabber is primarily created to provide a flexible instant messaging system.

Flexibility and straightforward use of the P2P approach is one of the main reasons for its growing popularity among the users. However, most of the systems surveyed above were developed in an ad-hoc way, with a lot of software bugs, and to a large extent neglecting issues of security and scalability. A great deal of work in the P2P area is dedicated to more scalable and reliable methods for routing through the network (e.g. Kademlia [111], Tapestry [112]). On the other hand, most P2P architectures carry considerable security concerns, such as the ability to penetrate firewalls, or the potential to be misused and expose the host machine to uncontrolled sharing of the stored data. Although there is no common uniform security solution (since there is no common P2P architecture and operation), there is significant ongoing research in the field. Some of the current solutions are being improved to include mechanisms for consistency-checking of shared files in order to prevent the spread of corrupted data (e.g. an updated version of Freenet), or for blockage of non-cooperative peers which do not share but only download the material, so-called 'free-riders' (as for example in Kazaa, the application that runs over FastTrack network).

A significant number of P2P architectures are concerned with a collaborative group-oriented working [113]. Some of the architectures are being revised or developed in order to incorporate mechanisms for data encryption, peer authentication, and access control, that are essential in this context. This is normally achieved through a hybrid P2P, where a dedicated member (or centralised node) performs group management. Groove is a real-time collaborative editing application that enables the decentralised ad-hoc formation of groups [114]. Its architecture is very similar to Rhubarb [115]. User can join a group if invited by an existing member – chair/coordinator (it is not specified whether this requires a consensus of the group). This includes mutual authentication through some form of certificate. After the initial authentication, new member is given (by an inviter) a group key, used for encrypting all group communication. Whenever a member leaves a group, a new key is created and securely broadcasted by the group coordinator to all the members. Rhubarb proposes an election mechanism of a new coordinator if the existing one leaves or goes offline. For that purpose, all nodes in the group must maintain the 'state of the group' – a list of members and their corresponding certificates. With Groove all of a group's documents, messages and applications (in addition to a list of members and their corresponding certificates) are stored and replicated across user machines so that all of a group's members can access the materials online or offline. Somewhat different, [116] proposes one or more dedicated authorities to maintain a single group, in order to address scalability and reliability issues. Only the authority can introduce a new member into a group, through mutual authentication. In order to support this with multiple group authorities, the scheme allows all the authorities to share the same public/private key pair (used for signing the authentication message); this is a possible pitfall of the system, since the possibility of a private key being compromised increases. Members' communication is preceded with the authentication (based on the 'group membership' received from the authority); the protocol explicitly supports only unicast connections, although it might be possible to extend this in order to support multicast sessions among the group members. More of the concern is that there is no procedure specified for a member leaving a group.

All of the above schemes emphasise firewall transparency as one of the key functionalities, which may be good for their users, but not necessarily for the security of a particular intranet. Although they distinguish the peers with different functionalities (by introducing some form of privileged member), they are not concerned with multiple degrees of authorisation. In [117], the issue of members with a different degree of privileges within a group has been addressed. The approach allows a number of group members to have the ability to store and delegate the appropriate privileges to the rest of the group. For most of the intended actions, a peer must apply for the authorisation rights, which are granted by the higher-level peers by means of short-lived certificates. If a member is removed from the group, higher-level peers are informed

and any subsequent member's certificate-related requests are denied. The mechanism assumes pre-existence of the group, so it does not comment on how certificate-granting entities are chosen and how policy is initially delivered. In general, it is not clear whether the motivation for having several certificate-granting entities is redundancy for reliability or splitting of authorities for security.

As a response to the current status of P2P, there is an effort of large companies to provide a suitable methodology for more structured development of P2P applications. Perhaps the most ambitious is JXTA Project, a Sun Microsystems' initiative (http://www.jxta.org) [30]. JXTA aims to develop a common network-programming platform for enabling P2P networking. It has several main objectives: interoperability (across different P2P systems and communities), platform independence (it does not built upon specific programming language or network protocol), and ubiquity (available for any device with a "digital heartbeat") [118]. It comprises multi-layered software architecture and a number of communication protocols, aimed at addressing generic requirements of distributed systems/applications. For example, it defines a peer as an entity that supports some of (but not necessarily all) JXTA protocols; as such, a peer can be a "processor, process, machine, or a user". From a security perspective, the framework aims to develop an architecture that provides: authentication, access control, audit, encryption, and non-repudiation. It does not aim to provide recommendations for a specific security policy approach or encryption mechanism. However, TLS protocol (Transport Level Security) for secure communication (independent of JXTA protocols), and X.509 v3 certification framework for authentication and authorisation are their choices for a generic framework [119]. Also, in the spirit of peer-to-peer networking, JXTA considers mechanisms for bypassing firewalls. However, [118] points that firewall penetration is not an ideal solution, and that this is an area of active research.

3.3.1 Current Status in Peer-to-Peer Security

Current state-of-the-art in P2P systems suggests that the issue of security has still not been fully addressed, which is one of the main obstacles for their more formal use. The main concerns pointed in the recent publication [120] are listed here:

- Firewall penetration. When a machine hosting a resource resides behind a firewall, HTTP traffic (used for many P2P applications) may be blocked at the firewall for incoming traffic, but it is normally not for outgoing connections. Instead of direct downloading, a requestor can send a message to the host behind the firewall instructing it to make an outbound connection and upload the file.
- Spreading of malicious software. Since a file is mainly searched according to its name, users can download and execute a virus, worm, etc. disguised with the name of a searched

[30] JXTA stands for "Juxtapose, as in side by side".

document. In the case of a Trojan Horse, once downloaded by a client, it can use a P2P connection for receiving remote commands or sending information [121].

- Most P2P applications can turn a computer into a server. There are several concerns here: an unskilled user can unintentionally share confidential data. Software bugs (exploited by an attacker) can result in a system disclosing sensitive information. Also, the intentional malicious use from the authorised insider can do the same, with a much higher impact.
- There is no uniform approach to communication encryption. Most of P2P systems do not include encryption, leaving the communication open to eavesdropping (which makes them unsuitable for corporate use). On the other hand, those that apply encryption, use it in a way that prevents firewall checking of the content going in and out of the organisation (end-to-end encryption).
- External attacks. Combination of P2P software bugs and malicious code can be used to compromise a company intranet by a remote user. For example, if a home user is using a VPN to connect to a corporate LAN, while running P2P application at the same time, "there are ways to manipulate that situation to gain the access to a corporate network through a VPN tunnel" [121].

The great potential of P2P architectures has been widely recognised. Applications such as distributed computing, collaborative work and content sharing are potentially very attractive for any organisational infrastructure. Still, the lack of appropriate level of security features is one of the main obstacles for wider usage of P2P within a corporate environment. As pointed in [113], [120] authentication, authorisation, and integrity and confidentiality of the data are the main requirements, not only for the corporate, but also for any informal use of P2P networks. In such a dynamic and distributed environment as P2P is, security becomes even more of a concern; and although sometimes it can put the constraints on some of the features of the system, it needs to be built-in with all the other functionalities, at the design level.

3.4 Grid Framework and Web Services

The concept of the Grid has emerged as a new approach to high-performance distributed computing infrastructure. Based on the Internet, the Grid seeks to extend the scope of distributed computing to encompass large-scale resource sharing including massive data-stores and high-performance networking, and shared use of computational resources, be they supercomputers or large networks of workstations. The Grid concept has been generalised to cover any Virtual Organisation, defined as any dynamic collection of individuals and institutions which are required to share resources to achieve certain goals [122]. Following this, hybrid Grid Computing and Peer-to-Peer solutions are now being put forward for applications in commerce and industry, including - but not restricted to - supporting distributed collaborative design and engineering, or distributed supply chains. In these contexts, the emerging Web

Services technologies are already playing a key role [123] and there has been a significant worldwide effort to consolidate Web Services and Grid Computing concepts [124]. Rapid advances are now being made in agreeing protocols and machine-processible message/document formats in order to enable open application-application communication and lead to ad hoc integration of systems across organisational boundaries into collaborations that may last for a single transaction or evolve dynamically over many years. Existing approaches to security within distributed systems are stretched by the extreme conditions imposed by the Grid, and significant effort is being undertaken in order to provide suitable security model. The most representative examples are the Grid Security Infrastructure (GSI) [125] developed as part of the Globus project [126], and Virtual Organisation Membership Service (VOMS) [127] developed by the EU projects DataGrid [128] and DataTAG [129]. The main objective of these security architectures (functionally very similar) is to provide a mechanism for controlling the user authorisation at the access points to the resources. In order to address this, both approaches make a clear distinction between the users, members of a Virtual Organisation (who claim access to the resources) and the resources (which are generally not part of a VO, but possessed by a service provider).

The Globus approach makes use of a trusted third party called Community Authorisation Service (CAS) [130], whose role is to issue authorisation certificates to the users. Each user is initially introduced in the VO by obtaining a X.509 public-key certificate from a certification authority. The service provider establishes the relationship with various VOs via the CAS. The CAS contains the access policy set by the provider with respect to each VO, as well as any policy constraints set by the VO for each of its users. The CAS is responsible for managing the security policies that govern access to the resources. Such a setup implicitly assumes a longer-term relationship between a VO and a service provider. When a user requires access to the resources, the CAS provides an intermediate layer in order to remove the need for a direct trust relationship between specific users and services. Instead, both clients and services establish trust with the CAS server as an intermediary. The CAS is responsible for managing the security policies that govern access to a community's resource. It allows resource owners to grant access to blocks of resources to a community as a whole, and lets the community itself manage fine-grained control (by setting internal policy for the community members). In order to gain access to a CAS-managed community resource, a user must first acquire a capability, defined by a CAS-maintained community policy database, from the CAS Server. A user approaches the CAS with two certificates: a certificate signed by the CA, and a self-signed public-key certificate (called 'proxy certificate'). The properties of proxy certificate are: it is short-lived, it contains a key from new key pair generated by user, and it is signed with user's primary private key. At the CAS, proxy certificate is combined with capabilities defined for the user. The purpose of this process is to relate user's identity with its capabilities, and to bind user's identity to the

access permissions. In addition, usage of the proxy certificates enables delegation of authorisation. For example, if a user applies for a job to a resource, the accomplishment of that task may require more resources. In order to avoid repeated user authentications (with the same or different CAS), the proxy certificate and capabilities presented by a resource perform that instead. Effectively, this forms a type of 'chain' that may extend until all required resources are gathered. Obviously, this delegation can be used in a malicious way by a hostile entity (therefore the proxy certificate is short lived, as defined by user). Once the capability is obtained, it is used to allow the user to access the resource, by checking it against the local policy information.

The approach used by VOMS (Virtual Organisation Membership Service) [131] to manage authorisation for users and resources in VOs closely follows the one of the CAS, with the main distinction of how policy is managed which impacts the way proxy certificates are used. VOMS issues *authorisation proxy certificates* in the following way: the user and VOMS authenticate each other and user submits the request and its credentials. After checking the credentials, VOMS generates an authorisation proxy certificate, signed by VOMS' private key. The user can repeat this process for several VOMS entities, e.g. if access to a number of resources (controlled by different VOMS) is needed. Finally, a user creates its own *credential proxy certificate* that contains authorisation proxy certificate(s) as its extensions. This certificate is signed by user's private key from the long-lasting key pair certified by a CA. This is the 'proxy' certificate used for job submission and credential delegation: The credential proxy certificate is used to apply for a job at the Resource Provider which is able to interpret the presented credentials (in the context of its local policy) and decides whether or not to grant access. For long running jobs, the credential proxy certificate is kept at a server, which can use the certificate to apply for further tasks if they are needed.

The main difference between the two approaches is that the CAS issues proxy certificates that contain the permissions on what actions a user is allowed to perform (based on both VO policy and resource provider policy). On the other hand, the VOMS-issued proxy certificates contain the user's credentials (based on a VO policy), and the resource providers (are able to) interpret these credentials (based on their local policy). This distinction has (potential) implications. According to [130], the CAS approach (of completely centralised policy) can achieve 'better consistency' in the environment where policies are changing dynamically. According to [131], the VOMS approach (by separating VO policy and resource provider policy) allows the ultimate access control decision to be made locally, by a resource provider that hosts the resource(s).

The framework and objectives of Web Services are quite similar to those of Grid. It aims to provide a language-neutral, platform-independent way of linking applications within

organisations, across enterprises, and across the Internet. The security framework for web services does not address the specific setup or architecture [132]. Instead, it aims to define a set of protocols with suitable XML-based [31] message format that should provide a basis for authentication, authorisation, confidentiality, and data integrity [32]. Another important requirement is that communication maintaining these properties should be processible by firewalls (rather than providing a mechanism for firewall penetration). Recently, initial steps towards merging the Grid framework and Web Services have been made [124].

3.5 Summary

This chapter has described the main approaches for supporting security in distributed communication systems.

Virtual Private Networks are being standardised for a commercial applications and offer consistent security model but, in order to maintain the level of security, impose a lot of restrictions in the flexibility of the communication.

Secure multicasting provides a scaleable means for maintaining group security and membership through key management mechanisms, but issues of sender authentication and fine-grained access control within a group are still active areas of research.

Peer-to-peer architectures demonstrate the power of direct interactions among users, introducing scalable and flexible means for communication. However, security has only recently stepped on to the list of priorities in the P2P research community; these efforts are moving from the pure peer-to-peer architectures, and are going towards the hybrid solutions.

The recent emergence of Grid computing and Web Services paradigms aim to develop a virtual organisations framework (and supporting security infrastructure), which should provide resource sharing and distributed collaborations among any number of individuals and institutions.

Currently, one of the main research challenges in the field of distributed collaborative environments is to provide a suitable security architecture that can offer scalable, flexible and secure means for maintaining dynamic and distributed collaboration perimeters across the administrative domains. The following chapter introduces an architecture developed to meet these requirements.

[31] XML stands for eXtensible Markup Language.
[32] With that respect, Web Services are being compared with JXTA Project to a large extent.

Chapter 4 Framework for Distributed and Secure Group Communication

The convergence of service and telecommunications technology is enabling new and more dynamic forms of collaborative environment where networked entities, be them (human) agents, applications, or service instances, share information and resources in order to achieve a common objective. Several architectural paradigms to support such virtual collaboration are emerging, mainly in the Peer-to-Peer, Web Services and Grid domains. Such collaborations are usually dynamic, often short in duration, and enacted by potentially large groups of collaborating peers which may join or leave the group as needed. They cut across organisational boundaries, therefore taking place on open networks (such as the Web or the Internet) and they may involve complex policies constraining possible interactions.

This chapter introduces a novel architecture of Closed User Groups (CUG), initially proposed in **[DJO1]**, for supporting the dynamic formation and self-management of distributed collaboration networks. CUGs can be understood as coordinated groups of peers who reside in different organisational domains, managed by an administrator. The issues described are signalling protocol and usage of certificates for group management, and functionalities for protecting dynamic CUG perimeters comprising data encryption and (distributed) end-entity security policy enforcement mechanism.

4.1 Motivating Example: Inter-Organisational Collaboration

Motivations for this research are introduced via the representative example in Figure 7. A team of engineers at a University is working on recommendations for a new aircraft wing. As a part of the work, researcher Alice needs to perform material analysis from the data obtained.

This is conducted on line by using specialised services provided by different Service Providers (SP1 and SP2), which have a long-term contractual agreement with the University. The services include analysis tools (hosted at SP1) and additional computational power outsourced to a supercomputing centre acting as SP2. Although Alice is qualified to use the tool, she prefers to do it with support of Bob, an engineer from a Research Institute who designed the prototype of the tool.

Alice belongs to team of researchers administered by the local administrator at the University, and not all of them may be familiar with the tool and the analysis process. The main activities of the material analysis are executed by end services hosted by SP1 and SP2. At some point into the analysis, the computation becomes more intense, and so more resources are called upon to

assure that the analysis proceeds satisfactorily. As the analysis progresses, Alice shares the results with her team at the university.

Each administrator wants to protect its local "private" resources from the general "public" which may include hostile agents. At the same time seamless interaction between Alice and the end-to-end services, as well as the computer Comp1 and its allocated resources, is highly desirable in order to facilitate collaboration objectives, i.e. a material analysis. The goal is, as the analysis proceeds, to create overlaying security perimeters to protect different collaboration teams that may exist over time (as a firewall would do in a fixed topology) while ensuring the security of each member as defined by their administrator(s).

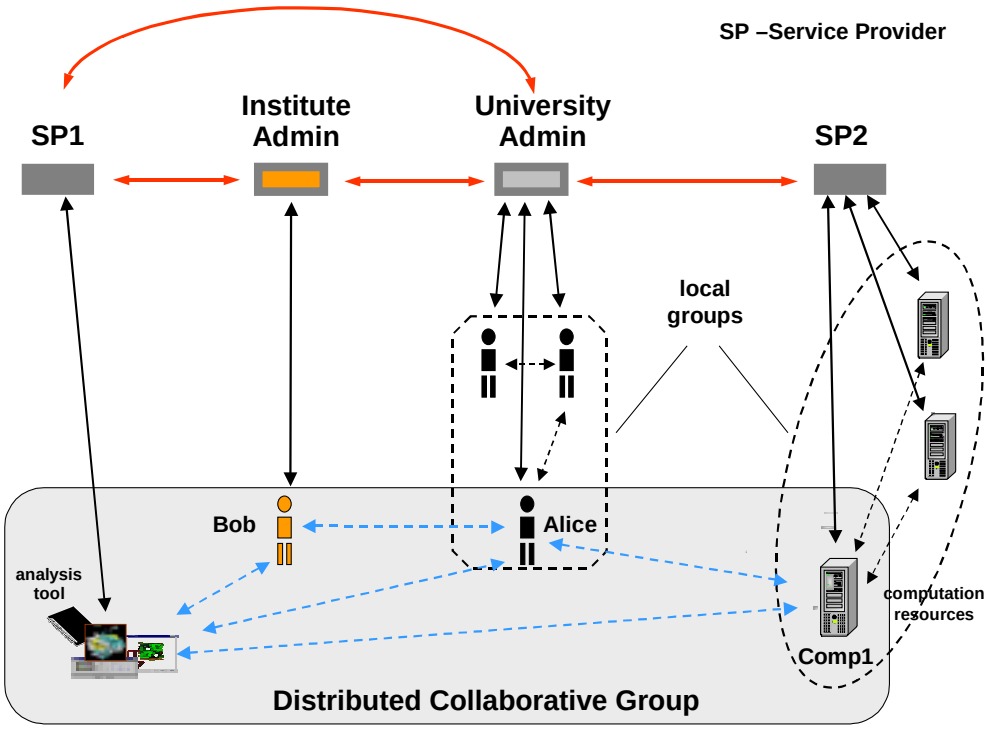


Figure 7: Motivating Example: Distributed Collaborative Project

This scenario highlights several issues related to secure collaboration in dynamic virtual organisations, as pointed in **[DJO3]**:

- Collaboration of entities that are controlled by different institutions. Each institution will have their own policies on access control and conditions of use.
- There is no single centralised administrative point. Security has to be achieved via a devolved policy management scheme combined with distributed enforcement at a peer level.
- The same entity may interact in different collaborations. A separation between those interactions has to be achieved.

- Collaborating entities may play different roles in their organisation and various collaborations, and different security policies may apply.
- Entities may be called upon to participate in the task without previous knowledge of the other participants. Trust between the entities needs to be established in a real time on a peer-to-peer basis.
- Entities need to be protected from their collaborators and the whole collaboration team has to be protected from outsiders, including other entities residing with the participating institutions.

A suitable architecture needs to be able to provide a security infrastructure that meets these requirements. The rest of this chapter will introduce such an architecture and gradually address the above requirements.

4.2 Closed User Groups (CUG) Architecture Overview

The logical structure of the architecture distinguishes several types of entities that participate in the formation of CUG environment: clients, administrators and (optional) trust authorities.

- **Clients** are networked entities and can be (human) agents, applications, or service instances. Software for supporting CUG and firewall functionality is localised to each host, be it a personal computer or mobile communications device. In order for client to involve in CUG communication, Administrator (which is normally entity of the same organisation) has to provide initial setup (registration). Every client is allocated a dedicated administrator node, which remains responsible for managing its clients' security policy.
- **Administrators** are responsible for a population of clients, and they can maintain many CUGs simultaneously. At a client's initial setup, an administrator defines client's security policy through means of certificates and firewall rules, and controls it as long as client remains registered. During that period, clients can request to create a new CUG or to join to a number of existing CUGs that are managed by their own or any other administrator. This is carried out by the administrator who is in charge of the CUG, through creation of CUG-specific certificate and its delivery to the appropriate client.

Administrator nodes may also be in touch with various Trust Authorities (third parties) who may assist them with authenticating the claims of their potential collaborators (e.g. for financial transactions or claims regarding professional expertise), or for initial authentication of the clients at setup process.

The architecture is illustrated in Figure 8. The circles representing different businesses do not have any topological implication, but only refer to the logical structure of the organisation. Members of different groups are distinguished by colour. Departments in the organisation

would naturally group their staff in different CUGs. However, clients can be members of several groups – such as the example of finance/sales person from the diagram. The firewall functionality is localised to each host, which (through login process) controls personalised user access through the firewall at the given host based on the certificates in its possession for each identified user(s). This enables different working environments for clients sharing the same terminal. The firewall enforces organisational policy (set by administrator at client's initial setup) and regulates client's interactions within CUGs (on the basis of group-specific certificate). All hosts then become part of a large distributed firewall, also providing all the features offered by a traditional firewall choke point.

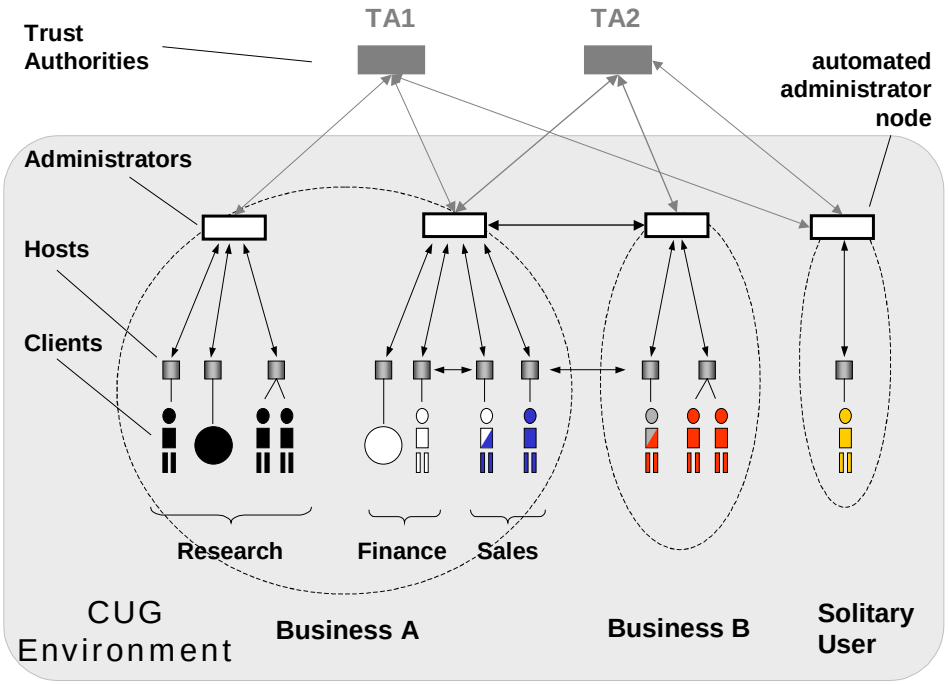


Figure 8: CUG framework: hybrid architecture and entities involved [33]

From the viewpoint of group-work, the CUG model distinguishes two main classes of roles: *group members* and *group manager*.

- **Group members** are peers (clients) who are informed of each others identity and location and interact with each other on a peer-to-peer basis by using a form of group certificate embedded in the messages they interchange. Messages with certificate information that does not match the required group certificate are either deleted or ignored without further processing.
- **Group manager** (Administrator) is responsible for managing group membership and issuing or updating group certificates (in terms of group membership and policy). The group

[33] Please note that a client does not necessarily refer only to human agents, but may also include applications, service instances, or any type of resources.

managers of one level in the organisational hierarchy may themselves be viewed as members of a CUG at a level above the level of the teams they manage.

This structure resembles line management in structured organisations. The managers maintain many simultaneous groups (distinguished and constrained by the group certificate issued by a manager) in terms of group memberships and security policies, and remain responsible from group creation until its termination.

For a particular CUG, the administrator and group manager can be the same entity, only performing a different role in each context. As an administrator of the organisation (which will be referred to as *local administrator* – **LA**), its main responsibility is managing basic company policies in terms of introducing (setting-up) new clients and assigning privileges to them. As a *group manager*, it is responsible for managing assigned CUGs from the group creation until its termination. By joining a group, clients become *group members* within the scope of that particular CUG. However, clients are free to become group members of the CUG(s) managed by entity other than their local administrator, which involves inter-administrator communication.

4.2.1 Basic Interactions in the System – Hybrid Architecture

Initial setup with a dedicated administrator is the prerequisite for all the clients in order to apply for CUG membership. Based on policy restrictions, a client may be free to decide to participate in CUGs of interest. Also, participation could be compulsory, (e.g., within an organisation, different resources and/or employees could be allocated to a specific project or work task that forms a basis for CUG creation). Once allocated, responsibility of CUG management and maintenance remains with the manager until CUG termination. All existing CUG members are informed of the arrival or departure of a client member and the CUG exists until the last member leaves. The role of the manager is to provide updated information on CUG membership and to maintain level of CUG security by defining authorisation privileges and assigning them to CUG members through certificates. Types of certificates and their usage are further elaborated in the next Section 4.3 Security Policy and CUG Management.

Once the client is admitted to a group, it interacts directly with other CUG members, without the manager's involvement. Occasionally, interactions between administrators may take place as appropriate, in order to support management of inter-organisational CUGs. Figure 9 illustrates three main types of interactions in the CUG environment:

- ***me2me*** (*member-to-member*): direct peer-to-peer communication between the members of the CUG, such as file transfers, chat, white boarding, process invocation, etc. This is

supported with a certificate unique to each group, which is issued by a manager responsible for the group creation and maintenance.

- ***me2ma*** (*member-to-manager*): regarding the interactions related to CUG management, e.g. manager's updates of group policy. This is supported through mutual manager-member authentication by means of certificates. Here is important to stress that client's participation in a CUG is not a prerequisite for this type of interaction (e.g. one of the typical examples of such a interaction is client's negotiation with the administrator for creating/joining a new group).
- ***ma2ma*** (*manager-to-manager*): direct peer-to-peer communication between members of the 'group of managers'. A typical example of this type of interaction would be supporting the formation of extranets, when clients from different organisations (or organisational units) engage in collaboration that spans organisational and geographic boundaries.

As shown in Figure 9, communication within CUG environment is supported on two distinct peer levels (one among administrators and another among CUG members), as well as through hierarchical interactions between clients and their administrators. Manager(s) of different organisations maintain different groups (distinguished by the different colour of the clients), and some clients are members of several groups.

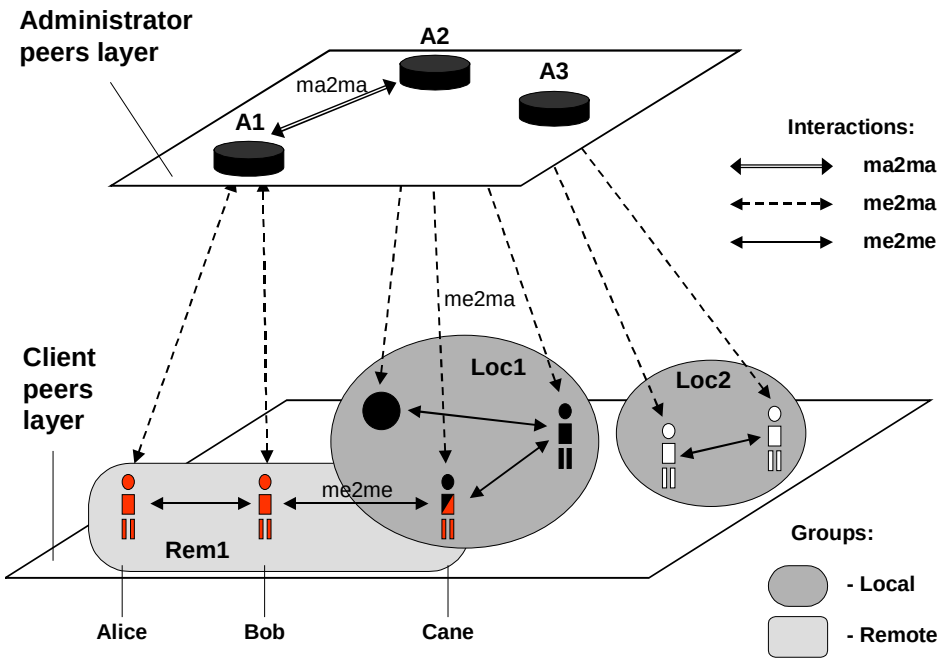


Figure 9: Types of interactions and groups within CUG environment

If a CUG manager is also the administrator of all the clients involved in the group (such as example of groups Loc1 and Loc2), that type of group is referred to as a *local group*. The

relationship between the administrator and clients exists even before they become CUG members, and management of that group is straightforward – through direct communication between CUG manager and its members.

However, if CUG members reside in different organisational domains, they are initially affiliated to their own administrator(s), but there can be only one CUG manager per group. Normally, this will be one of the administrators whose clients want to participate in the group, and this type of CUG is called a *remote group*. An example of such a scenario is the group Rem1 and Cane from Figure 9. This type of interaction is an important feature of the proposed architecture since it contributes the scalability and flexibility of group-work.

4.2.1.1 Interactions for Supporting Inter-Domain Groups

The assumption is that Cane wants to join an already existing group Rem1 for purpose of inter-organisational collaboration. Until that point, Rem1 is just a local group where A1 is both the administrator and the CUG manager of Alice and Bob who are simply *local CUG members*. For either of those two clients, joining Rem1 was a straightforward process involving direct request to A1 and a positive response.

However, Cane belongs to a different organisational domain, and therefore has a different administrator (A2) responsible for maintaining his security policy and firewall set-up. Since A1 is the manager of remote group Rem1, communication between Cane and A1 needs to be provided. Normally, a client is prevented from contacting a remote administrator node directly, due to policy restrictions imposed by its local administrator.

For Cane to be able to initiate joining the remote group Rem1, prior endorsement of its administrator A2 is required. Typically, Cane will contact its administrator A2 with a request to join a remote group (***me2ma*** interaction). If A2 approves this, then A2 forwards client's request to A1 (***ma2ma*** interaction). A1, being manager of the group, can endorse or reject client's joining request. If the request is accepted, A1 will create a certificate defining Cane's privileges within group Rem1, and send it to A2. Administrator A2 acts as a proxy, sending requests and receiving responses on behalf of its client Cane. If the group policy defined by A1 (for the purpose of Rem1) does not conflict with the policy that applies to Cane's role in the organisation (which is managed by A2), the response will be forwarded to the client. The certificate provided by A1 will enable Cane to establish direct ***me2me*** communication with Alice and Bob within the group Rem1. However, Cane will still not be able to contact group's manager A1 directly, and for any further requests will again have to go via his administrator A2.

Clients from local group Loc2 would normally have to follow similar procedure in order to join remote group Rem1. However, depending on the policy at particular clients' firewall, they may be able to contact remote group managers directly, in order to join CUGs administered by them. In such a scenario a client can choose to join any of the remote groups (an example for this would be a senior manager role in a large organisation with a number of branches worldwide).

This approach introduces additional flexibility for the creation of inter-organisational collaboration CUGs. Normally, there is no limit on the number of administrative domains which clients can form the same CUG. Clients belonging to a number of administrators can participate in the same group, as long as common policy agreement can be achieved among administrators. The group manager, who is chosen via self-initiative or voting, remains in the same role during the group's lifetime. A group manager can simultaneously maintain many groups, and is at the same time an administrator of the clients that originally belong to its organisation.

4.3 Security Policy and CUG Management

The management of the security policy is a complex process involving policy definition, policy distribution and policy enforcement. In order to deliver a robust and secure infrastructure for managing of dynamic groups, the following issues need to be addressed:

- A scalable means for management and updating of the organisational and group policy.
- Protection of the communication (data exchange) between the entities involved in the group.
- A suitable way for the policy enforcement that offers protection to dynamic CUG perimeters, as well as to the individual members (both from other group participants and from malicious outsiders).

Group management within the CUG environment is supported through the use of public key certificates (for authentication), attribute certificates (for authorisation) and (optional) use of symmetric keys (for data confidentiality) [6]. Policy deployment model of the CUG architecture combines default security settings with a role-based access control approach. The proposal, summarized in **[DJO4]**, exploits a variation of the distributed firewall concept [68]. Policy is defined at the administrator / group manager level based on the anticipated role of the client in the organisation / group. It is then distributed to the end-entities (client hosts) by the means of the certificates and firewall rules, where it is enforced by each individual entity that participates in a distributed firewall. The security perimeter can be easily extended to safely include remote hosts and networks (e.g., telecommuters, extranets), therefore eliminating any topological obstacles to the proposed paradigm of CUGs. There are two essentially different classes of policies:

- *Local Policies*, which are owned and maintained by the administrator and apply to the clients associated with its organisational structure. Those policies are defined at the creation of the organisational structure, and are accordingly modified [34].
- *CUG Policies*, which are defined at the CUG creation and are maintained by the CUG manager. In the case of a remote CUG, the definition of these policies is not solely the responsibility of the CUG manager – they can be (and normally are) negotiated with the clients' dedicated administrator(s), who can impose constraints on the CUG policy in order to avoid the possible conflicts with the organisational (local) policy.

Local policy is distributed to the client at the initial setup (registration). It comprises of: default firewall rules for the client host (for securing the lower level of the communication protocol stack), an authentication certificate (that establishes the identity of a client within the architecture) and the access privileges (based on a defined role of a client within the organisation).

CUG policy is shorter-term than local policy, being more dynamic and addresses smaller population. It is defined based on the client's role in the group, and delivered at the time of joining by means of the attribute certificate. It provides a more sophisticated method in terms of access control, authorising the members (only) to perform certain actions during the group communication.

4.3.1 Authentication

Authentication of the entities within the CUG environment is supported through the use of PKI (Public Key Infrastructure) certificates - PKC. The paradigm of public key certificates typically assumes that every subject creates its public-private key pair. The public key is then transferred to a certification authority (CA) in an authenticated way (by some off-line means), which then signs it with its private key. Certificates accompany the messages exchanged, verifying the authenticity of the sender and (indirectly) the integrity of the message. Currently, the most widely exploited technology for the use of PKI certificates is X.509, proposed as an Internet standard by Network Working Group of IETF [32].

The certificate structure of each organisation participating in the CUG environment can be seen as a single-level autonomous PKI, where the administrator is the CA and clients are subjects. A client becomes a member of the overall CUG environment by sending an initial *register request* message to the administrator, where each client presents its identity through a public-key certificate issued by some of the commercial CAs belonging to any PKI, as long as the PKI is

[34] In the 'real world', this would be normally drawn from the recommendations of the high-level managers in the organization.

recognised by the administrator. (VeriSign [39] is an example of a widely recognized certification authority). In a similar way, the administrator authenticates itself to a client.

This enables the administrator and a client to exchange public keys with confidence, after which the administrator creates a local public-key certificate, to be used by a client for all the authentication-related purposes within the CUG environment. Through use of a 'local' PKC the administrator keeps control of who is registered with the organisation and for how long (through assigning the validity period of the certificate). There are a number of fields contained in the PKC certificate. Figure 10 reviews those relevant to the CUG architecture:

- *Certificate Serial Number* is unique per CA (which is the administrator).
- *Validity Period* is expiration date (the reference time needs to be agreed or specified within the certificate).
- *Subject Public Key Value* is client's public key, generated by the client and securely transferred to the administrator at the initial setup.
- *Subject Unique ID* is the unique identifier issued by the administrator, and it needs to be unique per administrator [35].
- *Issuer Unique ID* is the unique identifier of the administrator.
- *Digital Signature Algorithm* is a reference to a specific algorithm used by the administrator to sign the certificate with its private key.
- *Issuer Digital Signature* is the product of the applying of the administrator's private key to the remainder of the certificate, using the referenced algorithm.

Further on, this 'local' authentication certificate will be referred to as PKC (PKI certificate) – the usage of the authentication certificate issued by a commercial CA will be additionally pointed out and its use justified. Normally, these are always the certificates belonging to the administration nodes and exceptionally to the client (at initial setup).

The PKC does not grant any authorisation privileges to the 'subject' – its primary use is to authenticate the subject (carrier of the public key) to other entities and to establish the unique identity of the client in case of inter-domain interactions. Its use is limited to the CUG environment – it has to be recognized and validated in all CUG interactions, but it may not be recognized outside of the CUG architecture. However, it is reasonable to expect that the management of the organisation would try to assure that the certificates issued by its administrators are widely accepted; i.e. by having a reputable commercial CA to issue the certificates to the administrator(s).

[35] One of the aspects where PKI paradigm proves impractical is that it fails to satisify a requirement of the unique identifiers on the global scale. However, this is not a problem in a 'small-scale' CUG environment since the IDs are issued by the security administrator to a finite population of clients, and the combination of the issuer's ID and the subject's ID is unique.

The example of the usage of the local PKC is client's negotiation with the administrator for creation or joining a group. Optionally, the client can authenticate the administrator against administrator's original certificate, issued by a commercial CA. Another example would be a mutual authentication of the CUG members prior to the establishment of peer-to-peer session within the group. Members may reside in different administration domains and therefore may not be able to contact the CUG manager directly for the purpose of group management (as demonstrated earlier). However, if a CUG manager is also the administrator of one of the members, it has (by the definition of CA) to accept the request of another client and verify the certificate presented, acting as certification authority.

A 'commercial' PKCs are used within the CUG environment for the authentication of the administrator nodes, and as a 'root' certificate upon which the trust and legitimacy of the local certificates is built on. For that reason, the uniqueness of the administrators' IDs built in the certificates has to be agreed on prior to the establishment of inter-domain collaboration. Also, if these certificates are issued by different certification authorities, all the CAs need to be recognized by all the parties involved.

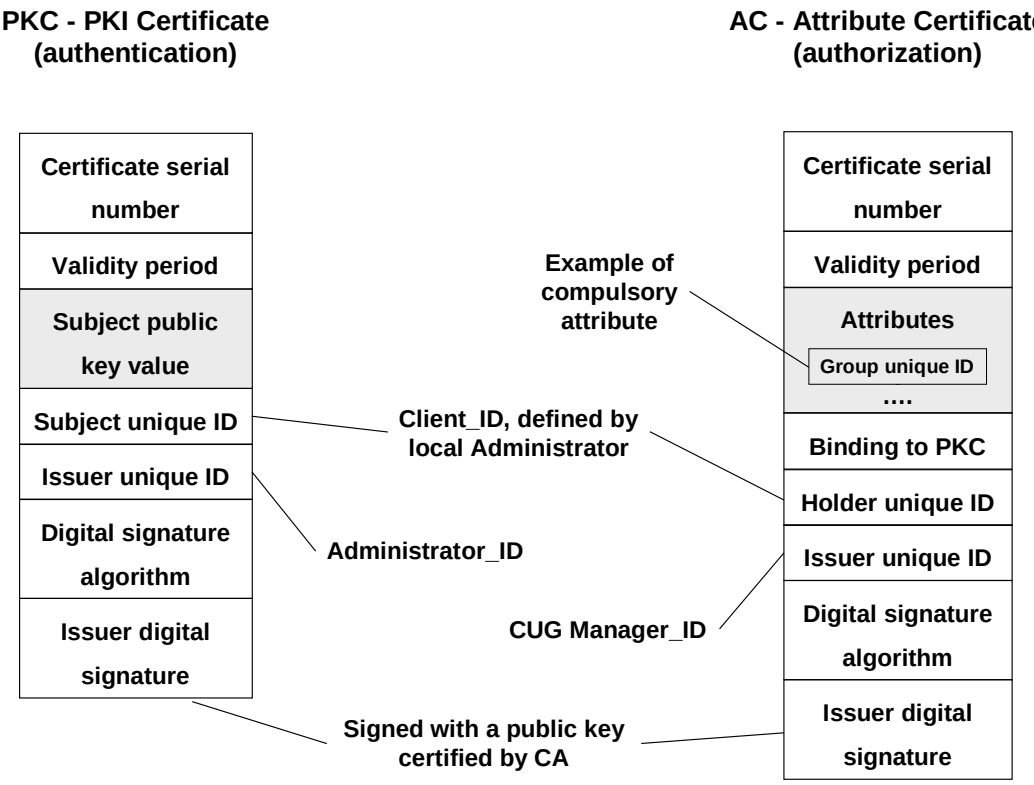


Figure 10: Relevant Fields of PKI and AC Certificates

4.3.2 Authorisation

Another type of certificate used within the CUG architecture is authorisation certificate, used primarily to regulate group management and member's privileges within the group. There are a number of approaches for allocating authorisation rights to the entities in the system, sometimes depending on the approach taken in the policy definition. In addition to PKI certificates, the specification of X.509 certificate format provides attribute certificates (AC) where the issuing authority can specify a set of credentials authorising the holder of the certificate to claim certain privileges [48].

Within the context of the CUG architecture, the AC issuer is a CUG manager, which defines the group policy (based on the roles and privileges they carry), and expresses it as credentials granted to the group members via attribute certificates. Credentials are the abstraction of the privileges, and each group member is granted the credentials that relates to its anticipated role in the group. In addition to this, each group member is given at joining time a set of rules (comprising a CUG policy). Once in place, it enables mapping of the certificate credentials into authorisation permissions during peer-to-peer communication between group members. There are a number of fields contained in the AC certificate, as specified by [48]. Figure 10 reviews those relevant to the CUG architecture:

- *Certificate Serial Number* is unique per CUG manager.
- *Validity Period* is the expiration date (the reference time needs to be agreed or specified within the certificate).
- *Attributes* field carries the information about the holder, i.e. about the actions it is entitled to undertake within the CUG, as defined by the certificate issuer (CUG manager).
- *Holder Unique ID* is the unique identifier, the same as the *Subject Unique ID* in the PKC issued by the administrator. Such an approach facilitates the correlation between the group member's identity and its credentials, if this data needs to be verified.
- *Issuer Unique ID* is the unique identifier of the CUG manager.
- *Digital Signature Algorithm* is a reference to a specific algorithm used by the CUG manager to sing the certificate with its private key.
- *Issuer Digital Signature* is the product of the applying of the CUG manager's private key to the remainder of the certificate, using the referenced algorithm.

Normally, an attribute certificate does not contain the holder's public key, but it may contain the reference to the public key certificate and / or CA which can verify the holder's PKC. In such a way, the PKC and AC (which are presented separately) can be easily correlated for verification purposes. This also allows flexibility in the policy definition, since the PKC (and client's duration in the environment) will normally last for a longer time than the AC (and the membership to a group). This information can be referenced as an attribute. The attribute field

contains the sequence of attributes. Some of them are compulsory for every CUG and every member, and those are: ID of the group, ID of the member's local administrator and serial number of public key certificate. The rest of the attributes carry the client's credentials, as defined by the CUG manager.

The attribute certificate is created and delivered to a member at the time of its joining a group. If a new member resides in a domain of the administrator other than the CUG manager, then inter-administrator communication is required to support the certificate delivery. Before the joining negotiation commences, this activity would be typically preceded by communication with a common trust authority to validate the authenticity of both administration parties. Depending upon the nature of the communication, or the administrators' preferences, different trust authorities may be consulted. Assuming that both parties agree to negotiate then the CUG manager will generate and hand over the certificate to the local administrator of the client host wishing to join the CUG. Normally, the administrator acts as an intermediary and forwards this message to the client, enabling it to become the group member.

However, while in the possession of the certificate, the administrator can examine it against the client's privileges in the organisation, set by the local administrator at the time of client's registration. If the privileges given by the group-specific certificate exceed those initially granted to the client, the administrator may refuse to accept it and can contact the CUG manager requesting the modification of the client's role and adjustment of the credentials to an acceptable level. If the certificate is accepted, the administrator endorses it by co-signing it before forwarding it to its client.

4.3.2.1 Roles and Privileges

Roles are logical abstractions used to express the group policy and relationships in a structured way. They consist of a set of policy statements defining what type of interactions a particular member can perform or accept during CUG communication.

From a CUG perspective, the CUG manager, whose location is immaterial, correlates a role to the set of credentials and assigns it to a member via an attribute certificate. The actual group policy is transferred to a member in the form of a role-matrix that expresses mapping from the credentials (contained in the certificate) onto the role. This enables a CUG member to obtain a particular role only if appropriate credentials are presented, which allows it to access and enforce the associated privileges. This approach addresses scalability in several ways:

- First, roles are more generic than member IDs and reduce the overhead of managing and enforcing security policies.

- If the privileges associated with any role are modified by the CUG manager, the update to the role-matrix is delivered to the members without issuing new and revoking old attribute certificates (AC) for the whole CUG.
- If a role of the specific member is changed during its membership in the CUG, it will be issued new AC by the CUG manager. The old certificate will be revoked and the rest of the CUG will be informed, but no changes need to be made in the CUG policy.
- Finally, each endpoint implementing a firewall does not need to maintain information about the enforcement of the complete set of policies that may apply to a large network. Instead it is only concerned with the policies that are relevant to the CUGs it participates in, and from those only the subset of rules that needs to be retrieved and enforced during the CUG communication, depending on the roles of the entities participating in the current session.

In a similar way, the local administrator defines the organisational policy and delivers it to a client at the initial setup (local policy, as defined before). The privileges specified here do not express the possible relationships within the organisation (since all peer-to-peer communication is carried out via CUGs); rather, they can be used by an administrator to put a constraint on the CUG policy that applies to a client, and prohibit certain actions. This can be also done by the administrator during the endorsement of a client's membership to a CUG (as elaborated earlier).

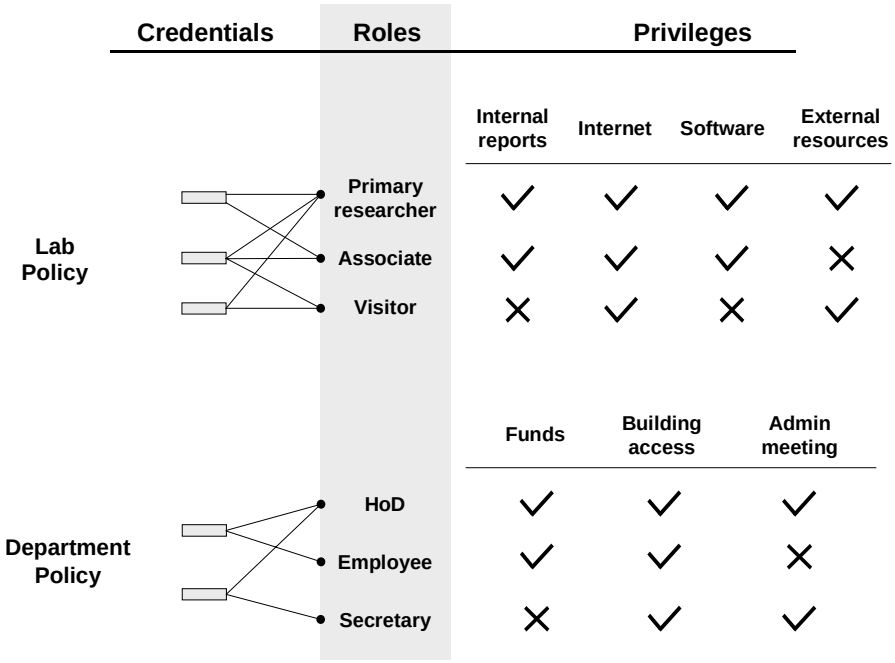


Figure 11: Example of Logical CUG Policy Expressed as Role-Matrix

An example of the CUG policy expressed as a role-matrix is given in Figure 11. The department can be seen as an organisational structure (e.g. part of an University), where different employee roles are defined. Also, within a department, there are a number of research laboratories,

organised as CUGs. The policy of each of the CUGs defines the privileges associated with a specific role; the model given can be easily elaborated in more detail where, for example, it is explicitly specified which software is accessible for the researchers of each separate lab. In the context of organisational policy, both the 'primary researcher' and 'associate' would have the same privileges if their role were defined as 'employee'. Therefore, if they were trying to access internal report that is classified as 'admin', the attempt would fail. However, if a primary researcher gets promoted to a Head of department (and the appropriate part of his/her local policy is updated accordingly), the same document will become available to him/her.

4.3.3 Confidentiality

Confidentiality of the communication between the entities (through data encryption) is an important factor in enhancing overall security of the scheme. On the other hand, performance of the secure protocols (in general) mainly suffer due to time-consuming encryption operations. Therefore, this can be regarded as an optional feature for the applications requiring confidentiality.

Encryption in the CUG architecture is supported through the combined use of symmetric and asymmetric encryption, targeting the following:

- Protecting of peer-to-peer communication between group members (data transfer).
- Protecting the transfer of the security policy.

Symmetric keys are used for the interactions where large amount of data are transferred (introducing less processing overhead) or where the number of entities sharing them is not very large. Since it is easier to break symmetric encryption, these keys need to be updated more frequently (or distributed per-session). This can introduce a significant performance overhead. They are used for data transfer between CUG members, for transfer of security policy at initial client's setup and for inter-administrator interactions to support group management of remote CUGs.

For the interactions where large number of entities would need to share a symmetric key (such as interactions between the CUG manager and its members) this approach is avoided since it may decrease the level of security of the system. For example, if a single symmetric key was used for interactions between the CUG manager and its members, it would need to be updated every time a member leaves the CUG. This would have to be done in a very consuming manner (either with Diffie-Hellman or RSA key exchange mechanism [6]) and then followed by 'useful' data encrypted with a symmetric key. This can be justifiable only if the amount of data to be transferred is significant. The usage of encryption will be addressed in detail in Section 4.4 Description of Security Protocol, both from the scalability and security viewpoint. The encryption keys used within the CUG architecture are summarized in the Table 1.

Table 1: Summary of encryption keys and its usage in CUG architecture

Type of key	Key holder(s)	Description of usage
Symmetric Key 1	CUG Managers/ Administrators	To secure communication between local administrator and CUG manager while client is in remote CUG
Symmetric Key 2	CUG members	Between users in the same CUG during the single session. Created by one of the members and distributed using public/private key pair 2
Symmetric Key 3	Administrator/ client	Used at the initial setup for transfer of security policy and client's public key
Public/ private pair	CUG Managers/ Administrators	Signed by CA and used for administrators' authentication within and out of CUG architecture
Public/private pair 1	Clients	Signed by CA and used for client's authentication at initial setup request and out of CUG architecture
Public/private pair 2	Clients	Signed by local manager, used for the client's authentication within the CUG environment and for secure delivery of AC certificates

In addition, although it is not focus of this research, it is important to note that all the keys (as well as the security policies) should be securely stored while on the disc (e.g. password protected and / or encrypted). Also, a history of the interactions that a client has performed as a member of different groups should be maintained and kept secure. This data can be used by an independent Trusted Third Party (TTP) in order to resolve the dispute that may have arisen as a result of CUG policy non-compliance. Mechanisms for this are examined, as one of the possible avenues to extend the architecture beyond the scope of this thesis, which is discussed in Section 7.3 Further Work.

4.3.4 Centralised Policy Management: Administrator

The definition of the policy requirements and management of group membership is supported through the functionalities at the administrator node, which effectively acts as a Security Service Provider (SSP) to the end entities. This information is used to regulate the specific actions a client is permitted to undertake and to ensure a sufficient level of both host and CUG security from the outsiders (through policy enforcement). Consistent with the distinction of the local and CUG policy, the general structure of the administrator node distinguishes three main logical components:

- Administrator module for supporting the *local policy*.
- CUG Manager module for supporting the *CUG policy*.
- Protection infrastructure module for providing the security for previous two units.

This research considers only the first two modules. Distinct from the client node, where local and CUG policy recommendations are essential for enforcing the security of distributed group, the administrator node (which can be seen as a 'server') is a high-value node. It is a common practice in any form of organisation for this type of entity to be protected with the additional commercial products, including even the physical security (i.e. stored in a room with restricted

physical access). The methods and systems to provide a suitable security are well documented in literature, and may include multiple firewalls, intrusion detection systems, internal and external gateways for total separation of the system (via de-militarised zone), etc [9],[59]. In addition, a continuous back-up process, as well as redundancy of the system for increased reliability is a preferable approach. Standard implementation of the GSM PLMN (Public Land Mobile Network) adopts the similar approach when deploying servers for keeping user-data and billing information [133],[134].

The general structure of the administrator node is given in Figure 12. The separation of administrator and CUG manager functionalities is consistent with the logical notion of local and CUG policy, allowing increased deployment flexibility. For large systems, it is possible to implement them as separate nodes, even for the same organisational domain. Since inter-administrator communication is provided, this leaves an option of total separation (as two different entities that maintain their own PKI certificates and identities), or as two logically distinct part of the same unit that deploy the common policy (in a different context).

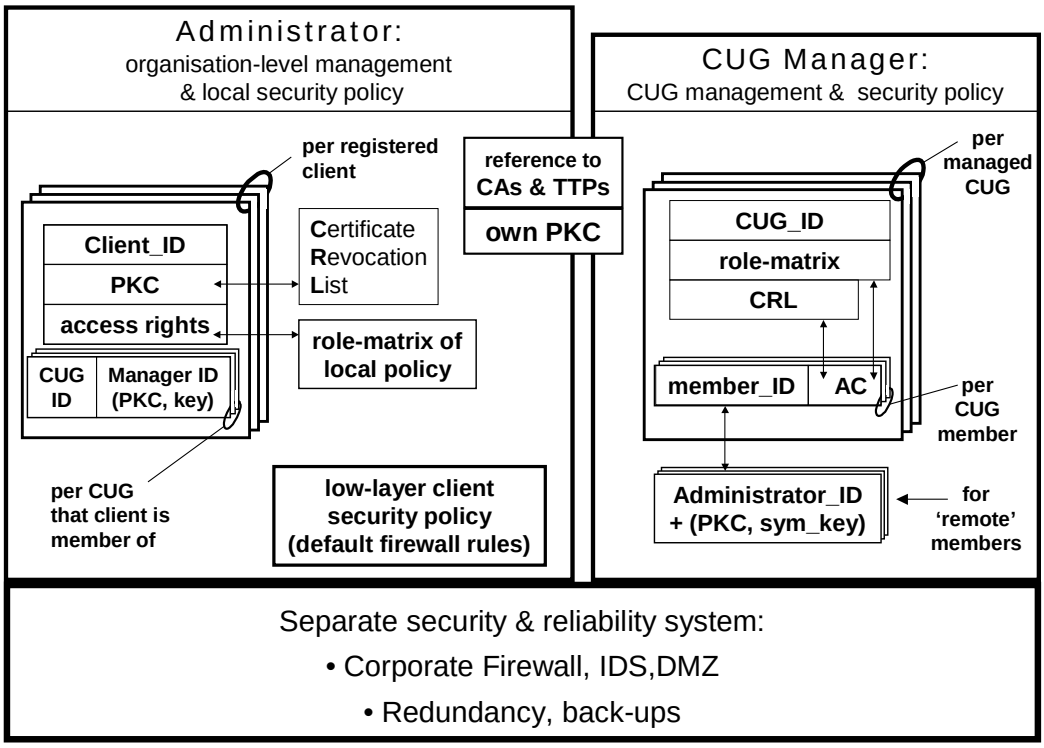


Figure 12: Functionalities of Administrator Node

The administrator module provides management of the client population and security policy deployment at the organisational level. At a client's setup it allocates a locally unique ID and, based on the client-generated public/private key pair, generates the PKC for a given ID, which allows the administrator to set a validity period of the client's registration. Also, based on a pre-

defined client's role in the organisation, the administrator defines access privileges (from the role-matrix) for that particular client. In addition, the administrator maintains the set of rules comprising the low-level generic security policy (that applies to all the registered clients), such as connection restrictions, permitted ports, URIs and/or protocols.

For every registered client, the administrator keeps the information about which CUGs a client is member of, as well as the IDs of the CUG managers, their public key certificates (PKC) and the shared symmetric key to support inter-administrator communication (in the case of remote groups). There is no need to store the actual CUG policy or even the part of that policy that applies to the client, since that has been endorsed and agreed upon in the process preceding the client's inclusion in the CUG. Also, the administrator maintains the list of the clients' revoked certificates (i.e. of the clients deregistered before the validity period ends). Upon the expiration of the client's PKC certificate, an administrator may renew it, as well as to update the client's access privileges if client's role in the organisation changes at any point in time.

An administrator additionally has to accept any request regarding validation of its clients' certificates, regardless of whether the request originates from within or outside of the organisation. This is by definition the responsibility of the Certification Authority (the certificate issuing entity). The administrator may also maintain the list of the available/preferred CUGs and advertise it at the client's setup or at the policy update interactions.

The CUG manager is responsible for maintaining and managing a number of groups in terms of security policy and group management. For every newly created group it allocates a locally unique identifier and defines the role-matrix. At the time of the client's inclusion in the group, the CUG manager creates the member's attribute certificate (AC), based on the member's anticipated role and validity period in the group (as described previously). The role-matrix and members' role may be negotiated among (or suggested by) the administrators which clients will become the group members. The AC, the role-matrix, and the list of current CUG members are then delivered to a client via its administrator. For every given group, the CUG manager maintains the list of its members and their certificates, as well as the list of revoked certificates (CRL). ACs are expected to be of much shorter lifetime than PKCs, due to their nature. Also, the accurate definition of the validity period (e.g. based on the purpose of the client's membership) may save from maintaining large CRLs; for example, one of the approaches suggested in [48] is issuing of single session ACs that can be re-negotiated. Similar to the administrator functionality, CUG manager keeps the details of the administrators' IDs, their

public key certificates (PKC) and the shared symmetric key, for each of the 'remote' members[36].

In case that administrator and CUG manager constitute different functionalities of the same entity, they may share the same PKC (granted by a commercial CA) which uniquely authenticates the administration node. Together with the list of recognized commercial Certification Authorities and Trust Third Parties, this would be the only information shared between the administrator and CUG manager functionalities.

The actual policy deployment is always performed by the administrator. Once delivered to a client, the policy is enforced by the enforcement agents on the client's host. Enforcement agents are controlled and (re)configured by the corresponding administrator using a master/slave interaction model. Neither the client nor the CUG manager can access and reconfigure these enforcement agents. Notably, such a distribution of responsibilities and decoupling of policy specification from deployment is consistent with the CUG interaction model: A (remote) CUG manager can interact with a member only via that member's administrator. CUG policies and policy updates are communicated by the CUG manager to the corresponding administrators who have the responsibility for their deployment among their clients that participate in that CUG. Deployment is initiated by compiling the CUG policy statements and distributing to the corresponding members the rules that apply to them. Prior the deployment, an administrator performs a check of a policy suggested by the CUG manager. In case there is a conflict between the local policy (imposed by the administrator) and the CUG policy (suggested by the CUG manager in relation to the particular CUG), in the current approach, the local policy will always override the CUG policy.

4.3.4.1 Names and Identities

In order for the proposed architecture to function properly, it is necessary that each entity be uniquely distinguished by means of an identifier. For convenience, two types of identifiers are proposed: names and identities, as follows.

Identity (ID) is a true unique identifier. The architecture contains the identities of the clients and the identities of the groups. The former are defined by the administrator at the time of client's setup, and are embedded in the PKC (public key certificate) carrying client's public key and used for the authentication of the client. Those identities are adopted by the CUG manager at the time of the client's inclusion in the CUG and are related to the client's AC (attribute certificate) that contains credentials for mapping a role and retrieving the privileges of a client during CUG

[36] If there is a number of CUG members originating from the same administrator, the full information will need to be stored only once, with the cross-reference to the appropriate member.

interactions. The latter (group identities) are defined by the CUG manager at the time of the CUG creation, and are also embedded in the ACs granted to the group members. Both client and group identities retain the value as long as corresponding entity is present in the CUG environment. For the sake of scalability and ease of the operation, it is not required for the IDs to be unique across the whole CUG environment, as long as they are unique per entity which is creating them. For example, uniqueness can be achieved if the IDs of the entities are formed as a concatenation of the ID generated at the time of the certificate issuance, and the ID of the issuing entity (effectively, this would correspond to the uniqueness 'per domain', as proposed for X.509 certificate format [135]). However, the uniqueness of the IDs of issuing authorities needs to be confirmed prior to the formation of the CUG environment. Also, the certificate issuing authority must continue issuing unique IDs as long as it remains a CA. At the client host, relevant IDs (of the CUGs and group members) are stored by the administrator at the distributed firewall instance, and as such are not accessible or modifiable by the client itself.

Names are an optional description of the entities that are reachable by a client. For a human user, this would normally be a descriptive string that carries a meaning, related to the entity functionality, personal relationship, etc. As such, the names do not have to be unique on a larger scale. The default names (given by the administrator / CUG manager) can be modified by a client, or simply adopted (e.g. the administrator can advertise the available groups under the suitable thematic names). Once allocated, the names are stored at the client host and are the only part of the application accessible by a client. For example, if Alice chooses to contact Bob under the group 'research project', the application will take those user-defined identifiers and translate them into unique IDs that have the operational meaning for the firewall instance.

4.3.5 Distributed Policy Enforcement: The Client

Once distributed by the administrator, the relevant policy is stored at the clients' distributed firewall instances, accessible for enforcement purposes during the interactions (right and left part of the Figure 13, respectively).

Upon the successful agreement of the registration, the client receives relevant *local policy* by its administrator. It consists of the client's PKC (signed by the administrator), access rights, and default security rules for the network-level firewall operation. In addition, the client stores the administrator's details (ID and PKC), to be used for every subsequent communication regarding client's requests, group management and policy updates. The default security rules effectively comprise the client's distributed firewall configuration for the packet filtering, according to the previously defined organisational policy. In addition, communication passing at this level is subject to a traffic monitoring via a host-based intrusion detection system (IDS), integrated with

the packet filtering mechanism[37]. However, the focus of this research is not to develop a new intrusion detection system. One of the suitable schemes that can be adopted is a network-level monitoring using packets as the data source. Such an approach is described in [64], based on the modelling of normal states and state transitions of the TCP/IP protocol. The data of the default security rules is occasionally updated (triggered by the administrator), in order to modify the filtering rules or to update the signatures and patterns governing the traffic monitoring, based on the newly discovered vulnerabilities.

CUG policy is received upon the client's inclusion in the group. For every CUG that it participates in, the client maintains an appropriate database, consisting of a personal attribute certificate, the current list of the CUG members, a role-matrix for retrieving the privileges associated and the list of the revoked certificates. CUG policy applies only within the domain of a particular group, whereas the local policy set initially applies to all the client's interaction, in different contexts and across the all CUGs it is member of.

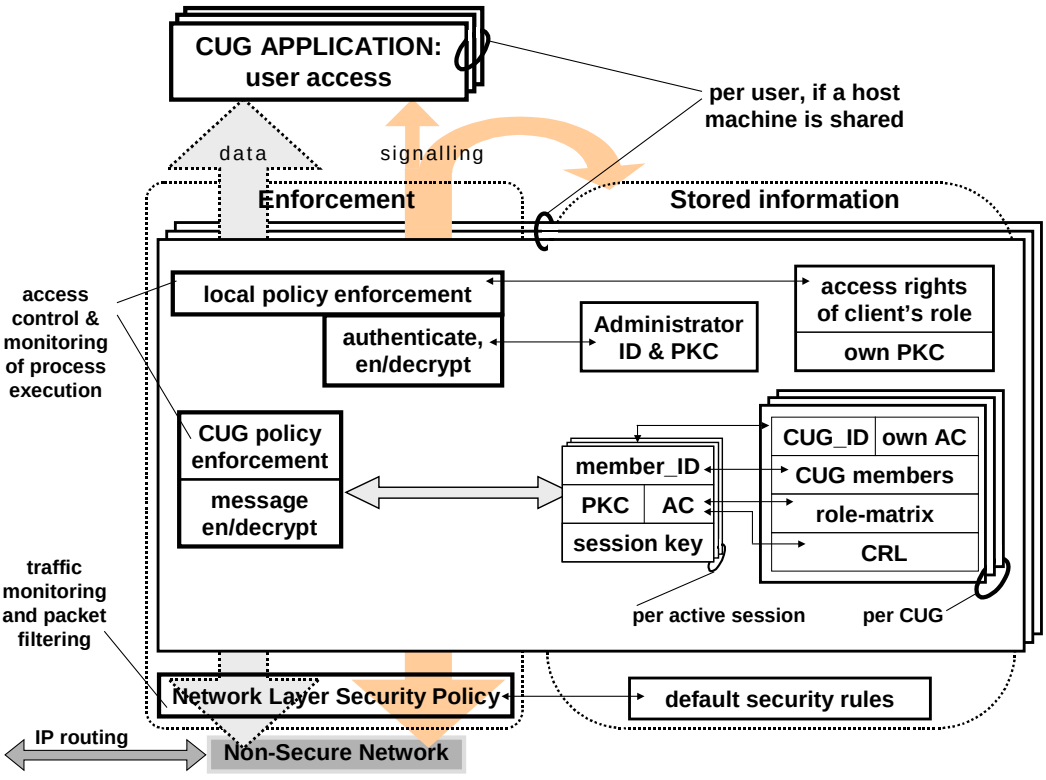


Figure 13: Functionalities of Client Node and Security Policy Enforcement

The firewall functionality itself is localised to each client host, be it a personal computer or a mobile communications device. This controls personalised client access through the firewall at the given host based on the certificates in its possession for each identified client(s). All hosts

[37] In a simplified form, this has been successfully tried (although not in a distributed environment); e.g. [71] describes a simple implementation of integrated firewall / intrusion detection mechanism in the kernel of Linux operating system.

then become part of a large distributed firewall providing protection to the dynamic CUG perimeters. Typically, the member(s) at each host terminal are individually assigned privileges that permit their host firewall to send and receive all forms of information between themselves and other members of the same group as defined by the certificate(s) they have in common. In addition to this, several users can share the same machine. The default security rules are common for the machine, but the separate security profiles are provided for each of the users (based on the organisational role and the CUG membership), though they share the same (physical) firewall instance.

Figure 13 indicates the layered order of the actions performed for security policy enforcement **[DJO5]**. Default security rules, operating on a network layer are consulted for every interaction. Communication between the client and its administrator is additionally secured by checking the local policy:

- For every *outgoing communication*, the client's intended action is compared against the privileges of the client's role (access rights). If the check is successfully passed, the data is (optionally) encrypted with the administrator's public key. The whole message is then digitally signed with the client's private key and as such passed onto a network layer and sent to the recipient via open network.
- *Incoming communication*, which normally arrives encrypted with the client's public key and signed with the administrator's private key, is authenticated and decrypted. Subsequently, if the compliance with the local policy is confirmed, the data is delivered to the application. However, a large portion of the client-administrator communication is to do with group management, carrying the information that is not intended for the user itself. This is denoted with a split arrow in the Figure 13, where the appropriate data is used to update the security policy and/or group management-related information, and the notification on the success of the operation goes to the user.

Since there is an implicit client's trust in the administrator, the previous relationship between the two and the plain hierarchy limiting the scope of the communication, security enforcement is not as strict as in the case of peer-to-peer interactions within the CUG.

In order to initiate a particular peer-to-peer session, one member presents to the intended recipient a message containing the attribute certificate (AC) and its intended action. At the recipient's firewall instance, the credentials from the AC are used for obtaining the privileges associated, via role-matrix. Additionally, this initial negotiation is accompanied with the clients' public key certificates (PKC) issued by the administrator(s), allowing the communicating entities to authenticate each other. If the intended action is authorised, a recipient replies with the message containing its own AC for that group, augmented with the session key (encrypted

with the public key of the sender). If the first entity verifies recipient's certificates, a peer-to-peer session can commence. If the data is certified with the AC for the group that the recipient does not have clearance then the local host firewall will reject it.

The privileges retrieved from the role-matrix are cached for the session duration. Effectively, the local policy is augmented with the subset of CUG's authorisation privileges related to the appropriate client, therefore acting as a 'temporary' application level firewall. Subsequently, when the host receives message from the CUG peers, the decryption key is applied to the data and intended action compared against the cached privileges. Providing that the authorisation is confirmed, and that the intended action does not collide with the local policy, access is granted and the data is passed to the intended user's application layer unhindered. In the case of several concurrent sessions, corresponding authorisation data is retrieved as the separate 'temporary' firewall databases, and the appropriate database is consulted based on the referenced CUG_ID and member_ID. The encryption gives the additional assurance, since the incorrect mapping (of group/members' identities) will result in applying the wrong session key and producing meaningless plaintext.

For every interaction, per-message security checks are performed at a distributed firewall instance:

- For an *outgoing message* from the CUG member, the intended action is checked against the set of member's privileges, both within the CUG and local policy. If compliance is confirmed, the message is encrypted, accompanied with the reference to the CUG and sender's ID, and digitally signed with the sender's private key. It is then transmitted via the firewall and non-secure network to the recipient.
- For an *incoming message*, after packet stream is examined, the message is decrypted with the appropriate session key. Optionally, the sender's public key can be applied in order to check the digital signature[38]. After this, the sender's intended action is compared against the cached 'temporary' firewall database, and if acceptable the data is passed onto the application layer accessible to the user.

If at any point non-compliance with the security policy is detected, the communication is blocked, resulting in logging the event at both parties. This information is then passed to the appropriate administrators. Since the administrators to certain extent account for the actions of their clients, this could initiate negotiation process among them, with the CUG manager or Trusted Third Party (TTP) acting as an arbiter, aiming to resolve the conflict. This aspect of the architecture functionality goes beyond the scope of this thesis, and may form a direction for future work, which will be outlined in Section 7.3 Further Work.

[38] In some of the current applications, this is performed on a random basis.

4.3.6 Policy Updating

The updating of the security policy within a CUG environment is a continuous process that is happening on several planes:

- Updating of access rights related to the local and CUG policies, addressing the structural changes within the organisational / group policy.
- Updating of group membership and related lists of revoked certificates (both PKC and AC) in order to accommodate dynamic changes of the CUG security perimeters.
- Updating of default security rules that are enforced at each terminal, in order to counter discovered vulnerabilities or new attacks by modifying the filtering rules and / or updating the signatures and patterns governing the traffic monitoring.

Users can ask to have the policy updated but the decision is made by the administrator, which retains the responsibility for setting the local policy and approving the CUG policy, and for the programming of the client firewall rules. As with the initial policy deployment, the corresponding administrator / CUG manager keeps all the updates, whereas the client receives only the subset that applies to it.

The updates of local policy which are triggered by the organisational changes can be several. If the organisational policy is changed in the sense of restructuring of the role-matrix (i.e. modifying the set of privileges associated with a certain role) or as change of the security level (e.g. update of the restricted ports) this will require a major action by the administrator which then needs to communicate the updates to all of its registered clients, similar to the process of initial setup. If a client's public key certificate has expired, the administrator can re-use the client's public key and re-issue the PKC with new expiration date, notifying only the client concerned. If a client's role in the organisation (and related access rights) has changed, this update will affect only that client. However, the administrator may wish to contact CUG managers of the groups that client is currently member of, to re-assign the client's privileges in the CUG.

The updates of CUG policies are simpler, and affected by the changes in the role-matrix. If a role of a CUG member changes, it is re-issued a new attribute certificate (AC), while the old AC is revoked and all the CUG members are informed of that. They keep the certificate number in their CRLs (Certificate Revocation Lists) until it has expired. However, this does not affect the actual role-matrix distributed to the members at the time of their joining to a CUG. On the other hand, if the role-matrix itself is modified (i.e. the privileges associated with the certain roles), the CUG members need to be updated via their administrators. In this case, however, ACs of the members do not have to be changed.

The second group of updates addresses the changes in the group structure. Normally, it is expected this type to be the most frequently performed of all. Whenever members leave the group, its structure is changed and there is a number of privileges and attribute certificates that need to be revoked. In this case timely updates are essential, in order to minimize the chance for the expelled members to use their privileges within the CUG afterwards. This can be achieved either through triggered or periodical CUG manager's updates sent to the group, which will be discussed in more detail in the section 4.4 Description of Security Protocol, describing the signalling protocol. In a similar way, if a client is deregistered (or its public/private key pair has been compromised), it is the responsibility of the administrator to initiate the update, where this information will be passed to all the CUGs that client is/was a member of. This consuming operation can be minimized with the appropriate management, where the CUG membership will be limited to the period ending before the expiration of client's registration (i.e. validity of the PKC). However, both of these actions need to be taken only if the membership is revoked before the actual expiration time of the certificates. Therefore, if the member is excluded from the group in accordance with the CUG policy (i.e. based on the previously defined duration), the rest of the group will not need to be notified since the expired certificates cannot be used anyway. For the process of member's joining a group, such an update is not necessary – if a new member initiates the session exercising its legitimate rights, the other party can always seek the authentication at client's administrator.

During the runtime of the system, different types of network activity will be seen and the distributed firewalls may experience various non-legitimate interaction. Detection of the intrusions and malicious activity does not necessarily mean that they have been prevented. Report logs of these actions are submitted to the administrator for auditing. The information extracted may be used for the updating the default security rules, to protect other hosts from attacks that the client's firewall has seen, but they have not. Optionally, this type of information could be communicated among different administrator nodes, subject to an agreement (this may be more realistically confined between administrators of the same organisation).

Different types of policy updates are summarized in Table 2 and Table 3.

Table 2: Policy Updates Affecting a Single Client

Condition	**Update of**	**Performed by**
PKC expired	PKC	Administrator
Change of client's privileges	Access rights	Administrator
AC expired	AC	CUG Manager
Change of member's role	AC	CUG Manager

Table 3: Policy Updates Affecting a Number of Clients / CUGs

Condition	**Update of**	**Performed by**	**Applying to**
PKC revoked	PKC and PKC-CRL	Administrator	All the relevant CUGs
Change of local policy	Local role-matrix	Administrator	Organisation
Change of security level	Default security rules	Administrator	Organisation
AC revoked	AC and AC-CRL	CUG Manager	CUG
Change of CUG policy	CUG role-matrix	CUG Manager	CUG

4.4 Description of Security Protocol

This section describes details of signalling protocol that supports the CUG architecture and illustrates protocol functionalities through examples of more complex interactions. Group operations are described through sequence diagrams of message exchange, where security aspects of authentication, authorisation and (optionally) message encryption are taken into consideration. Table 4 lists the acronyms and notation used within this chapter.

Table 4: Acronyms and Notation Used for Protocol Description

Ad	Administrator
Man	CUG manager (same entity as administrator, but performing different role)
Cl	Client
Mem	CUG member (same entity as client, but within group)
CA	Certification Authority (Trusted Third Party)
M = {}	Message
enc*(k; m)*	Encryption of message m with secret key k
Px/Sx	Public and private key pair in a public-key encryption system
sig*(Sx; m)*	Digital signature on a message m using a private key Sx
h()	One-way hash function
$PC^{X}Y$	PKI authentication certificate PC of node Y, issued by authorised (and trusted) entity X (this can be either CA or administrator Ad)
$AC^{X}Y$	Attribute certificate AC for CUG authorisation of node Y, issued by authorised (and trusted) entity X (normally that would be CUG manager Man)
$S^{X,Y,Z}K$	Symmetric session key for interactions among X, Y, Z…, normally created by either of the parties involved
rand	Large random number: when included in the reply message it confirms to the sender that the original message has been received

4.4.1 Administrator – Client Messages (hierarchical)

All of the operations described in this section, apart of client's setup and deregistration, belong to the class of *CUG operations*. This means that they are directly related to supporting the communication and management of a particular group. On the other hand, operations of setup and deregistration are used to regulate a client's status in the organisational context and to provide basic support for the client's inclusion in the overall CUG environment.

4.4.1.1 Initial Setup of Client (Registration)

Setup of client's host is performed by the administrator and represents initial introduction of a client into the CUG environment. At setup, the administrator defines security policy of a client. The policy, that is compiled off-line and then delivered to the client, consists of:

- Authentication PKI certificate which identifies that a client belongs to the particular organisation. This certificate, being issued locally (by the administrator) enables the administrator to directly control client's presence in the system by defining expiration time (after which client will have to either re-apply or will be issued automatically with a new certificate). Normally, the validity period must not exceed the expiration time of administrator's certificate issued by CA ($PC^{CA}Ad$).
- Attribute certificate defining client's role and privileges within the company and (potential) interactions regarding CUG creation and joining.
- Firewall rules and patterns for traffic monitoring used in IDS. These are provided in a file that can be installed on top of existing software at client's host.

Ideally, this process should be performed manually – it corresponds to the opening of an account for a new employee of the company (in the case of human client), the installation of a new or an update of an existing service (at Application Service Provider), or similar. However, in nomadic environments this operation may have to be carried out remotely.

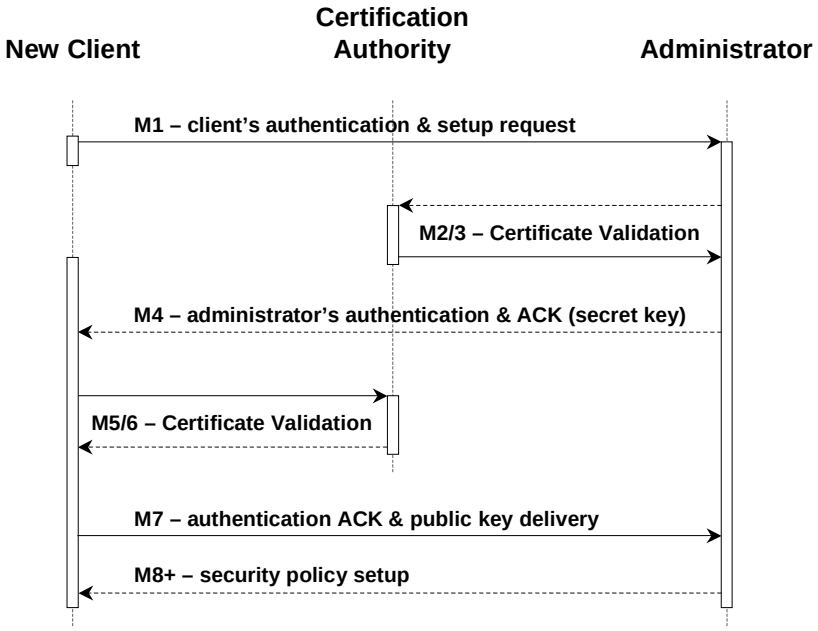


Figure 14: Initial setup of remote client

Initially, the client requests setup by contacting the allocated (or chosen) administrator:

M_1 = {Src = Cl, Des = Ad, PC^{CA}Cl, req, rand1, Sig(S_{Cl}; m_1)},

Where m_1 is hash function of the message used for non-repudiation and also to verify that the message has not been modified during transmission:

m_1 = h(PC^{CA}Cl, req, rand1)

The client presents its PKI certificate issued by trusted party (Certification Authority - CA) in order to confirm its identity. Upon receipt, administrator examines the certificate and the client's request. Optionally, the administrator can contact the CA (issuer of the certificate) who verifies the certificate's validity:

M_2 = {Src = Ad, Des = CA, val_req, PC^{CA}Cl, Sig(S_{Ad}; m_2)},

M_3 = {Src = CA, Des = Ad, val_ack, Sig(S_{CA}; m_3)},

Similar to before, the hash of the corresponding message is digitally signed with sender's public key; for message M_2, that is [39]:

m_2 = h(val_req, PC^{CA}Cl)

After the client's identity is confirmed and the request accepted, the administrator authenticates itself to the client:

M_4 = {Src = Ad, Des = Cl, PC^{CA}Ad, ack, rand1, rand2, enc (P_{Cl}; $S^{Ad,Cl}$K), Sig(S_{Ad}; m_4)},

A random value created by the client (rand1) is re-sent and accompanied by rand2 (created by the administrator). This provides the additional confidence that the response message directly relates to the request and that no message has been lost. In addition, the administrator generates a symmetric session key $S^{Ad,Cl}$K to be used for secret transfer of security policy. The key is encrypted with the client's public key P_{Cl}, and can be decrypted only with client's private key S_{Cl}, known only to the client.

Upon receipt, the client can contact CA to verify administrator's identity. This corresponds to the message pair M_5-M_6 from the Figure 14, and the process and message content is analogous to the message pair M_2-M_3. If the client accepts administrator's authentication, it creates public/private key pair P_{Cl}'/S_{Cl}', which will form the basis for issuing of client's 'local' authentication PKI certificate (PC^{Ad}Cl). As already explained, this certificate is created by client's dedicated administrator, allowing the administrator to control the client's presence in the system by defining the validity period of the certificate. Using the previously received symmetric key $S^{Ad,Cl}$K, the client encrypts its public key P_{Cl}' and replies to the administrator, including rand2 and newly generated rand3:

M_7 = {Src = Cl, Des = Ad, ack, rand2, rand3, enc ($S^{Ad,Cl}$K; P_{Cl}'), Sig(S_{Cl}; m_7)}

Upon receipt, the administrator derives the client's PKI certificate (PC^{Ad}Cl) and delivers it together with the security policy to the client. This data is encrypted with the previously agreed symmetric key:

[39] The same notation is used throughout this chapter. Unless otherwise stated, the signature is always applied to the hash of the whole message. Therefore, display of the hash will normally be avoided for the reasons of clarity.

M_8 = {Src = Ad, Des = Cl, rand3, data, Sig(S_{Ad}; m_8)},

Where *data* denotes the policy consisting of authentication certificate, security policy rules (access rights and firewall/IDS rules):

data = $PC^{Ad}Cl$, enc ($S^{Ad,Cl}K$; rules#, Seq)

Seq is used to protect from replay-attack: it confirms that all the messages are received and in order. A malicious entity could try to capture and re-send some of the messages, therefore trying to modify the corresponding policy. However, *Seq* is encrypted with a key known only to the administrator and the client. Therefore, bogus messages will be detected as replayed since a malicious entity is not able to modify the encrypted part. The symmetric key $S^{Ad,Cl}K$ is never re-used but is destroyed after the transfer of the policy is completed.

A procedure similar to setup takes place when the certificate and/or policy rules of an existing client are to be updated. Again, this should ideally be performed manually, but in the case of a remote client the process as above is executed. The only distinction with the initial setup is that the old certificate is being revoked and replaced by the new one, and the policy rules maybe updated.

4.4.1.2 Deregistering a Client

Deregistration of a client can be either performed at the client's request or decided by the administrator (e.g. if a human client is no longer an employee of the organisation or if computation resources are being replaced by new ones).

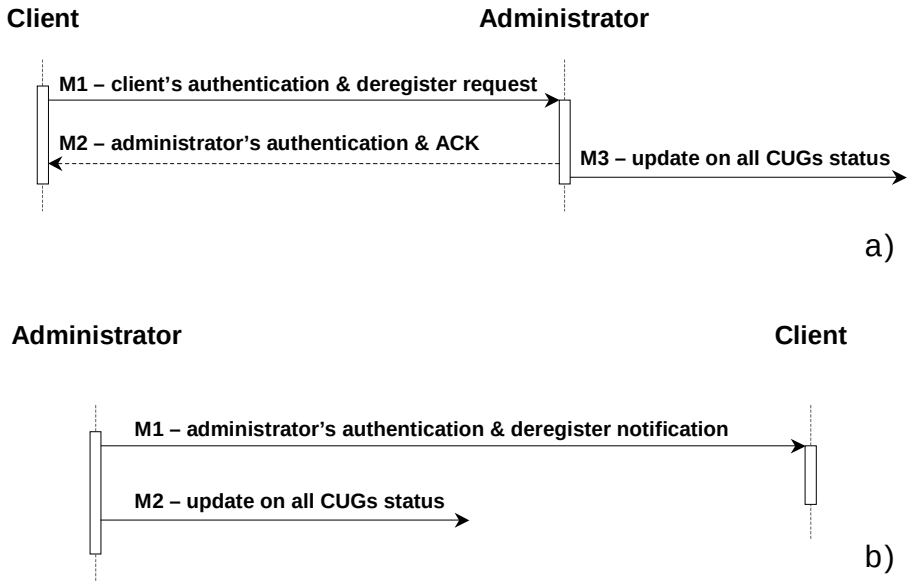


Figure 15: Deregistration of client: a) requested; b) forced

Figure 15 illustrates these two operations. An example of a client requesting deregistration would be if a client wants to change commercial service provider due to not being satisfied with the existing service delivery. In such a case, the client sends a deregister request to the administrator, authenticating itself with the PKI certificate granted by the administrator at setup:

M_1 = {Src = Cl, Des = Ad, PC^{Ad}Cl, req, rand1, Sig(S_{Cl}, m_1)}

If the administrator approves this, it sends an acknowledge message, notifying the client that its request has been accepted:

M_2 = {Src = Ad, Des = Cl, PC^{CA}Ad, notify, rand1, rand2, Sig(S_{Ad}, m_2)}

This is an important action, since the time spent with the provider could be used as a criterion for billing purposes, or the client can decline any liability for the actions being accounted for after this time.

Finally, the administrator has to update the status of all the groups the client was member of by notifying current CUGs members that the client has left. This is a complex operation, especially if inter-administrator communication is needed. This thesis considers several alternatives, described separately, in Section 4.4.4.2 Group Membership Revocation.

Figure 15b illustrates the deregistration process if initiated by the administrator. It consists simply of the administrator's notification to the client:

M_1 = {Src = Ad, Des = Cl, PC^{CA}Ad, ack, Sig(S_{Ad}, m_1)}

Subsequently, the administrator has to update the group status. The process is analogous to the one described in Figure 15a, only differing in that the initial client's request is avoided.

4.4.1.3 CUG Creation

Normally, the creation of the group is performed at the client's initiative. The client sends a request to its administrator:

M_1 = {Src = Cl, Des = Man, PC^{Ad}Cl, req, rand1, Sig(S_{Cl}, m_1)}

The message includes the client's authentication certificate, and *req* indicates the client's requirements to assist the administrator in determining whether to approve the request and what type of the group to choose. If the request is accepted, the administrator replies to the client:

M_2 = {Src = Man, Des = Cl, PC^{CA}Man, ack, rand1, rand2, enc (P_{Cl}; AC^{Man}Cl, rules#), Sig(S_{Man}, m_2)}

Where the notation is as follows:

- AC^{Man}Cl is the group-specific certificate with the embedded group identity, the client's identity and attributes defining the client's role in the group.
- rules# comprises the full policy of the group; once put in the place, it enables the mapping of attributes into authorisation permissions during peer-to-peer communication between group members.

Also, note that the CUG manager and administrator may be actually the same entity. The notation used in the messages above differs from the previous section in order to make a distinction between the two roles. Therefore: $PC^{CA}Man = PC^{CA}Ad$ and $P_{Ad}/S_{Ad} = P_{Man}/S_{Man}$.

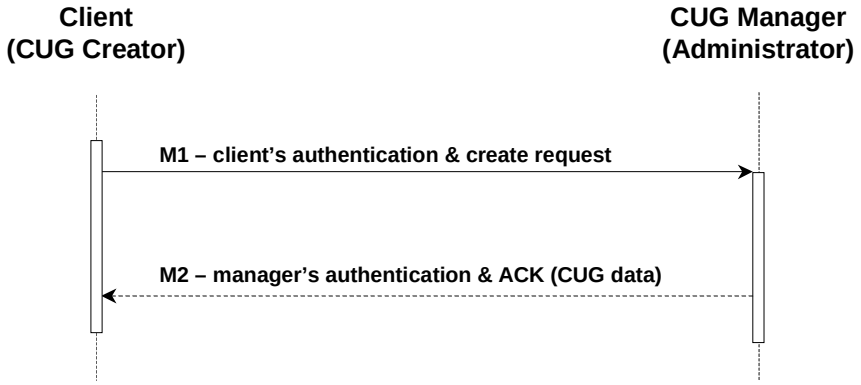


Figure 16: Creation of a CUG

It is important here to note that the authentication certificate can be transmitted in plaintext, whereas the attribute certificate may need to be encrypted. The motivation for this is as follows: the authentication certificate carries the public key of an entity which by definition should be publicly accessible to anyone who may need it. However, the attribute certificate carries information that reflects the privileges of the specific member. As such, it can be used by a malicious entity in two straightforward ways:

- If communication of the particular member is tampered with over the whole set of groups it is member of, this information taken together can form member's 'profile' (privacy violation) [41].
- If communication of the particular CUG manager with various members is tampered with, the information extracted could be compiled together to roughly reflect the CUG policy (industrial espionage)[40].

In such a way, by encrypting only the attribute certificate and security rules, the amount of data that needs to be encrypted is minimized, whilst security is not compromised.

4.4.1.4 Joining CUG

The process of joining a group is effectively similar to group creation. The client sends a joining request to the CUG manager:

$$M_1 = \{Src = Cl, Des = Man, PC^{Ad}Cl, req, rand1, Sig(S_{Cl}, m_1)\}$$

[40] Exactly the same applies for the CUG manager and particular group.

If the request is approved, the CUG manager replies with the appropriate group certificate and the policy for the group:

M_2 = {Src = Man, Des = Cl, PC^{CA}Man, ack, rand1, rand2, enc (P_{Cl}; AC^{Man}Cl, rules#, list), Sig(S_{Man}, m_2)}

When a new client joins the group, the notification is not sent to the current members. This action is avoided due to scalability of the protocol. However, the list of current group members is sent to the new member (denoted as *list* within the encrypted part of the message), together with the group certificate and policy. This enables the new member to contact anyone in the group. The legitimacy of the new member is confirmed at the establishment of a peer-to-peer session through the group certificate that is signed by the CUG manager. Consequently, the recipient will update its list of group members.

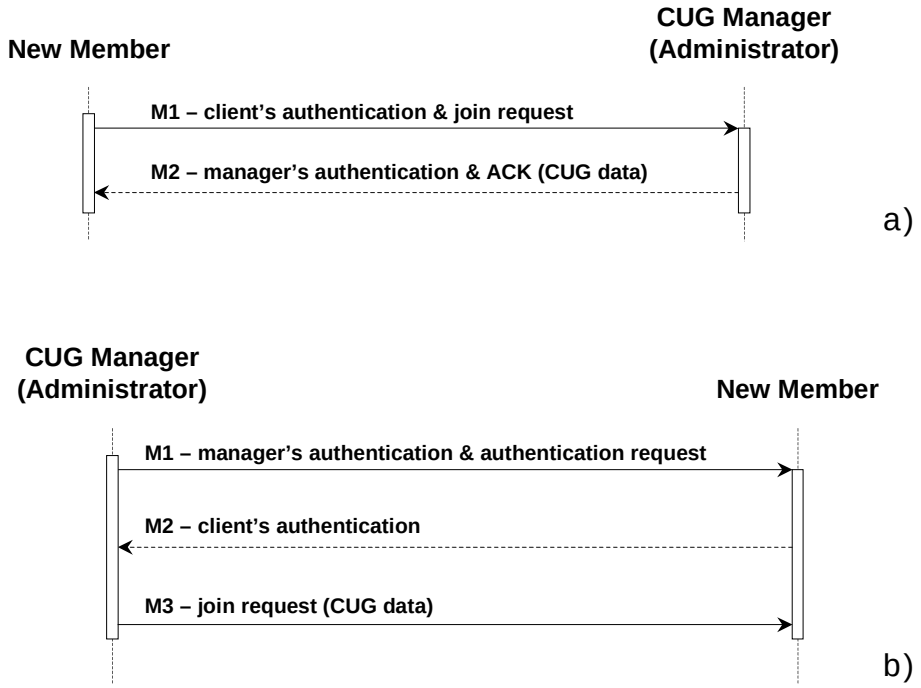


Figure 17: Client joining CUG: a) requested; b) appointed

Optionally the client can be appointed by its administrator to join a group. This process is illustrated in Figure 17b. The appropriate policy is delivered to the client only upon successful mutual authentication. The corresponding messages are listed below:

M_1 = {Src = Man, Des = Cl, PC^{CA}Man, req, rand1, Sig(S_{Man}, m_1)}

M_2 = {Src = Cl, Des = Man, PC^{Ad}Cl, ack, rand1, rand2, Sig(S_{Cl}, m_2)}

M_3 = {Src = Man, Des = Cl, ack2, rand1, rand2, enc (P_{Cl}; AC^{Man}Cl, rules#, list), Sig(S_{Man}, m_3)}

4.4.1.5 Leaving CUG

Depending on the scope of the group, members are free to leave at any point in time or upon completing their appointed task. This is normally performed through the exchange of a member's request and manager's acknowledge reply message:

M_1 = {Src = Mem, Des = Man, PC^{Ad}Cl, req, rand1, Sig(S_{Cl}, m_1)}

M_2 = {Src = Man, Des = Mem, PC^{CA}Man, ack, rand1, rand2, Sig(S_{Man}, m_2)}

Similar to the operation of the client's deregistration, it is the manager's responsibility to update the group status. The difference is that in this case the update needs to be performed only within the specific group that member has left. This operation will be addressed in detail in Section 4.4.4.2 Group Membership Revocation.

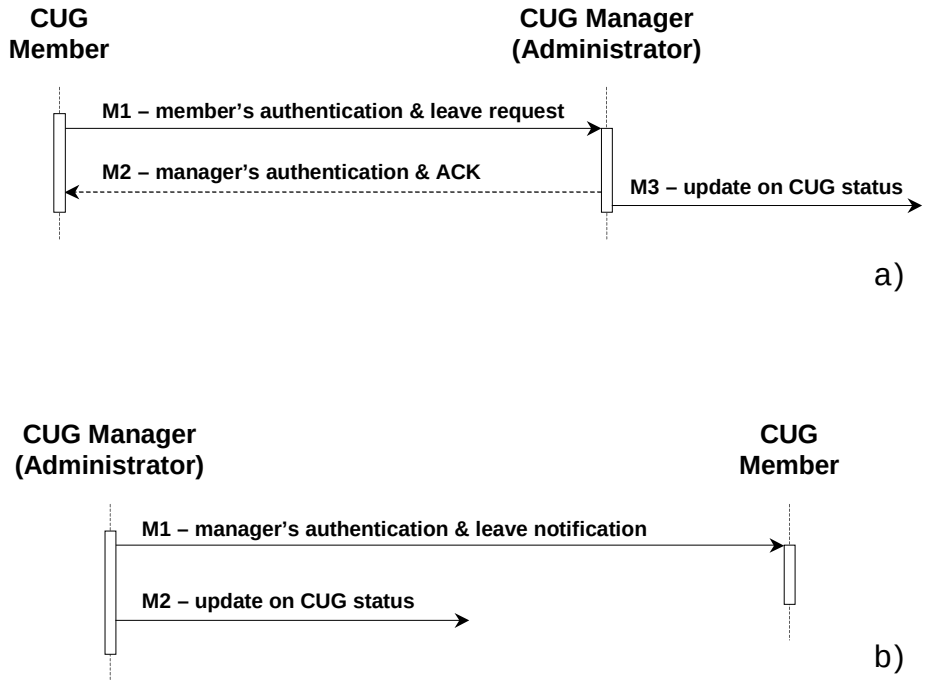


Figure 18: Client leaving CUG: a) requested; b) expulsion

In addition to this, the CUG manager reserves the right to expel a member from the group at any point in time. This action will normally be taken if the member breaches group policy. It consists of the manager's notification to the member concerned (and the group status update):

M_1 = {Src = Man, Des = Mem, PC^{CA}Man, notify, Sig(S_{Man}, m_1)}

4.4.1.6 CUG Removal

Once created, a group exists as long as there is at least one member present. When the last member leaves the group (or is expelled) the group is naturally removed by the CUG manager. Therefore, the message exchange for this operation corresponds to the process of a member leaving the group, as described in Section 4.4.1.5 Leaving CUG. The only distinction is that the

CUG manager does not perform the update on the group status (the last message in Figure 18a and b) since there is no one left to notify.

4.4.1.7 CUG Broadcast (Administrator's Update Info)

This operation is a single message broadcasted by the CUG manager to the valid CUG members. Depending on the particular requirements, the CUG broadcast can be used by the CUG manager to notify CUG members of a new client joining or a member leaving the group, or other information relevant to the particular CUG. In addition, this message would be used for advertising administrator services or groups of interest that it is managing, broadcasting general information, such as company news, etc, in which case it originates from the dedicated administrator and is sent to the local clients only.

The most critical application of this operation is providing a mechanism for revocation of certificates and updating of CUG status after a member has left or has been expelled from a group. This is also applied after the client's deregistration, assuming that it has been involved in a CUG(s) at the time. This operation is discussed in more detail in Section 4.4.4.2 Group Membership Revocation, which also describes additional methods for the certificate revocation involving CUG members.

4.4.2 Client – Client Messages (peer-to-peer)

Operations described in this section belong to the class of *CUG operations*, providing peer-to-peer interactions of the clients in order to support data exchange and CUG management.

4.4.2.1 Authentication of CUG Peers (Session Establishment)

Once admitted to a group, a client can engage in peer-to-peer communication with other CUG members. Point-to-point is the simplest form of group interaction, presented in Figure 19a. The initiator (Alice) sends a request message authenticating itself:

$M_1 = \{Src = Alice, Des = Bob, PC^{Ad}A, req, rand1, Sig(S_A, m_1)\}$

If the session is accepted, Bob generates a symmetric session key $S^{A,B}K$ (to be used for the transfer of data) and delivers it securely to Alice [41]. In addition, Bob presents his authentication certificate $PC^{Ad}B$ that can be used by Alice to check Bob's signature:

$M_2 = \{Src = Bob, Des = Alice, PC^{Ad}B, ack, rand1, rand2, enc(P_A; S^{A,B}K), Sig(S_B, m_2)\}$

The protocol is also able to support more complex multicast interactions, such as given in Figure 19b. Although there are a number of algorithms for multicast key management, the approach here takes the modification of '*N root/leaf pairwise keys*' for key distribution [91].

[41] Authentication certificate $PC^{Ad}A$ carries the public key of Alice, which Bob uses for encrypting the session key, and which can be decrypted only by Alice's private key P_A, known only to Alice.

The initiator of the session (Mem_1) is referred to as a *root*. It generates a list of the participants (*leaves*) and sends it with the requests to each of them (an example is for the recipient Mem_N):

M_1 = {Src = Mem_1, Des = Mem_N, $PC^{Ad}Mem_1$, M_list, req, rand1, Sig(S_{Mem1}, m_1)}

Where the list of participants is: *M_list* = mem_2,….,mem_N.

The leaves respond with an acceptance/rejection message. An acceptance by member Mem_N takes the form:

M_2 = {Src = Mem_N, Des = Mem_1, $PC^{Ad}Mem_N$, ack, rand1, rand(N), Sig(S_{Memi}, m_2)}

In such a way, the root creates the record of the public keys of the members who accepted the session, and exchanges the encryption key only with them. This approach contributes to scalability since the most time consuming operation, public-key encryption, is minimized. In the meantime, the root generates a symmetric session key $S^{1,..,N}K$ and the updated list of participants *M_list**, and encrypts it with the appropriate public key and sends it to the corresponding member(s):

M_3 = {Src = Mem_1, Des = Mem_N, ack, rand(N), rand1', enc(P_{MemN}; $S^{1,...N}K$, M_list*), Sig(S_{Mem1}, m_3)}

The messages are accompanied with the appropriate random value previously received (rand(N)), and since the list of participants is encrypted its contents can only be understood by those involved in the session.

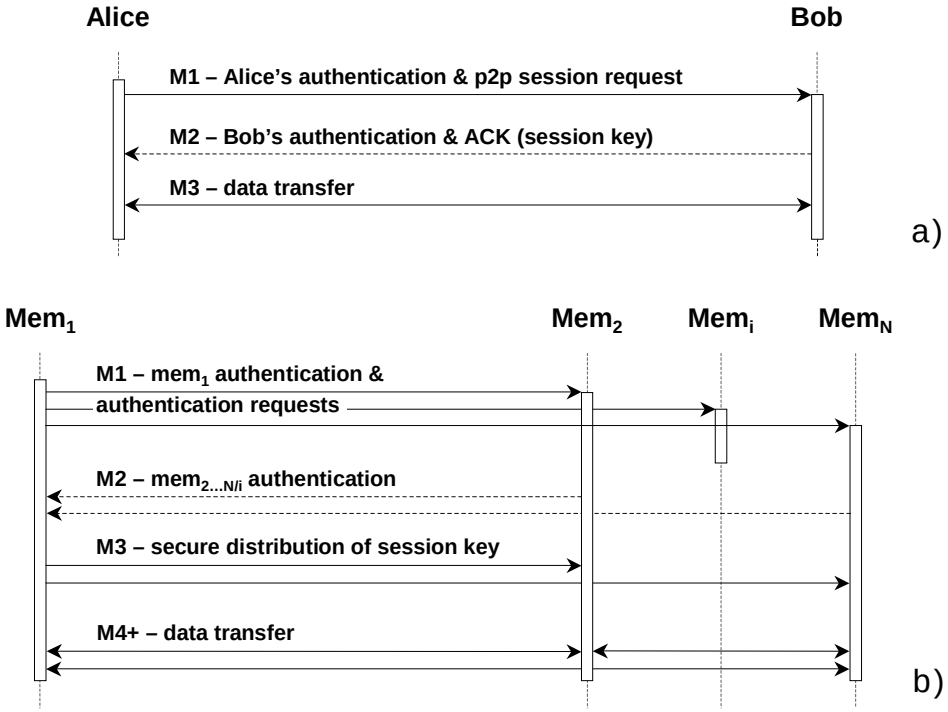


Figure 19: CUG session establishment and data transfer: a) unicast; b) multicast

The 'N root/leaf pairwise keys' algorithm is particularly suitable for the CUG architecture since it is non-hierarchical, so any CUG member can initiate the session. The distinction between this

one and the original key distribution scheme is that the approach described in [91] involves an additional step prior to the distribution of the session key. It consists of point-to-point interaction between the root and each of the leaves when keys for the encryption of the session key are exchanged [42]. This effectively corresponds to the formation of a structure similar to a group, where the root has a role that is similar to that of an administrator. Within the CUG environment this is not needed since the formation of the group is provided through other means (i.e. certificates issued by the CUG manager).

4.4.2.2 Data Transfer

Once the session key is in place, the actual data transfer is straightforward, and involves message encryption and presenting the attribute (group) certificate to enact the appropriate privileges at the recipient. This operation is the same for unicast or multicast sessions, and is elaborated via message M_3 from Figure 19a, sent from Alice to Bob upon the session key exchange:

M_3 = {Src = Alice, Des =Bob, rand2, rand3, enc($S^{A,B}K$; $AC^{Man}A$, data, Seq), Sig(S_A, m_3)}

The recipient (Bob) decrypts the data using the session key $S^{A,B}K$ and retrieves the set of Alice's privileges based on the attributes embedded in the group certificate of Alice ($AC^{Man}A$). The message content is compared against the Alice's privileges (Is she allowed to send it?) and against Bob's privileges for that particular CUG. (Is he allowed to receive and process this type of information?)

The same mechanism is happening at Alice's host with the message $M_{3'}$ sent by Bob.

M_3 = {Src = Bob, Des =Alice, rand3, rand4, enc($S^{A,B}K$; $AC^{Man}B$, data, Seq), Sig(S_B, $m_{3'}$)}

It needs to be noted here that the session establishment protocol described in the previous section does not authorise the entities to any action, but only authenticates their identities allowing subsequent authorisation certificates to be verified. If Alice would present an authorisation certificate for the wrong group, the access would not be allowed. This also means that initial communication is possible even if clients are not in the same group, and the group membership is confirmed only subsequently. This was a design choice in order to enable encrypted transmission of authorisation certificates.

4.4.2.3 Optional Client's Functionalities

This operation employs a single message broadcasted by the appointed CUG member to the rest of the CUG. As part of the member's role in the group, the CUG manager can allow certain clients to act as 'appointed members'. This class of member has privileges (and responsibility) to replicate specific type of message received by the CUG manager and to deliver them onwards to the CUG members.

[42] Therefore, for N participants there will be N-1 so called KEKs (key encryption keys).

This operation is an optional functionality of the CUG-supporting protocol. The motivation for introducing it is to decrease the burden on the CUG manager in the case when manager is responsible for large number of groups with a considerable population each[43]. By delegating broadcast responsibility to the appointed group member, the CUG manager effectively compiles and sends a single message only. It is then the appointed member who multiplies and re-sends this message to the whole CUG. However, a single client is allowed to take the 'appointed' role only in one of the groups it participates in, which is decided by the dedicated administrator. Also, since this delegation of responsibility can decrease the security of the group, each CUG is limited to one 'appointed member' only. The entity that this role is delegated to is determined by the administrator, and the criteria (depending on the nature and requirements of the CUG) can be based upon the administrator's trust in the client, the type of the network connection, the CUG creator, etc.

Similar to administrator's broadcast, the most critical application of this operation is providing a mechanism for revocation of certificates and updating of CUG status after a member has left or has been expelled from the group. It is addressed in more detail in Section 4.4.4.2 Group Membership Revocation.

4.4.3 Administrator - Administrator Messages (peer-to-peer)

The operations described in this section belong to the class of *CUG operations*, providing peer-to-peer interactions between the CUG manager and administrator in order to support management of remote CUGs.

Normally, a client is prevented from engaging in a remote group directly, by not being permitted to contact the CUG manager. This communication has to be endorsed by the client's administrator who acts as an intermediary, intercepting and approving messages between its client and the remote CUG manager.

4.4.3.1 Establishment of Initial Relationship for Remote CUG Management

The inter-administrator communication is triggered by client's request to join remote group:

M_1 = {Src = Cl, Des = Ad, PC^{Ad}Cl, req, rand1', $Sig(S_{Cl}, m_1)$}

The field *req* carries information about the group the client wants to join. The administrator examines the request, and if it is approved the client's message needs to be forwarded to the remote CUG manager. However, the administrator and the CUG manager may not have prior knowledge of each other. For that reason, an initial relationship has to be established. The

[43] This functinality is envisaged as the enhancement of the basic version of protocol, after protocol performance evaluation.

process is presented in Figure 20a. It involves mutual authentication, using public key certificates issued by certification authority (CA) trusted by both parties:

M_2 = {Src = Ad, Des = Man, PC^{CA}Ad, req, rand1, Sig(S_{Ad}, m_2)}

In response, the CUG manager authenticates itself with PC^{CA}Man and generates a secret key that will be used for transfer of group details (including group certificate) and for all future updates concerning the group status:

M_3 = {Src = Man, Des = Ad, PC^{CA}Man, ack, rand1, rand2, enc(P_{Ad}; $S^{Ad,Man}$K), Sig(S_{Man}, m_3)}

The key $S^{Ad,Man}$K is known only to the administrator and the CUG manager and remains valid for the duration of client's participation in the remote CUG. Either the CUG manager or administrator reserve the right to change the key (periodically or on as-needed basis) in order to maintain security. This is performed in a manner similar to the initial key agreement. In addition, the same key is used for other administrator-manager interactions; for example, if another administrator's client wants to join the same or any other group hosted by the CUG manager, or vice versa: for example, when a group is hosted by the administrator, and the client wishing to join originally resides with the CUG manager entity [44].

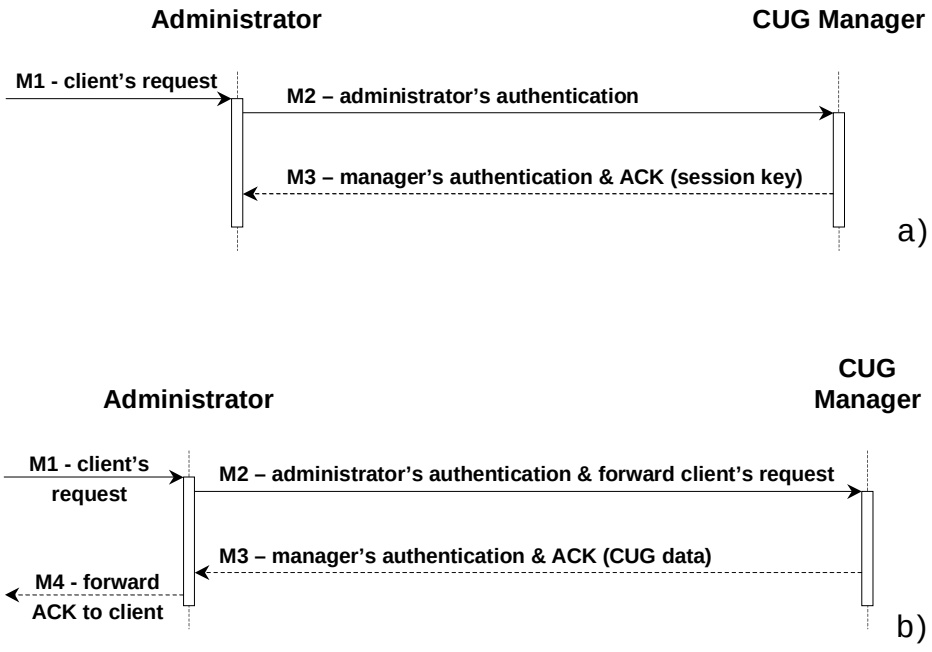


Figure 20: Peer-to-peer interactions of administration nodes: a) initial authentication; b) message interception

[44] Of course, the latter case assumes that the roles of administrator and manager nodes are now swapped. The change of terminology is omitted only to avoid confusion.

4.4.3.2 Message Interception

Once authentication of the administration nodes and key exchange have been performed, the client's request is forwarded to the CUG manager. This is presented in Figure 20b, which is a continuation of the upper part of the figure (the authentication process is omitted for presentation purposes). Upon the receipt of the client's request:

M_1 = {Src = Cl, Des = Ad, PC^{Ad}Cl, req, rand1', Sig(S_{Cl}, m_1)}

(and mutual authentication), the administrator examines the client's request and (if approved) forwards it to the CUG manager:

M_2 = {Src = Ad, Des = Man, req', rand2, Sig(S_{Ad}, m_2)}

Where *req'* contains the original request *req* and reference to the client's ID. If the request is accepted, the CUG manager replies to the administrator:

M_3 = {Src = Man, Des = Ad, ack, rand2, rand3, enc($S^{Ad,Man}$K; AC^{Man}Cl, rules#, list), Sig(S_{Man}, m_3)}

AC^{Man}Cl is a group-specific certificate, *rules#* comprises the full policy of the group and *list* is the list of current group members. Then, the administrator decrypts the message (AC) and reviews client's privileges in order to resolve any conflicts that may occur between local policy and CUG policy. Assuming that AC^{Man}Cl is acceptable, the administrator encrypts AC with the client's public key P_{Cl} and delivers it to client:

M_4 = {Src = Ad, Des = Cl, PC^{CA}Ad, ack, rand1', enc (P_{Cl}; AC^{Man}Cl, rules#, list), Sig(S_{Ad}, m_4)}

The administrator authenticates itself with PC^{CA}Ad, and includes rand1 in the reply message, therefore giving confidence to the client that its request was processed by the appropriate entity. As before (Section 4.4.1.4 Joining CUG) the information received is sufficient for the client to engage into peer-to-peer CUG communication (as described in Section 4.4.2 Client – Client Messages (peer-to-peer)).

4.4.4 Examples of Protocol Functionalities

This section illustrates protocol functionalities through examples of more complex interactions. The most frequent operations within the CUG framework are members' joining and leaving a group. The CUG architecture is developed to support a dynamic environment where it is expected that a large portion of these interactions will be performed across organisational boundaries. The joining of clients to a remote group, as well as clients' leaving a group (and the related process of certificate revocation and group updates) consist of a number of basic steps explained previously. In addition, a scenario describing the discovery of a hostile member is illustrated through the set of actions normally taken.

4.4.4.1 Remote CUG Joining and P2P Session Establishment

The basic process of client's joining a CUG has been demonstrated in Section 4.4.1.4 Joining CUG for the case of a local group. The process of joining a remote group is effectively similar, though it incorporates inter-administrator communication (described in Section 4.4.3 Administrator - Administrator Messages (peer-to-peer)), in addition. The example elaborated here brings the two aspects of the protocol together, following the message exchange from the client's joining request until the successful establishment of a peer-to-peer CUG session, under the following assumptions:

- The client joining a group is already registered with its local administrator.
- A previous relationship between the administrator and CUG manager does not exist, therefore they need to establish trust (through mutual authentication).
- The root certification authority CA is the same for both organisations that the administrator and the CUG manager belong to. However, if the organisations are established with different CAs, cross-validation of the administrators' certificates needs to be performed. Prior to this, trust in the Certification Authorities needs to be established and their public keys obtained by the administrators through some secure and authenticated off-line means of communication.
- The peer-to-peer CUG session is illustrated via the unicast session example. However, the process of multicast session, described in Section 4.4.2 Client – Client Messages (peer-to-peer), can be applied in a straightforward way, and is not influenced by the joining operation.
- Referring to the notation from Figure 21, the *Administrator* and *CUG manager* are respectively local security administrators of the *Client* and the *CUG member*.

Figure 21 depicts the whole process, initiated by the client through the join request message:

M_1 = {Src = Cl, Des = Ad, PC^{Ad}Cl, req, rand1', $Sig(S_{Cl}, m_1)$}

The client authenticates itself at the administrator via the PC^{Ad}Cl certificate, granted by the same entity at registration. The request field can either refer to the specific CUG, or to a type of service that the client wants to obtain (in which case the administrator can identify and allocate the most appropriate CUG). If the authentication and the request are approved, the administrator contacts CUG manager for mutual authentication and to establish the initial relationship:

M_2 = {Src = Ad, Des = Man, PC^{CA}Ad, req, rand1, $Sig(S_{Ad}, m_2)$}

Upon receipt, the CUG manager (optionally) contacts the Certification Authority in order to verify the administrator's certificate PC^{CA}Ad:

M_3 = {Src = Man, Des = CA, PC^{CA}Man, req_PC^{CA}Ad, $Sig(S_{Man}, m_3)$}

Which is acknowledged by the CA:

M_4 = {Src = CA, Des = Man, ack_PC^{CA}Ad, $Sig(S_{CA}, m_4)$}

If this has been performed satisfactorily, the CUG manager creates a symmetric key $S^{Ad,Man}K$ and delivers it to the administrator, encrypted with the administrator's public key P_{Ad}:

M_5 = {Src = Man, Des = Ad, PC^{CA}Man, ack, rand1, rand2, enc(P_{Ad}; $S^{Ad,Man}K$), Sig(S_{Man}, m_5)}

The message contains the public key certificate of the CUG manager, enabling the administrator to authenticate the manager at the CA (the messages pair M_6/M_7 in Figure 21 is effectively the same as M_3/M_4, only the originator is different). This ends this phase of the authentication: public keys are temporarily stored to check the signature until the joining process is completed, and the encryption key $S^{Ad,Man}K$ is kept by both of the parties as long as client remains a member of the CUG[45].

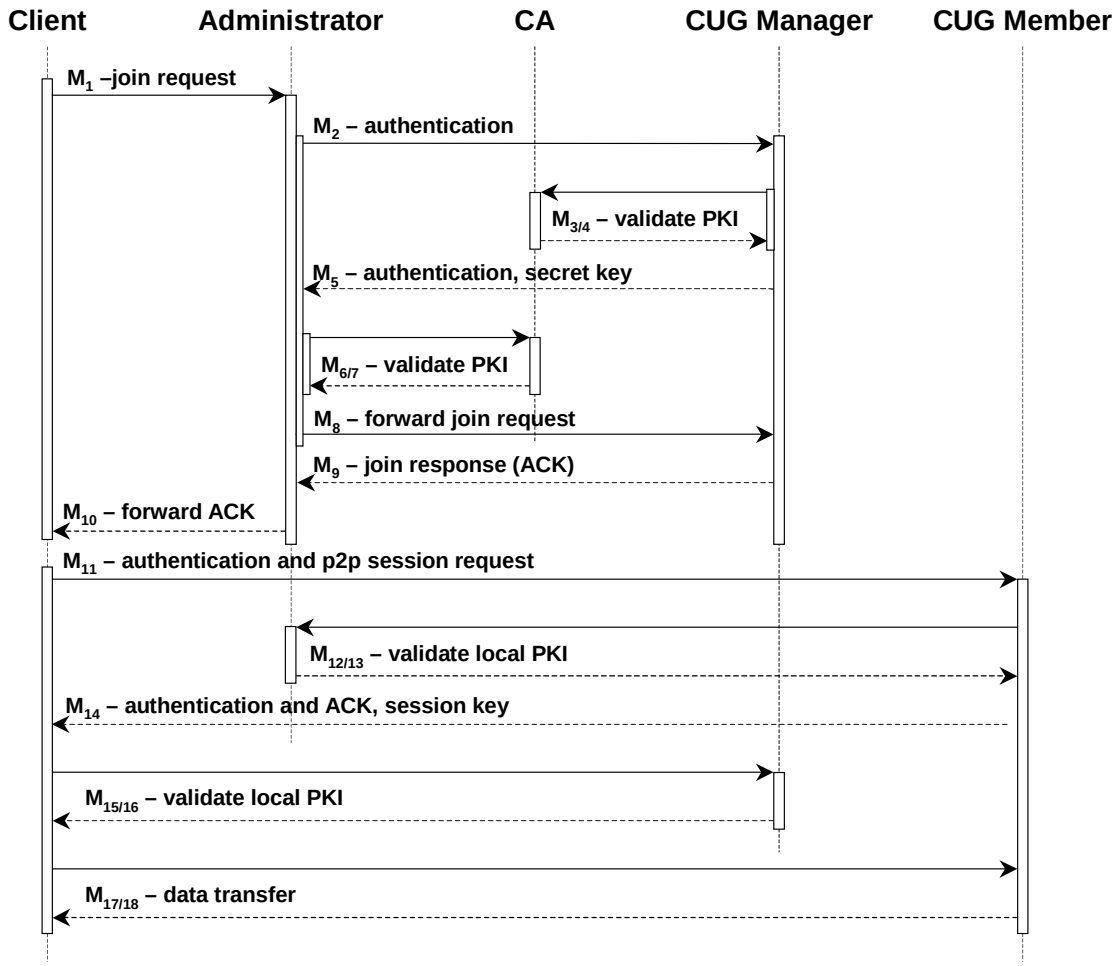


Figure 21: Joining and P2P Session in Remote CUG

Next, the administrator forwards the client's request to the CUG manager:

M_8 = {Src = Ad, Des = Man, req', rand2, rand3, enc($S^{Ad,Man}K$; data1), Sig(S_{Ad}, m_8)}

[45] As already mentioned in section 4.4.3 Administrator - Administrator Messages (peer-to-peer), this key is periodically updated.

Where the original request is augmented with the clientID (*req'*). If the CUG manager accepts the request, it defines the privileges of the new CUG member (via attributes within the group certificate $AC^{Man}Cl$), and compiles the CUG policy rules (*rules#*) and the list of current CUG members (*list*). This data is encrypted with the inter-administrator key and delivered in the acknowledgement message to the administrator:

M_9 = {Src = Man, Des = Ad, ack, rand3, rand4, enc($S^{Ad,Man}K$; $AC^{Man}Cl$, rules#, list), $Sig(S_{Man}, m_9)$}

Upon receipt, the administrator decrypts and examines the CUG-related data, and if the terms are accepted it forwards it to its client (this time encrypted with the client's public key)[46]:

M_{10} = {Src = Ad, Des = Cl, $PC^{CA}Ad$, ack, rand1', enc (P_{Cl}; $AC^{Man}Cl$, rules#, list), $Sig(S_{Ad}, m_{10})$}

This concludes the process of the client's joining which, now that it is a member of the CUG, can start peer-to-peer communication with the rest of the group. As described in Section 4.4.2 Client – Client Messages (peer-to-peer), the authentication of the peers is performed before a session can commence. One of the members (named the 'client' in Figure 21) sends the session request, authenticating itself via a public-key certificate:

M_{11} = {Src = Cl, Des = Mem, $PC^{Ad}Cl$, req, rand1'', $Sig(S_{Cl}, m_{11})$}

Optionally, the recipient may wish to verify the certificate, for which purpose the issuing authority needs to be contacted. In the example from Figure 21 and the related message M_{11}, the issuing authority is the administrator. Therefore, the CUG member contacts the administrator node requesting certificate validation:

M_{12} = {Src = Mem, Des = Ad, $PC^{Man}Mem$, $val\_PC^{Ad}Cl$, $Sig(S_{Mem}, m_{12})$}

Which is acknowledged:

M_{13} = {Src = Ad, Des = Mem, $ack\_PC^{Ad}Cl$, $Sig(S_{Ad}, m_{13})$}

Next, the CUG member creates the session key and delivers it securely to the client encrypted with client's public key.

M_{14} = {Src = Mem, Des = Cl, $PC^{Man}Mem$, ack, rand1'', rand2'', enc(P_{Cl}; $S^{Cl,Mem}K$), $Sig(S_{Mem}, m_{14})$}

In addition, it presents its public key certificate, which can be validated by the interaction between the client and CUG manager (note that, regarding this particular interaction, the CUG manager acts in the role of the local administrator):

M_{15} = {Src = Cl, Des = Man, $PC^{Ad}Cl$, $val\_PC^{Man}Mem$, $Sig(S_{Cl}, m_{15})$}

M_{16} = {Src = Man, Des = Cl, $ack\_PC^{Man}Mem$, $Sig(S_{Man}, m_{16})$}

This may seem to oppose to the previous statement that the client is not allowed to contact a manager of a remote CUG directly. However, regarding certificate validation, the CUG manager acts as the certification authority (CA), and not as the entity managing the group.

[46] However, if the policy is not accepted, this would either trigger re-negotiation process between the administrator and the CUG manager, where revised certificate will be presented by the CUG manager.

Normally, the issuer of the certificate verifies it, which is the local administrator in the CUG architecture.

Finally, the session can be established – the messages exchanged carry the attribute certificate and the useful data in its encrypted part. In addition, the presence of the random numbers and the sequence identifier protect the message session from the replay attacks and assure the proper order of the messages received:

M_{17} = {Src = Cl, Des = Mem, rand2'', rand3'', enc($S^{Cl,Mem}K$; $AC^{Man}Cl$, data, Seq), Sig(S_{Cl}, m_{17})}

M_{18} = {Src = Mem, Des = Cl, rand3'', rand4'', enc($S^{Cl,Mem}K$; $AC^{Man}Mem$, data, Seq), Sig(S_{Mem}, m_{18})}

4.4.4.2 Group Membership Revocation

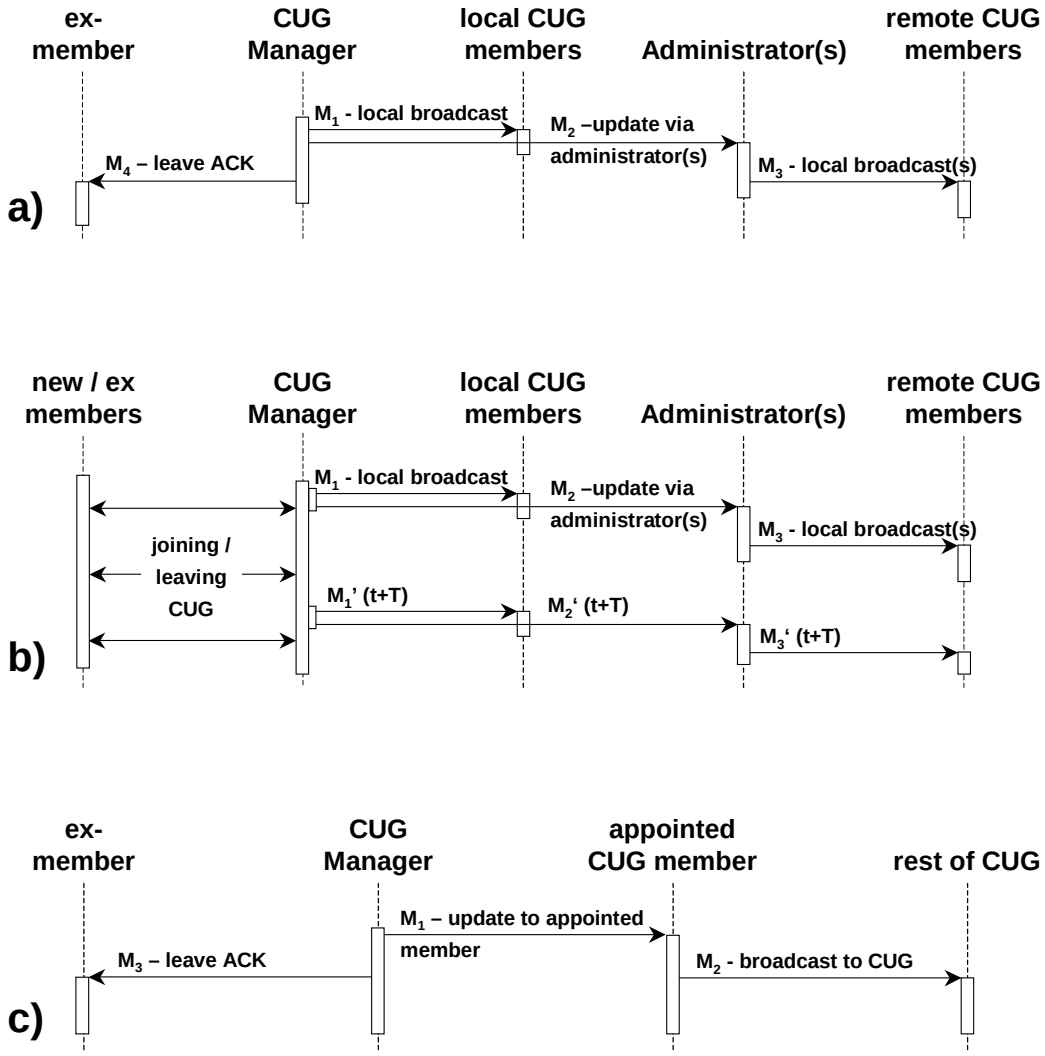


Figure 22: Membership revocation mechanisms: a) instant administrator's update; b) periodic administrator's update; c) update via appointed member

One of the aspects of enforcing security perimeters of the groups is through updating the list of revoked certificates and the group membership list when a member leaves the group voluntarily or through expulsion. In a dynamic CUG environment, timely updates of lists and its delivery to the valid group members is essential. Depending on the dynamics of the system and the security needed, this is always a trade-off between the system performance and security. This research considers three schemes for the revocation of clients' privileges within dynamic collaborative groups. Although different in nature, the schemes are directly supported by the protocol functionalities described in the previous sections.

[a] Instant manager's update

Directly upon a member's removal, the group manager updates the lists of valid group certificates and group members, and delivers it directly to the rest of the group. For the remote group members (i.e. associated with a local administrator different from the group manager), this information has to be passed via their local administrator(s). From a security viewpoint, this approach is advantageous, since the new information is (instantly) delivered to all valid group members directly by the group manager (who is implicitly trusted by its members). However, depending on the dynamics of the group membership, this could represent a significant performance problem since the whole burden is on the group manager.

Simultaneously, the group manager compiles and distributes three types of messages, targeting different parts of the CUG population: local members, remote members and the 'ex-member'.

Message M1 is broadcasted to the local CUG members, informing them about the member that has left / been expelled from the group:

$M_1 = \{Src = Man, Des = LM^{Broadcast}, PC^{CA}Man, info, Sig(S_{Man}, m_1)\}$

The CUG manager retrieves the data associated with the particular group, and creates the list of the local members $LM^{Broadcast}$. Also, it compiles the useful information - *info*, containing the identity and the certificate number (unique identifier) of the ex-member (XmemID and $AC^{Man}Cl\_No$, respectively), referencing the identity of the group concerned (groupID):

info = XMemID, groupID, $AC^{Man}Cl\_No$

The authentication certificate of the group manager is added, and the message is digitally signed. It is important to note here that the 'useful' part of the message (including manager's certificate) is the same for all the CUG members that should receive it. The message references data that is not valid anymore, and proper authentication is sufficient to assure the members to accept and process it. Therefore, separate encryption targeting every of the recipients is not necessary. This enables the manager to create and digitally sign a single message in an 'off-line' manner, and to reuse it during the broadcast, targeting each and every entity from the previously compiled $LM^{Broadcast}$ list.

However, dealing with the updates of the remote members is somewhat more complex. As addressed earlier, the CUG manager cannot contact the remote members directly due to restrictions at client's incoming firewall enforcer, imposed by the client's administrator (i.e., every message originating from the entity other than local manager or valid group member will be destroyed without further processing). However, the CUG manager maintains the list of the remote members and the appropriate local administrators, as a part of the CUG database. An initial relationship between CUG manager and local administrator(s) is established at clients' joining in the CUG, and remains the mode of non-direct communication between the CUG manager and its remote members. For every administrator node (that the CUG manager contacts for the purpose of non-direct group management), the CUG manager compiles the list of the associated clients who are current group members (RML – remote members list). Based upon this, the manager sends a single 'revocation' message to each of the identified administrators:

M_2 = {Src = Man, Des = Ad, ref = RML, PC^{CA}Man, info, Sig(S_{Man}, m_2)}

The info field has the same structure as sent to the local members (see above). Obviously, since the RML list will be different for every administrator, the messages will have to be separately compiled and signed. It is then the responsibility of the administrator to forward it to the appropriate client upon receipt [47]. Each of the administrators that receive the message compares the RML for the given groupID against the data they possess, and examines the content of the message. For every matching clientID (normally, the whole list should match), and provided the message is validated, it is broadcast in the similar manner as to the 'CUG manager – local members' process, only this time it is authenticated and signed by the administrator:

M_3 = {Src = Ad, Des = $RML^{Broadcast}$, PC^{CA}Ad, info, Sig(S_{Ad}, m_3)}

Parallel with this process, the CUG manager contacts the 'ex-member', informing it that its CUG membership has been revoked:

M_4 = {Src = Man, Des = XMem, PC^{CA}Man, leave, Sig(S_{Man}, m_4)}

As given on Figure 22a, the 'ex-member' is presented as a local client of the CUG manager. However, if membership is being revoked to a remote member, the above message M_4 would accordingly consist of two separate messages:

CUG manager – local administrator:

$M_{4'}$ = {Src = Man, Des = Ad, ref = Xmem, PC^{CA}Man, leave, Sig(S_{Man}, $m_{4'}$)}

Local administrator – 'ex-member':

$M_{4''}$ = {Src = Ad, Des = XMem, PC^{CA}Ad, leave, Sig(S_{Ad}, $m_{4''}$)}

[47] This also decreases the burden of the CUG manager's communication and shifts it to the administrators who are now performing the message multiplication and broadcast.

[b] Periodic manager's update

In this scenario, the group manager periodically updates the group certificate and membership lists and delivers it directly to the rest of the group (see Figure 22b). Again, inter-administrator communication takes place where appropriate, if the group contains remote members. Periodic updating of the certificate revocation lists is a standard solution adopted in ITU X.509 [136], and represents a compromise between performance and security, through variation of the update period **T**.

The process of the message exchange is the same as in case a), only that it is triggered periodically, and not by the event of members leaving the group. Therefore, the message contents will not be elaborated. The operation can be observed comparing the actions within Figure 22a and Figure 22b.

[c] Appointed member's update

With this approach, for every group it maintains, the manager appoints a so-called 'special member', who is responsible for replicating the messages received by the CUG manager, and distributes them to the rest of the group in a peer-to-peer manner, using the new membership list. Obviously, from the performance point of view, the group manager benefits from this *delegation of authority*. Furthermore, there is also a performance benefit for the administrators of remote group members, since this inter-administrator communication is avoided. The communication burden is shifted to the CUG member peer-to-peer plane, which can be significant for the large groups.

The CUG manager initiates the process by sending the single message to the special member:

M_1 = {Src = Man, Des = SpMem, PC^{CA}Man, ref = SpMem_fwd, info, Sig(S_{Man}, m_1)}

As before, the information field contains identity of the ex-member and group concerned, as well as certificate number (unique identifier): info = XMem, groupID, AC^{Man}Cl_No. The reference in the message body indicates that the message needs to be forwarded to the whole CUG - exclusive of the ex-member, as well as specifying the forwarding entity. Upon receipt, the special member takes the action of multiplying and distributing the message to the rest of the group:

M_2 = {Src = SpMem, Des = $CUG^{Broadcast}$, M_1, PC^{Man}SpMem, Sig(S_{SpMem}, m_2)}

The message is sent in the plaintext, enabling the recipients from the $CUG^{Broadcast}$ domain to read it in a straightforward manner. The recipients can also be assured that the message initially originated from the CUG manager, since the message M_2 contains the original message M_1, signed by the manager. Non-repudiation is additionally assured through the presence of special member's authentication key PC^{Man}SpMem and its digital signature Sig(S_{SpMem}, m_2) applied to the hash of the new message – by comparing the sender's identity from the certificate with the

one referenced in the body of the original message M_1, the recipients can confirm the authenticity and legitimacy of the information provided.

Finally (as in the previous two schemes), the CUG manager informs the 'ex-member' that its CUG membership has been revoked:

$M_3 = \{Src = Man, Des = XMem, PC^{CA}Man, leave, Sig(S_{Man}, m_3)\}$

From a scalability point of view, this scheme is the most beneficial to the CUG manager. However, it potentially carries serious security concerns, due to the fact that the administrator must delegate a certain amount of trust to the special member. If the special user was to become compromised, it could operate maliciously ant try to misuse the information received. However, the special member cannot modify the original data (as demonstrated above), and the worst-case scenario is that it may refuse to forward the message. Also, the problem arises if the special user is the one to leave the group. In such a case, the CUG manager could intervene directly and perform the group update through the one of the two mechanisms previously described - i.e. scheme a) or scheme b).

However, this scheme may be useful for relatively non-critical applications, such as advertising administrator services or other groups of interest and broadcasting general information, such as company news, etc.

4.5 Motivating Example Revisited: CUG-Enabled Cross-Organisation Collaboration

Referring back to the example from Section 4.1, we can now consider it from the perspective of the CUG architecture.

In order to perform the desired analysis, Alice uses its administrator who negotiates on her behalf with different service providers (SP) that offer different services. Based on the requested service, the SPs allocate resources for performing a particular task. In addition, Alice's administrator contacts the administrator of Bob's institute in order to agree on Bob's privileges, and Bob is given a certificate. It is assumed that Alice and Bob have previously agreed that they wish to pursue the analysis, for example via phone or at a coffee break. As the resources provided by the entity Comp1 of SP2 may not be sufficient, additional computational resources are identified and brought together, forming a new CUG created for that purpose (illustrated as a local group of SP2 in Figure 7).

Some of the interactions between different resources and that of Alice and Bob with some of the resources are mandatory. Obviously, it would be highly inefficient if all the analysis-related

communication would have to go through the administrators and SPs, following the same path as for the resource reservation. Following the architecture described in the above sections, administrators and SPs communicate appropriate certificates to the entities involved in the analysis, in order to support creation of Distributed Collaborative Group. It is the most probable (but not necessary) that the university will act as a group manager, since its member has initiated the collaboration. Also, since Bob's institute does not have any prior agreement with the supercomputer centre SP2, Bob's privileges in the group may include direct access to the analysis tool, but not to the computation resources.

It is possible to include all the resources required for Alice's analysis work into one CUG with a lifetime corresponding to the duration of the analysis. However, different issues could be raised here. For example, it is possible that the computation services required from the local group if SP2 are needed only as an intermediate result, and therefore there is no need to consume those resources throughout the duration of the analysis. Also, SP2 may wish to restrict the availability of the computing power (other than Comp1) to internal access.

In addition, perhaps not all the researchers from Alice's local group are qualified to use the tool, and SP1 wants to prevent non-qualified access. If that changes - i.e. another member of Alice's research team obtains the qualification (normally by the means of digital authorisation certificate), it may apply to join the collaboration. On the other hand, if Alice's qualification becomes outdated, she will be prevented from the group access (e.g. either her certificate will be revoked, or a new one will not be issued).

Such an approach facilitates direct interaction of the researchers with the services needed, therefore enabling the analysis to be conducted in an efficient way.

4.6 Architecture Summary

The architecture proposed in this chapter provides mechanisms where secure Closed User Groups can be dynamically altered in terms of membership and policy constraints. The hierarchical structure of the architecture provides the flexibility of both peer-to-peer (P2P) interaction and centralised community management. It supports on-demand creation and management of dynamic distributed collaborations in the form of secure groups of peers (users, services, resources, etc.) that cut across geographical and enterprise boundaries. The architecture has been developed with two main goals in mind:

- Enabling communication within dynamically created collaboration groups, that is: secure, scalable, robust and independent of network topology.
- Enforcing security perimeters that adapt to the dynamic evolution of a collaboration group (in terms of membership and security policy).

These goals are addressed through the following means:

- A hybrid architecture for establishment and management of dynamic distributed groups, consisting of a set of security protocols:
  - A hierarchical protocol between client(s) and administrator(s), for the introduction of clients and security policy distribution.
  - A peer-to-peer protocol among administrators, for supporting creation and management of inter-organisational collaborative environments.
  - A peer-to-peer protocol for securing communication among group members, which provides authentication, authorisation, non-repudiation and confidentiality.
- Certificates to manage CUG membership and privileges.
- Role based security policies describing relationships within CUGs.
- A mechanism for the distributed enforcement of local and CUG security policies, to protect individual members within a CUG and the CUG environment as a whole.

The following chapter describes the simulation model developed to validate and evaluate the message-exchange protocol of the architecture.

Chapter 5 Simulation Modelling

5.1 Introduction

In the previous chapter the architecture for secure and dynamic group working was presented and the specification of the group management protocol was described. Architecture design was an iterative process of refinements in order to meet security and performance requirements. The whole process was subject to a thorough security evaluation, in part by applying the author's previous experience in the field of security risk analysis **[DJO2]**. While the architectural discussion is largely qualitative, some estimates of the system performance are useful. The remainder of the thesis is concerned with investigating the performance of proposed message-exchange protocol under various conditions.

Having proposed the group management protocol, its performance characteristics can be evaluated through mathematical analysis or through a simulation model. However, the nature of the networks is such that the performance estimation becomes too complex for mathematical analysis, without unwarranted simplifying assumptions being made. Due to statistical variations of the sources (client and administrator nodes in this work) it is not possible to analytically determine the exact number of generated messages or the reaction upon receipt of them. In addition, changes in the behaviour of the stochastic system over time are very difficult to represent through analysis, whilst this data is naturally provided as a simulation output. Therefore a simulation model was used for the performance analysis.

A distributed network system can be modelled with an event driven simulator. The model can be built using a simulator developed specially for this purpose or by using a commercial simulator. Purpose-built simulators may be faster in terms of execution time, but the development time can be prohibitive. Commercial simulators are typically not as fast because their code is 'fully featured', but on the other hand, the existence of library models allows faster construction of more detailed models. In addition, the benefits of documentation and debugging facilities improve the quality of the resulting model. Finally, the ability to plug in existing models can be beneficial, for comparisons or for further work.

The commercial simulator used for this research was OPNET^{TM}, a general telecommunications network simulation tool [137]. OPNET is a discrete event simulator: the discrete events in the models described in this chapter relate to signalling messages and the event scheduling mechanism. OPNET uses graphical interface where simulation models are defined according to five levels. The first level is the project level, where the high-level components of a network are identified. At this level network models are created and edited, the simulations are run and the results are analysed. The second level is the network level, where the topology of the simulated

network model and interconnections of the nodes are defined. At the third level (the node level) the elements that make up the network nods and their interconnections are defined; these elements include queues, processes, sources, receivers and transmitters. After this, the functionality of each process or queue element is defined in terms of a finite state diagram and the transitions between states; this is the fourth level. The fifth and final level is where the processing in each of the states in the finite state machine is defined in C code.

5.2 Network Model

A typical network model used for the simulations consists of several administrator nodes and several thousands of client nodes. In addition, every configuration contains a 'statistics node', which does not participate in any communication. It is developed only for setting the initial conditions and data shared by all nodes, and collecting the global statistics at periodic intervals.

In general, the number of the administrator and client nodes varies in different simulation scenarios, in order to examine the protocol scalability. The actual distance between nodes is omitted; the delivery of the messages is modelled as a single-hop, independent of the underlying infrastructure. However, the message encounters a certain transmission delay, as explained in Section 5.5 Traffic Modelling. The simulation scenarios were performed for a duration of several weeks of simulated time, allowing the network to exhibit all the features of described protocol for a number of times. The typical client population ranged from between 1000 and 5000 nodes, whereas the number of the administrators was varied between 1 and 5, depending on the experiment.

The following sections describe physical and functional properties of the node and process models used. OPNET offers a range of modules that model different aspects of the network. However, most of the modules used in this research are purpose-built, only using OPNET functions for packet and memory management and event scheduling. This has enabled the research to focus on the aspects relevant to the proposed architecture, without dissipation effort on the low-level protocol stack and packet transmission mechanisms.

5.3 Node Models

The general model of the client and administrator nodes is presented in Figure 23. All the nodes comprise of single ingress and egress sub-queue buffer, and a purpose-built state machine. The 'client' and 'administrator' state machines are built according to the requirements described in the previous chapter, simulating: the features of message creation and delivery, certificate creation and assigning, database management, group management and negotiation, and corresponding processing and transmission delays encountered.

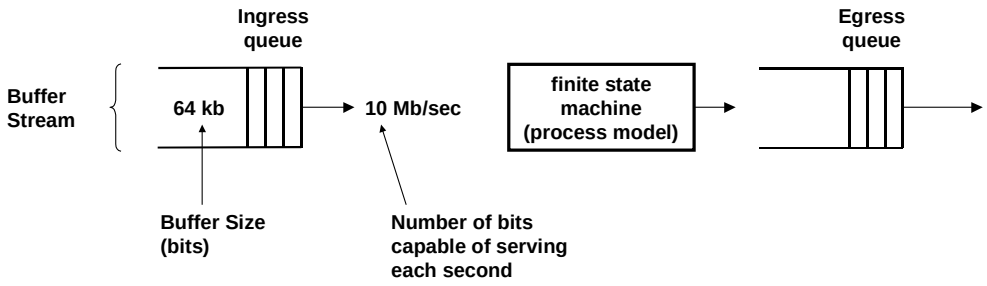


Figure 23: Generic Simulation Model of a Node (both for Client and/or Administrator)

5.3.1 Unique Node Identifiers

These are identities that serve for unique identification of the entities in the simulation model and are inherited from the built-in OPNET's feature to allocate different Object_id to every object placed in the network model. In such a way, different object_ids are allocated to a node, as well as to the queues and process model within a node. Object_ids are integer values, starting from 1 (for the network itself) and incrementing subsequently. Generally, nodes are allocated with the Object_id in no particular order, however the identifier does not change until the runtime termination (e.g. client process model maintains the same Object_id regardless of how many register/deregister events the node experiences).

The consistent use of the OPNET's Object_ids greatly facilitates the management of data structures, as well as modelling of node interactions. Within the scope of this research, these are used for message routing, as a part of certificates, and as a reference in the maintained databases.

5.3.2 Administrator

The administrator node maintains databases of the registered clients and of the CUGs it is responsible for. At initialisation of the simulation, the administrator obtains a large random number, which subsequently forms the part of the certificate that is issued to the clients. In order to maintain the uniqueness of this number, it is created by the statistics node for every administrator present. Its purpose is to model administrator's public key and digital signature, which by definition has to be unique in order for PKI system to operate correctly.

The administrator maintains several data structures, depicted in Figure 24. Whenever the client's registration is approved, it is added to the *Clients list*, which denotes the administrator's database within the scope of 'local policy'. Subsequently, when client creates/joins a group, that group_ID is added to the particular client. This is done only for the 'local groups', maintained by that administrator, in order to facilitate orthogonal searches when actions related to the CUG management are performed.

A typical example of when this is useful would be a client's deregistration. If a client requests to deregister, the administrator needs to update the group member lists of all the relevant CUGs (that the client was a member of – in a typical simulation scenario client is not member of more than 20 groups). Instead of searching through all the possible CUGs (which can be as many as 1000) in the *CUGs list*, administrator 'finds' the client in the *Clients list* and, based on the retrieved information, accesses only the relevant groups. Therefore, this can be seen as a reference link between the client and the CUGs it is member of.

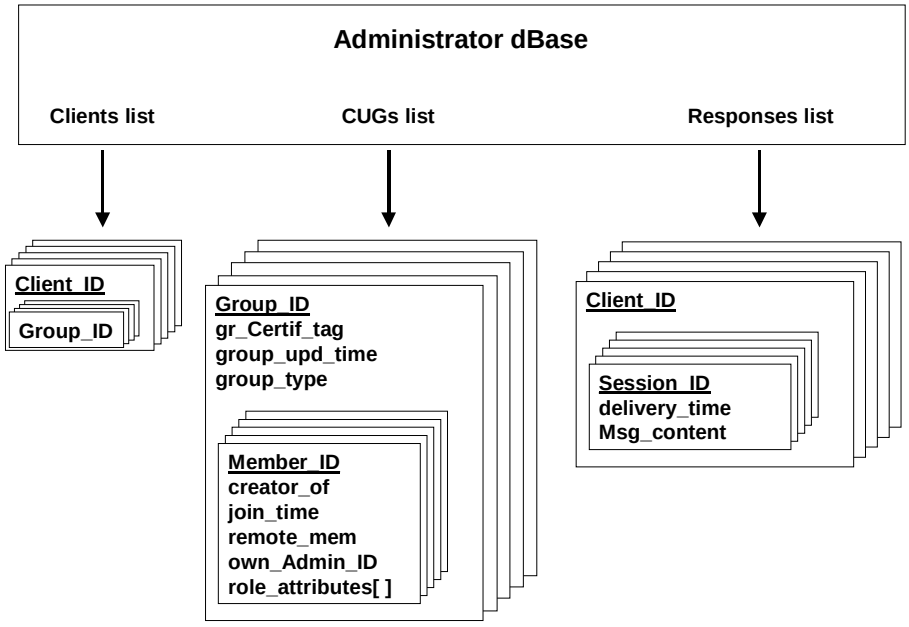


Figure 24: Data Structures Maintained at Administrator

For the purposes of group management, the administrator (now in the role of CUG manager) maintains a separate list of the valid groups that it has created at some point in time. Each of the elements in the *CUGs list* is a true 'CUG policy', maintaining all the data needed to support appropriate functioning and management of the CUGs. Creation of a group is initiated by a client's request. At creation, the CUG manager defines several parameters:

- Group_ID - a unique identifier of the group.
- gr_Certif_tag – a common part of the group certificate, that models the manager's signature over the attribute certificate issued to a member. The 'update of the certificate' over the group population would correspond to the change of this value. Subsequently, in any peer-to-peer or manager-member CUG communication, after the Group_ID from the message is matched with the one from the database, legitimacy of the message is confirmed only if the gr_Certif_tag(s) are the same. In order to keep 'per-group' uniqueness of the tag, the value

is chosen to be the group creation time (or subsequently the group 'update' time). The prospect of having several groups created at the same time is avoided due to single sub-queue model, enabling that only one packet at the time can be passed to the administrator module for processing.

- group_upd_time – the simulation time of the 'next group update', for the periodical broadcast of group update messages. At each group update, it is incremented by a constant value. The next update is scheduled via self-interrupt at the defined time, when the group to be updated is identified by comparing the current time with the value of this parameter.
- group_type – a parameter randomly chosen by the manager. Its purpose is to influence the generation of peer-to-peer traffic that characterizes different types of groups, such as business, social, etc.

In addition to this, every group contains the list of current members. The following information is associated with every member:

- Member_ID – inherited from OPNET's Object_id.
- creator_of – field distinguishing the group creator, which has importance only in some of the scenarios preformed to investigate revocation mechanisms.
- join_time – simulation time of clients' joining a group.
- remote_mem – field marking if a member is originally associated with another administrator.
- own_Admin_ID – the Object_id of the original administrator (used only for the remote group members).
- role_attributes[ ] – an array of 3 integers modelling the privilege-related attributes, as the model allows different types of communication to be enacted among the CUG members (e.g. email, file transfer, print job submission, etc).

Responses list is a part of the ARQ mechanism used to prevent deadlocks in the system: as every client's request is accompanied by the unique incremental number (Session_ID), every response message is stored (Msg_content), referenced via this number. The field delivery_time denotes the simulation time when message is replied to a client, enabling messages to be deleted if the request has not been repeated after the time-out period [48].

The finite state machine model of the administrator module is given in Figure 25. Once initialisation of the module is completed at the beginning of the simulation, the module goes into an idle state. The largest portion of the module's activity is governed by the incoming messages. For every message received, its type is examined in the idle state. The messages

[48] For reliability, the CUG communication protocol makes use of its own ARQ mechanism.

received can be generally divided into two groups: local and remote. The appropriate states deal with different types of the messages received and generated, upon which the message is sent and the processing time during which the module is 'frozen' is estimated.

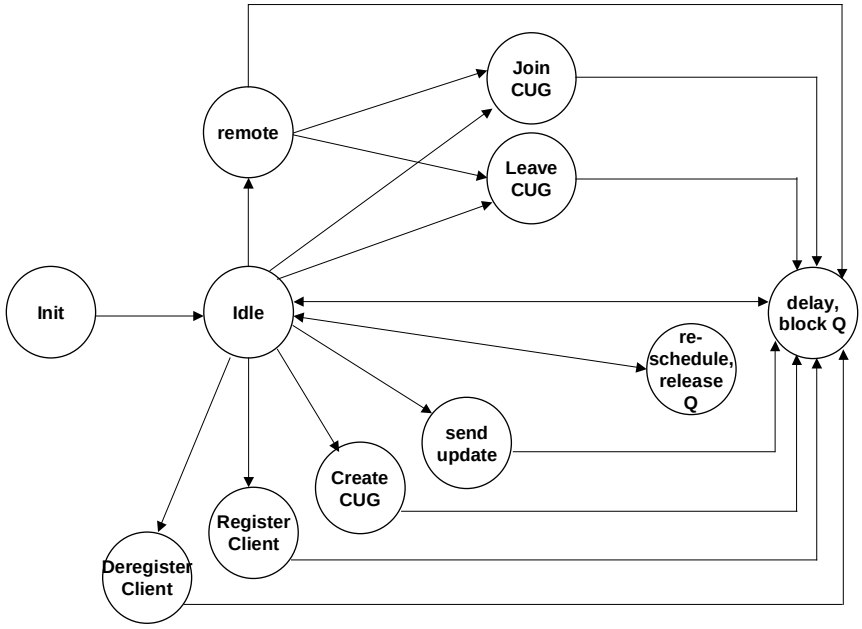


Figure 25: Finite State Machine of the Administrator Module

Local messages refer to the clients' requests related to the inclusion in the CUG environment, and these can be: register with administrator, deregister, create a new CUG, join existing CUG, leave a CUG. The corresponding flag in the message format distinguishes these messages, and determines the transition from the idle state [49]. The message is then dealt with in the appropriate state and the acknowledge response message is generated and sent back to the client. The acknowledgement can be either positive or negative (i.e. client's request has been accepted/rejected). In the case of a negative response, the acknowledgement value zero is sent. In the case of the positive response, the related information is included in the message. Table 5 gives the message content for each message type generated at the administrator module.

Remote messages refer to the creation of inter-domain CUGs, and have significance only in the simulation scenarios performed with the several administrators. For the purpose of inter-domain CUGs, clients' requests are limited to joining one of the existing CUGs and leaving a CUG. In addition, these messages may refer to inter-administrator communication, related to the forwarding of clients' requests as explained earlier. The remote type of message is distinguished by a suitable flag in the message format, which determines the transition to the 'remote' state. If

[49] The message format is explained in detail in Section 5.4.

the received message is the original client's request, it is forwarded to the referenced 'remote' administrator node. If the received message is a forwarded request, the type of message determines the transition to the join/leave state, where the request is served, and the message is sent back to the client via its local administrator.

Table 5: Content of the Messages Generated at the Administrator Module. (In addition, all the messages are accompanied with the model of the administrator's PKI certificate)

Message type	Meaning	Message content
Register response	Registration to the CUG environment permitted	Local PKI certificate for the client, ACK = 1
Deregister response	Deregistration from the CUG environment permitted	ACK = 1
Create response	Creation of new CUG accepted	CUG ID, AC certificate for the client, ACK = 1
Join response	Joining to the existing CUG accepted	CUG ID, AC certificate for the client, list of current members, ACK = 1
Leave response	Leaving a CUG of client's choice accepted	ACK = 1
CUG update	Administrator's update of the CUG status, broadcasted to the current CUG members	CUG ID, list of current members
Forwarding of request/response	Local Administrator's forwarding to the remote Administrator or back to the client	Unchanged content, update source/destination
ARQ response	Resending of the message taken from the ARQ database, following the Session_ID	Unchanged message content, ARQ_flag = 1

In addition, automatic repeat requests from the clients (ARQ) are identified in the idle state through the resend flag, and in the case of such a request the response with the corresponding Session_ID is taken from the ARQ database and replied to. The rest of the states are omitted and the transition leads directly to 'blockQ' state (denoted with the two-way arrow in Figure 25). While in the 'blockQ' state, the calculated processing time is communicated to the ingress queue module through inter-process communication, and during this time the ingress queue is disabled from serving and sending any further packets. This mechanism is described in more detail in Section 5.3.4 Queues. The simulation time when the ARQ message has been resent is updated, as the mechanism allows stored messages to be deleted after the set timeout period expires.

Also, the administrator module generates the 'CUG update' messages, targeting all the current members of the particular group. This is implemented as the administrator module schedules the event in the future, triggered either instantly after the member has left, or periodically (based on the pre-defined update period).

Finally, all the actions performed during the single process invocation add up to the total processing delay that occurs in the module for that particular invocation. The state 'delay, block Q' executes an immediate event at the ingress queue of the node model that causes queue process module to block and stop further message delivery until it is unblocked, and schedules

an event in the future that will set transition to the 'reschedule, release Q' state. In the 'reschedule, release Q' state, the timers related to the processing delay are reset, and the instant event at the ingress queue is executed, causing the queue to unblock. The principle behind this process and the way the delay is calculated is further elaborated in Sections 5.3.4 and 5.5.1. Generally, the processing delay takes into account data structure manipulation and the actions related to the encryption and certificate generation. Following the approach adopted in the group management protocol (as described in Section 4.4 Description of Security Protocol), Table 6 summarizes relevant encryption and authentication -related delays for each of the messages processed.

Table 6: Security Procedures at Administrator Performed for Different Types of Messages

Message type	Processing delay
Register response	PKI certificate generation, asymmetric encryption, digital signature
Deregister response	Digital signature
Create response	AC certificate generation, asymmetric encryption, digital signature
Join response	AC certificate generation, asymmetric encryption, digital signature
Leave response	Digital signature
CUG update	Digital signature
Forwarding request to the remote Administrator	Symmetric encryption, digital signature
Forwarding response to the client	Asymmetric encryption, digital signature
ARQ response	None – message is already compiled, stored in ARQ database

5.3.3 Client

Clients are able to sequentially register as many times as they like, by randomly choosing one of the available administrators. For the period of registration all management-related communication goes via that administrator, and at every deregistration all the previous data is destroyed. Client modules maintain the data structures as shown in Figure 26.

For every group it participates in, the client module maintains the following data (given by the CUG manager at the joining time or via update interactions):

- Group_ID - a unique identifier of the group.
- gr_Certif_tag – a common part of the group certificate, that models the manager's signature over the attribute certificate issued to a member.
- own_role_attributes[ ] – an array of 3 integers defined by the CUG manager at the joining time, that models the privilege-related attributes of the client.
- group_type – to distinguish between types of groups, such as social, business, etc.
- group_remote – field marking if that particular CUG is remote (associated with the CUG manager other than client's administrator).
- CUG_manager_ID – the Object_id of the CUG manager (used only for the remote groups).

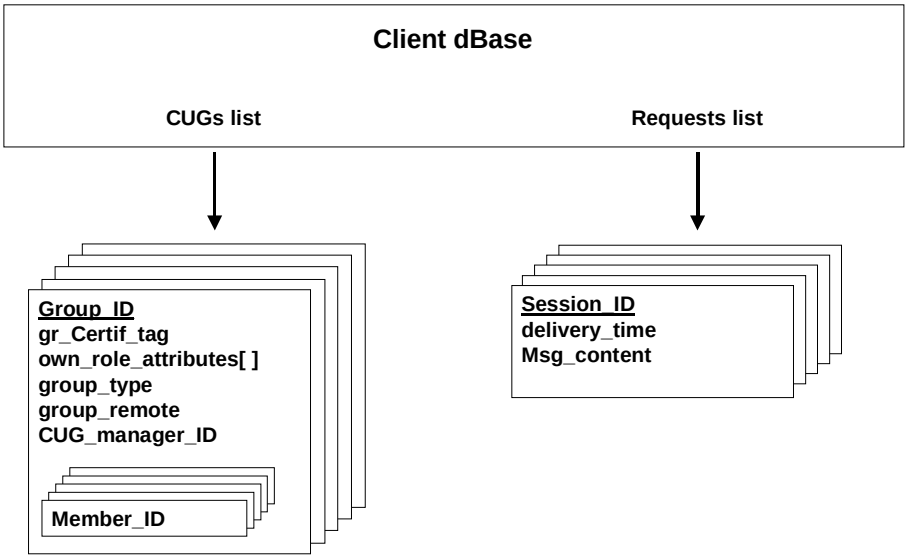


Figure 26: Data Structures Maintained at Client

Similar to the administrator module, the *Responses list* is a part of the ARQ mechanism: for every sent request, a unique incremental number (Session_ID), message (Msg_content), and the simulation time when it has been sent (delivery_time) is stored. If the response (carrying the same Session_ID) is not received within the time-out period, the message is re-sent and the delivery_time is updated.

The finite state machine model of the administrator module is given in Figure 27. Once initialisation of the module is completed at the beginning of the simulation, the module goes into the idle state. During initialisation, one of the available administrator nodes is chosen and the registration event is scheduled to happen at some time in the future.

The request messages related to the CUG management are generated in 'initiate request' state and delivered to the chosen administrator node. Every message is stored in the 'resend' database (ARQ list) and the event is scheduled after timeout period (following the transition to the ARQ state) to resend the message if the acknowledge reply has not been received. For replies received within the timeout period, the message with the corresponding Session_ID is deleted from the ARQ list. Every reply message from the administrator is examined in idle state for the message type, which is used to select the appropriate next state. The model allows the client to choose the CUG it wants to leave (and this information is included in the request message), whereas for create and join requests the client is allocated a CUG by the administrator. Separate states are used to simulate peer-to-peer communication among the group members.

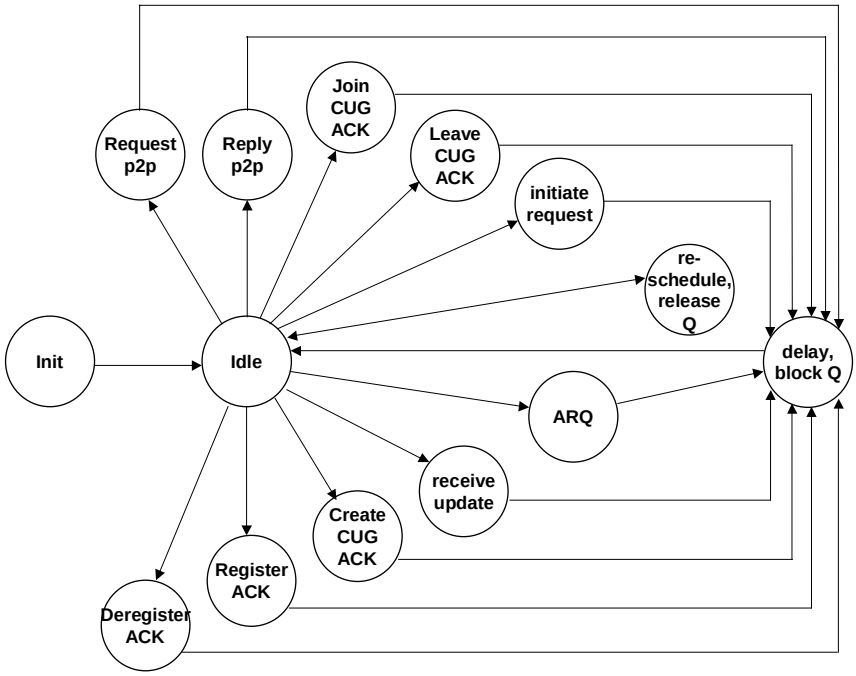


Figure 27: Finite State Machine of the Client Module

Dynamics and patterns of message exchange are modelled according to the case studies reported in the literature, which is further explained in Section 5.5.2. In general, upon receipt of a reply message, a number of choices for scheduling further message-generation events are available in the corresponding states. This depends on the nature of the received message (namely: request accepted/rejected) and the type of the message, which is summarized in Table 7.

Table 7: Scheduling of Events for Generation of Request Messages at Client Process Module

Message received	Event scheduled based on ACK value	
	Accepted	Rejected
Register response	Deregister, create, join, ARQ	Register, ARQ
Deregister response	Register, ARQ	Deregister, ARQ
Create response	ARQ	ARQ
Join response	Leave, P2P, ARQ	ARQ
Leave response	ARQ	ARQ

Most of the signalling traffic is implemented in a way that a client node cannot initiate a new request unless the response (either positive or negative) from the previous request of the same type has been received. However, the 'initiate requests' state always schedules the additional future event for create/join at the time of the generation of the corresponding request messages. This gives the possibility of the multiple requests being sent before a response has been received, and also models the generation of a subsequent request if the previous one has been rejected. Such a mechanism has not been implemented for leave messages (in order to avoid

client's leave request when it is not member of any groups), and neither for register/deregister (since client can be registered or deregistered only once at the time). In addition, the initial registration request is scheduled at the time of initialisation (for triggering the simulation), and a resend event after a timeout period is scheduled at every invocation of ARQ state.

Similar to the administrator module, all the actions performed during the single process invocation add up to the total processing delay, causing blocking and unblocking of the ingress queue at the client node. Table 8 summarizes relevant encryption and authentication -related delays for each of the messages processed.

Table 8: Security Procedures at Client Performed for Different Types of Messages

Message type	Processing delay
Register request	Key pair generation, asymmetric encryption, digital signature
Deregister request	Asymmetric encryption, digital signature
Create request	Asymmetric encryption, digital signature
Join request	Asymmetric encryption, digital signature
Leave request	Asymmetric encryption, digital signature
Peer-to-peer	Symmetric encryption, digital signature
ARQ request	None – message is already compiled, stored in ARQ database

5.3.4 Queues

OPNET provides a number of different queueing disciplines [138], which cover many common situations. The queue model used for the purpose of this research is acb_fifo (Figure 28). Its characteristics are:

- *Active*: autonomously dequeue and forward packets based on service rate, (vs. *passive* that instantly forwards packet when requested).
- *Concentrating*: all incoming packets are buffered into a predefined number of subqueues and forwarded to the same output stream (vs. *flow through*, where a dedicated subqueue exists for each pair of input/output streams).
- *Bit-oriented*: service delay is computed based on the bit-size of each packet (vs. *packet-oriented* which uses the same constant service rate for all packets).
- *FIFO* (*first-in-first-out*): packets exit the queue in the same order they arrive (vs. *LIFO*: last-in-first-out).

The majority of the experiments described in this thesis are performed to test the signalling group-management protocol, and the chosen queue is the single subqueue (therefore treating all the signalling messages with the same priority). The acb_fifo model takes into account different packet sizes, and enables modelling of the network throughput (by setting the service rate). However, the original queue model does not take into account the processing delay for specific encryption and authentication procedures, modelled in the administrator/client process modules.

In order to enable this functionality, the OPNET model has been modified as shown in Figure 28.

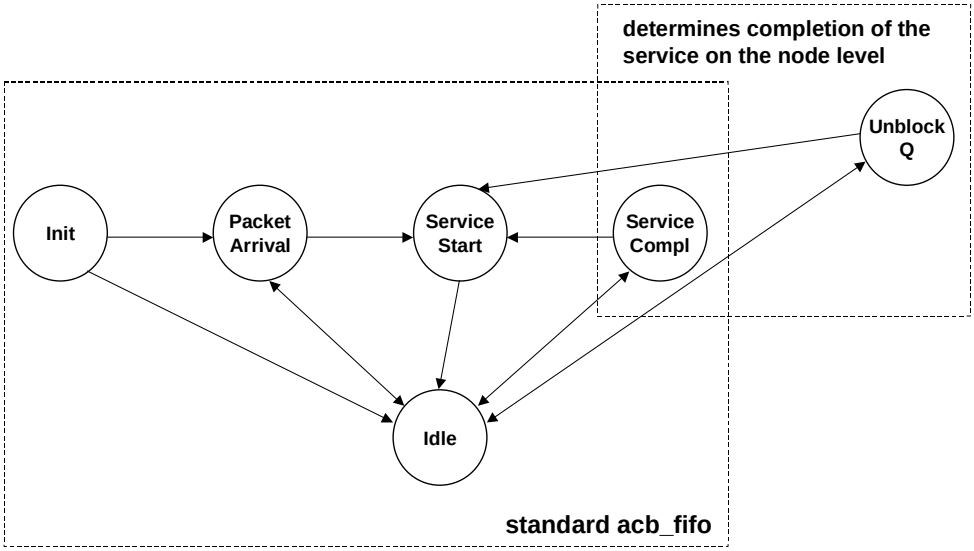


Figure 28: Finite State Machine of the Modified *acb_fifo* OPNET Queue Module

Initially, the packet service time is estimated (based on its bit-size) and the end of the service is scheduled in 'Service Start' state. An end of service event triggers the transition to the 'Service Compl' state, where the packet is sent and the next one taken for servicing from the head of the subqueue. In the modified model, the *Qblock* constant is set such as to block further servicing at the time of packet delivery, by disabling all transitions to 'Service Start' state. This constant can be reset only from the process module that receives packet, and only after the calculated processing delay expires (as explained in Sections 5.3.2 and 5.3.3). Such a mechanism increases the average queueing delay to the sum of the service times (based on the packet bit-size and the network connection speed), and the processing time (needed to perform specific group-management related actions). Also, for the packets generated in the process model, the remotely scheduled event at the ingress queue will trigger the transition to 'Unblock Q' state, which will then either block or unblock queue, depending on the type of events scheduled. This enables a processing delay to be taken into account even for those packets that are not received on the ingress queue, but are manipulated within the node. This principle, and the placement of modules within the network model, is further depicted in Figure 29.

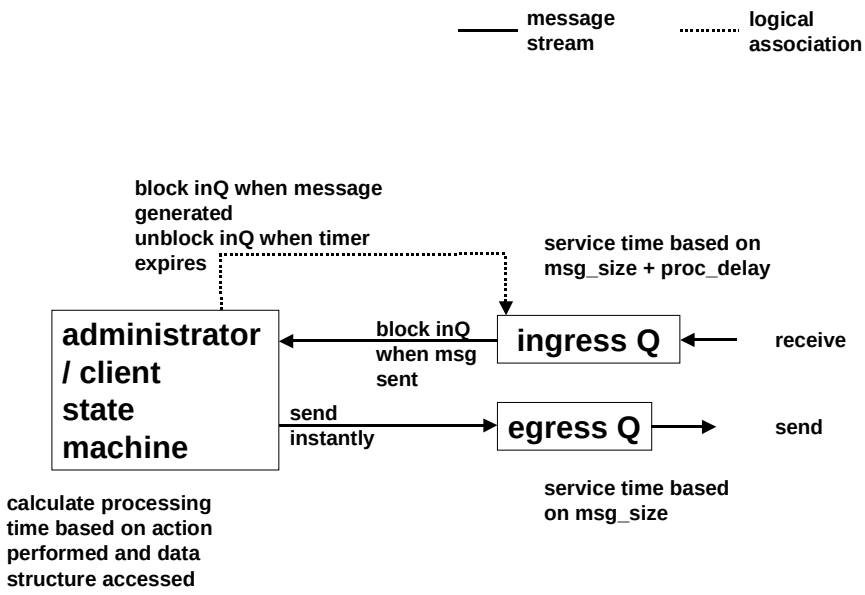


Figure 29: Placement of Queue Modules within Network Nodes and Inter-Module Communication

Regarding the buffer size at the ingress and egress queues, the decision was based on the values found in literature. For example, [145] reports that the default maximum buffer size of UNIX OS is 256kB, and the one of Linux OS is 64kB. Any value in this range seemed reasonable, so the buffer size of 128kB was adopted in the experiments. The service rate of the queue modules, set to 10Mbits / sec, is that of an Ethernet connection. The egress queue delivers packets with 100ms delay, which models propagation and transmission delays across a WAN (Wide Area Network).

5.4 Format of Messages

The message format used in the simulations is created with the OPNET 'Packet Format' editor, and is given in Figure 30. Signalling messages are modelled as a single packet of size 500 bytes. Peer-to-peer communication is modelled as a stream of packets, where only the first one carries 'useful' information (regarding attribute certificates) and the packet sizes available are 60, 500 and 1400 bytes. Common fields for both the signalling and peer-to-peer messages are the source, destination addresses and the appropriate certificate. For group management messages, relevant fields are:

- Session_ID – unique message identifier to support the ARQ protocol.
- Group_ID – references the group the client is joining or leaving (not used for register/deregister messages).
- mess_type – to distinguish the type of the signalling message.

- ACK – used in administrator's response to denote if request has been accepted (value 0/1).
- ARQ – used in client's request to denote resent message (value 0/1).
- rem_tag – to distinguish client's 'local' and 'remote' request (possible values are: 0 – local request; 1 – remote request; 2 – administrator's forwarding of request; 3 – remote response; 4 - administrator's forwarding of response).
- referenced_ID – used for remote request/response messages to denote the ultimate destination / source of the message.

If client's request to create/join a group has been accepted, or at the CUG update, the administrator includes the following data in the message:

- Member_ID_alloc – ID of the member that has left the group (if the CUG update is performed instantly after a member leaves the group).
- Group_ID_alloc – relevant group (either the one that the update refers to, or the new allocated group).
- gr_data – list of current CUG members (for the CUG update –performed either periodically or when group member leaves a group).
- Certificate_alloc – newly allocated group certificate (at client's joining / CUG creation) or authentication certificate (at client's registration).
- gr_type – referring to a social/business type of group

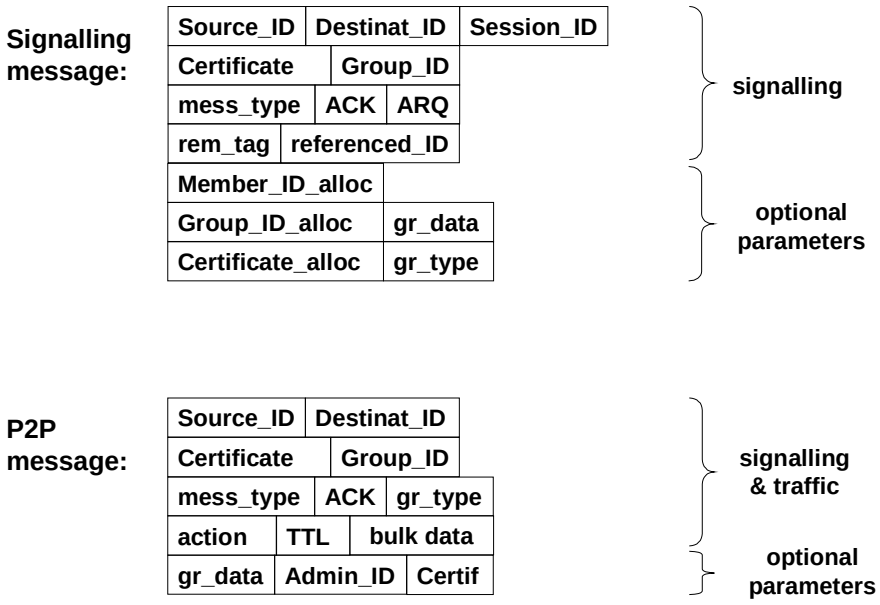


Figure 30: Message Format Used in Simulation Model

For the client level peer-to-peer message, the relevant fields are:

- Group_ID - references the group that the P2P session (and the corresponding certificate) refers to.
- mess_type – to distinguish a P2P message from a signalling one.
- ACK – used in response message to denote whether the request for a P2P session is accepted.
- gr_type - used to distinguish between different types of group (and therefore influence the way P2P traffic is generated).
- action – refers to the action requested by a client, in a peer-to-peer group communication. This is compared with the role attributes (from the group certificate) upon receipt, in order to determine whether the requester is allowed to make such a request.
- TTL – time-to-live, determines the number of hops that the message can be forwarded.
- bulk data – used to model payload of different sizes.

As already described in Section 4.4.4.2, some of the scenarios support the CUG updates being performed by a chosen 'special' group member. In such a case, the appointed member broadcasts to the group the message containing the following information:

- gr_data – data related to the group (i.e. current group members).
- Admin_ID – the idenitity of the entity that the information originates from.
- Certif – part of the certificate that models digital signature of the entity (i.e. administrator).

5.4.1 Certificates

The format of public-key certificates and attribute certificates is defined by [32],[47]. The same source gives some recommendations on the typical size of the certificate, which is comparable to several hundred bytes. In this research, certificates are modelled in two respects:

- As a means of authenticating source/destination, and defining set of the entities' permissions (based on the relevant certificate fields as identified in Section 4.3)
- Time delays introduced by the operations needed for certificate generation, distribution and manipulation.

Figure 31 depicts the way certificates are modelled. The Holder_ID and Issuer_ID are integer values inherited as Opnet Object_IDs. Group_ID is an integer value, incremented by the administrator every time a new group is created. Attrib1-3 are integer values defined at the time of a client's joining a group; they are used to determine what type of peer-to-peer traffic the client can generate based on a pre-defined hard-coded table that models privileges associated with a given role. The unique identifier is a real (floating point) value that models the certificate issuer's signature and the certificate serial number. For public-key certificates, its value is defined by the 'statistical node' at the simulation initialisation, and is different for each of present administrators. For attribute certificates, its value corresponds to the current simulation

time of the group creation. This ensures that one administrator does not have the same 'certificate' value for two groups it is managing (since the block/unblock mechanism of the ingress queue prevents processing of two messages at the same time). Per-member uniqueness of the attribute certificate (for the same group) is achieved by presenting the Holder_ID (effectively, sender).

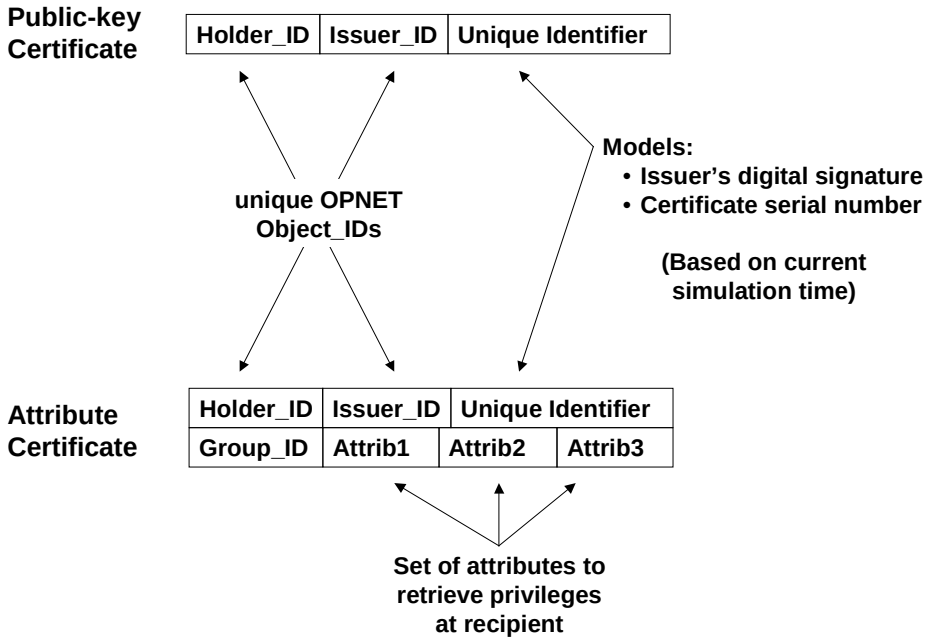


Figure 31: Modelling of Certificates for Simulation

As the main purpose of the simulation was to test the group management protocol, the main emphasis is on the interaction between clients and administrators. For messages originating from the administrator, a client checks 'Holder_ID' against the administrator's ID stored in the database at the registration. For messages originating from the client, the administrator compares the 'Holder_ID' against the client's ID, by searching through the database of registered clients. In addition to this, a check of the 'unique identifier' is performed in order to ensure that the administrator and a client belong to the same registration domain (this is particularly important in the simulation scenarios with several administrators). The 'Issuer_ID' would be needed in the case of the certificate validation, when the certificate issuer needs to be consulted. However, this aspect has not been included in the simulation experiments.

The approach in modelling time delays for certificate management is described in Section 5.5.1 Processing Delays.

5.5 Traffic Modelling

There is no distinct traffic source in the network. The traffic sources are client nodes themselves, which generate the appropriate signalling / data messages. Also, the administrator nodes generate the additional traffic, depending on particular interactions. The motivation was to test the capabilities of the nodes, and performance of the management protocol itself. The assumption is that the underlying infrastructure is capable of providing sufficiently realistic data flows, and for that reason additional background traffic has not been modelled.

5.5.1 Processing Delays

The processing delay is calculated taking into account magnitude of the data structure manipulation and modelling of cryptographic algorithms performed.

Although data structures are implemented as double-linked lists, the time needed for accessing the appropriate data is estimated based on balanced binary tree database organisation. For every list of size N, the number of average search iterations (*units*) performed is estimated as the maximum complexity of the binary search:

$$units = \log_2(N)$$

As an illustration, consider the example of a client who requests to leave a specific group. Let's assume that the number of clients registered with the administrator is 1000, and that administrator currently maintains 500 groups where size of each of them is 100 members. Upon receiving a request, the administrator checks if client is registered. In addition to comparing the certificate, this is performed by searching through the list of registered clients (size of 1000):

$$units = \log_2(1000)$$

If this is confirmed, the administrator needs to find the appropriate group (referenced in the client's request) and within group's list to match the corresponding client's ID. Therefore, the full number of operations is:

$$Units = \log_2(1000) + \log_2(500) + \log_2(100) = 26$$

The total time needed for data structure manipulation is calculated as the number of estimated units multiplied by the *base_time*. This value is taken from the speed rating for an Intel Pentium4, 2GHz CPU. Benchmarks both from Intel [140] and independent test [139] reported values over 1000 MIPS (million instructions per second). Having in mind that a basic operation such as an access/read/write to disc consists of dozens instruction, values in range of several micro seconds were used. Referring back to the above example, the total time needed for the data manipulation (e.g. for a base_time = 1 micro sec) is calculated as:

$$data\_time = units * base\_time = 26 \text{ micro sec}$$

The author is aware that modelling of different database types would impact on the performance results. However, the decision on the database organisation to be used is to large extent based on the choice of optimal implementation [141], which goes beyond of the scope of this research. Also, if compared to the average values used for the modelling of cryptographic algorithms (further down), it is obvious that this delay is nearly negligible.

Another aspect of the system delay that was modelled was that of the cryptographic algorithms. It is well known that applying security procedures cause significant delays in processing, which has been widely documented in the literature, both through analytical and experimental means [3],[6],[142].

In terms of this research, detailed security requirements have been identified, which have been elaborated through the protocol description. In terms of the simulation model, it is important to say that no actual cryptographic algorithms were implemented. These were modelled as a time delay (needed for performing certain operations), based on benchmark values obtained through measurements and reported in [143],[144]. The operations performed for different types of received / sent messages are given in Table 6 and Table 8, and the approach in modelling the delay is described in Section 5.3.4 Queues. Table 9 lists the values used in the experiments for modelling processing time of security procedures applied at signalling messages.

Table 9: Values used for Modelling of Security Procedures in the Simulations, taken from [143],[144]

Operation	Algorithm modelled	Time (msec)
Key pair generation	DH 1024 Key-pair generation	1100
Asymmetric encryption/ verification	RSA 1024 public key operation	0.6
Asymmetric decryption/ signature	RSA 1024 private key operation	43
Symmetric encryption/ decryption	DES encryption/ decryption	0.0215
PKI/AC Certificate generation	RSA 1024 private key operation	43

5.5.2 Signalling Messages

The principle of message generation at the administrator and client nodes is explained in Sections 5.3.2 and 5.3.3. In order to properly model the organisation structure, parameters for frequency of different events need to be identified, in addition to the appropriate model. Currently, there is a number of researches focusing on measurements and characterization of traffic in the Internet, and recently an increasing number of these are focusing on examining various peer-to-peer applications and protocols. When adjusting parameters for the experiments performed within this research, the following assumptions and reported results were considered:

- The duration of the clients' registration lasts for a relatively long time. In current e-commerce and peer-to-peer systems client population 'builds up' over very long period of

time (e.g. registration of users at Amazon, bank services, etc.) with only occasional deregistrations.

- According to [108] and [148], peer-to-peer systems exhibit no more than 15-20% population change during a 24-hour period. In the context of this research, this is interpreted as the group life-time could be expected to last for days.
- A recent detailed survey of the Jabber peer-to-peer platform, performed by Jabber Software Foundation [150], reports that 78% of Jabber servers have less than 100 users, and 89% less than 500.
- According to results reported in [108], collected during 8 days of observation of Gnutella P2P network, the overall number of captured hosts was between 8000-10000.
- Also, the same source [108] reports that over 50% of Napster's users and 60% of Gnutella's users are with broadband connections, with the majority concentrated around 1-3.5 Mbps. In this sense, the simulator developed for the purpose of this study, modelling network connections as 10Mbps Ethernet links would appear appropriate [50].

In consequence, different events in the simulation model are parameterised as follows (all the values are defined as simulation time):

- **Register**: At the program initialisation, clients register randomly in time, over the period of-4 weeks[51]. If registration is not approved, a client will try to register again within period of 12 hours. If successfully deregistered, client will attempt to register again within 1-2 days.
- **Deregister**: Once registered, client will schedule its deregistration request to occur within not less than 3 weeks, but not more than 4 weeks from successful registration acknowledge. If the deregistration request fails, another deregistration request is set to happen within 2-3 days.
- **Create and Join**: If a client's registration is approved, it will attempt either to create a new group or to join some existing ones within 24 hours. The probability of choice between a creation and joining is set as 90% vs. 10%, in favour of a join request. In addition, if the request for create/join is sent, another (corresponding) request is scheduled to happen within 2-3 days, or 12-24 hours, respectively.
- **Leave**: Once client receives the acknowledgement of successful creation or joining a group, it will schedule a request to leave a group. As suggested in [108] and [148], there is no more than 15-20% of population change in 24-hour period. Therefore, clients will request to leave the group within 12-24 hours after joining with the 10% probability. Otherwise, the request to leave the group is scheduled randomly within 1-8 days after the joining. If the original request was rejected, client will try again to leave a group, approximately within 24 hours.

[50] This also opens a potential issue of modelling several classes of users, partly based on bandwidth. However, this opportunity is not considered further though it could form a topic for further investigation.
[51] The choice of this value will be explained further down.

- **CUG Update (revocation)**: Various revocation mechanisms (as described in Section 4.4.4.2) have been tested. The approach used in the most of the experiments is that of periodically scheduled updates. In most of the experiments the period is set to 24 hours, although different update times were also tested and compared (see Section 6.2.2.4).

Other important parameters in the system are the actual number of groups maintained by the administrators, as well as the group size, and also the maximum number of groups that a client can be concurrently a member of. According to [150], most of the Jabber peer-to-peer groups have populations of less than 100 members. On the other hand, it is reasonable to assume that clients will rarely be members of more than ~10 groups: in a real life, an employee of a company could be involved in several projects, but very rarely more than 5. If we include on the top of that additional domains for the company and department, several news groups and open communities for information sharing, the assumption is that an average user will not be member of more than 10-15 separate CUGs. In addition, the empirical results on host connectivity from [149] report that "about 48% of the individual IPs communicate with at most one IP, and 89% with at most 10 other IP addresses" (at the time), supporting this assumption. From the perspective of this research, the interpretation of these measurements is that clients will be members of 10 different groups at most, with the majority of the groups having population of less than 100 members[52]. Therefore, if a client is already involved in the maximum allowed number of groups at the time when create/join request is scheduled to happen, it will give up the request. In a similar way, if the administrator maintains the maximum number of groups, it will refuse any further requests for group creation. Typical figures used in the experiments are summarized below:

- Maximum number of groups per client: 10 groups.
- Maximum number of groups per administrator: 200 groups.
- Maximum size of each group: 100 members.

5.6 Simulation Modelling Summary

In this chapter the simulation model, developed with the commercial simulator OPNET™, was presented. The structure and functionalities of client and administrator nodes, as well as of modified OPNET queue module were described. The traffic model and the processing delays were implemented according to the findings of measurements and benchmark testing, reported from academia and industry.

The next chapter discusses validation and verification of the simulation model. In addition, simulation scenarios are described, and performance results obtained and analysed.

[52] It is important to note here that those assumptions do not introduce any constraints on testing or functionalities of the protocol. Those are only the figures abstracted from the real network measurements in order to set-up realistic parameters in the system.

Chapter 6 Simulation Results and Analysis

6.1 Verification and Validation of the Simulation Model

In order to determine if the developed simulation model accurately represents the functional behaviour of the system described in Chapter 4, it needs to be verified and validated. Verification determines whether the simulation model performs as intended and the validation determines whether the conceptual simulation model is a true and accurate representation of the system under study. If the verification and validation of the model is satisfactory, and therefore the model is credible, it can be used as an aid in making conclusions and decisions [153].

6.1.1 Verification

Verification of the simulation model needs to be carried out at two levels, first on a fine scale by looking at the individual objects that make up the network and then at the whole network. In addition to purpose-built node processes to model the administrator and client functionality, the simulation model uses a library queue model supplied with OPNET. The library model has been verified and tested using purpose built test models.

The simulation architecture was verified at the network layer using screen printouts, file reports, and, in particular, OPNET ODB (debugging) tool, allowing for several kinds of traces and breakpoints to be applied during the execution of the simulation. The overall simulation results and the intermediate results obtained in traces and breakpoints were checked for consistency and coherency. In particular, packet tracing was applied to confirm the path of the packets in the system, and the content of the individual packets, as well as to reveal potential live-locks in the system. Various simple simulation scenarios were performed, where different types of messages were stimulated on purpose, and the path from the source to the (appropriate) destination was traced. This was particularly important in two cases:

- When functionality for supporting inter-Administrator communication was introduced, the correctness of the functionality for message forwarding was monitored, with respect to the types of message.
- When ARQ functionality was added, the content of the re-transmitted messages and 'flushing' of ARQ data structures (based on the time-out period defined) was traced.

The OPNET ODB package was particularly useful for checking the implemented mechanism for processing delay, since this functionality required inter-process communication. The functionality was quite straightforward to implement, since OPNET allows several types of inter-process communication, including forced invocation of remote events and specific types of statistics that could be both read by the system and written to an output file. However, the timing of these interrupts and the duration of the blockage period needed to be confirmed and

compared against the parameters set for the system. This is where the ODB package proved very helpful.

Test versions of the model were run for various durations and with different values of various parameters, in order to assess the stability of the simulator in terms of memory usage, oscillatory behaviour and potential deadlocks in the system. As an example, the diagram in Figure 32 gives the measured number of packets in the simulation model over the time. Once the stable state is established, the number of packets remains relatively constant, confirming that there are no unexpected occurrences or memory leakage during the model's runtime. Although the actual number of packets may seem large, the number shown also includes packets temporarily stored in the ARQ database. In addition, the simulation being performed is for a population of thousands of clients.

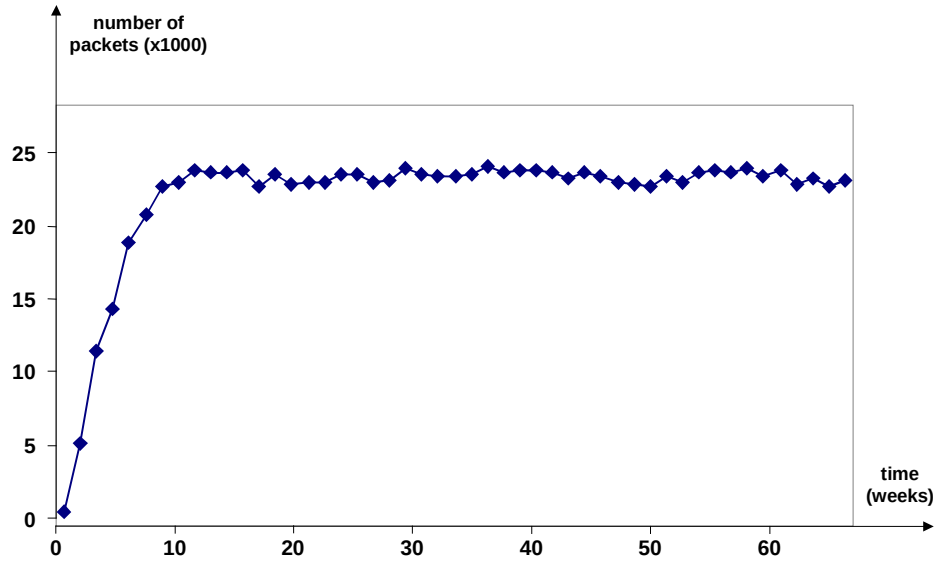


Figure 32: Number of Packets over Time

The results from pilot runs confirmed stable operation over a long period, enabling data to be collected during the steady state. The issue of deadlocks in the system is considered in Figure 33. The number of messages going in and out of client node was collected during time windows corresponding to 100 hours of simulation time (represented with a pair of columns in the figure) for the run period of 20 weeks of simulated time. As the average period when client is 'active' (registered) corresponds to 3-4 weeks, this enabled tracking behaviour of client's node over time, interrupted with several deregistrations, which effectively corresponds to a 'sleep' period. A single pair of columns represents the messages collected within one time window, and

distinguishes client's requests and administrator's response messages. Due to randomness of message generations, different periods of activity are not identical, but the graph demonstrates some aspects of the model's successful operation.

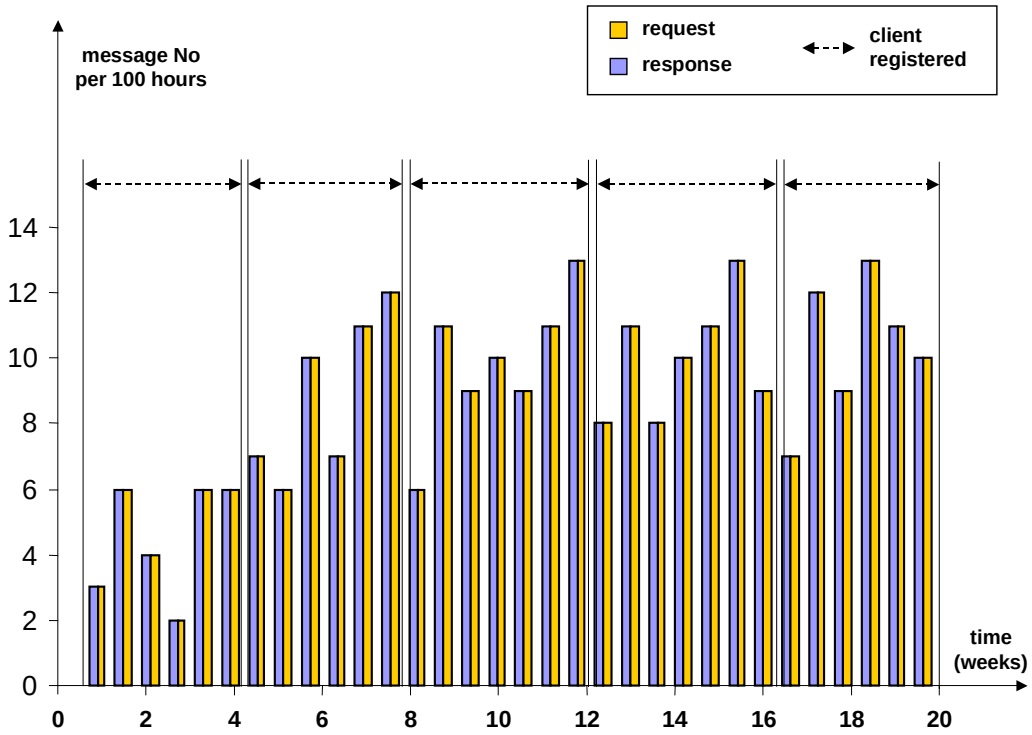


Figure 33: Example of Messages Generated at a Single Client Node over time

Additional verification of the model was performed through a simple experiment, where the number of administrators and client nodes was varied, but in such a way that ratio of number of client nodes per administrator remained constant. For example, experiments were performed for scenarios with 1 administrator and 1000 clients, 2 administrators and 2000 clients, and so on. During all the experiments, inter-administrator communication and remote operation was disabled. The aim was to create isolated partitions in the scenarios with several administrator nodes, and to keep approximately the same number of clients per administrator. This scenario enabled to examine and verify that the client population is equally distributed among the administration nodes. In such a setup, assuming the programming functions operate appropriately, one would expect that parameters for each of the administrator nodes in each partition remain the same. The values of average egress queue size measured at the administrator nodes are given in Figure 34.

The graph shows that the spread of the values increases with the number of administrator nodes, while the average remains relatively unchanged for different scenarios. However this is not easy to observe from the Figure 34 alone. For this reason, a plot of the queue size with increasing

number of client nodes per administrator is given in Figure 35, showing significant growth of the queue size.

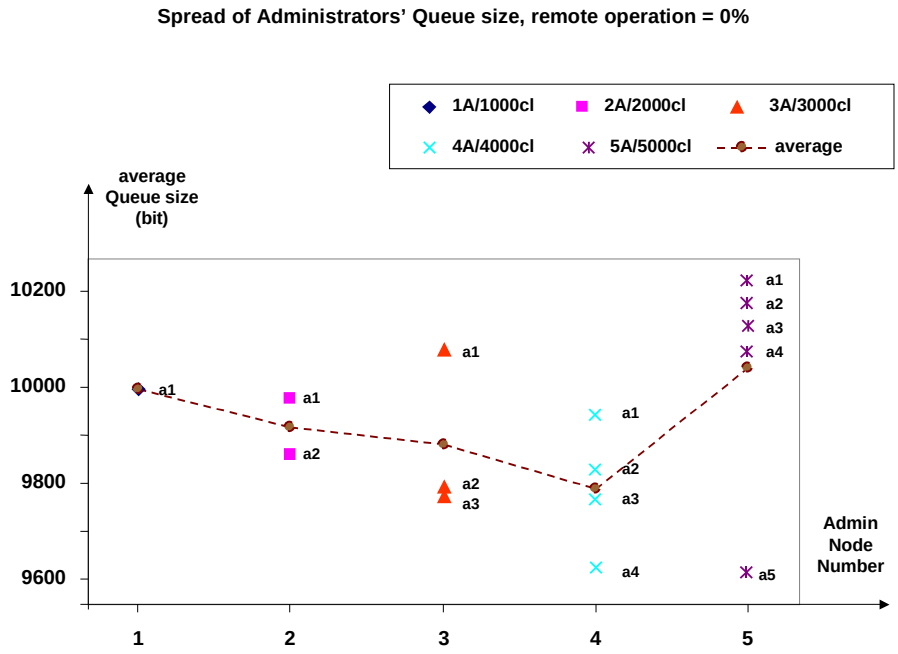


Figure 34: Queue Size with Remote Operation Disabled, with 1000 Client nodes per One Administrator node

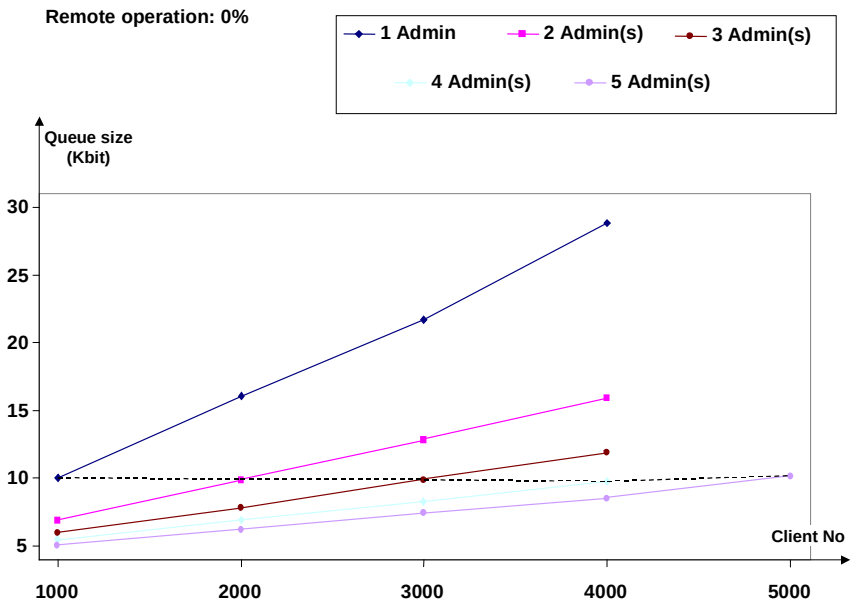


Figure 35: Increase of Administrators' Queue Size as a Function of the number of Client Nodes

The dashed line in Figure 35 connects the average points given in Figure 34. As this analysis was performed for the validation purposes only, further experiments with 5000 client nodes and

different number of the administrators were not performed. However, similar results are discussed in Section 6.2.1.2.

6.1.2 Validation

The simulation model has been validated in terms of accuracy of the collected simulation results, which were also compared against results obtained from the mathematic analysis of the model. In addition, the ARQ (*Automatic Repeated reQuest*) protocol, developed as the part of the model, was validated for its performance and reliability, which is addressed towards the end of this section.

In order to analyse the behaviour of the model, a simple mathematical calculation was performed to estimate the message generation in the system. The author is aware that the approach (to be described) is an approximation of the processes in the simulation model, but at the same time strongly believes that it gives an insight in the behaviour of the model and contributes to a fair prediction of the model's activities. The example that follows looks at the number of generated *register requests* over a period of time, depending on the number of client nodes in the model.

Registration request messages are easy to analyse since their frequency depends only on the average duration of the registration period, and on the average frequency of deregister request messages. Parameters for these two types of messages are defined prior to run-time. The register-deregister pair form a periodic loop which defines basic state of the client node, and its frequency is independent from other events in the model. In addition, these types of requests cannot be rejected by the administrator node, and the model does not support multiple per-client requests of these types. At initialisation, each client node randomly determines the time of its first register request, according to the equation:

$$Init = R_1 + random(r_1) \qquad (1)$$

Where R_1 is constant value in seconds (of simulated time) and r_1 is integer used as a parameter for the invocation of OPNET procedure 'uniform distribution' that generates real number in range $[0, r_1]$.

Once registered (i.e. after receiving an administrator's response), the client node schedules a deregister request for some time in the future, according to:

$$Dereg = D_1 + random(d_1) \qquad (2)$$

Where parameters D_1 and d_1 have similar meaning as above.

In a similar way, when a client node receives deregister notification, it schedules an event for register request:

$$Reg = R_2 + random(r_2) \quad (3)$$

And the process of (2) and (3) is repeated until the termination of the simulation.

Since the function random() gives a uniform distribution based on a specified parameter, the time after which all the client nodes will register for the first time can be approximated to (from (1)):

$$t\_init = R_1 + r_1/2 \quad (4)$$

The average duration of registered and de-registered time can be approximated in a similar way. Given this, the time between two subsequent registration requests (for each client node) can be estimated to the sum of registered and de-registered time (from (2) and (3)):

$$t\_ccl = D_1 + R_2 + ½*(d_1 + r_2) \quad (5)$$

Now, if total simulation time is *T*, it can be expressed as a function of registration time(s):

$$T = t\_init + k*t\_ccl \quad (6)$$

Where *k* is a number of periodic repetitions of register/deregister cycle, i.e. a measure of *how many times each client node registers in the system*. By substituting (4) and (5) in (6), the value for k can be obtained as:

$$k = \frac{T - R_1 - r_1/2}{D_1 + R_2 + \frac{d_1 + r_2}{2}} \quad (7)$$

The value *k* is an estimation of how many cycles of register/deregister are performed by each client node during the runtime *T*. During one cycle, each client will send one register request message. Also, this cyclic behaviour starts only after expiration of initial period *t_init*, during which every client sends another (the first one) register request message. Therefore, total number of register request messages for each client could be estimated as (*k+1*). For the total of *N* client nodes in the simulation scenario run for the time period *T*, the estimated number of generated registration requests *regNo* is:

$$regNo = N*(k + 1) \quad (8)$$

where *k* is given by the equation (7).

Table 10 gives the values used in the simulation runs for the purpose of this validation. Table 11 gives the value of k and number of registration requests calculated according to (8), for different number of client nodes.

Table 10: Parameters from Simulation Scenarios, used for Estimating Number of Requests

Parameter	R_1	r_1	D_1	d_1	R_2	r_2	T
Value (sec)	59200	200000	1814400	604800	86400	86400	9072000

Table 11: Theoretical Values for Number of Requests, Calculated from Equation (8)

k	3.9679458641260796011040869023239				
Number of client nodes	1000	2000	3000	4000	5000
Number of register requests	4967.946	9935.892	14903.84	19871.78	24839.73

For each scenario with different numbers of client nodes, a set of 10 independent experiments was performed, using a different seed for the initialisation of random number generator in OPNET. The number of administrator nodes was kept at 3, and all the experiments were run for a 15-week period of simulated time. The number of register requests, collected for the whole network is shown in Table 12. [53]

Table 12: Simulation values for the number register requests, run for different seeds

Number of clients	Number of request messages				
	seed_27	seed_666	seed_9	seed_15	seed_34
1000	4391	4426	4445	4411	4417
2000	8806	8879	8823	8837	8844
3000	13258	13268	13276	13288	13251
4000	17732	17687	17720	17716	17750
5000	22117	22073	22117	22123	22079

Number of clients	Number of request messages				
	seed_53	seed_81	seed_117	seed_236	seed_312
1000	4442	4409	4423	4399	4402
2000	8868	8833	8841	8879	8861
3000	13219	13264	13255	13275	13243
4000	17672	17639	17711	17721	17689
5000	22099	22160	22083	22145	22064

For the values obtained, a 95% confidence interval was calculated [152], which is illustrated in the example for 1000 client nodes. The estimation of mean value is calculated over the n samples (n = 10) as:

$$\overline{X(n)} = \frac{\sum_{i=1}^{n} X_i}{n} = 4416.5 \qquad (9)$$

The estimation of variance is calculated from:

$$\sigma^2(n) = \frac{\sum_{i}^{n} \left(X_i - \overline{X(n)}\right)^2}{n-1} = 316.5 \qquad (10)$$

[53] The following example describes the typical way the data is collected and evaluated for all the results presented in this thesis.

Finally, an approximate of (1-α)100% confidence interval for estimated mean is given by:

$$\overline{X(n)} \pm \left(t_{n-1,1-\frac{\alpha}{2}} \sqrt{\frac{\sigma^2(n)}{n}} \right) \qquad (11)$$

The value between the brackets is called *half-length of confidence interval* (for small samples of an approximately normally distributed population). The parameter $t_{n-1,1-\alpha/2}$ has a T distribution with n-1 degrees of freedom. For the sample size n<30, the exact T distribution needs to be considered, rather than a standard normal distribution [152]. For validation of the results from 10 experiments considering 95% confidence interval, $t_{9,\ 0.975}$ is needed, and its value is 2.262 (Table A.4 of the Appendix in [152]). Therefore, (11) becomes:

$$\overline{X(10)} \pm \left(t_{9,0.975} \sqrt{\frac{\sigma^2(10)}{10}} \right) = 4416.5 \pm 2.262 \sqrt{\frac{316.5}{10}} \qquad (12)$$
$$= 4416.5 \pm 12.72563$$

The mean and confidence interval results for other values of client population are given in Table 13.

Table 13: 95% confidence intervals, obtained with data from Table 12

Number of clients	Number of requests (calculated)	Number of requests (simulation)	Estimation of relative error (%)
1000	4967.946	4416.5 ± 12.72563	0.288138
2000	9935.892	8847.1 ± 17.33036	0.195887
3000	14903.84	13259.7 ± 13.95226	0.105223
4000	19871.78	17703.7 ± 23.11733	0.130579
5000	24839.73	22106.0 ± 22.78029	0.10305

Confidence interval is a quantitative measure, giving the absolute value of intervals where (1-α)*100% of the experiments produced the value of the measure of interest within the intervals[54]. In order to assess the accuracy of the results obtained, relative deviation of the intervals (with respect to the estimated mean) can be calculated and expressed as a percentage. According to [153], such expression is called *estimation of relative error*:

$$\frac{t_{n-1,1-\frac{\alpha}{2}} \sqrt{\frac{\sigma^2(n)}{n}}}{\overline{X(n)}} * 100\% \qquad (13)$$

Following the example with 1000 client nodes, the value obtained is:

[54] In this context, an 'experiment' is a set of simulation runs, and the 'measure of interest' is the estimated mean.

$$\frac{12.72563}{4416.5} * 100\% = 0.2881384\% \qquad (14)$$

This effectively means that the full-length of the 95% confidence intervals (the above value multiplied by 2) corresponds to less than 1% of the average value obtained in the experiments. The confidence interval results for other values of client population are given in the far right column of Table 13. The confidence intervals for corresponding values from Table 13 are given in Figure 36. [55]

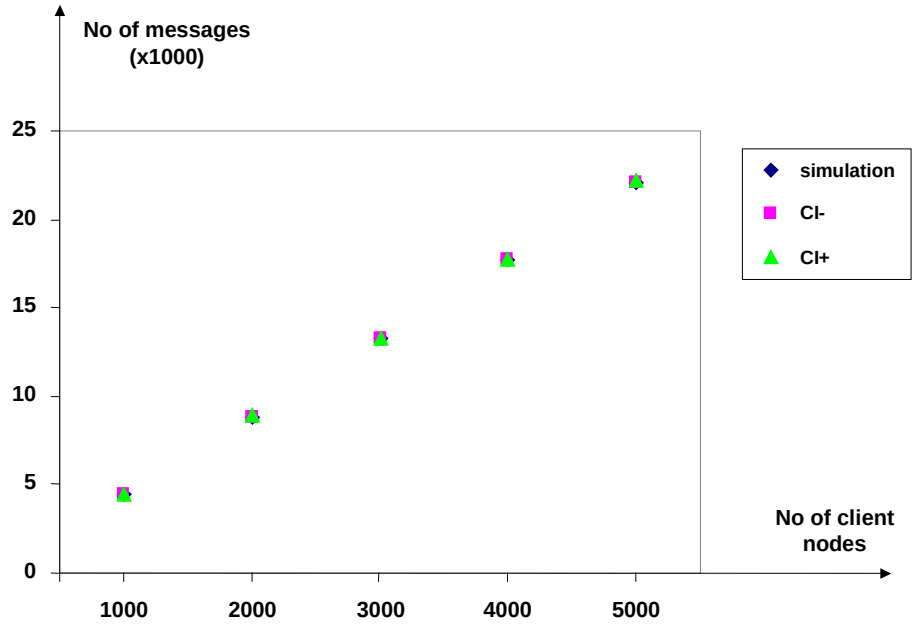


Figure 36: 95% Confidence Intervals for Simulation Results of Number of Register Request Messages

A graph showing the results of the mean of the simulation runs against the values analytically approximated (from Table 11) is given in Figure 37. The simulation results map closely to the calculated values, confirming that messages are generated as expected.

Values for generated number of messages obtained through the simulations are lower due to delays that occur in various parts of the system (processing time, transmission, queuing delay), which has not been taken into account in the analytical model.

[55] The confidence intervals for all the simulation runs were very small so they are ommited for clarity on the subsequent graphs, unless the 95% confidence intervals exceeded 5% of the average value obtained.

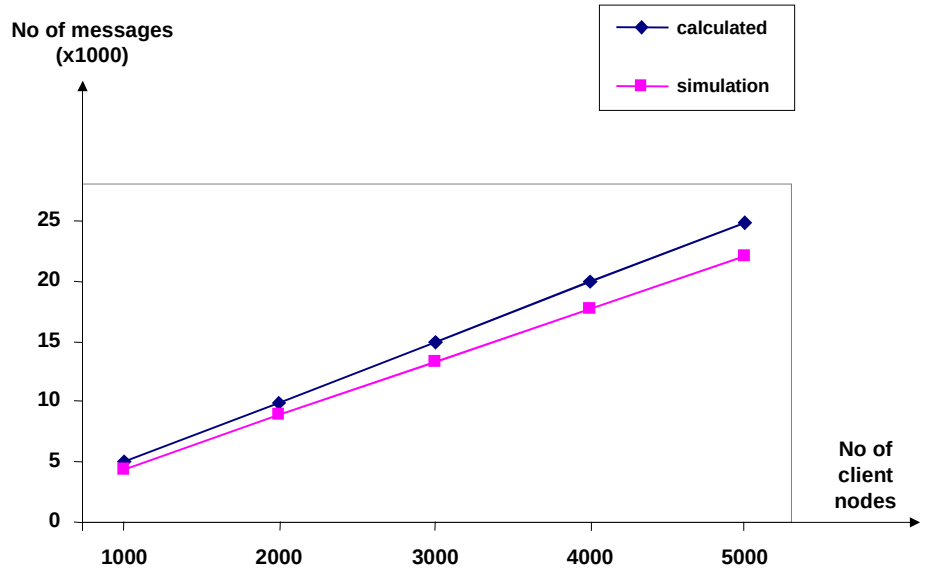


Figure 37: Number of Register Request Messages in Function of Number of Clients: Calculated vs. Simulated values

Both experiments and the above calculation (equations 1 to 8) were performed for various values of parameters from Table 10, and the comparison gave comparable results to those shown in Figure 37.

Due to a stochastic nature of the system, similar calculations would be very difficult to perform for estimating the expected total number of messages in the model. Requests such as create, join and leave are not independent events, and are also influenced by the limits set for the number and size of groups. Also, events such as update messages introduce additional interdependencies, making the calculation even more complex. Nevertheless, the previous discussion increases the confidence in valid functioning of the simulation model.

6.1.2.1 ARQ Protocol

The Automatic Repeated reQuest (ARQ) mechanism is not an essential part of the proposed architecture. However, it has been implemented in order to prevent deadlocks in the system, and for the testing the robustness of the architecture in the presence of packet loss. The approach was based on an Internet standard recommended by IETF [146]. This section presents the validation of the ARQ protocol against two main criterions that such a mechanism should meet: to be (reasonably) reliable, while not introducing a significant degradation of the performance at the same time [146].

The ARQ mechanism is implemented to support re-transmission of signalling messages only. A number of choices in employing ARQ have been standardised, and [146] gives a comprehensive and detailed guidelines for implementing different types of ARQ protocol, including choosing its persistency and retransmission time-out period.

Based on the recommendations given in [146], the type of ARQ used in this research is Sliding-Window, High-Persistence (highly reliable) ARQ Protocol, with 10 retransmission attempts [56] and time-out period of 1000 seconds. In [146], it is suggested that a 'single IP packet should not be delayed by the network for more than 120 seconds', but it also points out that it is in practice difficult to bound maximum path delay on the Internet, and suggests that estimation of the time-out period could be based on the calculation of packet round trip path delay as recommended in [147]. However, in this research ARQ time-out period is set to be constant at 1000 seconds, which is much higher than the observed round trip path delay, and still much lower than the average frequency of the message creation events in the system. In this way, the relative order of the delivered messages in the system can be preserved, even for the retransmitted messages. Regarding the maximum number of retransmission attempts, the value 10 is chosen arbitrarily. For validation, the average number of retransmissions is measured for various values of loss percentage in the system.

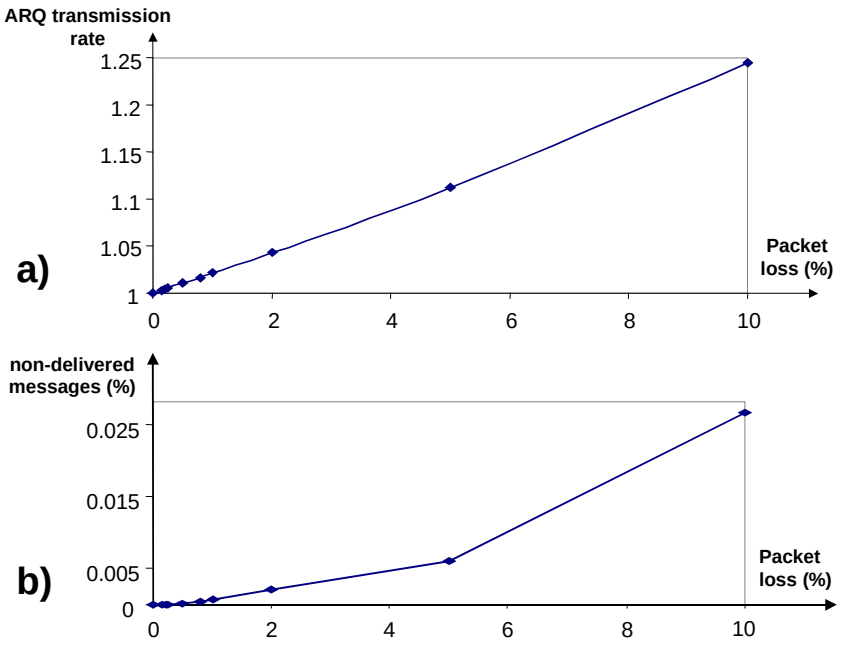


Figure 38: ARQ protocol in the presence of loss: a) Average number of request retransmissions (a value of 1 means that the original request was successful); b) Percentage of non-delivered messages after 10 retransmission attempts

[56] According to [146], Low-Persistence ARQ Protocols normally have 2-5 retransmission attempts.

Figure 38a gives the results from the experiments with 3 administrators and 3000 client nodes. The X-axis represents the value of the packet loss (in percent), and the Y-axis gives the number of transmission attempts for the same request message initiated by the clients. For the same simulation setup, Figure 38b gives the percentage of messages (out of the full number of messages) that have not been delivered even after 10 attempts, depending on the packet loss rate. The results are averaged over the full client population, for the simulation duration of 15 weeks of simulated time. Every point of the graph represents the averaged value from 8 independent experiments.

The actual values from the Figure 38a show that for a 10% packet loss, the retransmission rate is around 1.24, and the number of non-delivered messages is around 150. Such low values give confidence that ARQ protocol operates sufficiently good as implemented, even for very high loss in the system.

6.1.3 Credibility of the Results

A recent publication [151], surveying of a large number of scientific publications, questions the credibility of the reported simulation results due to a lack of detail about the simulator used and the results presented. In order to avoid similar criticism in this research, this section addresses the main issues pointed out in [151], namely the pseudo-random number generator and the description of how output data was collected and analysed.

6.1.3.1 Random Number Generator (RNG)

The purpose of random number generation is to produce a sequence of numbers, drawn from a uniform distribution over the range 0 to 1, which appears to be independent. A good random number generator (RNG) should appear to be uniformly distributed on [0,1] and should not exhibit any correlation between generated numbers. It must be fast and avoid the need for much storage. A random number sequence must be reproducible; this aids debugging.

Once a simulation model has been constructed, it is typically exercised under a number of different conditions in order to characterise the system it represents. While the model itself remains the same, aspects of its environment, or parameters that it uses are varied in order to establish patterns of behaviour or relationships between certain inputs of the system and selected outputs. Stochastically modelled elements depend on a random number source on which to base their behaviour. By ‘drawing’ from the source, these elements can incorporate variability into appropriate actions or decisions taken. By its very nature, it is impossible for a computer program to exhibit genuinely unpredictable behaviour. If a simulation program remains the same for multiple simulation runs, then any change in its behaviour must come from a change in its operating environment (i.e. its input). In particular, even the random

number stream used to implement stochastic behaviour, must be forced into a different mode in order to give different results from simulation to simulation [153],[154].

The mechanism used to select new random number sequence relies on starting the RNG in a different state. This initial state is known as the *random number seed* because it determines all future output of the RNG. For a simulation that incorporates stochastic elements each distinct random number seed value will produce different behaviour and give new results. Each particular simulation can be thought of as representing one possible scenario of events for the modelled system, but no single simulation can be used as an accurate measure of 'typical' system behaviour, since even atypical behaviours (provided they are possible) may be achieved for some random seed. Therefore, a technique that is frequently used is to run the simulation model multiple times while varying the random number sequence. The results obtained from the separate simulations can be combined (usually simply by averaging) to estimate typical behaviour.

Table 14: Comparison of the in-built Visual Studio 6.0 RNG and the Mersenne-Twister RNG

OPNET RNG					***Mersenne-Twister RNG***			
Seed	***Admin 1***	***Admin 2***	***(A1+A2)/2***		***Seed***	***Admin 1***	***Admin 2***	***(A1+A2)/2***
27	9975.628	9859.171	9917.399		*412797393*	9968.137	9700.289	9834.213
666	9705.497	10040.82	9873.156		*1545957324*	9868.764	9878.284	9873.524
9	9837.766	10085.23	9961.496		*476360241*	9815.663	10012.56	9914.113
15	9933.347	9848.071	9890.709		*1924914039*	9840.494	9909.734	9875.114
34	10090.77	9670.59	9880.682		*1349489104*	9912.223	10004.82	9958.52
53	9880.806	9664.294	9772.55		*1736967135*	9649.142	9934.74	9791.941
81	9910.221	9904.364	9907.292		*82415254*	10123.76	10119.83	10121.8
117	10074.11	9825.598	9949.856		*253934125*	9787.522	9978.003	9882.762
236	10183.09	9840.248	10011.67		*49466569*	10117.04	9766.055	9941.549
312	9876.834	10049.41	9963.12		*519204873*	9896.756	10030.9	9963.828
506	10010.64	9936.289	9973.465		*515554563*	9848.214	9972.722	9910.468
808	9904.225	9768.957	9836.591		*110984788*	9780.356	9747.069	9763.713
4	9725.446	10123.03	9924.236		*987635772*	10043.43	10061.29	10052.36
714	9971.887	10132.45	10052.17		*1217781258*	9650.646	9901.2	9775.923
715	9946.39	9557.581	9751.985		*1168126118*	10056.41	9842.67	9949.538
716	9809.375	9948.064	9878.719		*1956146634*	9862.976	9808.178	9835.577
717	9840.14	9888.31	9864.225		*1328467294*	9870.091	9807.607	9838.849
718	10004.32	9735.475	9869.897		*1403673633*	9774.806	9782.49	9778.648
719	9818.46	9748.388	9783.424		*431468653*	9788.96	10095.29	9942.127
720	9814.832	10110.97	9962.898		*1508202608*	9969.936	9950.445	9960.191
mean	9915.69	9886.864	9901.277		***mean***	9881.266	9915.209	9898.238
st. dev	120.983	167.2071	78.04261		***st. dev***	134.0737	121.3118	92.88871
95%CI	56.62115	78.25444	36.52464		***95%CI***	62.7477	56.77502	43.47276
rel. err	0.571026	0.791499	0.368888		***rel. err***	0.635017	0.572605	0.439197
CI-	9859.068	9808.61	9864.752		***CI-***	9818.519	9858.434	9854.765
CI+	9972.311	9965.119	9937.802		***CI+***	9944.014	9971.984	9941.711

In OPNET all of the random numbers draw from a single random number sequence initialised with the value of the seed environment attribute. The random number generator used to create

this sequence is provided by the host computer's operating system and may vary on certain platforms.

In this research, the OPNET random number generation source came from the Visual Studio 6.0 C++ compiler initially. However, in order to validate the random number generator, example simulation results were compared with the results obtained when the experiments were run with the Mersenne-Twister RNG [156]. The choice of applying Mersenne-Twister is since it is widely studied, as well as being a recommended RNG [151]. The results given in Table 14 represent the queue size values for the scenario with two administrator and 2000 client nodes, obtained in 20 independent runs [57]. A comparative plot of 95% confidence intervals in Figure 39 demonstrates that OPNET RNG performs satisfactorily in the context of the examined simulation model.

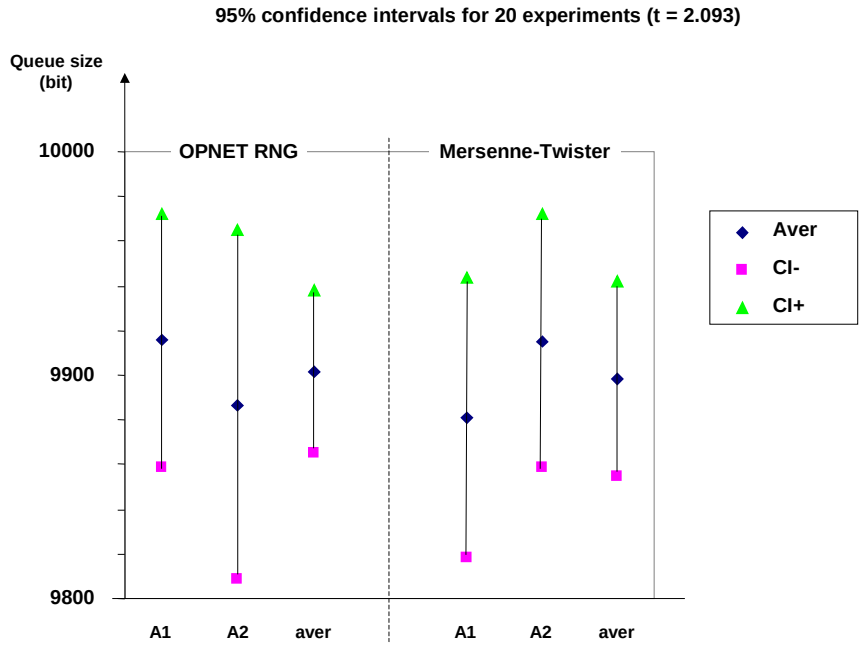


Figure 39: Plot of 95% Confidence Intervals from Table 14

Based on this validation, the results obtained with OPNET RNG are considered reasonable. However, during the examination, experiments using Mersenne-Twister proved to run as equally fast as those with the OPNET RNG. From that point onwards only the Mersenne-Twister RNG was used. Results presented in the rest of this chapter are from both of the experimental phases, but which RNG has been used in particular simulation runs is not indicated as both RNGs were considered to be satisfactory.

[57] This extends the examination explored towards the end of Section 6.1.1 Verification.

6.1.3.2 Output Data Analysis

One of the main parameters used for assessing the performance of the simulation model was queue size of the administrator nodes. Depending on the setup of the experiment, queue build-out was more significant either on ingress or egress queue module. The technique used for analysing the stability of the performance results was the confidence-interval approach based on independent data for steady-state parameters. The simulation model was considered to be in the steady state after 8 weeks of simulation time (Figure 40a supports this assumption). The queue size measurements were sampled from an 8 to 15 week period, following initial-deletion technique [153]. In order to obtain time-plot in Figure 40, 10-hour time-windows were used for initial averaging, therefore giving over 250 points of observation. Each of the values was averaged over the data obtained from 10 independent experiments, forming the estimated mean values shown in Figure 40 [58].

In order to assess the choice of steady state, estimation of relative error was calculated for every time-point from the different experiments. The graph in Figure 40b shows that the estimation of relative error for the calculated average (from Figure 40a) decreases below 1% beyond the chosen steady state of 8 weeks.

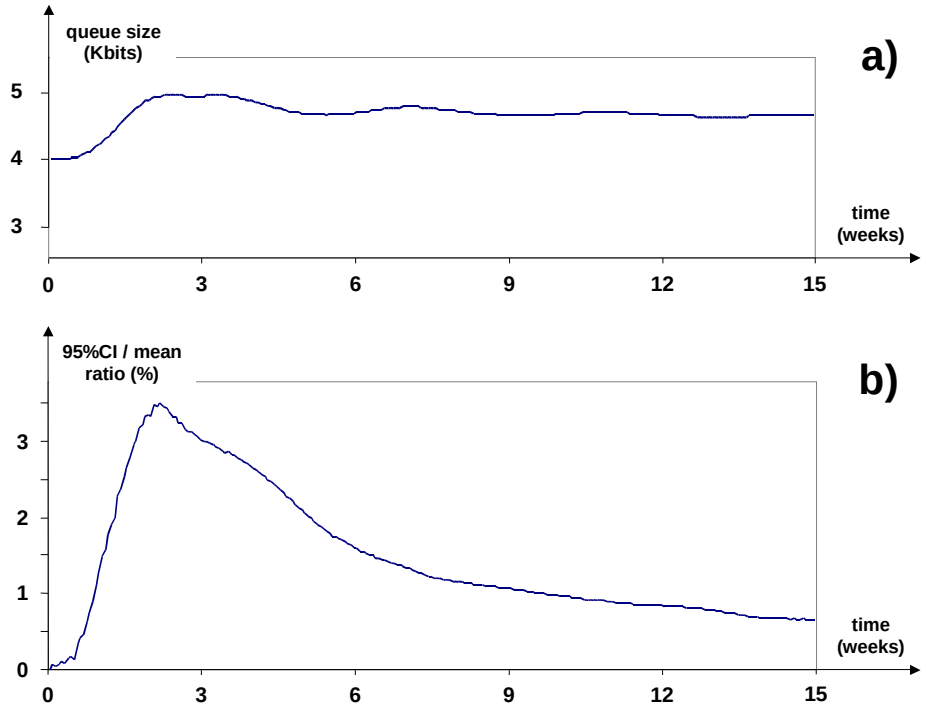


Figure 40: Estimation of Steady State: a) Queue Size in time; b) Accuracy of Experiments

On the graphs presented in the next section (i.e. Section 6.2 Performance Simulation Results), the estimation of steady-state mean for the queue size was performed using the replication/

[58] Figure contains over 250 time-sampling points which are ommited for clarity.

deletion technique [153],[155]. In this approach, the observations used for the estimates are taken beyond the warm-up period *l* (here, 8 weeks). According to [155], a number of n≥5 independent replications of the simulation need to be performed, each of length *m* observations, where *m* needs to be a large number. In this work, ten replications of the simulation were performed for each experiment, using a different seed for the random number generator. The queue size was sampled at every packet arrival, making the number of observations suitably large ($m >> 100{,}000$ over the 8-15 week period).

The previous paragraphs have described the approach taken in choosing the steady state of the model. In some of the experiments transition state lasted for different period of time, which was evaluated using the previous method. However, the number of observations in the steady state remained very large.

6.1.3.3 Choice of Simulation Parameters for Initial Registration

As already stated, most of the parameters used in the simulation model are based on the measurements data of analogous processes, as reported in the literature. Since the aim was to assess the performance of the proposed architecture while the system is in steady state, transition processes at the initialisation were not of the particular interest. However, since each run of the simulator starts from an empty network state, followed by the initial registration of a large number of client nodes, its potential implications were examined.

The behaviour of the system described in this section was examined during the building and validating phase of the simulation model. One of the phenomena observed is that the choice of parameter determining the timescale of initial clients' registration strongly impacts subsequent activities and patterns of behaviour in the system, as well as the output results. Initial registration is time period at the end of which all the client nodes have sent the register request message (at least once). In two sets of experiments this value was chosen to be either very small or very large, respectively. The output results were very different: in the first case, significant oscillatory behaviour in the system was observed, whereas in the latter case the steady state was reached much faster. Based on this observation, a more suitable parameter was chosen for the performance evaluation experiments. The rest of this section describes the problem, analysis conducted, and the conclusions obtained.

The graphs in Figure 41 show results from the experiments with 1000 clients and three administrator nodes. Once registered, clients stay in that state for approximately 3-4 weeks, and re-register after a very short period of non-activity (i.e. clients remain deregistered for approximately 1-2 days). In those two scenarios, all the parameters are kept the same, apart of the period of initial registration, during which all client nodes become registered.

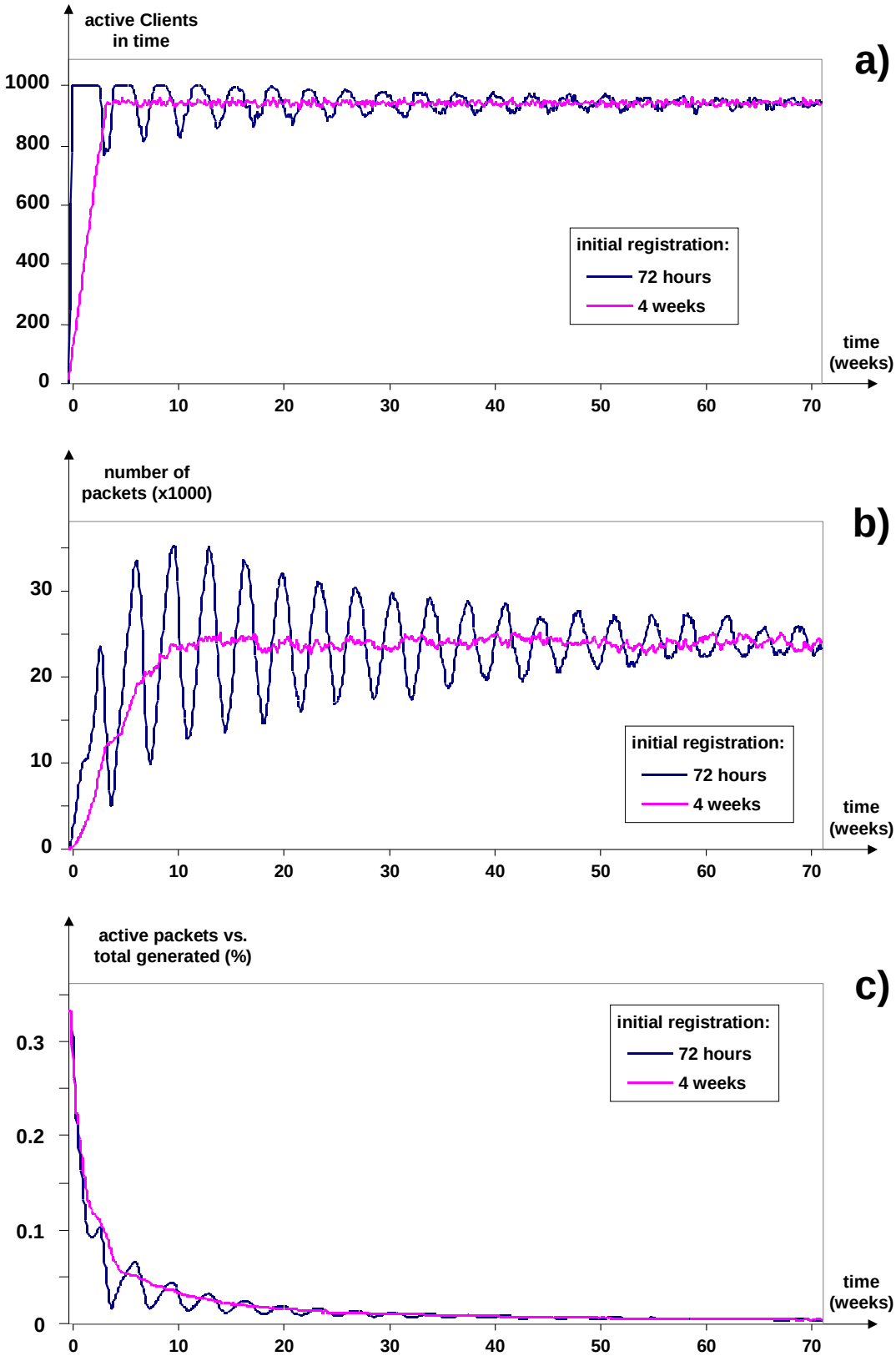


Figure 41: Impact of Initial Registration Period on Processes in the Model: a) number of registered Clients; b) active packets; c) percentage of active packets

The first value (72 hours), being very short, influences that all the client nodes perform register/deregister request in very short time period, causing oscillations both in the number of

currently registered clients and number of packets generated within the 10 hour time-window (used for collecting the data). Due to certain randomness in the duration of registered/deregistered period [59], the oscillations decline over the time, as the randomness in the model increases. However, it causes flat curve of a steady state to be reached after very long time.

The second value for initial registration period (4 weeks) is for purpose chosen to correspond to the estimated value of cyclic behaviour (i.e. from one to the next client's registration). In this way, randomness is achieved from the beginning of the simulation runtime, causing much flatter characteristics of the graphs.

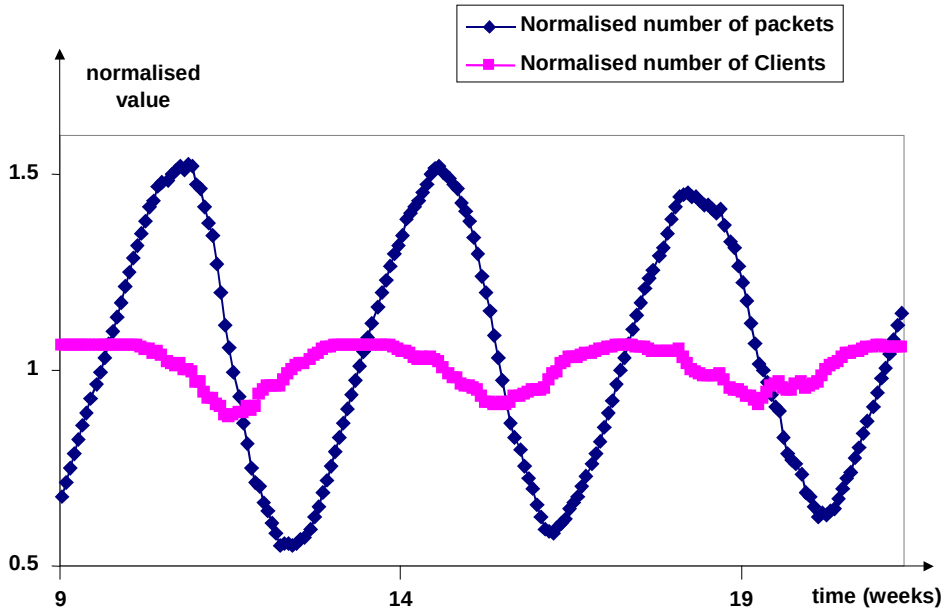


Figure 42: Oscillations in the System: Traffic Volume vs. Number of Clients

The period of the oscillations in Figure 41 corresponds to the duration of the average cyclic period between two subsequent registrations. The amplitude is proportional to the initial registration period, and the lessening of the oscillations in time depends on the level of randomness in the duration of registered/deregistered period (i.e. parameters d_1 and r_2 from equations (2) and (3), respectively, in Section 6.1.2 Validation). Another point to note is that the peaks for the maximum number of active packets in the system coincide with the maximum rate of the clients' deregistration (the steepest part of the curve of active clients, at its decline). This can be explained: at this point there is still a large number of groups and registered clients, and the number of packets due to the clients' activity and the administrators' updates are augmented

[59] This has been explained through analytical example in Section 6.1.2 Validation.

with the additional deregistration requests. As this may not be very easy to observe by comparing Figure 41a and Figure 41b (due to a figure resolution), please refer to Figure 42. The values are normalised with the corresponding average values. The time axis indicates which part of Figure 41a and Figure 41b the snapshot represents, with the respect to the overall simulation time.

The results with different client population show the same general shape, only with different absolute values and the magnitude of oscillations. Based on this examination, 4-week period of the initial clients' registration is chosen for other experiments.

6.2 Performance Simulation Results

The simulation model was essentially designed from scratch, based on the architecture developed during the research phase. The architecture design was an iterative process of refinements in order to meet security requirements of the model, whilst achieving satisfactory performance. In this sense, validation of the model represented a particular challenge, as there was no appropriate benchmark of a comparable system found in the literature. In order to ensure that the model is a true and accurate representation of a potential real system, most of the parameters in the simulation model are based on the measurements data and experience reported about systems that this architecture aims to resemble, facilitate and improve upon. To the large extent, the motivation and approach to this has been dealt with in the previous chapter (see Section 5.5 Traffic Modelling). This chapter comments on the various simulation results performed under the range of conditions. The main parameters used for the assessment, and the way they were collected, are:

- *Size of ingress/egress administrator queues*: at each packet arrival, the queue size is recorded. At the end of the simulation, the accumulated value is divided by the number of observations in order to obtain the average value.
- *Delay at ingress/egress administrator queues*: collected in the same way as the queue size.
- *Packet round trip time*: simulation time when packet is sent was subtracted from the simulation time when the response was received. The accumulated value over all packets transferred during the runtime was divided by the number of observations in order to obtain the average value.
- *Processing delay*: delay caused by each event is accumulated over the runtime for each node. From this, the average value is derived when the accumulated value is normalised by the total simulation time; in general, the results are presented as processing time per 1 minute of runtime.

6.2.1 General Evaluation of the Architecture

6.2.1.1 Functionalities of Signalling Protocol

Some of the basic functionalities of the architecture are demonstrated through analysis of different types of messages generated during the runtime. In general, all types of signalling messages can be divided in two categories (Figure 43):

Hierarchical messages, transmitted between clients and their administrators. These include all the requests initiated by the clients, and responses sent back by the administrators. These also include all the update messages imitated by the administrators for the members of different groups.

Peer-to-peer messages, communicated between different administrators (group managers) for purpose of the management of remote clients. These include requests for remote join/leave, and corresponding responses. Also, part of the group update messages (sent to remote clients via their administrators) fall into this category.

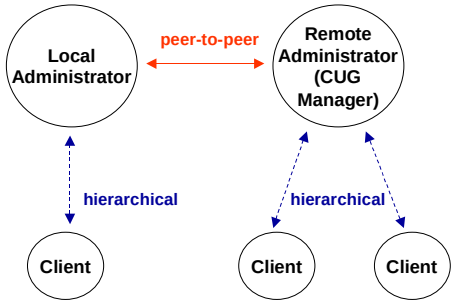


Figure 43: Overview of Signalling Messages in the System

All of the above are illustrated in the subsequent diagrams. The simulation setup comprised three administrator nodes, and the probability of remote join/leave operations was set to 5%. The number of client nodes was varied between 1000-5000, and changes in the number of messages exchanged were observed. The other relevant operational parameters are described with each of the figures separately.

Figure 44 shows different types of signalling messages in the system. Message statistics are recorded at the client nodes, and summed over the whole population. For all types other than updates, only the client's request are shown; however, the number of the generated requests was exactly the same as the number of the responses received, since experiments were performed with packet loss set to zero (the effects of the packet loss are discussed in Section 6.2.2.5 Robustness). The number of update messages shown represent the number recorded as received by client nodes (this has been compared with the number generated by the administrators, and it was exactly the same).

The general observation is that number of messages increases linearly with the growth in the client population.

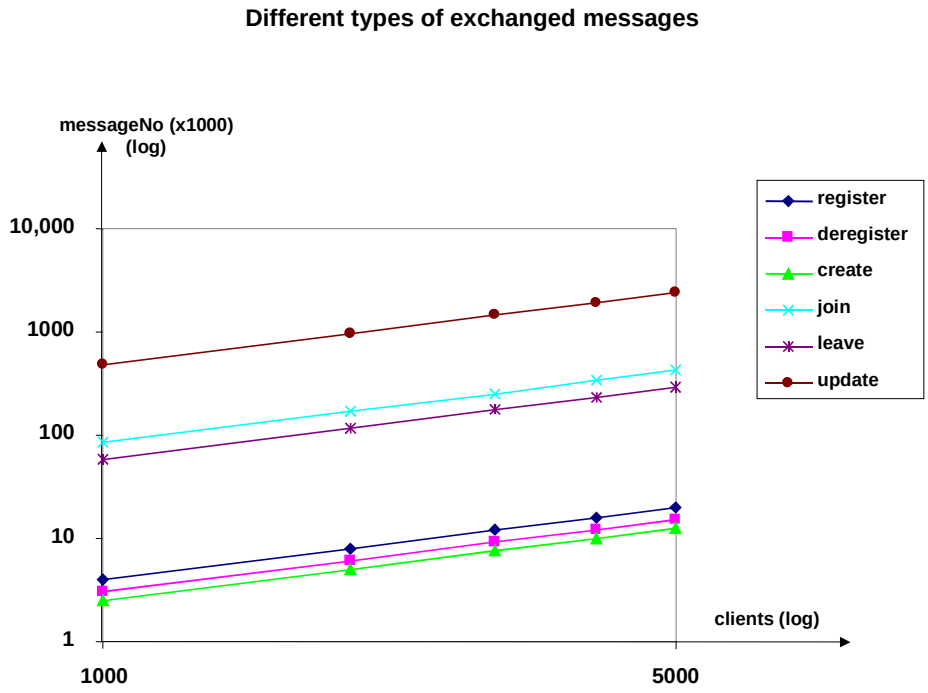


Figure 44: Types of Signalling Messages

Number of update messages greatly exceeds the number of any other message type. This is because the period of updates was set to 24 hours (of simulated time), which directly impacts how frequently update reports are being generated by the administrator(s). This is examined in more details in Section 6.2.2.4 Frequency of Periodic Updates.

The number of generated register requests slightly exceeds the number of deregister requests, and similar results can be observed for join/leave pair. For register/deregister pair this is expected – although every registration triggers deregistration, at the moment of the runtime termination there is more register requests generated since those started first. However, if simulations were run indefinitely, these two numbers would mach. The same stands for join/leave pair: upon successful joining, exactly one leave message is scheduled to happen.

An interesting observation is that the number of create group requests contributes the least in terms of traffic volume. As already explained in Section 5.5.2 Signalling Messages, both create and join requests are constrained with the number of groups a client is maximally admitted to be a member of. Once that point is reached, previously scheduled requests are ignored, and the new ones scheduled for some time in future. Therefore, these events will generate requests only if a client has left a group in the meantime. Since the frequency of join requests is much higher than

of those for create group, client join requests are predominant compared to the number of create request messages generated.

In Figure 45, the relationship between hierarchical and peer-to-peer signalling messages is shown. For clarification, message types described as “client->LA” and “LA->client” do not represent all the messages exchanged between clients and their local administrators, but only those that are forwarded to the remote group managers.

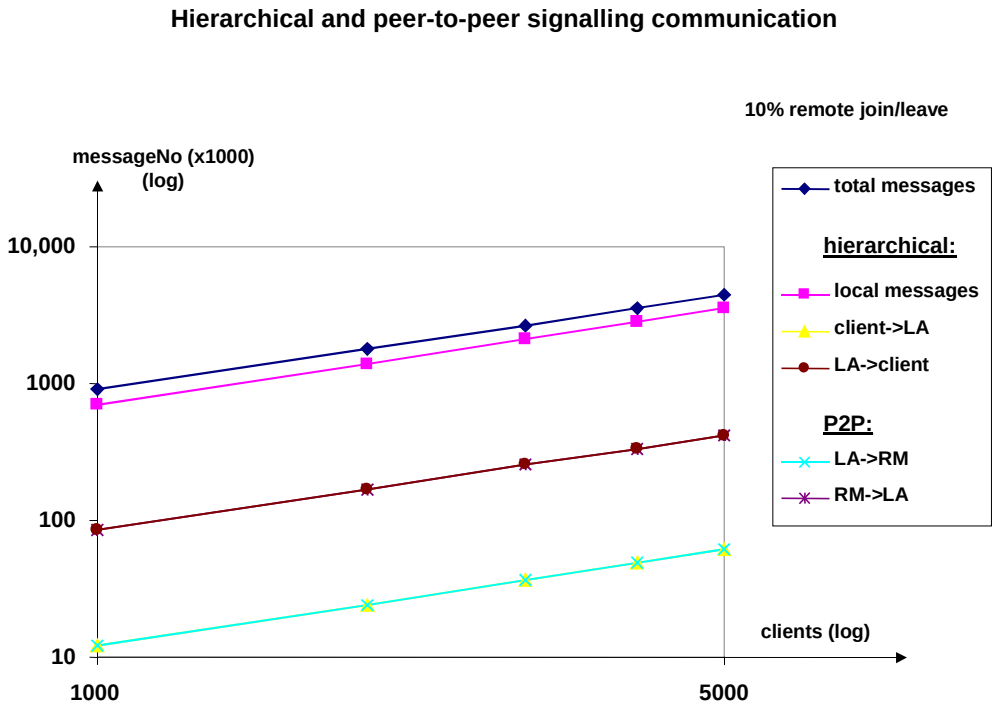


Figure 45: Hierarchical and Peer-to-Peer Signalling Messages

The number of remote requests generated at client nodes matches the number forwarded by their local administrators (LA) to the targeted group managers. Similar to this, the number of responses generated at the ‘remote’ group managers (RM) corresponds to the number of responses forwarded by administrators to their local clients. However, the values for the number of generated response messages are higher than those for the requests. This is due to a portion of group update messages (generated by a CUG manager for remote group members), which follow the message flow in the direction of response messages. The direction from client to remote group manager includes only client’s join/leave requests; in contrast, the direction from remote group manager to the client includes corresponding responses, as well as periodic updates for every group the remote client is a member of. The previous elaboration also explains inter-dependencies between the peer-to-peer messages at the administration level and certain types of hierarchical messages (i.e. the remote ones). In addition, similar to the diagram in

Figure 44, logarithmic growth of the traffic volume with the increasing number of clients can be observed.

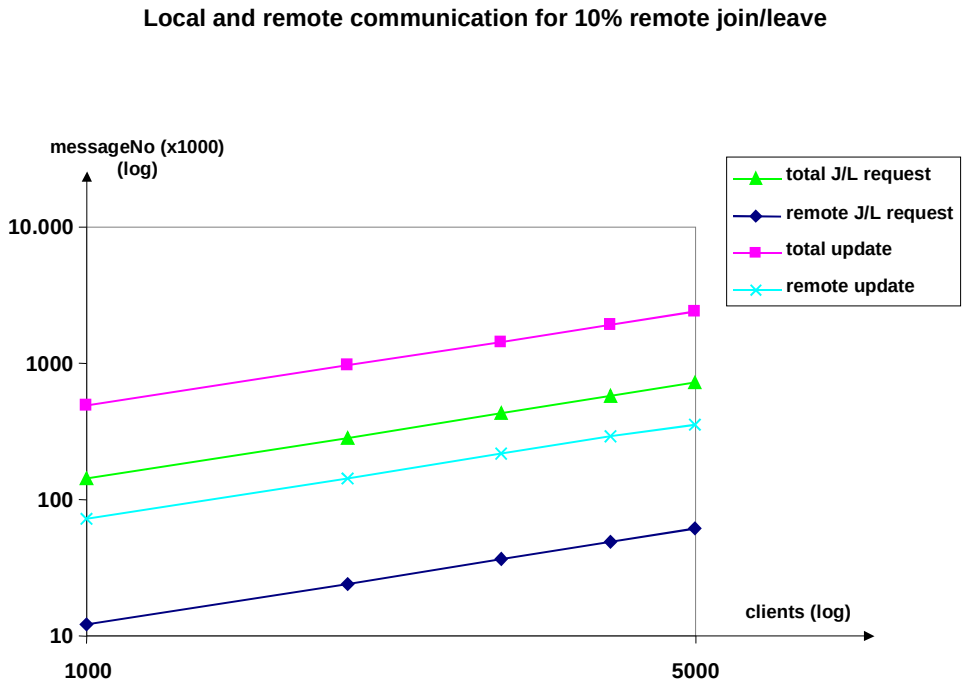


Figure 46: Local vs. Remote Communication in Multi-Administrator Environment

Finally, the total number of the messages (comprising all the different types: local/remote requests, responses and updates) is given for a benchmark comparison. It can be observed that the remote communication contributes approximately 10% of the overall traffic volume. This result is consistent with the simulation model, as the probability of remote operation was set to exactly 10 % for this set of experiments. This is presented in more detail in Figure 46. The values that should be compared are total vs. remote update (red and blue lines), and total vs. remote join/leave request (yellow and navy lines). The latter represent the sum of join and leave requests (i.e. "total J/L request" is also shown in Figure 44, but separated into two different values for join and leave).

6.2.1.2 Performance of Signalling Protocol

Results in this section look at the performance of the signalling protocol for the same simulation setup as in the previous section. The aim is to assess the general behaviour of the architecture. The processing delay of nodes was limited to only those introduced by data structure manipulation. Time needed for encryption and authentication mechanisms was not included; that examination is dealt with in Section 6.2.2 Evaluation of Architecture with Encryption and Authentication, onwards.

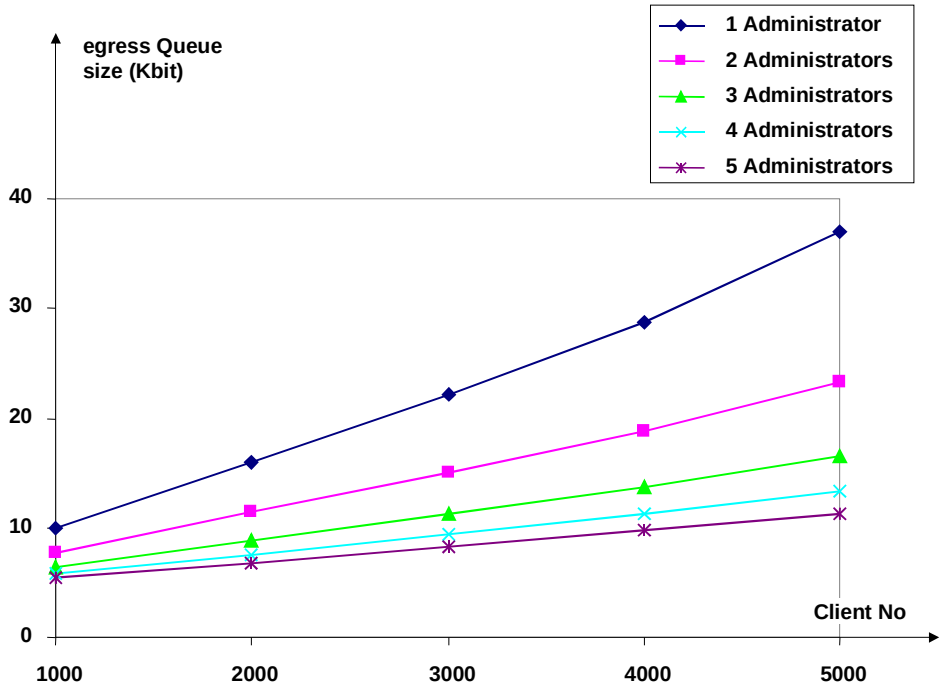


Figure 47: Egress Queue Size at Administrator vs. Number of Clients, for Low Processing Delay

The graph in Figure 47 gives the egress queue size of the administrator node. The queue size grows proportionally with the number of client nodes. This is caused by the increased number of generated response and update messages. However, when more administrators are introduced, the queue size drops as the client population served is equally shared among the administrator nodes.

Figure 48 shows the ingress queue size at the administrator node. It shows the opposite result of the previous observation, when more administrators are introduced. In fact, the queue size for experiments with one administrator node is flat, whereas the increase occurs only in multi-administrators scenarios. This suggest that the growth of the ingress queue does not occur due to the number of clients served and messages generated, but due to some other phenomenon. Detailed analysis of the model gives the following explanation:

In general, the number of messages received at the administrator node does not create enough traffic to cause build-up of the ingress queue (such as the basic scenario with 1 administrator). Multi-administrator scenarios suffer from the additional traffic caused by inter-administrator signalling communication, and this is the reason for queue build-outs. However, it is not the amount of traffic, but the traffic patterns that causes this. As given in Figure 45, the update messages sent to remote group members cause the most of inter-administrator communication. The significant property of update messages is that they are broadcast instantly to the whole group, separated in time only by the processing delay needed to perform certain operations.

Therefore, the administrator nodes in multi-administrator scenarios periodically receive a large number of messages with very small inter-arrival time, which causes growth at the ingress queue. This does not happen in the scenario with one administrator, since the only messages received are clients' requests, which are much less frequent [60]. This observation is very important since it becomes more pronounced once delays for security functions are introduced (this is addressed in Section 6.2.2.1 Scalability).

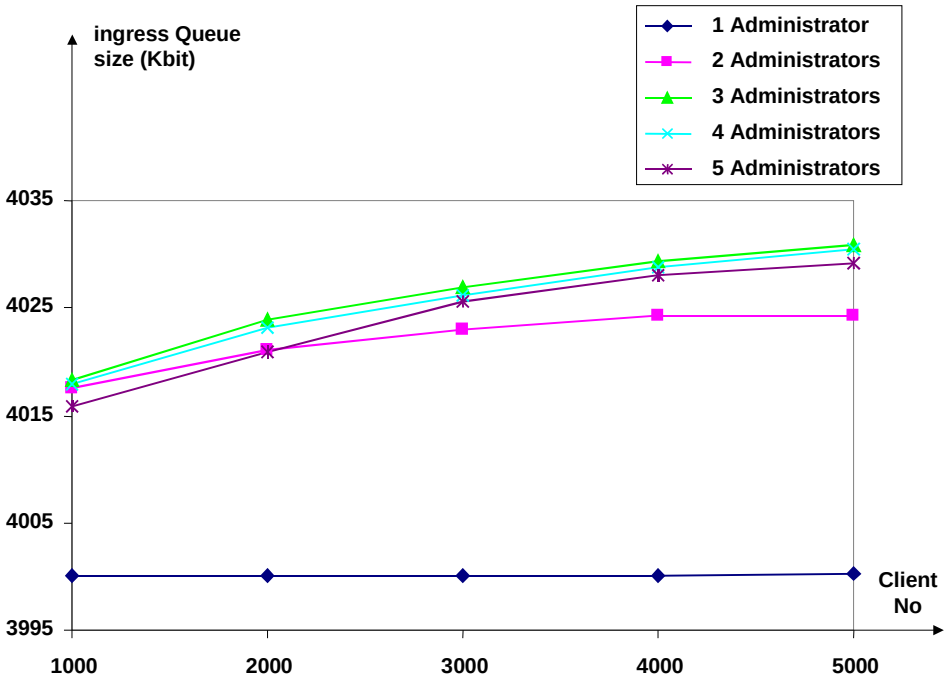


Figure 48: Ingress Queue Size at Administrator vs. Number of Clients, for Low Processing Delay

In addition, as observed in Figure 48, the ingress queue size for the scenario with two administrator nodes is slightly lower compared to the values with three or more administrators. This also contradicts the conclusion made for the egress queue, that the queue size should drop when more administrators are introduced. Again, the explanation is due to inter-administrator communication: for two-administrators, inter-administrator communication at the administrator node can be received only from one source (the other administrator). For other scenarios, this peer-to-peer traffic can be received by more than one source, potentially causing several 'update streams' to be received at the same time. Figure 48 also shows that this does not create significant difference if number of administrators is higher than two.

[60] This is also the main reason for growth of egress queues, as observed in Figure 47.

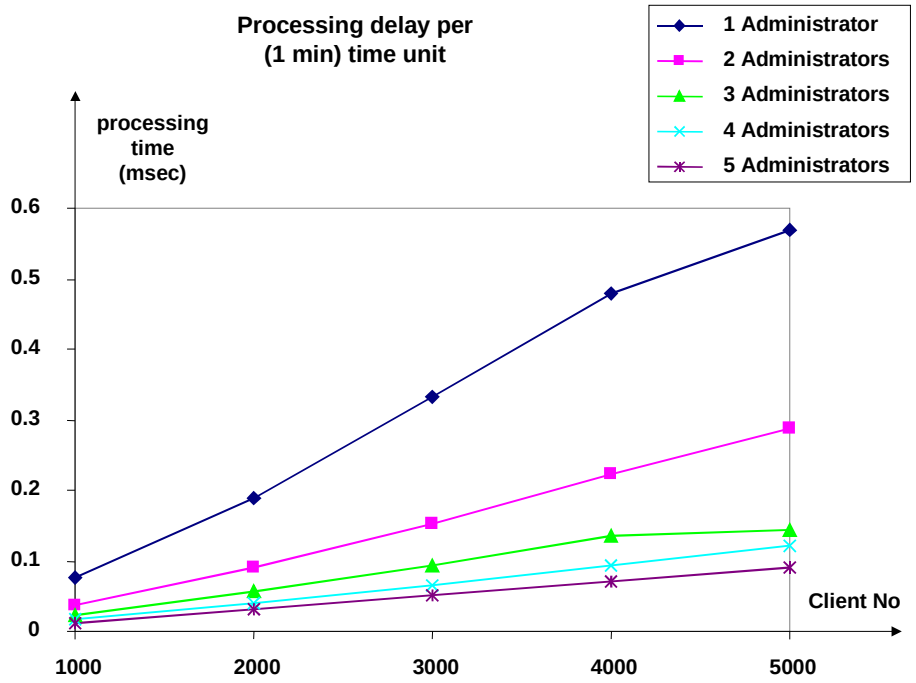


Figure 49: Processing Time at Administrator vs. Number of Clients, for Low Processing Delay

It is also important to note here the significance of processing delay. In general, the model is developed in a way that processing delay (calculated for each action performed) blocks the ingress queue, therefore shifting the burden (even for the broadcast messages) from egress to ingress queue. However, the experiments discussed here do not suffer a lot from this phenomenon: The processing delays are calculated only for data structure manipulation, and are very small compared to the ones for security procedures (e.g. compare values in Figure 49 with those in Figure 53). Effectively, the ingress queue is not blocked for long enough to cause significant queue build-out. Also, the update messages, being separated for insignificant time values, arrive at the egress queue with very small inter-arrival times, causing the main burden at the egress queue size. A more detailed examination that takes into account encryption/ authentication delays gives somewhat different results, which are discussed from Section 6.2.2 onwards.

The processing delay shown in Figure 49 demonstrates more intensive communication in the scenarios with fewer administrator nodes. This is as expected since the delays depend on the search through the list of groups and registered clients.

In addition, Figure 50 shows how the packet round trip time changes with the number of administrator nodes. It is constant in any scenario for 2-5 administrator nodes, since the communication path never exceeds two hops (in each way). The basic transmission and propagation delay (which is set to 100msec each way), is augmented with the delays occurring

at queues and the processing delays. There is practically no difference when the client population changes, since the processing delay is in general negligible compared to the message transmission time. Round trip time increases in multi-administrator scenarios, due to a portion of messages forwarded as remote requests. Compared to the one-administrator scenario, the growth is not significant (bearing in mind that transmission is modelled as taking 100ms), since not all the messages travel two hops, but only 5% of join/leave requests.

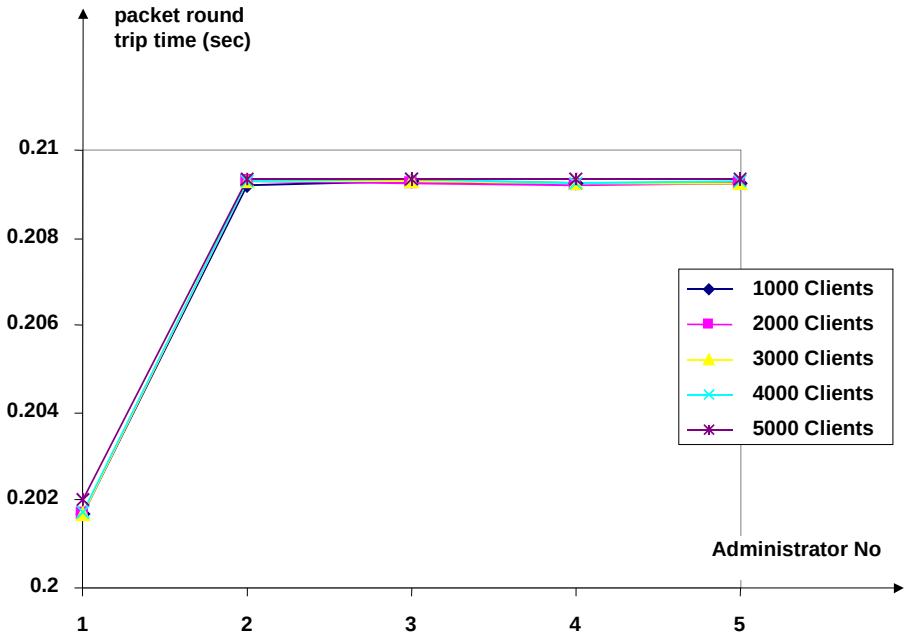


Figure 50: Packet Round Trip Time in Function of Number of Hops

6.2.1.3 Choice of Revocation Mechanism

One of the early experiments was performed to evaluate various proposed mechanisms for updating of group status and revoking group certificates. As described in Section 4.4.4.2 Group Membership Revocation, three different mechanisms were proposed:

- Administrator update – whenever a member leaves a group, the administrator in charge broadcasts the message informing the group.
- Periodic update – at defined intervals (different for each group managed), the administrator broadcasts update message to the group.
- Appointed member update – for each existing group, a member is chosen to perform this function on behalf of the administrator. Whenever a member leaves the group, the administrator compiles the list of current members and sends that message to the appointed member, who then broadcasts it to the whole group. If the appointed member is the one to leave, the administrator chooses another participant in the group.

As the simulation parameters used in this experiment somewhat differ from those in the previous section, it was not possible to cross-compare the results. However, all three schemes are tested and the results for egress queue size at the administrator node for different scenarios are presented in Figure 51.

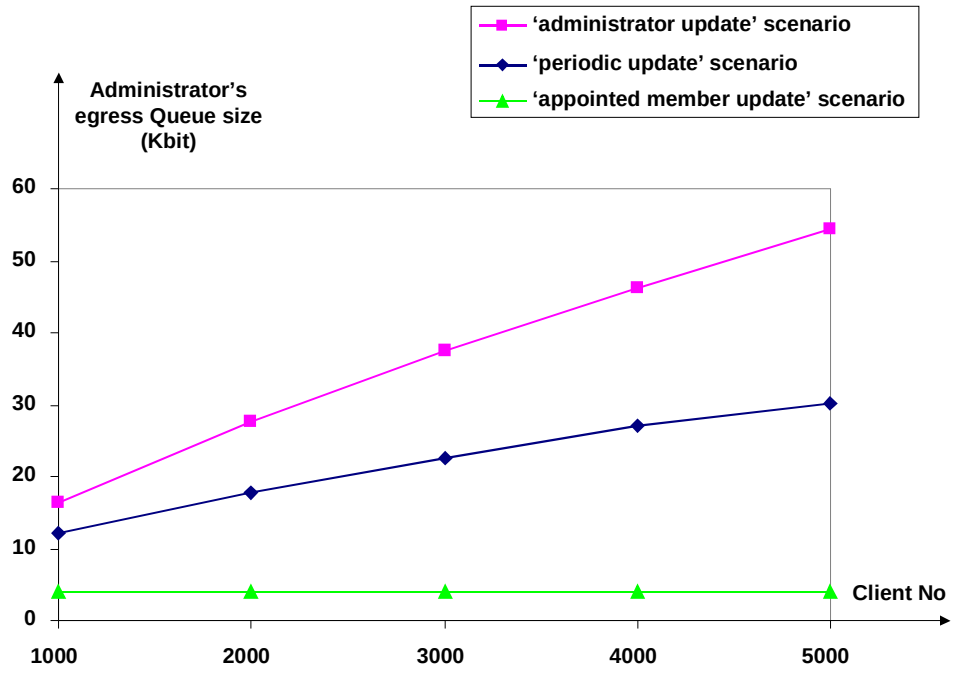


Figure 51: Performance of Administrator Node in Different Revocation Scenarios

The average user group joining/leaving period was 12 hours (simulated time), and the probability of joining a remote group was set to 10%. The maximum size of the group was set to 100 users. Each administrator was limited to managing of 500 simultaneous groups, and each user was allowed to be a concurrent member of 20 groups at most. For the results presented, the experiments were performed with three administrator nodes, a varying user population from 1000 to 5000, and observing the size of egress queue of the different entities.

Figure 51 shows a linear increase of the queue size relative to the client population for the scenarios with instant and periodic administrator updates. For a given update period (set to 100 hours of simulated time), the administrator's queue size is significantly lower when compared with the instant response approach. For the scenario with appointed member, the administrator's queue does not suffer from the growing client population, as expected. However, the burden has now been shifted to the corresponding appointed members, as can be seen in Figure 52. For the purpose of the experiments presented here, the scope of the appointed member role has been limited to a single group that it participates in; i.e. no client is allowed to be the appointed member in two or more groups. Figure 52 also shows the queue size of the ordinary group

member, which remains unchanged. Exactly the same results are obtained in all the other results; i.e. clients' queue sizes do not grow in any of the experiments where clients only generate requests and receive the responses.

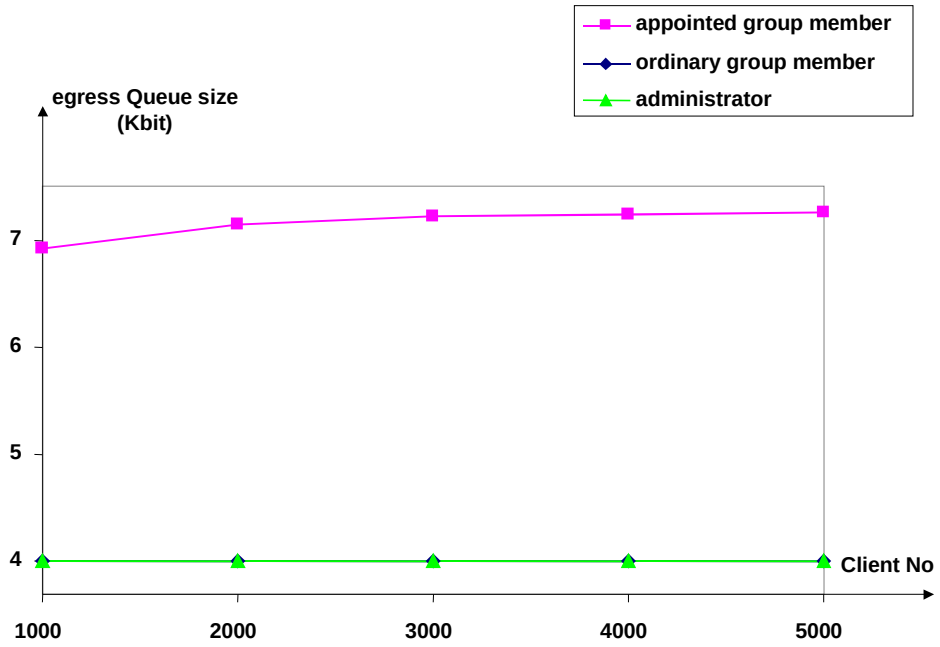


Figure 52: Comparison of Administrator's and Members' Queue Performance in 'Appointed Member Update' Revocation Scenario

From a scalability point of view, the scheme with the appointed member shows the best performance. At the same time, it is the least secure, due to the fact that administrator must delegate a certain amount of trust to this user. If the appointed member was to become compromised, it could operate maliciously until the administrator directly intervenes and issues a new group membership list and updates the list of revoked certificates. However, this scheme may be useful for relatively non-critical applications, such as advertising of administrator's services or other groups of interest, broadcasting general information, such as company news, etc. Further discussion on the security of various schemes is dealt with in Section 4.4.4.2, as well as in Chapter 7 **Error! Reference source not found.**.

Finally, the results for periodic update scenario presented in Figure 51, show the administrator's performance for only one chosen value for the periodic update frequency. It is very likely that the value chosen will impact the performance, as it directly determines the number of generated messages in time. This effect has been explored further and the results are given in Section 6.2.2.4 Frequency of Periodic Updates.

6.2.2 Evaluation of Architecture with Encryption and Authentication

This section discusses the performance of the architecture whilst incorporating values for performing security functions, as described in Section 5.5.1 Processing Delays. As a result, much higher processing delays are observed, which influences other output data.

6.2.2.1 Scalability

The setup for these experiments consist of a number of scenarios where either the number of client nodes or the number of administrators were varied. For the purpose of multi-administrator experiments, the probability of remote join/leave operations was set to 5%. In all the experiments, the frequency period of the group update messages was set to 1 week of simulated time (the experiments with different values of update frequency are examined in Section 6.2.2.4 Frequency of Periodic Updates).

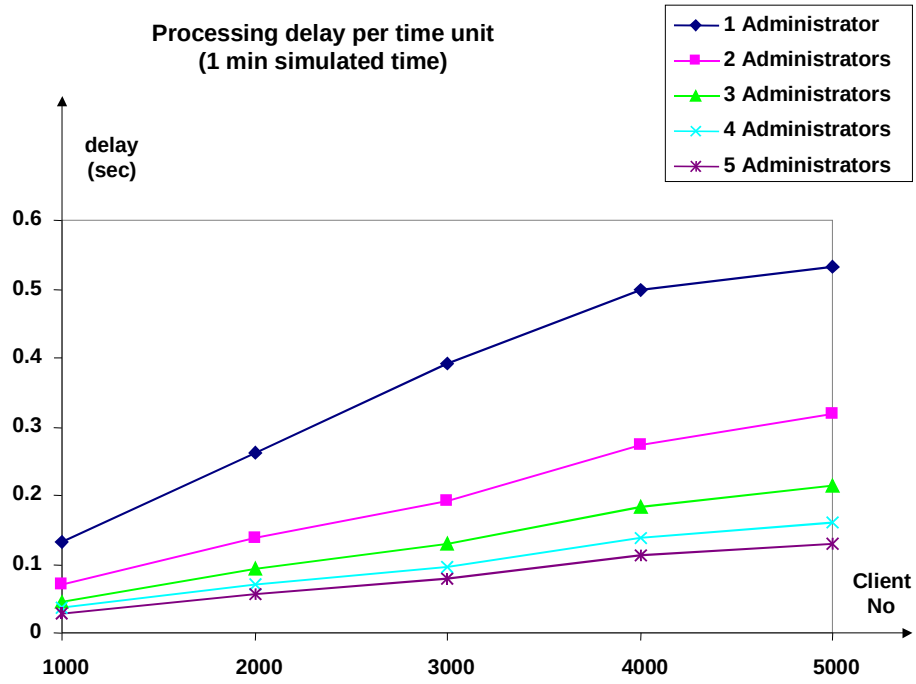


Figure 53: Processing Delay at Administrator vs. Number of Clients

Figure 53 gives the values for processing delays at administrator node, for simulation set-ups with different number of client and administrator nodes. Compared to Figure 49, the values are much higher. This is the expected result – values for modelled security functions are comparable to tens of milliseconds (per function), whereas those for data structure manipulation are comparable to microseconds. The processing delay does not depend anymore on only the number of messages processed, but also on the message type. For example, the most 'costly'

operations are register, create/join a group, since these require a creation of the certificate [61], in addition to other authentication and encryption functions. On the other hand, messages forwarded by an administrator as a part of remote communication are modelled with symmetric encryption (instead of asymmetric), and therefore require much less processing time (this is examined in detail in Section 6.2.2.2 Impact of Remote Join / Leave).

The results in Figure 53 show that the processing delay grows with the client population (as more messages and encryptions performed), but it drops when more administrators are introduced (since the client population is equally shared).

The effect of processing delay is to some extent contradictory: higher values indicate that more messages are processed within (and sent from) the node, which should result in the growth of the egress queue. At the same time, higher values mean that the ingress queue is 'blocked' for longer time, meaning that the growth should appear at the ingress queue, allowing time for the 'flushing' of the egress queue.

Diagrams of administrator's queue sizes are given in Figure 54 and Figure 55. The above effect of processing delay, combined with the effect of decreased message inter-arrival times in multi-administrator scenarios, represent a particular challenge to obtaining a meaningful interpretation of the data gathered.

Figure 54 shows the ingress queue size at the administrator node. Similar to Figure 48, the queue size for a single administrator does not grow. Even though the processing time for a single administrator is very high, messages arrive at the ingress queue with high inter-arrival times, which allows any message in the queue to be served before the next one arrives. As already explained, this is not the case in the multi-administrator scenarios due to update messages sent to remote group members, as a part of inter-administrator communication. However, unlike Figure 48, the queue size is directly proportional to the number of administrator nodes. The stream of update messages to remote group members, which causes build-out of ingress queue, is additionally delayed at the queue due to the significant processing time at the administrator node. This time depends on the number of client nodes served by the administrator, which is higher if a client population is shared between fewer administrator nodes, as shown in Figure 53. Also, the total number of messages processed per administrator is shown in Figure 56a.

[61] Certificate creation is modelled as the additional digital signature (i.e. very costly private-key operation). This effectively corresponds to the signing of the certificate, while 'filling in' the certificate fields is assumed to be included in the database search.

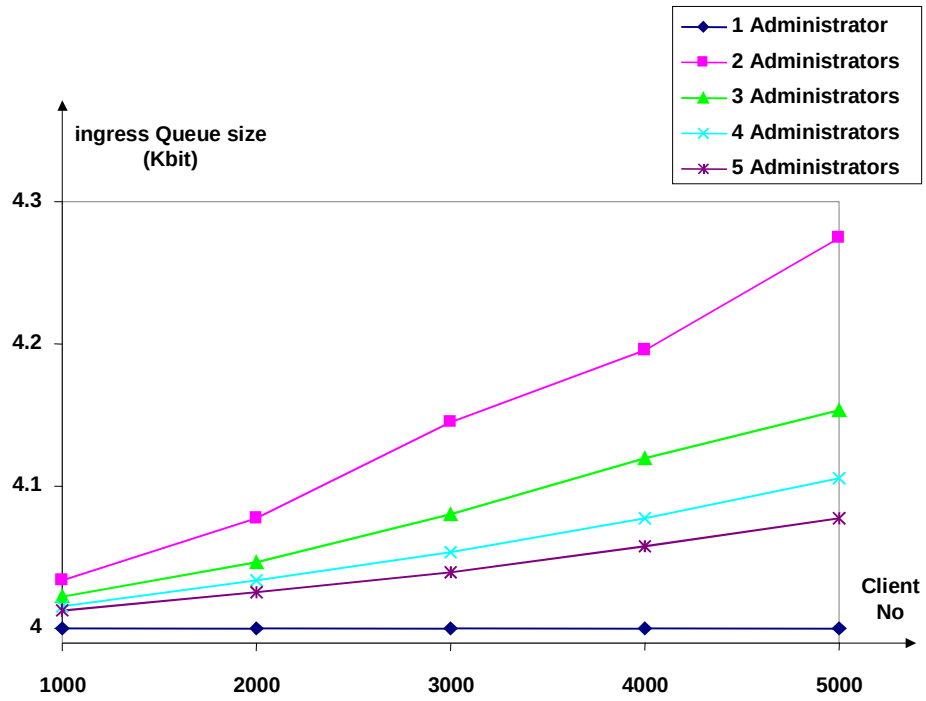


Figure 54: Ingress Queue Size at Administrator vs. Number of Clients

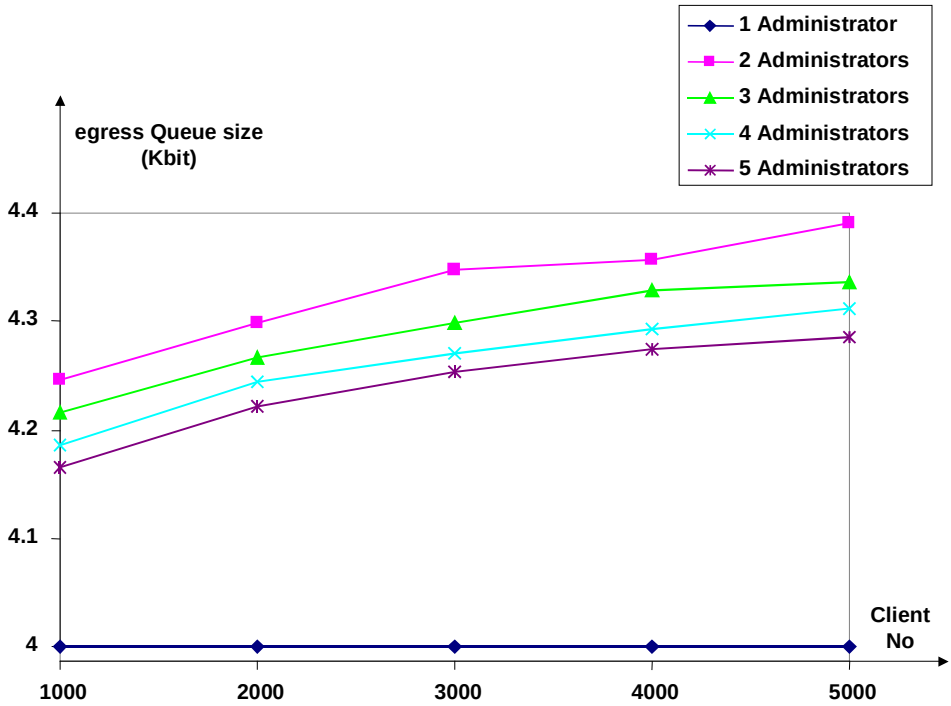


Figure 55: Egress Queue Size at Administrator vs. Number of Clients

The effect of inter-administrator communication can also be observed from the egress queue size, shown in Figure 55. As already explained, the processing delay allows the egress queue to be served more efficiently, and the values in Figure 55 are in general much smaller than those in Figure 47, where the processing delay is very low. Therefore, the size of egress queue in this

scenario is not a true representation of the actual activity and number of messages manipulated at the administrator node. However, this graph can be used to demonstrate the significance of inter-administrator communication on the system performance.

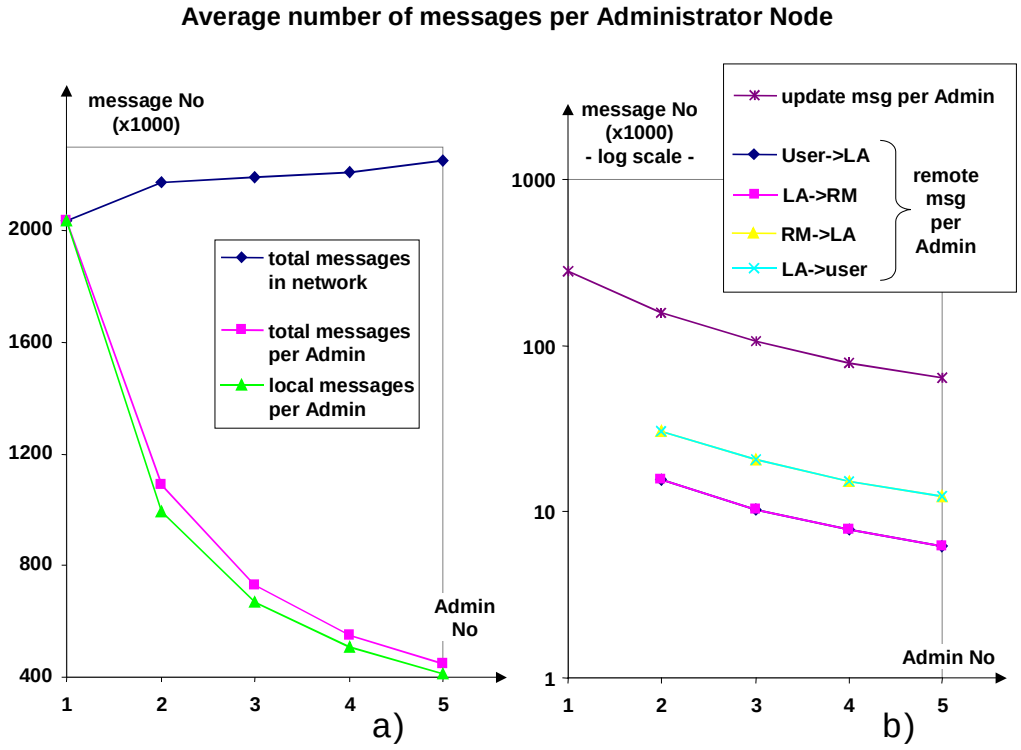


Figure 56: Change of Traffic Structure with Increase of a Number of Administrator Nodes: a) total messages; b) forwarded & update messages

This is shown in Figure 56, as the number of different message types recorded for different scenarios. The message number shown is scaled per-administrator, giving an indication of the average amount of traffic observed by each administrator node in multi-administrator scenarios. Figure 56a shows the overall number of messages per administrator, which reflects the processing delays observed in the nodes. The difference between 'total messages per Admin' and 'local messages per Admin' corresponds to the build-out of the queues. This is shown in more detail in Figure 56b, by observing of a number of different types of inter-administrator communication. Due to the logarithmic scale, the point for a single administrator is not shown, but it crosses Y-axis at the zero point. For a setup with one administrator node, inter-administrator communication is not possible; therefore, the only type of update messages occurring in this case are broadcast messages for local groups. The figure also shows the full number of update messages per administrator node, giving an indication which portion of messages in the model contributes to the queue build-outs in the simulation setup with a high processing delay. Please note in Figure 56b that the other four types of messages (for scenarios with two and more administrators) all refer to the communication related to remote groups.

In addition, Figure 56a gives the total number of messages during the simulation runtime. The growth in the number of messages with increasing number of administrator nodes occurs due to the remote communication, and directly corresponds to the sum of values for 'remote messages per Admin', shown in Figure 56b.

6.2.2.2 Impact of Remote Join / Leave

The results presented so far have demonstrated the essential impact of the remote join/leave operations on the system performance. In order to examine this further, experiments with different probability of the remote operations were performed.

The simulation setup consisted of 3000 client nodes, with the number of administrator nodes changing in different scenarios between 2 and 5, and the periodic update being performed at 1 week intervals. For each of the scenarios, a set of experiments was run for the following values of remote operation probability (%): 0, 1, 5, 10, and 20.

The results in Figure 57 and Figure 58 show approximately linear growth of the administrator's queues, which is influenced both by the increased number of packets exchanged in inter-administrator communication, and the more frequent 'blocking' of ingress queue for processing these packets.

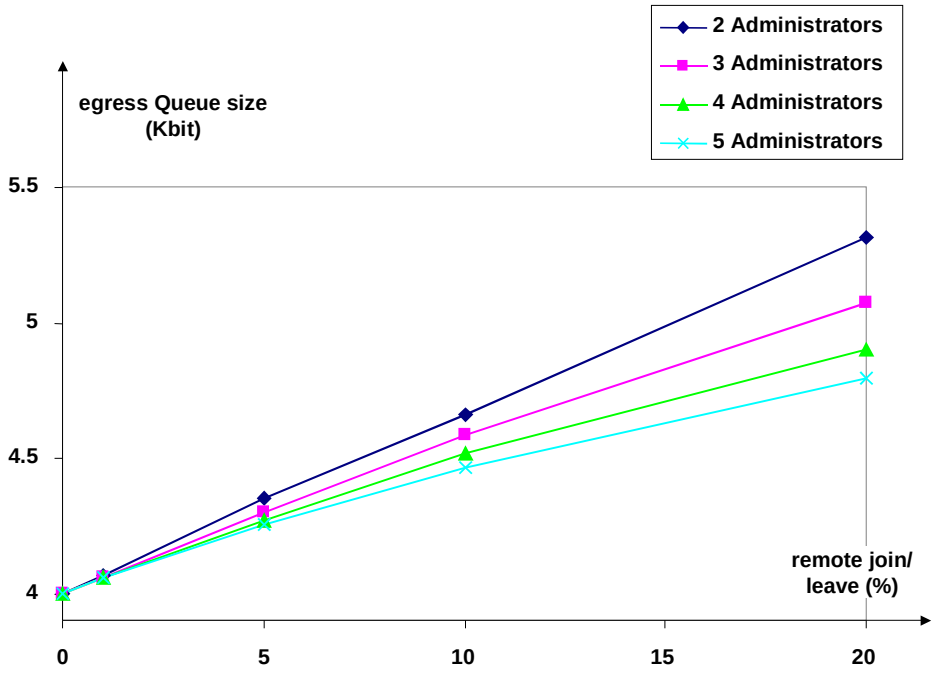


Figure 57: Egress Queue Size at Administrator vs. Remote Operation Probability

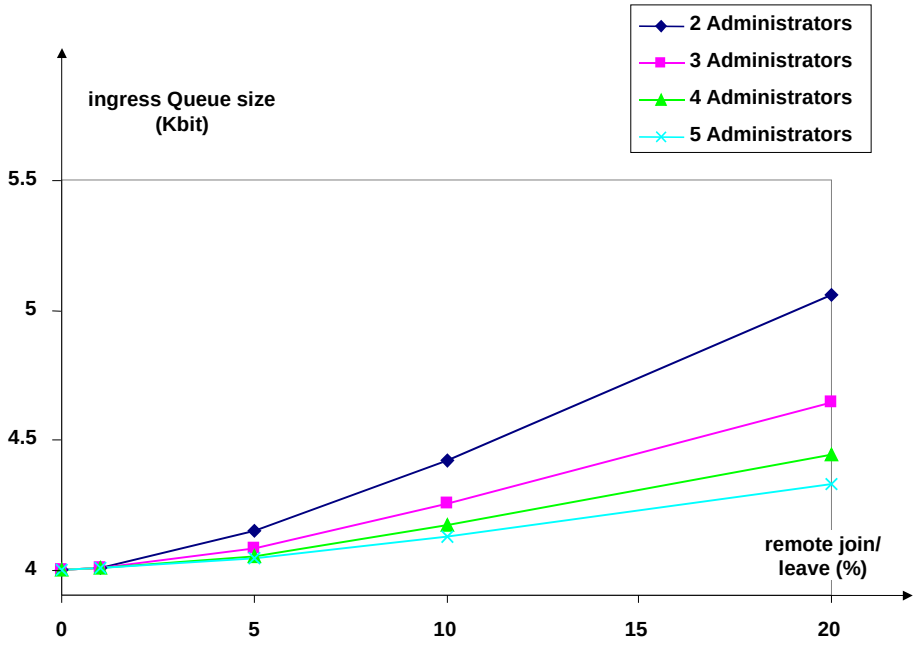


Figure 58: Ingress Queue Size at Administrator vs. Remote Operation Probability

It is also worth noting that for the 0% of remote operation probability, both egress and ingress queues remain unchanged across the range of experiments. This is consistent with the conclusions from the previous section: for 0%, each administrator and its clients act as 'isolated' islands, effectively acting as in the experiments with only one administrator node.

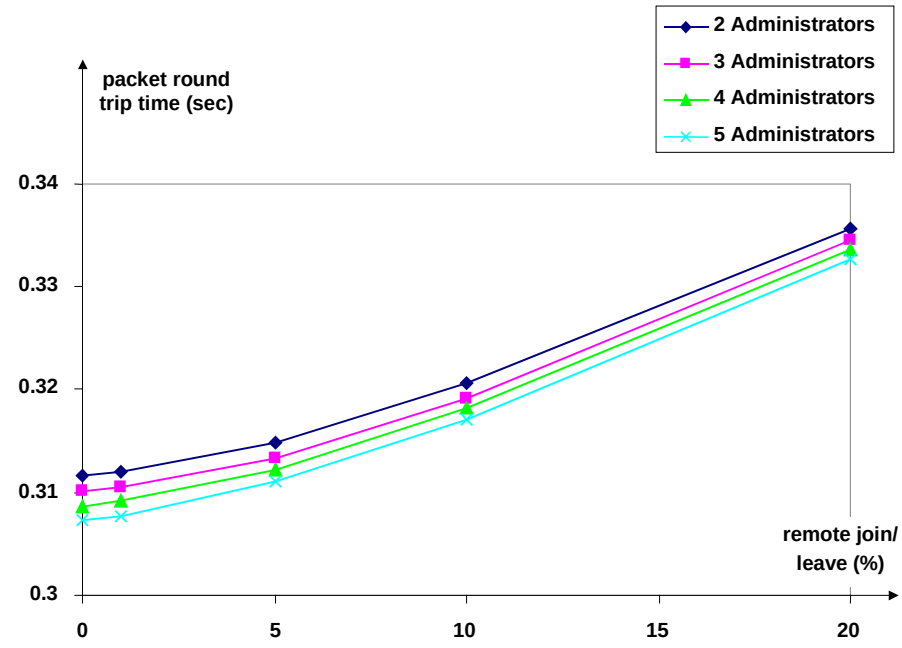


Figure 59: Packet Round Trip Time in function of Remote Operation Probability

The graph in Figure 59 shows the change of the packet round trip time with increasing remote operations. The increase is as expected, since there are more two-hop messages (between the clients and corresponding 'remote group managers'), as the probability of the remote join/leave operation increases. The packet round trip time is higher for scenarios with fewer administrator nodes, which is influenced by the higher processing delay (see Figure 60 and Figure 62).

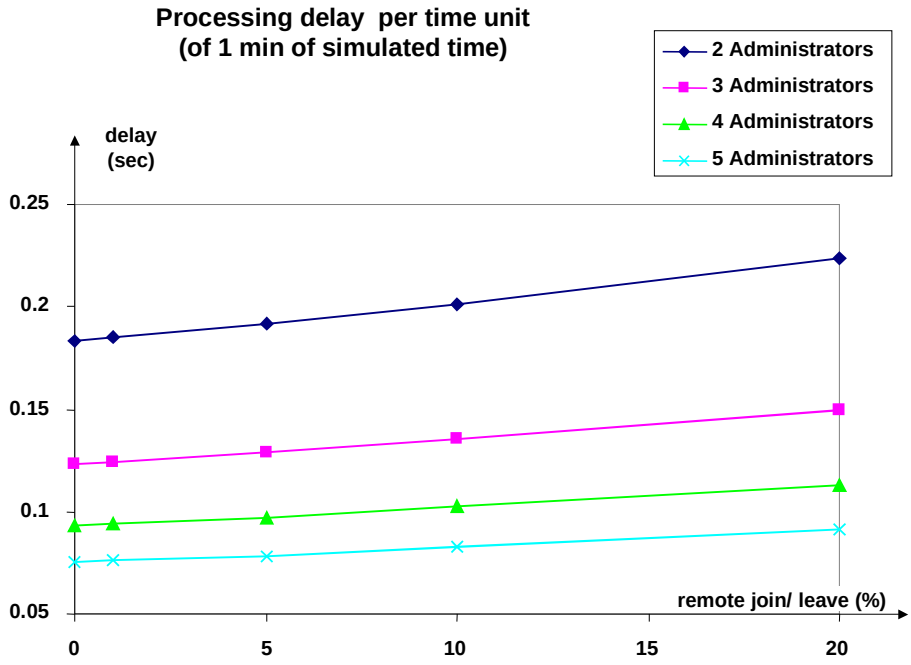


Figure 60: Processing Delay per Time Unit

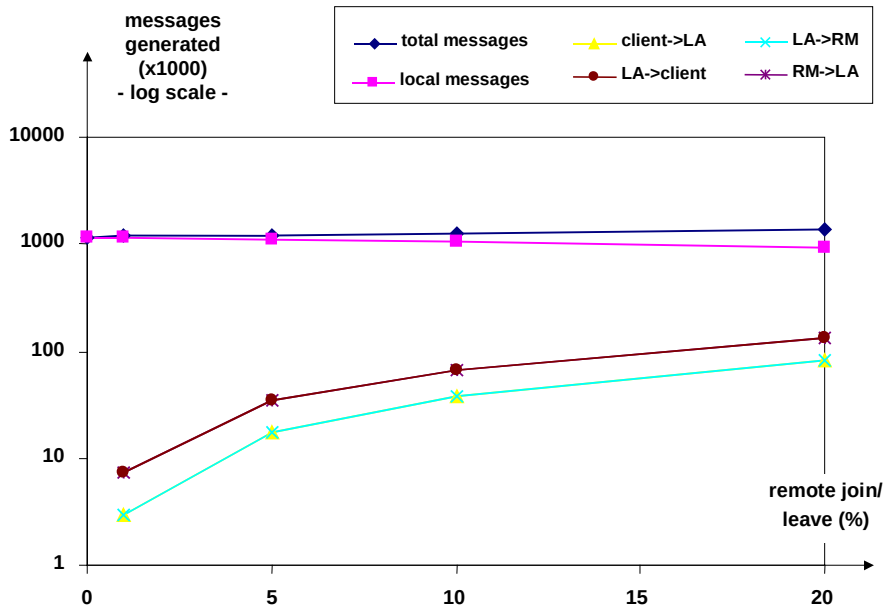


Figure 61: Change of Traffic Structure with Increase of Remote Operation Probability

The processing delay in Figure 60 demonstrates two aspects of the architecture. Firstly, the delay is higher for the scenarios with fewer administrator nodes, which occurs when a larger client population is served. Secondly, the delay increases as the remote operation probability grows, due to an increasing number of forwarded messages (Figure 61 compares two-hop messages with the overall number of messages in the simulation).

Figure 60 shows the growth of the average administrator's node activity. Remote operations effectively create the effect of an increased number of messages in the system, as more and more messages are being involved in two-hop communication, introducing certain processing effort at each node they pass. This growth in the number of messages may be difficult to observe by looking at the 'total messages' in Figure 61, due to the logarithmic scale. However, the increase in peer-to-peer inter-administrator communication (i.e. messages 'LA->RM' and 'RM->LA') directly relates to this. For the scenario where remote operations are set to 0%, there is no inter-administrator communication (i.e. the number of 'total messages' is equal to the number of 'local messages'). Due to the logarithmic scale, points for remote messages for this scenario are not shown in Figure 61, but they cross Y-axis at the zero point.

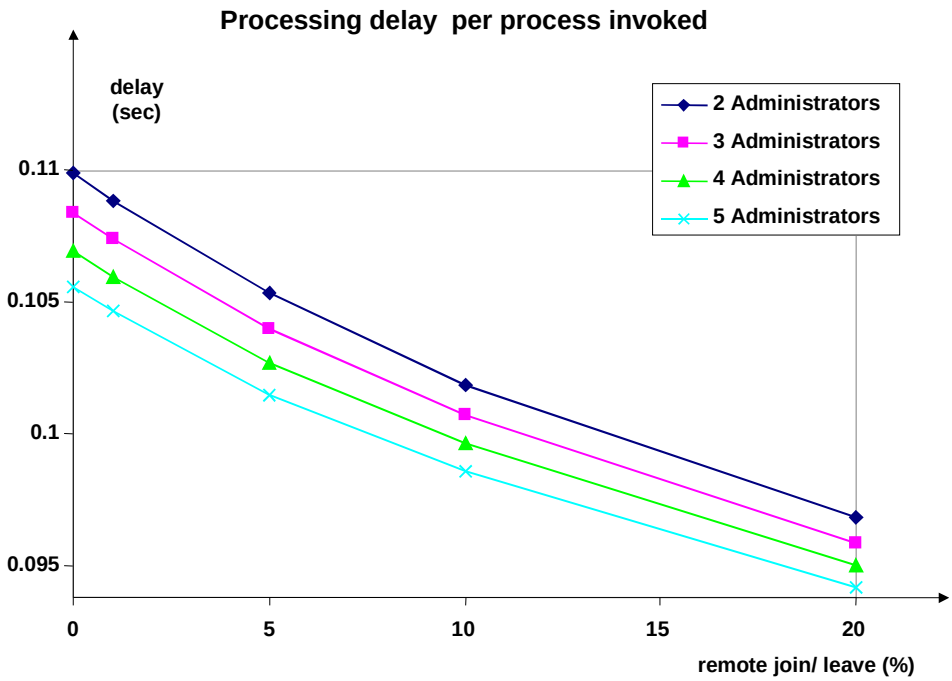


Figure 62: Processing Delay per Executed Event

Another property, directly related to this, can be observed in Figure 62. It shows the average processing delay per each process executed. This was calculated as the total processing delay (the accumulated value over the runtime for each node) was divided with the total number of processes invoked at that node during the runtime.

This delay drops with an increase of remote join/leave operations, since the forwarded messages are less costly. Namely, the client-administrator communication is typically followed by asymmetric encryption and signing (when sent), and the corresponding decryption and signature validation (upon receipt). In contrast, in the administrator-administrator communication the corresponding encrypt/decrypt process employs as symmetric encryption, which is faster to perform. This may appear to be a benefit. However, since more events are being executed, increased remote operations add to the processing overhead as given in Figure 60. (The same can be approximately observed by correlating corresponding values in Figure 61 and Figure 62.)

6.2.2.3 Scalability of Grouping

The results described throughout this chapter are performed with the basic simulation setup, where each administrator is restricted to maintaining no more than 200 simultaneous groups, each of which can grow to a maximum size of 100 members. At the same time, each client is allowed to be member of 10 different groups at most. As already pointed out, the periodic update messages delivered to each group in equal intervals has a significant effect. Both the number of groups, and the size of each of them, directly determine the number of messages that will be sent. In order to explore this aspect of the architecture further, additional experiments with different numbers of groups / group sizes were performed. The details of the experimental setup and analysis of the results is given below.

The scenario is performed using a similar setup to the experiments in the previous section. The setup comprised 3 administrators and 3000 client nodes, with 5% of remote join/leave requests. The processing delay for authentication and encryption were taken from benchmark values, as well as the additional delay for data structure manipulation. Updating group members was performed in equal periodic intervals of 24 hours (of simulated time).

The purpose of the experiment was to examine the impact of the number of groups maintained by the administrator. The approach taken was to consider the situation when each of the participating clients is allowed to be simultaneously member of 10 groups at most. In the scenario with 3 administrator and 3000 client nodes, each administrator would register 1000 clients on average. For each administrator in order to be able to provide membership to 10 groups for each client, it needs to allow a total of 10,000 group-member spaces, and this value was kept constant:

Group_Size * Group_No = 10,000

However, this can be achieved through a range of options, where administrator can maintain a large number of small groups or a few very large groups. The performance of administrator(s)

was tested for several different scenarios, with extremes ranging from 10 groups of 1000 members each, all the way up to 1000 groups - each with 10 members only.

Although the actual number of generated messages remained unchanged through all the experiments, the number of messages sent at the same time (due to a 'group update' mechanism) was expected to create differences in the behaviour of the system. The number of update messages sent over a given time is the same for 10 groups of 1000 members, and for 1000 groups of 10 members. However, update messages for each group are broadcast to all the members at the time determined by the time of the group creation. For a small number of groups this does not happen as frequently as when the number of groups is large, but the number of messages sent 'in one go' is much larger, leading to more bursty traffic from (and between) the administrator node(s).

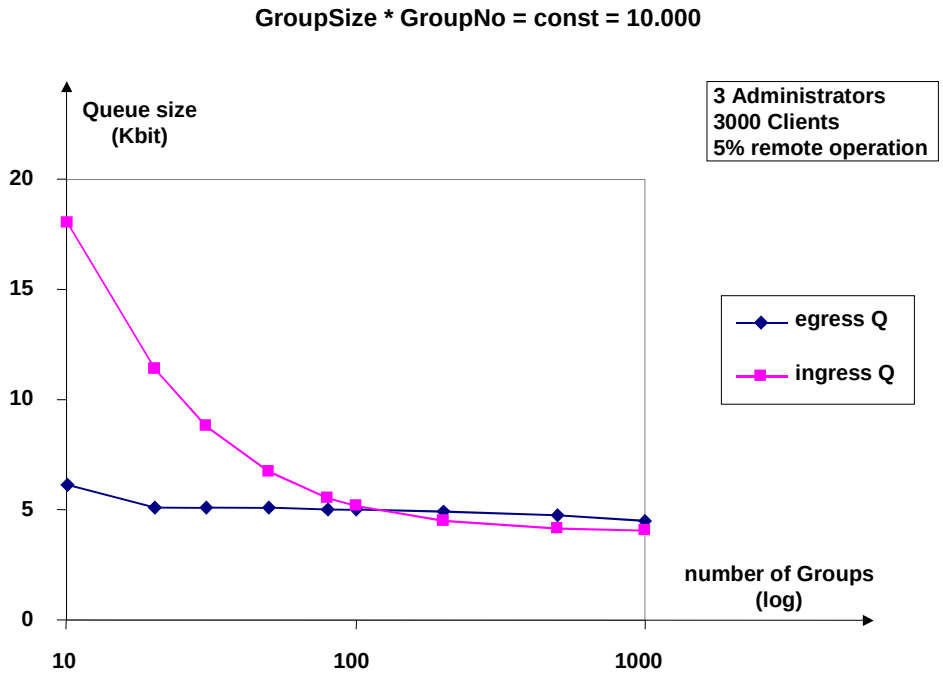


Figure 63: Performance of Administrator Queue for Different Group Sizes

This effect does not cause a burden at the administrator's egress queue (Figure 63) since the delivery of each message is delayed by the estimated time needed for message encryption. However, this mechanism causes longer blockages of the ingress queue (proportional to the number of messages sent) when updates take place. In addition, a large number of messages transmitted at once between administrators decreases the average packet inter-arrival time at the administrator ingress queue (on average, 5% of update messages are sent by the group manager to 'remote' members). All this causes a significant burden at the administrator's ingress queue for larger groups.

Figure 64 gives the values of average processing time (per process invoked). Regardless of the number of groups, the client processing time remains constant since the activity of a client remains unchanged. For the administrator node, processing delay per time unit remains relatively unchanged, due to a constant value of GroupSize*GroupNo. However, for larger groups periodic updates cause more messages to be generated 'at once', whereas for a larger number of small groups this is more spread in time. This effect cannot be observed by looking at the processing time (since it is averaged over the simulation runtime), but is visible from the values for queue delay shown in Figure 65. The slight decrease in the administrator's processing time 'per event' comes from the part of the processing delay introduced by data structure manipulation, which is directly proportional to the size of the data structure accessed. This part is higher in a scenario with a few large groups: in this case, searches need to be performed through a long list of groups at the same time; if the administrator searches for a particular client, the list of the group members for each group is smaller. These two effects have opposite effects. However, the list of groups is searched more often for a particular group (join, leave, update), than a list of group members (only required if a client is leaving or deregistering). In fact, searches through the list of groups need to be performed when looking for a particular group member. However, as already noted, the data structure manipulation time is much smaller than the processing time corresponding to the encryption/authentication mechanisms and therefore does not have a significant impact.

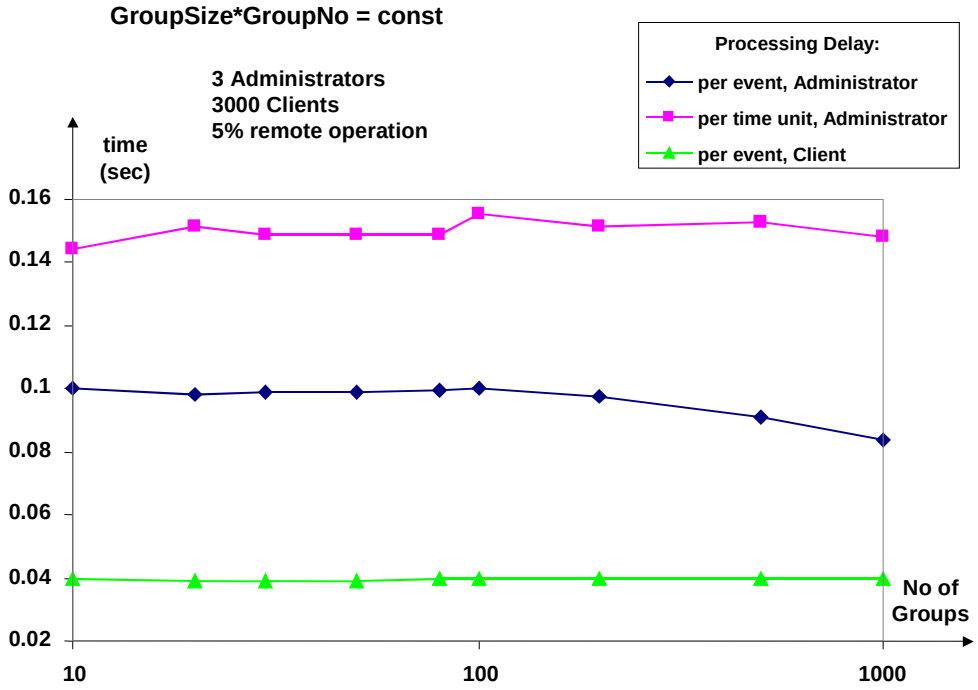


Figure 64: Processing Delays for Different Group Sizes

Finally, the number and size of groups influence the packet round trip times in a straightforward manner: transmission times between nodes are kept the same in all the experiments, and changes in packet round trip time are influenced by (and roughly correspond to) the queue delay observed at the administrator nodes (Figure 65). Again, (consistent with the queue size observed - see Figure 63) this is much more significant at the administrator's ingress queue. (The characteristic for egress queue delay follows the general shape as the ingress' one, but with much smaller values that are difficult to observe on the graph in Figure 65.)

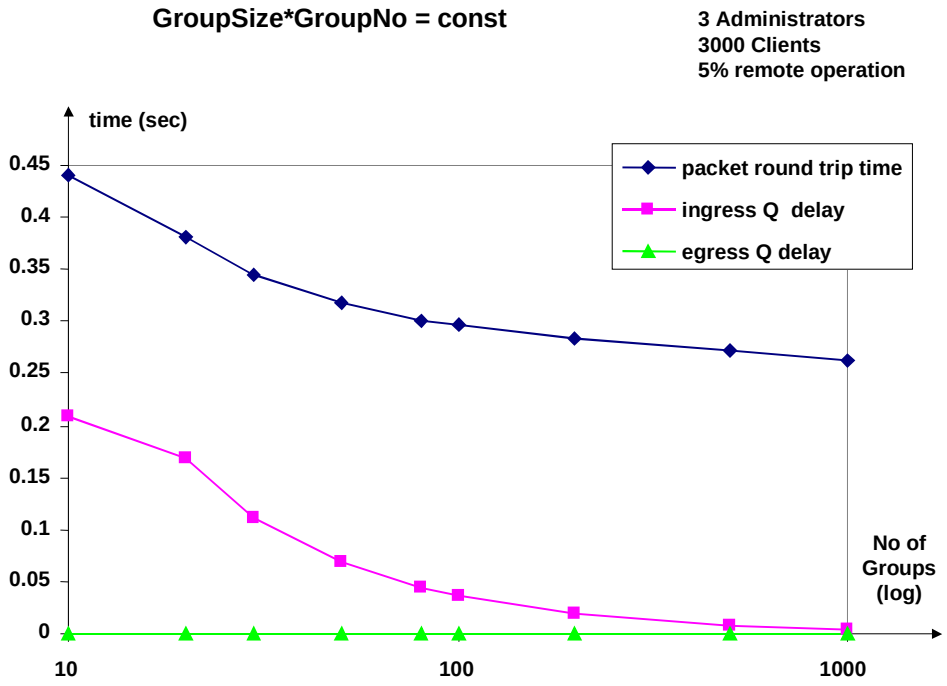


Figure 65: Delays in the System for Different Group Sizes

6.2.2.4 Frequency of Periodic Updates

This section examines periodic update mechanism in more detail. In general, the frequency of periodic updates is a direct trade-off between security and scalability of the system [32]. In order to explore this, experiments were performed with a typical setup of 3000 client nodes, 3 administrators and 5% of remote join/leave operation. Different scenarios were performed for the following values of periodic updates (in hours of simulated time): 5, 10, 24, 50, 100, and 168 (1 week) [62].

The values in Figure 66 show a significant increase in the queue sizes for more frequent periodic updates. Particularly significant is the exponential growth of the ingress queue. Both the broadcast messages created at the administrator node, and those forwarded to the remote group members influence the queue size. For the egress queue, the effect is due to the actual

[62] As suggested in [32], typical value for the revocation period can be anything between 1 day and 1 month.

number of messages served. For the ingress queue, the effect is both due to increasing number of messages, as well as more frequent 'blocking' of the queue due to the processing delay.

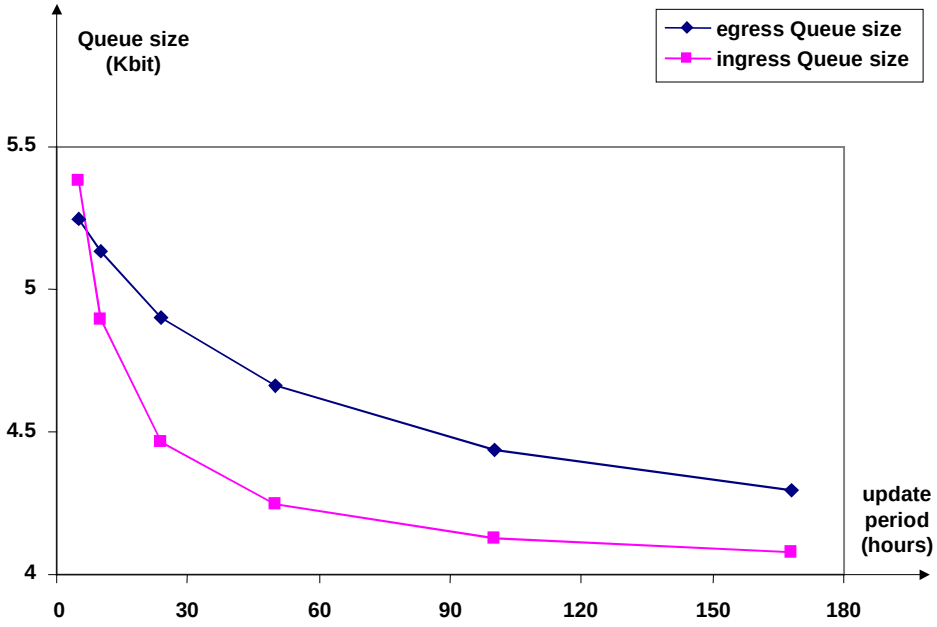


Figure 66: Administrator Queue Size for Different Periodic Update Values

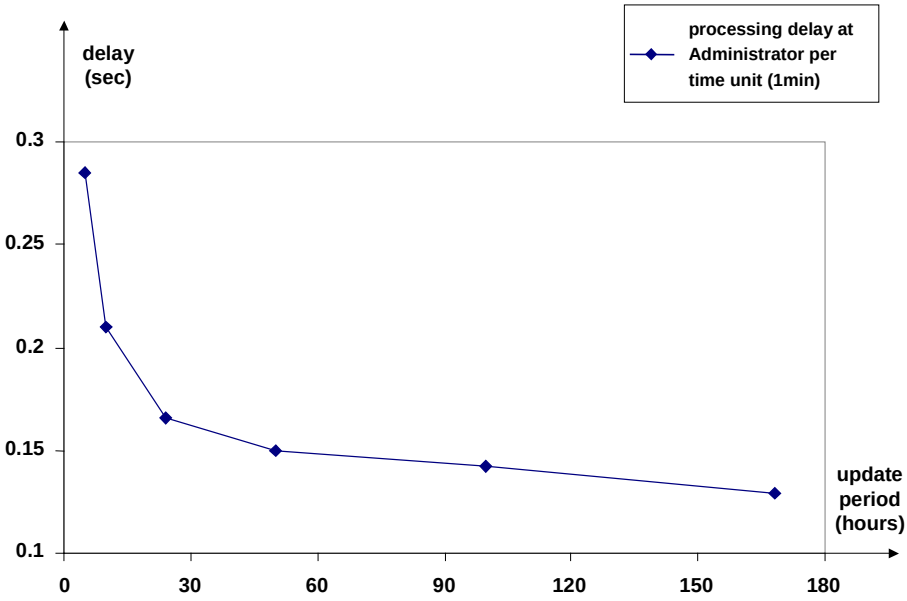


Figure 67: Processing Delay for Different Values of Periodic Updates

Figure 67 shows the increase in processing delay, which is significant for more frequent message updates, as expected. Also, a comparison of the processing delay with the ingress

queue size in Figure 66 demonstrates that they are closely related. Namely, for more frequent updates more messages are generated and need to be processed, resulting in a higher processing times being observed. In addition, due to a mechanism of queue blocking whenever 'processing' of a message takes place, the ingress queue is blocked for longer (on average). At the same time, for more frequent updates, more messages are received at the ingress queue due to inter-administrator communication (this is supported by the characteristics of egress queue size in Figure 66; i.e. more messages are actually being sent for smaller update periods). Both of these cause faster build-out of the ingress queue when more frequent update periods are employed.

6.2.2.5 Robustness

Further experiments were performed to test the robustness of the architecture in the presence of packet loss in the system.

In general, packet loss can occur either since packets are being discarded at ingress/egress queues (due to the overloading of the system), or due to unreliability of underlying transport infrastructure and/or link failure. The first one is an embedded property of the system, directly influenced by the protocol design. It reflects the scalability of the system, and is something that cannot be removed by upgrading the underlying infrastructure. This aspect of the evaluation has been covered in the previous sections, through a range of scenarios with different operational parameters. The ARQ protocol was in place in order to prevent any deadlocks and deliver retransmissions on as-needed basis. However, benchmark values for packet generation vs. queue sizes and system throughput were set to fully support traffic activity in the system, and no packet loss or retransmissions due to time-outs were recorded.

The second reason for packet loss, i.e. unreliability of underlying transport infrastructure and/or link failure, is not influenced by the architecture design. It depends on the properties and quality of services provided by the underlying transport mechanism [146]. This aspect of a distributed communication system has not been modelled in detail. Therefore, in order to simulate unreliable packet delivery, loss was artificially introduced. This has been done by arbitrarily choosing a packet to be destroyed instead of delivered, at the outgoing stream of the node (i.e. egress queue). At the initialisation of the simulation run, the *percentage packet loss* parameter was set to a predefined value and kept constant during the simulation run. In addition, at every invocation of the packet delivery procedure, a random [0,100) value was calculated. By comparing the percentage packet loss with a random value obtained, a decision on whether the packet is destroyed or delivered is made. A series of experiments were performed for the following values of the percentage packet loss parameter: 0.00; 0.15; 0.20; 0.25; 0.5; 0.8; 1.0; 2.0; 5.0; 10.0. The choice of the values has been motivated by a recent measurements of Internet Service Provider backbone traffic [157], which reports an average packet loss in the range of 0.0

- 0.14%, obtained at different ISP backbones. For this reason, the experiments were performed more densely for the small values. However, the architecture was tested for several of large values in order to capture the behaviour of the system in the extreme conditions. In order to make the extreme conditions more pronounced, client nodes were configured to self-initiate registration only at the initial stage of the simulation runtime, and subsequently to act upon the scheduled events. Therefore, if any request/response message is 'lost', a client would either: re-initiate the request (through ARQ mechanism), or remain idle (if the ARQ mechanism is disabled).

The experiments, as described above, were performed both with and without the ARQ mechanism. The motivation behind this was two-folded:

- To observe changes in the availability of the nodes.
- To measure communication overhead introduced with the ARQ mechanism.

A typical scenario of 3000 client nodes and 3 administrator nodes, with 5% of remote operation was chosen. Normally, clients would register with an administrator by sending a register request message. If packet loss is present in the system, some of these messages will be destroyed instead of being delivered to the administrator. Furthermore, the same can happen with the administrator's response message, needed to acknowledge the client's registration. If either of these messages is 'lost', the client initiates a repeated request after time-out period. This mechanism increases communication activity, but greatly improves the chance that the request/response messages will eventually be delivered. In the absence of ARQ mechanism, client only waits for the response, and remains in a 'sleep mode' until the simulation terminates. The simulation results are presented in Figure 68 and Figure 69.

Figure 68 shows the average number of registered clients during steady state conditions. Results with and without ARQ mechanism are compared. As the packet loss increases, the ARQ mechanism provides retransmission, and supports very well the robustness of the system, even for very high values of packet loss. (The actual number of retransmitted messages and the performance of ARQ is discussed in Section 6.1.2.1 ARQ Protocol.) Conversely, if the communication reliability is not supported with an ARQ mechanism, the system performance degrades severely, and nearly half the nodes are removed from the network.

Examining Figure 69, looking at the cost of implemented ARQ mechanism, gives results as expected. Without ARQ support, the queue size at the administrator node falls with increasing packet loss, due to a decreased overall number of packets in the system. With ARQ in place, the queue size linearly increases, proportional with the packet loss in the system. This happens for two reasons: the overall number of packet grows due to ARQ retransmissions. Also, the

frequency of retransmissions is much higher than the average frequency of client requests, which causes the average packet inter-arrival time to drop, adding to the increase in queue size.

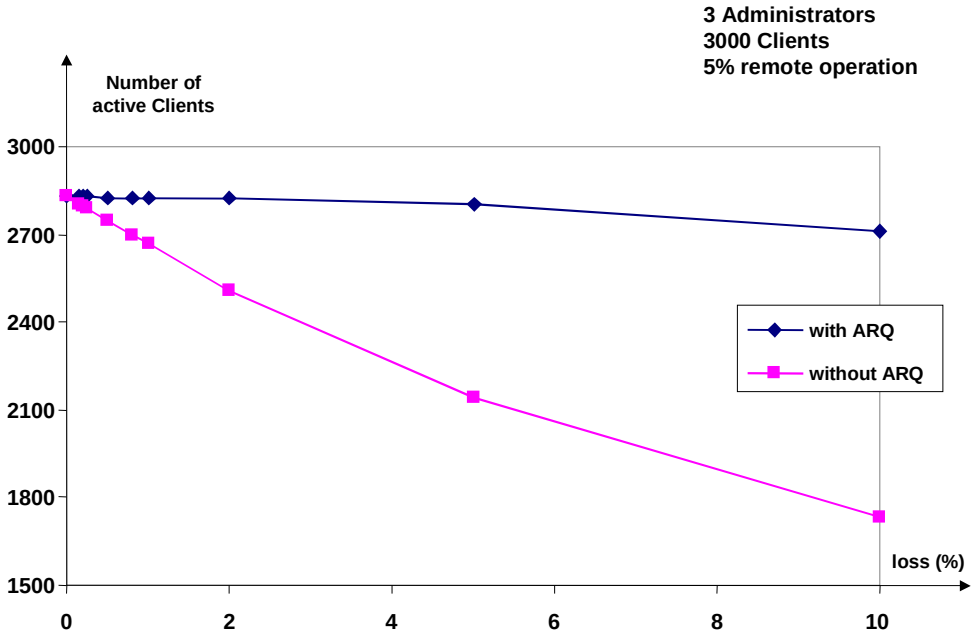


Figure 68: Architecture Robustness Achieved through ARQ Protocol

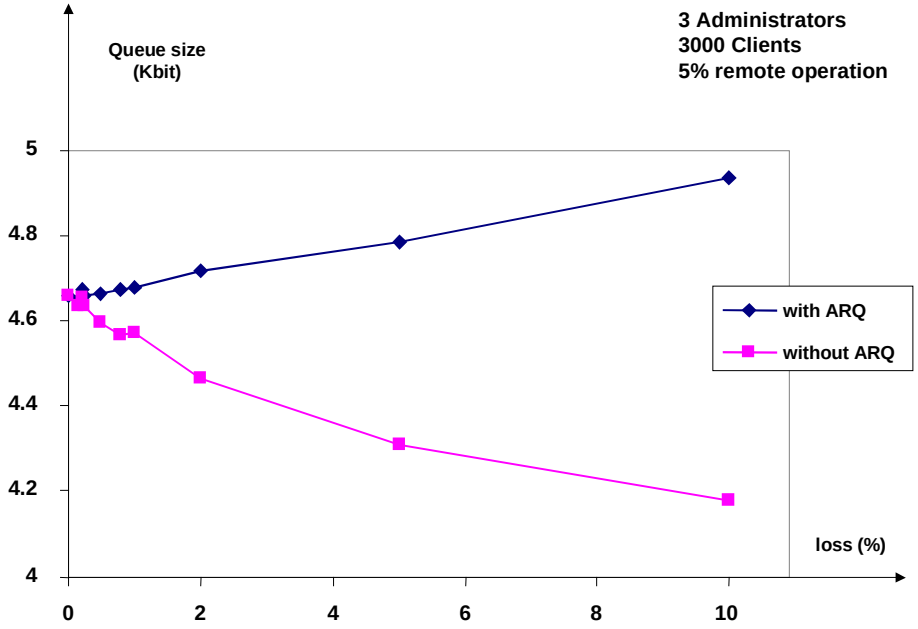


Figure 69: Performance Overhead due to ARQ Mechanism

A more detailed observation for relative low packet loss (0 – 0.5 %) further supports the need for ARQ mechanism. It shows that the availability of the network is maintained nearly as without loss, with negligible communication overhead due to the ARQ mechanism. Comparing it with the number of available nodes in the same range of packet loss but without ARQ (in Figure 68), the benefit is apparent. As already stated, this probability of packet loss is in line with (and even exceeds) the values of loss normally exhibited in the Internet [157].

6.3 Analysis of Simulation Results

This chapter has examined the performance of the simulation model (described in Chapter 5) of the architecture proposed in Chapter 4. Part of the validation process was used to determine suitable values for some of the simulation parameters (i.e. initial registration period) in addition to those identified from the reported network measurements.

The most important conclusion drawn is that the introduction of the processing delays for security procedures strongly impacts on the performance of the system. Another important observation is that inter-administrator communication introduces a significant performance overhead; however, under steady-state conditions, the introduction of more administrators contributes to the scalability of the scheme.

A detailed discussion of the results is presented in Chapter 7.

Chapter 7 Conclusion and Further Works

7.1 Discussion

There is increasing interest in using the Internet for business-related interactions. New technologies and applications are facilitating inter-organisational communication, forming virtual organisational units and dynamic communities that are overcoming geographical boundaries traditionally imposed by a hierarchical structure of a local network. However, lack of common standards and flexible security solutions for such a dynamic and pervasive e-business environment is still one of the main obstacles for wider and more confident business-to-business use of the Internet infrastructure. For example, some of the requirements and challenges for next generation collaborative working environments are identified in [4] as: middleware integration of P2P design with centrally-managed services, management support for inter-organisational processes, support for authentication and authorisation infrastructures within the middleware, self-organising mechanisms to support security and management of collaborative groups.

The main objective of this research was to develop a framework that can provide a new distributed, secure working environment, allowing for dynamic collaboration groups without topological constraints. In achieving this, the author has developed a hybrid architecture, which provides a new approach for flexible inter-organisational collaborative environments, through hierarchical security management and peer-to-peer security enforcement.

The confidentiality of the communication is achieved through combined use of symmetric and asymmetric encryption, and the authentication is achieved through public-key certificates within the scope of local Public Key Infrastructure (PKI), managed by an administrator node. The authorisation rights for different clients are defined at the administration level, both within the scope of the organisation and within the scope of a group, and distributed to the corresponding clients by means of attribute certificates, using local Privilege Management Infrastructure (PMI) [63]. As discussed in Sections 2.2 and 2.3 of Chapter 2, encryption functions and certificate infrastructures provide a means for establishing and maintaining secure and authenticated communication, with the hierarchical certificate infrastructures being more suitable for the deployment of a consistent policy model preferred by corporate environments. The aspects of policy definition and policy negotiation between different administration nodes (including, for example, electronic contracts that may need to be agreed between different parties participating in a group) is outside the scope of this thesis.

[63] The structure of PMI cloesly resembless that of PKI, with the distniction that the latter one manages public-key certificates and the first one attribute certificates which do not contain a public key.

The enforcement mechanism, contained at the end-entity participating in a group, maintains only relevant (but complete) policy for a particular client, based on its privileges in an organisation and in the participating groups. Also, a client's attribute certificates contain the credentials of a client corresponding to its role within various groups. The enforcement takes place for incoming messages (by comparing the certificate credentials with the request), and for outgoing messages (by comparing client's policy with respect to the intended action). This research does not cover the actual means for implementing enforcement mechanisms. The distributed firewall mechanism, initially proposed in [68], which various approaches are described in Section 2.4.1, considers enforcing security policy rules at the access control point, but is less concerned with the mechanisms for the delivery of these rules within a dynamic environment. On the other hand, trust management systems, summarised in Section 2.3.3, provide a framework for the management of authorised access through usage of attribute certificates and access control mechanisms, but normally require that part of privilege certificates and policy rules be independently collected from public directories upon specific requests.

The CUG architecture suggests that the protection of administrator nodes should be ideally supported with separate security features, such as hardware firewalls and a dedicated intrusion detection system. This is the preferred option, since the administrators are high-value assets, and their protection and availability need to be assured by means independent of the mechanisms they manage. However, the architecture does not restrict the setup where administrator nodes are protected by a distributed firewall instance, running on the node and enforcing administrator-level privileges.

In general, the operations of the CUG architecture are supported through a set of protocols. Delivery of relevant policy and management of groups is done through hierarchical client-server interactions, as well as through peer-to-peer interactions on the administrator level, for supporting inter-organisational groups. These interactions are similar to those in extranet Virtual Private Networks (VPN). VPNs, being refined through the years of commercial use, offer a consistent security model suitable for policy management typical of corporate (and cross-domain) environments, but do not provide scalability and flexibility needed to support increasing numbers of nomadic users. The main advantage of the CUG architecture over VPNs comes from the flexibility introduced by the additional peer-to-peer protocol at the client level, which supports direct interactions among the group members, both in a unicast and multicast manner. The security of a group is achieved through protected communication and the distributed enforcement at the distributed firewall instance at the each entity participating in a group. Various peer-to-peer architectures have been proposed by a number of researchers, addressing the issues of flexibility and scalability of the communication. In order to support the

security of the system, they are moving to hybrid models (that resemble hierarchical structures), but still do not provide a consistent model for inter-organisational collaborations. In general, these proposals do not include mechanisms for security enforcement (such as, for example, distributed firewall mechanisms), but mainly concentrate on protection of the communication through data encryption and authentication (in some cases through digital certificates).

The CUG architecture proposed in this thesis brings together aspects of peer-to-peer and client-server communication model, with digital certificates for distribution of security policies and distributed firewalls for policy enforcement. The architecture itself offers a unified approach for supporting communication and security within cross-domain dynamic distributed collaborative groups, through: flexible and scalable way of the communication, robustness and dynamics of the group management, and security of the overall environment (both in terms of the users and of actual communication). In this thesis, the emphasis has been on the general framework and evaluation of the signalling protocol, although a multi-layered mechanism for distributed security enforcement has been proposed. The thesis does not cover aspects of security policy definition, meaning the language for specifying the policy or the actual structure of the certificates in order to accommodate a particular policy deployment model.

One of the important considerations is the choice of a group update mechanism, essential for maintaining security and propagating information concerning current group members. From the security perspective, the purpose of the group update mechanism is not only to inform the members on the current scope of the group, but also to enable the alteration of current members' privileges by re-issuing the certificates they posses and modifying the enforcement rules at the distributed firewalls. In this sense, the advantage of a periodic update mechanism is that it gives certain flexibility to the administrator to perform off-line compilation of new rules and certificates, and to deliver these at pre-determined times. This approach may also be beneficial for clients who would know in advance when to expect the update, if the absolute time is agreed at the level of a CUG. This feature would not be straightforward to implement with 'instant administrator update' scenario. Since the timing of updates will not be known in advance, an administrator may be forced to perform group updates and policy updates separately, which would introduce additional overhead. This approach, commonly adopted for multicast and ad-hoc networks, on the other hand provides a higher level of security by minimising the time period when the scope of the group is uncertain. As demonstrated, the 'appointed member update' scenario offers the best scalability, and is one of the simpler enhancements often used in group multicast protocols. However, this means that the administrator must delegate a significant amount of trust to this member which is not the preferred option, particularly in cross-domain groups where it may be difficult to choose a suitable client node. Also, within the

CUG architecture, this functionality may be adapted to cover broadcast of the messages carrying a list of current members, but not the updating of the policy.

Another important consideration from the security viewpoint is the amount of information contained in the attribute certificates. As described in Chapter 4, every client in the CUG architecture makes use of one attribute certificate per group it participates in. However, as pointed out in Section 2.3.3, there are access control mechanisms that support attribute based on the credentials presented in the certificate, where the full role of an entity (within a group) may be described by a number of certificates. Although the CUG architecture supports this (see the description of the structure of the certificates in Section 4.3.2), this issue has not been considered further. However, one of the obvious choices would be to have a basic attribute certificate that binds the client's identity to a group, and (at least) another certificate that carries set of credentials. In this way, the basic certificate can be presented at the initial session establishment to increase confidence at the authentication process. Since it does not carry a lot of information about the group policy (i.e. sometimes, group names can be public as well), the basic certificate is not security-critical. Once the authentication is performed and the membership to a group confirmed, all the subsequent attribute certificates can be transferred encrypted in order to obtain access rights.

The simulation model used to evaluate the proposed Closed User Groups architecture has been described in Chapter 5. It has been primarily developed to evaluate the performance of security protocols for group and certificate management, whereas the distributed enforcement mechanism has been left to the design-level description as proposed in Chapter 4. All of the protocol functionalities described in Section 4.4 have been implemented and validated, and the performance results were analysed in Section 6.2. The remaining part of this section further discusses the results obtained.

The most significant conclusion drawn is that the introduction of the processing delays for security procedures strongly impacts on the performance of the system. In general, when compared to the scenario without high processing delays, the observation is that the egress queue size drops whereas ingress queue size increases. This happens due to the built-in mechanism for blocking the ingress queue, which prevents further messages from being serviced for the estimated time needed to process request/response message[64]. Since the

[64] Taken out of the context of this research, this effect could be applied to regulate the traffic through the node, by automatically shifting the burden between ingress and egress queue. For example, if thresholds for the ingress and egress queue sizes are defined, ingress queue can be blocked when the threshold of the egress queue is exceeded, and unblocked when its own threshold is reached. Thresholds could be defined statically, dynamically, by employing hysteresis, or blockage can be activated only for certain message/packet types. Since all building blocks involved in this process reside within a single node it is possible to establish this type of control, as demonstrated through the simulation results. Obviously, such an approach is not practical if the build-out of the ingress queue is generally higher, but it may be beneficial in the systems such as the model described in this thesis, where one incoming message initiates the broadcasting of a number of messages.

mechanism developed allows ingress queues to be directly controlled by the estimated processing effort, these results are also valuable in evaluating the processing cost at the administrator under the range of conditions, in addition to the assessment of the message passing – related to scalability of the system. Effectively, this information (together with the measured values for the processing time) gives an insight into the cost of actions that need to be performed on the CPU in order to perform security functions.

On the other hand, the experiments performed with low processing delay nearly completely remove the interdependencies between the number of messages served and their content. As such, the results obtained from these experiments may be valuable in estimating the performance of the system if the administrator nodes were developed with separate dedicated resources both for maintaining the queues, and processing the information.

Another important observation is that inter-administrator communication introduces a significant performance overhead. This is caused by two phenomena:

- The administrator, acting as an intermediate node in the path between clients and corresponding remote group manager, forwards every request and subsequent response message. This effectively increases a number of messages going through the intermediate node, compared to the situation where inter-administrator communication is avoided.
- All of the forwarded messages, but in particular the periodic updates to remote group members, contribute to decreased inter-arrival times at the ingress queue of the intermediate node. The round trip of forwarded client requests and subsequently received responses, as well as forwarding of remote updates messages, have much higher frequency than the average request messages received by the administrator in a single-administrator scenario.

Therefore, the expected benefit in the performance when more administrator nodes are introduced is only partial, and can be observed only in the scenarios without significant processing delay. This demonstrates that the burden at the administrator node decreases when more management nodes are introduced. This observation may be valuable under circumstances when inter-administrator communication is rare, and if more management nodes are introduced only to accommodate larger client population. In these conditions, as demonstrated, administrators and their clients behave as ‘isolated’ parts of the simulation model, exhibiting the same properties as a single administrator would do in a correspondingly smaller environment. However, this situation is of no particular interest in this research, since the inter-administrator protocol is one of the fundamental concepts of the proposed CUG architecture, developed to facilitate management of cross-domain groups.

Also, the general benefit of the multi-administrator approach can be evaluated by comparing the processing delays recorded. These values are consistently higher for scenarios with fewer administrators in the experiments performed both with low and high processing delays.

In order to simulate the inter-organisational environment, more detailed experiments were performed with multi-administrator scenarios, stimulating inter-administrator communication under the range of conditions. The general observation is that in multi-administrator environment, the model scales better with more administrator nodes due to the decomposition of the client population. Experiments with non-equal client populations per administrator have not been performed (which may be a common occurrence in the real system), but the performance of administrator nodes in such a setup can be estimated through interpolation. This is reasonable to assume: the main performance overhead comes from message forwarding and inter-administrator communication. As the administrator can act only on the behalf of its local clients, those serving smaller client population will perform better.

There is a major benefit from maintaining a large number of small groups, compared to a few large groups. Scalability of the model improves due to a fact that the update messages are more distributed in time, as less messages are broadcast (to the small group) at each periodic update. Therefore, it is better for an administrator, under the same conditions, to manage a lot of small separate communities, rather than a large one (or a few). This is very valuable result, consistent with one of the main hypothesis of this research: grouping of clients into the communities of common interest facilitates their management, since the security rules need to be distributed only to the part of population that is (potentially) engaged in the communication determined by their membership.

The main performance overhead of the peer-to-peer interactions at the administrator level are due to decreased inter-arrival times of the incoming messages. The essential cause of this is due to the mechanism used for the group update, and by the clients' ability to generate requests for remote groups. The amount of remote requests generated directly determines the number of responses, as well as the number of update messages sent to remote group members. In the simulation model used in this work, the frequency of remote requests (and clients' actions in general) is modelled as averaged values of actions in time, the same for all the clients. These values are extracted from the various traffic measurements reported in the literature. Results in all of the sources consulted give an indication on how 'majority' of the population behaves, but also demonstrate that there may be large spread in the activity of individual clients. This is in particular true for various peer-to-peer architectures observed, where the amount of traffic generated by different nodes can have significant variations. This has not been captured through the simulation model, since all the client nodes in the model behave the same (with a certain

degree of randomness). However, this is unlikely to affect the administrators' performance, but only performance of various clients in the model, and it was not the focus of the investigation.

Several mechanisms for group update broadcasting have been proposed, and compared for performance and security. Experiments have demonstrated the scalability of the 'appointed member update' mechanism. However, the results obtained are performed for a set-up with signalling communication only. Normally, the 'appointed member' would be also involved in the data communication within the CUG, in which case data traffic will impact the performance of client node, and potentially compromise reliability of the signalling. However, the signalling messages do not occur very frequently, and would indeed contribute to the overall group traffic much less than the data traffic would be expected to. By implementing a priority queueing mechanism for the signalling messages, higher reliability and robustness of the scheme can be achieved even for the applications of high volume traffic. In such a case, delivery of signalling messages is independent on the amount of data traffic exchanged. Although prioritising signalling messages could cause the additional loss of data, this is less critical; also a number of mechanisms (including ARQ) exist in current communication networks to offer support for message reliability and delivery, across different layers of protocol stack.

The 'instant' and 'periodic' administrator update are two other mechanisms for group update broadcasting that have been proposed. These update mechanisms are comparable, both in the performance and security (if the appropriate frequency is chosen), with the latter one giving certain flexibility to the administrator in scheduling the time for updates. In addition, the security implications of each scheme have been discussed above. A detailed examination of the periodic update scheme demonstrates that the update frequency directly impacts the performance of the administrator. As already noted, this is a direct trade-off between the scalability and security of the model: more frequent updates assure more timely delivery of the information about group members and revoked group certificates, but introduce additional overhead at the same time. This may need to be taken into account, depending on the type and scope of the group and security needed for the individual groups. Experiments where the update period is not the same for all groups have not been performed. In addition, since the group update is the only message type where one incoming message initiates a number of outgoing messages and broadcast to a group, it may be possible to implement priority queueing at the administrator, with respect to the type of signalling message received or being sent. This may help in achieving higher robustness of the system. However, the system (of the examined size of the population) has demonstrated successful operation even without this improvement, as the messages loss was observed only when loss was artificially introduced at egress queues.

The robustness of the architecture has been achieved by using an ARQ (Automatic Repeated reQuest) protocol. The experiments conducted even under extreme conditions demonstrate a tolerable level of performance despite the overhead introduced by the ARQ mechanism. More detailed examination, performed for values of packet loss that commonly occur in the Internet, shows that the overhead is insignificant compared to the benefit.

The simulation model developed during this research served to assess the basic functionalities and performance, and to investigate the potential concerns of the new architecture. A number of issues and points of further interest have been outlined and addressed through the discussion above. Various simulation scenarios can be developed to further explore the model.

However, the existing simulation model is not suitable for evaluation of other aspects of the architecture, such as enforcement mechanisms and policy management model. In order to assess these features, and demonstrate the practical value of the CUG architecture (particularly in terms of security), a real implementation may be more suitable. Also, the architecture presents a basic framework for secure and dynamic group working. In order to bring it closer to the real examples of emerging e-business models, further advancements are needed. All of these are discussed towards the end of this chapter.

7.2 Conclusions

The emerging paradigm of Virtual Organisations, aiming to provide dynamic and secure environment for distributed collaborations, is imposing new requirements on current communication and security models. Significant research efforts, developing certificate infrastructures and distributed firewall mechanisms on one side, and Grid and peer-to-peer architectures on the other, are trying to address some of the important issues raised.

The objective of this research was to develop a suitable architecture that can provide secure and flexible means for the management of dynamic collaborative environments. The main outcome of the work is the new hybrid architecture of Closed User Groups. The CUG architecture combines aspects of peer-to-peer and client-server communication models, enabling flexible and scaleable formation and management of cross-domain groups. In addition, a mechanism for distributed enforcement of security policies is proposed, based on distributed firewall paradigm and a hierarchical model of public-key and attribute certificate infrastructures.

A performance evaluation through simulation has demonstrated the prospective benefits of per-group security policy distribution, but has also pointed out some potential issues of inter-administrator communication. Although a more detailed simulation model could examine these

issues further, a prototype implementation would be more beneficial, by including aspects of real policy deployment and actual enforcement mechanisms.

Nevertheless, the satisfactory operation of the CUG architecture has been demonstrated. In embryonic form it offers a framework that facilitates management of dynamic security perimeters, suitable to support interactions across the organisational domains.

7.3 Further Work

There are several directions where the work presented in this thesis could be developed further.

One of the plans is to develop a test-bed implementation of the proposed architecture, and initial steps in this direction have already been made. Recently, the author has been approached by the members of EU Project GRASP (Grid-Based Application Service Provision) [158], which aims to design and implement (using GRID technologies) a layered architecture for one-to-many and many-to-many service provisioning business models. The initiative concerning this research is to adopt the described CUG architecture for providing security mechanisms within the business model architectures proposed in GRASP. The initial evaluation of the applicability of the CUG architecture has already been done, and it has been concluded that the hybrid message-exchange protocol and the framework for certificate-based grouping are generic enough in order to be applied as a security layer within the GRASP models. The architecture would be implemented as a middleware using Microsoft.NET platform. A hybrid protocol would be developed by means of the SOAP (Simple Object Access Protocol) message format, and the certificates and privilege allocation mechanism would be implemented through a combination of local public-key infrastructure and security tokens directly provided by the SOAP specification [159]. The plans for distributed security enforcement are less concrete as they (at this stage) may depend on the platform and the specific business model, although the multi-layered approach (as proposed in this thesis) will be considered.

The CUG architecture proposed in this thesis provides a framework for secure and trusted operations of dynamic groups. In order to enable inter-organisational collaborations in a business context, a further requirement is to support these interactions in a similar way to traditional organisations, to ensure the compliance with their existing business agreements. To this end the CUG architecture can make use of services provided by the specialised systems for management of electronic contracts. These services, for example, facilitate recording of newly created terms of agreement, monitoring their performance, and arbitration and enforcement in the case of contract non-compliance. An approach to this has already been proposed in **[DIM]**, as a joint effort.

Finally, another possible direction for future work is to extend the proposed CUG architecture in order to include mobility of the users. It may be possible to extend the proposed architecture to accommodate the Mobile IP framework [160]. This could be achieved by moving part of the security enforcement functions from the end-entity to the access routers (such an approach has already been proposed in [161], as means for optimising CPU usage on, normally less powerful, mobile nodes). In order to achieve this, the current protocol would need to be extended to include the transfer of the corresponding security rules in order to accommodate node mobility and roaming services.

References

[1] 2002 Australian Computer Crime and Security Survey, www.auscert.org.au/

[2] Computer Security Institute (CSI): CSI/FBI Computer Crime and Security Survey, Reports for 2001-2003, available from: http://www.gocsi.com/

[3] Stallings W.: Cryptography and Network Security. Prentice Hall, 1999. ISBN 0138690170

[4] Expert Group on Collaboration@Work: Next Generation Collaborative Working Environments 2005-2010. Report of European Commission / European Space Agency Expert Group (Kick-Off Meeting), Brussels, Belgium, 4th May 2004

[5] Wang G.W.: Inter-Networking Security. Proc of IEEE International Conference on Communication Systems (ICCS), November 1994, Singapore

[6] Menezes A., Oorschot P.van, Vanstone S.: Handbook of Applied Cryptography. CRC Press, 1996. ISBN: 0-8493-8523-7, www.cacr.math.uwaterloo.ca/hac/

[7] Coras: General Definitions and Terminology, www.nr.no/coras

[8] Graham R.: Hacking Lexicon. http://www.robertgraham.com/pubs/hacking-dict.html

[9] Chirilo J.: Hack Attacks Revealed: A Complete Reference with Custom Security Hacking Toolkit. John Wiley & Sons, 2001, ISBN 0-471-41624-X, pp. 944.

[10] IBM Report: Safe Surfing: How to Build a Secure World Wide Web Connection. IBM Redbooks SG24-4564-00, April 1996, http://www.redbooks.ibm.com

[11] Shirey R.: Internet Security Glossary. RFC 2828, Category: Informational, IETF, May 2000.

[12] Adan M., Elmgren E., Fosdike D., Granum D., Mendoza D., Spangberg L.O., Tang R., Zeier E.: AS/400 Internet Security Scenarios: A Practical Approach. IBM Redbooks SG24-5954-00, July 2000, www.redbooks.ibm.com

[13] Group of Authors: Hack Proofing Your Network: Internet Tradecraft, Syngress Publishing Inc., 2000.

[14] Computer Viruses FAQ: http://www.faqs.org/faqs/computer-virus/

[15] Daemen J., Rijmen V.: AES Proposal: Rijndeal, September 1999.

[16] Diffie W., Hellman M.E.: New directions in cryptography. IEEE Transactions on Information Theory, 22/6, June 1976, pp 644-654.

[17] Rivest R.L., Shamir A., Adleman L.M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21/2, February 1978, pp 120-126.

[18] Neuman C., Yu T., Hartman S., Raeburn K.: The Kerberos Network Authentication Service (V5). Internet Draft, IETF, 15th February 2004 (expires 15th August 2004). http://www.ietf.org/internet-drafts/draft-ietf-krb-wg-kerberos-clarifications-05.txt

[19] Tung B., Neuman C., Hur M., Medvinsky A., Medvinsky S., Wray J., Trostle J.: Public Key Cryptography for Initial Authentication in Kerberos. Internet Draft, IETF, 20th February 2004 (expires 20th August 2004). http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-pk-init-18.txt

[20] ITU Recommendation X.509, The Directory: Authentication Framework (also ISO/IEC 9594-8, 1995). International Telecommunications Union, June 1997.

[21] Kent S.: Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management. RFC 1422, IETF, February 1993.

[22] Kohnfelder L.M.: Towards a Practical Public-key Cryptosystem. Bachelor Thesis, Massachusetts Institute of Technology, May 1978.

[23] Wiener M.J.: Performance Comparison of Public-Key Cryptosystems. RSA Laboratories' CryptoBytes, Vol. 4, No 1, Summer 1998, pp 1-5; http://www.rsasecurity.com/rsalabs/cryptobytes/

[24] Branchaud M.: A Survey of Public-Key Infrastructures. MSc Thesis. Department of Computer Science, McGill University, Montreal, March 1997.

[25] PGP Corporation, http://www.pgp.com

[26] Atkins D., Stallings W., Zimmermann P.: PGP Message Exchange Formats. RFC 1991, Category: Informational, IETF, August 1996.

[27] PGP Corporation: Introduction to Cryptography. May 2003; www.pgp.com/products/whitepapers

[28] Gerck E.: Overview of Certification Systems: X.509, PKIX, CA, PGP & SKIP. Proc of Black Hat Conference, Las Vegas, USA, July 1999. http://www.blackhat.com/html/bh-usa-99/bh3-index.html

[29] IETF's Public Key Infrastructure X.509 (PKIX) Charter, http://www.ietf.org/html.charters/pkix-charter.html

[30] Adams C., Farrel S.: Internet X.509 Public Key Infrastructure: Certificate Management Protocol. RFC 2510, Category: Standards Track, IETF, March 1999.

[31] Chokhani S., Ford W., Sabett R., Merrill C., Wu S.: Internet X.509 Public Key Infrastructure: Certificate Policy and Certification Practises Framework. RFC 3647, Category: Informational, IETF, November 2003.

[32] Housley R., Polk W., Ford W., Solo D.: Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, Category: Standards Track, IETF, April 2002.

[33] Myers M., Ankney R., Malpani A., Galperin S., Adams C.: X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol (OCSP). RFC 2560, Category: Standards Track, IETF, June 1999.

[34] Boneh D., Ding X., Tsudik G., Wong C.M.: A Method for Fast Revocation of Public Key Certificates and Security Capabilities. Proc of 10th Usenix Security Symposium, Washington DC, USA, August 2001; http://www.usenix.org/events/sec01/boneh.html

[35] Boneh D., Franklin M.: Identity Based Encryption from the Weil Pairing. SIAM Journal of Computing, Vol. 32, No. 3, pp. 586-615, 2003; http://epubs.siam.org/sam-bin/dbq/article/39852

[36] Identity-Based Encryption E-mail: http://crypto.stanford.edu/ibe/

[37] Callas J.: Identity Based Encryption, CTO Featured Topic, PGP Corporation; November 2003; http://www.pgp.com/company/ctocorner/index.html

[38] The PKI Page; http://www.pki-page.org/

[39] Verisign Home; http://www.verisign.com/

[40] Verisign Certification Authority Practice Statement; https://digitalid.verisign.com/test_ca_cps.html

[41] Ellison C.: SPKI Requirements. RFC 2692, Category: Experimental, IETF September 1999.

[42] Ellison C., Frantz B., Lampson B., Rivest R., Thomas B., Ylonen T.: SPKI Certificate Theory. RFC 2693, Category: Experimental, IETF September 1999.

[43] Wang Y.: Simple Public Key Infrastructure (SPKI). Seminar on Network Security, Authorization and Access Control in Open Network Environment, Proceedings of the Helsinki University of Technology, December 1998; http://www.hut.fi/~yuwang/publications/SPKI/SPKI.html

[44] Rivest R.L., Lampson B.: SDSI- A Simple Distributed Security Infrastructure. (version 1.0), September 1996; http://theory.lcs.mit.edu/~rivest/sdsi10.html

[45] Clarke D., Elien J.-E., Ellison C., Fredette M., Morcos A., Rivest R.L.: Certificate Chain Discovery in SPKI/SDSI. Journal of Computer Security, Vol. 9, Issue 4, 2001, pp. 285-322

[46] IETF's Simple Public Key Infrastructure (SPKI) Charter; http://www.ietf.org/html.charters/spki-charter.html

[47] Park J.S., Sandhu R.: Smart Certificates: Extending X.509 for Secure Attribute Services on the Web. Proc of 22nd National Information Systems Security Conference, Crystal City, Virginia, USA, October 1999.

[48] Farrel S., Housley R.: An Internet Attribute Certificate Profile for Authorization. RFC 3281, Category: Standard Tracks, IETF, April 2002.

[49] Sandhu R.S., Samarati P.: Access Control: Principles and Practice. IEEE Communications Magazine, Vol. 32, No 9, September 1994.

[50] Sandhu R.S.: Future Directions in Role-Based Access Control Models (Keynote Lecture). Proc of International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS 2001), May 2001, St. Petersburg, Russia

[51] Dulay N., Lupu E., Sloman M., Damianou N.: A Policy Deployment Model for the Ponder Language. Proc of 7th IEEE/IFIP International Symposium on Integrated Network Management, Seattle, USA, May 2001.

[52] Linn J.: Attribute Certificates: An Enabling Technology for Delegation and Role-Based Controls in Distributed Environments. Proc of 4th ACM Workshop on Role-Based Access Control, Fairfax, Virginia, USA, October 1999.

[53] Herzberg A., Mass Y., Michaeli J., Ravid Y., Naor D.: Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. IEEE Symposium on Security and Privacy (S&P 2000), Berkeley, California, USA, May 2000.

[54] Al-Kahtani M.A., Sandhu R.: A Model for Attribute-Based User-Role Assignment. 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA, December 2002.

[55] Blaze M., Feigenbaum J., Ioannidis J., Keromytis A.: The KeyNote Trust-Management System Version 2. RFC 2704, Category: Informational, IETF, September 1999.

[56] Chadwick D.W., Otenko A., Ball E.: Role-Based Access Control with X.509 Attribute Certificates. IEEE Internet Computing, Vol. 7, Issue 2, March/April 2003, pp. 62-69

[57] Thompson M.R., Essiari A., Mudumbai S.: Certificate-Based Authorization Policy in a PKI Environment. ACM Transactions on Information and System Security (TISSC), Vol. 6, Issue 4, November 2003, pp. 566-588

[58] Chadwick D.W., Otenko A.: A Comparison of the Akenti and PERMIS Authorisation Infrastructures (invited paper). Proc of ITI/IEEE 1st International Conference on Information and Communications Technology (ICICT 2003), Cairo, Egypt, December 2003, pp. 5-26

[59] Cheswick W.R., Bellovin S.M.: Firewalls and Internet Security: Repelling the Wily Hacker, 1st Edition. Addison Wesley Professional, April 30th, 1994, ISBN 0201633574, pp. 320; also available at http://www.wilyhacker.com/1e/

[60] Snyder J.: Pushing firewall performance. Network World Fusion Newsletter, 12th March 2001; http://www.nwfusion.com/reviews/2001/0312rev.html

[61] Pfleeger C.P., Pfleeger S.L.: Security in Computing, 3rd Edition, Prentice Hall PTR, February 2003, ISBN 0130355488

[62] Blakley B.: Three Myths of Firewalls. http://web.mit.edu/kerberos/www/firewalls.html

[63] Computer Security Institute (CSI): 2003 CSI/FBI Computer Crime and Security Survey, 8th Annual. Computer Security Issues & Trends, Spring 2003; http://www.gocsi.com/

[64] Das K.: Protocol Anomaly Detection for Network-based Intrusion Detection, SANS (SysAdmin, Audit, Network, Security) Institute InfoSec Reading Room, January 2002; http://www.sans.org/rr/

[65] Snapp S.R., Brentano J., Dias G.V., Heberlein L.T., Ho C., Levitt K.N., Mukherjee B.: DIDS (Distributed Intrusion Detection System)- Motivation, Architecture, and an Early Prototype. Proc of 14th National Computer Security Conference, October 1991, pp. 167-176.

[66] Janakiraman R., Waldvogel M., Zhang Q.: Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention. Proc of 12th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), June 2003. Linz, Austria. pp. 226-231.

[67] Gamez D., Bigham J., Cuthbert L.: An Architecture for Anomaly Detection and Repair in Large Complex Critical Infrastructures. Proc of 10th International Conference on Telecommunication Systems, Modelling and Analysis, Monterey, California, USA, 2002.

[68] Bellovin S.M.: "Distributed Firewalls". ;login:, The Magazine of USENIX Association, November 1999, pp.37-39. http://www.usenix.org/publications/login/1999-11/features/firewalls.html

[69] Ioannidis S., Keromytis A.D., Bellovin S.M., Smith J.M.: Implementing a distributed firewall. Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000), Athens, Greece, November 2000, pp. 190-199

[70] Keromytis A.D., Ioannidis S., Greenwald M.B., Smith J.M.: The STRONGMAN Architecture. Proc of 3rd DARPA Information Survivability Conference and Exposition (DISCEX III), Washington, DC, USA, April 2003, pp. 178-188.

[71] Jin H., Xian F., Han Z., Li S.: A Distributed Dynamic micro-Firewall Architecture with Mobile Agents and KeyNote Trust Management System. Proc of 4th International Conference on Information and Communications Security, Singapore, December 2002, LNCS 2513, pp. 13-24

[72] Gangadharan M., Hwang K.: Intranet Security with Micro-Firewalls and Mobile Agents for Proactive Intrusion Response. Proc of International Conference on Computer Networks and Mobile Computing, (ICCNMC'01), Beijing, China, October. 2001, pp. 325 – 332

[73] Meredith L.M.: A Summary of the Autonomic Distributed Firewalls (ADF) Project. Proc of 3rd DARPA Information Survivability Conference and Exposition. Washington DC, USA, April 2003.

[74] Payne C., Markham T.: Architecture and Applications for a Distributed Embedded Firewall. Annual Computer Security Applications Conference (ACSAC'01), New Orleans, Louisiana, USA, December 2001.

[75] Munson J.C., Wimer S.: Watcher: The Missing Piece of Security Puzzle. Proc of 17th Annual Computer Security Applications Conference (ACSAC'01), New Orleans, Louisiana, USA, December, 2001.

[76] Provos N.: Improving Host Security with System Call Policies. Technical Report 02-3, Center for Information Technology Integration, University of Michigan, November 2002; http://www.citi.umich.edu/techreports/ (also, proc of 12th USENIX Security Symposium, Washington, DC, USA, August 2003; pp. 257-272)

[77] Ferguson P., Huston G.: What is a VPN? Revision 1, April 1998, http://www.employees.org/~ferguson/vpn.pdf

[78] Venkateswaran, R.: Virtual private networks. Potentials, Journal of IEEE, Vol.20, Issue 1, February/March 2001; pp.11-15

[79] Metz C.: The Latest in Virtual Private Networks: Part I. Journal of Internet Computing, IEEE, Vol.7, Issue 1, January/February 2003; pp. 87-91.

[80] Smith B.R., Agrawala P., Badea M., Johnson J., Vernaillen T., Walsh A.: iSeries IP Networks: Dynamic! IBM Redbooks, SG24-6718-01, September 2003, www.redbooks.ibm.com

[81] Rigney C., Livingston S.W., Merit A.R., Simpson W.: Remote Authentication Dial In User Service (RADIUS). RFC: 2865, Category: Standards Track, IETF, June 2000.

[82] Ansper A., Buldas A., Freudenthal M., Willemson J.: Scalable and eficient PKI for inter-organizational communication. Proc of 19th Annual Computer Security Applications Conference (ACSAC 2003), December 2003 Las Vegas, Nevada, USA; pp.308-318.

[83] Kent S., Atkinson R.: Security Architecture for the Internet Protocol. RFC 2401, Category: Standard Tracks, IETF, November 1998.

[84] Dierks T., Allen C.: The TLS Protocol - Version 1.0. RFC 2246, Category: Standard Tracks, IETF, January 1999.

[85] De Clercq J., Paridaens O.: Scalability implications of virtual private networks. Communications Magazine, IEEE, Vol.40, Issue 5, May 2002; pp.151-157.

[86] O'Guin S., Williams C.K., Selimis N.: Application of Virtual Private Networking Technology to Standards-Based Management Protocols Across Heterogeneous Firewall-Protected Networks. Proc of IEEE Military Communications Conference (MILCOM'99), Atlantic City, New Jersey, USA, November 1999, Vol.2, pp.1251-1255.

[87] Boutaba R.: On Managing Virtual Private Networks. In IEEE Canadian Review, No 39, winter 2002, pp.19-22.

[88] Gleeson B., Lin A., Heinanen J., Armitage G., Malis A.: A Framework for IP Based Virtual Private Networks. RFC 2764, Category: Informational, IETF, February 2000.

[89] Dondeti L.R., Mukherjee S., Samal A.: Survey and Comparison of Secure Group Communication Protocols. Technical Report, University of Nebraska-Lincoln, June 1999; http://citeseer.ist.psu.edu/dondeti99survey.html

[90] Rafaeli S., Hutchison D.: A survey of key management for secure group communication. ACM Computing Surveys (CSUR), Vol.35, Issue 3, September 2003; pp. 309 – 329

[91] Wallner D., Harder E., Agee R.: Key Management for Multicast: Issues and Architectures. RFC 2627, Category: Informational, IETF, June 1999.

[92] Mittra S.: Iolus: A Framework for Scalable Secure Multicast. Proc of ACM SIGCOMM'97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication; Cannes, France, September 1997; pp. 277-288

[93] Canetti R., Pikas B.: A Taxonomy of Multicast Security Issues. Internet Draft, IETF, November 1998; http://www.wisdom.weizmann.ac.il/~bennyp/draft-canetti-secure-multicast-taxonomy-01.txt

[94] Ballardie T., Crowcroft J.: Multicast-Specific Security Threats and Counter-Measures. Proc of Symposium on Network and Distributed System Security (SNDSS'95), San Diego, USA, February 1995

[95] Agarwal D.A., Chevassut O., Thompson M.R., Tsudik G.: An Integrated Solution for Secure Group Communication in Wide-Area Networks. Proc of 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia, July 2001

[96] Ateniese G., Steiner M., Tsudik G.: New Multiparty Authentication Services and Key Agreement Protocol. IEEE Journal on Selected Areas in Communications, Vol.18, No.4, April 2000; pp. 628-639

[97] Balenson D.M., Dinsmore P.T., Heyman M., Kruss P.S., Scace C., Sherman A.T.: Dynamic Cryptographic Context Management (DCCM). Final Report, NAI Report #0776, NAI Labs, Security Research Division of Network Associates, Inc., Glenwood, USA, April 2000

[98] McDaniel P., Prakash A., Honeyman P.: Antigone: A Flexible Framework for Secure Group Communication. Proc of 8th USENIX Security Symposium, Washington DC, USA, August 1999; http://www.usenix.org/publications/library/proceedings/sec99/mcdaniel.html

[99] Weis B.: The Use of RSA Signatures within ESP and AH. Internet Draft, IETF, December 2003 (expires: June 2004); http://www.ietf.org/internet-drafts/draft-ietf-msec-ipsec-signatures-00.txt

[100] Hardjono T., Weis B.: The Multicast Group Security Architecture. RFC 3740, Category: Informational, IETF, March 2004

[101] Napster; http://www.napster.com

[102] O'Reilley P2P Directory Listings; http://www.openp2p.com/pub/q/p2p_category

[103] Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. Proc of 1st IEEE International Conference on Peer-to-Peer Computing (P2P 2001), Linköping, Sweden, August 2001, pp. 101-102.

[104] Oram A. (editor): Peer-to-Peer: Harnessing the Power of Disruptive Technologies. Published by O'Reilly & Associates, Inc., March 2001, ISBN 059600110X

[105] Seti@Home, http://setiathome.ssl.berkeley.edu/

[106] Gnutella, http://gnutella.wego.com ; also, http://www.gnutella.com/

[107] Freenet, http://freenet.sourceforge.net

[108] Saroiu S., Gummadi P.K., Gribble S.D.: A Measurement Study of Peer-to-Peer File Sharing Systems. Technical Report W-CSE-01-06-02, Department of Computer Science & Engineering, University of Washington, Seattle, June 2002.

[109] FastTrack, http://developer.berlios.de/projects/gift-fasttrack/

[110] Peter S.A.: Jabber Technology Overview. 2001, http://docs.jabber.org/general/html/overview.html

[111] Maymounkov P., Mazieres D.: Kademlia: A Peer-to-peer Information System Based on the XOR Metric. Proc of 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), MIT Faculty Club, Cambridge, MA, USA, March 2002; http://www.cs.rice.edu/Conferences/IPTPS02/

[112] Zhao B.Y., Huang L., Stribling J., Rhea S.C., Joseph A.D., Kubiatowicz J.D.: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications, Vol.22, No.1, January 2004; pp. 41-53.

[113] Barkai D.: Technologies for sharing and collaborating on the Net. Proc of 1st International Conference on Peer-to-Peer Computing (P2P 2001), Linköping, Sweden, August 2001; pp.13-28.

[114] Groove; http://www.groove.net/

[115] Wierzbicki A., Strzelecki R., Swierezewski D., Znojek M.: Rhubarb: a tool for developing scalable and secure peer-to-peer applications. Proc of 2nd International Conference on Peer-to-Peer Computing, 2002. (P2P 2002), Linköping, Sweden, September 2002; pp.144-151.

[116] Delco M.R., Ionescu M.D.: Secure Peer-to-Peer File Sharing within Dynamic Groups. Technical Report, Computer Science Division, University of California, Berkeley, 2001; available at: http://www.comp.nus.edu.sg/~zhanghan/paper/berkley-dg.pdf

[117] Fenkam P., Dustdar S., Kirda E., Reif G., Gall H.: Towards an access control system for mobile peer-to-peer collaborative environments. Proc of 11th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2002), Pittsburgh, Pennsylvania, USA, June 2002; pp.95-100.

[118] Gong L.: JXTA: a network programming environment. Internet Computing, Journal of IEEE, Vol.5, Issue 3, May-June 2001; pp.88-95.

[119] Yeager W., Williams J.: Secure peer-to-peer networking: the JXTA example. IT Professional, Journal of IEEE, Vol.4, Issue 2, March-April 2002; pp.53-57.

[120] Lawton G.: Technology news - Is peer-to-peer secure enough for corporate use. Computer, Journal of IEEE, Vol. 37, Issue 1, January 2004, pp.22 – 25.

[121] Chien E.: Malicious Threats of Peer-to-Peer Networking. Symantec Security Response Whitepaper, December 2001; http://securityresponse.symantec.com/avcenter/reference/malicious.threats.pdf

[122] Foster I., Kesselman C., Tuecke S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Published in International Journal of Supercomputer Applications, 2001; http://www.globus.org/research/papers/anatomy.pdf

[123] Castro-Leon E.: The Web Within the Web. Spectrum, Journal of IEEE, Vol.41, Issue 2, February 2004; pp.36-40.

[124] Tuecke S.: The Future of Grid and Web Services. Technical Tutorial, Imperial College, London, UK, January 2004 (org. by National e-Science Centre UK); http://www.nesc.ac.uk/esi/events/385/

[125] Welch V., Siebenlist F., Foster I., Bresnahan J., Czajkowski K., Gawor J., Kesselman C., Meder, S., Pearlman L., Tuecke S.: Security for Grid services. Proc of 12th IEEE International Symposium on High Performance Distributed Computing, Seattle, Washington, USA, June 2003; pp.48-57.

[126] Globus Project home page, http://www.globus.org

[127] Frohner A. et al: DataGrid Security Design Report. EU Deliverable, March 2003; http://edms.cern.ch/document/344562

[128] DataGrid Project homepage, http://www.edg.org

[129] DataTAG Project home page, http://www.datatag.org

[130] Pearlman L., Kesselman C., Welch V., Foster I., Tuecke S.: The Community Authorization Service: Status and Future. Conference for Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, California, USA, March 2003.

[131] Alfieri R. et al: Managing Dynamic User Communities in a Grid Autonomous Resources. Proc of Conference for Computing in High Energy and Nuclear Physics (CHEP03), La Jolla, California, USA, March 2003.

[132] IBM/Microsoft: Security in a Web Services World: A Proposed Architecture and Roadmap. Version 1.0 (A joint security whitepaper from IBM Corporation and Microsoft Corporation), April 2002; http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/

[133] Ericsson Technical Manual: “AXE 10 Operation & Maintenance”, Ericsson Systems, 1998.

[134] Ericsson Review: “Understanding Telecommunications”, September 2003, http://www.ericsson.com/support/telecom/

[135] Santesson S., Polk W., Barzin P., Nystrom M.: “Internet X.509 Public Key Infrastructure Qualified Certificates Profile”, RFC 3039, Category: Standards Track, IETF, January 2001.

[136] Housley R., Ford W., Polk W., Solo D.: “Internet X.509 Public Key Infrastructure Certificate and CRL Profile”, RFC 2459, Category: Standards Track, IETF January 1999.

[137] Mil3 Home Page, OPNET. http://www.mil3.com

[138] OPNET Online Documentation - General Models: Process Model/Queuing, OPNET Modeler 8.1, MIL3, March 2002.

[139] Bennet K.: Testing of Pentium4 at 2GHz. http://www.hardocp.com/article.html?art=Nywx

[140] Intel Pentium 4 Processor, http://www.intel.com/products/desktop/processors/pentium4/index.htm

[141] Loosley C., Douglas F.: High-Performance Client/Server: A Guide for Building and Managing Robust Distributed Systems. John Wiley & Sons, Inc. 1998. ISBN 0471162698

[142] Mijatovic V., Djordjevic I.: IPSec – The Analysis of Cryptographic Algorithms. Proc of 7th Annual IEEE International Telecommunications Forum (TELFOR’99), Belgrade, Yugoslavia, November 1999. (not available in English)

[143] Dai W.: Speed Comparison of Popular Cryptographic Algorithms – Benchmark Testing; http://www.eskimo.com/~weidai/benchmarks.html

[144] Wiener M.J.: Performance Comparison of Public-Key Cryptosystems. CryptoBytes, The technical newsletter of RSA Laboratories, Vol4, No1, Summer 1998.

[145] Tierney B.L.: TCP Tuning Guide for Distributed Applications on Wide Area Networks. Usenix ;login:, February 2001. http://www-didc.lbl.gov/tcp-wan-perf.pdf

[146] Fairhurst G., Wood L.: Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366, Category: Best Current Practice, IETF, August 2002.

[147] Paxson V., Allman M.: Computing TCP's Retransmission Timer. RFC 2988, Category: Standards Track, IETF, November 2000.

[148] Nogueira D., et al: A Methodology for Workload Characterization of File-Sharing Peer-to-Peer Networks. 5th IEEE Workshop on Workload Characterization (WWC-5), Austin, Texas, USA, 2002. pp 118-126, also: http://www.dcc.ufmg.br/~diego/index_eng.html

[149] Sen S., Wang J.: Analysing Peer-to-Peer Traffic Across Large Networks. Proc of 2nd Internet Measurement Workshop, Marseille, France, 6-8 November 2002. http://www.icir.org/vern/imw-2002/ . Also, to appear in ACM/IEEE Transactions on Networking (ToN), 2004. http://www.research.att.com/~jiawang/my_research.html#p2p

[150] Jabber Software Foundation: Jabber Peer-to-Peer System Survey, March 2003, available at: www.jabber.org/jsf/surveys.php?PHPSESSID=a6686b59b234649b796a421cd685aded

[151] Pawlikowski K., Jeong H.D.J., Lee J.S.R.: On Credibility of Simulation Studies of Telecommunication Networks. IEEE Communications Magazine, January 2002. pp 132-139

[152] Walpole R.E., Myers R.H., Myers S.L., Ye K.: Probability and Statistics for Engineers and Scientists, 7th Edition. Prentice Hall 2002. ISBN 0-13-098469-8

[153] Law A.M., Kelton D.: Simulation Modelling and Analysis. 3rd Edition, McGraw-Hill Higher Education, 1991. ISBN 0-07-059292-6

[154] Holness F.M.: Congestion Control Mechanism within MPLS Networks. PhD Thesis, University of London, September 2000

[155] Bodanese E.: A Distributed Channel Allocation Scheme for Cellular Networks using Intelligent Software Agents. PhD Thesis, University of London, November 2000

[156] Mersenne-Twister Home Page. http://www.math.keio.ac.jp/~matumoto/emt.html

[157] Newman D.: Core Competency - ISP Backbone Test. Network World Fusion Newsletter, 16th December 2002. http://www.nwfusion.com/research/2002/1216isptest.html

[158] GRASP Project Home Page; http://www.eu-grasp.net

[159] SOAP Specification version 1.2, W3C Recommendation, June 2003; http://www.w3.org/TR/soap/

[160] Johnson D., Perkins C.: Mobility Support in IPv6. Internet Draft, Mobile IP Working Group, IETF, June 2003 (expires December 2003); draft-ietf-mobileip-ipv6-24.txt

[161] Lehtonen S., Ahola K., Koskinen T., Lyijynen M., Pesola J.: Roaming Active Filtering Firewall. Proc of Smart Objects Conference (SOC'03), May 2003, Grenoble, France; http://www.grenoble-soc.com

Appendix: Author's Publications

[DJO1] Djordjevic I., Phillips C.: Certificate-based Distributed Firewalls for Secure e-Commerce Transactions. 40th Annual FITCE Congress, August 2001, Barcelona, Spain; published in Journal of the Institution of British Telecommunication Engineers, Vol. 2, part 3, pp. 14-19.

[DJO2] Djordjevic I., Scharf E., Raptis D., Gran B.A.: Suitability of Risk Analysis Methods for Security Assessment of Large Scale Distributed Computer Systems. Proc of 6th International Conference on Probabilistic Safety Assessment and Management (PSAM 6), San Juan, Puerto Rico, USA, June 2002; published by Elsevier Science Ltd, July 2002, ISBN 0-0804-4120-3

[DJO3] Djordjevic I., Dimitrakos T.: Towards Dynamic Security Perimeters for Virtual Collaborative Networks. 2nd iTrust Workshop, Imperial College, University of London, September 2003, London, UK, http://www-dse.doc.ic.ac.uk/Events/itrust/

[DIM] Dimitrakos T., Djordjevic I., Milosevic Z., Jøsang A., Phillips C.I.: Contract Performance Assessment for Secure and Dynamic Virtual Collaborations. Proc of 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC'03), September 2003, Brisbane, Australia

[DJO4] Djordjevic I., Phillips C.: Architecture for Secure Work of Dynamic Distributed Groups. Proc of 1st IEEE Consumer Communication and Networking Conference (CCNC'04), January 2004, Las Vegas, Nevada, USA

[DJO5] Djordjevic I., Phillips C., Dimitrakos T.: An Architecture for Dynamic Security Perimeters of Virtual Collaborative Networks. Proc of 9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), April 2004, Seoul, Korea

[BIG] Bigham J., Jin X., Gamez D., Djordjevic I., Phillips C.: Dynamic Trust Management of Semi-Automated Complex Systems. IIIS International Conference on Computing, Communications and Control Technologies (CCCT'04), August 2004, Austin, Texas, USA